Uncertainty Estimation in Medical Imaging
MSc in Data Science
University of Bath

Hamish Hornby

September 2020

Uncertainty Estimation in Medical Imaging

Submitted by: Hamish Hornby

for the degree of MSc in Data Science of the
University of Bath
January 2015

COPYRIGHT

Declaration

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of MSc in Data Science in the Department of Computer Science. No portion of the work in this dissertation has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

Signed: Hamish Hornby

1 Abstract

Machine learning models used for medical diagnostics require accurate uncertainty estimation as overconfident incorrect decisions can have serious consequences, such as misdiagnosis. Deep ensembles have been proven to produce the most accurate uncertain estimation for deep learning problems which most medical imaging tasks tend to belong to. Deep ensembles however require significantly more computational resource for training and at test time, therefore it is not feasible for hospitals with limited computational resources to implement deep ensembles on some medical imaging tasks. This project considers methods of reducing these computational requirements while maintaining the accuracy and uncertainty estimation of deep ensembles. This is done by considering a multi-label classification task on Chest X-rays as this is one of the most commonly used diagnostic tests in healthcare. Two approaches are proposed by this project to reduce the computational requirements, the first of which is using a pretrained model as a starting point for each model in a deep ensemble. This approach not only halves the training time but results in superior performance in both accuracy and uncertainty estimation on the CheXpert dataset. The effect of transfer leaning on uncertainty estimation has not been well studied in literature therefore this approach can be considered a novel contribution. The second approach considers ensemble distribution distillation which has already been proposed in literature by Malinin, Mlodozeniec and Gales (2020) to distil an ensemble into a single model to reduce the computational resources required at test time. The approach proposed by Malinin, Mlodozeniec and Gales (2020) is not suitable for the multi-label classification task, this project proposes a novel contribution whereby ensemble distribution distillation is extended to multi-label classification tasks by parameterising a mixture of Beta distributions. This method is shown by this project to perform as well as the form of ensemble distribution distillation already proposed in literature.

2 Acknowledgements

I would like to thank my supervisor, Dr Olga Isupova for her help and encouragement throughout this project as well as my family for support during this project.

Contents

1 Abstract **4**

2 Acknowledgements **5**

3 Introduction **9**

4 Literature Review **11**
- 4.1 Types of Uncertainty . . . 11
- 4.2 Uncertainty Estimation in Deep Learning . . . 12
  - 4.2.1 Conventional Neural Networks . . . 12
  - 4.2.2 Bayesian Neural Networks . . . 13
  - 4.2.3 Monte Carlo Dropout . . . 14
  - 4.2.4 Deep Ensembles . . . 15
  - 4.2.5 Ensemble Distillation & Ensemble Distribution Distillation 18
- 4.3 Classification of Chest X-rays . . . 20
  - 4.3.1 Convolutional Neural Networks . . . 20
  - 4.3.2 State of The Art Models . . . 21
  - 4.3.3 Transfer Learning . . . 22
  - 4.3.4 Conditional Training . . . 22
- 4.4 Uncertainty Evaluation & Performance Metrics . . . 25
  - 4.4.1 ROC Curves . . . 25
  - 4.4.2 Ranking Based Methods . . . 25
  - 4.4.3 Calibration . . . 26
  - 4.4.4 Dispersion . . . 27
  - 4.4.5 Proper Scoring Rules . . . 28
  - 4.4.6 Patch Accuracy vs. Patch Uncertainty . . . 28
  - 4.4.7 Domain Shift & Out of Domain Inputs . . . 29
- 4.5 Methods of Reducing Uncertainty . . . 29

5 Objectives **29**

6 Problem Setup **30**
- 6.1 Data Preprocessing . . . 30
- 6.2 Deep Ensembles . . . 33
  - 6.2.1 Uncertainty Estimation from Deep Ensembles . . . 33
- 6.3 Ensemble Distribution Distillation . . . 35
  - 6.3.1 Multi-label Classification . . . 37
  - 6.3.2 Target Smoothing . . . 38
  - 6.3.3 Temperature Annealing . . . 38
  - 6.3.4 Uncertainty Estimation from Ensemble Distribution Distillation . . . 40

7 Data, Methodology & Training **41**
7.1 Data Partitioning, Cleaning & Augmentation 42
7.2 Model Architectures & Training 42
7.2.1 Deep Ensembles - Random Initialisation 43
7.2.2 Deep Ensembles - Pretrained 44
7.2.3 Ensemble Distribution Distillation 44

8 Results **45**
8.1 Accuracy 45
8.2 Uncertainty 46
8.2.1 Negative Log Likelihood 47
8.2.2 Expected Calibration Error 47
8.2.3 Patch Accuracy vs. Patch Uncertainty 48
8.2.4 Out of Distribution Detection 49

9 Conclusions & Further Work **50**

10 Appendices **53**
10.1 ROC Curves 53
10.1.1 Deep Ensemble - Random Initialisation 53
10.1.2 Deep Ensemble - Pretrained 54
10.1.3 Ensemble Distribution Distillation - Beta Mixture 55
10.1.4 Ensemble Distribution Distillation - Dirichlet 56
10.2 12 Point Ethics Checklist 56

List of Figures

1 Label Hierarchy of 14 CheXpert Diseases 23
2 Illustration of Subset of Training Data Used for Conditional Training 24
3 Example of a tree with 4 diseases A, B, C & D 24
4 Total Uncertainty derived from predictive entropy 35
5 Effect of Temperature Annealing on Beta Distribution 39
6 ROC: Atelectasis 53
7 ROC: Cardiomegaly 53
8 ROC: Consolidation 53
9 ROC: Edema 53
10 ROC: Pleural Effusion 53
11 ROC: Atelectasis 54
12 ROC: Cardiomegaly 54
13 ROC: Consolidation 54
14 ROC: Edema 54
15 ROC: Pleural Effusion 54
16 ROC: Atelectasis 55
17 ROC: Cardiomegaly 55

18 ROC: Consolidation 55
19 ROC: Edema 55
20 ROC: Pleural Effusion 55
21 ROC: Cardiomegaly 56

List of Tables

1 Summary Statistics of 5 Competition Labels 31
2 Uncertain Label Policy for Each Class 32
3 Mean Accuracy & AUC of Deep Ensembles 45
4 AUC and Accuracy of Ensemble and Distlled Models on Cardiomegaly Class 46
5 AUC and Accuracy of Ensembles and Distribution Distilled Model 46
6 Negative Log Likelihood of Test Set 47
7 NLL on Cardiomegaly class 47
8 Expected Calibration Error of Test Set 48
9 ECE on Cardiomegaly class 48
10 Patch Accuracy vs. Patch Uncertainty & Conditional Probabilities of Test Set 48
11 Patch Accuracy vs. Patch Uncertainty on Cardiomegaly Class . 49
12 Mean Uncertainty Measures for Test Set 49
13 Mean Uncertainty Measures for Gaussian Noise OOD Inputs . . 50
14 Mean Uncertainty Measures for MNIST OOD Inputs 50

3 Introduction

Deep learning methods have become incredibly popular recently in machine learning for a number of fields such as computer vision and natural language processing. They have vastly improved accuracy on a lot of machine learning benchmark datasets. Probably the most famous example of this is AlexNet, proposed in Krizhevsky, Sutskever and Hinton (2012), whereby enormous improvements in accuracy were made on the LSVRC-2010 ImageNet dataset by using deep learning approaches. The significance of this work was that it was the first deep learning model to achieve substantial improvement over traditional computer vision methods based on hand-engineered features.

Despite achieving impressive results in a number of supervised learning tasks, standard neural networks are poor when it comes to uncertainty quantification and tend to produce overconfident predictions. Overconfident incorrect predictions can be harmful or offensive (Amodei et al., 2016), an example of this could be an overconfident prediction of a diagnosis of a serious medical condition which could lead to the condition being undetected or unnecessary treatment being given. Therefore proper uncertainty quantification is crucial to enable practical applications of artificial intelligence.

A model that is able to resolve the type of uncertainty it is predicting would also be extremely useful in practical applications. For example it is useful for a model to be able to quantify whether a prediction is uncertain due to some inherent class overlap in the data or whether the uncertainty is due to model making predictions on data in areas where training data is either sparse or non-existent. An example of how accurate uncertainty quantification would be useful is if a machine learning model is attempting to diagnose a medical condition based on a medical test and the model gives a result but indicates it is highly uncertain in its prediction. If the model indicates that the high level of uncertainty is due to a weakness in the model then the test can be passed to a medical professional to make the prediction. If the model indicates that the uncertainty is due to some inherent class overlap or due to noise in the data then a medical professional would likely come to the same conclusion as the model, therefore more tests can be undertaken to collect more information to reach the correct diagnosis. If the model is confident in its prediction, a medical diagnosis can be made with no further testing. Using a machine learning model that can guide the decision making process of medical staff has a lot of potential to improve the accuracy of diagnosis's and reduce the overall cost to the hospital.

As already mentioned in some examples models that provide predictive distributions have a lot of potential to be applied in the medical field due to the serious consequences of incorrect overconfident predictions. This project aims to build a practical and scaleable deep learning model for medical imaging that provides accurate uncertainty measurements. There are many Bayesian methods that do not use deep learning that naturally lend themselves to accurate

uncertainty quantification. However these methods scale poorly with the high dimensionality of the data in image based tasks. Therefore this project will only focus on deep learning approaches to uncertainty quantification.

The CheXpert dataset contains over 224,316 chest x-rays from 65,240 patients(Irvin et al., 2019). Each X-ray has 14 labels assigned to it corresponding to 14 common thorax diseases, each of the 14 labels either corresponds to a positive, negative or uncertain diagnosis.

The best performing model on this dataset was developed by Pham et al. (2019) whose work shows impressive results outperforming radiographers on average on all image labels. This model used an ensemble of convolutional neural networks all with over 100 layers. An ensemble of deep learning models of this size requires significant computational resource at test time. Therefore this model may not be practical for hospitals with limited computational resource. It is important that the model produced by this project can also be deployed with limited computation.

A further complication exists whereby different hospitals and doctors follow slightly different procedures to collect chest X-rays. Patients from different countries and ethnicity's may also be under represented within the dataset a model is trained on. Both of these factors can lead to differences in the data the model could be used on in practice and the dataset it was trained on. Any practically applicable model must be either be robust enough to maintain its performance in response to the dataset shift explained here or show a reduced confidence in its predictions.

Transfer learning could also provide an approach whereby machine learning models could be practically deployed where the test data differs from the training data. A model can be pre-trained on a large readily available dataset such as the CheXpert dataset then can be fine tuned on a different dataset relevant to the hospital. This fine-tuning can be achieved with less training data and a smaller amount of training time. However the effect of transfer learning on uncertainty estimation has not been thoroughly tested. Therefore in this study the effect on transfer learning on uncertainty estimation will be studied empirically.

In summary this project will explore how to produce the best uncertainty estimates from previous research then will consider the challenges of deploying such a model with respect to the training and testing requirements. Methods that provide potential solutions to these challenges will be implemented on the CheXpert dataset as it is one of the largest readily available datasets for an extremely common medical test.

4 Literature Review

This section will provide a short review of the research on some of the topics relevant to the field of uncertainty estimation as well the most significant research on classification of chest X-rays. This section will include how uncertainty can be understood and deconstructed, the factors that can lead to uncertain predictions, an overview of some of the most popular deep learning methods that attempt to provide accurate uncertainty measurements and how these models can be compressed.

4.1 Types of Uncertainty

Machine learning models can give a measure of uncertainty. There are two sources of uncertainty; data uncertainty and knowledge uncertainty. Some models can provide predictions for one or both of these sources of uncertainty. Data and knowledge uncertainty can be added to approximate the total uncertainty of a prediction(Gal, 2016)(Kendall and Gal, 2017).

Data uncertainty also referred to as aleatoric uncertainty can be thought of as some inherent property of the data that results in data-points with the same input values giving different output values. In the case of classification this leads to class overlap in the data. Malinin, Mlodozeniec and Gales (2020) state that data uncertainty an irreducible component of the uncertainty in predictions that arises from the complexity, multi-modality and noise in the data. If a machine learning model can estimate the data uncertainty it can be thought of as a "known-unknown" i.e. the model has identified some noise in the data that it cannot resolve so it cannot give a confident prediction. The model is giving uncertain predictions due to the fact that some inherent class overlap in the data is present rather than any weakness in the model and no alterations to the model will be able to reduce this uncertainty.

Knowledge uncertainty also referred to as epistemic uncertainty (Gal, 2016) is due to weaknesses in the model which could be due to a number of reasons. One of which is that at test time the model is trying to make a prediction based on input data that is either in the domain of the training data but in an area where training data is sparse or the test input is outside of the domain of the training data meaning training data is non-existent close to the test input. Another reason for high measures of knowledge uncertainty could be that the underlying distribution that generated the training data is different to the one that has generated the testing data. This is known at dataset shift (Quionero-Candela et al., 2009) and is a situation that often arises in real world problems, for example the underlying distribution that generates the data could change over time meaning the distribution of the test data gradually moves further away from that of the training data. Knowledge uncertainty can be thought of as an "unknown-unknown", the model does not know how the output is distributed as it does not have sufficient information about its input.

Being able to separate the two components of your models uncertainty prediction can be incredibly useful in a practical sense. For instance if you see your predictions all have low knowledge uncertainty and the uncertainty predictions are composed primarily of data uncertainty, then you know modifications to your model through training or structure to attempt to reduce the uncertainty will be futile as data uncertainty is irreducible. However if you see that the uncertainty predictions are composed of a high level of knowledge uncertainty, then these uncertainty predictions can be reduced. For example you could use active learning techniques to collect data in areas of high knowledge uncertainty rather than high data uncertainty to improve the performance of your model(Malinin, Mlodozeniec and Gales, 2020).

4.2 Uncertainty Estimation in Deep Learning

This section will review some of the main methods of uncertainty estimation in deep learning. It will include some of the advantages and disadvantages of each method in terms of accuracy of predictions, quality of uncertainty predictions and computational cost. It is not an exhaustive list in terms of methods as well as the benefits and drawbacks of each method.

4.2.1 Conventional Neural Networks

Neural networks have produced incredible accuracy measurements on a handful of benchmark data sets but do not give predictive distributions in their standard form. However they can model aleatoric uncertainty by placing a distribution over the output of the model(Gal, 2016), this only requires minor modifications to the loss function and the outputs of the model. For example in regression this could simply be modifying a neural network to output the mean and standard deviation of a Gaussian distribution (Nix and Weigend, 1994). This can be done by modifying the loss function so that the so that the observation noise is not assumed to be constant, Kendall and Gal (2017) demonstrates how to model the aleatoric uncertainty in a heteroscedastic way. A draw back of this method is the assumption that the output is distributed by how the outputs of the neural network have been defined which is an assumption that does not always hold. Nevertheless it is a simple and easily implementable way of modelling the aleatoric uncertainty for regression problems. This can also be simply done for classification by using the softmax activation function on the final layer to give the output as probabilities for each class and by using an appropriate loss function such as negative log likelihood. This will automatically model the aleatoric uncertainty through the probabilities it assigns to each class.

