University College London
Engineering Faculty
Department of Computer Science

Evidential Deep Learning for Uncertainty Estimation in Object Detection

Harry Rose

Supervised by:

Dr Simon Julier
Dr Murat Sensoy
Dr Federico Cerutti

September 2019

Abstract

This work focuses on uncertainty estimation in deep learning models used for object detection. Notably the use of Evidential Deep Learning, a method inspired by Subjective Logic to enhance classifications with uncertainty estimates. Uncertainty awareness is important to improving the robustness of object detectors when dealing with out of distribution examples and potential adversarial attacks. Little work exists in the specific application of uncertainty awareness in object detection and existing methods rely on sampling based approaches which are impractical for most safety critical applications. Thus motivating the need for real-time uncertainty aware object detectors. This report shows how to modify existing architectures into evidential frameworks for almost free, using the outputs of deterministic networks to parameterise a Dirichlet distribution over class labels. Considering the equivalence of subjective opinions and Dirichlet distributions, the parameters of the model can then be used to determine the epistemic uncertainty for a classification in the same pass. Considering current state-of-art object detection systems, YOLO is chosen for its extremely fast inference. The report demonstrates how one can modify and train the YOLO in an evidential way to enhance real-time detections with real-time uncertainty scores.

Contents

Chapter 1

Introduction

1.1 Motivation

Deep learning techniques have been shown to have extremely impressive results across a breadth of tasks. Drastically improving the state of the art in speech recognition, translation models and object recognition [25]. A renaissance moment for the computer vision community came with AlexNet [23]. The paper was seen as a breakthrough moment that used convolutions neural networks to roughly halve the error rate in the object recognition task ImageNet [7] to classify over 1.2 million images from 1000 classes. These models however remain largely blackbox predictors with little interpretability as to what they have learnt and actually understand. As such deep learning models are ignorant about their own confidence and it is impossible to know whether the model is making sensible predictions or simple guessing at random.

A tragic example from 2016 came with Tesla. The company's statement reported that the Tesla's Autopilot system was unable to detect the white side of a trailer against the brightly lit sky and hence the brake was not applied [1]. Considering uncertainty in safety critical systems will likely lead to better decision making in the face of erroneous predictions as well as increase public perception and confidence regarding autonomous systems. Understanding the limitations of the models instead of following their outputs blindly is paramount in a world where the potential to impact people in meaningful ways is greater than ever. Whether in the form of autonomous vehicles or robotic surgeons, detection systems and the classifiers within them will increasingly become key components of these technologies. Furthermore, the information they provide will likely propagate forward into tangible decisions and actions that affect the environment around us. With the push for autonomy in seemingly all aspects of our lives, the importance of understanding what our models do and do not know is crucial.

In standard Deep Neural Networks (DNNs), the objective function is typically optimized in order to minimize the loss on training data [18]. This doesn't necessarily reflect the true objective which would mean the model can generalise well to new unseen data. Secondly predictions from large DNNs are the combination of millions of parameters and nonlinearities; mispredictions are thus hard to understand. Considering the output of a DNN, a classification point estimate is usually interpreted to be the confidence probabilities for each class label. This is wrong and dangerous since most neural networks pass the output through a softmax activation function such that the output are normalized scores. DNNs thus tend to yield overconfident predictions which lead to false positives with extremely high scores [11, 43].

Sensoy [38] demonstrated this overconfidence explicitly using a Convolutional Neural Network to classify an image of the digit 1 from the MNIST dataset. Figure 1.5 below from Sensoy's paper plots the estimated probability for each class with rotation angle. For small perturbations of the digit, the classification probability is greatest for the true label displaying high probability for the digit 1 and near 0 for the others. However once the image begins to be rotated noticeably the probability of the true class begins to fall while probabilities for the wrong classes increase. Suggesting more confidence in the wrong label than the truth. At 60 degrees rotation the probability that the image is a 2 exceeds the probability of the true class. At roughly

90 degrees rotation the probability implies near certainty that the image is a 2 instead of a 1. Since the training of the classifier is done simply to minimize the loss in prediction, the model is completely ignorant to its own confidence.

