Analysis of Uncertainty in Different Neural Network Structures using Monte Carlo Dropout

by

Md. Farhadul Islam
22341042
Sarah Zabeen
19241004
Fardin Bin Rahman
20101592
Md. Azharul Islam
19301257
Fahmid Bin Kibria
19201063

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering
or B.Sc. in Computer Science

Department of Computer Science and Engineering
School of Data and Sciences
Brac University
January 2023

© 2023. Brac University
All rights reserved.

Declaration

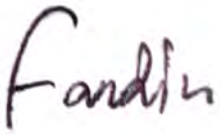
It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Md. Farhadul Islam
22341042

Sarah Zabeen
19241004



Fardin Bin Rahman
20101592

Md. Azharul Islam
19301257

Fahmid Bin Kibria
19201063

Approval

The thesis titled "Analysis of Uncertainty in Different Neural Network Structures using Monte Carlo Dropout" submitted by

1. Md. Farhadul Islam (22341042)
2. Sarah Zabeen (19241004)
3. Fardin Bin Rahman (20101592)
4. Md. Azharul Islam (19301257)
5. Fahmid Bin Kibria (19201063)

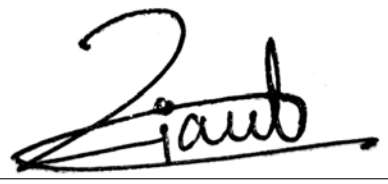
of Fall, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering or B.Sc. in Computer Science on January 19, 2023.

Examining Committee:

Supervisor:
(Member)

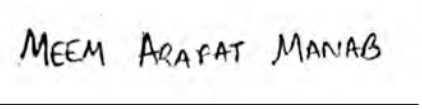
Annajiat Alim Rasel
Senior Lecturer
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)



Dewan Ziaul Karim
Lecturer
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)



Meem Arafat Manab
Lecturer
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi
Associate Professor and Chairperson
Department of Computer Science and Engineering
Brac University

Ethics Statement

This note is a disclaimer to indicate that the research paper in question has not violated the preservation of human rights, welfare or dignity. The honesty and integrity of the research participants are incorporated in the work, which also includes highlighted citations from various reliable sources. No discrimination or disparity has been made in the data collection, and the results have not been manipulated for any prejudiced reasons. Rest assured, we hope our work reflects our respect for all intellectual property and is useful in the long term welfare of humanity.

List of Publications

- **Islam, M.F., Zabeen, S.**, Mehedi, M.H.K., Iqbal, S., Rasel, A.A. (2022). Monte Carlo Dropout for Uncertainty Analysis and ECG Trace Image Classification. In: Krzyzak, A., Suen, C.Y., Torsello, A., Nobile, N. (eds) Structural, Syntactic, and Statistical Pattern Recognition. S+SSPR 2022. Lecture Notes in Computer Science, vol 13813. Springer, Cham. https://doi.org/10.1007/978-3-031-23028-8_18

Other Submissions:

- **How Certain are Transformers in Image Classification: Uncertainty Analysis with Monte Carlo Dropout** (Accepted for Publication)

  Authors: **Islam, M.F., Zabeen, S., Islam, M.A., Rahman, F.B.**, Ahmed, A., Karim, D.Z., Rasel, A.A., and Manab, M.A.

  Fifteenth International Conference on Machine Vision (ICMV 2022), Rome Italy.

- **RNN Variants vs Transformer Variants: Uncertainty in Text Classification with Monte Carlo Dropout** (Accepted for Publication)

  Authors: **Islam, M.F., Rahman, F.B., Zabeen, S., Islam, M.A.**, Hossain, M.S., Mehedi, M.H.K., Manab, M.A, Rasel, A.A.

  25^{th} International Conference on Computer and Information Technology (ICCIT), Cox's Bazar, Bangladesh.

- **Exploring Node Classification Uncertainty in Graph Neural Networks** (Under Review)

  Authors: **Islam, M.F., Zabeen, S., Rahman, F.B., Islam, M.A., Kibria, F.B.**, Manab, M.A., Karim, D.Z., and Rasel, A.A.

- **UnIC-Net: Uncertainty Aware Involution-Convolution Hybrid Network for Two-level Disease Identification** (Under Review)

  Authors: **Islam, M.F., Zabeen, S., Rahman, F.B., Islam, M.A., Kibria, F.B.**, Manab, M.A., Karim, D.Z., and Rasel, A.A.

Abstract

Deep learning technologies developed at an exponential rate throughout the years. Starting from Convolutional Neural Networks (CNNs) to Involutional Neural Networks (INNs), there are several neural network (NN) architectures today, including Vision Transformers (ViT), Graph Neural Networks (GNNs), Recurrent Neural Networks (RNNs) etc. However, uncertainty cannot be represented in these architectures, which poses a significant difficulty for decision-making given that capturing the uncertainties of these state-of-the-art NN structures would aid in making specific judgments. Dropout is one method that may be implemented within Deep Learning (DL) networks as a technique to assess uncertainty. Dropout is applied at the inference phase to measure the uncertainty of these neural network models. This approach, commonly known as Monte Carlo Dropout (MCD), works well as a low-complexity estimation to compute uncertainty. MCD is a widely used approach to measure uncertainty in DL models, but majority of the earlier works focus on only a particular application. Furthermore, there are many state-of-the-art (SOTA) NNs that remain unexplored, with regards to that of uncertainty evaluation. Therefore an up-to-date roadmap and benchmark is required in this field of study. Our study revolved around a comprehensive analysis of the MCD approach for assessing model uncertainty in neural network models with a variety of datasets. Besides, we include SOTA NNs to explore the untouched models regarding uncertainty. In addition, we demonstrate how the model may perform better with less uncertainty by modifying NN topologies, which also reveals the causes of a model's uncertainty. Using the results of our experiments and subsequent enhancements, we also discuss the various advantages and costs of using MCD in these NN designs. While working with reliable and robust models we propose two novel architectures, which provide outstanding performances in medical image diagnosis.

Keywords: Deep Learning · Neural Network · Monte Carlo Dropout · Uncertainty

Dedication

We dedicate this thesis to students who are uncertain, confused and indecisive in their life which hinders their growth and development. We wish students do not become too uncertain like some specific deep learning models, which may lead to risky situations. We hope, their "Learning Process" does not get hindered by any source of "noise" or obstacles, and that their capabilities are not in question since we believe every student is modeled with enough potentiality to make the right decisions for themselves

Acknowledgement

We owe all we have accomplished so far to the Almighty Allah, who gave us the means, and the focus we needed to finish our study on schedule. We owe a great deal of gratitude to our supervisor Annajiat Alim Rasel and to our co-supervisors, Dewan Ziaul Karim and Meem Arafat Manab, for their valuable assistance and support. We would also want to extend our gratitude to our co-authors in the publications relevant to our thesis who helped us improve the quality of our research. Finally, we would want to thank our caring parents for always being there for us and praying for us.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

AE Autoencoder

AI Artificial Intelligence

ANN Artificial Neural Network

APPNP Approximation of personalized propagation of neural predictions

BERT Bidirectional Encoder Representations from Transformers

BNN Bayesian Neural Network

CCT Compact Convolutinal Transformers

CNN Convolutional Neural Network

DL Deep Learning

DNN Deep Neural Network

FFN Feed-Forward Network

FN False Negative

FP False Positive

GAT Graph Attention Network

GCN Graph Convolutional Network

GNN Graph Neural Network

GRU Gated Recurrent Unit

GS GraphSAGE

HI Health Indicator

HMC Hamiltonian Monte Carlo

IGBT Insulated Gate Bipolar Transistor

INN Involutional Neural Network

LSTM Long Short-Term Memory

MCD Monte Carlo Dropout

MCMC Markov Chain Monte Carlo

MLP Multilayer Perceptron

ML Machine Learning

MSE Mean Squared Error

NLP Natural Language Processing

NN Neural Network

PPNP Personalized propagation of neural predictions

ReLU Rectified Linear Unit

RMSE Root Mean Squared Error

RNN Recurrent Neural Network

ROC Receiver Operating Characteristics

RUL Remaining Useful Life

SD Standard Deviation

SGD Stochastic Gradient Descent

SGHMC Stochastic Gradient Hamiltonian Monte Carlo

SGLD Stochastic Gradient Langevin Dynamics

SNR Signal-to-Noise Ratio

SOTA State-of-the-Art

SWAG Stochastic Weight Averaging Gaussian

SWT Swin Transformer

TN True Negative

TP True Positive

UQ Uncertainty Quantification

VAE Variational Autoencoder

ViT Vision Transformer

XLNet Generalized Auto-Regressive model

Chapter 1

Introduction

1.1 Problem Statement

The British statistician George E. P. Box. is attributed a rather famous aphorism, "all models are wrong", which is typically extended to "all models are wrong, but some are useful". This proverb recognizes that statistical and empirical models can be beneficial in spite of their inevitable limitations due to the complexity of the actual world. In deep learning, our predicted results are considered non-deterministic, making the model uncertain. This property of deep learning is said to be stochastic. However, a model's uncertainty is a major key factor in its decision making capability. In medical and financial fields especially, uncertain predictions may lead to complexity issues. Thus, the reliability of a deep neural network model's results need to be validated using uncertainty approximation [1], [2].
In the field of Bayesian machine learning, we frequently deal with uncertain outcomes and probabilistic models. In order to understand the most likely and least likely ways to infer from observable data, models like Gaussian processes are utilized. These models, which describe distributions of probability over functions, are applied to quantify the probability of each possible outcome. This stochastic view of machine learning provides a new boundary of confidence for tasks such as data analysis and automated decision making. This knowledge could be put to use by a biologist to examine her data, or by an autopilot system to determine whether or not it should halt. When it comes to being able to make decisions or evaluate data, a model is frequently required to have the ablility to figure out whether its output is definitive, that is to be able to pose the question, "Do I need to use a more diverse dataset? or modify the model? or possibly exercise caution while making decisions?". Such concerns are crucial to Bayesian machine learning, where they have been the subject of substantial research [3]. In contrast, when employing deep learning models [4], we typically just have baseline estimations of predictions and parameters. In order to employ such models, we have to give up the resources we rely on to answer the questions posed at the beginning of this article, which could lead to a scenario wherein we have no idea whether or not the model is producing reasonable predictions. Deterministic functions are often considered to be appropriate representations for the greater part of deep learning models. Consequently, they are regarded as working in an environment that is very dissimilar to that of probabilistic models, which contain uncertainty information. It could be for this reason that the closeness of current deep learning to probabilistic modeling comes

as such a surprise.
Understanding and defining the questions at hand is an essential first step in model-based prediction. As opposed to problems where the dataset is quite enriched, for which both the queries and accompanying frameworks are often unknown. Successful models that identify spontaneous patterns in datasets are sought for by a variety of AI-based technologies, from cutting-edge hardware to machine learning (ML) techniques. The complexity, multimodality, inconsistency, noise, and incompleteness of these data are all too common. Numerous decision-making procedures, such as those involving nuclear safety, risk assessment, and accompanying expenditures, can indeed be supported by the quantification of uncertainty nowadays, and these procedures may usually involve managing billions of dollars. No amount of forecasting is useful if it lacks the ability to quantify uncertainty. Similar difficulties can be found in the data-heavy field of ML, particularly with the robust deep learning (DL) models. There is a critical obligation to co-develop uncertainty quantification for this area in order to create useful decision support from 'learned' models constructed on complicated datasets. The future of prediction resides in the intersection of model-based and data-based approaches. Uncertainty quantification for AI-based techniques is a prerequisite for achieving this goal.
There are profound social and psychological ramifications in outsourcing to computers for decisions that influence people's life, and hence the basic concepts of the DL models vary greatly. Nevertheless, in order to render DL and other data-driven methods and techniques practically applicable, advancements in the comprehension of the architecture of these predicting models, the integration of model and data-driven methodologies with firmly justifiable uncertainty quantification, as well as the establishment of the uncertainty quantitative analysis for the DL discipline will be required.
Uncertainty in deep learning is important because it allows models to quantify their confidence in predictions, which may be especially useful in safety-critical applications where a high level of credence is needed before taking action. In addition, deep learning models incorporated with uncertainty quantification can also help in enhancing the its own robustness and performance. The model's effectiveness can be further improved by modeling uncertainty, which in turn increases the model's ability to make well-informed choices. For example, a model that is uncertain about its predictions may choose not to make a prediction, or it may choose to make a conservative prediction that is less likely to be incorrect [5].
Furthermore, modeling uncertainty aids in comprehending the judging and deciding process of the model, which is something that can be helpful for both debugging and interpreting the model. For example, if a model is uncertain about its predictions, it may be possible to identify the features of the input that are causing the uncertainty and to use this information to improve the model [6].
Without the quantification of uncertainty to substantiate the deep learning models' decisions, it would make faulty predictions [7], [8].
Engineering breakthroughs in machine learning sector led to the application of previously experimental systems with practical data. One such context would be the transfer of decision-making authority to an automated process under potentially fatal risks. Control of critical systems, medical decision-making or recommendation systems, drone and autonomous vehicle operation, high-frequency trading's potential to disrupt the global economy and so on are examples of such processes. All of

these mentioned scenarios pertain to the wider scope of AI security.

1.2 Background Information

1.2.1 What is Uncertainty in Deep Learning

Uncertainty, as it relates to deep learning, is the degree to which a deep learning model exhibits either confidence or ambiguity in the predictions or decisions that it makes. It is a measurement of how effectively the model generalizes to data that has not been seen before and measures the amount of randomness or lack of information that is present in the model's output. This may be measured in a variety of ways, including probability distributions and measurements of entropy, amongst other possible approaches.
Projected probabilities seem acceptable as an indicator of model uncertainty in terms of categorization, as there is a great deal of logic behind this. Consider the case where we are training our model for a categorization task for handwritten digit identification, however we only ever provide the values 0 through 8 as training examples, however the value 9 is not used. Next, we presume that this model would produce reliable results for 0-8 data samples.
Consider the following scenarios: we see the number 2 (Figure 1.1) on the input image, and sure enough, our network is able to make a strong prediction.

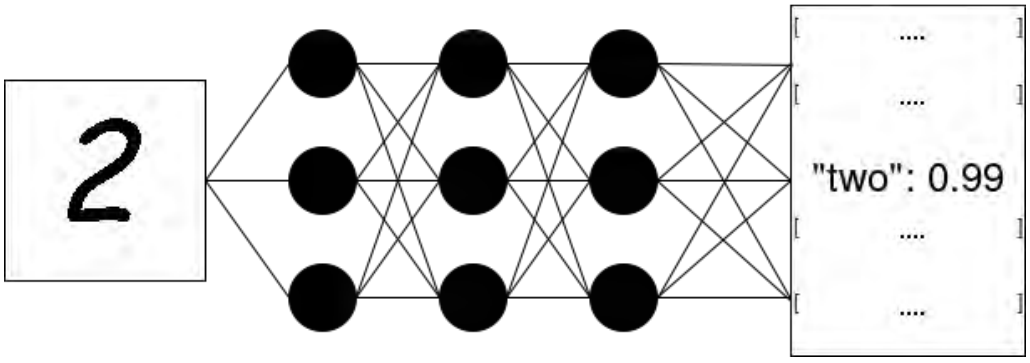


Figure 1.1: Example of Predicting the Digit 2

Example 7 (Figure 1.2) is more challenging since assigning the correct class to the sample (no matter it be class 7 or 1) is more complicated. This serves as a perfect demonstration of the aleatoric uncertainty that will be discussed in greater detail later on.
Moving forward, we start the network with a sample of data from the class "9" (Figure 1.3). Since the model has never encountered this kind of data before, the results it produces are likely to be highly inaccurate. Although there may be good reason to believe that the predicted distribution will be flattened and high in entropy, our model's output can be totally unpredictable.
It is almost a trait of neural networks to provide overly confident forecasts [9], and as a consequence of this, it is conceivable to make a confident projection with highest probability in the value "4" even when the scenario involves a sample model of "9". As a result, the findings of the model cannot be used as an indicator for uncertainty.

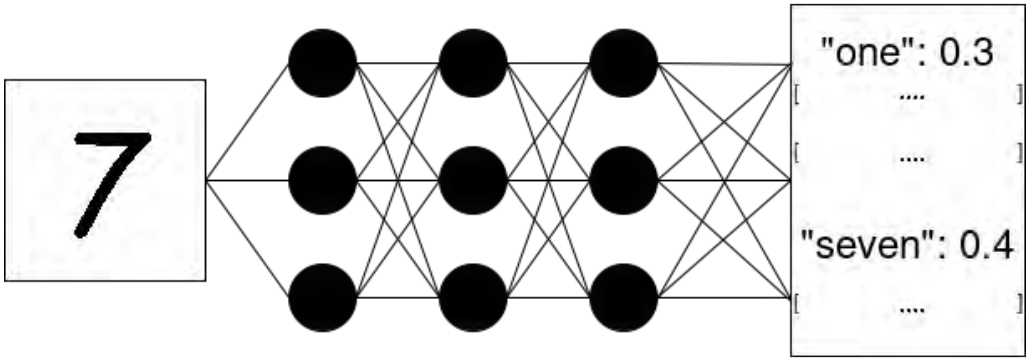


Figure 1.2: Example of Predicting the Digit 7

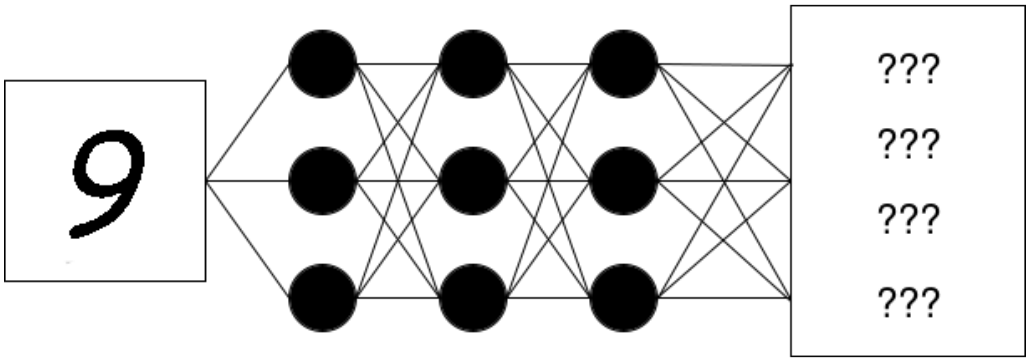


Figure 1.3: Example of Predicting the Digit 9

1.2.2 What Causes Uncertainty in Deep Learning

Gawlikowski et al. [10] discuss the elements that influence the occurrence of uncertainty. We shall address the causes of uncertainty from the next paragraph.

Factor I: Variability in Real World Situations

When it comes to the actual world, the entirety of our surroundings are notoriously unpredictable and subject to being impacted by frequent changes. Attributes like congestion, light level, temperature and the shape and size of physical items are all influenced by these modifications. Environmental factors can also have an impact on how things appear; for instance, the appearance of plants can change dramatically from after rainfall to after droughts. A distribution shift occurs when out-of-sample data (unexpected external conditions) does not fit the same pattern as the training data. When there is a change in the distribution over which the network is trained, the results can be dramatically different.

Factor II: Error and Noise in Measurement Systems

In some cases, the reliability of the neural network's prediction can be affected by the reliability of the observations themselves. It may be the result of inadequate data being included in the observations, such as the picture quality, or the failure to account for false or minimally available data modalities. Another probable ex-

planation is that the image resolution was too low. In addition, this could result from mechanical tension, motion, or sensor noise, all of which contribute to erroneous measurements. Furthermore, improper labeling contributes to the overall level of measurement uncertainty through noise and error. Label noise is a causative factor in machine learning models; it causes a loss of credibility in the model's accurate classifications when it is being trained.

Factor III: Errors in the Model Structure

The accuracy, precision as well as the level of uncertainty of a neural network's predictions are directly tied to the its architecture. Over-fitting and under-fitting on the training dataset might occur, for example, when there are too many features limiting the memory space. In terms of neural network uncertainty, it has been proven that deeper networks are more likely to exhibit overconfidence in their softmax output [15], which implies that they are all the more likely to predict with excessively greater probability the category with the greatest probability score [9].

Factor IV: Errors in the Training Procedure

During the train phase of a neural net, there are a great deal of parameters that need to be provided (stopping criteria, learning rate, optimizer, regularization, batch size, etc.). There are also stochastic actions taken during the learning phase itself (in terms of both batch size and initialization of weights). Because of how the local optima are affected by all these changes, it is highly implausible that two training cycles will result in the exact same model parameterization. Uncertainties on the network's learning variables are also introduced by a training dataset with lack of balance or poor representation of particular areas in the data distribution, as has already been previously discussed in the data procurement process. This could be mitigated through the use of augmentation to boost diversity, or through the modification of the loss function so that the influence of individual groups or areas is minimized.

Factor V: Errors Caused by Unknown Data

The ability of a neural network learned using samples from a particular environment W1 to handle data from another environment W2 has been demonstrated, in particular for classification tasks. Such is the situation, for instance, when a model which is trained on pictures of dogs and cats is presented with a picture of a bird. Here, we presume an environment has only realistic input sources for a forecasting job, thus we believe that the uncertainty does not stem from the data collection procedure. In spite of the fact that the actual outcome could be equivalent to an excessive amount of noise on the sensor or the full malfunction of the sensors, the data that we have reviewed here constitute a legitimate sample, albeit for another endeavor or domain.

1.2.3 Types of Uncertainty

Data uncertainty emerges due to coinciding class or noisy data [11], whereas knowledge uncertainty arises due to misalignment of training and testing data [12]. However, measuring uncertainty based on knowledge seems more challenging. Aleatoric and epistemic uncertainty are two prior erratic forms [13].
There are two distinct types of uncertainty: epistemic uncertainty and aleatoric uncertainty. Epistemic uncertainty is often referred to as model uncertainty. Meanwhile, aleatoric uncertainty is sometimes called data uncertainty.These ambiguities are often combined into a single number and forecasted as a whole, which is referred to as the predictive uncertainty [12]. Recovering the two different aspects of uncertainty might be useful in a few different situations.

Aleatoric Uncertainty

The term "aleatoric" (sometimes known as "statistical") uncertainty is used to describe the inherent unpredictability in every experiment's results. This suggests that our observed labels may contain noise, maybe as a result of inaccuracies in our measurements. Utilized data is the source of this form of uncertainty.

Epistemic Uncertainty

Uncertainty due to a lack of information, or the agent's epistemic condition, is referred to as epistemic (or systematic) uncertainty. There are two components to this kind of uncertainty: uncertainty from within the model parameters and uncertainty in the structure. In the first situation, we may not know which model parameters to use for prediction since several models may be able to describe a particular dataset. Is there a preferred model structure? is another way of putting it. How can we properly parameterize our model to extrapolate and interpolate data?

Predictive Uncertainty

Predictive uncertainty, or the degree to which we trust a given forecast, can be cultivated by introducing elements of chance (or "aleatoric uncertainty") and of knowledge-based (or "epistemic") doubt. Figure 1.4 represents the types of uncertainty how aleatoric and epistemic uncertainty build up predictive uncertatinty.
Uncertainty in forecasts can be attributed to aleatory factors, which are the data's inherent imprecision (also known as uncertainty in data). Due to the fact that it is not a property of the model, this type of uncertainty cannot be mitigated. Conversely, lack of information, results in epistemic doubt (otherwise called knowledge uncertainty). Models may be defined in model-based prediction to handle a wide range of problems. The data sets for information-rich problems may be quite large yet lacking in detail [14]. In certain cases, it may be possible to use AI-based techniques to develop effective prototypes that define input's potential aspects. Sometimes the information we have is sketchy, loud, contradictory, or spans many categories [15]. Several important decisions nowadays rely on uncertainty quantification (UQ), and estimates supplied without UQ are often inaccurate. Knowing the function of UQ in deep learning (DL) is essential for understanding its stages [16], [17]. Deep learning (DL) models are built on top of extensive and pertinent data samples that may be

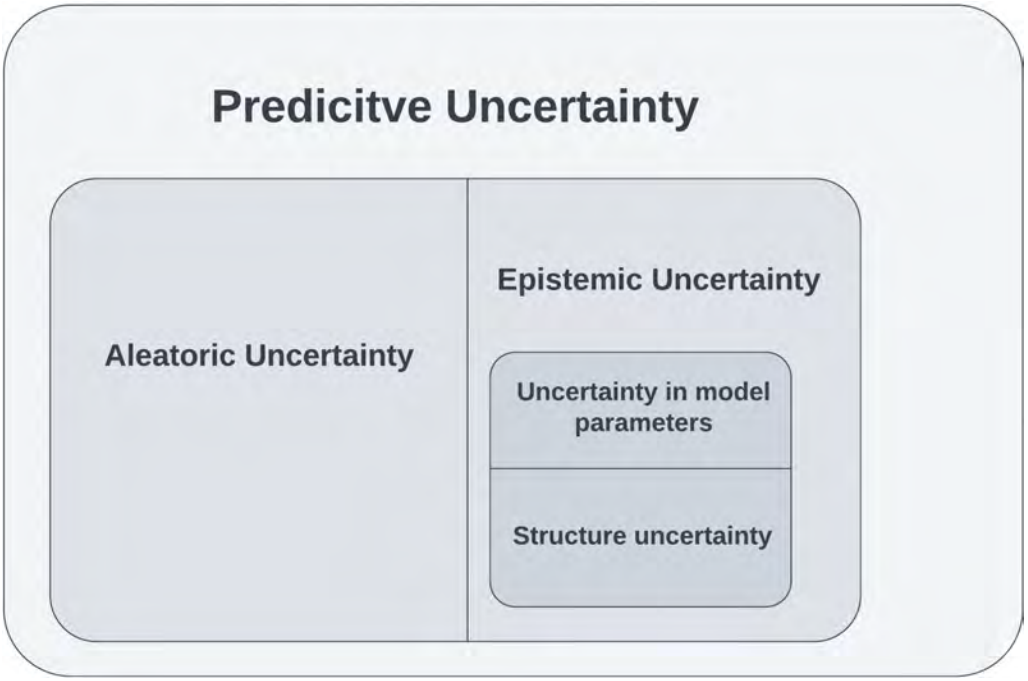


Figure 1.4: Predictive Uncertainty and sub-parts

accessed during the decision-making process. Then, prototype training is initiated with input data once the DL scenario has been constructed to accomplish certain performance goals using the best DL architecture. By iteratively refining the training process, the network's learning parameters may be fine-tuned to optimal levels of performance.

Bayesian techniques such as Bayesian neural networks (BNNs) and Gaussian processes are popularly used in cases where uncertainty measurements are classified along with predictions. These architectures create predictive distributions wherein the BNN weights are added to the prior distribution [18] and Gaussian processes include priors over functions [19]. Maddox et al. [20] propose a method to determine uncertainty using stochastic weight averaging. These methods are effective theoretically, but tough to implement in practice. Gal and Ghahramani [21] have proposed feasible and uncomplicated method for representing model uncertainty. They have noticed that unpredictable nature in model can be determined with utilization of dropouts, which also decrease model complexity and reduce overfitting [22]. It is a technique that generates an ensemble of predictions by conducting several stochastic forward passes within a neural network employing active dropout during the test phase. This approach is popularized as the Monte Carlo Dropout (MCD) [23]–[28]. The method is well-favored as it can calibrate well with huge datasets without altering the existing architectural structure of the model. In simple terms, MCD is a arithmetic mean of several networks' ensemble. [26], [29], [30].

In our study, we utilize this method to determine uncertainty in deep learning models. Our experiment highlights factors which are responsible for high uncertainty and depicts how to resolve the issue of high uncertainty in multiple fields of DL. We provide a full overview of MCD in NNs, and propose a reliable framework for ECG trace image classification.

The confidence score or softmax score is the resultant of the final layer of a deep neural network when trained with a categorical cross-entropy loss. This is not a

reliable measure of uncertainty for a few reasons:

- First, it is only a point estimate of the model's output, meaning that it only gives the most likely class label without providing any information about the uncertainty of the prediction.
- Second, the softmax function used to compute the confidence score is designed to produce outputs that are easy to interpret as probabilities, but these probabilities are not always well-calibrated, meaning that the confidence score may not accurately reflect the true likelihood of the predicted class label.

Monte Carlo dropout is a way of approximating Bayesian deep learning and it provides a more robust measure of uncertainty by averaging over multiple models. During training, dropout is applied to the activations of some neurons in the network, effectively training an ensemble of models with different architectures. During inference, dropout is applied to the activations of the same neurons and then numerous forward passes of the model are performed with different dropout masks. The output of these forward passes can be utilized to assess the uncertainty of the model's predictions by computing the variance of the predicted class probabilities. Therefore, Monte Carlo dropout is preferred over other methods because it provides a probabilistic measure of uncertainty that is better calibrated and more robust to overfitting.

1.3 Thesis Structure

The upcoming part of this thesis has been constructed as follows:

- In the first part of Chapter 2, an overview is given of the various approaches currently in use to assess uncertainty in DL models. After that, we talk about the work that has already been done using MCD, as well as the findings from MCD.
- In Chapter 3, we illustrate our research objectives and workflow.
- Chapter 4 includes the datasets that were utilized in the experiments of our thesis.
- Chapter 5 contains the detailed descriptions of our methodologies; such as, Monte Carlo Dropout, Neural Network Models. It also contains the descriptions of two proposed models. One of them was previously presented in [31].
- In Chapter 6, we provide the results of our experiments as well as the experimental process we utilize.
- In Chapter 7, we go through the main takeaways and findings that we made from our experiments.
- The extent of our study as well as the areas in which it may be expanded in the future are presented in Chapter 8. In addition to that, it digs into the topic of extending the model that we have proposed.
- Finally, Chapter 9 synopsizes the whole thesis and concludes our thesis.

Chapter 2

Literature Review

2.1 Uncertainty Quantification

There are several studies on representing uncertainty in deep learning models. Such as Bayesian Methods, SGD Based Approximations, Methods for Calibration of DNNs. Abdar et al.[32] presented a comprehensive survey of the uncertainty quantification approaches.

2.1.1 Bayesian Methods

In Bayesian model averaging, a distribution is placed atop model specifications, which are marginalized to build a full predictive distribution. The foundational studies of Neal [18] and MacKay [33] established Bayesian techniques to learn using NNs in the late 1990s. Modern NNs, in constrast, frequently include hundreds of thousands of parameters, and the posterior probability inference of these parameters is very non-convex, necessitating mini-batch techniques to get to a suitable solution space [34]. Bayesian techniques have been essentially intractable for current neural networks as a result of these factors.

2.1.2 Markov Chain Monte Carlo (MCMC)

MCMC was widely used for reasoning with other NNs evident in Hamiltonian Monte Carlo (HMC) discovery of Neal [18]. Yet, HMC requires full gradients that are analytically complex for utilization in advanced NNs. Chen et al. [35] introduced Stochastic Gradient HMC (SGHMC) where the usage of stochastic gradients is allowed in bayesian method for conclusion. For this reason, SGHMC is important for expandability and exploration for a solution with respect to high generalization. In the stochastic gradient situation, first order Langevin dynamics is employed in stochastic gradient Langevin dynamics (SGLD) [36]. SGHMC and SGLD asymptomatically sample from the posterior in case of indefinitely tiny sizes of increments. Changing stochastic gradient MCMC can be complicated and using limited learning models can cause approximation inaccuracy in reality.

2.1.3 Variational Inference

Graves [19] proposes that the weights of NNs be fitted with a Gaussian variational posterior approximation. The reparameterization strategy was suggested by Kingma and Welling [37] for training deep latent variable models, and numerous variational inference techniques using adjusted parameters were developed for deep NNs. While variational approaches function well on modestly sized networks, they are challenging to train on bigger designs like deep residual networks, according to other studies.

2.1.4 SWA-Gaussian for Bayesian Deep Learning

Uncertainty approximation can also be completed using a method known as SWA-Gaussian (SWAG) [20] . SWA stands for Stochastic Weight Averaging, and it iterates with an adjusted rate of learning routine and it is capable of calculating the moment of stochastic gradient descent (SGD). A Gaussian is then fitted in order to establish an approximation of the posterior probability distribution over neural network weights. This gaussian distribution is used to process the Bayesian model averaging. SWAG approximates the shape of the actual posterior estimate, according to their experiments.

2.1.5 SGD Based Approximations

Mandt et al.[38] introduced a new idea where the iterates of the average form of SGD as an MCMC archetype after using stochastic calculus method to study the dynamics of SGD. Using SGD, Chen et al. [39] figured out the problem of statistical conclusion of true model parameters when the population loss function is substantially convex and the certain conditions of smoothness are fulfilled.

2.1.6 Methods for Calibration of DNNs

The authors Lakshminarayanan et al. [29] suggest that in order to improve calibration of Deep Learning Models, it is best to use ensembles of many networks and an antagonistic loss function to describe uncertainty. To get reliable results from your training data, it's important that your testing data follows the same distributional guidelines as your training data, as suggested by Hu et al.[40]. There are many cases when knowing the distributions of the testing dataset is impossible, especially those involving uncertainty prediction problems. As a result, achieving superior results is tough for conventional learning algorithms. Temperature scaling is a process that rescales the logits of DNN outcomes. Guo et al. [8] were able to construct it for the purpose of achieving better calibration using only a single hyperparameter and a validation set. Kuleshov et al. [41] propose calibrated regression by making use of a rescaling procedure that is comparable to the one shown here. These methods are considerably more difficult to implement in comparison.

2.1.7 Evidential Deep Learning

Sensoy et al. [42] presented a direct model similar to the stance of Theory of Evidence [43], [44]. The idea is to evaluate the softmax, which is the usual result in a classification system. This softmax evaluation is used as a parametric collection of

unambiguous distribution. In our work, this collection is replaced with the parameters of a Dirichlet density. This allows us to display the predictions of the learned model in the form of a distribution spread over several viable softmax outputs instead of a singular value estimated for the result. Thus, we can say that the density can be instinctively interpreted as a clockwork system of these exact estimates.With the help of regular backdrop, neural network weights are used to miniaturize the resulting model's loss function. The data which is unfamiliar is sent back to prior belief as the system evaluates that the data itself admits its lack of awareness. Lastly, uncertainty approximation is authorized in a solitary model, while the forward pass is permitted through sorting multiple distributions.