Previous work shows that aleatoric uncertainty can be relatively easily modelled. However for classification tasks Guo et al. (2017) show significant empirical evidence that modern deep neural networks are prone to overconfident predictions and are generally poorly calibrated meaning these aleatoric uncertainty predic-

tions are often inaccurate. This research found that modern architectures of neural networks are prone to overconfidence , meaning the output probability for the prediction is likely to be higher than the empirical long-running frequency indicates.

A neural network in its standard form is composed of deterministic weights, biases and activation functions. In this form they have no way to model epistemic uncertainty meaning they are also prone overconfident predictions on out of domain examples or areas of sparse training data. This lack of epistemic uncertainty modelling means that neural networks in this form are incapable of identifying out of domain inputs (Amodei et al., 2016). This results in models that produce overconfident predictions even when given nonsensical inputs(Nguyen, Yosinski and Clune, 2014) or when faced with adversarial attack(Szegedy et al., 2014).

4.2.2 Bayesian Neural Networks

Bayesian neural networks can capture the epistemic uncertainty by modelling a neural networks weights as distributions learned from training data D, instead of point estimates and therefore it is possible to predict the output distribution y of some new input x through the predictive posterior distribution, shown in the equation below.

$$p(y|x, D) = \int p(y|x, \theta)p(\theta|D)\partial\theta \tag{1}$$

This takes the epistemic uncertainty into account as a prediction is the "weighted sum" of each outcome for each possible θ configuration of the model, with more probable θ configurations having a higher weight. The probability of a θ configuration depends on training data D.

For non trivial cases the term $p(\theta|D)$ is intractable. In Bayesian neural networks there are two main streams to approximate this term. There are the stochastic approaches and deterministic approaches. Stochastic approaches approximate the model parameters using sampling algorithms such as Markov Chain Monte Carlo. Deterministic approaches use techniques such as Laplace's method or Variational Inference to approximate the posterior over the weights $p(\theta|D)$. Both of these techniques formulate a proposal distribution $q(\theta)$ to approximate the posterior over the weights i.e. $q(\theta) \approx p(\theta|D)$.

In stochastic methods samples of θ approximately distributed to the posterior are obtained by simulating a Markov chain with $p(\theta|D)$ as its stationary distribution. Neal (1996) pioneered the sampling approach for Bayesian neural networks using Hamiltonian Monte Carlo methods to approximate the posterior over the weights. To date, it is considered a "gold standard" method for approximate Bayesian inference due to the impressive results achieved, but does not scale to large Deep Neural Networks or large-scale datasets(Gustafsson,

Danelljan and Schön, 2019)(Gal, 2016). It is worth mentioning that stochastic/sampling approaches have the advantage that they do not make assumptions about the proposal distribution meaning they are not constrained to a choice of proposal distribution $q(\theta)$(Gal, 2016).

Deterministic approaches generally use variational inference in which a proposal distribution $q_\phi(\theta)$ is explicitly chosen characterised by variational parameters ϕ. The best possible approximation of the posterior $p(\theta|D)$ is found by minimising the Kullback-Leibler (a measure of the similarity of two distributions, the lower the more similar the two distributions) between the proposal distribution and the posterior(Gustafsson, Danelljan and Schön, 2019). While principled, variational methods generally require sophisticated implementations, especially for more expressive variational distributions $q_\phi(\theta)$, although they scale better with larger datasets than sampling approaches, the choice of proposal distribution for the weights can be unsuitable and limit the accuracy of the model(Louizos and Welling, 2016)(Zhang et al., 2018).

Variational and sampling approaches to Bayesian deep learning have not proved to be the most popular in recent years due to poor scalability with respect to large datasets and also due to difficulty of implementation, especially for non experts. The following two sections will explore two methods that have proved for more popular in recent years due to superior accuracy, scalability and ease of implementation(Gustafsson, Danelljan and Schön, 2019).

4.2.3 Monte Carlo Dropout

MC-Dropout(Kendall and Gal, 2017)(Gal and Ghahramani, 2016), is a simple and scalable approach to deep learning with uncertainty estimation. The algorithm consists in training a network with dropout probabilities before every layer and then, at testing time, keeping dropout to sample M outputs of y where each sample of y has been generated with different weights set to zero. Each different set of weights used to generate the samples of y corresponds to a sample from the approximate distribution of the model parameters $p(\theta|D)$. The model prediction y is the mean of the different outputs. One major benefit of this approach is that both epistemic and aleatoric uncertainty are easily modelled, this will be further discussed in the next section about ensembling methods since MC dropout can be interpreted as a form of ensembling(Lakshminarayanan, Pritzel and Blundell, 2017)(Srivastava et al., 2014).

This algorithm approximates the posterior with a product of Bernoulli distributions. Given a dropout probability p it corresponds that each weight of the network θ_i has a probability of p of being set to zero. The approximating distribution of each θ_i can be seen as a mixture of 2 Gaussians with small variances and the mean of one of the Gaussians is fixed at zero. Therefore MC-dropout can be interpreted as using Variational Inference to approximate the posterior distribution of the model parameters.(Gal, 2016)(Kendall and Gal, 2017)

This approach introduces the dropout rate p as a hyper-parameter which is important with respect to the models accuracy and uncertainty estimation. One problem is that finding the optimal value for p can be time consuming especially if p is layer dependent. Therefore methods of automatically tuning this parameter are very useful. Concrete Dropout(Gal, Hron and Kendall, 2017) follows a variational interpretation of MC-dropout to provide a gradient based solution to tune p. This method shows comparable performance to grid-search of p(Gal, Hron and Kendall, 2017) and an improvement in model calibration compared to vanilla MC-dropout(Mukhoti and Gal, 2018a).

4.2.4 Deep Ensembles

Ensembling of deep neural networks was originally introduced by Lakshminarayanan, Pritzel and Blundell (2017) as a practical method to estimate uncertainty. Ensembling has already been proven to improve performance in terms of accuracy for many deep learning problems (Goodfellow, Bengio and Courville, 2016). Lakshminarayanan, Pritzel and Blundell (2017) were able to show that deep ensembles were also able to outperform approximate Bayesian approaches to deep learning in terms of uncertainty quantification. Due to their state of the art results in terms of accuracy and uncertainty predictions they are an incredibly powerful tool.

The original method proposed by Lakshminarayanan, Pritzel and Blundell (2017) simply trains the same network multiple times but with random initialisation of weights and using the negative log likelihood cost function. As mentioned earlier MC-dropout can be interpreted as a form of ensembling with shared weights. There are a number of ways to model aleatoric and epistemic uncertainty in ensembles, the general principle is if the model is uncertain then the output of the ensembles should be diverse, akin to high epistemic uncertainty, while if the members of the ensemble should all be similar if the uncertainty is due to data uncertainty. In regression the output of the ensemble y can be estimated in terms of the output of each model y_i and the number of ensembles N. The aleatoric and epistemic uncertainties can be easily modelled as well if each model outputs an estimate of the aleatoric uncertainty $\sigma_{a,i}^2$. The equations to for the output y and the aleatoric and epistemic uncertainties of the ensemble or MC dropout model are as follows.

$$y = \frac{\sum y_i}{N} \qquad \sigma_a^2 = \frac{\sum \sigma_{a,i}^2}{N} \qquad \sigma_e^2 = var(y_i) \tag{2}$$

For a classification problem where each model outputs probabilities the output prediction y is the same for regression. Depeweg et al. (2017) proposed a method for modelling total uncertainty that could be resolved into data and knowledge uncertainty. The total uncertainty, σ_T could be estimated by the entropy of the predictive posterior i.e. the output of the ensemble y, this is

intuitive as maximum entropy for a single label classifier corresponds to an output probability of 0.5 and minimum of zero entropy at 0 or 1 probability. The aleatoric uncertainty, can be estimated by the average entropy of all of the models in the ensemble. The epistemic uncertainty can then be estimated by simply computing the total uncertainty minus the aleatoric uncertainty.

$$\sigma_T^2 = entropy(y) \qquad \sigma_a^2 = \frac{\sum entropy(y_i)}{N} \qquad \sigma_e^2 = \sigma_T^2 - \sigma_a^2 \tag{3}$$

The general principle is that whenever the the ensemble is uncertain the total uncertainty should be high regardless of reason. Whenever the aleatoric uncertainty is then each each member of the ensemble should be consistent in giving uncertain predictions as they have seen this data before. However when the epistemic uncertainty is high the members of the ensemble should give a diverse set of predictions that average to give an output with high uncertainty.

It is essential to have this diversity in the ensemble members to properly quantify the epistemic uncertainty. As previously stated ensemble methods are known to improve predictive performance in deep learning. Deep ensembles are able to achieve a reduced overall error because of the diversity in the errors of each model, perfectly correlated errors do not bring about any performance gains to the ensemble. However uncorrelated errors reduce the expected ensemble error proportionally to the number of employed instances(Goodfellow, Bengio and Courville, 2016). Different solutions can be reached by each member of the ensemble and therefore a diverse ensemble is achieved due to the large number of local minima present in the loss function for most deep learning problems and the sub-optimal optimisation strategies involved needed to make optimisation computationally feasible. Model uncertainty can be explained by ensemble variance. Firstly the members of the ensemble will tend to output similar values when the inputs are similar to the training data, as each members weights although they are most likely different will have been optimised for the training data. Although as the input data becomes more dissimilar to the training data, the difference in model parameters between models due to different sub-optimal solutions reached starts to have a greater effect on the output of each model. Therefore a higher variance in outputs is given as the test data differs from the training data. Without the diversity in the ensemble members this reduction in error and uncertainty quantification due to epistemic uncertainty would not be achieved, hence it is essential to ensure diversity in the ensemble. Diversity also leads to an ensemble that is more robust to adversarial attack, essentially the more similar models are in the ensemble the more likely all of them will be fooled by adversarial attack and the more diverse the models are the more likely that they will give a diverse set of predictions to give a highly uncertain result(Szegedy et al., 2014).

Lakshminarayanan, Pritzel and Blundell (2017) originally proposed deep ensembles as a non-Bayesian solution to uncertainty estimation. However with minor modifications to the original methodology proposed by Lakshminarayanan, Pritzel and Blundell (2017) it is possible to interpret it as a Bayesian inference technique(Pearce et al., 2018)(Duvenaud, Maclaurin and Adams, 2016). Even without the modifications, ensembling can be interpreted as a Bayesian approximation with an implicit distribution of the model parameters(Gustafsson, Danelljan and Schön, 2019).

There are a number of techniques to increase ensemble diversity if sufficient diversity is not provided by random initialisation of weights. These include regularisation techniques such as weight decay and early stopping. Weight decay is commonly known as L_2 regularisation simply adds a term to the cost function that penalises weights with a larger magnitude, essentially attempting to simplify the model by bringing more of the weights closer to zero and reduce over-fitting. However to increase diversity in deep ensembles this technique is slightly altered. Anchored Ensembling (Pearce et al., 2018) instead of penalising the magnitude of the weights with respect to zero penalises the weights with respect to the value each weight was initialised with. The weights are initialised by draws from a Gaussian distribution with mean zero and covariance matrix Σ. Early stopping simply limits the number of gradient descent steps that can be taken during training. Both of these techniques simply restrict the final values of the weights can differ from the initial values, this increases the diversity of the ensemble as each model is initialised with different weights. Both methods have drawbacks that they introduce extra hyper-parameters which need tuning to give optimal ensemble performance. Anchored Ensembling requires a weight decay term to be tuned as well as the covariance matrix Σ which can differ for each layer, early stopping requires the maximum number of gradient steps descent steps to be tuned as a hyper-parameter.

Bootstrapping is a well known technique to increase the diversity of ensembles such as Random Forests. Instead if each member of the ensemble being trained on the dataset K random draws with replacement from the original dataset are made for each of the ensemble members to be trained on. If the original dataset is a good approximator for the underlying distribution that generates the data then each bootstrap draw that each model is trained on will also be a good approximator. This technique increases the diversity of ensemble by increasing the diversity in the training sets for each model. Bootstrapping has been shown to increase diversity in shallow ensembles, however its use in within neural nets and especially chest X-ray classification is likely to be less beneficial due to the large amount of training data. The reason behind this is that as the training dataset grows in size the individual bootstrap draws will lose their diversity with respect to one another(Lakshminarayanan, Pritzel and Blundell, 2017). This technique may also be unnecessary as recent literature on neural networks suggests that their loss functions are characterised by an extremely large amount of equivalent local minima(Goodfellow, Bengio and Courville, 2016). Therefore

stochastic gradient descent should provide some degree of diversity even when trained on the same training data due to different models reaching different local minima in the loss function.

4.2.5 Ensemble Distillation & Ensemble Distribution Distillation

Although ensembles have proven to be extremely powerful in terms of accuracy and uncertainty estimation, they require far more computational resource to train and make predictions. Although this is easily parallelisable in memory limited applications it is not always practical to have an ensemble with a large number of models. For example in an object detection system used for driverless cars the amount of memory available on board is limited. Therefore using a deep ensemble for prediction may not produce results fast enough to be of any use. In the field of chest X-ray classification extremely large networks are used often with 100+ layers (Rajpurkar et al., 2017). Training and/or deploying an ensemble of networks of this magnitude may be unfeasible for hospitals with limited resource. Therefore knowledge distillation and model compression techniques could be crucial in the practical use of Chest X-ray classification models. This section will discuss two similar techniques of ensemble knowledge distillation.

Ensemble Distillation is a technique whereby a neural network is trained to attempt to match the output of an ensemble. Essentially it will try and predict the ensembles predictions by being trained on pairs of the input data and the corresponding ensemble output. Hinton, Vinyals and Dean (2015)proposed a method for ensemble distillation and noted that a distillation approach may be taken rather than training a single model directly from the training data because it is less likely to overfit to the training data. The objective of a machine learning model is to generalise well to new data at test time, although generally models are trained to optimise performance on the training data when the real objective is to generalise well to new data. However the information about how to generalise well is not always available or easily extracted from the training data when using a single model trained directly on the training data. Deep ensembles can achieve this generalisation through diversity in the members of the ensemble. Therefore the information about how to generalise well is present in the ensemble, by distilling this ensemble into a single model it is possible to transfer the knowledge about generalisation to the single model.Balan et al. (2015) proposed a simple way to distill ensembles into a single model while preserving an uncertainty measure for classification tasks. The training instances are simply the inputs x to the ensemble paired with the mean softmax probabilities for each output. The loss function is simply the Kullback-Leibler divergence between the output softmax probabilities and the mean of the ensemble.

Given that an ensemble generalises well to new data, then a small model distilled from the ensemble will typically perform better than a model of the same size trained directly on the training data(Hinton, Vinyals and Dean, 2015). Not only

does this bring about a computational performance gain in comparison to the ensemble but the distilled model is also robust to adversarial attack(Papernot et al., 2015). The work by Hinton, Vinyals and Dean (2015) also found that distilled networks can be trained on much less data than the ensemble, this is because the ensemble outputs which have a high entropy are much more informative than the original training data set with a single label. This is due to the fact that the high entropy prediction is due to a number of similar training points outputting different classes, so one high entropy prediction from the ensemble contains far more information than a standard labelled data point with the same input. Consequently the distilled model can be trained on much less data and with a much higher learning rate. Once the ensemble has been trained it is very cheap to create the training data for the distilled model as labelled data is not required.