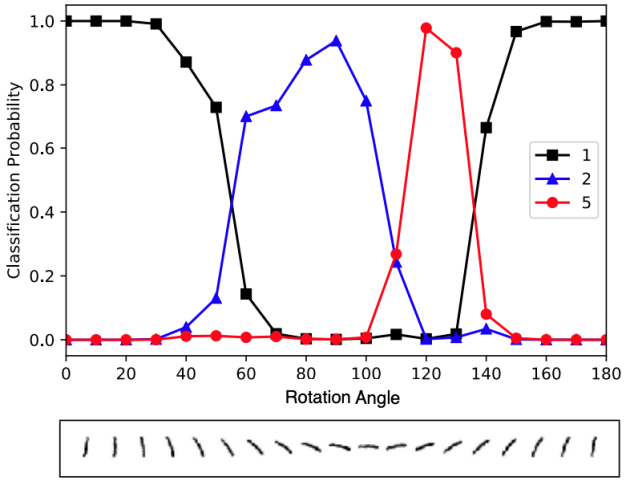


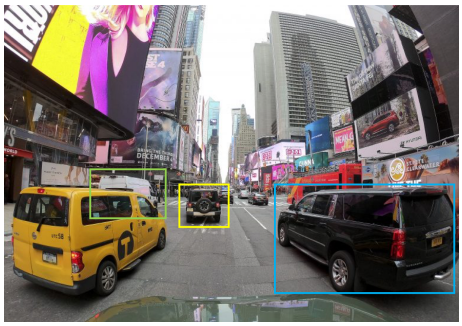
Figure 1.1: Classification probabilities for three labels 1, 2 and 5 for the rotated MNIST digit 1 using the CNN LeNet-5. Image taken from [38].

A major challenge and current limitation of these models is dealing with unknown examples. Standard DNNs use a softmax function to normalise output scores to produce class probabilities for class labels seen in training. But if the DNN has been trained to recognise different breeds of dogs, how should it classify an image of a cat. The model should have the ability to output "I don't know." This motivates the problem of uncertainty estimation and gaining an understanding of what our model does and doesn't know.

In this work, the problem setting is object detection due to the vast number of real world applications and safety critical nature of the problem. Introducing uncertainty estimation to object detection systems used in robotics and autonomous driving is paramount due to the high risk nature of these applications. In these agents an object detection system would fit into a wider end-to-end system for operating in the environment. Predictions made by an object detector may be fused with other sensor predictions and in turn be used to perceive the surrounding environment. The role of an agent usually extends from simply perception but to that of a complex goal-driven system [41]. Information derived in object detection will ultimately be propagated forward to determine the actions and decision made by the agent.

For example, if the detection system in an autonomous car detects a nearby object as a pedestrian but misjudges the location then the resulting decision of whether to continue, break and veer to avoid the pedestrian may be sub-optimal and result in fatality. Or in the Tesla example discussed above, if the bright light around the trailer was detected as an out of distribution setting, the detector should be aware. Awareness of uncertainty could then raise an alert or simply brake to limit the risk of catastrophe.

(a) Plain image



(b) Image with possible detections

Figure 1.2: Motivating problem 1 - cars merging between lanes may appear in orientations different to those experienced in training. Additionally, cars may be partially occluded which need to be detected.

(a) Rain blurring the lense

(b) Intense sunlight imposing the image

Figure 1.3: Motivating problem 2 - in both cases the car is directly infront of us. However the environment, rain in the left case and intense sunlight in the right add difficulty to the detection and to the classification. In these environments heightened awareness is crucial and high uncertainty should be given to cases like these.

A common dangerous assumption in Deep Learning is the conditions which our models are trained our will be the same which our model will encounter in deployment. As discussed above for a classifier this implies that objects it will need to classify will be of the same set of classes it was trained on. Since the environment in which object detectors we hope to deploy are ever changing we need to handle for so called open-set conditions [3] whereby distributions of each class can be ever changing and when novel unseen classes are detected these should be included going forward.

Out of distribution detection will be the focus of this work. We desire detectors and classifiers within them to state ignorance and say it doesn't know when presented with unknown objects. Awareness of out of distribution settings could then be used to solve other problems such as incremental learning. Since features, texture and appearance of objects which were seen in training can be different in test time and thereby the system is expected to be able to learn from new samples of previously seen classes. In addition we expect in reality to deal with classes we have not seen before therefore we expect a robust detection system to be able to expand its knowledge and learn representations of new classes without jeapordising performance on the other known classes. Solutions to these problem settings can be solved with knowledge of uncertainty estimation.