2.1.8 Active Learning

Active learning is a type of machine learning technique that allows for the automated assembly of the training dataset based on either a list of data or a statistical distribution. In general, this kind of learning is referred to as "active learning." The estimates, as well as the measurements used to assess the degree of uncertainty, such as entropy, have typically been of a probabilistic character.
"Sampling strategy" or "query strategy" refers to the process of determining the most useful instances to label next. This may also be thought of as "query strategy." The term "acquisition function" refers to the scoring function that is used in the sampling procedure. If the data points with the better scores are labeled, it is anticipated that this will generate a higher value for the model training. There are many alternative approaches to sampling, such as the Uncertainty Sampling Method, the Diversity Sampling Method, the Expected Model Change Sampling Method, and so on. The term "uncertainty sampling" refers to a collection of methods that may be used to locate unlabeled elements in your existing machine learning model that are close to a decision boundary. In spite of the fact that it is simple to determine when a model is confident — there is one result with a very high level of confidence — there are many ways to calculate uncertainty, and the method that you select to use will be contingent on the use case you have in mind as well as the information that you have available. The cases for which the classifier has the least amount of confidence provide the most useful information. The underlying assumption here is that the cases for which the model has the least amount of confidence will almost certainly be the examples that provide the most challenge; more specifically, the examples that are located close to the class borders. Observing cases that are difficult to understand will provide the learning algorithm with the most valuable information on the class boundaries. There are several works regarding this way of measuring uncertainty[45]–[48].

2.2 Monte Carlo Dropout Implementations and Applications

MCD applications are increasing day by day. It can be seen in many fields of deep learning, especially in cases where risk factors are high. MCD allows uncertainty quantification, which can measure the robustness and reliability of a model. We will discuss some previous applications which were based on MCD.

The use of MCD is often seen in the healthcare or medical applications. Since, the risk factor is quite high. Camarasa et al. [25] propose a quantitative comparison of carotid artery lumen and vessel wall segmentation using numerous sequences of magnetic resonance (MR) images. They use four uncertainty metrics over 144 models. Stoean et al. [23] exhibits necessity and advantages of MCD in medical applications, where the major complication stems from the unusual nature of sensor-derived data. Milanés-Hermosilla et al.[49] categorize motor images using Convolutional Neural Network with a shallow depth and an ensemble model. In addition, they utilise MC Dropout in order to assess the level of ambiguity associated with their estimates, which contributes to the increased dependability of the framework. Avci et al.[50] also propose using dropout throughout training and test phases as well as averaging several predictions to increase accuracy while minimizing and measuring uncertainty. Tabarisaadi et al. [51] propose the computer aided image-based dermatological treatment of cutaneous malignancies. To diagnose skin cancer, uncertainty measurement processes like MCD, Bayesian Ensembling, and Spectral Normalized Neural Gaussian Process are employed. The comparison of the aforementioned algorithms' levels of performance are examined from a variety of perspectives. The cardiac cine data was used to train a U-Net segmentation algorithm. MCD sequencing of the U-Net architecture was used to each frame of the robust circulatory datasets in order to evaluate Standard deviation(SD) maps. The best frames for epicardial and endocardial segmentations were chosen with the use of an uncertainty estimate calculated from the total of the SD values. Their results[52] show that it is possible to accomplish robust and automated myocardial segmentation by using a model trained on cardiac cine imaging data to dynamic myocardial perfusion data. Subjects having higher degree of ambiguity in terms of endocardial and epicardial segmentation might be forwarded for additional review and rectification by medical specialists using the uncertainty estimates as their screening tool. An uncertainty-based criterion for decision referral, as demonstrated by Chagas et al. [53], may significantly boost the performance of Membranous nephropathy classification, bringing accuracy up to 96%. As a last step, they analyzed the relation between uncertainty and complexity results as described by pathologists. Gour and Jain [54] construct a model with the foundation of convolutional neural networks, termed UA-ConvNet, that includes an evaluation of correlated uncertainty in the model's predictions for the automated identifcation of COVID-19 illness via chest X-Ray scans. The suggested method makes use of the EfficientNet-B3 model and MCD, where it has been modified using the chest X-ray pictures. They show that their proposed approach is more effective than the current methods in identifying COVID-19 patients from chest X-ray images. Xia et al.[55] provide a framework (MCD, Bayesian, Ensemble) to assess the estimated uncertainty's capacity to capture variations in biosignal datasets of varying kinds and intensities. Five exemplary uncertainty quantification approaches are evaluated on three classification tasks using breathing sounds and electrocardiography data.

In case of other applications, Ma et al. [28] propose the categorization of radio frequency transmitters and show that the suggested method outperforms the basic ensemble average techniques. To save the computational cost, they simply enable dropout layers near the neural network's output and reuse the computation from previous levels during testing, leaving any additional dropout layers off. Three Insulated Gate Bipolar Transistors(IGBTs) were used for aging experiments, which

are run until the transistors fail to operate, by Xiao et al.[56]. A MCD and predictive model depending on its own attention is presented for Health indicator(HI) estimation and Remaining Useful Life(RUL) calculation of the tested transistors to ensure the dependability of the electric system. By including uncertainty into the prediction of the multi-step-ahead data, MCD is coupled to offer prediction confidence in this experiment. Laakom et al.[57] present a technique that aggregates several deep learning approaches based on their output uncertainty to eliminate the effect of lighting on the scene's colorful pigments and return them to their natural state. They have used MCD for assessing the relative uncertainty of each method, and estimating the final illumination by summing the estimates from the several models and weighting them by the log-inverse of their respective uncertainties. In order to accurately record the human face's movement throughout the performance of a emotional gesture, Heidari and Iosifidis [58] use a spatio-temporal bilinear layer as their skeleton. As an added bonus, it uses MCD for uncertainty quantification of the model, which is crucial for analyzing and treating such scenarios. To make YOLOv3's crater detection more trustworthy, Myojin et al. [59] set out to remove false positives from detection results by measuring uncertainty reflecting variance while detecting several times using MCD. Since the U-Net architecture design of YOLOv3 model may catch objects of varying sizes and is effective for identifying craters, they suggested a way of calculating uncertainty for the setting criteria for each object detection layer that captures an item of a different size. Furthermore, we analyzed uncertainty by disentangling its axial position and diameter in order to assess it from a variety of perspectives. Therefore, more false positives might be weeded out in comparison to assessment utilizing mixed detection layers, where the position and size are not separated. By giving more weight to the models with the highest MCD prediction confidence, a new ensemble technique is proposed by Yang and Staib [60]. Classification performance is improved above the results of a single model thanks to their ensemble technique, which substantially reduces the likelihood of overfitting. More importantly, they show that the model can recognize outliers in the dataset and achieved a prediction score of 0.8929. In another paper[61], authors propose using MCD to improve the generalizability of a DNN for voice augmentation. We demonstrate the DNN's improved enhancement performance under unseen noise and SNR circumstances using MCD. The dilemma of amplitude versus offset inversion is addressed by Junhwan et al.[62] using a CNN model with MCD. Aleatoric uncertainty is capable of determining the anomalous data, while the epistemic uncertainty allowed us to evaluate the model's performance on test data and reveal that the risk of prediction increased as one moved further away from the train data. Thereby, we may use the data based upon the degree of uncertainty to interpret the forecasts with less danger. In order to determine the pixel-by-pixel error in their predictions, Fisher et al.[63] train a U-Net model with the application of MCD on satellite photos. Results reveal that the suggested model is superior to the prior state-of-the-art model, with better performance and reduced uncertainty when evaluated on an unknown geographical location. Ghoshal et al.[64] assess uncertainty in deep learning approaches using drop-weights-based BNN. They demonstrate that the prediction performance of the model was significantly correlated with its degree of uncertainty. When attempting to foretell the amount of organic carbon in the soil, Padarian et al.[65] examine the ambiguity of a deep learning soil spectral model using two different assessment methods. If the

model is aware of generating forecasts that varies from the training data when it is supposed to give higher prediction ranges, then both methodologies may be used to quantify uncertainty. As part of the ensemble-building procedure of BatchEnsemble, Jain and P.K.[66] integrate MC-dropout. As long as a few parameters are tweaked, the suggested method, Monte-Carlo BatchEnsemble, may generate ensembles with lower prediction correlation. The experimental outcomes verify the efficacy of the suggested method for image classification. By using sequential Monte Carlo adaptation of the masks in a dropout regularisation network layer, Carreno-Medrano et al.[67] present a straightforward method of neural network adaptation. As seen in a real-world application of human behavior modeling, these masks provide for some interpretability by collecting both local and some global contextual information. When combined with multi-task training, the results reveal that this straightforward adaptation technique may outperform computationally complicated meta-learning schemes and help avoid the catastrophic forgetting issues that plague gradient descent in online adaptation contexts. A reversible form of steganography using deep learning is proposed by Chang [68], and the uncertainty of prediction models is analyzed using a Bayesian approach. The MCD is used to provide an approximation of the speculative distribution, from which both random and systematic errors in our predictions may be calculated. To achieve unsupervised uncertainty estimation, a two-headed network of neurons is built. The contribution of uncertainty analysis is confirmed by experimental findings showing exemplary steganographic results in comparison to a non-Bayesian standard. Unmanned aerial vehicle building video and satellite-captured post-disaster pictures are used to develop and train two distinct networks by Cheng et al [69]. Digital damage assessment with the help of Artificial Intelligence can yield results that are more explicable and risk-aware by employing MCD for the investigation of ambiguity, estimating reliability of model based on its confidence score and attention region.

In order to demonstrate effectively methods of predictive uncertainty perform in practice in the research area of NLP, Landeghem et al.[70] study the application of predictive uncertainty techniques focusing on classification of texts using a number of different labels and classes. To be able to analyze the reason for well-known scalable uncertainty estimation methods like MCD and Deep Ensemble, as well as notable extensions like Heteroscedastic and Concrete Dropout, consistently under-estimating uncertainty, the authors perform experiments to test 1-D CNNs and prior-trained transformers on 6 hand written text categorization datasets. Using current findings on how variational Bayesian and ensembles approaches handle the loss landscape, they argue that combining posterior approximation processes improves uncertainty estimates. By examining in-domain calibration, classification via cross-domain, and novel class resilience, the authors conclude that their suggested technique combining Deep Ensemble with Concrete Dropout displays improved performance, even at a lower ensemble size. In another work, Shelmanov et al.[71] compare multiple uncertain estimates in text classification tasks for the cutting-edge Transformer model ELECTRA and the speed-oriented DistilBERT model. They use many stochastic passes with the MCD and a dropout based on Determinantal Point Processes to derive uncertainty estimates. Hu et al.[72] also suggest using empirical uncertainty in out-of-distribution identification for tasks involving text classification. They present a low-cost framework that uses auxiliary outliers as well as pseudo off-manifold samples to train the model along with prerequisite understanding of a specific class that

has a large absence for non-uniformly dispersed data.
In computer vision, MCD-based Transformer models have also been implemented. Using ViT and a semi-supervised framework, Wang et al. [73] propose medical picture semantic segmentation using the capabilities of ViT. Utilizing MCD and an uncertainty estimation approach, they enhance the semi-supervised performance. In the training process, the method of filtering out unclear images based on an uncertainty value and the weighted sum of two losses is being looked into more. Moreover, Zhao et al.[74] offer a novel Transformer-based architecture mixed with Bayesian theory for facial emotion recognition. They also adjust the feature extractor module and training approach to account for the uncertainty of the training data. For camouflaged object recognition, Yang et al.[75] introduce uncertainty-guided transformer reasoning, a technique that use a stochastic representation architecture and transformers to effectively deliberate under uncertainty. Initial estimates and related uncertainty are obtained by learning a conditional distribution across core output. Consequently, reasoning about these uncertain areas using an attention approach yields conclusive forecasts.
Other than these applications on CNN, Transformers, RNNs, there have been no significant work on INNs and GNNs based on MCD particularly. Dwivedi et al. [76] provide a benchmark framework that can be utilized to evaluate novel GNN architectures and findings. However, the primary emphasis of this work is on the performance of GNN and its insights. Only considering model performance may lead to a lack of attention for model's reliability when applied to out-of-scope data. In our work, we explore the unexplored area of these architectures utilizing MCD.

2.3 Monte Carlo Dropout Analysis and Findings

Other than applications, there are several works on MCD analysis and findings. The literature described below, shows many analytical aspects of MCD and its behaviour. The number of studies focusing on MCD is increasing day by day. Seoh et al. [77] propose a qualitative analysis of Monte Carlo Dropout, where they propose potential benefits and costs from their results. Another study suggested by Verdoja et al. [22] focuses on behaviour of MCD. Their findings suggest that the MCD characteristics found on a single-layer linear NN remain intact as the NN expands in scale and complexity.
Sicking et al. [78] established that in the case of indefinitely wide layers, random neural networks with constant parameters under dropout converge to Gaussian processes. A proposed rationale raises expectations that poorly correlated, indefinitely broad networks trained with (full-batch) gradient descent may exhibit comparable behavior.
A total of three semi-artificial data generators were developed and compared by Miok et al [79]. The architecture of the variational autoencoder serves as the foundation for the VAE generator. Both the MCD-AE as well as the MCD-VAE make use of Monte Carlo Dropout within the systems of autoencoders and variational autoencoders, respectively. Their research shows that MCD-AE and MCD-VAE produce results more quickly than VAE. Data is generated by MCD-AE and MCD-VAE the same way as that of selected seeding instances. If the seed instances supplied are anomalous or otherwise noteworthy, this can be a really helpful feature.

In addition to enhancing the majority of classification metrics for ordinal, multi-class, as well as binary models, Lemay et al [80]. establish that MCD models also allow a substantial rise in repeatability — in other words, enhancement of at least one repeatability metric. Consistency in results is an essential part of having a reliable model. Predictions from ideal repeatable models are consistent across multiple experiments performed under the same conditions. In practice, this translates to more consistency between the retest and test results with regards to class and score agreement. No matter what disease was being addressed or the way the models were constructed (DenseNet or ResNet), repeatability improved. Not all regression models benefited from MC iterations, and in other cases (such as knee osteoarthritis and preterm birth retinopathy classification), it even worsened classification performance.

For increased resistance to noisy labeling, Goel and Chen [81] propose using MCD to further sparsify and regularize deterministic neural networks. As a result of its sparser yet less volatile representation, MCD is more capable to perform and generalize in noisy-label circumstances. Regularization is provided by MCD to ensure that neurons are not too impacted by the noisy labels. However, these neurons do not undergo regularization at the time of testing, which contributes to their resilience against noisy labels. MCD models construct representations that are less susceptible to overfitting to noisy labels because there are fewer available parameters to over-explain the training label noise.

Chapter 3

Research Objective

In chapter 2, we have learned about several applications of MCD in computer vision, natural language processing, etc. In spite of this, we identified the following limitations in the existing research works:

- They focus on one particular application
- Most of them only focus on increase in performance
- There is no benchmark for uncertainty quantification with MCD
- Lack of any comparative analysis or application among state-of-the-art NNs despite several works being published with ANNs, CNNs and Transformers (in NLP).

Our study compensates for the absence of such important aspects by focusing on the following research objectives.

- Providing a complete roadmap for MCD in state-of-the-art NNs
- Exploring the unexplored area of uncertainty estimation using MCD
- Analysis of difference in performance after using MCD
- Comparison between different models for same tasks
- Providing a reliable framework for risky tasks such as medical image diagnosis
- We propose two novel architectures, one in ECG trace image classification and other one in skin cancer identification and malaria detection from parasitized cell images
- We use popular datasets in this work for benchmarking the NN models in terms of uncertainty measure.

Figure 3.1 represents our whole workflow, where we visualize the tasks performed in this thesis.

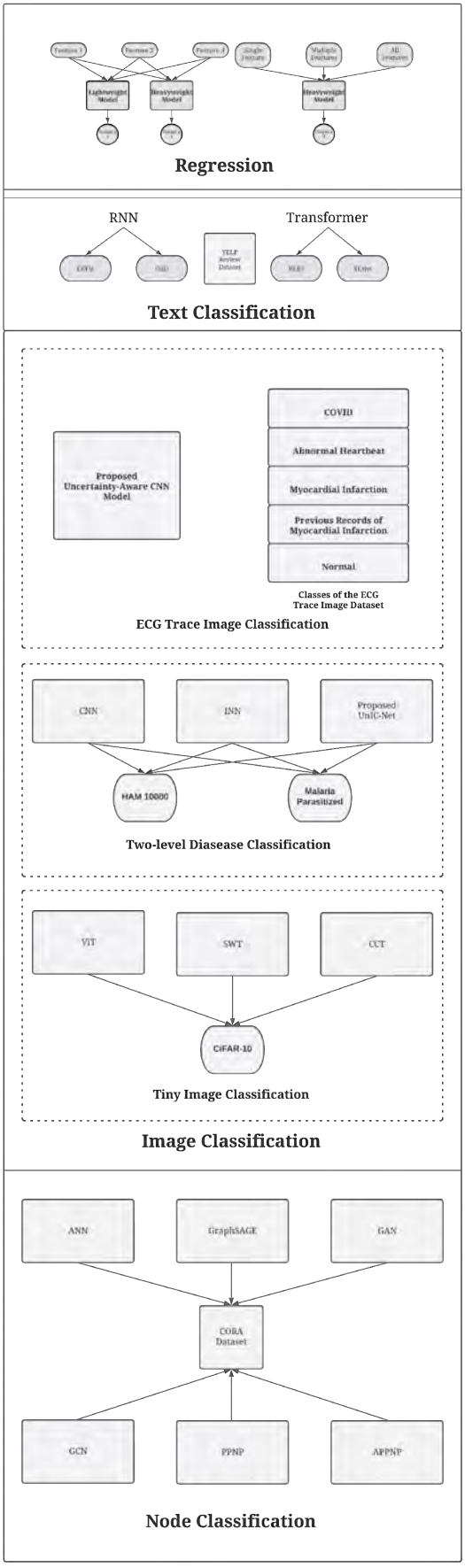


Figure 3.1: Block Diagram of our Research Workflow

Chapter 4

Dataset Description

4.1 California Housing Price

Dataset used in this experiment is known as 'California Housing Dataset'. This dataset was initially featured in [82]. The California Housing Price dataset is a collection of information about housing prices in California. It contains information about the location, size, and price of homes in California, as well as other features such as median income, median house age, and number of rooms. The dataset contains 20,640 observations, with has 9 feature columns, where the 'median house value' is column generally taken as the target.

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
0	-114.31	34.19	15.0	5612.0	1283.0	1015.0	472.0	1.4936	66900.0
1	-114.47	34.40	19.0	7650.0	1901.0	1129.0	463.0	1.8200	80100.0
2	-114.56	33.69	17.0	720.0	174.0	333.0	117.0	1.6509	85700.0
3	-114.57	33.64	14.0	1501.0	337.0	515.0	226.0	3.1917	73400.0
4	-114.57	33.57	20.0	1454.0	326.0	624.0	262.0	1.9250	65500.0

Figure 4.1: First 5 Samples of the California Housing Price Dataset

The dataset includes the following features:

- longitude: a measure, in degrees, from the east to west position of a location
- latitude: a measure, in degrees, from the north to south position of a location
- housing median age: the median age of homes in the area
- total rooms: the total number of rooms in all homes in the area
- total bedrooms: the total number of bedrooms in all homes in the area
- population: the total population in the area
- households: the number of households in the area
- median income: the median income of households in the area
- median house value: the median value of homes in the area (target variable)

Besides its obvious relevance as a standard for evaluating the efficacy of various machine learning models, this dataset also finds extensive application in the field of housing price research. For predicting the median house value in a region based on other variables in the dataset, it is often used to train and assess regression models. The dataset is often used to study the relationship between housing prices and various socio-economic factors, such as median income, population density, and age of housing. Additionally, this dataset can be used to study the impact of geographical factors such as latitude and longitude on housing prices.
In our experiment, we use a split dataset with a train to test ratio of 7.5:1.5. There are 17000 train data and 3000 test data used in the experiment. We use the raw dataset, therefore it did not require any preprocessing. Figure 4.1, shows first five samples of the dataset.

4.2 Yelp Review Polarity

The Yelp Review Polarity dataset [83] is a collection of reviews, labeled as either positive or negative, retrieved from the Yelp website. This dataset is compiled by interpreting negative polarity for 1 and 2 ratings and positive polarity for 3 and 4 ratings.

	class	text
0	0	Been going to Dr. Goldberg for over 10 years. ...
1	1	I don't know what Dr. Goldberg was like before...
2	1	I'm writing this review to give you a heads up...
3	0	All the food is great here. But the best thing...
4	1	Wing sauce is like water. Pretty much a lot of...
5	1	Owning a driving range inside the city limits ...
6	1	This place is absolute garbage... Half of the...
7	0	Before I finally made it over to this range I ...
8	0	I drove by yesterday to get a sneak peak. It ...
9	1	After waiting for almost 30 minutes to trade i...

Figure 4.2: First 10 Samples of the Yelp Review Polarity Dataset

The dataset contains 560,000 reviews, with as many of positive reviews as there are negative reviews. It also comprises 280,000 instances for training and 19,000 instances for testing, which are further grouped by polarity type. Each review is represented as a sequence of words, and the dataset includes a total of 1.6 million unique words. Negative and positive polarity are represented by class 1 and class 2 respectively. In our experiment, the positive class is converted to 0 while the negative class remains the same since binary classification is performed. Figure 4.1, shows first ten samples of the processed dataset.

The dataset includes the following features:

- Review text: the text of the review
- Sentiment: either the positive or the negative sentiment of the review

The dataset is widely used for research in NLP tasks and sentiment analysis. It is widely utilized in the training and evaluation of ML and DL models for text sentiment classification. The dataset is useful for studying the relationship between the words used in a review and the sentiment of the review. It can also be used to study the influence of various factors on the sentiment of reviews, such as the length of the review, the use of specific words, and the presence of punctuation.

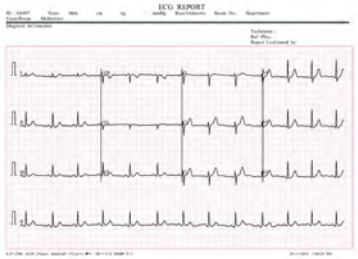
4.3 ECG Trace Images

We use a [84] consisting of ECG scans of patients with cardiovascular conditions and COVID-19 in our research. This dataset comprises information on 1937 individual patients. Data is collected through electrocardiogram machines named "EDAN SERIES-3" which are installed in hospitals and clinics all over Pakistan.

(a) Covid-19

(b) Abnormal Heartbeat

(c) Myocardial Infarction (MI)



(d) Normal

(e) Previous History of MI

Figure 4.3: Sample of each classes from the dataset

The dataset consists of 12-lead standard electrocardiogram (ECG) scans from a variety of patients, labeled as to whether or not they exhibit the following five conditions:

Normal Person, **Myocardial Infarction (MI)**, **Abnormal Heartbeat (HB)**, **Previous History of MI (PMI)**, and **COVID-19**. Small shifts in the ECG readings are typical with cardiac issues. For instance, the time between two peaks or a specific wave. Oftentimes, these variations are employed as diagnostic markers. Some excerpts from the dataset are displayed in Figure 4.3.
The version that was processed was obtained from the source [85]. The data are trimmed from this source so that they only include the most relevant parts of the signal [86]. A reduction in computing time and an increase in classification accuracy is resultant because of this. In addition, the ECG scans from the dataset have been modified by transforming them to grayscale (one channel) from RGB (three channel) images and scaling them to a resolution of 70×70.

4.4 Malaria Parasitized Blood Cell Dataset

This research makes use of a dataset that was obtained from the National Institutes of Health [87]. It is made up of segmented cells that were taken from pictures of thin blood smears on slides that were taken during the Malaria Screener research activity. Slides smeared by thin blood and stained with Giemsa were obtained from 150 patients infected by P. falciparum and 50 healthy individuals at the Chittagong Medical College Hospital in Bangladesh. These slides were then photographed.

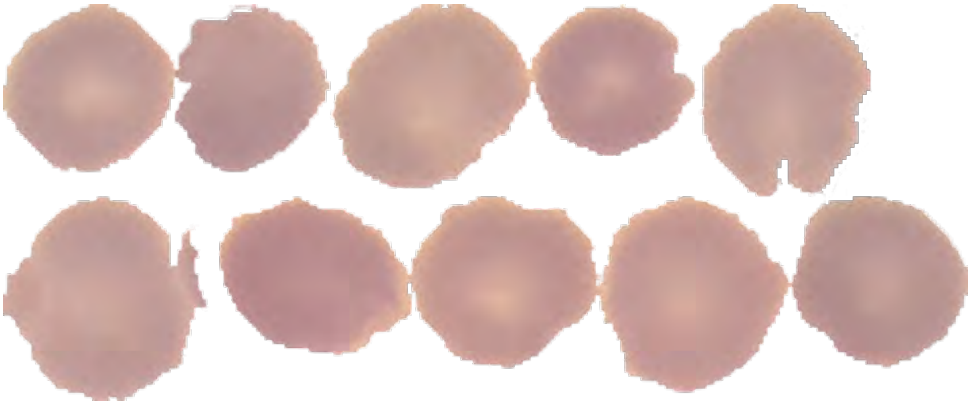


Figure 4.4: Sample Images of Normal Cells

An experienced slide reader at the Mahidol-Oxford Tropical Medicine Research Unit was responsible for the manual annotation of the photographs. There are a total of 27,558 segmented cell photos in the database, with an equal number of parasitized and uninfected red blood cell images which are segmented. Both sets consist of 13,779 images individually. Those that were considered positive included plasmodium, whereas samples that were considered negative did not contain plasmodium but might contain other sorts of items, such as staining artifacts or contaminants.
The segmented red blood cell patches range in size from 110 to 150 pixels and feature three color channels (RGB). These patches were later re-sampled to have an output dimension of 200 by 200 pixels, where channel depth is three, and the floating point precision used is 32-bit (FP32). This was done in order to ensure compatibility with the data input specifications of the different classification algorithms employed in this study. In addition, a variety of pre-processing strategies are utilized, the specifics of which will be elaborated upon in a subsequent section of this discussion.

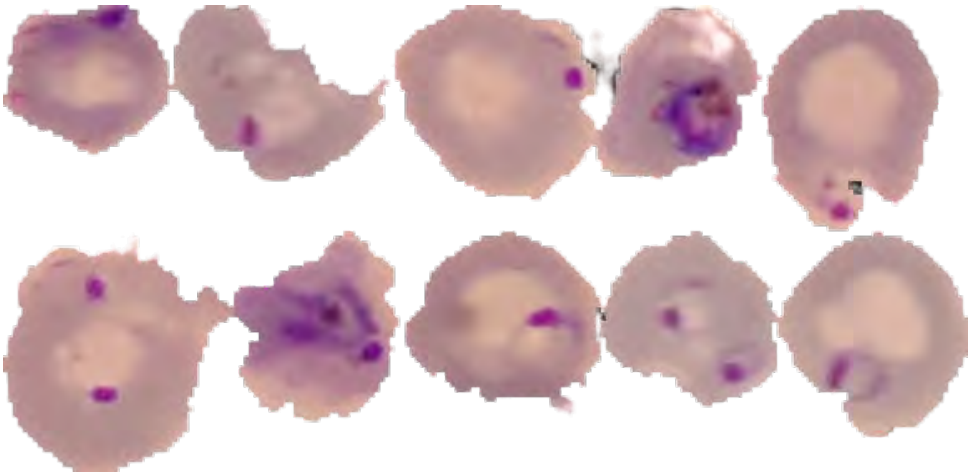


Figure 4.5: Sample Images of Parasitized Cells

Figure 4.4 shows some samples from the dataset that have segmented red blood cells that have not been infected, and Figure 4.5 shows some samples that contain segmented red blood cells that have been parasitized.

4.5 HAM10000: Skin Cancer Dataset

Due to its limited size and insufficient variety, the current collection of dermatoscopic pictures poses considerable obstacles when it comes to the neural net training for the computer - aided diagnosis of skin lesions with pigmentation. These obstacles can be particularly difficult to overcome. We have made use of the dataset HAM10000 [88], also known as Human Against Machine with 10000 training photographs, in order to assist with this situation.

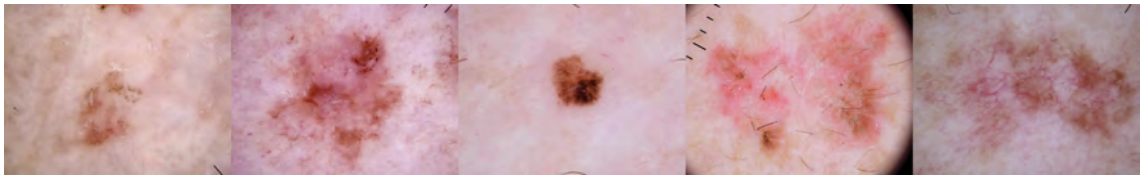


Figure 4.6: Sample Images of Bowen's disease, also known as "actinic keratoses and intraepithelial carcinoma" or "akiec"

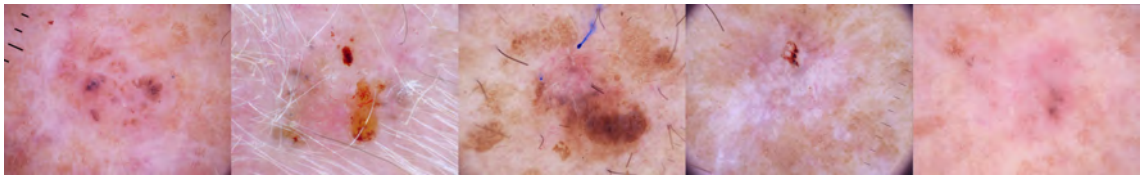


Figure 4.7: Sample Images of basal cell carcinoma (bcc)

The final dataset contains 10,015 dermoscopic images, all of which may be utilized in the training dataset for machine learning research in the academic world. The

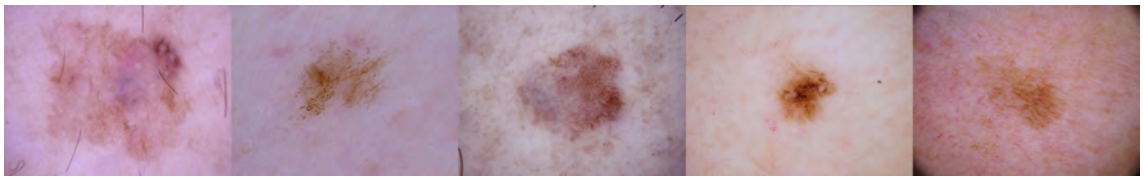


Figure 4.8: Sample Images of benign keratosis-like lesions (bkl)

dataset includes a sample population that is representative of the entire range of important diagnostic subcategories that are relevant to the study of pigmented lesions. These cases include but are not limited to: melanoma (mel), Bowen's disease (akiec, abbreviated from actinic keratoses and intraepithelial carcinoma), melanocytic nevi (nv), lesions like benign keratosis (seborrheic keratoses or solar lentigines and lichen-planus like keratoses, bkl), vascular lesions (angiokeratomas, angiomas, pyogenic granulomas and hemorrhage, vasc), dermatofibroma (df), as well as basal cell carcinoma (bcc). Figure 4.6 represents some samples of akiec disease. Other diseases such as, bcc, bkl, df, mel, nv and vasc samples are shown in Figures 4.7, 4.8, 4.9, 4.10, 4.11 and 4.12 respectively.

Figure 4.9: Sample Images of dermatofibroma (df)

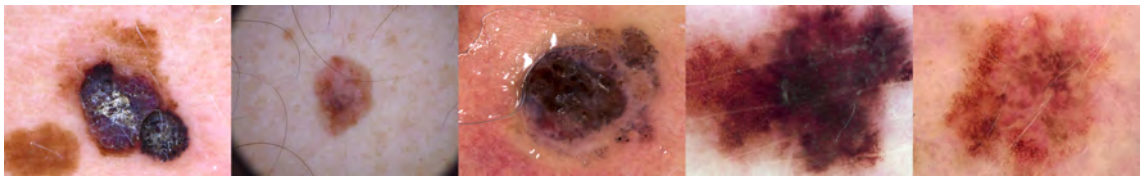


Figure 4.10: Sample Images of melanoma (mel)

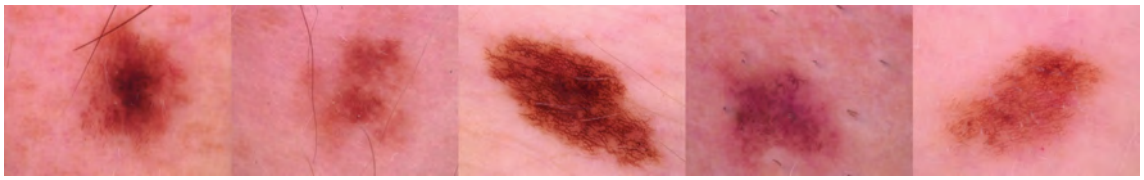


Figure 4.11: Sample Images of melanocytic nevi (nv)

In the remaining cases, the final result will be determined by either a further investigation, the consensus of the experts, or verification by invivo confocal microscopy. Histopathology is used to confirm more than fifty percent (50%) of all lesions. The collection contains lesions that have several photos attached to them.

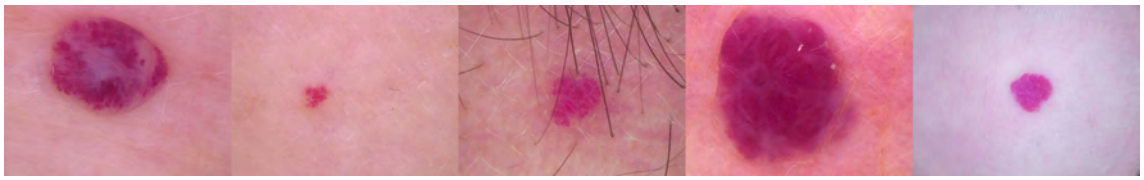


Figure 4.12: Sample Images of vascular lesions (vasc)

This dataset is widely used for research on skin lesion classification and is considered as one of the largest publicly available datasets for this purpose. The dataset was created by collecting images from different sources and different patients. The dataset is useful for developing and evaluating machine learning models that can assist in the diagnosis of skin lesions.