Ensemble distillation is able to provide a scalable and practical alternative to uncertainty estimation with only minor losses in terms of accuracy compared to the ensemble. However the ability to resolve the uncertainty into its two forms is lost as the distilled model only predicts the mean of the ensemble so information about the diversity of the predictions is lost. As a result of this this method performs badly on uncertainty evaluation tasks such as out of domain detection (Malinin, Mlodozeniec and Gales, 2020).

Ensemble Distribution Distillation is a technique proposed by Malinin, Mlodozeniec and Gales (2020) based on similar concept to Ensemble Distillation. This technique aims to capture information about the diversity of the ensemble as well as the mean so that the uncertainty can of the distilled model can be resolved. This is done by distilling the ensemble into a Prior Network(Malinin and Gales, 2018). Malinin, Mlodozeniec and Gales (2020) state that Prior networks model a conditional distribution over categorical distributions by learning the parameters of a Dirichlet distribution and that by doing this a single model can emulate an ensemble.

The principle is essentially that the distilled model outputs the parameters of a probability distribution to approximate each of the M members of the ensembles predictions for a given input. By predicting the output of each of the models in the ensemble the diversity of the ensemble is captured in a single model. For classification a Dirichlet distribution is often chosen as it allows closed form solutions for uncertainty expressions to be obtained. Although a Dirichlet distribution may be too limited to fully capture the behaviour of an ensemble and other distributions may need to be considered. The distilled network is trained by minimising the negative log likelihood of each of the outputs of the models in the ensemble given the distribution parameters the distilled network has predicted. This technique allows the ensemble to be distilled into a single Prior Network.

In experiments conducted by Malinin, Mlodozeniec and Gales (2020) both Ensemble Distillation and Ensemble Distribution Distillation produced superior results in terms of accuracy compared to a single model and only slightly worse than the original ensemble. Although on uncertainty evaluation metrics such as misclassification detection and out of distribution detection, Ensemble Distribution Distillation performs far better than Ensemble Distillation. On the out of distribution tests Ensemble Distillation performs even worse than the single model, this is likely due to the distillation model throwing away information about the diversity of the ensemble. While Ensemble Distribution Distillation performs very well on this test as it is able to retain the information about the diversity of the ensemble.

However Ensemble Distribution Distillation has currently only been applied to single-label classification tasks where each exemplar can only belong to one label. The X-ray classification problem proposed by this project is a multi-label problem where multiple labels can return true values for one exemplar. Therefore Ensemble Distribution Distillation is not suitable in its current form and would need to be adapted for multi-label classification.

In summary Ensemble Distribution Distillation could be extremely useful for building a scalable and accurate model that performs well on uncertainty evaluation for classification of chest X-rays if the approach can be adapted to a multi-label setting.

4.3 Classification of Chest X-rays

Classification of chest X-rays is a computer vision task, this section will list the methods typically used in deep learning for computer vision as well as the state of the art models used to classify chest X-rays. Deep learning is suited to computer vision tasks as visual scenes are often hierarchically organised. This is represented well by the multiple layers that make up a deep neural network.

4.3.1 Convolutional Neural Networks

Due to the extremely high dimensionality of images fully connected neural networks would require an extremely high number of parameters even in shallow networks for computer vision tasks. This means fully connected networks would have high memory, computation and training time requirements. Also a high number of parameters often means that the model is prone to overfitting due to over complexity.

Convolutional neural networks have been extremely successful in computer vision tasks as they incorporate prior knowledge about the images into the structure of the network thereby massively reducing the number of parameters of the network. Image statistics are translation invariant i.e. a picture of a cat is still a picture of a cat invariant of where the cat appears in the image. This can be

built into the model by tying lots of weights together to reduce the number of parameters. Low level features e.g. edges are expected to be local. This is built into the model by only allowing local connectivity(patches) between layers. High level features e.g. eyes, noses are expected to be relatively coarser to the rest of the image. This is built into the model by subsampling more and more up the hierarchy, this in turn reduces the number of parameters further(Goodfellow, Bengio and Courville, 2016).

Convolutions and pooling are two of the methods of incorporating this prior knowledge into the architecture of the network. Convolutions build the translation invariance and locality of low level features into the network, while pooling builds the prior about coarseness of high level features into the model. Convolutions translate different filters over an image or set of nodes to get a single output for each translation over the image/nodes. Pooling simply takes the outputs from a rectangular neighborhood of nodes and puts them through some function to give a single output. This is the average out put of the neighborhood or maximum value. Each layer of a convolutional neural network is typically consists of three stages. The first being a convolutional stage where several convolutions are performed in parallel to produce a set of linear activations. Next these activations are fed through an activation function, often a ReLU. Lastly the output of the activations is fed through a pooling layer.

4.3.2 State of The Art Models

Recently a number of studies have produced models that are able to outperform radiologists on classification of X-rays. A model produced by Rajpurkar et al. (2017) was one of the earlier models to outperform radiologists using a 121 layer convolutional neural network. However 14 separate networks were trained for each label, indeed one of the main challenges this dataset brings is producing a single model that accurately makes predictions on all 14 labels. To date Pham et al. (2019) have produced the most accurate model on all 14 labels. This was achieved using an ensemble of 6 convolutional neural networks based of the following well known architectures; DenseNet-121, DenseNet-169, DenseNet-201 [18], Inception-ResNet-v2, Xception and NASNetLarge. Experts have identified a hierarchy in the labels of the dataset, this method uses conditional training to exploit the dependencies between labels to transfer this expert knowledge into the model and improve performance. This study also proposed a novel method to deal with the labels in the dataset that had been given an uncertain diagnosis. Label smoothing regularisation is applied to the uncertainty labels. This simply means assigning a random value between two limits to the uncertainty labels where the limits are hyper-parameters.

Although the model produced by (Pham et al., 2019) is an ensemble that outputs a probability for each label no uncertainty evaluation is conducted on the model to quantify whether the uncertainty predictions are informative. In terms of further work on this study out of domain detection is mentioned by the au-

thors as a potential avenue to improve the model's practicality. Chest X-rays are likely to differ based on factors such as gender, ethnicity, age and type of machine used to generate the X-ray. Consequently distributional shift between the training set and the test set is likely, producing criteria for detecting out of domain inputs is essential for practical applications.

4.3.3 Transfer Learning

Transfer learning can significantly reduce the training time required by initialising a neural network with the weights of a model that has been trained on a readily available dataset such as ImageNet. As models trained on the CheXpert dataset are publicly available hospitals with a smaller sample of training data which is likely to have undergone some distributional shift from the CheXpert dataset could benefit from using models pretrained on the CheXpert dataset. Using a pretrained model in this scenario would likely reduce then training time and prevent overfitting on a smaller sample of training data which is more appropriate for a a specific hospitals patients and procedure in collecting chest X-rays.

However as previously mentioned accurate uncertainty quantification is a key requirement. Deep Ensembles currently provide the most accurate uncertainty estimates but require random initialisation of weights to provide diversity across models in the ensemble, meaning pretraining is not possible. However as stochastic gradient descent is used as an optimiser for neural networks, models initialised with the same weights can differ after training. The effect of using just stochastic gradient descent to provide the diversity within an ensemble has not been studied extensively. If stochastic gradient descent can be shown to provide sufficient diversity within an ensemble then use of pretrained models would be compatible with deep ensembles.

4.3.4 Conditional Training

As mentioned a hierarchical structure exists within the labels of the CheXpert dataset. Pham et al. (2019) exploit this hierarchy and show that their conditional training method improves the accuracy of their model compared to a 'flat' classifier which treats every label independently. The hierarchy has been constructed by domain experts and is not the product of another machine learning study. Figure 1 shows the hierarchy proposed by Van Eeden et al. (2012) and Rajpurkar et al. (2017). A simple way of interpreting this is that a positive diagnosis for Atelectasis also means the patient must have a positive diagnosis for Lung Opacity, but a positive diagnosis of Lung Opacity. Some of the leaf nodes in this hierarchy can be particularly difficult to classify using a machine learning model as they contain very few positive samples, exploiting the hierarchy within the labels has been shown to improve performance on these labels Pham et al. (2019).

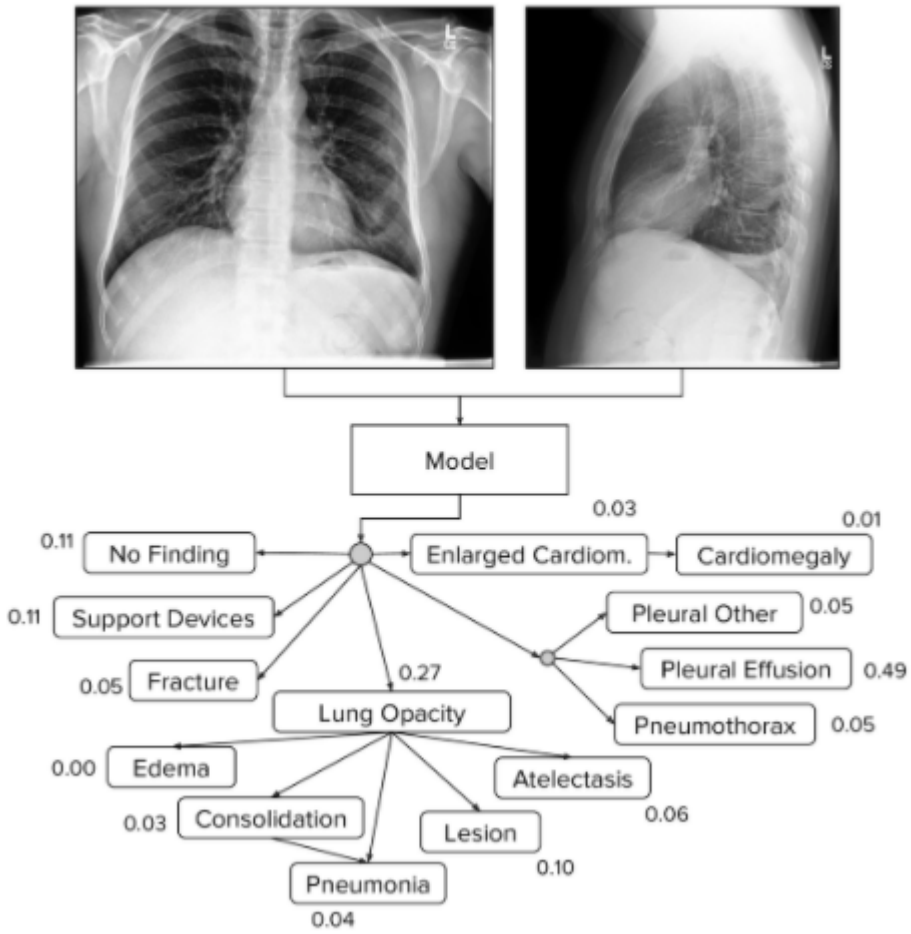

Figure 1: Label Hierarchy of 14 CheXpert Diseases

Presuming the medical validity of the hierarchy the training procedure is then broken into two steps. The first is called conditional training and which attempts to learn dependent relationships between parent and child labels e.g. the conditional relationship between Lung Opacity and Atelectasis. This step focuses on distinguishing the lower level leaf labels. This is done by taking a subset of the training data which contains only positive parent labels to classify the child labels, illustrated by Figure 2.

The model is then pretrained on this subset of training data. The second step exploits transfer learning by freezing all the weights of the pretrained model except the last layer and then retraining the model on the entire dataset. This step aims to improve the networks ability to predict the parent labels.

Figure 2: Illustration of Subset of Training Data Used for Conditional Training

Pham et al. (2019) state that for this training strategy the output of a label is the conditional probability that it is positive given its parent labels are positive. However at test time all labels should be unconditionally predicted, which can be done through a simple application of Bayes Rule. The unconditional probabilities of each label are computed by multiplying all the conditional probabilities produced by the model along the path from the prediction label to the root node.

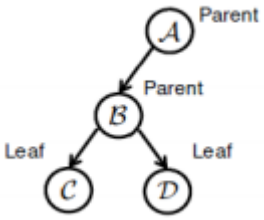

Figure 3: Example of a tree with 4 diseases A, B, C & D

Figure 3 shows an example of a tree with 4 diseases A,B,C & D. A model trained under the described procedure would output the probabilities $p(A), p(B|A), P(C|A, B)$ & $P(D|A, B)$. As a positive child node requires its parent nodes to also be positive the probability of a child node being positive can be expressed as the joint probability of the child node and all parent nodes being positive shown by equation 4

$$p(C) = p(C, A, B) \tag{4}$$

Using Bayes rule this can then be expressed in terms of the conditional probailities along the path to the root node outputed by the model, shown in equation 5.

$$p(C) = P(C|A, B)P(B|A)P(A) \tag{5}$$

This ensures that the probability of a child label being positive is always less that than of its parent label being positive as would be the case in a clinical setting.

4.4 Uncertainty Evaluation & Performance Metrics

This section will discuss some of the methods proposed in literature used to evaluate the accuracy of uncertainty estimation.

4.4.1 ROC Curves

Receiver-operator characteristic(ROC) curves are commonly used to evaluate the performance of classifiers. An ROC curve is produced by plotting the true positive rate(TPR) against the false positive rate(FPR) at varying classification thresholds. Increasing the classification threshold means more fewer exemplars are classified as positive lowering the number of both false positives and true positives. The TPR and FPR and given by the following, where TP, FN, FP and TN are the number of results that are true positives, false negatives, false negatives and true negatives respectively.

$$TPR = \frac{TP}{TP + FN} \qquad FPR = \frac{FP}{FP + TN} \tag{6}$$

The metric used to evaluate this curve is simply the area under the ROC curve (AUC) this is simply the integral of the ROC curve and gives a value between 0 and 1, with 1 being a perfect score. This metric provides a measure of performance across all classification thresholds. Given a random positive example and a random negative example the AUC can be interpret ted as the probability that the model predicts a higher probability of a positive label for this randomly chosen positive example compared to its prediction of the randomly chosen negative label. AUC is commonly used as it is scale invariant and classification threshold invariant. So provides a meaningful way of comparing accuracy across models. However it is not very informative for uncertainty estimation.

4.4.2 Ranking Based Methods

Ranking based methods produce quantitative indices by first ranking predictions based on their respective uncertainty predictions. Uncertainty can be evaluated by computing how the error varies as we remove the predictions with the highest levels of uncertainty. An accurate uncertainty estimator should correspond to improved accuracy as the most uncertain predictions are removed. Whether or not the uncertainty model obeys this can be captured by a confidence curve. A confidence curve is a plot of the error on the y-axis vs. the confidence percentile on the x-axis i.e. the error on the subset of $n\%$ of the test data ordered by its uncertainty estimates. The leftmost point on this curve is simply the error rate on the entire test dataset. Ideally this curve should never increase as we remove the most uncertain data points. The slope of the confidence curve is also

informative about the quality of the uncertainty estimation. If the gradient i negative the larger the magnitude the better as it corresponds to the error rate reducing more as uncertain predictions are removed. In classification perfect performance on a confidence curve would correspond to all correct predictions having lower uncertainty predictions than incorrect predictions. Perfect performance on the confidence curve can be computed by the values that would be given by the true error, this is called oracle ordering(Ilg et al., 2018). This kind of evaluation assesses the ordering of predictions by there confidence.