Evidential Deep Learning [38] is the focused method for uncertainty estimation in this work. Evidential Deep Learning has theoretical grounding in Subjective Logic [20] and allows for uncertainty estimation without extensive modification of existing deep frameworks in real time. The proposed evidential framework uses subjective logic to decompose predictions made into belief mass for each classification and with an uncertainty measure. This method has shown to improve on state of the art uncertainty benchmarks: dealing with out of distribution examples and robustness against adversaries. Using this approach, Sensoy demonstrated in the same example of the rotated MNIST digit 1. The modified network was able recognise the out of distribution query and output high uncertainty as the rotation perturbation increased. In the below image 1.4 taken from the paper, now when the digit is rotated past 30 degrees, the classification probability for the true class falls to near zero, while the uncertainty score increases. At 50 degrees rotation, the model is near 100% uncertain and returns near zero probability scores for digit labels. Clearly detecting the rotated digit is out of distribution and signalling ignorance to the problem.

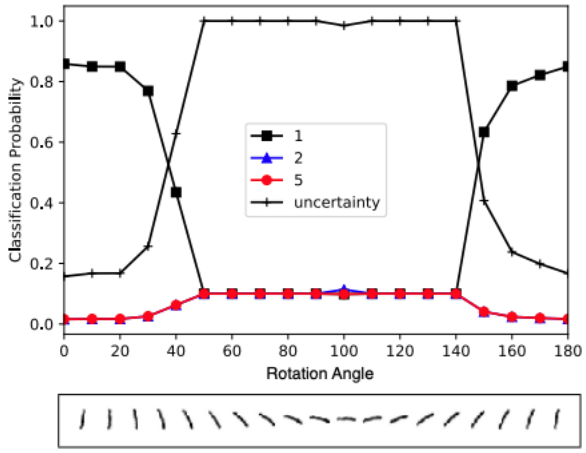


Figure 1.4: Classification probabilities for three labels 1, 2 and 5 alongside the uncertainty estimate for the rotated MNIST digit 1 using the proposed method of Evidential Deep Learning. Image taken from [38].

1.2 Dissertation Overview

This dissertation is structured to provide a logical and clear discussion around the problem of uncertainty estimation in Deep Learning Models and its application to object detection. **Chapter 2** reviews the literature around uncertainty estimation. What is uncertainty and how is it measured and providing an overview of current methods available to estimate uncertainty in Deep Learning. Limitations of existing methods are discussed and the need for a real time method of uncertainty estimation is motivated.

Chapter 3 introduces the main idea of the work. Subjective Logic and Evidential Deep Learning for uncertainty estimation. This chapter provides theoretical justification for this method before showing the reader how to implement into existing neural network architectures.

Chapter 4 discusses the motivating problem, object detection. This chapter is meant to be a self contained introduction and overview of the object detection problem and current state of the art methods.

Finally **Chapter 5** ties everything together. Linking the evidential framework in **Chapter 3** to object detection systems from **Chapter 4**. Presenting the result of this work - an uncertainty aware object detector. This chapter contains the framework used in my implementation and discusses the experimental results.

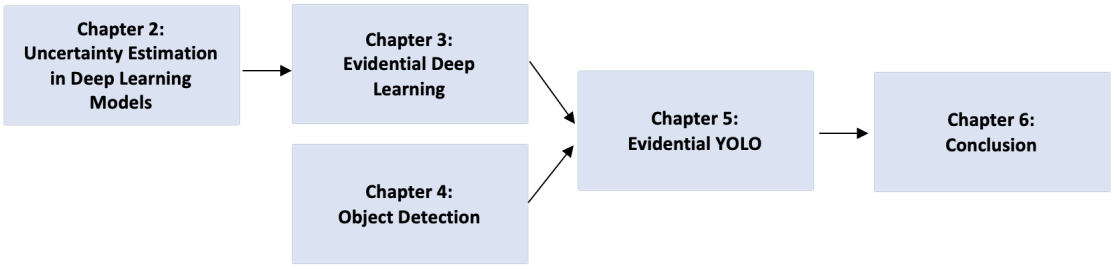


Figure 1.5: Chapter dependencies: Chapter 2 motivates the problem of uncertainty estimation in deep learning. Chapter 3 contains the proposed method of using an evidential framework motivated by subjective logic. Chapter 4 is a standalone chapter on Object Detection. Chapter 5 fuses Evidential Deep Learning into object detection systems and discusses the results.