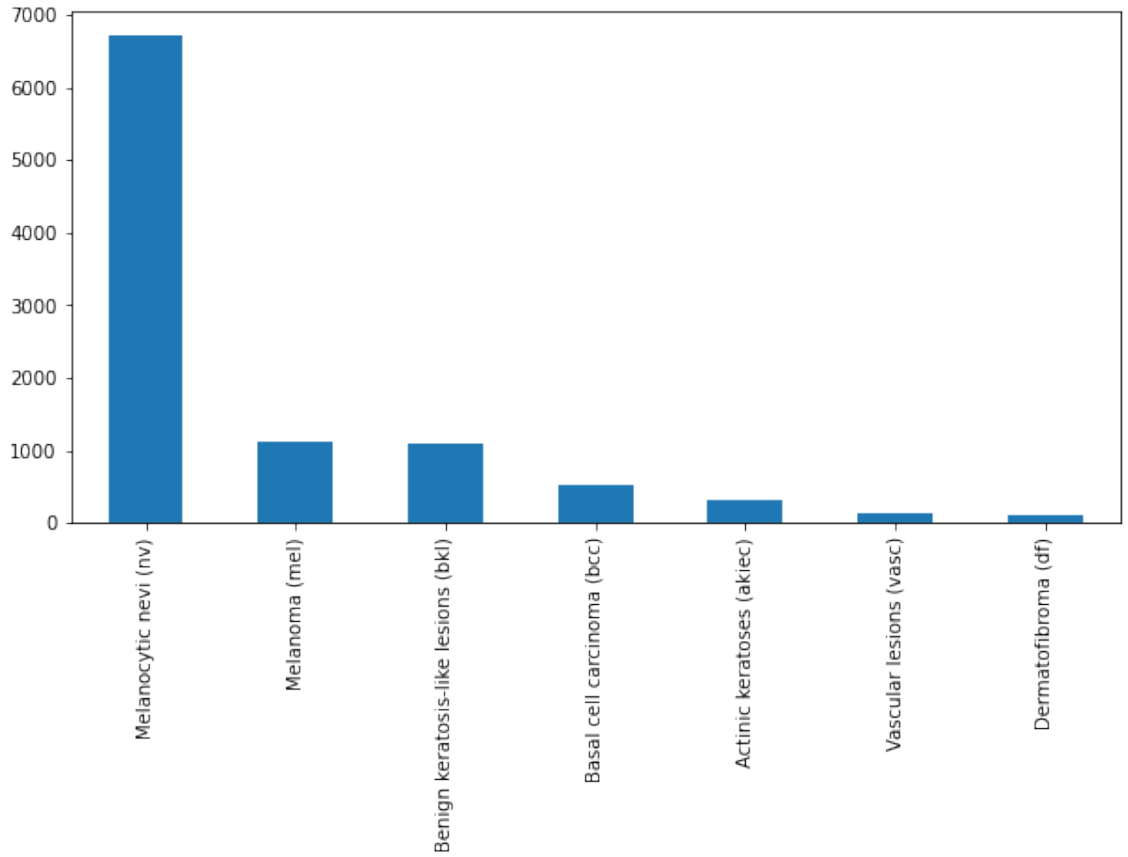


Figure 4.13: Frequency of Each Class in HAM10000

4.6 CIFAR10

For the purpose of our experiment, we make use of the CIFAR-10 dataset [89]. The CIFAR-10 dataset is comprised of 60,000 RGB images with a resolution of 32×32 pixels, organized into 10 classes with 6,000 images each.
There are 50,000 training images and 10,000 testing images. The 10 classes are: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The dataset

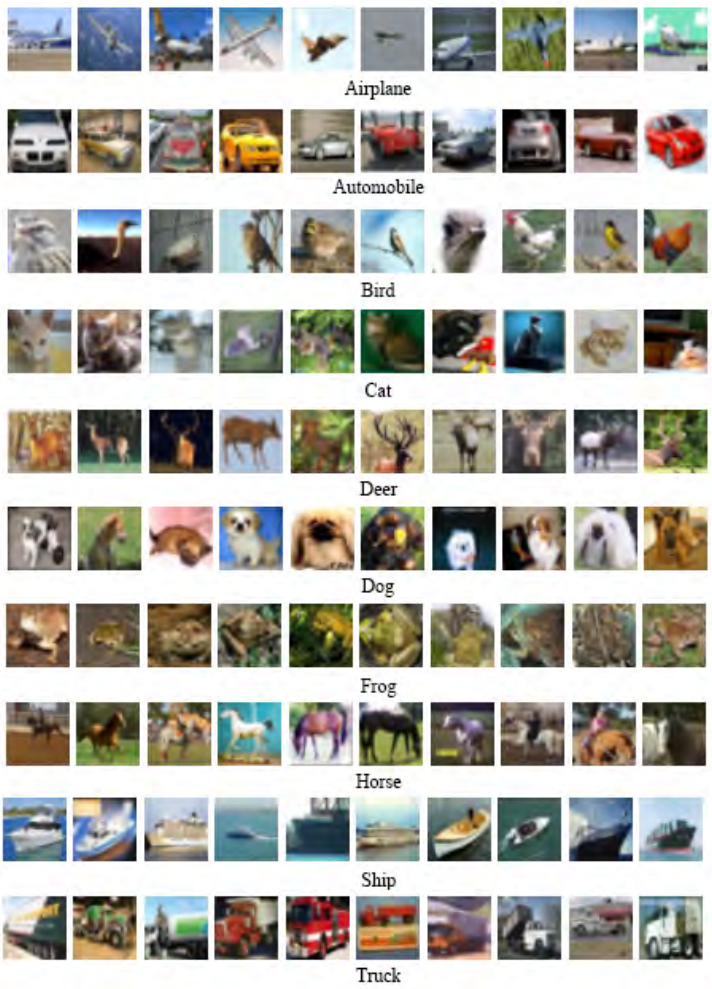


Figure 4.14: Image Samples from the CIFAR10 Dataset

allocate 10,000 images to five batches for training and one batch for testing, each, from the entire collection. 1000 images are selected at random per class for the test batch. The rest of the images are distributed in equal amounts to each batch of the train set, meaning they consist of exactly 5000 images. However, there may be more images of a particular class than another in each batch. For validation, 10% data is used from the training set. We do not process the images here, we use raw dataset for benchmarking. Figure 4.14 shows some samples of the dataset.
The scans used in the CIFAR-10 dataset are small and low resolution, making it a good dataset to use for developing and testing image classification models that are designed to work with small images. Additionally, the dataset's small size allows for fast experimentation and iteration, which makes it a popular choice for researchers and practitioners.

4.7 CORA

The CORA dataset [90] is a compilation of scientific publications on various topics in computer science. It contains 2,708 machine-readable research papers, each labeled with one or more of seven classes: Probabilistic Methods, Genetic Algorithms, Neural Networks, Case Based, Rule Learning, Reinforcement Learning, and Theory.

Each paper is represented by a bag-of-words feature vector and a citation graph that represents the connections between papers. Each distinct publication available in the dataset is characterized by a word vector of either 0 or 1 denoting the non-appearance or appearance of the relevant dictionary word containing 1433 distinct terms. The CORA dataset consists of a collection of items and their relationships, allowing for the testing of machine learning techniques that can handle relations.

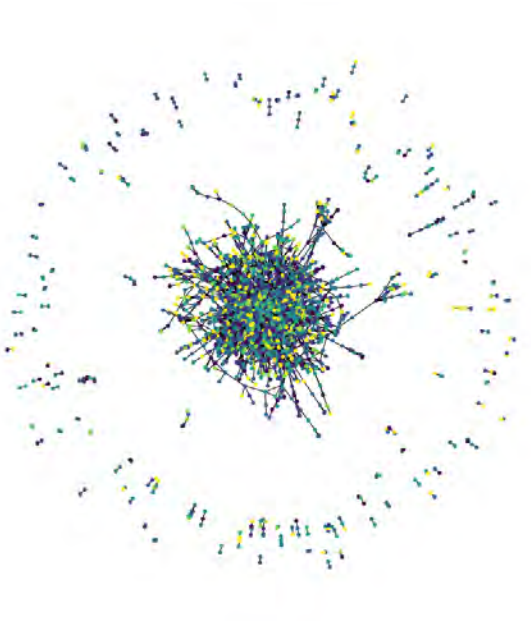


Figure 4.15: Graph Visualization of CORA dataset

The dataset is widely used for evaluating machine learning models, especially for graph-based algorithms, and is commonly used as a benchmark dataset in the field of scientific literature analysis. Our work is based on node classification benchmark, based on uncertainty. Therefore, this dataset is a great choice for our study.
Figure 4.15 is the the visualization of citation graph for the CORA dataset. Each paper is represented by a node in the graph, and the node's color indicates the topic of the paper. Please be aware that we are only displaying a subset of the papers that are included in the collection; the subset size is 3,000.

Chapter 5

Methodology

5.1 Monte Carlo Dropout

5.1.1 Dropout

Deep neural network models are known to have numerous non-linear hidden layers, thus they are immensely expressive, which allows them pick up on subtle nuances in the connection between outputs and inputs. However, if training data is insufficient, many of these intricate connections will be due to sampling noise. Due to this, the noise will remain in the set for training but will not be included in the actual test data, despite the fact that both sets of data are derived from the same distribution. Since this causes overfitting, numerous strategies have been devised to address the problem. One such strategy involves pausing the training when the validation set's performance begins to deteriorate, followed by implementing weight restrictions of varying types, for instance, the L1 and L2 regularization as well as sharing of soft weight (Nowlan and Hinton, 1992) [91]. If computing power is unlimited, then the most effective method for "regularizing" a model having a set number of parameters is to take an average of predictions obtained from all possible configurations of those parameters and then weight them based on the posterior probability that each configuration would have given the training data. For basic or small models, this can be estimated very effectively (Xiong et al., 2011; Salakhutdinov and Mnih, 2008)[92], [93], however it is desirable to obtain a level of performance close to that of the Bayesian gold standard with significantly less computational power expended. Our goal is to accomplish this through approximation of the geometric mean of the predictions made by an exponentially large amount of learning models which share parameters. Each prediction will have the same weight.
Combining several models is typically effective in improving machine learning techniques. Averaging the outputs of multiple individually trained nets is too costly for big neural networks. If Neural Network models are to be merged, we must ensure that they have different designs or were trained using different sets of data. Finding suitable hyperparameters for each model is difficult, and training big networks needs a lot of computational power. Having enough data to train multiple networks independently on distinct subsets of data can be challenging when working with large networks. Even if multiple large networks could be trained, it would be unrealistic to use them all at once during testing for swift-responding systems.
These two problems can be solved through the implementation of dropout [26]. In

NNs, "dropout" is used to define the action of omitting certain hidden and invisible units. Figure 5.1 depicts how this is done, i.e, the neuron along with its inbound and outbound links are momentarily disconnected. However, the neuron which gets "dropped" is chosen at random. Basically, an individual unit has an independent and predetermined probability "p". The probability "p" may be selected by a validation set. It may also be set at 0.5 which is the optimal value for many networks and its applications. The best probability of retention for input units is almost 1. In summary, the output of every neuron is proliferated using a binary mask which is derived from Bernoulli distribution, in the training stage. This is how the neurons are initialized to zero, following which the NN is applied at the testing stage.
Simply put, when dropout is applied to a neural network, a relatively "thin" network is sampled from the original network. All the nodes that were able to avoid being dropped due to dropout make up the shortened network (Figure 5.1). It is feasible to view an n-unit neural network as a network comprised of a collection of 2n thin networks. All the weights in these networks are shared, thereby limiting the overall amount of parameters to less than or equal to $O(n^2)$. A freshly sampled and trained thin network is developed for every demonstration of a learning instance. In other words , training a neural network with dropout is comparable to training a set of 2n thinned networks where there is heavy weight sharing although training of each thinned network is not frequent. Predictions from an increasingly large number of thinned models cannot be explicitly averaged during testing phase. Nevertheless, in real life cases, a straightforward approximation of the mean performs excellently. For experimental purposes, the target is to employ a singular neural network without the use of any dropout. In this network, the weights are reduced versions of the originals used during training. As illustrated in Figure 5.2, the output weights of units at testing time, that are maintained at fixed probability of "p" during training are multiplied by p. This guarantees that the output of every hidden unit during testing phase is identical to the expected performance (under the distribution utilized to exclude units during training phase). A single neural network can be created with shared weights from 2n networks by employing this scaling strategy during testing. We observed that compared to training with alternative regularization approaches, training a network with dropout and utilizing this approximation averaging strategy during testing phase leads towards relatively lower misclassifications on a large array of classification tasks.
The use of dropout in neural network models assists in minimizing cases of overfitting. Standard backpropagation learning generates fragile co-adaptations that function for the training data but are inapplicable to unseen data. This co-adaptation is disrupted by random dropout, which renders the existence of any concealed unit untrustworthy. This approach was discovered to enhance the performance of neural networks in a wide variety of application areas, including natural language processing, computer vision, and almost anything else involving neural networks. This shows that dropout is a generic strategy that is not domain-specific.

5.1.2 Dropout as a Bayesian Approximation for Representing Uncertainty in Deep Learning Models

The idea of employing dropout was brought forth by Gal and Ghahramani [21], who used it as an estimation of probabilistic Bayesian models for deep Gaussian processes.

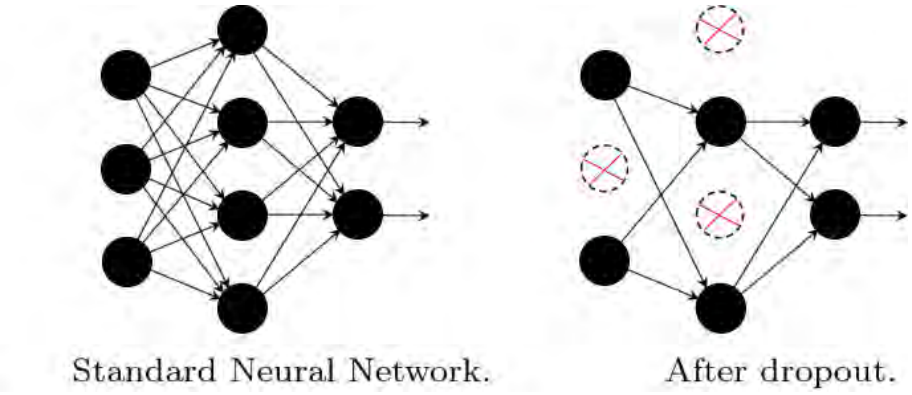


Figure 5.1: Dropout in Neural Networks

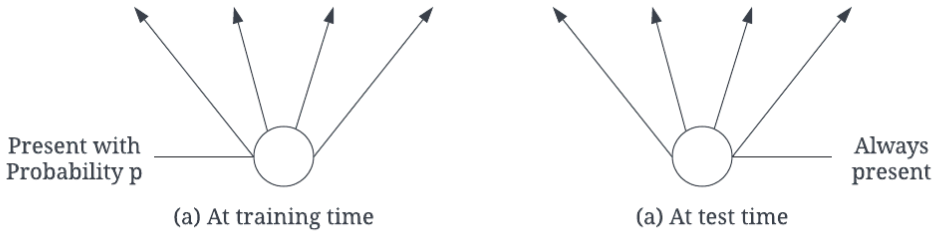


Figure 5.2: Presence of an neural unit during training(a) and testing(b) time

A set of predictions showcasing the uncertainty estimations can be produced using MCD. The MCD method involves executing several stochastic forward passes in a NN by employing activated dropout during the testing stage.
An NN model which is trained with dropout f_{nn} can obtain uncertainty for a certain sample x by collecting the predictions of T inferences with different dropout masks. Here $f_{nn}^{d_i}$ depicts the model with dropout mask d_i. Therefore, we get a representation of the range of model results for some given sample x shown below:

$$f_{nn}^{d_0}(x), ..., f_{nn}^{d_T}(x) \tag{5.1}$$

Calculating the mean and variance yields an ensemble prediction. In this case, the model's uncertainty about x is estimated, and the forecast refers to the sample-average of the posterior probability distribution for this model.

$$Predictive\, Posterior\, Mean,\, p\, =\, \frac{1}{T}\sum_{i=0}^{T} f_{nn}^{d_i}(x) \tag{5.2}$$

$$Uncertainty,\, c\, =\, \frac{1}{T}\sum_{i=0}^{T}[f_{nn}^{d_i}(x) - p]^2 \tag{5.3}$$

The dropout NN is not modified, only the outcomes of the stochastic forward passes are collected. Through this technique, the predictive mean and model uncertainties are evaluated. As a result, existing dropout trained NN models can have the data applied to them.
The probabilistic interpretation of dropout known as MCD enabled the deep learning community to get model uncertainty from already existing deep learning models.

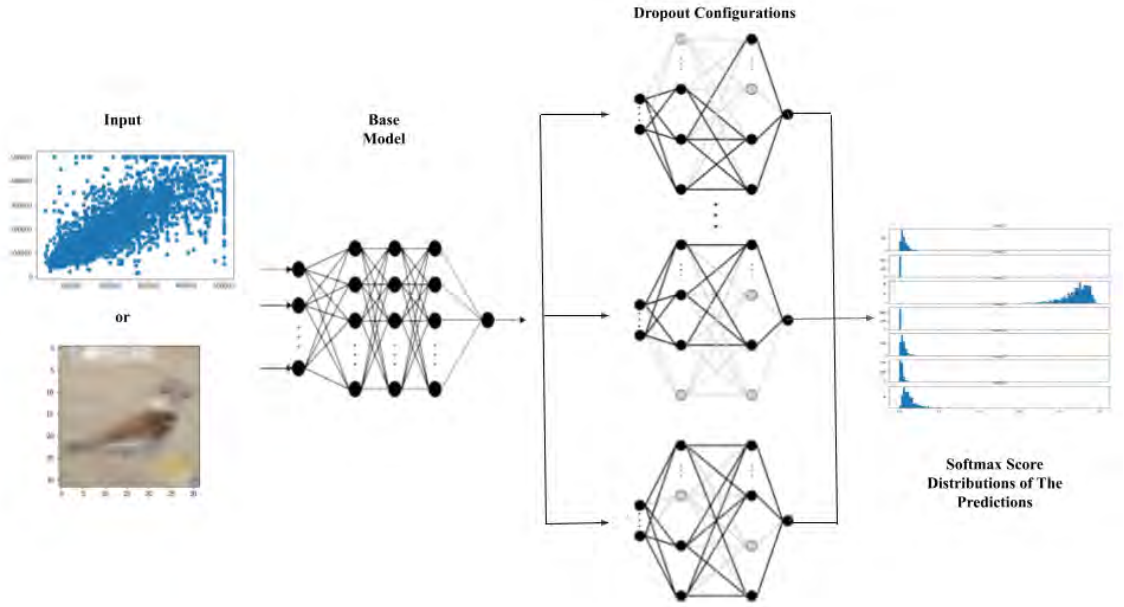


Figure 5.3: How Monte Carlo Dropout Works

Authors that were responsible for the introduction of MCD demonstrate uncertainty analysis and measurement in tasks including regression, classification, and reinforcement learning. In this investigation, we are particularly interested in regression and classification problems. In our thesis of MCD for uncertainty analysis, we make use of both well-known and state-of-the-art NN architectures.

5.1.3 How Do We Measure Uncertainty with Monte Carlo Dropout

Regression

For uncertainty representation from regression, we look at the MCD prediction plot. The way of interpreting uncertainty in regression tasks is to look at the predictions spread. When comparing multiple plots, if one prediction plot is more spread than the other, it means, the model is more uncertain. We put more importance on vertical spread. This is a clear example of aleatoric uncertainty. Horizontal spread usually refers to epistemic uncertainty.

Classification

We locate the most uncertain cases. This will be beneficial for comprehending our dataset or identifying problematic areas of the model. We select the most uncertain samples from the monte carlo prediction, using the variance of their softmax score. The accuracy of the Monte Carlo ensemble is a representation of how well the model performs in terms of uncertainty. The greater the accuracy of the Monte Carlo ensemble, the more the confidence one may have in the model.
When comes to data-wise uncertainty analysis we use predictive entropy. The predictive entropy is utilized for evaluating the model uncertainty on a specific image.

An uncertain sample is selected, and the predictive entropy relays how "surprised" the model is to see the particular image. The model is said to be sure about its prediction's accuracy if the value is "low". Similarly, a "high" value insinuates that the model is uncertain about the image.

$$Entropy,\ H \approx -\sum_{c}^{C}(\mu_c)log(\mu_c) \tag{5.4}$$

We calculate entropy using (5.4) where, $\mu_c = \frac{1}{N}\sum_n {p_c}^n$ is the class-wise mean softmax score.By computing the variance of the anticipated softmax score, we choose the images. Softmax turns the actual values into probabilities, therefore the score we get are simply probabilities. Additionally, the image indexes are sorted to locate an uncertain sample from the test set of data. We compare the uncertainty by taking a random sample from the test dataset to see how well the model performs with a new meaningful data. Moreover, to determine the model's uncertainty of an out-of-scope input, we randomly create a noise image and show the predictive behaviour.
Figure 5.4 is a probability distribution graph. Here is an example of binary classification softmax score/probability distribution. The model predicts class 1 as the output. In the figure, x-axis is the softmax score distribution and y-axis is the number of samples. Meaning, how many samples have provided a particular softmax score. In a correct prediction we want most of the bars in the right most position (comparing to the other class). Also, the number of samples has to be as high as possible. In simple words, we want the bars to be in the right most position and the height or the value of y-axis to be as high as possible. Also, we want it to be very thin, meaning most of the predictions are indicating the same outcome. But in wrong predictions, we want the opposite. This indicates that the model is uncertain about its prediction and necessary steps should be taken to avoid this low level of confidence.

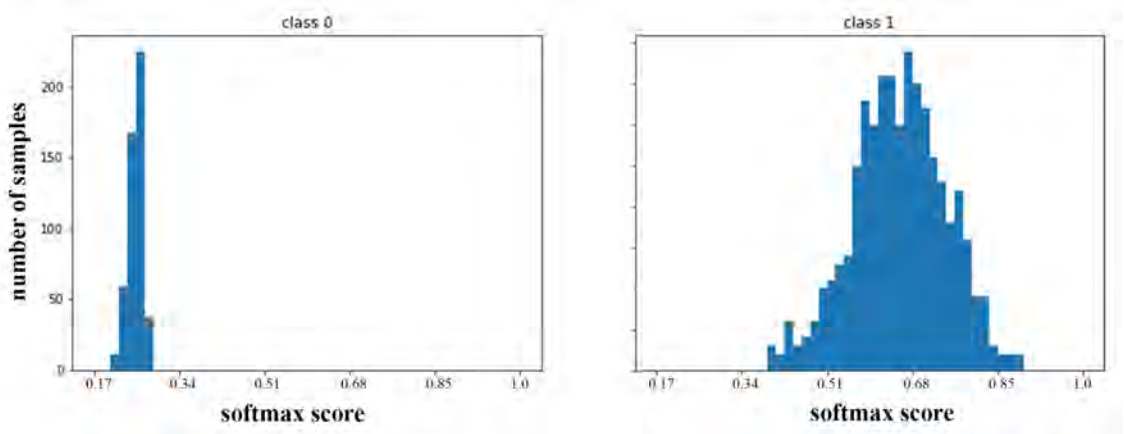


Figure 5.4: How to Analyze Classification Uncertainty

This is how we examine uncertainty sample by sample, which is a vital step whenever we are working with risky applications. In this research, we demonstrate prediction on a sample-by-sample basis for every classification task.

5.2 Neural Network Models

We will discuss about all of the NN models used in our study. We explain the neural network architecture and describe our development and modifications of the NN models.

5.2.1 Artificial Neural Network

ANNs, a specific kind of predictive model, can process data and make forecasts accordingly. With the help of accessible research and statistical data, they are intended to gain knowledge and programmed to perform data categorization, envision of results and facilitate inference. Comparable to the surface response approach, an inculcated Artificial Neural Network maps input parameters to a specific output and generates frequent consistent outcomes than traditional arithmetical analytic methods, for example regression analysis, while requiring substantially less computer work. [94]–[108]. ANNs operate in a manner analogous to that of the biological network of neurons. [109]–[121]. Synthetic nerve cell is considered as ANN's building block and a quantitative model that mimics the behaviour of a biological neuron. The input is transmitted via synthetic nerve cell and output is generated after nonlinear function processing. Additionally, weights are added to the input parameters prior to their arrival at the neuron to imitate the stochastic character of the biological nerve cell. In order to construct an Artificial Neural Network, three major processes are necessary: First, building ANN's structure; second, defining preparation procedure necessary for the ANN's development stage; and third, identifying the arithmetical functions that define the quantitative model. The training phase of an artificial neural network plays significant decision making role since it selects best weight parameters and reduces loss function. Consequently, diverse artificial NNs adopt diverse enhancement methodologies. Each neuron's behavior is defined by the summation and activation functions, which are mathematical functions.

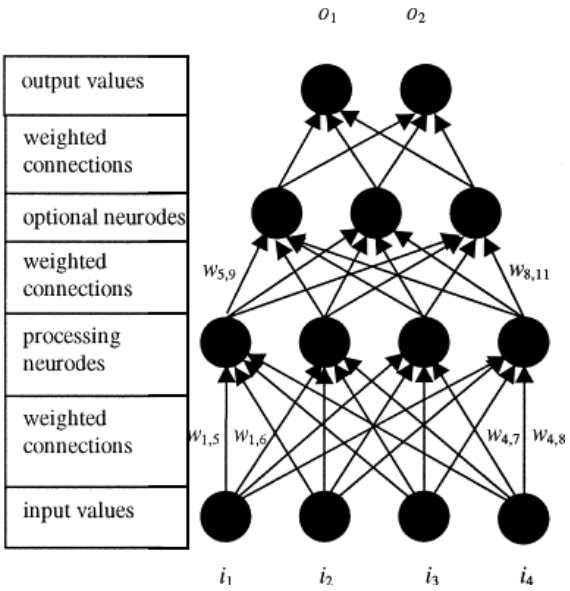


Figure 5.5: ANN Diagram

Layer	Output Shape	Parameters
Dense	(None, 100)	200
MCD	(None, 100)	0
Dense	(None, 200)	20200
MCD	(None, 200)	0
Dense	(None, 200)	40200
MCD	(None, 200)	0
Dense	(None, 100)	20100
MCD	(None, 100)	0
Dense	(None, 1)	101
Total		80,801

Table 5.1: Model Summary of Lightweight MCD Model

Layer	Output Shape	Parameters
Dense	(None, 1000)	2000
MCD	(None, 1000)	0
Dense	(None, 1000)	1001000
MCD	(None, 1000)	0
Dense	(None, 2000)	2002000
MCD	(None, 2000)	0
Dense	(None, 2000)	4002000
MCD	(None, 2000)	0
Dense	(None, 1000)	201000
MCD	(None, 1000)	0
Dense	(None, 1)	1001
Total		9,009,001

Table 5.2: Model Summary of Heavyweight MCD Model

5.2.2 ANN models for Regression

For our regression task, we utilize two ANN variants, where one of the model is lightweight and the other one is heavyweight. As Table 5.1 represents, the lightweight model contains 4 layers with 100 nodes in each and another layer as the classifier layer. On ther other hand, the heavyweight model has 5 layers with 1000 nodes in each. Table 5.2 shows the full architecture of the heavyweight model. In both of the ANN models, we use 40% dropout rate, which is for extracting variations in predictions.

5.2.3 Convolutional Neural Networks

Animals' image perception is a remarkable process and a simple one for them. Nevertheless, there are several underlying complexities in the process by which a machine grasps a picture. What animals see is the image collected by their eyes, which is subsequently carried to the cerebral cortex for decoding by neurons. The central nervous system of animal brains[122] was the foundation for the creation of the Convolutional Neural Network (CNN), a machine learning architecture that attempts

to simulate the ocular characteristics of animals. In terms of picture comprehension tasks such as classification, segmentation, detection, localization and so on, the convolutional neural architecture offers a significant improvement. Convolutional NNs are widely used because of their impressive picture understanding capabilities. CNNs are constructed using convolutions and contain biases and weights that may be learnt and are akin to human neurons. CNNs are made up of four main parts: fully-connected layers, convolutional layers, activation functions, and pooling. This paper offers a very concise introduction to CNNs, as described in [123]. The region of the mammalian brain responsible for sight is composed of neurons that isolate image characteristics. Each brain cell extracts unique features, facilitating the comprehension of pictures. The convolutional layer is modeled atop nerve cells in order to retrieve attributes such as colors, edges, textures, and gradient direction. Learning-capable convolutional filters, also called kernels, take the form $n \times m \times d$ where d denotes thickness of image. The forward propagation involves convolution operation, that is the kernels are mapped over the height and width of the input matrix. The inner product of the input with the filter components is computed afterwards. CNN automatically detects kernels which are triggered by edges, colors, textures, etc. A layer of activation function receives the hidden layer's convoluted output, which creates non-linearity and produces classification results.
Let us take into account an input tensor denoted by the letter X, which has the dimensions H, W, and C_{in}. To begin, we have a set of convolution kernels with C_{out} that are of the form K, K, C_{in}. Then, multiplication-addition operation is executed on the input tensor and the kernels result in the production of an output tensor denoted by Y that has the dimensions H, W, and C_{out}.

In Figure 5.6 above $C_{out} = 3$. Now we have the shapes H, W, and 3 for the output tensor. Since the convolution kernel does not take into account the location of the input tensor, it is considered to be geographically independent. However, the output vector is unique for every channel due to the fact that the convolution filter used to generate each channel is not the same.

Proposed Model for ECG Trace Image Classification

An image with a resolution of 70×70 pixels serves as the basis for our CNN model's initialization process. Subsequently, we employ a network of six 2D convolutional layers, each with a 3×3 kernel size. As an additional measure against model overfitting, we placed a batch normalization layer following every layer of convolution. We utilize batch normalization since it permits each layer in the network to learn more autonomously, allowing us to increase the learning rate. Furthermore, to lessen the computation cost, we implement a max pooling layer following every layer of batch normalization which has a pool size of 2×2. We employ the default strides for all of the convolutional layers, which are (1,1). We implement Relu as the activation function because, unlike the Sigmoid/Tanh function, its gradient is not saturated, speeding up the convergence of stochastic gradient descent (SGD). Finally, we utilize a 20% dropout from the aforementioned layer once we have flattened the values into 1 dimension. Our suggested model incorporates dropout to increase its robustness and eliminate any trivial relationships among the neurons. The dropout rate is fixed according to our experiments, which suggests that 20% dropout rate provides the best raw performance along with reliability. Additionally, we implement the

Table 5.3: Number of Parameters and Layer Details of Proposed CNN

Layer	Output Shape	Parameter
2D Convolution Layer	(None, 70, 70, 16)	160
Batch Normalization	(None, 70, 70, 16)	64
2D Max Pooling	(None, 35, 35, 16)	0
2D Convolution Layer	(None, 35, 35, 32)	4640
Batch Normalization	(None, 35, 35, 32)	128
2D Max Pooling	(None, 17, 17, 32)	0
2D Convolution Layer	(None, 17, 17, 64)	18496
Batch Normalization	(None, 17, 17, 64)	256
2D Max Pooling	(None, 8, 8, 64)	0
2D Convolution Layer	(None, 8, 8, 96)	55392
Batch Normalization	(None, 8, 8, 96)	384
2D Max Pooling	(None, 4, 4, 96)	0
2D Convolution Layer	(None, 4, 4, 128)	110728
Batch Normalization	(None, 4, 4, 128)	512
2D Max Pooling	(None, 2, 2, 128)	0
Monte Carlo Dropout	(None, 2, 2, 128)	0
2D Convolution Layer	(None, 2, 2, 256)	295168
Batch Normalization	(None, 2, 2, 256)	1024
2D Max Pooling	(None, 1, 1, 256)	0
Monte Carlo Dropout	(None, 1, 1, 256)	0
Flatten	(None, 256)	0
Monte Carlo Dropout	(None, 256)	0
Dense	(None, 5)	1285
Total parameters: 488,229 Trainable parameters: 487,045 Non-trainable parameters: 1,184		

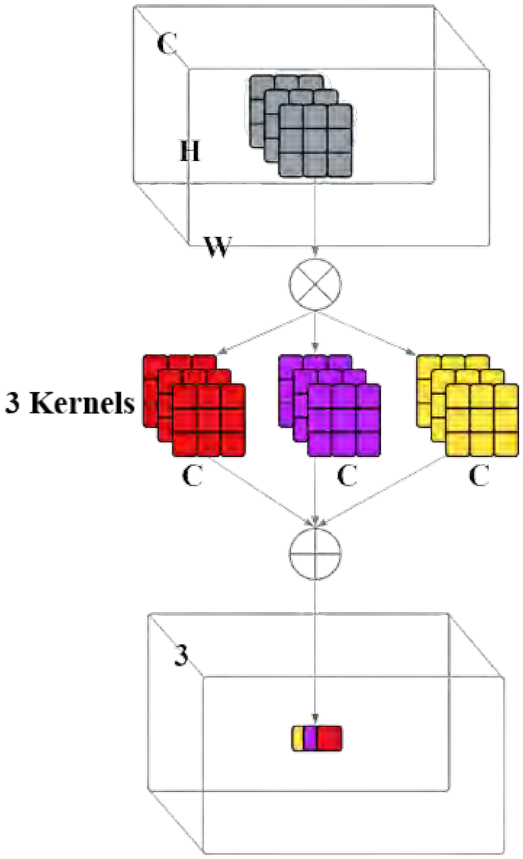


Figure 5.6: CNN Diagram

MCD layer as the dropout layer itself. Batch normalization and dropout are both methods of regularization, however they serve other purposes as well. When the model is being trained, dropout has an influence on the standard deviation of the distribution. However, when the model is being validated, dropout has no effect on the distribution. In other words, batch normalization does not eliminate any data but dropout does. Since all nodes have been classified at this stage, the softmax activation function is applied to the final result. The data that is passed into the function is converted into a value that falls between 0 and 1. In order to accomplish the same task again, we make use of MCD in the final two convolutional blocks. The model incorporates a total of 488229 parameters. Our proposed architecture is illustrated in Table 5.3.

5.2.4 Involutional Neural Networks

Most state-of-the-art computer vision neural networks are built on a framework of convolution. Being both spatially agnostic and channel-specific, convolution kernels are used in signal processing. Due to this, it has trouble adapting to unique visual patterns in relation to new physical locations. As a result of both locational and receptive-area-of-convolution-related challenges, recording lengthy spatial connections is a complex task.