From the confidence curve measures of uncertainty evaluation can be obtained so different models can be compared. Area under the confidence curve is not a suitable evaluation measure as a more accurate model will have an advantage even if it quantifies uncertainty poorly.

However performance compared to the confidence curve generated by oracle ordering can produce metrics which are comparable across different models. One of these metrics is called Area Under the Confidence-Oracle error, $AUCO$. This is simply the difference sum of the difference in error between between each quantile of the confidence curve. For q quantiles where the model gives the error rates $h^i = (h^i_1, h^i_2, ..., h^i_{q-1})$ and oracle ordering gives $h^o = (h^o_1, h^o_2, ..., h^o_{q-1})$, the $AUCO$ is defined as the following.

$$AUCO = \sum_{n=1}^{q-1} h^i_n - h^o_n \tag{7}$$

A value of zero for the $AUCO$ would indicate perfect performance in terms of ranking based on uncertainty.

$AUCO$ does not take into account the monotonicity of the confidence curve. The Decrease Ratio,DR gives a quantitive evaluation of this property.

$$DR(h^{(i)}) = \frac{\|\{h^i_j | h^i_j \geq h^i_{j+1}\}\|}{q-1} \quad \forall j \, \epsilon \, 1, .., q-1 \tag{8}$$

Where, $DR = 1$ corresponds to a perfectly non-increasing confidence curve. This metric captures the noise in the confidence curve rather than the uncertainty quality directly so should be used in combination with other metrics.

4.4.3 Calibration

Ranking based methods are limited in the fact that they do not take into account the magnitude of the uncertainty estimates, only the ordering of them. The calibration of a model refers to how well the model's predicted probability distributions match the observed empirical frequencies. Calibration has been an active area of research in deep learning as while deep learning models have been shown to be highly accurate, they have also been found to be poorly calibrated(Guo et al., 2017). Calibration is independent of model accuracy and is

important for model interpretability. A well calibrated model will provide uncertainty estimates that are informative to the user as they will be informative on their own rather than just in respect to the other predictions from the model.

Expected Calibration Error (ECE) is a common metric to assess a models calibration and therefore accuracy of uncertainty estimation. ECE measures the correspondence between predicted probabilities and empirical accuracy (Naeini, Cooper and Hauskrecht, 2015). For K classes the ECE is defined as:

$$ECE = \sum_{i=1}^{K} p(i) \cdot \|acc(b_i) - conf(b_i)\| \tag{9}$$

Where $b(i)$ is the ith bin in this case that is simply each class, $p(i)$ is the fraction of predictions that fall into that bin, $acc(b_i)$ and $conf(b_i)$ correspond to the accuracy and average confidence for each bin b_i.

Maximum Calibration Error (MCE) is another useful calibration metric that is suitable for high risk applications like chest X-ray classification as it models the worst case scenario(Guo et al., 2017).

$$MCE = max_{i=1}^{K}(p(i) \cdot \|acc(b_i) - conf(b_i)\|) \tag{10}$$

4.4.4 Dispersion

Calibration metrics can show good results even if the uncertainty estimates are nonsensical. For example if the uncertainty estimates are constant but match the empirical accuracy over the entire distribution, then the model is perfectly calibrated. However this is of no use as the uncertainty is not input dependent. Dispersion can be used as a diagnostic tool that should be used alongside calibration metrics. Dispersion is a measure of the input dependency of the uncertainty estimates. The most common metric is the coefficient of variation (c_v) which is simply the standard deviation of the uncertainty measurements normalised by the mean uncertainty. A c_v of zero corresponds to the example mentioned above where a model could be perfectly calibrated but uncertainty measurements are input independent. A higher c_v means the the models uncertainty estimates vary more with the input.

It is important to note that dispersion cannot be used on its own to evaluate uncertainty estimation. A higher c_v by itself does not necessarily correspond to more accurate confident estimates simply more input dependent. Also different datasets have different "true" dispersion values, a given value of c_v could be naturally low for a homogeneous dataset. The c_v also does not take into account the absolute value of the uncertainty estimates, simply their distribution about the mean. Although it is still a useful tool to be used alongside calibration metrics as it is possible to check if any improvement in calibration is due to model improvement or due to reduction in dispersion.

4.4.5 Proper Scoring Rules

Proper scoring rules like the brier score(Brier, 1950) and negative log likelihood are popular metrics for evaluating the quality of uncertainty estimates. Proper scoring rules are metrics whee an optimum score corresponds to a perfect prediction unlike calibration metrics where this is not necessarily the case. Both Brier score and negative log likelihood penalise high probability predictions assigned to incorrect labels and low probability predictions assigned to correct labels.

4.4.6 Patch Accuracy vs. Patch Uncertainty

Patch Accuracy vs. Patch Uncertainty (PAvPU) is an uncertainty metric proposed by Mukhoti and Gal (2018b) for image segmentation tasks that combines the probability that a model is accurate given that it is confident in its prediction and the probability that a model is uncertain given that its prediction is inaccurate. It can easily be applied to classification tasks. This metric is built on the assumption that a model with accurate uncertainty quantification should produce accurate results when it is certain of its predictions and it should produce uncertain predictions when those predictions are inaccurate. These two characteristics can be expressed through conditional probabilities $p(accurate|certain)$ and $p(uncertain|inaccurate)$. Ideally these conditional probabilities would evaluate to 1 if a model produces accurate uncertainty estimates. To evaluate these probabilities an uncertainty threshold needs to be defined based on the models predictive uncertainty e.g. predictive entropy. Whereby predictions that exceed this uncertainty threshold are categorised as uncertain and predictions that do not are categorised as certain. The threshold is a hyperparameter to tune for a given model. The conditional probabilities can then be evaluated. With the uncertainty threshold the number of predictions that are accurate and certain (n_{ac}), inaccurate and certain (n_{ic}), inaccurate and uncertain (n_{iu}) and accurate and uncertain (n_{au}) can be computed. From these values the conditional probabilities can be computed and are also useful metrics for a models performance.

$$p(accurate|certain) = \frac{n_{ac}}{n_{ac} + n_{ic}} \tag{11}$$

$$p(uncertain|inaccurate) = \frac{n_{iu}}{n_{iu} + n_{ic}} \tag{12}$$

The PAvPU metric combines good performance in both of these metrics i.e. high accuracy for certain predictions and high uncertainty for inaccurate predictions. It is computed by the following equation.

$$\text{PAvPU} = \frac{n_{ac} + n_{iu}}{n_{ac} + n_{au} + n_{ic} + n_{iu}} \tag{13}$$

Higher values of all of these metrics indicate better performance. It should be noted that they depend on both an accuracy threshold and an uncertainty threshold, which should be optimised for each model if using these metrics for comparison.

4.4.7 Domain Shift & Out of Domain Inputs

A models uncertainty measures need to be evaluated when the test data has been subject to distributional shift and when the test data is completely out of domain. This can be done by evaluating the metrics already mentioned under both of these scenarios. Distributional shift is applied by adding perturbations to the input data, one would expect the predictive uncertainty to increase as larger perturbations are added. Out of domain inputs are applied by simply using test data from a completely different test set

For a single model misclassification detection and out of domain input detection can be used as metrics by setting a threshold in the models uncertainty predictions. If the uncertainty exceeds this value the label prediction is rejected.

4.5 Methods of Reducing Uncertainty

Deep learning models often require extremely large amounts of labelled data, in medical applications obtaining the labelled data can often be expensive as the data needs to be labelled by highly trained professionals. In this scenario where large amounts of data is needed and expert knowledge is expensive, it is extremely useful to know what extra data would be the most informative as expert knowledge can be used in areas where it is most effective. The epistemic uncertainty can be used in this scenario as points with the highest epistemic uncertainty represent regions of sparse or non-existent training data and therefore the areas where extra data would be the most informative. Extra data in these areas would reduce model uncertainty and increase model accuracy far more than blindly collecting extra data. Active learning techniques could be implemented to complete the task of finding the most suitable points where expert knowledge would be most informative(Settles, 2010). In this approach the model would identify what unlabelled data would be the most informative. The choice of these data points can be determined through an acquisition function, many of these functions rely on the epistemic uncertainty to rank each unlabelled data points potential informativeness(Houlsby et al., 2011). Using this method could vastly reduce the amount of labelled data needed from an expert which could prove incredibly useful in practical applications.

5 Objectives

The aims of this project are first to create a Deep Ensemble trained on the CheXpert dataset with random initialisation of weights as this will provide a baseline for uncertainty estimation from which the performance of other models can be compared to.

Another Deep Ensemble will then be trained initialised with pretrained weights from a publicly available model trained on the CheXpert dataset. A number of uncertainty metrics will then be used to assess if the pretrained ensemble

provides comparable uncertainty estimation to the baseline. The quality of uncertainty estimates from ensembles trained from pretrained models has not been extensively studied by current research. If pretrained models are able to produce ensembles with accurate uncertainty estimation then this method could be extremely useful for hospitals attempting to implement machine learning models on a smaller training set more representative of the patients the model would be used on in practice.

The second main goal of this project is to distill the pretrained ensemble into a single model that preserves its uncertainty estimation. Ensemble Distribution Distillation is a technique that has been proposed in literature that has been proven to preserve a Deep Ensemble's accuracy and uncertainty estimates however this approach is not suitable for multi-label classification tasks where multiple positive labels for a given exemplar are possible. Therefore one of the key contributions of this project is the adaptation of Ensemble Distribution Distillation to multi-label tasks via a different loss function to the one proposed originally by Malinin, Mlodozeniec and Gales (2020).

6 Problem Setup

The CheXpert dataset contains 224,316 chest X-rays of 65,240 patients which have been converted into 1024 x 1024 images with 255 grey levels. Each image has 14 labels assigned to it which correspond to 14 common thorax diseases. Each label has been assigned one of a positive, negative, uncertain or unmentioned diagnosis based on medical reports provided by radiographers. The dataset contains a validation set of 200 images that a group of experts have independently labelled for each observation to produce a validation set with no uncertainty in the labels. The same procedure has been applied to the test set of 500 images which is not publicly available which is used to rank entrants to the CheXpert competition. Entrants are ranked based on their AUC scores on this held out test set across only 5 of the 14 labels. Therefore the models proposed in this project will only be evaluated across these 5 labels, the 5 thorax diseases on which the models are trained and tested on are Atelectasis, Cardiomegaly, Consolidation, Edema and Pleural Effusion. These 5 labels were selected by the competition regulators based on their clinical importance and prevalence within the dataset. Models were submitted for evaluation on this test set on 26th August 2020 but results were not received before the conclusion of this project. Therefore the validation set provided labelled by a panel of experts was used as the test set to evaluate model performance.

6.1 Data Preprocessing

The original dataset provided by the CheXpert competition contains 224,316 1024 x 1024 pixel images. However this brings the size of the dataset to 439Gb which was not feasible to train a model with given the computational resources

available. Therefore a heavily down-sampled version of the dataset was used also given by the CheXpert competition, these images have a reduced resolution of 320 x 320 pixels. The downsampling of the images most likely had a detrimental effect on the model accuracy but reasonable accuracy was still achieved in all models, therefore the results from experiments on the uncertainty estimation are still valid and would be able to scale to the dataset in its high resolution form.

The dataset contains many labels which have either been assigned as 'unmentioned' or 'uncertain', based on the attached radiographers reports of each X-ray. It is common and logical to assign each 'unmentioned' label as a negative label as a specific disease is generally only mentioned in the radiographers report if medical staff make a positive diagnosis or had suspected the patient may test positive based on the patients symptoms. The summary statistics after mapping all unmentioned labels on the 5 labels used is given by Table 1.

Pathology	Positive (%)	Uncertain (%)	Negative (%)
Atelectasis	29333 (15.53)	29377 (15.66)	128931 (68.71)
Cardiomegaly	23002 (12.26)	6597 (3.52)	158042 (84.23)
Consolidation	12730 (6.78)	23976 (12.78)	150935 (80.44)
Edema	48905 (26.06)	11571 (6.71)	127165 (67.77)
Pleural Effusion	75696 (40.34)	9419 (5.02)	102526 (54.64)

Table 1: Summary Statistics of 5 Competition Labels

Table 1 shows that the uncertain labels make up a significant proportion of each of the 5 classes used in this project. The handling of the uncertain labels represents of a key challenge of this dataset, different policy's have been developed in literature and are listed as follows.

- **U-Zeros:** Assign all uncertain labels to 0 (negative)
- **U-Ones:** Assign all uncertain labels to 1 (positive)
- **U-Ignore:** Ignore all uncertain labels during training
- **U-MultiClass:** Treat the uncertain labels as their own class
- **U-SelfTrained:** Train a model using the U-ignore policy. Then use the probability predictions of this model for each of the uncertain labels to relabel the uncertain labels.
- **Label Smoothing Regularisation(LSR):** For each uncertain label relabel with $y \sim \mathcal{U}(a, b)$. Where $0 < a, b < 1$, a and b are hyperparameters to tune(Pham et al., 2019).

Pham et al. (2019) currently have the best performing model in the competition. The LSR policy was one of the methods proposed in their work which improved

the accuracy of their model and was used for all classes. The idea behind the LSR policy is that it enables a model to make use of the large amount to uncertain labels to add to the training data but prevents the model from making an overconfident prediction on potentially mislabelled data. This is done by providing a soft target between 0 and 1 for the uncertain labels instead of a 'hard' 0 or 1. Pham et al. (2019) conducted which compared all the uncertainty policies mentioned above for each of the 5 competition classes with multiple hyperparameters for the LSR policy. This project will use the best performing policy for each class from these experiments. The uncertainty policy for each class is shown in Table 2.

Class	**Policy**
Atelectasis	LSR: a=0.55, b=0.85
Cardiomegaly	LSR: a=0.55, b=0.85
Consolidation	LSR: a=0.0 b=0.3
Edema	LSR: a=0.55, b=0.85
Pleural Effusion	U-Zeros

Table 2: Uncertain Label Policy for Each Class

However their model used the same policy for all of the classes while their empirical results show that different policies perform better for each of the 5 test classes. This project will use the uncertainty policy found to perform best on each class in the experiments conducted by Pham et al. (2019) and is shown in Table 2. The hyperparameters for the LSR policy were determined empirically in these experiments and the optimal values found by Pham et al. (2019) to be used on this project are likely related to the frequency each class returns a positive label.

The test set of 200 images used in this project is not included in the figures in Table 1, it has no uncertain labels as a panel of experts has decided on a positive or negative label for each class so models could be properly tested. During training the model that performs best on the validation set is saved for testing, therefore only images which originally have no uncertain labels across the 5 test classes. Of the 224,316 chest X-rays excluding the test set, 159,329 of the images contain no uncertain labels across the 5 competition classes. From this subset 1000 images were randomly chosen for the validation set.

The preprocessing and policy for uncertain labels give the training set D.

$$D = \{(\mathrm{x}_i, \mathrm{y}_i); i = 1, , , N\} \tag{14}$$

Each x_i is the image associated with the label y_i. The labels are given by the vector $\mathrm{y}_i = [y_1, , , y_K]; K = 5$, where the label y_{ik} for image i and class k $y_{ik} \epsilon [0, 1]$.