Reassessing the convolution's key features might help in preventing the previously mentioned complications [124]. The "Involution kernel" is proposed by the authors as it is both channel-agnostic and may be used in any given location. The authors contend that self-attention is an instance of the Involution design paradigm due to the geographically specific nature of the operation.

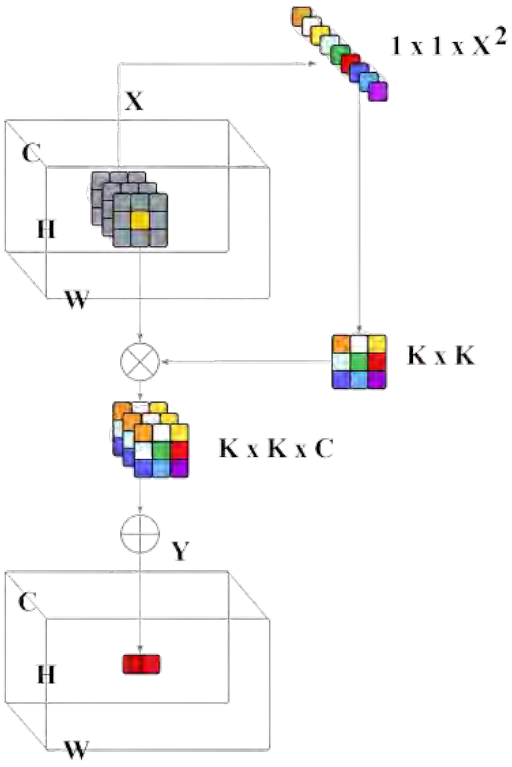


Figure 5.7: INN Diagram

Comprehension of convolution is a prerequisite for the firm understanding of the concept of involutions. Tensor X with dimensions H, W, and Cin is the input. In the beginning, we have a collection of K, K, Cin form of $Cout$ convolution kernels. Multiplying and adding the input tensor with the kernels yields a tensor Y with the dimensions H, W, and C_out.

In Figure 5.7, $Cout = 3$ resulting in a tensor output of the form H, W, and 3. It is also notable that the geographical location of the input tensor has no effect on the convolution kernel, making it location agnostic. On the other hand, since each channel is based on a unique convolution filter, the resulting tensor is channel-specific.

The goal is to devise a mechanism that is both geographically specific and channel-agnostic. It's not easy to develop those qualities. Since the resolution of the input tensors varies, a fixed number of Involution kernels cannot be used to process them (one for each spatial point).

The authors suggest that this problem be resolved by basing the creation of each kernel on a set of spatial coordinates. This approach should simplify the handling

of input tensors with varying degrees of resolution. Figure 2 depicts this method of kernel creation. Here, we generate K, K, C filters, where C is the total number of subchannels. We generate C filters and send each of them to each of the C input channels instead of utilizing a singular filter and sending it to all of the channels at once.

Proposed UnIC-Net Model for Skin Cancer and Malaria Classification

One involution layer is used at the beginning of our proposed hybrid model, and then the ReLU activation function is applied. Subsequently, we implant 2 convolution layers powered by ReLU activation functions. Following the involution and convolution blocks, a dropout layer with a rate of 15% is added. Our investigations reveal that a dropout rate of 15% offers the best raw performance together with the highest level of dependability. Accordingly, we have decided to keep the dropout rate at this level. Any other rate either provide very poor performance or lackings in the reliability issue. In the final step, we convert all of the information that we retrieved into one-dimensional data using the fully connected block. It is composed of 6 dense layers with a total of 256, 128, 96, 64, 32, 16 nodes. The reason of using this particular number of dense layers with the respective number of nodes is, the utilized data requires multiple dense layers to extract all features properly. This is also visible in our preliminary experiments. Using less number of layers and nodes gives less accuracy and giving more increases the problem of overfitting. Softmax is used as the activation function for the classifier layer. The weight parameters in our model totals at 3,324,861. The overview of our model is included in Table 5.4.

5.2.5 Recurrent Neural Networks

The neural networks' capacity to instantly self-learn and self-adapt allows them to model and predict complicated nonlinear patterns given a series of input points with predefined outputs. RNNs are one such type of neural network that creates cycles as a result of one or more connections between the neurons. These cycles in RNN store the data and transmit feedback from one neuron to another. This method creates an internal memory and makes it easier to learn sequential facts. A RNN's usage of loops enables information to be carried while input is being read. They are distinct from other neural networks because their memory enables them to identify relationships between events. Theoretically, these loops can be deployed everywhere in the network and in any direction [125]. Figure 5.8 represents the internal structure of RNN.
Both LSTM and GRU are common RNN subtypes used in sequential data.

Long Short-Term Memory & Gated Recurrent Unit

Since its first introduction in 1997 by Sepp Hochreiter and Jurgen Schmidhuber, LSTM has been extensively employed and has given rise to several versions. Input gates and forget gates are added to LSTM in comparison to conventional RNNs to address the gradient disappearance and explosion issues. This allows for the collection of long-term information and improved performance in lengthy sequence text. GRU's internal structure is comparable to LSTM, while its input and output structure is identical to that of a standard RNN [126]. Figure 5.9 and Figure 5.10 below

Table 5.4: Number of Parameters and Layer Details of Proposed UnIC-Net (1 layer Involution)

Layer	Output Shape	Parameter
Input Layer	[(None, 28, 28, 3)]	0
Involution Layer	((None, 28, 28, 3), (None, 28, 28, 9, 1, 1))	43
ReLU	(None, 28, 28, 3)	0
Batch Normalization	(None, 28, 28, 3)	12
ReLU	(None, 28, 28, 3)	0
2D Convolution Layer	(None, 26, 26, 64)	1792
2D Convolution Layer	(None, 24, 24, 64)	36928
Batch Normalization	(None, 24, 24, 64)	256
2D Convolution Layer	(None, 22, 22, 64)	36928
Batch Normalization	(None, 22, 22, 64)	256
2D Max Pooling	(None, 11, 11, 64)	0
Monte Carlo Dropout	(None, 11, 11, 64)	0
Flatten	(None, 7744)	0
Dense	(None, 256)	1982720
Dense	(None, 128)	32896
Dense	(None, 96)	12384
Dense	(None, 64)	6208
Dense	(None, 32)	2080
Dense	(None, 16)	528
Dense	(None, 2 or 7)	34
Total parameters: 2,113,065 Trainable parameters: 2,112,799 Non-trainable parameters: 266		

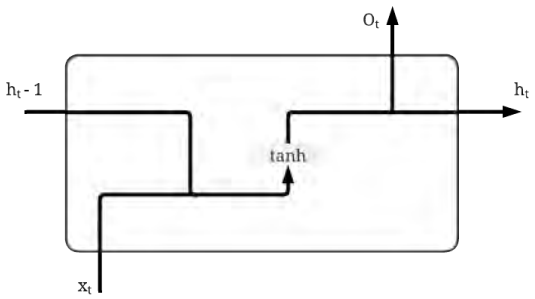


Figure 5.8: RNN Diagram

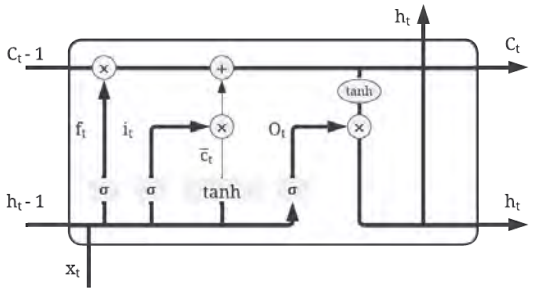


Figure 5.9: LSTM Diagram

illustrate a comparison of the internal structures of LSTM and GRU. Information cannot be encoded from back to front using either LSTM or GRU. In situations with greater categorization granularity, like the five-category jobs of strong and weak commendatory terms, weak disparaging term, and neutrality, the interplay amongst words of degree, emotion negativity words should be considered. This problem is overcome with a device known as a bi-directional long-short-term memory (Bi-LSTM), as well as a bi-directional gate recurrent unit (Bi-GRU). Forward and backward LSTMs or GRUs are stacked in order to improve the bidirectional semantic dependence. Bi-LSTM or Bi-GRU are frequently superior to LSTM or GRU, however training will take longer.
In our research, we have compared stacked Bi-LSTM and stacked Bi-GRU models to rather strong transformer models; hence, we have employed stacked versions of both of these models. The model is comprised of two Bi-LSTM layers, and the embeddings layer is where it all begins. After the classifier layer comes the MCD layer, which is inserted just before it. In a similar manner, the GRU model consists of 2 Bi-GRU layers, and the other parts of the model are the same as those found in the LSTM model. Both of them possess MCD layers that have a dropout rate of 30%. We use 30% dropout rate because all of the utilized models tends to provide optimal results with this rate, which will be a fair evaluation in our experiments.

5.2.6 Transformers

In practice, it is simpler to express the relationship between each pair of words using a fully connected graph and then to allow the model to determine the structure on its

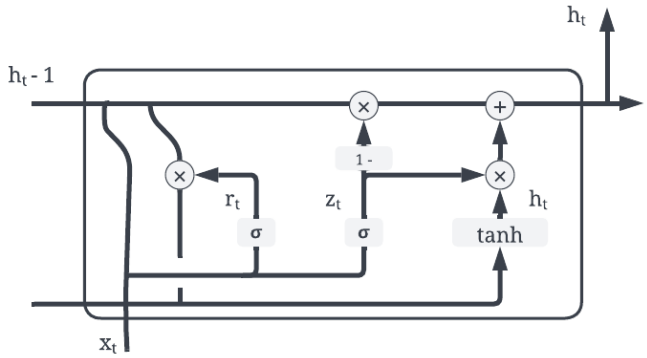


Figure 5.10: GRU Diagram

own. Frequently, the self-attention mechanism dynamically calculates link weights, inferring the relationship between words. Figure 5.11 shows the self-attention mechanism. An effective fully-connected self-attention model is the Transformer [127]. Moreover, it necessitates extra modules, including residual connections, positional embeddings, layer normalization, and position-wise feed-forward network (FFN) layers. Because they learn the contextual representation of a word with a localization bias, sequence models have difficulty capturing long-distance interactions between words. Figure 5.12 shows the encoding and decoding mechanism of transformers.

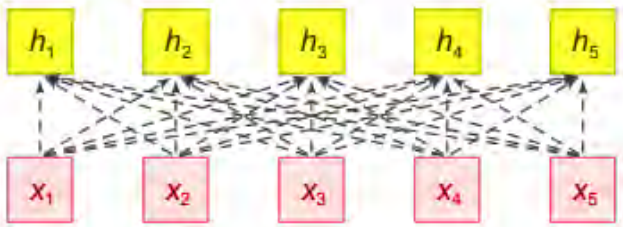


Figure 5.11: Self Attention Network

BERT

Simultaneously conditioning both the right and the left context in every layer, BERT [128] architecture is capable of pre-training deep bidirectional representations from unlabeled text. Thus, the BERT model simply requires a final output layer in order to produce popular models which are viable for handling a comprehensive amount of tasks.
These tasks range from answering questions to inferencing languages without having to majorly alter the architecture for a specific task. As we know, a directional model reads a given text either from right-to-left or left-to-right. In contrast, encoders in

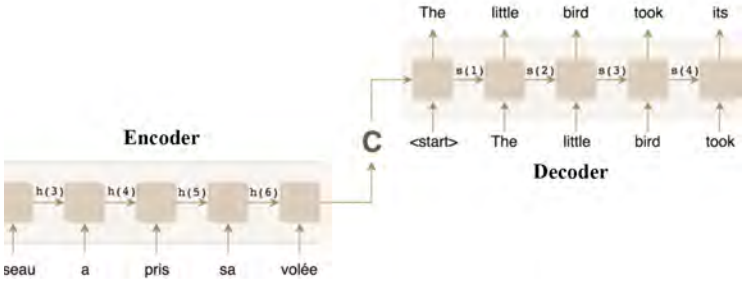


Figure 5.12: Transformer Encoder

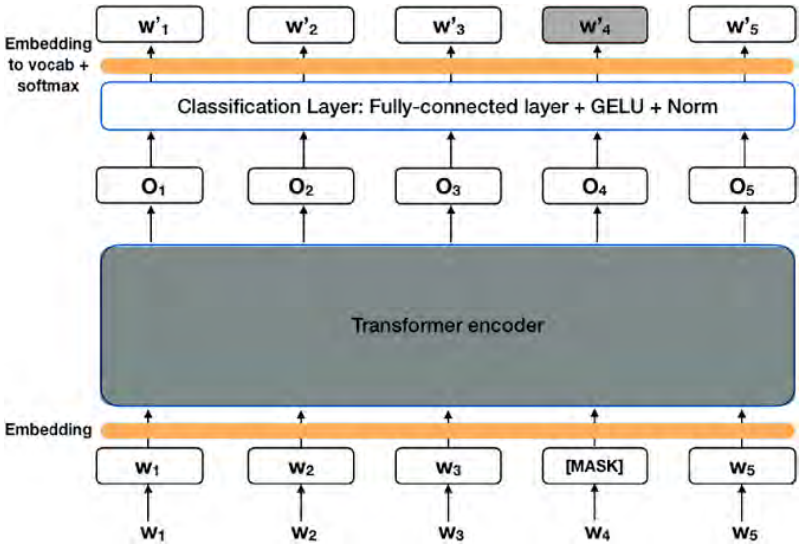


Figure 5.13: BERT Diagram

Transformer models scan the full string of words in one go, hence earning the name bidirectional or non-directional. This key attribute enables the model to learn all surrounding-based contexts of a particular word. Figure 5.13 illustrates the workflow of BERT.

XLNet

One major issue with BERT is essentially its pre-training objective on masked sequences i.e the Denoising Autoencoding objective. An autoregressive pretraining technique called XLNet[129] is a rather generic option for enabling the learning of bidirectional contexts. The task is accomplished via maximization of the anticipated likelihood over all permutations of the factorization order. Furthermore, the autoregressive formulation allows the system triumph over the drawbacks of BERT.It doesn't have the denoising of inputs as in the autoencoding objective and removes the unidirectionality from a traditional autoregressive objective. Figure 5.14 shows the mechanism of XLNet.

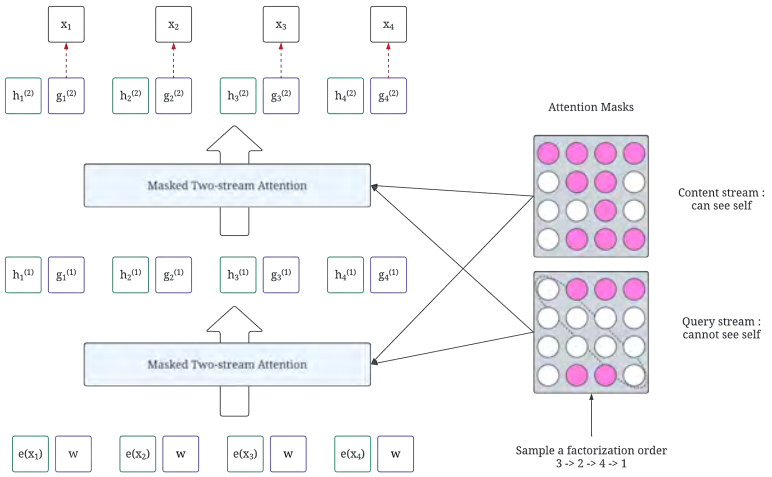


Figure 5.14: XLNet Diagram

Baseline Vision Transformer

However, sequence models can usually be trained easily and tend to perform well across a variety of NLP applications. Vision Transformer (ViT) [130] is the earliest research which illustrates how Transformers may "completely" take-over standard convolutions in DNNs on huge picture datasets. The initial Transformer model is employed [131] on a sequence of picture "patches". These patches were flattened as vectors with few modifications. The system was already trained on a substantial private dataset (JFT dataset [132] with 300 million pictures). It was later calibrated to conform to downstream recognition criteria, such as ImageNet classification. Transformers acquire information from exceptionally elaborate datasets whereas ViTs are pre-trained using a medium-range dataset. The two models would not yield comparable results. This is due to the fact that prior knowledge regarding the pictures (inductive biases, such as translation equivariance), is encoded by CNNs, which reduces the data requirements.
ViT were first introduced by Dosovitskiy et al. [133] in order to criticize the significance of CNN in attention-based models for vision-related tasks, as they were previously implemented with convolutions [134], [135], or at least some of its properties [136]. Despite the fact that self-attention based models had a number of benefits, such as extended range connections, there was still potential for advancements like scalability. As a result, Positional Embedding, Image Tokenization, the Transformer Encoder, Classification Token, and a Classification Head were the mechanics used to create ViT. The basic illustration of ViT can be seen in Figure 5.15.

Swin Transformer

According to Liu et al. [137], Swin Transformer (SWT), a modified version of ViT with a hierarchical design, is proposed. For rendering purpose, Shifted windows are integrated with the modification. This approach is strongly advised since it limits its self-attention evaluation to non-intersecting local windows, which improves the model's efficacy. Additionally, the method allows for connections between windows. As long as the computational cost is maintained linearly with regard to image size, this form of ViT's architecture allows for modeling at various calibrations. Hence,

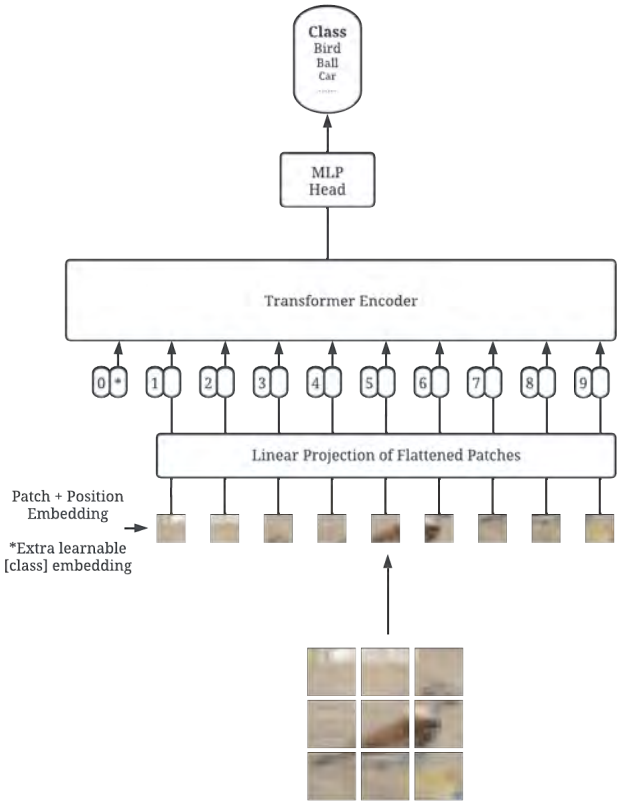


Figure 5.15: VIT Diagram

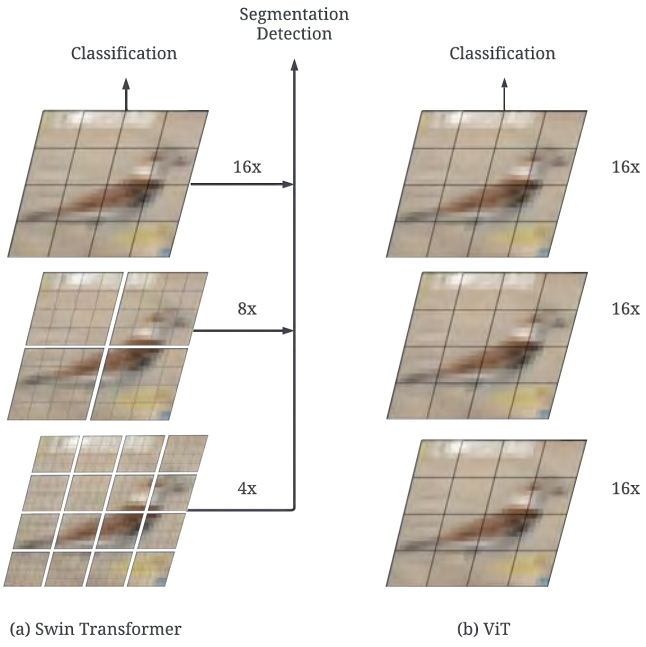


Figure 5.16: SWT Diagram

these qualities make SWT very adaptable to a variety of visual tasks. Figure 5.16 shows the key mechanism of SWT and the difference with ViT.

Compact Convolutional Transformer

Compact Convolutional Transform (CCT), which incorporates convolutions into transformer models, is the idea put out by Hassani et al [138]. The CCT technique improves inductive bias and eliminates the need for positional embeddings by using sequence pooling and patch embedding in place of convolutional embedding. One of the numerous benefits of the CCT is that input parameter flexibility is proliferated along with precision. Figure 5.17 shows the architectural design of CCT.

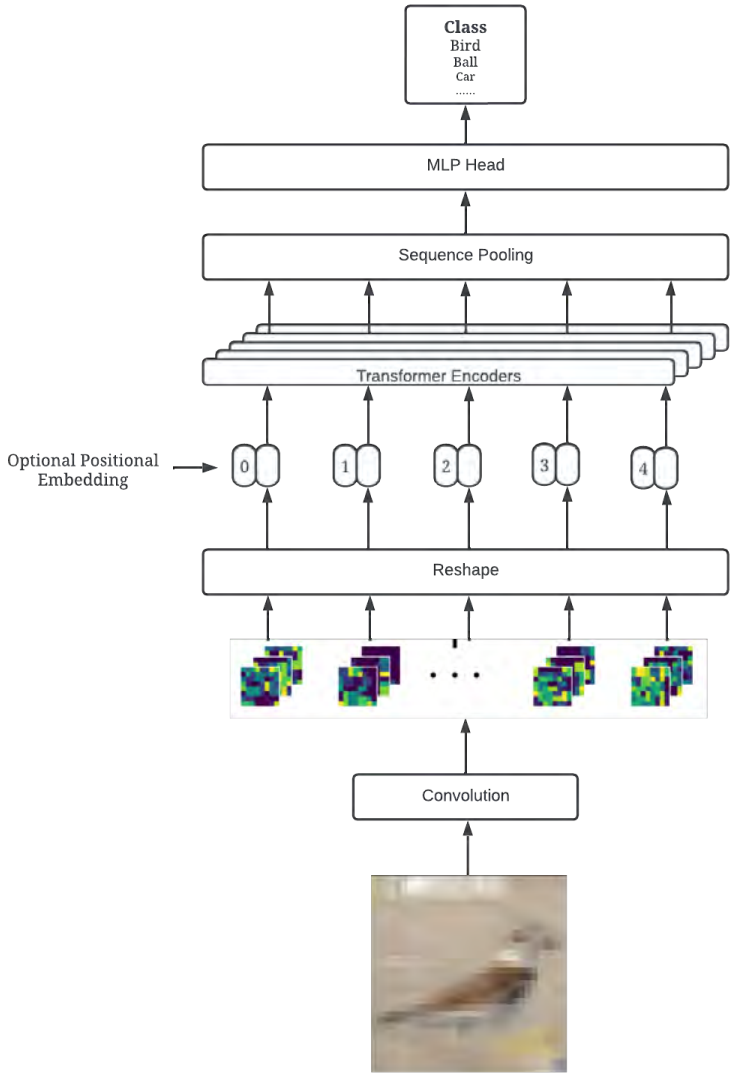


Figure 5.17: CCT Diagram

5.2.7 Graph Neural Network

Artificial neural networks like Graph Neural Networks (GNNs) can be used to make predictions on the node, edge, and graph levels. The concept behind graph neural networks is to calculate a state for every node in a graph, which is then repeatedly updated based on the states of nearby nodes. These models spread information and

provide node representations that can be "conscious" of the more extensive network structure because of layering[139] or recursive techniques[140]. In the past, using kernel functions to compute task-agnostic features was the most common method to deal with complicated structures. However, GNNs have lately gained favor since they can quickly and automatically extract pertinent characteristics from a graph. GNNs are more appealing since such kernels are non-adaptive and frequently computationally costly. Figure 5.18 is a basic representation of a GNN model.

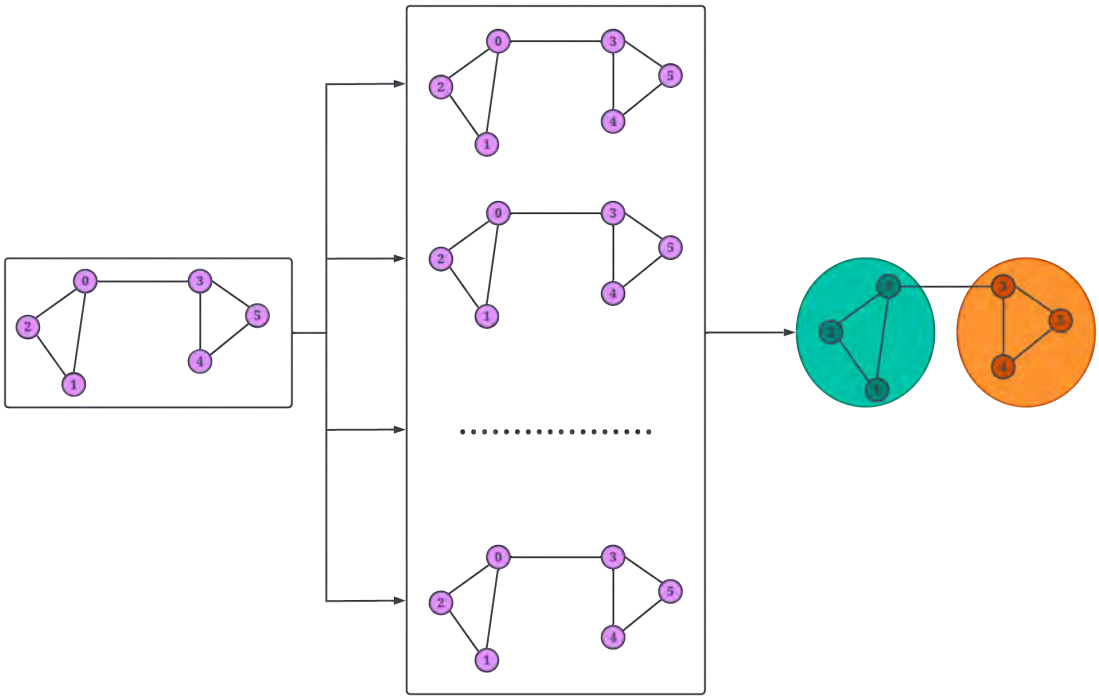


Figure 5.18: GNN Diagram

Graph Convolutional Neural Networks

The mathematical background of Graph Convolutional Neural Networks(GCNNs) is found in the fields of graph signal processing [141], [142] and spectral graph theory, where signal operations such as the Fourier transform and convolutions are generalized to signals existent on graphs.
In particular, Bruna et al. [143] and Henaff et al. [144] presented spectral graph theory, which gave rise to GCNNs. CNN-like parameterized filters may be created using GCNNs based on spectral graph theory but such filters are computationally costly, which makes them sluggish. Several studies have suggested employing Chebyshev polynomials or a first-order approximation of spectral graph convolutions to approximate smooth filters in the spectrum domain to circumvent the processing bottleneck of spectral GCNNs [145], [146]. Due to its quicker training durations and greater prediction accuracy, we adopt the GCNN formulation developed by Kipf and Welling [146] in this study. Figure 5.19 is a simple illustration of a GCN model.

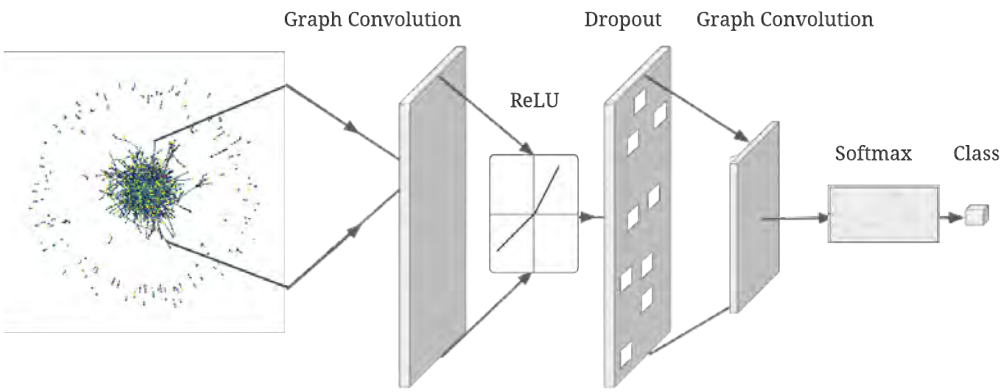


Figure 5.19: GCN Diagram

Graph Attention Networks

A SOTA neural network architecture, GAT (Graph attention network) [147] employs data in the form of graph. It leverages masked self-attention layers as a method of compensating for the drawbacks of any previous techniques which used convolutions or estimations of graphs. Nodes are explicitly attached with different weights in a neighborhood, the nodes are stacked in layers in order to attend to the neighboring features. No costly matrix operations are applied and no graph construction knowledge is applied. Our model is made relevant for inductive and transductive issues by taking into account many fundamental problems of spectral-based graph neural networks simultaneously.
Graph attention layers are the foundation of a GAT since they allow the model to prioritize different nodes in the graph when producing a prediction. As part of this process, a value is initialized to each node in the graph depending on its attention weight, which is then included into the final forecast.
The GAT's self-attention-based attention mechanism allows the model to prioritize different aspects of the input while producing a prediction. By applying the self-attention mechanism to the graph-structured data, GATs enable the model to assign relative priority to the various nodes and edges in the graph. The model is trained to learn a function that maps a node's local neighborhood to its own representation. To do this, the model applies a neighborhood sampling strategy to randomly select a fixed number of neighbors for each node. The selected neighbors' representations are then concatenated, and passed through a neural network architecture such as MLP to produce the final representation of the node. The output of the neural network can be passed through multiple layers, where each layer applies the same process of neighborhood sampling and aggregation. The final output of the last layer is used for the task at hand, for example, node classification. Figure 5.20 illustrates the key mechanism of GAT.

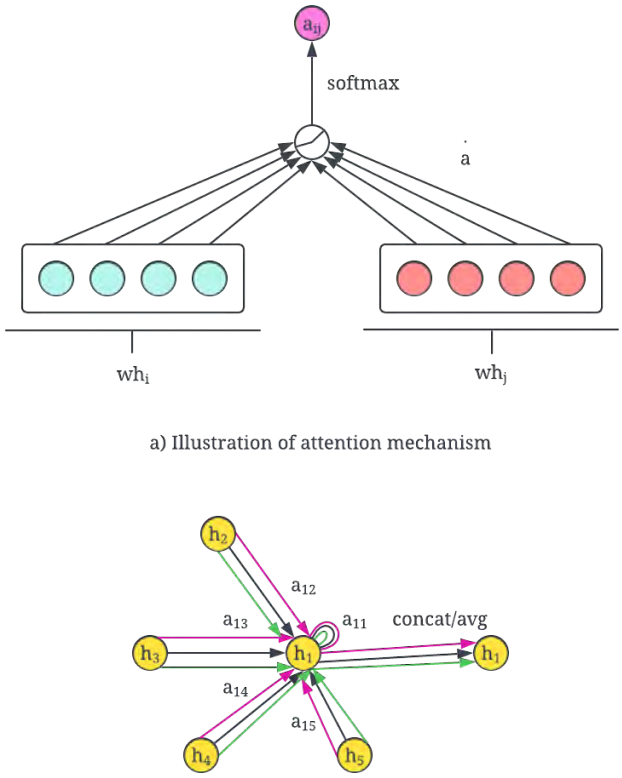


Figure 5.20: GAT Diagram

GraphSAGE

To efficiently construct node embeddings for unseen data, Hamilton et al. [148] propose GraphSAGE, a comprehensive, empirical technique that leverages node feature information. It accomplishes this by learning a function that generates embeddings through sampling and integrating neighborhood information rather than training separate embeddings for each node.
The model is trained to learn a function that maps a node's local neighborhood to its own representation. To do this, the model applies a neighborhood sampling strategy to randomly select a fixed number of neighbors for each node. The selected neighbors' representations are then concatenated, and passed through a neural network architecture such as MLP to produce the final representation of the node.
The output of the neural network can be passed through multiple layers, where each layer applies the same process of neighborhood sampling and aggregation. One interesting property of GraphSAGE is that we can train our model on one subset of the graph and apply this model on another subset of this graph. Figure 5.21 shows the main structure of GAT.

PPNP and APPNP

Gasteige et al. [149] exploit the connection between graph convolutional networks (GCN) and PageRank to develop a more effective propagation technique using indi-

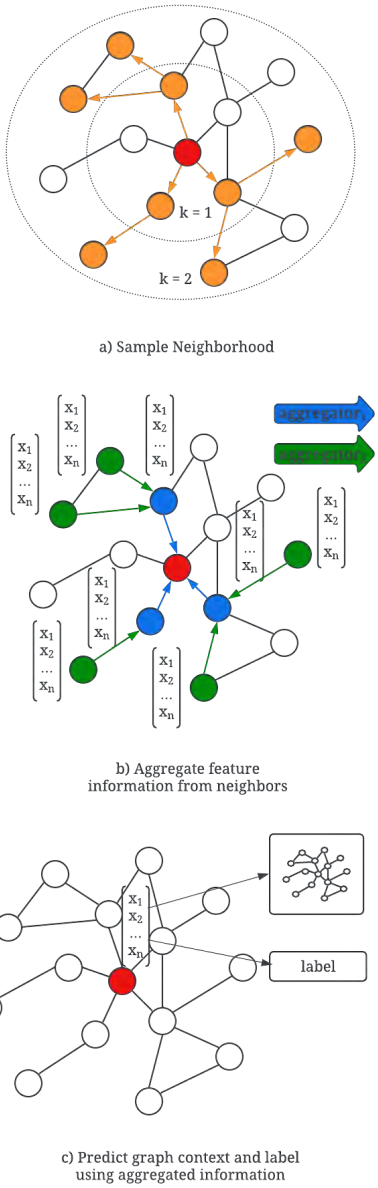


Figure 5.21: GraphSAGE Diagram

vidualized PageRank. Using this propagation method, they develop a basic model called personalized propagation of neural predictions (PPNP) and a speedy approximation called APPNP. It may be simply integrated with any existing neural network and makes use of a wide, malleable neighborhood for categorization.
PPNP is a GNN model that aims to improve the performance of node classification tasks by utilizing the graph structure of the data, it's based on the idea of personalized PageRank, where the model learns a personalized transition matrix for each node. The model learns a separate linear classifier for each node and uses the personalized transition matrix to propagate the classifier's predictions to the neighborhood of the node. PPNP has been shown to be accurate but it can be slow when training on large graphs.
APPNP (Approximate Personalized Propagation of Neural Predictions) is a faster approximation of PPNP. APPNP is a model which is similar to PPNP but it uses an

approximated personalized transition matrix which is computed by using a sparse approximation of the graph Laplacian. This allows the model to be faster to train and also it makes it more scalable to larger graphs. APPNP have been shown to be faster and more scalable than PPNP while still achieving similar or slightly better performance.
In the case of PPNP and APPNP, both models have been proposed for node classification tasks and have been shown to be effective in this task. PPNP is more accurate but APPNP is faster and more scalable.

Chapter 6

Experimental Analysis

6.1 Experimental Setup

The training and testing methods for this experiment are developed using Python libraries such as Tensorflow, and Keras. An NVIDIA RTX 3080TI GPU with 34.1 TeraFLOPS of performance is used to train and assess the models.

6.2 Evaluation Metrics

6.2.1 Regression

Commonly, two performance measures are kept in consideration while analyzing the prediction results to evaluate the performance of the proposed model: Mean Square Error (MSE), and Root Mean Square Error (RMSE). The metrics are represented as follows:

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(F_p - F_t)^2 \tag{6.1}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(F_p - F_t)^2} \tag{6.2}$$

where F_p is the predicted value and F_t is the actual value.

6.2.2 Classification

Several performance evaluation metrics such as accuracy, precision, recall, loss, F1 score, AUC are used to compare and validate the model's performance. Although accuracy is the most common metric used in classification tasks, we used several metrics to evaluate our model from different perspectives. Different evaluation metrics used in this research can be expressed using the following equations:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6.3}$$

$$Precision = \frac{TP}{TP + FP} \tag{6.4}$$

$$Recall = \frac{TP}{TP + FN} \tag{6.5}$$

$$Precision = \frac{TP}{TP + FP} \tag{6.6}$$

$$F1\,Score = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{6.7}$$

Here, TP and TN represent the data points that are correctly classified as true and false, respectively. On the other hand, FP and FN refer to the data points that the model inaccurately classifies as true or false. These are for binary classifications. For multi-class categorization, we go for micro and macro averaging. The scores obtained by macro-averaging are the arithmetic mean of the scores obtained by separate classes with regard to precision, recall, and F1-score. Precision scores are calculated by taking the total number of true positives for each particular class and dividing that number by the total number of expected positives for all classes. In most of our multi-class categorizations, we use the macro averaging. Only for the skin cancer identification task, we prefer, micro averaging since there is an imbalance

6.3 Model Performance and Uncertainty Estimation Results

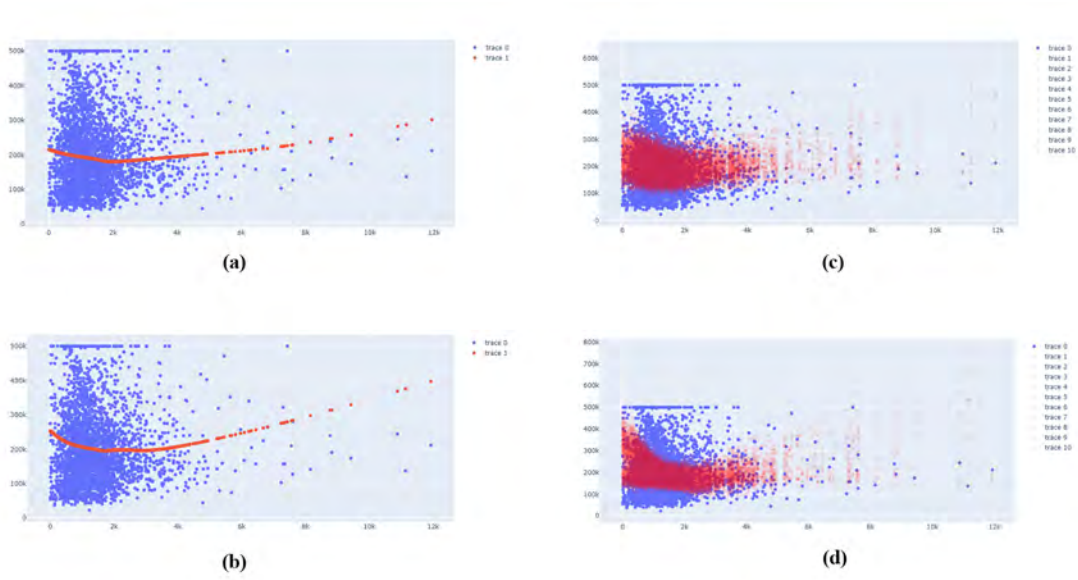


Figure 6.1: Selected Feature: Population; a) Lightweight Regular model b) Heavyweight Regular model c) Lightweight MC model d) Heavyweight MC model

6.3.1 Regression

ANN in Regression

As the goal and motivation of this experiment is to represent uncertainty, we do not focus on the evaluation metrics such as Accuracy, Mean Squared Error etc. The

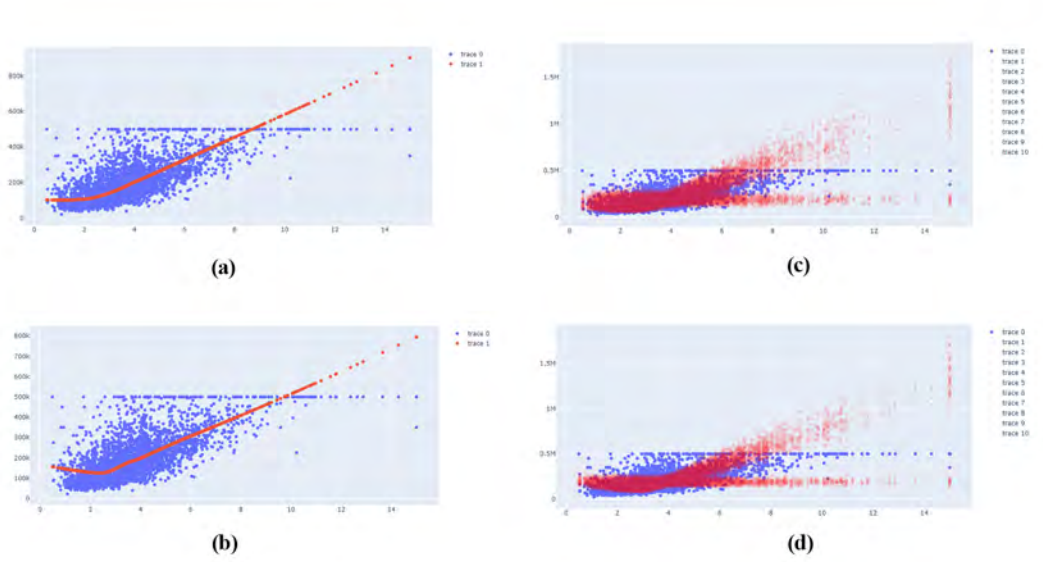


Figure 6.2: Selected Feature: Median Income; a) Lightweight Regular model b) Heavyweight Regular model c) Lightweight MC model d) Heavyweight MC model

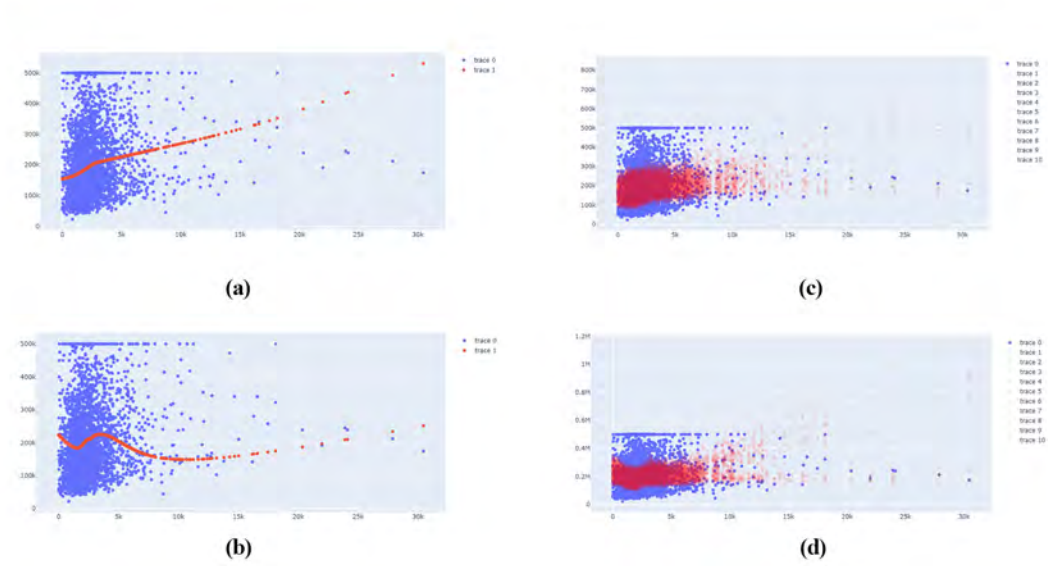


Figure 6.3: Selected Feature: Total Rooms; a) Lightweight Regular model b) Heavyweight Regular model c) Lightweight MC model d) Heavyweight MC model

models were created to show the difference of uncertainty between two different models and not to achieve high performance. In Table 6.1, we show the epochs taken to train each model in different scenarios.
In Figure 6.1 , we select the feature “Population”. We get two separate results here. Taking decisions from the regular prediction is risky and tough, as we can no be sure which one is the better fit. But when we look at our MC model representations, we can be sure that the heavyweight model is the better model as the model is less spread and more packed comparing to the lightweight MC model. In Figure 6.2, we select the “Median Income” feature. In this case the predictions look pretty similar

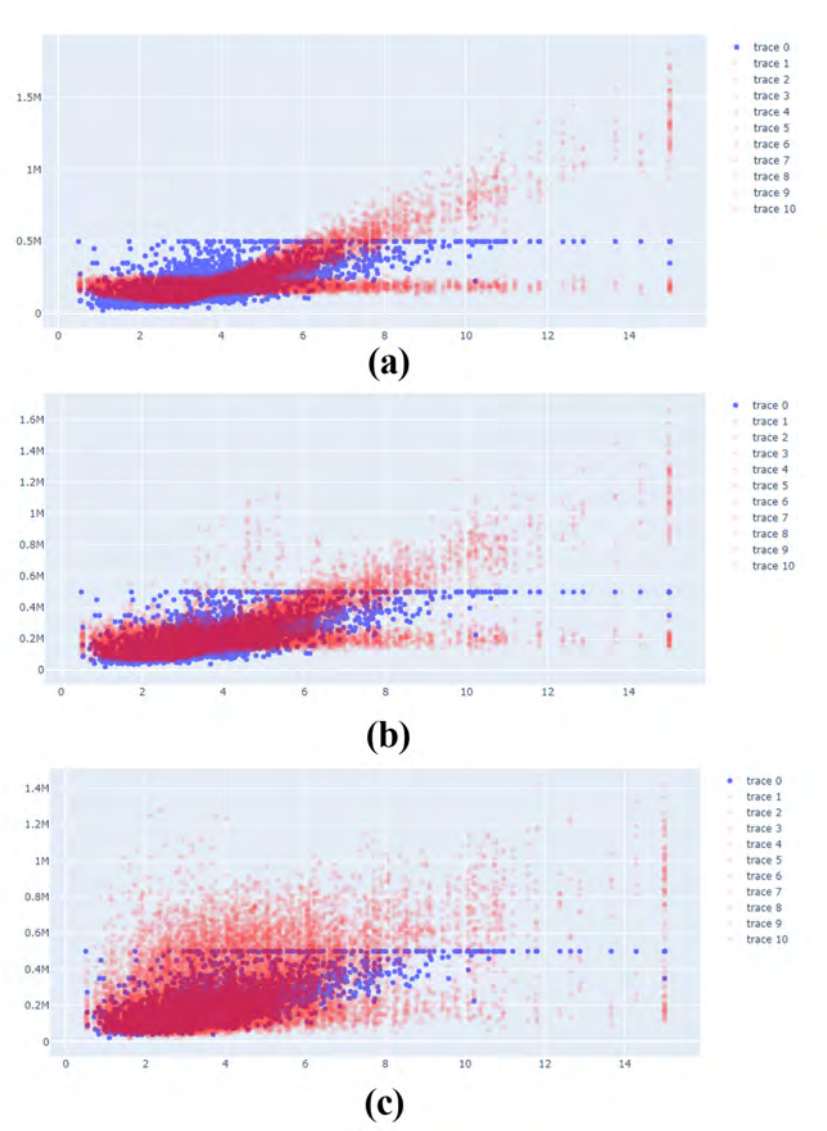


Figure 6.4: One Feature (a) vs Multiple Features (b) vs All Features (c)

but the uncertainty graphs gives us a more concrete visualization of the strength of these two models. In Figure 6.3, the decisions are pretty similar until we increase the input value. In this case, the model gets very uncertain when the input value gets high. This is a very helpful representation for us, as we can not get these information using regular neural networks.

Now, in Figure 6.4, we use different number of features to train our model and the results we get are quite self explanatory here. In Figure 6.4 (a) the model is quite confident with one feature (which is a very important feature for this dataset). The distribution looks more packed than the other two. In Figure 6.4 (b) the distribution is pretty similar but more wide and spread. When we use all features to train our model, the model gets very uncertain as there are unimportant features in the dataset and multiple features can add unwanted factors to the model. In Figure 6.4 (c) we get to see the result of taking all features.

The results lead us to conclude that representing uncertainty considering different

Table 6.1: Number of epochs for different tests

Feature Selected	Lightweight Model	Heavyweight Model	Lightweight MC Model	Heavyweight MC Model
Population	23	12	25	11
Median Income	23	12	22	13
Total Rooms	16	23	32	21
Multiple	39	31	31	14
All	22	19	28	15

conditions can help us to take major decisions with less risks.

6.3.2 NLP Application Result Analysis

RNN Variants - LSTM and GRU in Text Classification

We use LSTM and GRU in our experiments. Both LSTM and GRU takes 10 epoch to complete the training process. The rate of learning is 0.0000006. LSTM has 59,009,325 parameters and GRU has 55,049,235 parameters. LSTM achieved 91.93% test accuracy and GRU got 88.43% test accuracy. We select a random text from the test sample, the text is shown in Figure 6.5.

"After waiting for almost 30 minutes to trade in an old phone part of the buy back program, our customer service rep incorrectly processed the transaction. This led to us waiting another 30 minutes for him to correct it. Don't visit this store if you want pleasant or good service."

Figure 6.5: Randomly selected text for RNN variants and Transformer uncertainty estimation

From Figure 6.6 we see that, LSTM and GRU are quite identical in terms of predictive certainty. LSTM has 0.63 and GRU has 0.71 entropy of it's softmax score. We compare it with the performance of BERT and XLNet next.

Transformers in Text Classification

We train the transformer models - BERT and XLNet with 5 epochs. We set the dropout rate to 30% in both models. For Monte Carlo sampling we have predicted the models 50 times with 50 samples. Both of them have MCD layers embedded after the post-processing layers of the moodels. In every experiment model, we set the learning rate at 0.0000006. The number of parameters for BERT, and XLNet are 109 Million, and 110 Million respectively. All tests have a fixed batch size of 64. From Figure 6.7 and Figure 6.8 we can see BERT has a better fit in the training phase. BERT has achieved 87.11% training accuracy and 86.88% validation accuracy. The test accuracy of the BERT model is 87.02%. It has 86.22% precision and 88.11% recall. On the other hand, XLNet did not perform as well as BERT, having only 68.17% test accuracy, 65.53% precision and 80.68% recall.
We use the same text (Figure 6.5) used in LSTM and GRU experiment for the out-of-distribution prediction to measure uncertainty. From Figure 6.9 we see how well BERT performed comparing to XLNet. Even though both of them correctly predicted the randomly selected test sample, BERT is more certain. BERT has

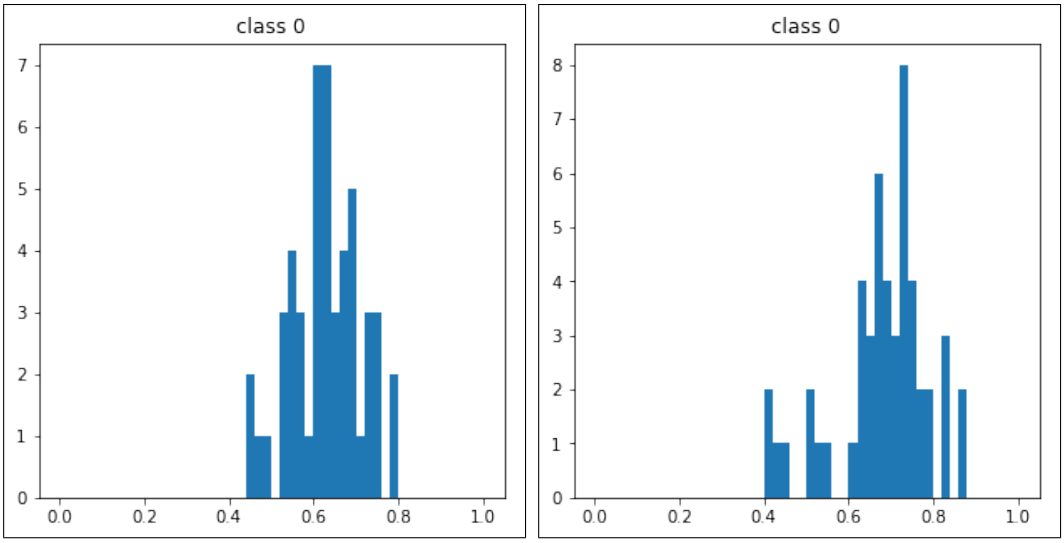


Figure 6.6: Probability Distributions of LSTM and GRU

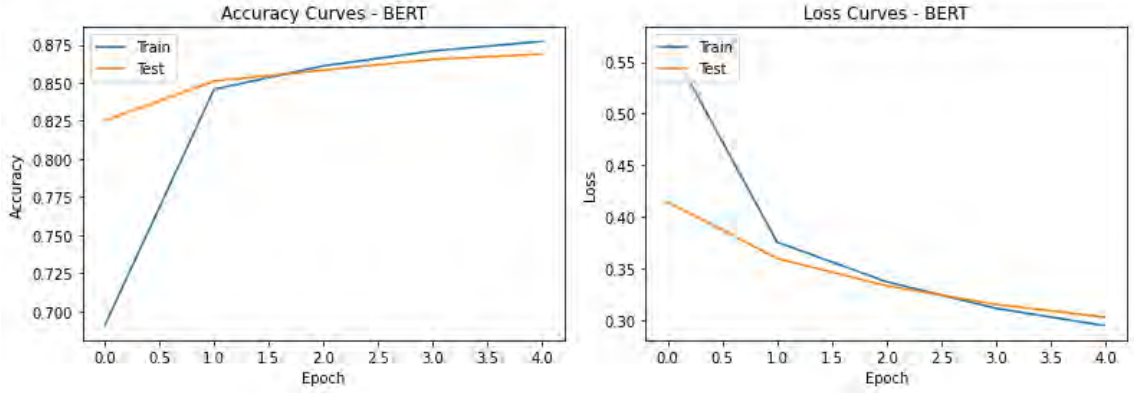


Figure 6.7: Train vs Validation Accuracy Curve and Train vs Validation Loss Curve of BERT-MCD

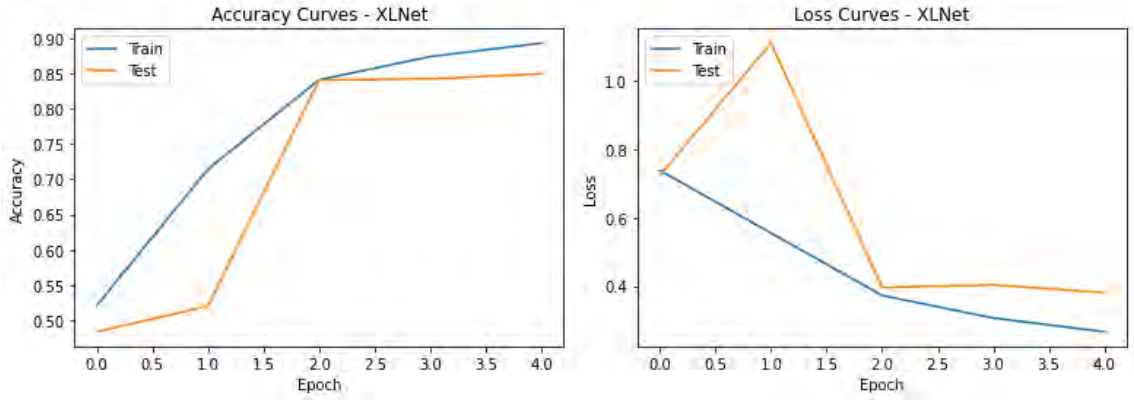


Figure 6.8: Train vs Validation Accuracy Curve and Train vs Validation Loss Curve of XLNet-MCD

0.31 entropy and XLNet has 0.85 entropy. From the results of RNN variants and Transformers, we can conclude that BERT outperforms all then LSTM, GRU and XLNet comes respectively.
For overall performance, BERT has 89.6% Monte Carlo ensemble accuracy. The rest

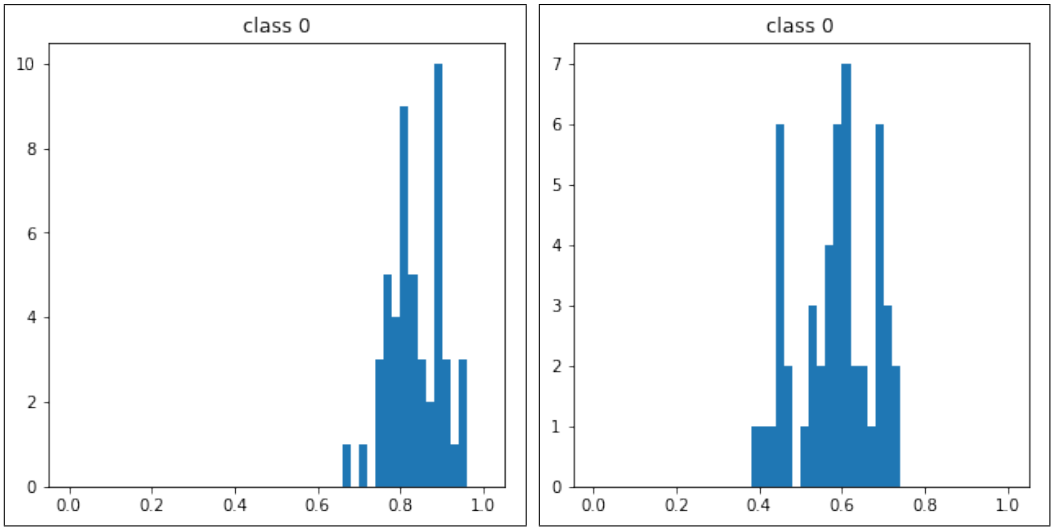


Figure 6.9: Probability Distributions of BERT and XLNet

Table 6.2: Results obtained from CNN-MCD method

Accuracy	Precision	Recall	Loss	F1 Score	AUC Score
93.9%	94.0%	93.0%	0.031	93.5%	97.6%

of the models, has 88.1% (LSTM), 86.3% (GRU), and 83.7% (XLNet).

6.3.3 Computer Vision Application Result Analysis

CNN in ECG Trace Image Classification

Metrics for evaluating the performance are essential for assessing the model's reliability after any image classification task has been completed. Precision, Recall, F1-score, and Accuracy are some of the well-known measures of performance evaluation that are used in the quantitative evaluation process to identify how well the CNN-MCD technique performs. The Area Under the Curve of the Receiver Operating Characteristic (ROC), also known as the AUC of ROC, is another useful metric for assessing performance. The CNN-MCD method's findings are shown in Table 6.2. Our results revealed 0.939 as the accuracy, with 0.940 as the precision. The results demonstrate that the recall is 0.930 and the loss is 0.031. Both the F1 and AUC scores sum up to 0.935 and 0.976 respectively. The usage of a dataset from the physical world causes the CNN-MCD technique to underperform with a smaller number of epochs; however, as demonstrated in Figure 6.11, the results become more consistent when 20 epochs have been used. After 70 iterations of the training process, we decide to stop because the validation performance is not incresing and overfitting might occur.
The distributions of the MCD forecasts and the ensemble's prediction are displayed in Figure. 6.12. Calculating the mean and variation of all potential model outputs yields these distributions. Similarly, performing these computations on the predictions are used to determine the samples of uncertainty. Table 6.3 displays two uncertain samples with a spectrum of their outcomes. In doing so, the dataset is refined and model issues are highlighted. The graph shows the classes 0 through 4 labelled as covid-19, HB, MI, Normal, and PMI. The first sample has a correct

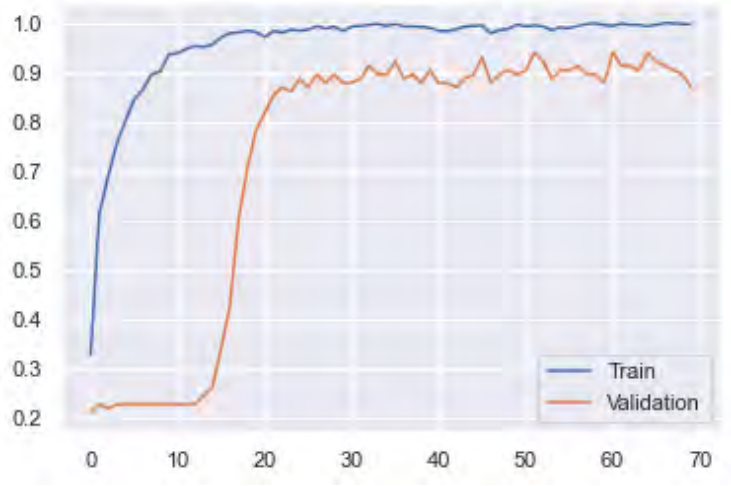


Figure 6.10: Training and Validation Accuracy Curves for CNN-MCD Method

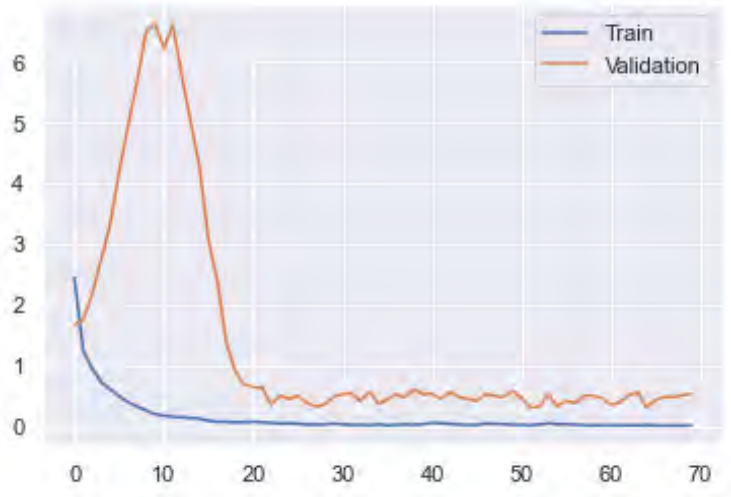


Figure 6.11: Training and Validation Loss Curves for CNN-MCD Method

label prediction of "HB", however it can be seen from the table, the model is rather confused about this prediction, mislabeling the input as "PMI" as well. The label cannot be predicted for the second sample. The model is unsure whether to identify its resultant predictions as "PMI" or "covid-19". As a result, MCD assists the model in highlighting uncertainty in a manner which precludes dubious diagnoses, that is uncertain diagnoses.
Comparing our suggested model to the state-of-the-art models in multi-class categorization, displayed in table 6.4, we find that ours performs significantly better. Widely used models like ResNet-50, VGG-16, and InceptionV3 could very well obtain an accuracy ranging from 79% to 83%. Our five-class classification performance is superior to that of the other methods listed in Table 6.4, which only classify data into three or four categories. Despite not being able to exceed the competition in terms of recall and precision in certain circumstances, it is nevertheless, quite close to those results.

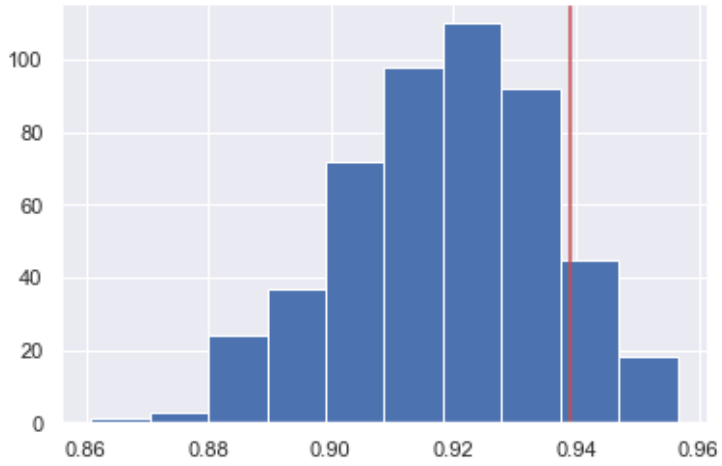
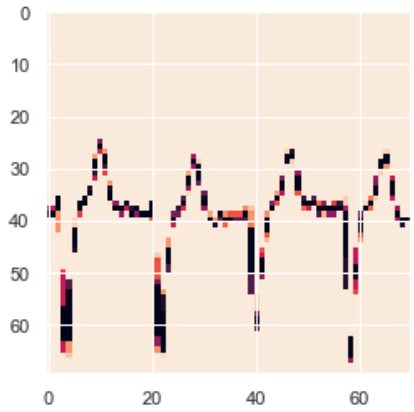
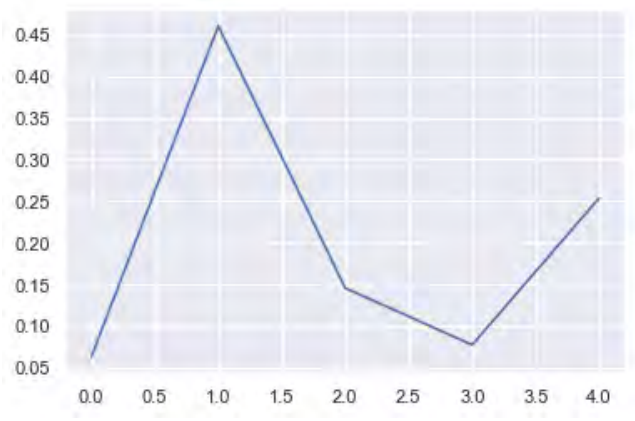
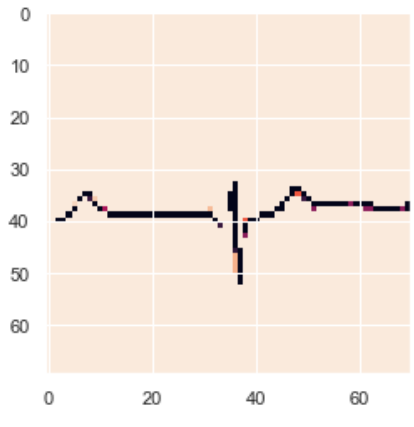
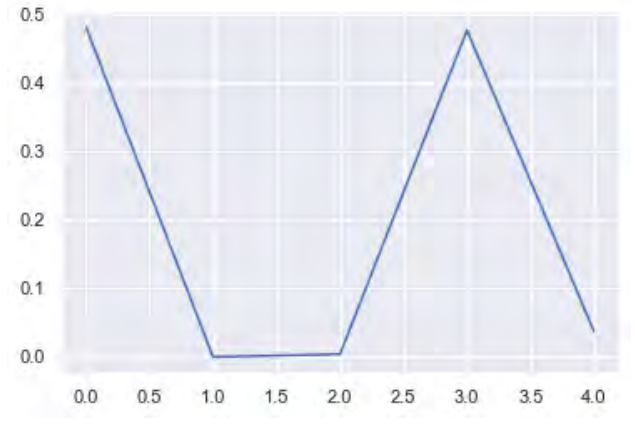


Figure 6.12: Distributions of the monte carlo prediction accuracies and prediction accuracy of the ensemble (Red).

Table 6.3: Uncertain samples and their predictions

Uncertain Sample	Probability Distribution of Prediction
	
	

Involution-Convolution Hybrid in Skin Cancer and Malaria Disease Classification

The combination of involution and convolution show promising results. For comparison, we have tested two variations of UnIC-Net, one having 1 involution layer and the other having 2 layers. Even though the performance is very close, but a

Table 6.4: Comparison among different models.