6.2 Deep Ensembles

To provide a baseline for uncertainty estimation one deep ensemble will be trained with random initialisation of weights. One of the aims of this project is to explore the effect of transfer learning on uncertainty estimation in deep ensembles. This will be done by training a deep ensemble where each models weights are initialised from a pretrained model which already possesses a high accuracy on the dataset, with the aim being to reduce the training time compared to the baseline model while also preserving the uncertainty estimation of the baseline. Diversity of the ensemble members is key to providing accurate uncertainty estimation. The baseline models diversity is provided by the random initialisation of weights and stochastic gradient descent. The key question for whether pretrained ensembles can provide accurate uncertainty estimation is if stochastic gradient descent on its own can provide sufficient diversity between members of the ensemble compared to the combination of random initialisation of weights and stochastic gradient descent.

In this project ensembles will be made up of M convolutional neural networks(CNN), since this is a multi-label classification problem the softmax activation function is not appropriate, instead a sigmoid activation function is used on the logits z_k at the last layer of each CNN to output each member of the ensembles predictions for the probability of each class π_k.

$$\pi_k = \frac{1}{1 + \exp(-z_k)}, \quad k = 1, , , 5 \tag{15}$$

Each CNN is trained using the cross entropy loss function to optimise the weights of each CNN θ, where $K = 5$ is the number of classes.

$$\ell(\theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} y_{ik} \log \pi_{ik} + (1 - y_{ik}) \log(1 - \pi_{ik}) \tag{16}$$

For image i and class k the prediction of the ensemble $\hat{y}_k$ is simply the average of all the members predictions for that class.

$$\hat{y}_{ik} = \frac{1}{M} \sum_{m=1}^{M} \pi_{imk} \tag{17}$$

6.2.1 Uncertainty Estimation from Deep Ensembles

From the ensemble predictions measures of uncertainty need to be derived. The simplest way to do this would be to use the maximum probability for each class shown in equation 18.

$$uncertainty = 2(1 - \max(p, 1 - p)) \tag{18}$$

However this uncertainty measurement from maximum probability can not be decomposed into data and model uncertainty.

Ideally an ensemble should output an uncertain prediction whenever the data or model uncertainty is high. When their is a large amount of data uncertainty in an input due to areas of class overlap in the training data all models in the ensemble should give similar uncertain predictions which average to an uncertain prediction. When the model uncertainty is high as the input is in an area of sparse or complete absence of training data then the members of the ensemble should give an diverse set of predictions which average to an uncertain prediction. Diversity of models is key to capturing model uncertainty as it relies on each member of the ensemble giving a different output for the same input. In summary when the model is data uncertain ensemble members should agree and when the model is model uncertain the ensemble members should disagree.

Given an ensemble that exhibits this behaviour Malinin, Mlodozeniec and Gales (2020) state that the total uncertainty can be expressed as the entropy of the predictive distribution i.e. the entropy of the prediction $\hat{y}$. Uncertainty in predictions due to knowledge uncertaintyKU can be expressed via measures of the spread of the ensemble such as the mutual information between the label $\mathbf{y}$ and the ensemble parameters given and input $\theta|\mathbf{x}$.

$$KU = MI(\mathbf{y}, \theta|\mathbf{x} \tag{19}$$

$$MI(\mathbf{y}, \theta|\mathbf{x}) = entropy(\hat{y}) - \mathbb{E}\left[entropy(\pi_m)\right] \tag{20}$$

The mutual information term is a measure of the knowledge uncertainty and the entropy of the predictions is a measure of the total uncertainty. Then by using the decomposition of total uncertainty into the sum of the knowledge uncertainty and the data uncertainty, it is clear that the final term in equation 20 corresponds to the data uncertainty. The data uncertainty is the expectation of the entropy of each of the ensemble members predictions.

The equations for the total, data and knowledge uncertainty are now given in terms of the notation of the problem setup for this project. The total uncertainty is expressed as the predictive entropy i.e. the entropy of ensemble predictions. The total uncertainty via predictive entropy TU for image i and class k is shown in equation 21 and 22.

$$entropy(\hat{y}_{ik}) = -\hat{y}_{ik}\log_2(\hat{y}_{ik}) - (1-\hat{y}_{ik})\log_2(1-\hat{y}_{ik}) \tag{21}$$

$$TU_{ik} = entropy(\hat{y}_{ik}) \tag{22}$$

Figure 4 shows the relationship between the ensemble probability predictions for one class and the total uncertainty. The data uncertainty can be expressed as the expected predictive entropy of each of the ensemble members. The data uncertainty DU from an ensemble of M models is given by equation 23.

$$\mathbb{E}\left[entropy(\pi_{imk})\right] = \frac{1}{M}\sum_{m=1}^{M} -\pi_{imk}\log_2(\pi_{imk}) - (1-\pi_{imk})\log_2(1-\pi_{imk}) \tag{23}$$

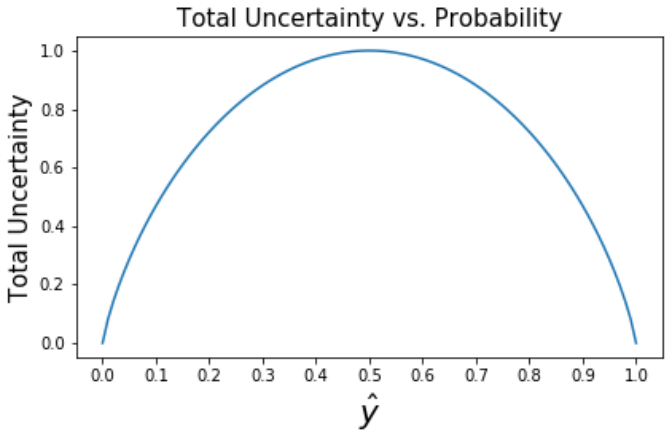

Figure 4: Total Uncertainty derived from predictive entropy

$$DU_{ik} = \mathbb{E}\left[entropy(\pi_{imk})\right] \tag{24}$$

The knowledge uncertainty KU is then simply given by the decomposition of total uncertainty into knowledge uncertainty and data uncertainty. Which is equal to the mutual information between the predictions

$$KU_{ik} = TU_{ik} - DU_{ik} \tag{25}$$

6.3 Ensemble Distribution Distillation

Practical use of ensembles can be computationally expensive due to the large number of models within the ensemble. For hospitals with limited computational resources use of a deep ensemble may not be feasible for classification of chest X-rays. Knowledge distillation provides a potential solution to this problem where by the ensemble can be distilled into a single model. Ensemble Distillation is a technique that preserves the accuracy of the Ensemble and distills an ensemble into a single model. However (Malinin, Mlodozeniec and Gales, 2020) showed that ensemble distillation provides lower quality uncertainty estimation than the ensemble as information about the diversity of the ensemble is lost. They also proposed Ensemble Distribution Distillation which preserves the accuracy and the uncertainty estimation of the ensemble by distilling the ensemble into a single model which outputs a distribution to model all of the ensemble member predictions. Ensemble Distribution Distillation (EnD^2) will be used on the pretrained ensemble to distill it into a single model.

The first step in EnD^2 is to create a transfer dataset D_T composed of the predictions of all members of the ensemble on the training and validation datasets to use as the training and validation sets for training the distilled model.

$$D_T = (\mathrm{x}_i, \pi_{imk}) \quad i = 1,,,N \quad m = 1,,,M \quad k = 1,,,K \tag{26}$$

EnD^2 is formulated by expressing the output of each ensemble member for a given input as a sample drawn from a distribution that we will aim to parameterise with the distilled model. By modelling ensemble member outputs as samples drawn from a probability distribution for which the distilled model predicts the parameters, the diversity of the ensemble is preserved as the output has to model all ensemble member predictions rather than just the mean as is the case in plain Ensemble Distillation (Hinton, Vinyals and Dean, 2015). The distilled model is trained via maximum likelihood estimation, e.g. attempt to predict the parameters which make all ensemble predictions most likely. To summarise for each input the model will predict the parameters of a distribution which maximise the likelihood of all ensemble predictions.

Malinin, Mlodozeniec and Gales (2020) proposed this approach and have implemented this by training a model to output the parameters of Dirichlet Distribution to make predictions and uncertainty estimates on a single label classification task with K classes. A Dirichlet Distribution is defined by concentration parameters $\boldsymbol{\alpha} = (\alpha_1, , , \alpha_K)$, where $\alpha_k > 0$. To ensure the distribution parameters are strictly positive an exponential function is applied to the output logits z_k. Ensemble member outputs $\boldsymbol{\pi_i}$ for each input i are then treated as samples from a Dirichlet distribution parameterised by the distilled models predictions.

$$\alpha_k = e^{z_k} \tag{27}$$

$$\boldsymbol{\pi_i} \sim \mathbf{Dir}(\boldsymbol{\alpha_i}) \tag{28}$$

The model is trained by minimising the negative log likelihood of the predictions of the ensemble members. Given model parameters ϕ, all ensemble member predictions $\boldsymbol{\pi}$ and model output $\boldsymbol{\alpha}$ the loss function is given by equation 32.

$$\boldsymbol{\alpha} = f(\phi, \mathrm{x}) \tag{29}$$

$$\alpha_0 = \sum_{k=1}^{K} \alpha_k \tag{30}$$

$$L(\phi) = -\mathbb{E}_{p(\mathrm{x})} \left[\mathbb{E}_{p(\boldsymbol{\pi}|\mathrm{x})} \left[\log p(\boldsymbol{\pi}|\mathrm{x}, \phi) \right] \right] \tag{31}$$

$$L(\phi) = \frac{-1}{N} \sum_{i=1}^{N} \left[\ln \Gamma(\alpha_{0(i)}) - \sum_{k=1}^{K} \ln \Gamma(\alpha_{ik}) + \right. \\ \left. \frac{1}{M} \sum_{m=1}^{M} \sum_{k=1}^{K} (\alpha_{ik} - 1) \ln \pi_{imk} \right] \tag{32}$$

Where $\Gamma(\alpha)$ is simply the gamma function shown in equation 33.

$$\Gamma(\alpha) = \int_0^\infty x^{\alpha - 1} \exp(-x) dx \tag{33}$$

The predictions of the distilled model are given by the mean of each variable parameterised by the output Dirichlet distribution. The prediction for class c is shown by equation 34.

$$\hat{y}_c = \frac{\alpha_c}{\sum_{k=1}^{K} \alpha_k} \tag{34}$$

6.3.1 Multi-label Classification

The task presented by this dataset is a multi-label classification task. The distillation procedure described in the previous section is not suitable for a multi-label task due to the distilled model only outputting one distribution making predictions of multiple positive labels impossible. One of the key contributions of this project is solving this problem by modifying the distilled model to output a mixture of Beta distributions so that Ensemble Distribution Distillation can be extended to a multi-label classification task.

In the multi-label case ensemble member outputs for each class have to be treated independently where each set of ensemble member predictions for 1 given class are modelled as draws from a distribution. This leads to the distilled model outputting the parameters for K distributions to predict for K classes. In this project a Beta distribution is used to model the M ensemble members predictions for each class $\boldsymbol{\pi}_k$ as samples from the Beta distribution.

$$\boldsymbol{\pi}_k = (\pi_{1,k},,,\pi_{M,k}) \tag{35}$$

$$\boldsymbol{\pi}_k \sim \mathbf{Beta}(\alpha_k, \beta_k) \tag{36}$$

The distilled model then predicts the parameters $\boldsymbol{\alpha}, \boldsymbol{\beta}$ for each of the K distributions that model all ensemble member predictions.

$$\boldsymbol{\alpha} = (\alpha_1,,,\alpha_K) \tag{37}$$

$$\boldsymbol{\beta} = (\beta_1,,,\beta_K) \tag{38}$$

$$(\boldsymbol{\alpha}, \boldsymbol{\beta}) = f(\phi, \mathrm{x}) \tag{39}$$

The distilled model is then trained in much the same way as the single-label approach by maximising the likelihood of all ensemble member predictions across the K output distributions. The negative log likelihood is used as the loss function for each output distribution then simply added over all outputs to give the loss function of the entire model. This is shown by equation 40, which was derived by considering equation 32 K times for 2 classes which are probability of positive label and probability of negative label which of course must sum to 1.

$$L(\phi) = \frac{-1}{NK} \sum_{i=1}^{N} \sum_{k=1}^{K} [\ln \Gamma(\alpha_{ik} + \beta_{ik}) - \ln \Gamma(\alpha_{ik}) - \ln \Gamma(\beta_{ik}) + \frac{1}{M} \sum_{m=1}^{M} [(\alpha_{ik} - 1) \ln \pi_{imk} + (\beta_{ik} - 1) \ln(1 - \pi_{imk})] \tag{40}$$

The predictions for a class y_c are simply given by the mean of the Beta distribution outputted to model the ensemble member predictions for that class.

$$\hat{y}_c = \frac{\alpha_c}{\alpha_c + \beta_c} \tag{41}$$

In literature Ensemble Distribution Distillation has only been applied with distillation into a Dirichlet distribution suitable for single label classification tasks. Therefore the formulation of Ensemble Distribution Distillation into a mixture of Beta distributions is a novel contribution which required the derivation of a new loss function.

The pretrained deep ensemble will be distribution distilled as described in this section outputting a mixture of Beta distributions. To validate the implementation of this contribution the predictions of just one of the classes will be distribution distilled using the method proposed by Malinin, Mlodozeniec and Gales (2020) which outputs a Dirichlet distribution.

6.3.2 Target Smoothing

Due to numerical precision of the implementation its possible an ensemble member prediction π_{imk} could be rounded to 0.0 or 1.0. In which case the loss function given by equation 40 cannot be computed due to the ln(0) term. This is avoided by applying a small amount of central smoothing to the ensemble member predictions shown by equation 42.

$$\pi_{ikm,smoothed} = (1 - \gamma)\pi_{ikm} + \gamma\frac{1}{2} \tag{42}$$

A value of γ used for all experiments is 1 x 10^{-4}.

6.3.3 Temperature Annealing

The distilled model is trained by minimising the negative log likelihood of the transfer dataset D_T. Malinin, Mlodozeniec and Gales (2020) state that this is equivalent to minimization of the Kullback-Leibler divergence between D_T and the distribution parameterised by the model $\mathbf{Dir}(\boldsymbol{\alpha})$. The training data is often made up of certain predictions with probabilities close to zero or 1, while the Dirichlet distribution predicted by the model often predicts uncertain predictions with probabilities nearer 0.5 upon initialisation. This means that upon initialisation of the model there is little non-zero common support between the ensemble predictions and the predicted distribution. Optimisation of the Kullback-Leibler divergence between samples and a proposal distribution is particularly difficult (Malinin, Mlodozeniec and Gales, 2020).

To make optimisation of the Kullback-Leibler divergence feasible Temperature Annealing is used whereby both the predictions of the ensemble and the predicted distribution parameters are 'heated up' which brings both the ensemble

member predictions and output distribution predictions closer to 0.5. This increases common support between the training data and the output distribution making the proposed optimisation easier.