Method	Classes	Accuracy	Precision	Recall
ResNet-50, InceptionV3, VGG-16[150]	5	78.08%, 79.35%, 83.74%	NA	NA
Proposed CNN Model of [150]	5	83.05%	96.12%	95.39%
Attention-based CNN with Residual Connections [151]	4	92.00%	95.99%	82.00%
ECG-BiCoNet [152]	3	91.70%	91.90%	95.90%
Proposed CNN-MCD	5	93.90%	94.00%	93.00%

Table 6.5: Performance of UnIC-Net on Malaria Parasitized Cell Images

Model Type	**Accuracy**	**Precision**	**Recall**
1 Layer Involution	96.81%	96.20%	97.45%
2 Layer Involution	94.95%	97.10%	92.65%

critical analysis show how they can be different in performance.
We will compare the involution kernels, so that we can visualize how the model is taking decision. First in Figure 6.13, we can see how the involution layer is focusing on the spatial features, especially the parasitized locations. It is signifying the parasitized area and making the feature extraction process easier. But, in Figure 6.14 we can see how 2 layers of involution learns too much from the data and makes the performance unstable. In most of the cases, this issue occurs. 1 layer involution does not create this problem. The same happens in the skin lesion dataset (Figure 6.16 and Figure 6.15).
Figure 6.17 and Figure 6.18 show the training vs validation curves, which show that our proposed model do not face any serious case of overfitting. Table 6.5 and Table 6.6 show the performance of our proposed UnIC-Net model. From the tables, it is clear that single layer involution works better in UnIC-Net. Moreover, UnIC-Net model has a very high recall, which means it is able to capture more diseased class images, which is very crucial for computer-aided-diagnosis.
Finally, we compare the results obtained from UnIC-Net with other existing models. From Table 6.8 and Table 6.7 we can come to a decision that, our model outperforms the mentioned work.
Now coming to the uncertainty analysis. UnIC-Net with 1 layer involutions has a better Monte Carlo ensemble accuracy than that of 2 layers. We have taken 500 samples and tested 500 times. In Malaria Parasitized Cell Images, UnIC-Net with 1 layer involution has 97.03% Monte Carlo ensemble accuracy and the one with 2 layers has 94.4% Monte Carlo ensemble accuracy. In HAM10000, UnIC-Net with 1 layer involution has 97.8% Monte Carlo ensemble accuracy and the one with 2 layers has 96.01% Monte Carlo ensemble accuracy. Table 6.9 is a comparison between these

Table 6.6: Performance of UnIC-Net on HAM10000 Images

Model Type	**Accuracy**	**Precision**	**Recall**
1 Layer Involution	98.79%	98.93%	99.03%
2 Layer Involution	97.39%	97.12%	98.33%

Table 6.7: Performance Comparison with Existing work on HAM10000

Model	**Accuracy**
Fixcaps[153]	96.49%
IRv2+Soft Attention[154]	93.4%
Skin lesion classification using loss balancing and ensemble of multi-resolution efficient nets[155]	92.6%
Classification Model - 2 with two path CNN model[155]	88.6%
ISIC 2019 Skin lesion analysis towards melanoma detection[155]	85.1%
Skin lesion analysis towards melanoma detection using siamese network[155]	83.2%
Our Model	**98.79%**

Table 6.8: Performance Comparison with Existing work on Malaria Parasitized Cell Images

Model	**Accuracy**	**Recall**
ResNet50 [156]	88.47%	89.61%
DenseNet121 [157]	90.94%	92.51%
DPN92 [158]	87.88%	86.81%
Customized CNN [159]	94.00%	93.10%
Designed CNN [87]	94.61%	95.20%
Our Model	**96.81%**	**97.45%**

Table 6.9: Monte Carlo Ensemble Accuracy in UnIC-Net Model Variants

Dataset	Model Type	MC Ensemble Accuracy
Malaria Parasitized Cell Images	UnIC-Net (1 layer involution)	97.03%
	UnIC-Net (2 layer involution)	94.4%
HAM10000	UnIC-Net (1 layer involution)	97.8%
	UnIC-Net (2 layer involution)	96.01%

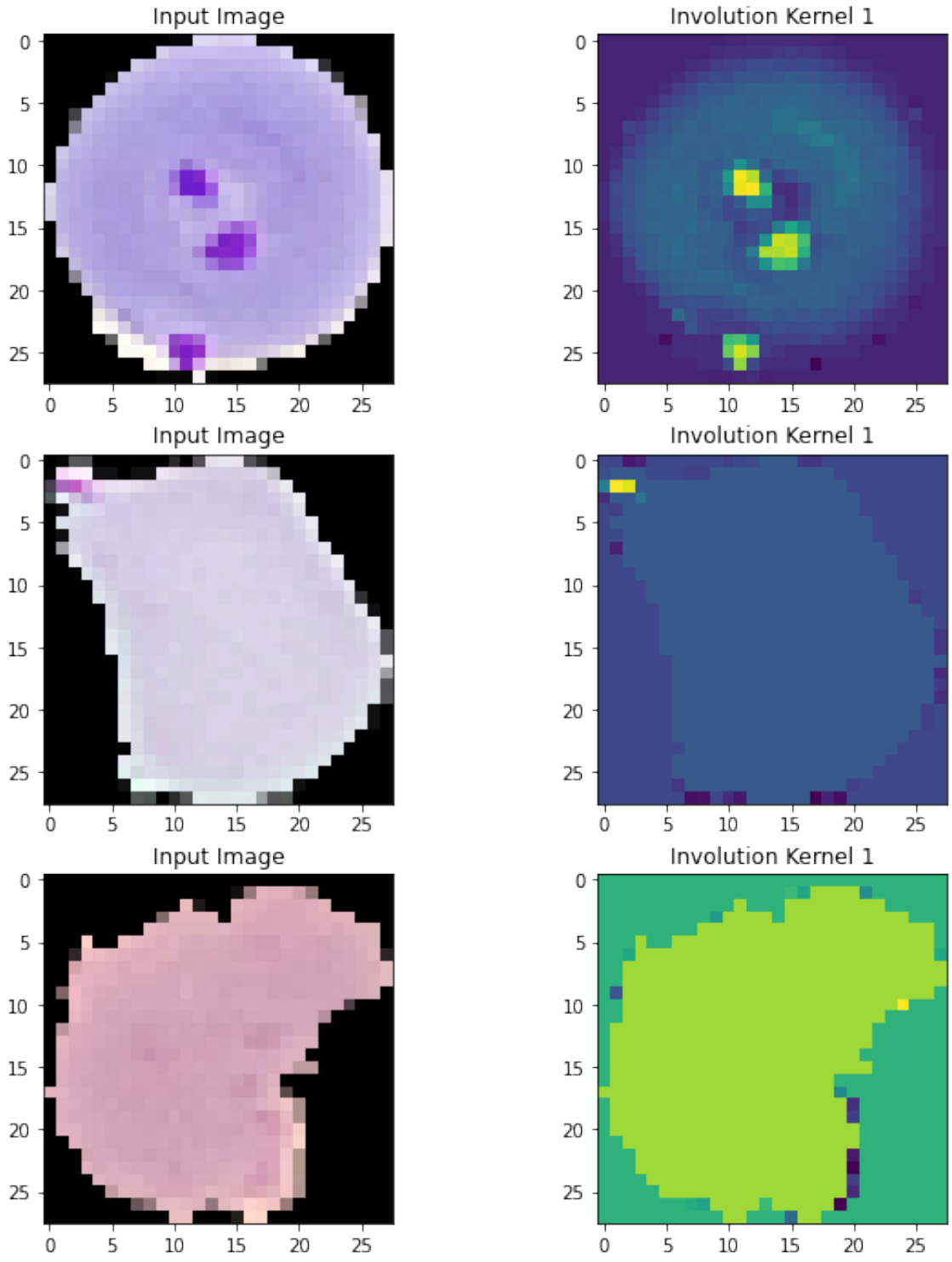


Figure 6.13: Involution Kernels of UnIC-Net (1 Layer Involution) on Malaria Parasitized Cells

two variants in terms of Monte Carlo ensemble accuracy, which indicates overall reliability.

A more detailed analysis also proves that UnIC-Net with 1 layer involution is more reliable. Since we want high entropy in wrong prediction, the 1 layer involution based UnIC-Net is preferred. Figure 6.19 is the sample where both variants of the

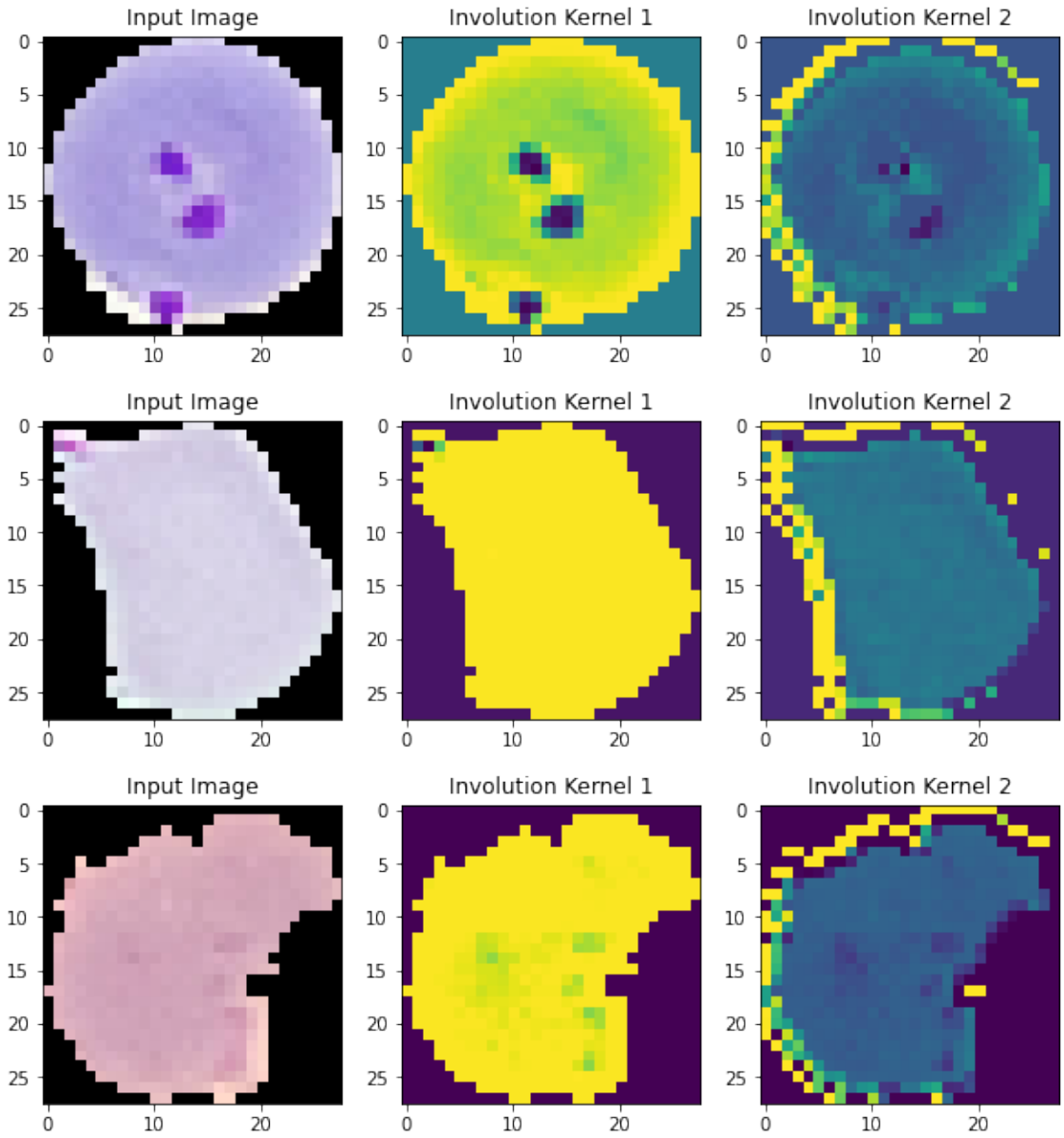


Figure 6.14: Involution Kernels of UnIC-Net (2 Layer Involution) on Malaria Parasitized Cells

proposed model predicted wrong. Figure 6.20 shows that it is more uncertain about the prediction, which has an entropy of 0.76. On the other hand, Figure 6.21 is more certain about it's prediction, with 0.62 entropy.
When we choose an uncertain sample (Figure 6.22), in this case. UnIC-Net (1 layer involution) predicted it correctly but UnIC-Net (2 layer involution) predicted it wrong. In this case, UnIC-Net (1 layer involution) has 0.32 entropy and UnIC-Net (2 layer involution) has 0.73 entropy. This means, UnIC-Net (1 layer involution) is a bit less confident about its correct prediction. But UnIC-Net (2 layer involution) is uncertain about its wrong prediction. Both of them are quite equal in this case.

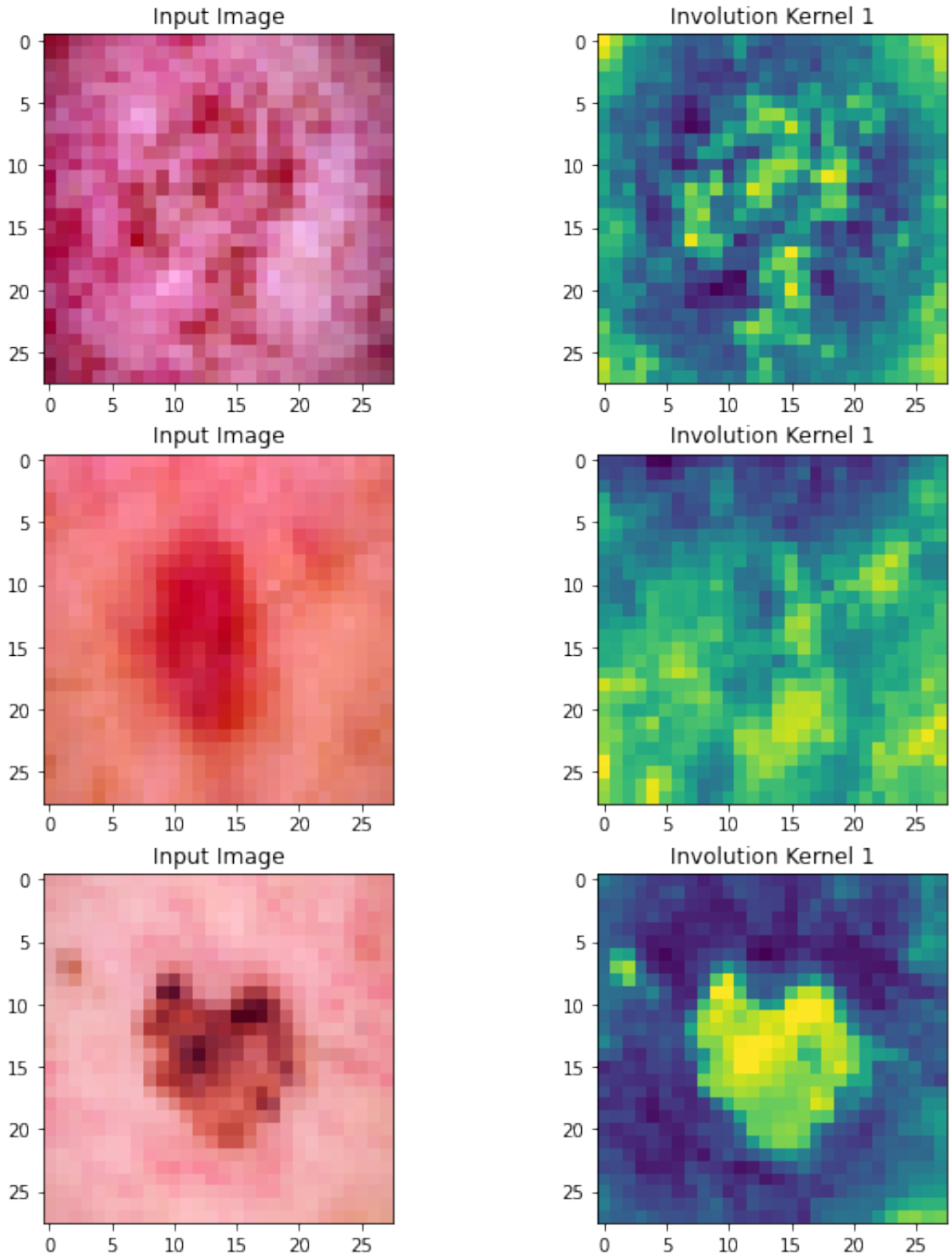


Figure 6.15: Involution Kernels of UnIC-Net (1 Layer Involution) on HAM10000

Transformers in Vision

For the performance evaluation, we use baseline transformer models (ViT, SWT, CCT), MCD-based transformer models with a dropout rate of 50% (ViT-MCD (0.5), SWT-MCD (0.5), CCT-MCD (0.5)), and MCD-based transformer models with a dropout rate of 10% (ViT-MCD (0.1), SWT-MCD (0.1), CCT-MCD (0.1)).
In table 6.10 we can see that, in most of the cases, MCD has increased the perfor-

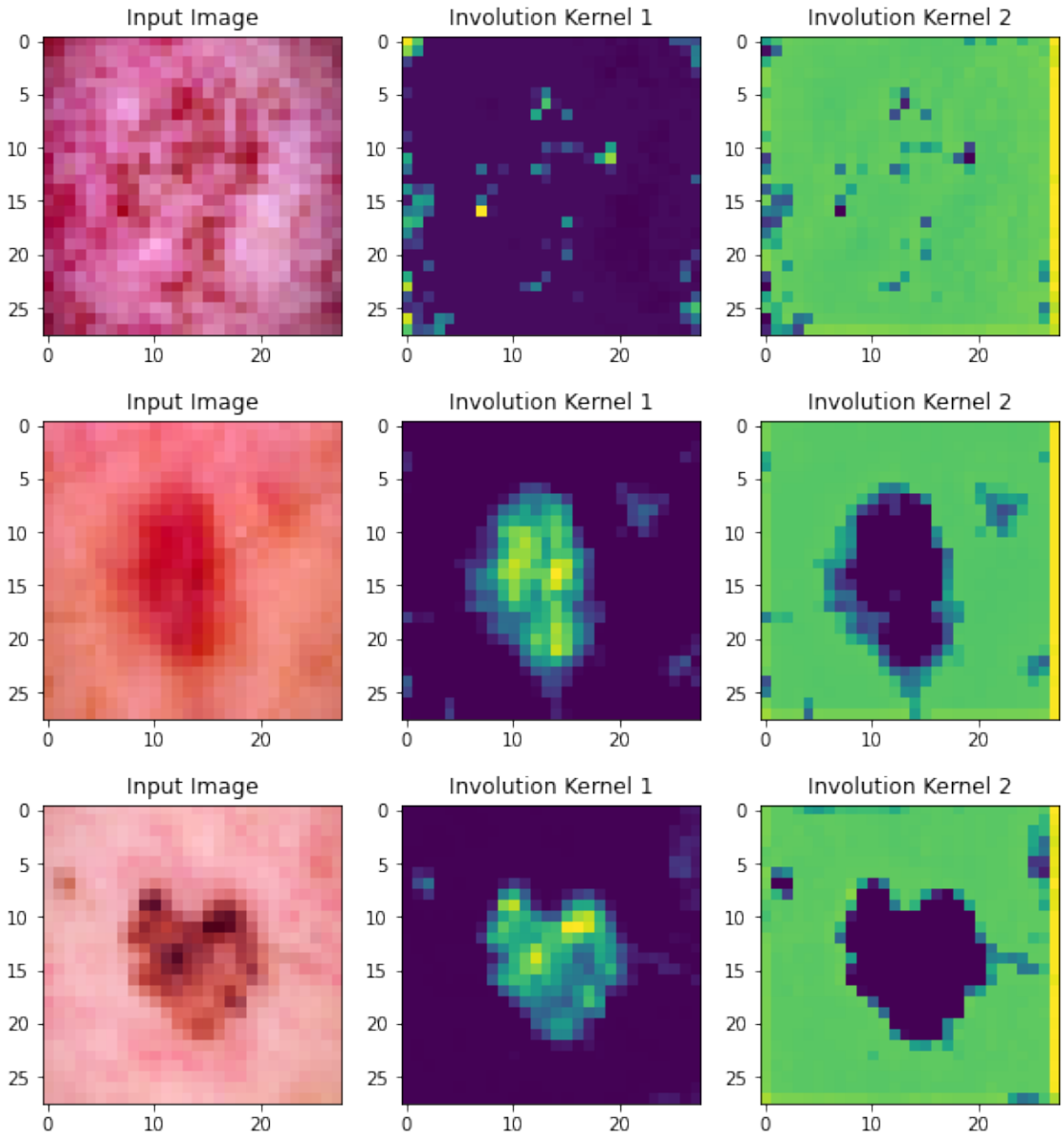


Figure 6.16: Involution Kernels of UnIC-Net (2 Layer Involution) on HAM10000

mance. Both SWT-MCD models have the best performance gain after the MCD embedding. Although, CCT-MCD model has the highest accuracy. Unlike the recall and precision measure of CCT models, ViT and SWT models have a higher precision than recall.

Using MCD embedded models with 50% dropout rate, to determine the distribution of predictions, we utilize 500 test samples and predict each sample 500 times. In other words, we employ Monte Carlo Sampling. The uncertainty resulting from the anticipated class-wise softmax value spread of the 500 test specimens necessitates this to be done. We get 10.59%, 71.8%, and 77.6% ensemble accuracy from ViT-MCD, SWT-MCD, and CCT-MCD respectively. Unlike regular accuracy score, this ensemble accuracy is achieved from the Monte Carlo Sampling with the 500 sample data. Figure 6.25 shows the Monte Carlo accuracies achieved from the experimented model.

Figure 6.24 represents the most uncertain samples. Figure 6.26a(left) is the ran-

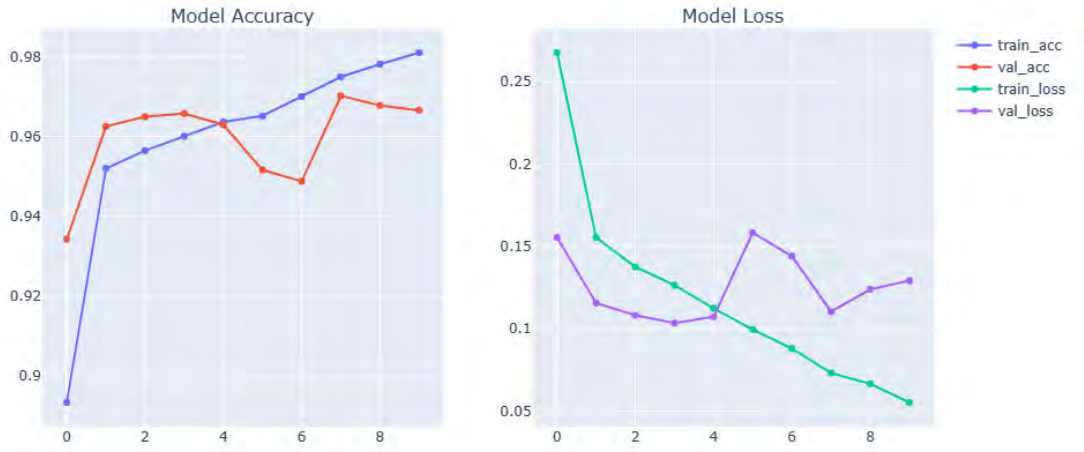


Figure 6.17: Training vs Validation Curves of Malaria Infected Cell

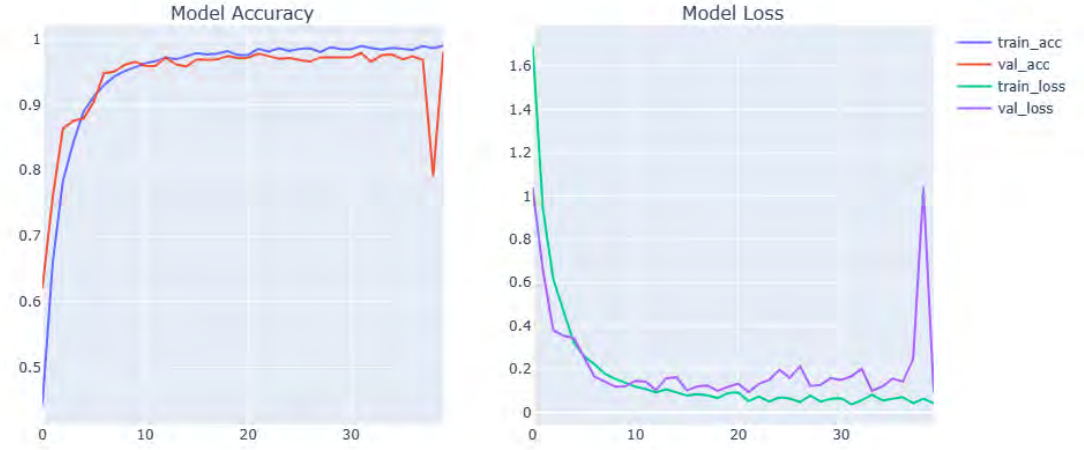


Figure 6.18: Training vs Validation Curves of HAM10000

domly selected image from the dataset. We can see the distribution of the predicted softmax score of three MCD-embedded models in Figure 6.28. The quantitative outcomes can be seen in Table 6.11 It shows that, both SWT-MCD and CCT-MCD models successfully classified the image with low predictive entropy. On the contrary, ViT-MCD could not identify the image and the predicitive entropy is comparatively higher, which means the predictions are more disordered than others.
We generate a random image and predict the image in the same way we did for the randomly selected test image. Figure 6.26b(right) is the randomly generated image which is basically a completely noisy data. Figure 6.27 shows the mean softmax score of the distribution. This represents that, the model is not very confident about it's decision as the gap between mean softmax score is not too significant.

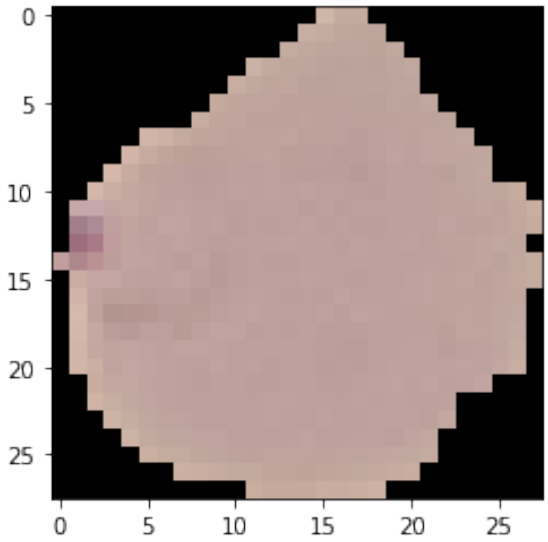


Figure 6.19: Sample of a Data with wrong predictions for all

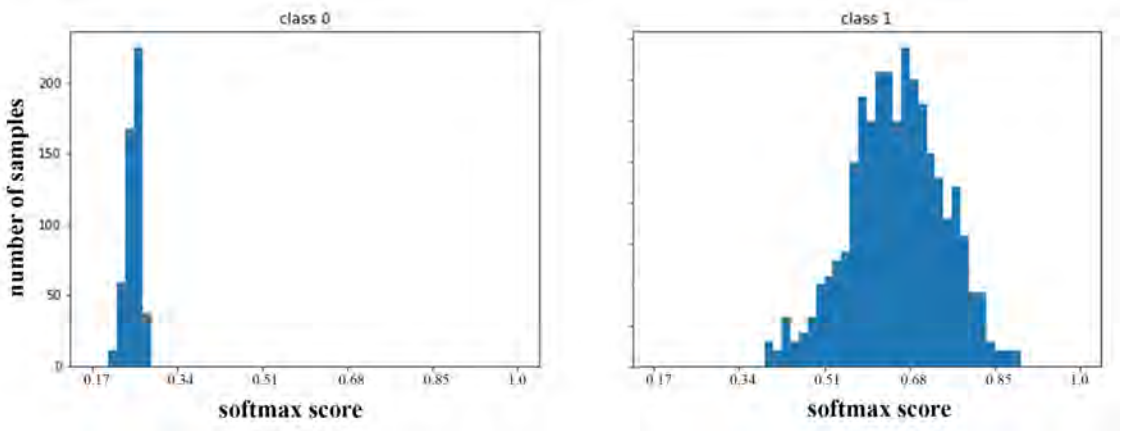


Figure 6.20: Softmax score distribution of the wrong prediction (1 layer)

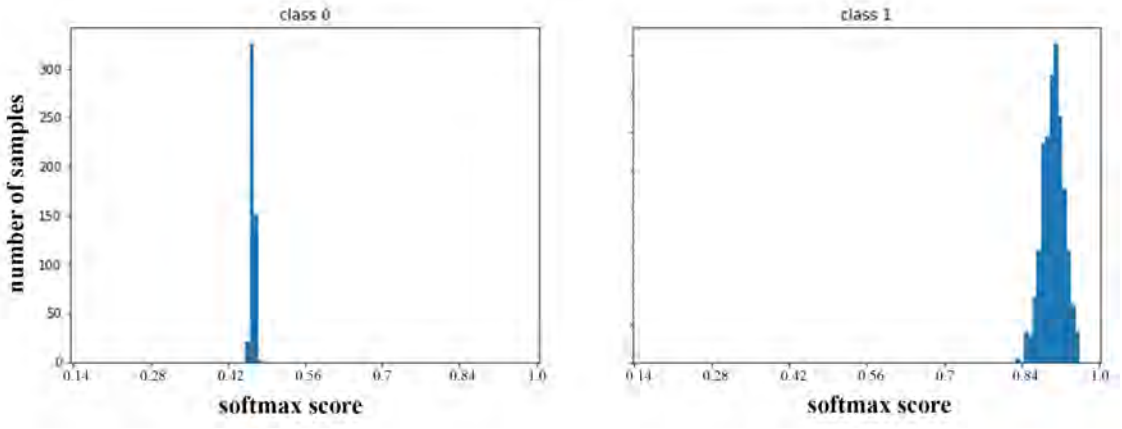


Figure 6.21: Softmax score distribution of the wrong prediction (2 layer)

The prediction of the SWT-MCD model is comparatively more confident about it's decision although this is not a good sign for the model, because the model's softmax scores should be disordered when dealing with a fully noisy data like the one we

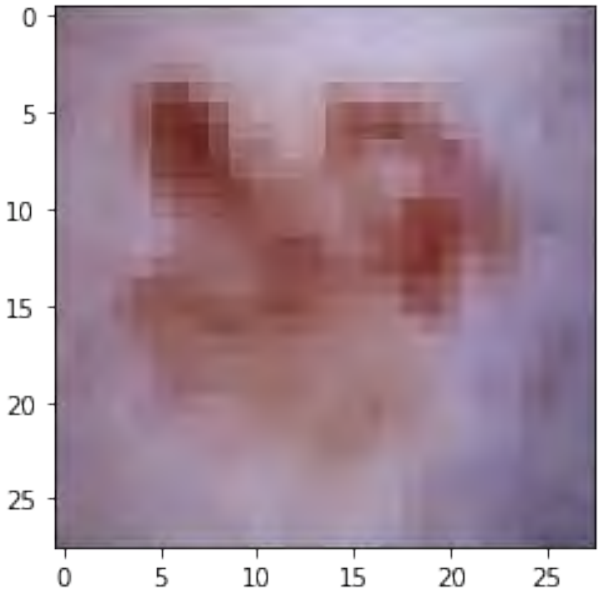


Figure 6.22: Sample of a Data with uncertain prediction

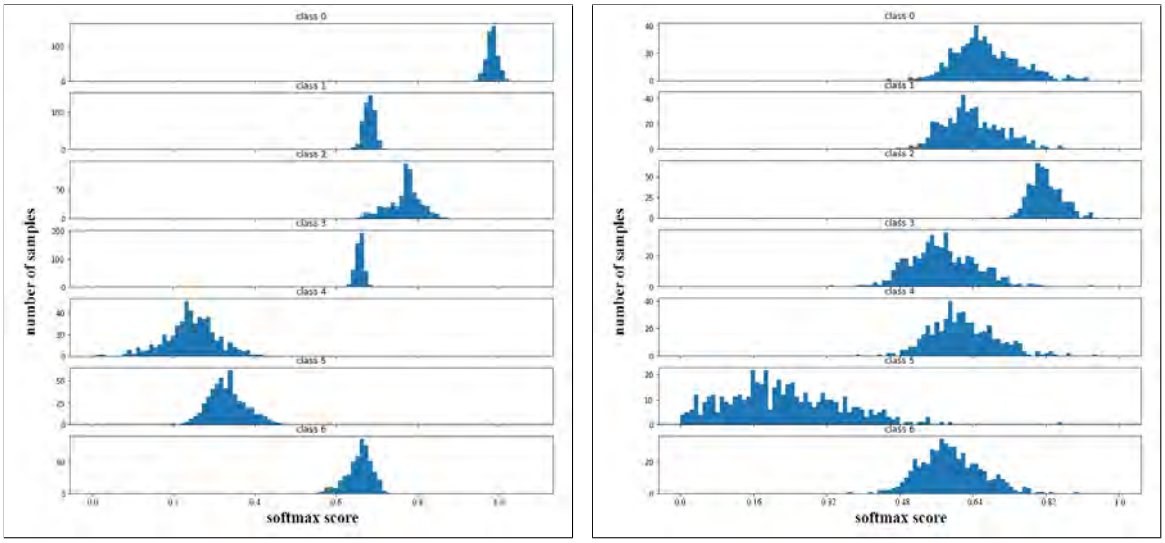


Figure 6.23: Softmax score distribution of UnIC-Net with 1 and 2 Involution Layer

have used. For that reason, CCT-MCD has the best outcome here.
Unlike the predictive performance, CCT and ViT have more similar characteristics in terms of uncertainty measurement and decision making. If we only consider the distribution of the softmax score, and the mean softmax score of the random image prediction; we'll see that ViT has more similarity with CCT in terms of uncertain behaviour.

6.3.4 GNN Result Analysis

GNN

We will analyse the performance first, beginning with accuracy and F1 score. From Table 6.12 we see that, GAT outperformed all in terms of accuracy and F1 score. The results achieved from FFN clearly shows how weak FFNs are in this task. Other

Table 6.10: Test Results of Transfomrer Models

Model	Accuracy	Top-3 Accuracy	Recall	Precision
ViT	67.8%	89.78%	49.78%	67.2%
ViT-MCD (0.5)	68.44%	91.21%	55.9%	69.6%
ViT-MCD (0.1)	68.2%	90.3%	51.70%	68.3%
SWT	68.78%	91.59%	53.78%	81.65%
SWT-MCD (0.5)	71.4%	92.31%	55.98%	**84.50%**
SWT-MCD (0.1)	67.2%	90.42%	49.70%	83.17%
CCT	77.9%	94.1%	93.39%	46.11%
CCT-MCD (0.5)	77.1%	94.87%	93.47%	46.71%
CCT-MCD (0.1)	**78.4%**	**94.97%**	**93.64%**	48.79%

Table 6.11: Uncertainty Estimates of a Randomly Selected Data

Model	Predicted Class	True Class	Predictive Entropy
ViT-MCD (0.5)	0	2	0.88
SWT-MCD (0.5)	2	2	0.36
CCT-MCD (0.5)	2	2	0.19

Table 6.12: Performance and Uncertainty Report of GNNs

Models	Case	Accuracy	F1	Predicted	Actual	Entropy
FFN	1	63.1%	59.13%	1	1	0.83
	2			0	5	0.85
GCN	1	80.15%	79.33%	1	1	0.47
	2			0	5	0.72
GAT	1	81.33%	82.41%	1	1	0.03
	2			2	5	0.77
GraphSAGE	1	79.12%	79.54%	1	1	0.08
	2			1	5	0.78
PPNP	1	80.89%	82.27%	1	1	0.33
	2			3	5	0.67
APPNP	1	80.31%	81.04%	1	1	0.36
	2			3	5	0.64

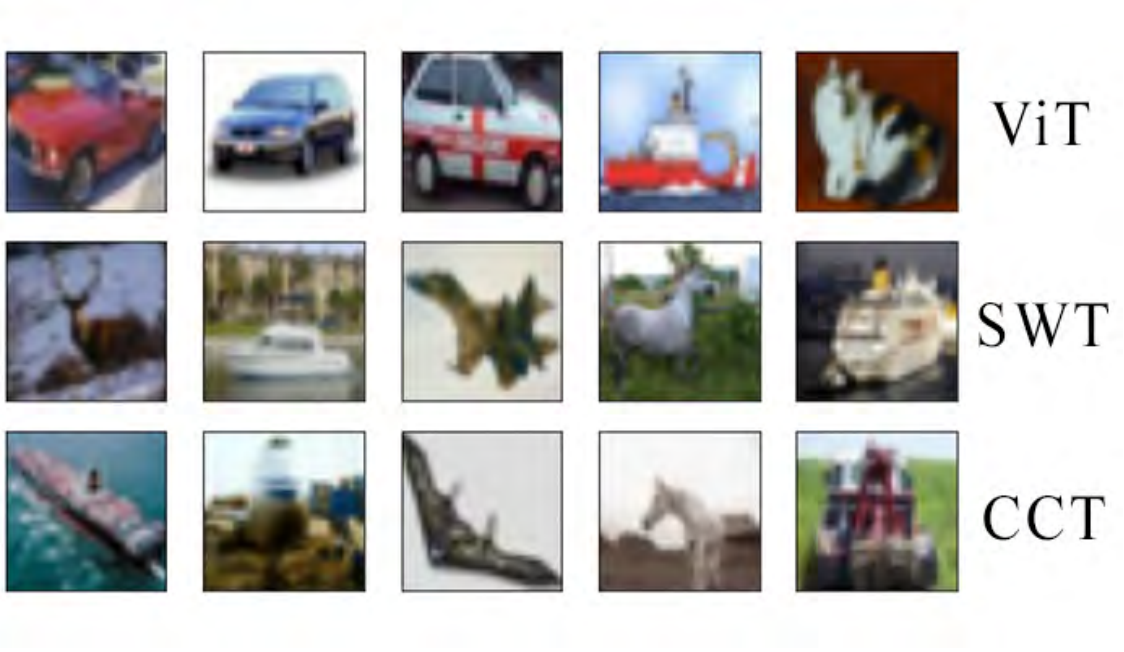


Figure 6.24: Most Uncertain Samples Selected by Variance (32 x 32 Pixels)

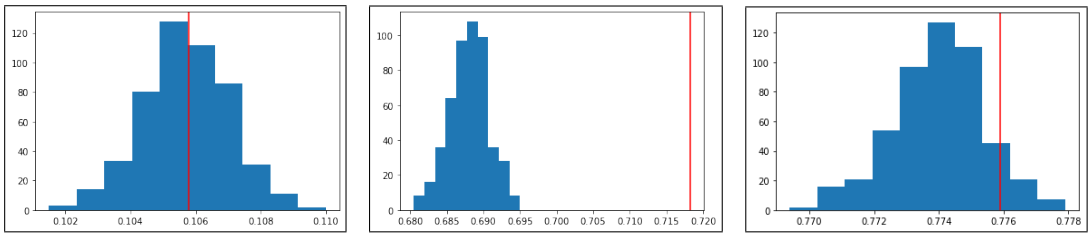
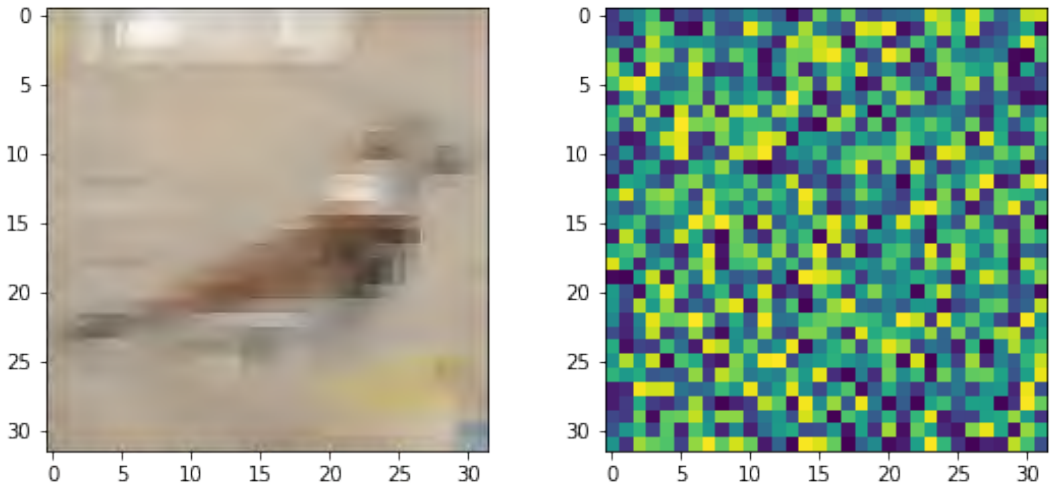


Figure 6.25: Distributions of The Monte Carlo Accuracy and Accuracy of The Ensemble in Red (from left, ViT-MCD, SWT-MCD, CCT-MCD)

four GNN models: GCN, GraphSAGE, PPNP and APPNP performs quite identical, although GraphSAGE has slightly low accuracy and F1 score. Table 6.13 contains the accuracy and loss curves of all utilized models in this experiment. From the figures of the PPNP and APPNP curves, we can see that they fit the graph better than the other models. GraphSAGE and GAT have more stable curves, indicating that they are learning steadily. Also, they take lesser epochs to complete their training. We have used earlystopping method to finish the training process when the model fails to increase performance.
Now coming to the overall reliability performance, GAT outperforms all here as well. GAT has 83.4% Monte Carlo ensemble accuracy. Other models have respective Monte Carlo ensemble accuracies, 40.7% (FFN), 79.9% (GCN), 82.6% (GraphSAGE), 79.2% (PPNP), 78.9% (APPNP). This shows that, reliability performance is different than the raw performance. Even though GraphSage has less accuracy, it outperformed GCN, PPNP, APPNP, FFN in terms of Monte Carlo ensemble Accuracy. This is an interesting finding that shows raw performance and uncertainty can vary.
From Table 6.14, we see the distribution of the softmax scores. The x-axis represent the score and the y-axis represents the number of samples, which means the chart represents the distribution of the softmax score. For correct predictions, we would



(a) (b)

Figure 6.26: Sample Test Image and Random Image For Uncertainty Estimation (32 x 32 Pixels)

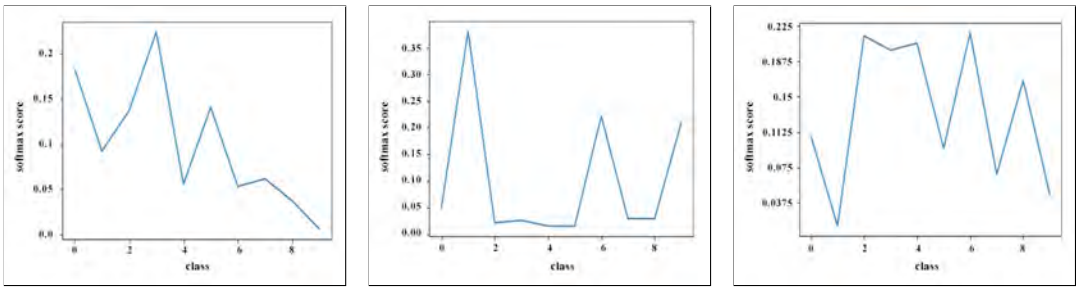


Figure 6.27: Mean Softmax Score of The Randomly Generated Image 6.26b (from left, ViT-MCD, SWT-MCD, CCT-MCD)

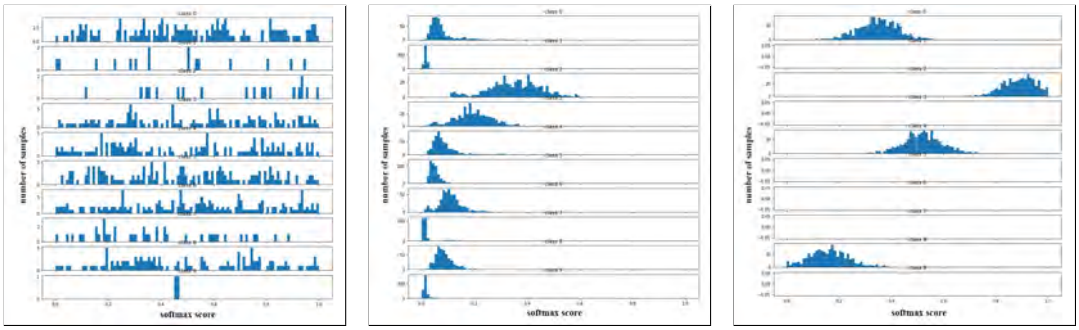


Figure 6.28: Probability Distributions of Predictions (from left, ViT-MCD, SWT-MCD, CCT-MCD)

want the entropy to be as low as possible. On the other hand, for wrong predictions, we would want high entropy. This means, the model is not confident about the prediction, and it can lead to faulty decisions. Table 6.12 has the quantitative measure of uncertainty estimation. In this uncertainty analysis, we have two cases, the first one is a randomly selected sample where all of the models predicted correctly. On the other hand, the second case is where a random uncertain sample was selected from the dataset. In the second case, some of them predict correctly, and some not. Considering both correct and incorrect predictions (in multiple samples), GAT outperforms all in this task as well. Having the least entropy in correct prediction and high in the wrong one (in multiple cases as well). GraphSAGE performs quite similarly in this case as well. However, GraphSAGE is more uncertain about its wrong prediction, which technically makes it better than GAT in terms of indicating faulty predictions. However, GCN did not perform as good as the other methods (except FFN). Even though it has a decent accuracy and F1 score. PPNP and APPNP proved to be superior in terms of uncertainty measure. But, the problem with PPNP and APPNP is, both are slightly more certain about its prediction even if it is wrong. Therefore, PPNP and APPNP provide more certain predictions but they do not ensure correct predictions. This issue is seen in the GCN model as well, but GCN is not very certain about its predictions in most of the cases. From the conducted experiment, we can say that spatial GNNs are performing much better than spectral GNNs.

Spatial Convolution operates on the local neighborhood of nodes and deduces the features of a node based on its local neighbors. For GCN, all nodes share the same parameters for the convolution kernel. As the degree of association between neighboring nodes varies, this influences the final outcome in some instances. To treat distinct nodes, various convolution kernel parameters must be used. GraphSAGE uses the same approach with a generalized aggregation function. GAT, on the other hand, employs different convolution kernel parameter values to treat distinct nodes. These modifications make the GAT more efficient and reliable than its competitors. PPNP and APPNP capture the impact of infinitely neighborhood aggregation and separate the feature transformation from propagation to simplify model construction, leading to confident predictions. Additionally, a good learning fit and a higher F1 score is achieved from these two models.

From the above analysis, we can say that GAT and GraphSAGE are safer to use in node classification tasks. They provide reliable predictions with highest or close to the highest accuracy, which are important for real-world tasks.

Table 6.15 highlights the experimental outcomes of our thesis. Here, we can tell which model has the highest level of raw performance and which is the most reliable.

Table 6.13: Accuracy and Loss Curves of Training vs Validation Phase

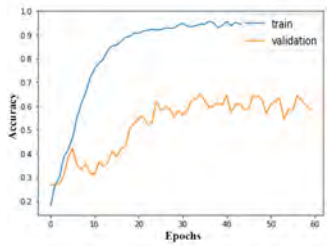
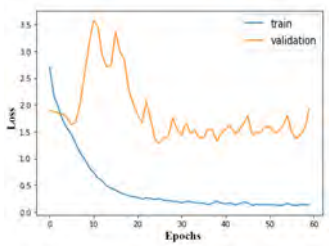
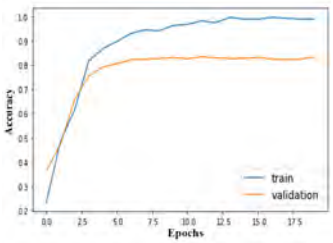
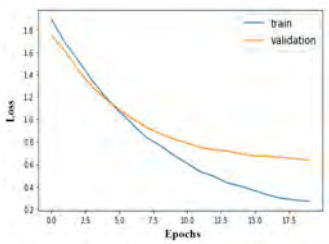
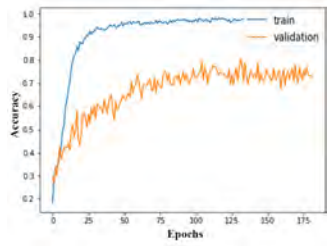
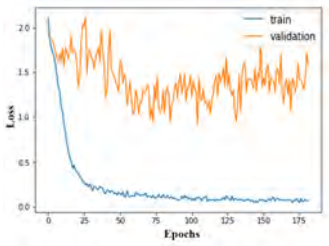
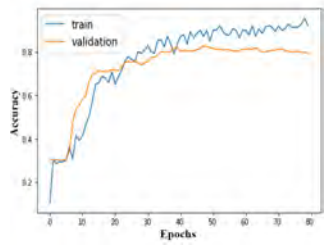
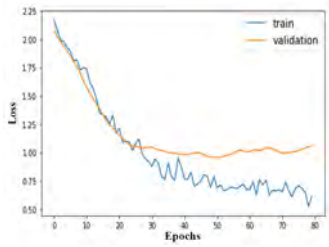
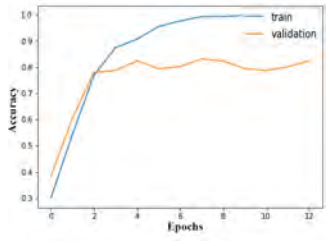
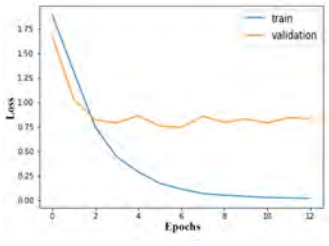
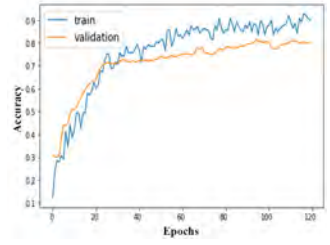
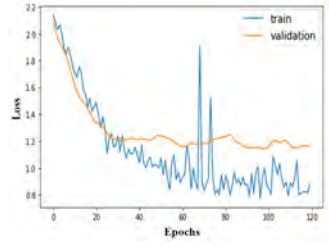
Accuracy	**Loss**
	
FFN	FFN
	
GraphSAGE	GraphSAGE
	
GCN	GCN
	
PPNP	PPNP
	
GAT	GAT
	
APPNP	APPNP

Table 6.14: Softmax Score Distribution of Correct and Incorrect Predictions

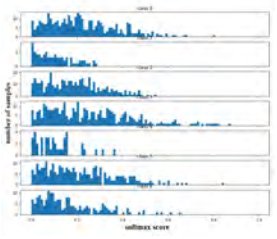
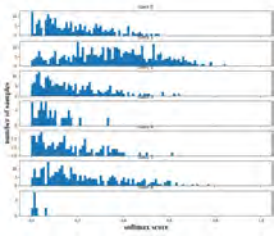
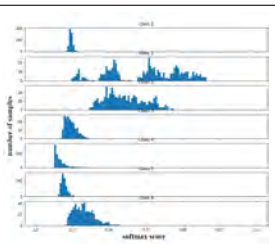
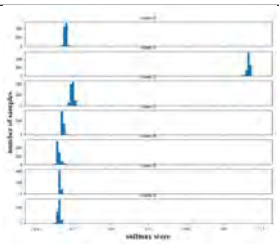
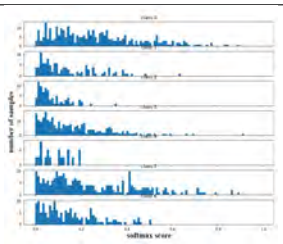
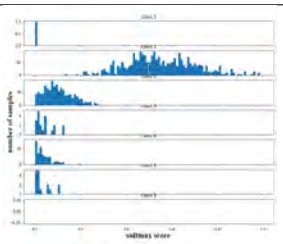
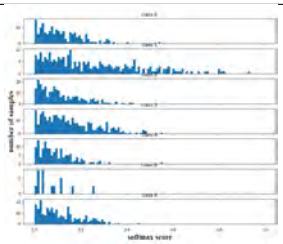
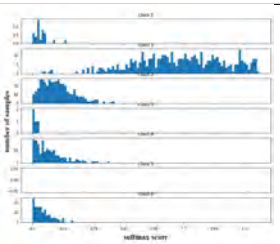
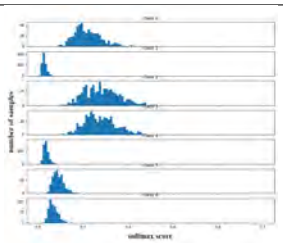
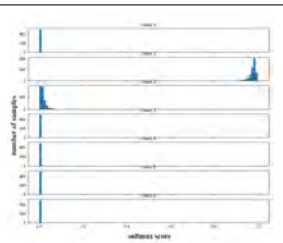
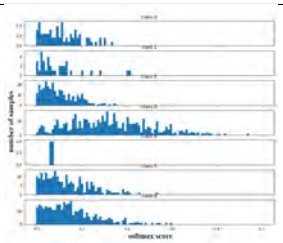
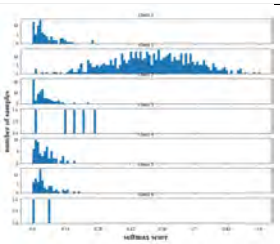
Incorrect Prediction	**Correct Prediction**
	
FFN	FFN
	
GraphSAGE	GraphSAGE
	
GCN	GCN
	
PPNP	PPNP
	
GAT	GAT
	
APPNP	APPNP

Table 6.15: Summary of our Experiments

Task	Models Utilized	Dataset	Best Performer	Most Reliable
Regression	Lightweight ANN, Heavyweight ANN	California Housing Price	Heavyweight ANN	Heavyweight ANN
Text Classification	LSTM, GRU, BERT, XLNet	Yelp Review Polarity	LSTM	BERT
Image Classification	Proposed CNN	ECG Trace Images	Proposed CNN	-
Image Classification	UnIC-Net 1 Layer, UnIC-Net 2 Layer	HAM10000	UnIC-Net 1 Layer	UnIC-Net 1 Layer
Image Classification	UnIC-Net 1 Layer, UnIC-Net 2 Layer	Malaria Parasitized Cells	UnIC-Net 1 Layer	UnIC-Net 1 Layer
Image Classification	ViT, SWT, CCT	CIFAR10	CCT	CCT
Node Classification	ANN, GCN, GAT, GraphSAGE, PPNP, APPNP	CORA	GAT	GAT

Chapter 7

Discussion and Key Findings

7.1 ANNs in Regression

As we have mentioned previously, uncertainty measures are strongly related to feature selection and model size. California house price dataset has less data in the end, which lead to uncertain predictions in that range. Therefore, data requirement is also an important factor for certainty in regression predictions.

7.2 RNN Variants vs Tranformer Variants

In terms of performance, LSTM outperforms other models used in our experiments on the Yelp Review Polarity dataset. LSTM is better in terms of both uncertainty measures and performance. GRU is very close to the LSTM results. The structural similarity is the main reason for them to perform similarly. Although LSTM could not beat BERT in both ways.
From our study we find that BERT outperforms XLNet in terms of both performance and uncertainty measurements. But LSTM and GRU outperformed in terms of performance only but the predictions were not as certain as BERT, with the same dropout rate of 35%. BERT also has has good fit during it's training phase, unlike XLNet. Currently, our experiments only include one particular dataset. We will be using multiple datasets to solidify the great performance of BERT.

7.3 MCD-Based CNN model increases robustness

In clinical data analysis and diagnosis, MCD plays a vital role to reduce the risk factors. Our proposed CNN model outperformed existing models in the particular dataset and the uncertainty aware framework makes it more feasible. The data-wise uncertainty analysis can play a key role when dealing with risky data. In addition to this, the ensemble process of the prediction components in this specific scenario raises the model's degree of resilience. It is possible for us to draw the conclusion that this kind of diagnostic framework may be extremely useful in the process of computer-aided medical diagnosis [160].

7.4 How Good the Combination of involution and convolution is

Frameworks such as Convolutional Neural Networks (CNNs) and Involutional Neural Networks (INNs) have proven useful for picture categorization. Since CNNs can grasp picture properties and patterns at the detailed level using convolutional layers, they are particularly suited for working with image data. On the other hand, INNs are a type of architecture that has been suggested to improve the interpretability and robustness of neural networks by learning to invert the data.
When applied to cell-like images, the combination of CNNs and INNs can perform better than either architecture alone because it can leverage the strengths of both architectures. CNNs can learn the local patterns and features in the image that are important for classification, while INNs can learn a more global, holistic representation of the image that is more robust to noise and variations in the image.
As described in the result analysis section, UnIC-Net with 1 layer involution is enough for the spatial feature extraction. Multiple layers of involutions causes overfitting, unstable results and makes the kernels more confusing.
In addition, INNs can be used to generate images from a low-dimensional latent space, which can be useful for data augmentation and for understanding the underlying generative processes of the data. This can be beneficial in cell-like images because it can be used to generate new examples of cells with various variations, which can increase the model's stability and applicability.
Therefore, the combination of CNNs and INNs can be beneficial in cell-like images because it can leverage the strengths of both architectures to achieve improved performance, robustness, and interpretability.

7.5 Best Among ViT, SWT and CCT

From the conducted experiments of transformers in vision, we can say that transformer based models are greatly affected by the use of MCD. The improvement in accuracy, precision and recall is visible in all of the models. Larger model, ViT is uncertain with it's predictions and also provides wrong predictions. In contrast, compact frameworks are performing better regarding certainty and as well as predictions. Even if the predictions are comparatively uncertain, it is classifying the images correctly, hence the monte carlo accuracy gap between models are quite large. Moreover, the test accuracy of ViT and the monte carlo accuracy is very far away. Which indicates how uncertain and unstable the model is with less amount of data. ViT and large-scale variations of ViT need a massive amount of training data to perform optimally.[161], [162]. In that case, CCT gives the best of both convolutional and transformer kinds.
The dropout rate also has a significant role in prediction, and certainty. Transformer models without convolutions, ViT and SWT are achieving higher precision but lower recall. And the CCT model is getting the opposite results. The main difference here is the model architecture. SWT being an updated and compact variant of ViT, both of them are showing similar performance in the test phase. But when it comes to uncertainty measurement, ViT and CCT are showing similar characteristics although the quantitative difference of the performance is quite significant. SWT is

more certain with it's decision if the input is out-of-scope.
From Figure 6.24, it is evident that ViT has difficulties recognizing automobile images. In contrast, SWT and CCT have mixed samples of uncertain predictions. Additionally, the most uncertain samples were not prevalent in every model., indicating that each model approaches its predictions differently. From both out-of-scope tests, we can infer that the ViT model does not have a high degree of confidence in its predictions, but the CCT model performs comparatively better with less training data than ViT. The same may be stated for the SWT model, however in this experiment, the SWT model fails to accurately predict random noise.

7.6 Insights from Graph Neural Networks

Graph Networks could not perform the same but in node classification, embedding of the attention network helped GAT performed much better in both areas. It has given GAT 7% more accuracy and the entropy measures are much better. We can see from Table 6.12 how certain the predictions are.
Graph Attention Networks (GATs), one kind of neural network, excel at processing information that is already organized in a graph format. It is more certain with its predictions because it uses self-attention mechanisms, which allow the network to concentrate on the most notable features of the graph for each prediction. In traditional CNNs, the convolutional layers are applied locally to a small neighborhood of the input, which is a fixed size regardless of the input size. However, in GATs, the attention mechanism allows the network to adaptively weigh the importance of different parts of the graph based on the specific input and task at hand, this gives the network a better understanding of the underlying structure of the input graph. Additionally, GATs also allow the network to acquire complex and abstract properties of the graph by stacking multiple attention layers, this improves the network's ability to adapt to unknown data. Furthermore, GATs also can be trained with Bayesian methods, which allows to model uncertainty in the predictions by placing probability distributions over the network's parameters, this can provide a measure of uncertainty for each prediction, making the network more certain about the predictions it is making.
In summary, GATs are more certain with its predictions because it uses self-attention mechanisms that allow the network to concentrate on the most significant features of the graph for each prediction, it also allows the network to learn more complex and abstract features of the graph and also can be trained with Bayesian methods to provide uncertainty estimates for the predictions.

Chapter 8

Future Research

8.1 Uncertainty Analysis

In this thesis work, we focus only on investigating characteristics of neural networks that have not before been investigated and have not been analyzed in terms of uncertainty. Nevertheless, this thesis only provides extensive information about classification and regression problems.
Deep learning with reinforcement is something that we are looking forward to working with. Due to the fact that uncertainty assessment is of utmost significance in applications of deep reinforcement learning[163]–[165].
In addition, we are keen to investigate more approaches to assessing uncertainty, evaluate how they compare favorably against MCD, and determine which method is more effective in a given scenario based on the outcomes. Additionally, various types of perturbation that might produce uncertainty will be investigated as part of this project. In order to come up with a novel approach to assessing uncertainty, we may combine the concepts of perturbation and the Monte Carlo simulation. Even though there are a number of research concerning the concept that was just described [166]–[170], we are hoping to come across one that concentrates on the data as well as the model. MCD only perturbs the model. Additional research might be conducted on the subject.

8.2 Reliable and Robust Models

We propose two novel architectures in this thesis. The scope of UnIC-Net will be broadened to include other medical images that share features with cell-like images. Currently, we are dealing with a variety of medical data that has already shown some interesting insights.
Aside from this, we are looking forward to delving further into NLP tasks, which has high ambiguity issues [171]. Wang et al. [172] examine the problem of mislabeling from the novel point of view of aleatoric uncertainty, which explains the underlying unpredictability of missing data. This may be a fascinating subject to investigate using MCD or any other strategy that can handle this problem more efficiently and effectively.

Chapter 9

Conclusion

When conducting AI practices in the real world, determining the level of uncertainty is really important. The quantification of uncertainty is of utmost importance when it comes to enhancing the trustworthiness of machine intelligence and lowering the risk profile associated with working with such technologies. It has evolved into an essential component of deep learning methods, particularly in deep learning applications that are used in the real world. Our research concentrates not only on the analysis of MCD uncertainty quantification but also on the enhancement of prediction performance and the identification of the factors that are significantly responsible for the uncertainty of a model. We include the uncertainty analysis of the most recent and SOTA techniques, such as INN, BERT, XLNet, ViT, CCT, SWT, GCN, GAT, PPNP and APPNP. Additionally, we suggest two distinct models, one of which is based on CNN, and the other which is a combination of INN and CNN. We present a visual analysis as well as a comparative examination of these two, and based on both, we are able to deduce that the performance of these two is superior based on all the datasets that were utilized. As a result, we have an up-to-date overview of uncertainty quantification using MCD, including the two reliable architectures, that we have proposed for use in three distinct tasks. This leads us to the conclusion that we have investigated recent developments in NN structures.

We believe that our extensive study will show researchers and practitioners how a model's level of confidence and raw performance can vary and prompt them to consider the significance of measuring uncertainty while utilizing DL models. This will result in more secure applications for forecasting in a variety of DL fields.

Bibliography

[1] E. Begoli, T. Bhattacharya, and D. F. Kusnezov, "The need for uncertainty quantification in machine-assisted medical decision making," *Nature Machine Intelligence*, vol. 1, no. 1, Jan. 2019. DOI: 10.1038/s42256-018-0004-1.

[2] H. F. Calvo-Pardo, T. Mancini, and J. Olmo, "Neural network models for empirical finance," *Journal of Risk and Financial Management*, vol. 13, no. 11, 2020, ISSN: 1911-8074. DOI: 10.3390/jrfm13110265. [Online]. Available: https://www.mdpi.com/1911-8074/13/11/265.

[3] Z. Ghahramani, "Probabilistic machine learning and artificial intelligence," *Nature*, vol. 521, no. 7553, pp. 452–459, 2015.

[4] J. Heaton, "Ian goodfellow, yoshua bengio, and aaron courville: Deep learning," *Genetic Programming and Evolvable Machines*, vol. 19, no. 1, pp. 305–307, Jun. 2018, ISSN: 1573-7632. DOI: 10.1007/s10710-017-9314-z. [Online]. Available: https://doi.org/10.1007/s10710-017-9314-z.

[5] K. Shridhar, F. Laumann, and M. Liwicki, "A comprehensive guide to bayesian convolutional neural network with variational inference," *CoRR*, vol. abs/1901.02731, 2019. arXiv: 1901.02731. [Online]. Available: http://arxiv.org/abs/1901.02731.

[6] W. J. Maddox, T. Garipov, P. Izmailov, D. P. Vetrov, and A. G. Wilson, "A simple baseline for bayesian uncertainty in deep learning," *CoRR*, vol. abs/1902.02476, 2019. arXiv: 1902.02476. [Online]. Available: http://arxiv.org/abs/1902.02476.

[7] A. Kendall and Y. Gal, *What uncertainties do we need in bayesian deep learning for computer vision?* 2017. DOI: 10.48550/ARXIV.1703.04977. [Online]. Available: https://arxiv.org/abs/1703.04977.

[8] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, *On calibration of modern neural networks*, 2017. DOI: 10.48550/ARXIV.1706.04599. [Online]. Available: https://arxiv.org/abs/1706.04599.

[9] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, *On calibration of modern neural networks*, 2017. DOI: 10.48550/ARXIV.1706.04599. [Online]. Available: https://arxiv.org/abs/1706.04599.

[10] J. Gawlikowski, C. R. N. Tassi, M. Ali, *et al.*, *A survey of uncertainty in deep neural networks*, 2021. DOI: 10.48550/ARXIV.2107.03342. [Online]. Available: https://arxiv.org/abs/2107.03342.

[11] B. T. Phan, "Bayesian deep learning and uncertainty in computer vision," M.S. thesis, University of Waterloo, 2019.

[12] Y. Gal, "Uncertainty in deep learning," Ph.D. dissertation, University of Cambridge, 2016.

[13] E. Hüllermeier and W. Waegeman, "Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods," *Machine Learning*, vol. 110, no. 3, pp. 457–506, 2021.

[14] J. Mukhoti and Y. Gal, "Evaluating bayesian deep learning methods for semantic segmentation," *arXiv preprint arXiv:1811.12709*, 2018.

[15] A. Malinin, "Uncertainty estimation in deep learning with application to spoken language assessment," Ph.D. dissertation, University of Cambridge, 2019.

[16] H. D. Kabir, M. Abdar, A. Khosravi, *et al.*, "Spinalnet: Deep neural network with gradual input," *IEEE Transactions on Artificial Intelligence*, 2022.

[17] X. Wang, Y. Zhao, and F. Pourpanah, "Recent advances in deep learning," *International Journal of Machine Learning and Cybernetics*, vol. 11, no. 4, pp. 747–750, 2020.

[18] R. M. Neal, *Bayesian learning for neural networks*. Springer Science & Business Media, 2012, vol. 118.

[19] A. Graves, "Practical variational inference for neural networks," in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds., vol. 24, Curran Associates, Inc., 2011. [Online]. Available: https://proceedings.neurips.cc/paper/2011/file/7eb3c8be3d411e8ebfab08eba5f49632-Paper.pdf.

[20] W. Maddox, T. Garipov, P. Izmailov, D. Vetrov, and A. G. Wilson, *A simple baseline for bayesian uncertainty in deep learning*, 2019. DOI: 10.48550/ARXIV.1902.02476. [Online]. Available: https://arxiv.org/abs/1902.02476.

[21] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," ser. ICML'16, New York, NY, USA: JMLR.org, 2016, pp. 1050–1059.

[22] F. Verdoja and V. Kyrki, *Notes on the behavior of mc dropout*, 2020. DOI: 10.48550/ARXIV.2008.02627. [Online]. Available: https://arxiv.org/abs/2008.02627.

[23] C. Stoean, R. Stoean, M. Atencia, *et al.*, "Automated detection of presymptomatic conditions in spinocerebellar ataxia type 2 using monte carlo dropout and deep neural network techniques with electrooculogram signals," *Sensors*, vol. 20, no. 11, 2020, ISSN: 1424-8220. DOI: 10.3390/s20113032. [Online]. Available: https://www.mdpi.com/1424-8220/20/11/3032.

[24] C. Leibig, V. Allken, M. S. Ayhan, P. Berens, and S. Wahl, "Leveraging uncertainty information from deep neural networks for disease detection," *Scientific Reports*, vol. 7, Dec. 2017. DOI: 10.1038/s41598-017-17876-z.

[25] R. Camarasa, D. Bos, J. Hendrikse, *et al.*, "Quantitative comparison of monte-carlo dropout uncertainty measures for multi-class segmentation," English, in *Uncertainty for Safe Utilization of Machine Learning in Medical Imaging, and Graphs in Biomedical Image Analysis - 2nd International Workshop, UNSURE 2020, and 3rd International Workshop, GRAIL 2020, Held in Conjunction with MICCAI 2020, Proceedings*, C. Sudre, H. Fehri, T. Arbel, *et al.*, Eds., ser. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Germany: Springer Science+Business Media, 2020, pp. 32–41, ISBN: 9783030603649. DOI: 10.1007/978-3-030-60365-6_4.

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html.

[27] P. Baldi and P. Sadowski, "The dropout learning algorithm," *Artificial Intelligence*, vol. 210, pp. 78–122, 2014, ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2014.02.004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370214000216.

[28] L. Ma and J. Kaewell, *Fast monte carlo dropout and error correction for radio transmitter classification*, 2020. DOI: 10.48550/ARXIV.2001.11963. [Online]. Available: https://arxiv.org/abs/2001.11963.

[29] B. Lakshminarayanan, A. Pritzel, and C. Blundell, *Simple and scalable predictive uncertainty estimation using deep ensembles*, 2016. DOI: 10.48550/ARXIV.1612.01474. [Online]. Available: https://arxiv.org/abs/1612.01474.