The output distribution and ensemble member predictions are 'heated up' by dividing the logits which are used for the activation function at the output layer by the temperature variable T. A temperature value $T = 1$ is equivalent to no temperature annealing and $T > 1$ is equivalent to applying some amount of temperature annealing to the outputs. This is shown by equation 43 for the Ensemble Distribution Distillation model.

$$\alpha_c = \exp(z_c/T) \tag{43}$$

Figure 5 shows the probability density functions of a Beta distribution with α=5 & β=1 with no temperature annealing applied as well as the resulting Beta distribution when temperature annealing is applied with $T = 5$. The temperature annealing redistributes the probability density more uniformly which results in a more uncertain prediction.

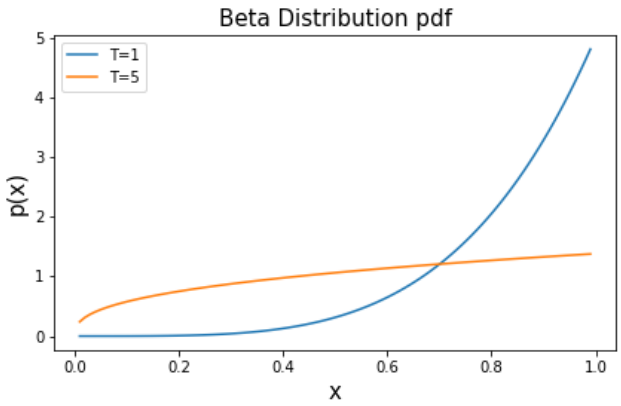

Figure 5: Effect of Temperature Annealing on Beta Distribution

The ensemble member outputs have already been computed, so the logits z have to be recovered by applying the inverse of the final activation function. The temperature can then be applied to the logits and the result is then applied to the activation function to recompute the ensemble predictions to get the 'heated up' predictions. This procedure is shown by equations 44-46 .

$$\pi = \frac{1}{1 + \exp(-z)} \tag{44}$$

$$z = \ln \frac{\pi}{1 - \pi} \tag{45}$$

$$\pi_{new} = \frac{1}{1 + \exp(z/T)} \tag{46}$$

Raising the temperature of the ensemble predictions and the output distribution allows for an easier optimisation problem, however this reduces the diversity of the ensemble by moving all predictions closer to 0.5. To allow for enough common support between the dataset and the predicted distribution to make optimisation feasible while and still model the ensemble diversity properly an 'annealing schedule' needs to be implemented. An 'annealing schedule' simply means gradually lowering the temperature as training progresses until it reaches 1 when no temperature is being applied. This allows enough common support at initialisation for optimisation of the Kullback-Leibler divergence between the training data and the output distribution. Then gradually reintroduces the diversity of the ensemble so the model can be trained to fully model the ensemble predictions.

6.3.4 Uncertainty Estimation from Ensemble Distribution Distillation

As Ensemble Distribution Distillation captures the diversity of the ensemble it possible to decompose the total uncertainty into measures of data and knowledge uncertainty. As with deep ensembles this is done by considering the knowledge uncertainty KU as the mutual information between the predictions $\hat{y}$ and the predicted distribution parameters $\boldsymbol{\mu}$. Total uncertainty is computed in the same way as deep ensembles by considering the predictive entropy. However the data uncertainty was considered as the average entropy over M samples for each ensemble member, in this case the data uncertainty is computed by considering the expected value of the differential entropy of the predicted probability distribution.

$$KU = MI(\hat{y}, \boldsymbol{\mu}) \tag{47}$$

$$MI(\hat{y}, \boldsymbol{\mu}) = entropy(\hat{y}) - \mathbb{E}\left[entropy(p(y|\boldsymbol{\mu})\right] \tag{48}$$

The total uncertainty TU_{ik} for image i and class k is given by the predictive entropy. This is identical to how the total uncertainty is computed for deep ensembles, the predictive entropy is given by equation 21.

$$TU_{ik} = entropy(\hat{y}_{ik}) \tag{49}$$

A measure of data uncertainty DU is computed by considering the expected differential entropy of the the output distribution $p(y|\boldsymbol{\mu})$.

$$DU_{ik} = \mathbb{E}\left[entropy(p(y_{ik}|\boldsymbol{\mu}_{ik})\right] \tag{50}$$

Malinin and Gales (2018) show how the expected differential entropy of a Dirichlet distribution is calculated in equation 53, where the distribution has parameters $\boldsymbol{\alpha}$.

$$\boldsymbol{\alpha} = (\alpha_1, , , \alpha_K) \tag{51}$$

$$\alpha_0 = \sum_{k=1}^{K} \alpha_k \tag{52}$$

$$\mathbb{E}\left[entropy(p(y|\boldsymbol{\alpha})\right] = -\sum_{k=1}^{K} \frac{\alpha_i}{\alpha_0} \left[\psi(\alpha_i + 1) - \psi(\alpha_0 + 1)\right] \tag{53}$$

Where $\psi(\alpha)$ is the Digamma function given by equation 54.

$$\psi(\alpha) = \frac{d}{d\alpha} \ln\left[\Gamma(\alpha)\right] = \frac{\Gamma'(\alpha)}{\Gamma(\alpha)} \tag{54}$$

As the distilled model trained in this project outputs a mixture of Beta distributions the formula for the expected differential entropy of the Dirichlet can be easily adapted to give the data uncertainty for each class.

$$DU_{ik} = \mathbb{E}\left[entropy(p(y_{ik}|\alpha_{ik}, \beta_{ik})\right] \tag{55}$$

$$\begin{aligned}\mathbb{E}\left[entropy(p(y_{ik}|\alpha_{ik}, \beta_{ik})\right] = &-\frac{\alpha_{ik}}{\alpha_{ik} + \beta_{ik}} \left[\psi(\alpha_{ik} + 1) - \psi(\alpha_{ik} + \beta_{ik} + 1)\right] \\ &- \frac{\beta_{ik}}{\alpha_{ik} + \beta_{ik}} \left[\psi(\beta_{ik} + 1) - \psi(\alpha_{ik} + \beta_{ik} + 1)\right]\end{aligned} \tag{56}$$

Now the total and data uncertainty have been computed through the predictive entropy and the expected differential entropy of the output distribution the Knowledge uncertainty can be computed.

$$KU_{ik} = TU_{ik} - DU_{ik} \tag{57}$$

7 Data, Methodology & Training

This section will outline the procedure for applying the techniques detailed in the previous section to the CheXpert dataset. Specifically the procedure for training a deep ensemble with random initialisation and a deep ensemble with pretrained weights can be found in this section. Two models will be distilled from the pretrained deep ensemble, one will be trained on just the Cardiomegaly class with the Ensemble Distribution Distillation method that outputs the Dirichlet distribution to model the ensemble predictions, the other will be trained on all 5 of the competition classes but will output a mixture of Beta Distributions. The Dirichlet distilled model is used to validate the implementation of the Beta-mixture distilled model. Python 3.6 was used to create all scripts used for this project.

7.1 Data Partitioning, Cleaning & Augmentation

The images provided in the dataset are accompanied by two csv files for the competition training and validation files which for each image contain the file path and label for each of the 14 diseases. As mentioned previously the validation set is used as the test set and a simple script was written to take 1000 of the images with no uncertain labels in the 5 competition classes from the training set to use as the validation set.

The JPG images are loaded using the cv2 module, then the PIL module is used so that transformations can be performed on the images for data augmentation purposes. Affine transformations are used on the training data to increase the size of the training set, specifically during training when each mini-batch is loaded each image is augmented by 2 additional images where the original image has a random rotation of ± 15 degrees and a random scale factor between [0.95,1.05] is applied to create an additional image with the same label. All transformations used for augmentation are applied using the put PyTorch module.

Each image is provided in a 320x320 pixel grey-scale format, after augmentation each image needs to be normalised to reduce source dependent variation, as is standard practice in machine learning. This is shown simply by equation 58 which ensures each image's pixels have a mean of 0 and a standard deviation of 1.

$$\mathbf{x} = \frac{\mathbf{x}_{\text{raw}} - \mu_{\text{pixels}}}{\sigma_{\text{pixels}}} \tag{58}$$

Each image is then padded with pixels of values 0 to increase the size of the image to 400x400 pixels. This ensures features near the edge of the original images are captured by the convolutions of a CNN. The data augmentation, normalisation and padding is applied to all models trained. Table 2 lists the policy for dealing with uncertain labels for each class during training, due to the partitioning of data both test and validation sets contain no uncertain labels.

7.2 Model Architectures & Training

The top performing model in the CheXpert competition uses an an ensemble of 6 models each with a different architecture(Pham et al., 2019), therefore ensembles in this project will contain 6 CNNs. The following architectures were used in this model DenseNet-121, DenseNet-169, DenseNet-201, Inception-ResNet-v2, Xception and NASNetLarge. Due to limited computational resources available training some of these architectures is not feasible, therefore the DenseNet-121 architecture was used for all models in ensembles and distilled models as it is one of the smaller arc. As this project aims to assess the uncertainty estimation and not outperform the original ensemble on accuracy therefore as long as reasonable accuracy is achieved the findings of this project remain valid. DenseNet-121 was

chosen as it is one of the smaller architectures in the ensemble and it is also the best performing architecture in the ensemble (Pham et al., 2019). Using one architecture for the deep ensembles is also more practical from the perspective of using a pretrained model as a starting point for the deep ensemble as then only one pretrained model is required. When a different architecture is used for each of the M models in the ensemble, M different pretrained models are required which may not be readily available. In summary the deep ensembles trained with random initialisation and pretrained weights are composed of 6 models all with the DenseNet-121 architecture and the distilled models also use the same architecture.

The code to train a single model on the CheXpert dataset and the pretrained model used for the pretrained ensemble was provided by Ye et al. (2020). PyTorch is used to train all models on the GPU cluster. Due to computation constraints ensemble members could not be trained in parallel meaning training two ensembles and two distilled models took up a large amount of time. The source code and pretrained model is also available at the following GitHub repository https://github.com/jfhealthcare/Chexpert

7.2.1 Deep Ensembles - Random Initialisation

The work done by Lakshminarayanan, Pritzel and Blundell (2017) has shown empirically that deep ensembles where the individual deep neural networks weights are initialised randomly outperform other methods with respect to uncertainty estimation. Therefore as a baseline to compare other models to a deep ensemble with random initialisation of the weights of each ensemble member will be trained.

As previously mentioned the DenseNet-121 architecture is used for all models. Each model has 5 output nodes with a sigmoid activation function used on the final layer and is trained using the cross entropy loss function given by equation 16. The exact same architecture and loss function is used for pretrained deep ensemble.

The weights θ of each of the CNNs are initialised by random draws from a normal distribution with mean 0 and variance 0.1 shown by equation 59.The biases of the network are initialised at zero.

$$\theta \sim \mathcal{N}(0, 0.1) \tag{59}$$

A different random seed is set for each of the models in the ensemble before each model is trained for 6 epochs. During training a model is tested on the validation set after every 500 steps. The model that performs gives the highest value for AUC on the validation set during training is the model used as an ensemble member. The Adam optimiser is used for training with a batch size of 8 and a learning rate of $1\text{x}10^{-4}$ is used which is reduced by a factor of 10 after 2 epochs.

7.2.2 Deep Ensembles - Pretrained

The deep ensemble with a pretrained weights is trained in attempt to provide the comparable accuracy and uncertainty estimation to the baseline with reduced training time. Each model in the ensemble is initialised with the same weights. These weights are taken from a pretrained model provided by Ye et al. (2020).

A different random seed is used to train each of the models in the ensemble. Each model is trained for 3 epochs, meaning half the training time is required for the pretrained ensemble compared to the baseline. The Adam optimiser is used with a batch size of 8 and a learning rate of $1\text{x}10^{-4}$ is used which is reduced by a factor of 10 after 2 epochs. Each model is tested on the validation set every 500 steps and the model that gives the best AUC throughout training is saved to be used as a member of the ensemble.

7.2.3 Ensemble Distribution Distillation

Another of the objectives of this project is to distill the pretrained ensemble into a single model via EnD^2 to attempt to preserve the ensembles uncertainty estimation. First this required the predictions of all ensemble members on the training and validation sets from which the distilled model will be trained on. This was done with a simple script which produced two csv files containing the path of each image and all the ensemble predictions for each of the 5 labels for the training and validation sets.

To extend the distribution distillation procedure proposed by Malinin, Mlodozeniec and Gales (2020) to multi-label classification, a new loss function was derived (equation 40). To validate this implementation the original distribution distillation procedure to distill a model that outputs a Dirichlet distribution was applied to just one of the competition classes so that the multi-label approach that outputs a mixture of Beta distributions proposed by this project could be validated.

The Cardiomegaly class was the class chosen to implement the Dirichlet distillation on. Whereby only the predictions from this class where used form the transfer dataset produced by the pretrained ensemble. For this task the distilled model has 2 output nodes with an exponential activation function to predict (α_1, α_2) to parameterise a Dirichlet distribution to fit all ensemble predictions for the Cardiomegaly class. The exponential activation function ensures $\alpha_1, \alpha_2 > 0$. The loss function is given by equation 32.

The Beta-mixture distillation model has 10 output nodes with an exponential activation function to predict the parameters $\alpha_1, , , \alpha_5$ & $\beta_1, , , \beta_5$. The loss function is given by equation 40. Both of the distilled models were trained with the same hyperparameters and the DenseNet-121 architecture.

The weights were initialised in the same way as the deep ensemble with random initialisation according to equation 59. The distilled models were trained for 6 epochs using the Adam optimiser with learning rate of $1\text{x}10^{-3}$ which is reduced by a factor of 10 after 2 epochs and again after 4 epochs. Target smoothing (equation 42) was applied to the ensemble predictions with γ=$1\text{x}10^{-4}$. A temperature annealing schedule was used in training described by section 6.3.3. During training for the first epoch a temperature of $T = 5$ was used, for the second epoch this was reduced to $T = 3$ and for all subsequent training this was reduced to $T = 1$ which is equivalent to no temperature annealing. The distilled model was tested on the transfer validation set every 500 steps and the model that produces the lowest loss function throughout training is saved as the distilled model to use for testing. No temperature annealing is applied during testing on the validation set.

The loss functions used are not included as part of PyTorch's in built loss functions. Therefore significant work was done to implement these custom loss functions and make them compatible with the AutoDiff for backpropagation.

8 Results

After training was completed all models were tested on the test set of 200 images with no uncertain labels on the 5 competition classes.

8.1 Accuracy

The accuracy of the ensembles and distribution distilled models are calculated to validate their implementations. Table 3 shows that both the ensembles trained produce good accuracy results which are comparable to the top 100 entries in the CheXpert competition. These accuracy results validate that the ensembles were trained correctly. The results for the accuracy of each ensemble also show that the pretrained ensemble benefits from the transfer learning from the pretrained weights. The implementation of ensemble distribution distillation on the

	Ensemble - Pretrained	**Ensemble - Random Init.**
Mean Accuracy	0.810	0.795
Mean AUC	0.883	0.876

Table 3: Mean Accuracy & AUC of Deep Ensembles

pretrained on ensemble also needs to be validated by checking that the accuracy of the pretrained ensemble was preserved during distillation. For the distillation into a mixture of Beta distributions the accuracy was compared to the Dirichlet distilled model to check that the extension proposed by this project does not have an adverse effect on the accuracy of the distillation procedure.