[30] J. Haas and B. Rabus, "Uncertainty estimation for deep learning-based segmentation of roads in synthetic aperture radar imagery," *Remote Sensing*, vol. 13, no. 8, 2021, ISSN: 2072-4292. DOI: 10.3390/rs13081472. [Online]. Available: https://www.mdpi.com/2072-4292/13/8/1472.

[31] M. F. Islam, S. Zabeen, M. H. K. Mehedi, S. Iqbal, and A. A. Rasel, "Monte carlo dropout for uncertainty analysis and ecg trace image classification," in *Structural, Syntactic, and Statistical Pattern Recognition*, A. Krzyzak, C. Y. Suen, A. Torsello, and N. Nobile, Eds., Cham: Springer International Publishing, 2022, pp. 173–182, ISBN: 978-3-031-23028-8.

[32] M. Abdar, F. Pourpanah, S. Hussain, *et al.*, "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," *Information Fusion*, vol. 76, pp. 243–297, Dec. 2021. DOI: 10.1016/j.inffus.2021.05.008. [Online]. Available: https://doi.org/10.1016%5C%2Fj.inffus.2021.05.008.

[33] D. J. C. MacKay, "Bayesian Interpolation," *Neural Computation*, vol. 4, no. 3, pp. 415–447, May 1992, ISSN: 0899-7667. DOI: 10.1162/neco.1992.4.3.415. eprint: https://direct.mit.edu/neco/article-pdf/4/3/415/812340/neco.1992.4.3.415.pdf. [Online]. Available: https://doi.org/10.1162/neco.1992.4.3.415.

[34] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, *On large-batch training for deep learning: Generalization gap and sharp minima*, 2016. DOI: 10.48550/ARXIV.1609.04836. [Online]. Available: https://arxiv.org/abs/1609.04836.

[35] T. Chen, E. B. Fox, and C. Guestrin, *Stochastic gradient hamiltonian monte carlo*, 2014. DOI: 10.48550/ARXIV.1402.4102. [Online]. Available: https://arxiv.org/abs/1402.4102.

[36] M. Welling and Y. W. Teh, "Bayesian learning via stochastic gradient langevin dynamics," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ser. ICML'11, Bellevue, Washington, USA: Omnipress, 2011, pp. 681–688, ISBN: 9781450306195.

[37] D. P. Kingma and M. Welling, *Auto-encoding variational bayes*, 2013. DOI: 10.48550/ARXIV.1312.6114. [Online]. Available: https://arxiv.org/abs/1312.6114.

[38] S. Mandt, M. D. Hoffman, and D. M. Blei, "Stochastic gradient descent as approximate bayesian inference," 2017. DOI: 10.48550/ARXIV.1704.04289. [Online]. Available: https://arxiv.org/abs/1704.04289.

[39] X. Chen, J. D. Lee, X. T. Tong, and Y. Zhang, *Statistical inference for model parameters in stochastic gradient descent*, 2016. DOI: 10.48550/ARXIV.1610.08637. [Online]. Available: https://arxiv.org/abs/1610.08637.

[40] R. Hu, Q. Huang, S. Chang, H. Wang, and J. He, "The MBPEP: A deep ensemble pruning algorithm providing high quality uncertainty prediction," *Applied Intelligence*, vol. 49, no. 8, pp. 2942–2955, Feb. 2019. DOI: 10.1007/s10489-019-01421-8. [Online]. Available: https://doi.org/10.1007%2Fs10489-019-01421-8.

[41] V. Kuleshov, N. Fenner, and S. Ermon, *Accurate uncertainties for deep learning using calibrated regression*, 2018. DOI: 10.48550/ARXIV.1807.00263. [Online]. Available: https://arxiv.org/abs/1807.00263.

[42] M. Sensoy, L. Kaplan, and M. Kandemir, "Evidential deep learning to quantify classification uncertainty," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31, Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/a981f2b708044d6fb4a7%5C%5C1a1463242520-Paper.pdf.

[43] A. P. Dempster, "A generalization of bayesian inference," in *Classic Works of the Dempster-Shafer Theory of Belief Functions*, R. R. Yager and L. Liu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 73–104, ISBN: 978-3-540-44792-4. DOI: 10.1007/978-3-540-44792-4_4. [Online]. Available: https://doi.org/10.1007/978-3-540-44792-4_4.

[44] A. Jøsang, *Subjective Logic: A Formalism for Reasoning Under Uncertainty*, 1st. Springer Publishing Company, Incorporated, 2016, ISBN: 3319423355.

[45] V.-L. Nguyen, M. H. Shaker, and E. Hüllermeier, "How to measure uncertainty in uncertainty sampling for active learning," *Machine Learning*, vol. 111, no. 1, pp. 89–122, Jan. 2022, ISSN: 1573-0565. DOI: 10.1007/s10994-021-06003-9. [Online]. Available: https://doi.org/10.1007/s10994-021-06003-9.

[46] A. Shelmanov, D. Puzyrev, L. Kupriyanova, *et al.*, *Active learning for sequence tagging with deep pre-trained models and bayesian uncertainty estimates*, 2021. DOI: 10.48550/ARXIV.2101.08133. [Online]. Available: https://arxiv.org/abs/2101.08133.

[47] A. Shapeev, K. Gubaev, E. Tsymbalov, and E. Podryabinkin, "Active learning and uncertainty estimation," in *Machine Learning Meets Quantum Physics*, K. T. Schütt, S. Chmiela, O. A. von Lilienfeld, A. Tkatchenko, K. Tsuda, and K.-R. Müller, Eds. Cham: Springer International Publishing, 2020, pp. 309–329, ISBN: 978-3-030-40245-7. DOI: 10.1007/978-3-030-40245-7_15. [Online]. Available: https://doi.org/10.1007/978-3-030-40245-7_15.

[48] Y. Tian, R. Yuan, D. Xue, *et al.*, "Role of uncertainty estimation in accelerating materials development via active learning," *Journal of Applied Physics*, vol. 128, no. 1, p. 014 103, 2020.

[49] D. Milanés-Hermosilla, R. Trujillo Codorniú, R. López-Baracaldo, *et al.*, "Monte carlo dropout for uncertainty estimation and motor imagery classification," *Sensors*, vol. 21, no. 21, 2021, ISSN: 1424-8220. DOI: 10.3390/s21217241. [Online]. Available: https://www.mdpi.com/1424-8220/21/21/7241.

[50] M. Y. Avci, Z. Li, Q. Fan, S. Huang, B. Bilgic, and Q. Tian, *Quantifying the uncertainty of neural networks using monte carlo dropout for deep learning based quantitative mri*, 2021. DOI: 10.48550/ARXIV.2112.01587. [Online]. Available: https://arxiv.org/abs/2112.01587.

[51] P. Tabarisaadi, A. Khosravi, and S. Nahavandi, "Uncertainty-aware skin cancer detection: The element of doubt," *Computers in Biology and Medicine*, vol. 144, p. 105 357, 2022, ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2022.105357. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010482522001494.

[52] Y.-C. Kim, K. R. Kim, and Y. H. Choe, "Automatic myocardial segmentation in dynamic contrast enhanced perfusion mri using monte carlo dropout in an encoder-decoder convolutional neural network," *Computer Methods and Programs in Biomedicine*, vol. 185, p. 105 150, 2020, ISSN: 0169-2607. DOI: https://doi.org/10.1016/j.cmpb.2019.105150. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169260719311538.

[53] P. Chagas, L. Souza, I. Pontes, *et al.*, "Uncertainty-aware membranous nephropathy classification: A monte-carlo dropout approach to detect how certain is the model," *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, vol. 0, no. 0, pp. 1–11, 2022. DOI: 10.1080/21681163.2022.2029573. eprint: https://doi.org/10.1080/21681163.2022.2029573. [Online]. Available: https://doi.org/10.1080/21681163.2022.2029573.

[54] M. Gour and S. Jain, "Uncertainty-aware convolutional neural network for covid-19 x-ray images classification," *Computers in Biology and Medicine*, vol. 140, p. 105 047, 2022, ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2021.105047. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010482521008416.

[55] T. Xia, J. Han, and C. Mascolo, *Benchmarking uncertainty quantification on biosignal classification tasks under dataset shift*, 2021. DOI: 10.48550/ARXIV.2112.09196. [Online]. Available: https://arxiv.org/abs/2112.09196.

[56] D. Xiao, C. Qin, J. Ge, P. Xia, Y. Huang, and C. Liu, "Self-attention-based adaptive remaining useful life prediction for igbt with monte carlo dropout," *Knowledge-Based Systems*, vol. 239, p. 107 902, 2022, ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2021.107902. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705121010613.

[57] F. Laakom, J. Raitoharju, A. Iosifidis, J. Nikkanen, and M. Gabbouj, "Monte carlo dropout ensembles for robust illumination estimation," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–7. DOI: 10.1109/IJCNN52387.2021.9534314.

[58] N. Heidari and A. Iosifidis, "Progressive spatio-temporal bilinear network with monte carlo dropout for landmark-based facial expression recognition with uncertainty estimation," in *2021 IEEE 23rd International Workshop on Multimedia Signal Processing (MMSP)*, 2021, pp. 1–6. DOI: 10.1109/MMSP53017.2021.9733455.

[59] T. Myojin, S. Hashimoto, K. Mori, K. Sugawara, and N. Ishihama, "Improving reliability of object detection for lunar craters using monte carlo dropout," in *Artificial Neural Networks and Machine Learning – ICANN 2019: Image Processing*, I. V. Tetko, V. Kůrková, P. Karpov, and F. Theis, Eds., Cham: Springer International Publishing, 2019, pp. 68–80, ISBN: 978-3-030-30508-6.

[60] H.-Y. Yang and L. H. Staib, "Leukemic b-lymphoblast cell detection with monte carlo dropout ensemble models," in *ISBI 2019 C-NMC Challenge: Classification in Cancer Cell Imaging*, A. Gupta and R. Gupta, Eds., Singapore: Springer Singapore, 2019, pp. 123–130, ISBN: 978-981-15-0798-4.

[61] N. P. M. and A. G. Ramakrishnan, *Using monte carlo dropout for non-stationary noise reduction from speech*, 2018. DOI: 10.48550/ARXIV.1808.09432. [Online]. Available: https://arxiv.org/abs/1808.09432.

[62] C. Junhwan, O. Seokmin, and B. Joongmoo, "Uncertainty estimation in avo inversion using bayesian dropout based deep learning," *Journal of Petroleum Science and Engineering*, vol. 208, p. 109 288, 2022, ISSN: 0920-4105. DOI: https://doi.org/10.1016/j.petrol.2021.109288. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0920410521009414.

[63] T. Fisher, H. Gibson, Y. Liu, *et al.*, "Uncertainty-aware interpretable deep learning for slum mapping and monitoring," *Remote Sensing*, vol. 14, no. 13, 2022, ISSN: 2072-4292. DOI: 10.3390/rs14133072. [Online]. Available: https://www.mdpi.com/2072-4292/14/13/3072.

[64] B. Ghoshal and A. Tucker, "Estimating uncertainty and interpretability in deep learning for coronavirus (covid-19) detection," *ArXiv*, vol. abs/2003.10769, 2020.

[65] J. Padarian, B. Minasny, and A. McBratney, "Assessing the uncertainty of deep learning soil spectral models using monte carlo dropout," *Geoderma*, vol. 425, p. 116 063, 2022, ISSN: 0016-7061. DOI: https://doi.org/10.1016/j.geoderma.2022.116063. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0016706122003706.

[66] S. Jain and S. P.K, "Monte carlo dropout based batchensemble for improving uncertainty estimation," in *Proceedings of the 6th Joint International Conference on Data Science Management of Data (10th ACM IKDD CODS and 28th COMAD)*, ser. CODS-COMAD '23, Mumbai, India: Association for Computing Machinery, 2023, p. 138, ISBN: 9781450397971. DOI: 10.1145/3570991.3571038. [Online]. Available: https://doi.org/10.1145/3570991.3571038.

[67] P. Carreno-Medrano, D. Kulić, and M. Burke, *Adapting neural models with sequential monte carlo dropout*, 2022. DOI: 10.48550/ARXIV.2210.15779. [Online]. Available: https://arxiv.org/abs/2210.15779.

[68] C.-C. Chang, "Bayesian neural networks for reversible steganography," *IEEE Access*, vol. 10, pp. 36 327–36 334, 2022. DOI: 10.1109/ACCESS.2022.3159911.

[69] C.-S. Cheng, A. H. Behzadan, and A. Noshadravan, "Uncertainty-aware convolutional neural network for explainable artificial intelligence-assisted disaster damage assessment," *Structural Control and Health Monitoring*, vol. 29, no. 10, e3019, 2022. DOI: https://doi.org/10.1002/stc.3019. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/stc.3019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/stc.3019.

[70] J. Van Landeghem, M. Blaschko, B. Anckaert, and M.-F. Moens, "Benchmarking scalable predictive uncertainty in text classification," *IEEE Access*, vol. 10, pp. 43 703–43 737, 2022. DOI: 10.1109/ACCESS.2022.3168734.

[71] A. Shelmanov, E. Tsymbalov, D. Puzyrev, K. Fedyanin, A. Panchenko, and M. Panov, "How certain is your Transformer?" In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, Online: Association for Computational Linguistics, Apr. 2021, pp. 1833–1840. DOI: 10.18653/v1/2021.eacl-main.157. [Online]. Available: https://aclanthology.org/2021.eacl-main.157.

[72] Y. Hu and L. Khan, "Uncertainty-aware reliable text classification," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery amp; Data Mining*, ser. KDD '21, Virtual Event, Singapore: Association for Computing Machinery, 2021, pp. 628–636, ISBN: 9781450383325. DOI: 10.1145/3447548.3467382. [Online]. Available: https://doi.org/10.1145/3447548.3467382.

[73] Z. Wang, J.-Q. Zheng, and I. Voiculescu, "An uncertainty-aware transformer for mri cardiac semantic segmentation via mean teachers," in *Medical Image Understanding and Analysis*, G. Yang, A. Aviles-Rivero, M. Roberts, and C.-B. Schönlieb, Eds., Cham: Springer International Publishing, 2022, pp. 494–507, ISBN: 978-3-031-12053-4.

[74] H. Zhao, Q. Wang, Z. Jia, Y. Chen, and J. Zhang, "Bayesian based facial expression recognition transformer model in uncertainty," in *2021 International Conference on Digital Society and Intelligent Systems (DSInS)*, 2021, pp. 157–161. DOI: 10.1109/DSInS54396.2021.9670628.

[75] F. Yang, Q. Zhai, X. Li, *et al.*, "Uncertainty-guided transformer reasoning for camouflaged object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2021, pp. 4146–4155.

[76] V. P. Dwivedi, C. K. Joshi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson, "Benchmarking graph neural networks," 2020. DOI: 10.48550/ARXIV.2003.00982. [Online]. Available: https://arxiv.org/abs/2003.00982.

[77] R. Seoh, *Qualitative analysis of monte carlo dropout*, 2020. DOI: 10.48550/ARXIV.2007.01720. [Online]. Available: https://arxiv.org/abs/2007.01720.

[78] J. Sicking, M. Akila, T. Wirtz, S. Houben, and A. Fischer, *Characteristics of monte carlo dropout in wide neural networks*, 2020. DOI: 10.48550/ARXIV.2007.05434. [Online]. Available: https://arxiv.org/abs/2007.05434.

[79] K. Miok, D. Nguyen-Doan, D. Zaharie, and M. Robnik-Šikonja, "Generating data using monte carlo dropout," in *2019 IEEE 15th International Conference on Intelligent Computer Communication and Processing (ICCP)*, 2019, pp. 509–515. DOI: 10.1109/ICCP48234.2019.8959787.

[80] A. Lemay, K. Hoebel, C. P. Bridge, *et al.*, "Improving the repeatability of deep learning models with monte carlo dropout," *npj Digital Medicine*, vol. 5, no. 1, p. 174, Nov. 2022, ISSN: 2398-6352. DOI: 10.1038/s41746-022-00709-3. [Online]. Available: https://doi.org/10.1038/s41746-022-00709-3.

[81] P. Goel and L. Chen, "On the robustness of monte carlo dropout trained with noisy labels," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, Jun. 2021, pp. 2219–2228.

[82] K. Pace and R. Barry, "Sparse spatial autoregressions," *Statistics Probability Letters*, vol. 33, no. 3, pp. 291–297, 1997. [Online]. Available: https://EconPapers.repec.org/RePEc:eee:stapro:v:33:y:1997:i:3:p:291-297.

[83] X. Zhang, J. Zhao, and Y. LeCun, "Character-level Convolutional Networks for Text Classification," *arXiv:1509.01626 [cs]*, Sep. 2015. arXiv: 1509.01626 [cs].

[84] A. H. Khan, M. Hussain, and M. K. Malik, "Ecg images dataset of cardiac and covid-19 patients," *Data in Brief*, vol. 34, p. 106 762, 2021, ISSN: 2352-3409. DOI: https://doi.org/10.1016/j.dib.2021.106762. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352340921000469.

[85] M. J. NKENGUE, *Ecg_image_cropped*, https://www.kaggle.com/datasets/marcjuniornkengue/ecg-image-cropped, version 4, 2022.

[86] M. Traina, S. Hernandez, D. Sanchez, *et al.*, "Prevalence of chagas disease in a u.s. population of latin american immigrants with conduction abnormalities on electrocardiogram," *PLOS Neglected Tropical Diseases*, vol. 11, e0005244, Jan. 2017. DOI: 10.1371/journal.pntd.0005244.

[87] S. Rajaraman, S. K. Antani, M. Poostchi, *et al.*, "Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images," en, *PeerJ*, vol. 6, e4568, Apr. 2018.

[88] N. C. F. Codella, V. Rotemberg, P. Tschandl, *et al.*, "Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (ISIC)," *CoRR*, vol. abs/1902.03368, 2019. arXiv: 1902.03368. [Online]. Available: http://arxiv.org/abs/1902.03368.

[89] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[90] A. McCallum, *Cora Dataset*, version V1, 2017. DOI: 10.18738/T8/HUIG48. [Online]. Available: https://doi.org/10.18738/T8/HUIG48.

[91] S. J. Nowlan and G. E. Hinton, "Simplifying neural networks by soft weight-sharing," *Neural Computation*, vol. 4, pp. 473–493, 1992.

[92] H. Y. Xiong, Y. Barash, and B. J. Frey, "Bayesian prediction of tissue-regulated splicing using RNA sequence and cellular context," en, *Bioinformatics*, vol. 27, no. 18, pp. 2554–2562, Jul. 2011.

[93] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using markov chain monte carlo," in *International Conference on Machine Learning*, 2008.

[94] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[95] E. Momeni, R. Nazir, D. J. Armaghani, and H. Maizir, "Prediction of pile bearing capacity using a hybrid genetic algorithm-based ann," *Measurement*, vol. 57, pp. 122–131, 2014.

[96] A. R. Bunawan, E. Momeni, D. J. Armaghani, A. S. A. Rashid, *et al.*, "Experimental and intelligent techniques to estimate bearing capacity of cohesive soft soils reinforced with soil-cement columns," *Measurement*, vol. 124, pp. 529–538, 2018.

[97] G.-G. Wang, L. Guo, A. H. Gandomi, G.-S. Hao, and H. Wang, "Chaotic krill herd algorithm," *Information Sciences*, vol. 274, pp. 17–34, 2014.

[98] G. Wang, L. Guo, H. Duan, H. Wang, L. Liu, and M. Shao, "Hybridizing harmony search with biogeography based optimization for global numerical optimization," *Journal of Computational and Theoretical Nanoscience*, vol. 10, no. 10, pp. 2312–2322, 2013.

[99] G. Wang, L. Guo, H. Wang, H. Duan, L. Liu, and J. Li, "Incorporating mutation scheme into krill herd algorithm for global numerical optimization," *Neural Computing and Applications*, vol. 24, no. 3, pp. 853–871, 2014.

[100] P. G. Asteris and K. G. Kolovos, "Self-compacting concrete strength prediction using surrogate models," *Neural Computing and Applications*, vol. 31, no. 1, pp. 409–424, 2019.

[101] P. Asteris, K. Kolovos, M. Douvika, and K. Roinos, "Prediction of self-compacting concrete strength using artificial neural networks," *European Journal of Environmental and Civil Engineering*, vol. 20, no. sup1, s102–s122, 2016.

[102] P. G. Asteris, A. K. Tsaris, L. Cavaleri, *et al.*, "Prediction of the fundamental period of infilled rc frame structures using artificial neural networks," *Computational intelligence and neuroscience*, vol. 2016, 2016.

[103] P. G. Asteris, P. C. Roussis, and M. G. Douvika, "Feed-forward neural network prediction of the mechanical properties of sandcrete materials," *Sensors*, vol. 17, no. 6, p. 1344, 2017.

[104] P. G. Asteris, A. Moropoulou, A. D. Skentou, *et al.*, "Stochastic vulnerability assessment of masonry structures: Concepts, modeling and restoration aspects," *Applied Sciences*, vol. 9, no. 2, p. 243, 2019.

[105] P. Psyllaki, K. Stamatiou, I. Iliadis, A. Mourlas, P. Asteris, and N. Vaxevanidis, "Surface treatment of tool steels against galling failure," in *MATEC web of conferences*, EDP Sciences, vol. 188, 2018, p. 04 024.

[106] G. M. Kotsovou, D. M. Cotsovos, and N. D. Lagaros, "Assessment of rc exterior beam-column joints based on artificial neural networks and other methods," *Engineering Structures*, vol. 144, pp. 1–18, 2017.

[107] A. Ahmad, G. Kotsovou, D. M. Cotsovos, and N. D. Lagaros, "Assessing the accuracy of rc design code predictions through the use of artificial neural networks," *International Journal of Advanced Structural Engineering*, vol. 10, no. 4, pp. 349–365, 2018.

[108] E. Momeni, D. J. Armaghani, M. Hajihassani, and M. F. M. Amin, "Prediction of uniaxial compressive strength of rock samples using hybrid particle swarm optimization-based artificial neural networks," *Measurement*, vol. 60, pp. 50–63, 2015.

[109] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[110] B. D. Ripley, *Pattern recognition and neural networks*. Cambridge university press, 2007.

[111] G. Zhang, B. E. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks:: The state of the art," *International journal of forecasting*, vol. 14, no. 1, pp. 35–62, 1998.

[112] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[113] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[114] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[115] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational*. MIT press, 1988.

[116] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for boltzmann machines," *Cognitive science*, vol. 9, no. 1, pp. 147–169, 1985.

[117] K. Fukushima, "Neocognitron: A hierarchical neural network capable of visual pattern recognition," *Neural networks*, vol. 1, no. 2, pp. 119–130, 1988.

[118] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[119] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[120] B. Widrow and M. A. Lehr, "30 years of adaptive neural networks: Perceptron, madaline, and backpropagation," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1415–1442, 1990.

[121] B. Cheng and D. M. Titterington, "Neural networks: A review from a statistical perspective," *Statistical science*, pp. 2–30, 1994.

[122] D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *The Journal of physiology*, vol. 148, no. 3, p. 574, 1959.

[123] F. Ciompi, B. de Hoop, S. J. van Riel, *et al.*, "Automatic classification of pulmonary peri-fissural nodules in computed tomography using an ensemble of 2d views and a convolutional neural network out-of-the-box," *Medical Image Analysis*, vol. 26, no. 1, pp. 195–202, 2015, ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2015.08.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1361841515001255.

[124] D. Li, J. Hu, C. Wang, *et al.*, *Involution: Inverting the inherence of convolution for visual recognition*, 2021. DOI: 10.48550/ARXIV.2103.06255. [Online]. Available: https://arxiv.org/abs/2103.06255.

[125] M. Kubat, "Neural networks: A comprehensive foundation by simon haykin, macmillan, 1994, isbn 0-02-352781-7.," *The Knowledge Engineering Review*, vol. 13, no. 4, pp. 409–412, 1999.

[126] H. Hettiarachchi and T. Ranasinghe, "Emoji powered capsule network to detect type and target of offensive posts in social media," in *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, 2019, pp. 474–480.

[127] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[128] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2018. DOI: 10.48550/ARXIV.1810.04805. [Online]. Available: https://arxiv.org/abs/1810.04805.

[129] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, *Xlnet: Generalized autoregressive pretraining for language understanding*, 2019. DOI: 10.48550/ARXIV.1906.08237. [Online]. Available: https://arxiv.org/abs/1906.08237.

[130] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, Eds., vol. 26, Curran Associates, Inc., 2013. [Online]. Available: https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

[131] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Baltimore, Maryland: Association for Computational Linguistics, Jun. 2014, pp. 655–665. DOI: 10.3115/v1/P14-1062. [Online]. Available: https://aclanthology.org/P14-1062.

[132] W. Wang, B. Bi, M. Yan, *et al.*, *Structbert: Incorporating language structures into pre-training for deep language understanding*, 2019. DOI: 10.48550/ARXIV.1908.04577. [Online]. Available: https://arxiv.org/abs/1908.04577.

[133] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, *An image is worth 16x16 words: Transformers for image recognition at scale*, 2020. DOI: 10.48550/ARXIV.2010.11929. [Online]. Available: https://arxiv.org/abs/2010.11929.

[134] I. Bello, B. Zoph, A. Vaswani, J. Shlens, and Q. V. Le, *Attention augmented convolutional networks*, 2019. DOI: 10.48550/ARXIV.1904.09925. [Online]. Available: https://arxiv.org/abs/1904.09925.

[135] F. Wang, M. Jiang, C. Qian, *et al.*, *Residual attention network for image classification*, 2017. DOI: 10.48550/ARXIV.1704.06904. [Online]. Available: https://arxiv.org/abs/1704.06904.

[136] P. Ramachandran, N. Parmar, A. Vaswani, I. Bello, A. Levskaya, and J. Shlens, *Stand-alone self-attention in vision models*, 2019. DOI: 10.48550/ARXIV.1906.05909. [Online]. Available: https://arxiv.org/abs/1906.05909.

[137] Z. Liu, Y. Lin, Y. Cao, *et al.*, *Swin transformer: Hierarchical vision transformer using shifted windows*, 2021. DOI: 10.48550/ARXIV.2103.14030. [Online]. Available: https://arxiv.org/abs/2103.14030.

[138] A. Hassani, S. Walton, N. Shah, A. Abuduweili, J. Li, and H. Shi, *Escaping the big data paradigm with compact transformers*, 2021. DOI: 10.48550/ARXIV.2104.05704. [Online]. Available: https://arxiv.org/abs/2104.05704.

[139] F. Errica, M. Podda, D. Bacciu, and A. Micheli, *A fair comparison of graph neural networks for graph classification*, 2019. DOI: 10.48550/ARXIV.1912.09893. [Online]. Available: https://arxiv.org/abs/1912.09893.

[140] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009. DOI: 10.1109/TNN.2008.2005605.

[141] S. Chen, A. Sandryhaila, J. M. Moura, and J. Kovačević, "Signal recovery on graphs: Variation minimization," *IEEE Transactions on Signal Processing*, vol. 63, no. 17, pp. 4609–4624, 2015.

[142] S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević, "Discrete signal processing on graphs: Sampling theory," *IEEE transactions on signal processing*, vol. 63, no. 24, pp. 6510–6523, 2015.

[143] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[144] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data (2015)," *arXiv preprint arXiv:1506.05163*, 2015.

[145] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in neural information processing systems*, vol. 29, 2016.

[146] M. Welling and T. N. Kipf, "Semi-supervised classification with graph convolutional networks," in *J. International Conference on Learning Representations (ICLR 2017)*, 2016.

[147] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, *Graph attention networks*, 2017. DOI: 10.48550/ARXIV.1710.10903. [Online]. Available: https://arxiv.org/abs/1710.10903.

[148] W. L. Hamilton, R. Ying, and J. Leskovec, *Inductive representation learning on large graphs*, 2017. DOI: 10.48550/ARXIV.1706.02216. [Online]. Available: https://arxiv.org/abs/1706.02216.

[149] J. Gasteiger, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," 2018. DOI: 10.48550/ARXIV.1810.05997. [Online]. Available: https://arxiv.org/abs/1810.05997.

[150] E. ırmak, "Covid-19 disease diagnosis from paper-based ecg trace image data using a novel convolutional neural network model," *Physical and Engineering Sciences in Medicine*, vol. 45, pp. 167–179, Mar. 2022. DOI: 10.1007/s13246-022-01102-w.

[151] N. Sobahi, A. Sengur, R.-S. Tan, and U. R. Acharya, "Attention-based 3d cnn with residual connections for efficient ecg-based covid-19 detection," *Computers in Biology and Medicine*, vol. 143, p. 105 335, 2022, ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2022.105335. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010482522001275.

[152] O. Attallah, "Ecg-biconet: An ecg-based pipeline for covid-19 diagnosis using bi-layers of deep features integration," *Computers in Biology and Medicine*, vol. 142, p. 105 210, 2022, ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2022.105210. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010482522000026.

[153] Z. Lan, S. Cai, X. He, and X. Wen, "Fixcaps: An improved capsules network for diagnosis of skin cancer," *IEEE Access*, vol. 10, pp. 76 261–76 267, 2022. DOI: 10.1109/ACCESS.2022.3181225.

[154] S. K. Datta, M. A. Shaikh, S. N. Srihari, and M. Gao, "Soft attention improves skin cancer classification performance," in *Interpretability of Machine Intelligence in Medical Image Computing, and Topological Data Analysis and Its Applications for Medical Data*, M. Reyes, P. Henriques Abreu, J. Cardoso, *et al.*, Eds., Cham: Springer International Publishing, 2021, pp. 13–23, ISBN: 978-3-030-87444-5.

[155] D. S. Charan, H. Nadipineni, S. Sahayam, and U. Jayaraman, *Method to classify skin lesions using dermoscopic images*, 2020. DOI: 10.48550/ARXIV.2008.09418. [Online]. Available: https://arxiv.org/abs/2008.09418.

[156] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[157] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2261–2269. DOI: 10.1109/CVPR.2017.243.

[158] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng, *Dual path networks*, 2017. DOI: 10.48550/ARXIV.1707.01629. [Online]. Available: https://arxiv.org/abs/1707.01629.

[159] Z. Liang, A. Powell, I. Ersoy, *et al.*, "Cnn-based image analysis for malaria diagnosis," in *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2016, pp. 493–496. DOI: 10.1109/BIBM.2016.7822567.

[160] B. Lambert, F. Forbes, A. Tucholka, S. Doyle, H. Dehaene, and M. Dojat, *Trustworthy clinical ai solutions: A unified review of uncertainty quantification in deep learning models for medical image analysis*, 2022. DOI: 10.48550/ARXIV.2210.03736. [Online]. Available: https://arxiv.org/abs/2210.03736.

[161] A. Hassani, S. Walton, N. Shah, A. Abuduweili, J. Li, and H. Shi, *Escaping the big data paradigm with compact transformers*, 2021. DOI: 10.48550/ARXIV.2104.05704. [Online]. Available: https://arxiv.org/abs/2104.05704.

[162] B. Chen, R. Wang, D. Ming, and X. Feng, *Vit-p: Rethinking data-efficient vision transformers from locality*, 2022. DOI: 10.48550/ARXIV.2203.02358. [Online]. Available: https://arxiv.org/abs/2203.02358.

[163] S. Shafaei, S. Kugele, M. H. Osman, and A. Knoll, "Uncertainty in machine learning: A safety perspective on autonomous driving," in *Computer Safety, Reliability, and Security*, B. Gallina, A. Skavhaug, E. Schoitsch, and F. Bitsch, Eds., Cham: Springer International Publishing, 2018, pp. 458–464, ISBN: 978-3-319-99229-7.

[164] J. Wu, Z. Huang, and C. Lv, "Uncertainty-aware model-based reinforcement learning: Methodology and application in autonomous driving," *IEEE Transactions on Intelligent Vehicles*, pp. 1–10, 2022. DOI: 10.1109/TIV.2022.3185159.

[165] J. Guo, U. Kurup, and M. Shah, "Is it safe to drive? an overview of factors, metrics, and datasets for driveability assessment in autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 8, pp. 3135–3151, 2020. DOI: 10.1109/TITS.2019.2926042.

[166] A. Rawat, M. Wistuba, and M.-I. Nicolae, *Adversarial phenomenon in the eyes of bayesian deep learning*, 2017. DOI: 10.48550/ARXIV.1711.08244. [Online]. Available: https://arxiv.org/abs/1711.08244.

[167] O. F. Tuna, F. O. Catak, and M. T. Eskil, "Exploiting epistemic uncertainty of the deep learning models to generate adversarial samples," *Multimedia Tools and Applications*, vol. 81, no. 8, pp. 11 479–11 500, Mar. 2022, ISSN: 1573-7721. DOI: 10.1007/s11042-022-12132-7. [Online]. Available: https://doi.org/10.1007/s11042-022-12132-7.

[168] H. Liu, R. Ji, J. Li, *et al.*, "Universal adversarial perturbation via prior driven uncertainty approximation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2019.

[169] K. Nobarani and S. E. Razavi, *Deep learning to advance the eigenspace perturbation method for turbulence model uncertainty quantification*, 2022. DOI: 10.48550/ARXIV.2202.12378. [Online]. Available: https://arxiv.org/abs/2202.12378.

[170] H. Zeng, Z. Yue, Y. Zhang, Z. Kou, L. Shang, and D. Wang, *On attacking out-domain uncertainty estimation in deep neural networks*, 2022. DOI: 10.48550/ARXIV.2210.02191. [Online]. Available: https://arxiv.org/abs/2210.02191.

[171] C. Beck, H. Booth, M. El-Assady, and M. Butt, "Representation problems in linguistic annotations: Ambiguity, variation, uncertainty, error and bias," in *Proceedings of the 14th Linguistic Annotation Workshop*, Barcelona, Spain: Association for Computational Linguistics, Dec. 2020, pp. 60–73. [Online]. Available: https://aclanthology.org/2020.law-1.6.

[172] C. Wang, F. Feng, Y. Zhang, Q. Wang, X. Hu, and X. He, *Rethinking missing data: Aleatoric uncertainty-aware recommendation*, 2022. DOI: 10.48550/ARXIV.2209.11679. [Online]. Available: https://arxiv.org/abs/2209.11679.