	Ensemble-Pretrained	**EnD2-Dirichlet**	**EnD2-Beta Mixture**
Cardiomegaly: Accuracy	0.790	0.765	0.775
Cardiomegaly: AUC	0.816	0.834	0.828

Table 4: AUC and Accuracy of Ensemble and Distlled Models on Cardiomegaly Class

Table 4 shows the accuracy of the pretrained ensemble which was subject to distillation and the accuracy of the Dirichlet distilled and Beta-mixture distilled models on the Cardiomegaly class. Note that the Dirichlet distilled model is only trained on the Cardiomegaly labels while all other models are trained on all 5 competition labels. Table 4 shows that both distillation procedures have been successful with the accuracy of the ensemble preserved. Extending ensemble distribution distillation to a output a mixture of Beta distributions has not shown adverse effects on the performance of ensemble distribution distillation. The Beta-mixture distilled model exhibits slightly higher accuracy than the Dirichlet distilled model this may be due to the model benefiting through multitask learning by sharing knowledge between the 5 classes it is predicting. The Dirichlet distilled model only has 1 class from which to learn from.

	Ensemble Pretrained		**Ensemble Random Init.**		**EnD^2**	
Class	**AUC**	**Acc.**	**AUC**	**Acc.**	**AUC**	**Acc.**
Atelectasis	0.822	0.730	0.820	0.725	0.825	0.745
Cardiomegaly	0.816	0.790	0.840	0.785	0.828	0.775
Consolidation	0.907	0.795	0.904	0.800	0.897	0.800
Edema	0.941	0.865	0.911	0.830	0.903	0.840
Pleural Effusion	0.927	0.870	0.907	0.845	0.894	0.855
Mean	0.883	0.810	0.876	0.795	0.869	0.803

Table 5: AUC and Accuracy of Ensembles and Distribution Distilled Model

Table 5 shows the accuracy & AUC of both ensembles and the distilled model on the 5 competition classes. Significant variation is seen in performance across the 5 classes. This is likely due to the fact that the dataset contains few positive labels for some of the classes leading to overfitting on those positive labels.

8.2 Uncertainty

This section lists a number of metrics used to assess the in domain and out of domain uncertainty performance of the two ensembles and the distilled model.

8.2.1 Negative Log Likelihood

The negative log likelihood(NLL) is one of the uncertainty metrics used in this project for in domain uncertainty estimation. The NLL is a proper scoring rule meaning that an optimum score corresponds to a perfect set of predictions. The NLL is given by equation 60, where y is the label and $\hat{y}$ is the prediction. The NLL gives a positive number where 0 corresponds to perfect predictions and the lower the NLL the more accurate the uncertainty estimation is.

$$NLL = \frac{-1}{NK} \sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log \hat{y}_{ik} + (1 - y_{ik}) \log(1 - \hat{y}_{ik}) \tag{60}$$

Table 6 shows the NLL scores for the two ensembles and the distilled model for the test set. The pretrained ensemble exhibits a better score than the ensemble with randomly initialised weights suggesting the diversity has not been adversely effected at all by using pretrained weights. However the distilled model does show a significantly worse NLL score than the pretrained ensemble it was distilled from. Table 7 shows the NLL of the Cardiomegaly class test set

	Ensemble - Random Init.	**Ensemble - Pretrained**	**EnD^2**
NLL	0.455	**0.416**	0.474

Table 6: Negative Log Likelihood of Test Set

predictions for both distilled models. Note that the Dirichlet distilled model is only trained on the Cardiomegaly class. Although the Dirichlet distilled model exhibits a slightly better NLL score the difference is marginal suggesting that the extension of ensemble distribution distillation to a multi-label classification task has not significantly reduced the performance of a distribution distilled models uncertainty estimation.

	EnD^2 - Beta-mixture	**EnD^2 - Dirichlet**
NLL	0.480	**0.475**

Table 7: NLL on Cardiomegaly class

8.2.2 Expected Calibration Error

The expected calibration error (ECE) is computed on the test set. The ECE relates to wether the probability predictions of the model relate to frequency with which the predictions are correct. For example when a model predicts a value of 0.8, the model should be accurate 80% of the time. This is detailed by

equation 9, lower values for ECE equate to more accurate uncertainty estimation. To calculate the ECE predictions were separated into 5 bins with equally spaced probability intervals.

	Ensemble - Random Init.	**Ensemble - Pretrained**	**EnD^2**
ECE	0.373	**0.210**	0.425

Table 8: Expected Calibration Error of Test Set

Table 8 shows the ECE results on the test set. The pretrained ensemble again performs better than the randomly initialised ensemble for this metric. The distribution distilled model shows significantly worse performance than the pretrained ensemble from which it was distilled.

	EnD^2 - Beta-mixture	**EnD^2 - Dirichlet**
ECE	0.334	**0.327**

Table 9: ECE on Cardiomegaly class

Table 9 shows no significant reduction in uncertainty estimation with respect to ECE when ensemble distribution distillation is extended to this multi-label problem.

8.2.3 Patch Accuracy vs. Patch Uncertainty

Patch Accuracy vs. Patch Uncertainty(PAvPU) assesses the quality of the uncertainty estimation by considering the proportion of predictions that are either accurate and certain or inaccurate and uncertain. A higher value for PAvPU equates to more accurate uncertainty estimation. This metric requires an uncertainty threshold to be defined so that predictions can be defined as either certain or uncertain. This was done for each model by simply grid-searching the uncertainty threshold and saving the threshold for each model which gives the highest value for PAvPU.

	Ensemble - Random Init.	**Ensemble - Pretrained**	**EnD^2**
PAvPU	0.815	**0.831**	0.815
p(accurate|certain)	0.820	**0.841**	0.832
p(uncertain|inaccurate)	0.081	0.137	**0.147**

Table 10: Patch Accuracy vs. Patch Uncertainty & Conditional Probabilities of Test Set

Table 10 shows the values for PAvPU as well as the conditional the probability a model is accurate given it is certain and the probability a model is uncertain given it is inaccurate for the 2 ensembles and the distilled model. The pretrained ensemble performs marginally better than the randomly initialised ensemble with respect to PAvPU. The distilled model does not entirely preserve the uncertainty estimation of the ensemble yet the reduction from the ensemble to the distilled model is marginal.

	EnD^2 - Beta-mixture	**EnD^2 - Dirichlet**
PAvPU	0.791	0.791

Table 11: Patch Accuracy vs. Patch Uncertainty on Cardiomegaly Class

Table 11 shows that extending ensemble distribution distillation to multi-label classification has not had any significant effect on this uncertainty estimation metric.

8.2.4 Out of Distribution Detection

This section outlines experiments where out of domain inputs were fed to the model. Ideally in this scenario a model should give completely uncertain predictions therefore the best performing models in these experiments will exhibit the highest uncertainty. Table 12 shows the average total, knowledge and data predictive uncertainties for the in domain test set. This provides a reference for the out of domain experiments.

	Ensemble - Random Init.	**Ensemble - Pretrained**	**EnD^2**
Total Uncertainty	0.796	0.706	0.837
Data Uncertainty	0.776	0.696	0.539
Knowledge Uncertainty	0.020	0.010	0.298

Table 12: Mean Uncertainty Measures for Test Set

Two sets of out of domain inputs were used, the first of which is simply random Gaussian noise whereby an image the same size as a x-ray from the training set is generated by randomly drawing each pixel from a Gaussian distribution with mean 0 and standard deviation 1. 200 different images were generated using this method, each model then made predictions on these images. Table 13 shows the average total, knowledge and data uncertainty on these out of domain inputs. Only the pretrained ensemble makes predictions with a higher uncertainty than on the in domain test set. Both the randomly initialised ensemble and the distribution distilled model give more confident predictions than on the in domain test set, in particular the EnD^2 model gives extremely confident predictions completely failing in this out of distribution detection challenge.

	Ensemble - Random Init.	**Ensemble - Pretrained**	**EnD^2**
Total Uncertainty	0.589	**0.940**	0.196
Data Uncertainty	0.307	**0.844**	0.078
Knowledge Uncertainty	**0.282**	0.096	0.118

Table 13: Mean Uncertainty Measures for Gaussian Noise OOD Inputs

The second out of domain inputs experiment is uses 200 randomly drawn images from the well known MNIST dataset which contains pictures of hand drawn digits. The results for this experiment is shown in Table 14, all models made predictions with a higher average uncertainty than on the in domain test set. The distribution distilled model performed particularly well on this experiment making significantly more uncertain predictions than the pretrained ensemble.

	Ensemble - Random Init.	**Ensemble - Pretrained**	**EnD^2**
Total Uncertainty	0.879	0.825	**0.940**
Data Uncertainty	**0.833**	0.790	0.591
Knowledge Uncertainty	0.046	0.035	**0.349**

Table 14: Mean Uncertainty Measures for MNIST OOD Inputs

Using the total uncertainty to detect out of distribution samples has been shown to be inconsistent in this study for both the randomly initialised ensemble and the distribution distilled model. However if only the knowledge uncertainty is considered both ensembles exhibit significantly higher knowledge uncertainty on the out of domain inputs relative to the knowledge uncertainty on the in domain test set. Although the distribution distilled model still fails to identify the Gaussian noise as an out of domain input.

9 Conclusions & Further Work

The aim of this project was to study the effect of transfer learning on uncertainty estimation of deep ensembles and to extend a the recently proposed ensemble distribution distillation method to a multi-label classification task.

Diversity between ensemble members is key to accurate uncertainty estimation. Deep ensembles use random initialisation of weights and stochastic gradient descent to provide diversity between models in the ensemble. While using pretrained weights has the potential to vastly reduce the training time required to train a deep ensemble, it removes random initialisation as a source of diversity. Therefore the performance of a pretrained ensemble with respect to uncertainty estimation must be explored.

The experiments in this project found that the pretrained ensemble not only benefited in terms of accuracy from pretraining but performed better than the randomly initialised ensemble on uncertainty estimation metrics negative log likelihood, expected calibration error and PAvPU. The pretrained ensemble also performs well on out of domain input detection tasks on Gaussian noise inputs and inputs from the MNIST dataset. The pretrained ensemble performs significantly better on the Gaussian noise inputs than the randomly initialised ensemble, but performs slightly worse but still respectably on the MNIST inputs. In summary the pretrained ensemble exhibits more accurate uncertainty estimation on the majority of metrics used in this project. These are encouraging results given that the pretrained ensemble required half the training time and does not show adversely effected uncertainty estimation compared to the randomly initialised ensemble. With regards to medical applications these results show the potential to use models pretrained on large publicly available datasets as starting points from which to build deep ensembles without compromising the uncertainty estimation of the ensemble. This could be particularly useful where a hospital would like to deploy such a model on a smaller dataset relevant to their patients that has been collected in a different country with different procedures to the CheXpert dataset.

The other main goal of this project was to extend ensemble distribution distillation to a multi-label task by outputting a mixture of Beta distributions instead of a Dirichlet distribution as is proposed in literature (Malinin, Mlodozeniec and Gales, 2020). The results from the experiments evaluating the accuracy and uncertainty estimation show that the Beta-mixture distilled model performs better than the Dirichlet distilled model with respect to accuracy. This is likely due to the model benefiting from multitask learning whereby the Beta-mixture distilled model generalises better by simultaneously learning about the other labels. While the Dirichlet distilled only learns about one class in this task. There is little difference in the accuracy of uncertainty estimation of the distilled models with respect to the negative log likelihood, expected calibration error and PAvPU. This shows that the extension of ensemble distribution distillation to a multi-label task has been successful satisfying one of the main goals of this project. Although the distilled model preserves the accuracy of the ensemble, the distilled model performs significantly worse than the ensemble on all in domain uncertainty metrics tested. The distilled model also completely fails on the Gaussian noise out of domain inputs making more certain predictions than on the in domain test set. The distilled model does however perform very well on the MNIST out of domain inputs outperforming all other models. In summary the new method of ensemble distribution distillation performs no worse than the method already proposed in literature validating the approach. The accuracy of the ensemble is preserved but the uncertainty estimation is not and the distilled model performs inconsistently on out of domain inputs.

There are numerous avenues for further work on this project. Conditional training is outlined in section 4.3.4 and is one of the key methods introduced by the

top performing model in this dataset (Pham et al., 2019). Conditional training exploits the hierarchy that exists in the classes to gain more accurate results, it would be interesting to study the effect this has on the uncertainty estimation of the deep ensemble.

Other chest X-ray datasets exist of the 14 common thorax diseases such as the NIH dataset (Wang et al., 2017). It would be interesting to see how well the models trained in this project perform on this dataset for example would any reduction in accuracy be accompanied by higher predictive uncertainty or would the model make overconfident inaccurate predictions. It would also be interesting to see if a same quality of uncertainty estimation can be achieved on the NIH dataset by training two deep ensembles one trained entirely on the NIH dataset with random initialisation and another trained using a model pre-trained on the CheXpert dataset but with a reduced training time. This was one of the original aims of the project however the NIH dataset is not available in a down-sampled form so it was not feasible to use with the computational resources available.

Bayesian Neural Networks also provide an avenue for further work as they provide accurate uncertainty estimation in a single model. However they have traditionally struggled to scale to the high dimensionality of computer vision problems. It would however be informative to see what can be achieved with a small model.

10 Appendices

10.1 ROC Curves

This section includes the ROC curves for each class in the test set.

10.1.1 Deep Ensemble - Random Initialisation

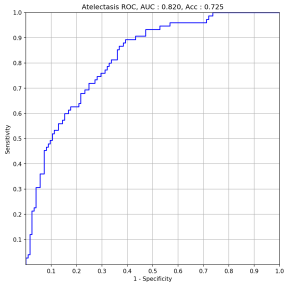

Figure 6: ROC: Atelectasis

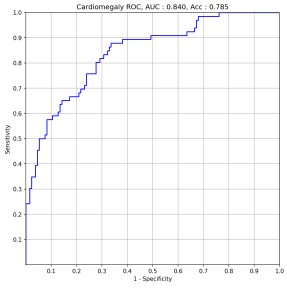

Figure 7: ROC: Cardiomegaly

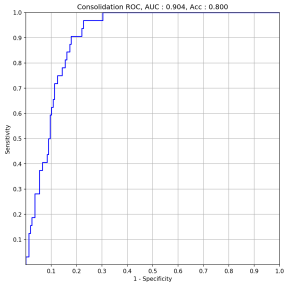

Figure 8: ROC: Consolidation

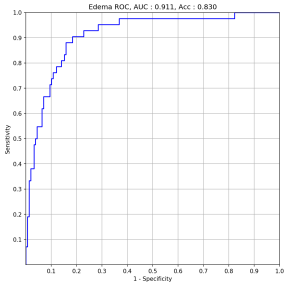

Figure 9: ROC: Edema

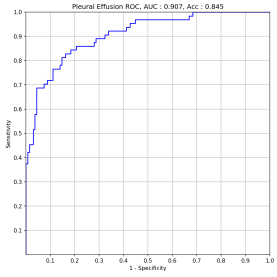

Figure 10: ROC: Pleural Effusion

10.1.2 Deep Ensemble - Pretrained

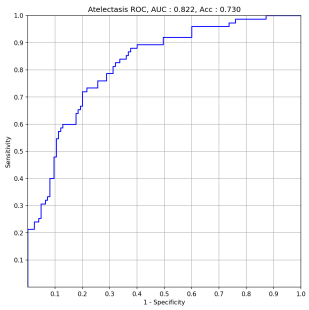

Figure 11: ROC: Atelectasis

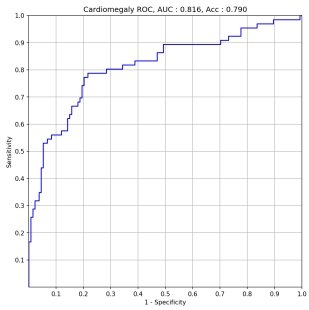

Figure 12: ROC: Cardiomegaly

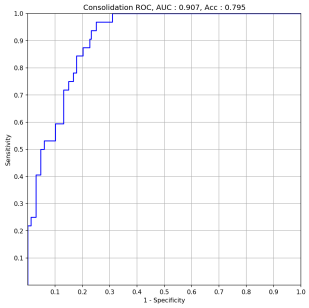

Figure 13: ROC: Consolidation

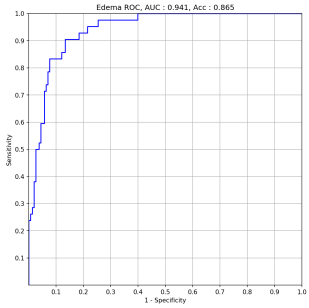

Figure 14: ROC: Edema

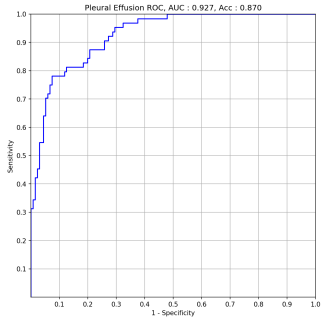

Figure 15: ROC: Pleural Effusion

10.1.3 Ensemble Distribution Distillation - Beta Mixture

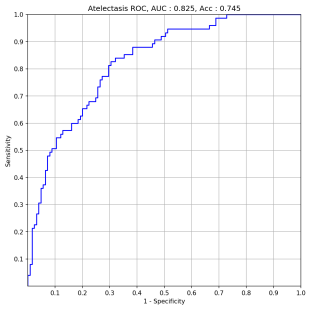

Figure 16: ROC: Atelectasis

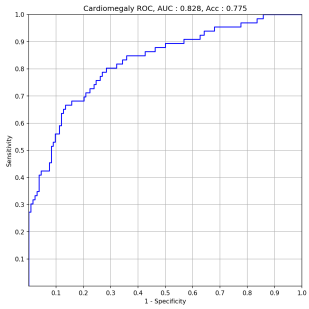

Figure 17: ROC: Cardiomegaly

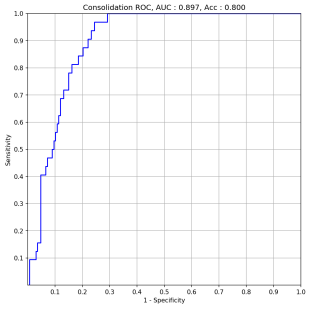

Figure 18: ROC: Consolidation

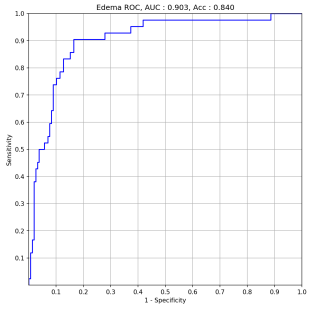

Figure 19: ROC: Edema

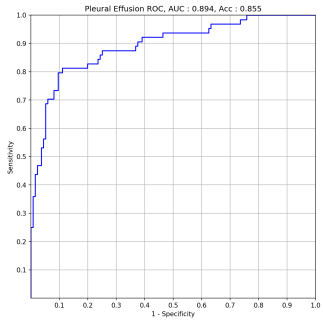

Figure 20: ROC: Pleural Effusion

10.1.4 Ensemble Distribution Distillation - Dirichlet

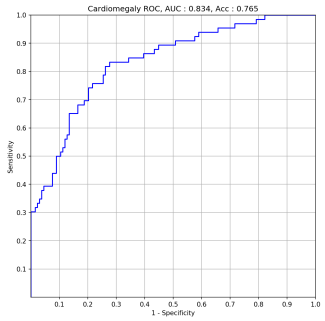

Figure 21: ROC: Cardiomegaly

10.2 12 Point Ethics Checklist

This form must be attached to the dissertation as an appendix.

Department of Computer Science

12-Point Ethics Checklist for UG and MSc Projects

Student Hamish Hornby

Academic Year or Project Title Uncertainty Estimation in Medical Imaging

Supervisor Olga Isupova

Does your project involve people for the collection of data other than you and your supervisor(s)? YES / NO

If the answer to the previous question is YES, you need to answer the following questions, otherwise you can ignore them.

This document describes the 12 issues that need to be considered carefully before students or staff involve other people ('participants' or 'volunteers') for the collection of information as part of their project or research. Replace the text beneath each question with a statement of how you address the issue in your project.

1. *Have you prepared a briefing script for volunteers?* YES / NO

   Briefing means telling someone enough in advance so that they can understand what is involved and why – it is what makes informed consent informed.

2. *Will the participants be informed that they could withdraw at any time?* YES / NO

   All participants have the right to withdraw at any time during the investigation, and to withdraw their data up to the point at which it is anonymised. They should be told this in the briefing script.

3. *Is there any intentional deception of the participants?* YES / NO

   Withholding information or misleading participants is unacceptable if participants are likely to object or show unease when debriefed.

4. *Will participants be de-briefed?* YES / NO

   The investigator must provide the participants with sufficient information in the debriefing to enable them to understand the nature of the investigation. This phase might wait until after the study is completed where this is necessary to protect the integrity of the study.

References

Amodei, D., Olah, C., Steinhardt, J., Christiano, P.F., Schulman, J. and Mané, D., 2016. Concrete problems in AI safety. *Corr*. 1606.06565, Available from: http://arxiv.org/abs/1606.06565.

Balan, A.K., Rathod, V., Murphy, K. and Welling, M., 2015. Bayesian dark knowledge. *Corr*, abs/1506.04416. 1506.04416, Available from: http://arxiv.org/abs/1506.04416.

Brier, G.W., 1950. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1), pp.1–3. Available from: https://doi.org/10.1175/1520-0493(1950)078<0001:VOFEIT>2.0.CO;2.

Depeweg, S., Hernández-Lobato, J., Doshi-Velez, F. and Udluft, S., 2017. Decomposition of uncertainty for active learning and reliable reinforcement learning in stochastic systems.

Duvenaud, D., Maclaurin, D. and Adams, R., 2016. Early stopping as nonparametric variational inference. In: A. Gretton and C.C. Robert, eds. *Proceedings of the 19th international conference on artificial intelligence and statistics*. Cadiz, Spain: PMLR, *Proceedings of Machine Learning Research*, vol. 51, pp.1070–1077. Available from: http://proceedings.mlr.press/v51/duvenaud16.html.

Gal, Y., 2016. *Uncertainty in deep learning*. Ph.D. thesis. University of Cambridge.

Gal, Y. and Ghahramani, Z., 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *Proceedings of the 33rd international conference on international conference on machine learning - volume 48*. JMLR.org, ICML'16, p.1050–1059.

Gal, Y., Hron, J. and Kendall, A., 2017. Concrete dropout. In: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, eds. *Advances in neural information processing systems 30*. Curran Associates, Inc., pp.3581–3590. Available from: http://papers.nips.cc/paper/6949-concrete-dropout.pdf.

Goodfellow, I.J., Bengio, Y. and Courville, A., 2016. *Deep learning*. Cambridge, MA, USA: MIT Press. http://www.deeplearningbook.org.

Guo, C., Pleiss, G., Sun, Y. and Weinberger, K.Q., 2017. On calibration of modern neural networks. *Corr*, abs/1706.04599. 1706.04599, Available from: http://arxiv.org/abs/1706.04599.

Gustafsson, F.K., Danelljan, M. and Schön, T.B., 2019. Evaluating scalable bayesian deep learning methods for robust computer vision. *Corr*, abs/1906.01620. 1906.01620, Available from: http://arxiv.org/abs/1906.01620.

Hinton, G., Vinyals, O. and Dean, J., 2015. Distilling the knowledge in a neural network. *Nips deep learning and representation learning workshop*. Available from: http://arxiv.org/abs/1503.02531.

Houlsby, N., Huszar, F., Ghahramani, Z. and Lengyel, M., 2011. Bayesian active learning for classification and preference learning. *Corr*, abs/1112.5745. Available from: http://dblp.uni-trier.de/db/journals/corr/corr1112.htmlabs-1112-5745.

Ilg, E., Çiçek, Ö., Galesso, S., Klein, A., Makansi, O., Hutter, F. and Brox, T., 2018. Uncertainty estimates for optical flow with multi-hypotheses networks. *Corr*, abs/1802.07095. 1802.07095, Available from: http://arxiv.org/abs/1802.07095.

Irvin, J., Rajpurkar, P., Ko, M., Yu, Y., Ciurea-Ilcus, S., Chute, C., Marklund, H., Haghgoo, B., Ball, R.L., Shpanskaya, K.S., Seekins, J., Mong, D.A., Halabi, S.S., Sandberg, J.K., Jones, R., Larson, D.B., Langlotz, C.P., Patel, B.N., Lungren, M.P. and Ng, A.Y., 2019. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. *Corr*, abs/1901.07031. 1901.07031, Available from: http://arxiv.org/abs/1901.07031.

Kendall, A. and Gal, Y., 2017. What uncertainties do we need in bayesian deep learning for computer vision? *Corr*, abs/1703.04977. 1703.04977, Available from: http://arxiv.org/abs/1703.04977.

Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In: F. Pereira, C.J.C. Burges, L. Bottou and K.Q. Weinberger, eds. *Advances in neural information processing systems 25*. Curran Associates, Inc., pp.1097–1105. Available from: http://papers.nips.cc/paper/4824.

Lakshminarayanan, B., Pritzel, A. and Blundell, C., 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, eds. *Advances in neural information processing systems 30*. Curran Associates, Inc., pp.6402–6413.

Louizos, C. and Welling, M., 2016. Structured and efficient variational deep learning with matrix gaussian posteriors. In: M.F. Balcan and K.Q. Weinberger, eds. *Proceedings of the 33rd international conference on machine learning*. New York, New York, USA: PMLR, *Proceedings of Machine Learning Research*, vol. 48, pp.1708–1716. Available from: http://proceedings.mlr.press/v48/louizos16.html.

Malinin, A. and Gales, M., 2018. Predictive uncertainty estimation via prior networks. In: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi and R. Garnett, eds. *Advances in neural information processing systems 31*. Curran Associates, Inc., pp.7047–7058. Available from: http://papers.nips.cc/paper/7936.

Malinin, A., Mlodozeniec, B. and Gales, M., 2020. Ensemble distribution distillation. *International conference on learning representations*. Available from: https://openreview.net/forum?id=BygSP6Vtvr.

Mukhoti, J. and Gal, Y., 2018a. Evaluating bayesian deep learning methods for semantic segmentation. *Corr*, abs/1811.12709. 1811.12709, Available from: http://arxiv.org/abs/1811.12709.

Mukhoti, J. and Gal, Y., 2018b. Evaluating bayesian deep learning methods for semantic segmentation. *Corr*, abs/1811.12709. 1811.12709, Available from: http://arxiv.org/abs/1811.12709.

Naeini, M.P., Cooper, G.F. and Hauskrecht, M., 2015. Obtaining well calibrated probabilities using bayesian binning. *Proceedings of the twenty-ninth aaai conference on artificial intelligence*. AAAI Press, AAAI'15, p.2901–2907.

Neal, R.M., 1996. *Bayesian learning for neural networks*. Berlin, Heidelberg: Springer-Verlag.

Nguyen, A.M., Yosinski, J. and Clune, J., 2014. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *Corr*, abs/1412.1897. 1412.1897, Available from: http://arxiv.org/abs/1412.1897.

Nix, D.A. and Weigend, A.S., 1994. Estimating the mean and variance of the target probability distribution. *Proceedings of 1994 ieee international conference on neural networks (icnn'94)*. vol. 1, pp.55–60 vol.1. Available from: https://doi.org/10.1109/ICNN.1994.374138.

Papernot, N., McDaniel, P.D., Wu, X., Jha, S. and Swami, A., 2015. Distillation as a defense to adversarial perturbations against deep neural networks. *Corr*, abs/1511.04508. 1511.04508, Available from: http://arxiv.org/abs/1511.04508.

Pearce, T., Zaki, M., Brintrup, A. and Neely, A., 2018. Uncertainty in neural networks: Bayesian ensembling.

Pham, H.H., Le, T.T., Tran, D.Q., Ngo, D.T. and Nguyen, H.Q., 2019. Interpreting chest x-rays via cnns that exploit disease dependencies and uncertainty labels. *medrxiv*. https://www.medrxiv.org/content/early/2019/11/29/19013342.full.pdf, Available from: https://doi.org/10.1101/19013342.

Quionero-Candela, J., Sugiyama, M., Schwaighofer, A. and Lawrence, N., 2009. Dataset shift in machine learning.

Rajpurkar, P., Irvin, J., Zhu, K., Yang, B., Mehta, H., Duan, T., Ding, D.Y., Bagul, A., Langlotz, C., Shpanskaya, K.S., Lungren, M.P. and Ng, A.Y., 2017. Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. *Corr*, abs/1711.05225. 1711.05225, Available from: http://arxiv.org/abs/1711.05225.

Settles, B., 2010. *Active learning literature survey.*

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(56), pp.1929–1958. Available from: `http://jmlr.org/papers/v15/srivastava14a.html`.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. and Fergus, R., 2014. Intriguing properties of neural networks. *International conference on learning representations*. Available from: `http://arxiv.org/abs/1312.6199`.

Van Eeden, S., Leipsic, J., Paul Man, S.F. and Sin, D.D., 2012. The relationship between lung inflammation and cardiovascular disease. *American journal of respiratory and critical care medicine*, 186(1), pp.11–16. PMID: 22538803. `https://doi.org/10.1164/rccm.201203-0455PP`, Available from: `https://doi.org/10.1164/rccm.201203-0455PP`.

Wang, X., Peng, Y., Lu, L., Lu, Z., Bagheri, M. and Summers, R., 2017. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. *2017 ieee conference on computer vision and pattern recognition (cvpr)*. pp.3462–3471.

Ye, W., Yao, J., Xue, H. and Li, Y., 2020. Weakly supervised lesion localization with probabilistic-cam pooling. `2005.14480`, Available from: `https://github.com/jfhealthcare/Chexpert`.

Zhang, G., Sun, S., Duvenaud, D. and Grosse, R., 2018. Noisy natural gradient as variational inference. In: J. Dy and A. Krause, eds. *Proceedings of the 35th international conference on machine learning*. Stockholmsmässan, Stockholm Sweden: PMLR, *Proceedings of Machine Learning Research*, vol. 80, pp.5852–5861. Available from: `http://proceedings.mlr.press/v80/zhang18l.html`.