Chapter 2

Uncertainty Estimation in Deep Learning Models

2.1 Formalizing Uncertainty

Assuming a distribution $p(\mathbf{x}, y)$ over features $\mathbf{x}$ and labels y then given an input vector $\mathbf{x}^*$ we can identify the predictive uncertainty for a classification model by $p(y^*|\mathbf{x}^*, \mathcal{D})$ where $\mathcal{D}$ denotes the training set $\{(\mathbf{x}_n, y_n)\}_{n=1}^N \sim p(\mathbf{x}, y)$ of the model. Making use of the Bayesian framework we can write this distribution explicitly by marginalizing over the model parameters θ as follows

$$p(y^*|\mathbf{x}^*, \mathcal{D}) = \int \underbrace{p(y^*|\mathbf{x}^*, \theta)}_{\substack{\text{Aleatoric} \\ \text{(Data)}}} \underbrace{p(\theta|\mathcal{D})}_{\substack{\text{Epistemic} \\ \text{(Model)}}} d\theta \tag{2.1}$$

This decomposition exposes two uncertainty components within the integral. The epistemic uncertainty or model uncertainty captures the ignorance in the model used to explain the observed data . Epistemic uncertainty includes both uncertainty in the model parameters and the model structure itself can be modelled using the posterior distribution over the model parameters given the training data $p(\theta|\mathcal{D})$. Epistemic uncertainty, can be seen as a reducible uncertainty since given enough data we can explain away the ignorance about the models parameters.

The aleatoric uncertainty or data uncertainty captures noise inherent in observations. This could be measurement error causing noisy labels in the data. We identify the aleatoric uncertainty via the posterior distribution over class labels given the model parameters and input vector by $p(y^*|\mathbf{x}^*, \theta)$. Unlike epistemic uncertainty, aleatoric uncetainty is not reducible and can not be explained away with more data. Thus regardless of the amount of data observed, a noisy example is best handled by a high entropy prediction [39]. Aleatoric uncertainty can split into two cases: homoscedastic or heteroscedsatic. In the homoscedastic scenario uncertainty is assumed constant whereas hetereoscedastic uncertainty is dependent on the input. Hetereoscedastic is clearly a more realistic assumption since some examples will be noiser than others.

Additionally one can model the distributional uncertainty arising from the disparity between the distribution of the test input $\mathbf{x}^*$ and the training data $\mathcal{D}$ the model was trained on. As described in [30], distributional uncertainty is usually captured as part of the epistemic uncertainty. However by considering it a separate source of uncertainty it can be modelled also. Introducing the distribution $p(\mu|\mathbf{x}^*, \theta)$ where μ represents a point estimate of the categorical distribution over class labels equation (2.1) can be modified to

$$p(y^*|\mathbf{x}^*, \mathcal{D}) = \iint p(y^*|\boldsymbol{\mu}) p(\mu|\mathbf{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\mu} d\boldsymbol{\theta} \tag{2.2}$$

with $p(y^*|\boldsymbol{\mu})$ the aleatoric (data) uncertainty, $p(\mu|\mathbf{x}^*, \boldsymbol{\theta})$ the distributional uncertainty and $p(\boldsymbol{\theta}|\mathcal{D})$ the epistemic (model) uncertainty. Gales and Malinin use this decomposition to argue that one can view this as

a hierarchical model in which all three uncertainties are linked and are dependent. The epistemic (model) uncertainty propagates into estimates of the distributional uncertainty which in turn propagates forward into estimates of the aleatoric (data) uncertainty. Thus model uncertainty has an affect on both data and distributional uncertainty and distributional uncertainty directly affects the estimates of data uncertainty.

In the setting of object detection, modelling epistemic uncertainty indicates the uncertainty in prediction for a particular observation while measuring the aleatoric uncertainty captures the noise in the observation. If we detect an abnormality from the training set we expect to receive a high value of epistemic uncertainty and conversely if we are detecting an object that is distant in the scene we would expect a high aleatoric uncertainty [9]. Clearly being able to capture both uncertainty measures is essential if deep technologies are to be readily adopted. Since epistemic uncertainty allows us to measure the limitations of the model whilst aleatoric can alert us to limitations or noise in a models sensory system.

The rest of this chapter looks at current methods for estimating uncertainty, their limitations and a motivation for the use of evidential deep learning for uncertainty estimation which will be the main focus of this work.

2.2 Methods of Uncertainty Estimation

Unfortunately the above integrals are usually intractable and analytically unobtainable. Hence we rely on methods for approximating and focus on two general methods to do so: sampling based and non-sampling based. Sampling based methods do as they say, requiring multiple samples of predictions for a given input and estimating the uncertainty based on the variance of these samples. Sampling free methods whereas are required to estimate the uncertainty using a single pass when making a prediction thus in theory requiring much less compute. Sampling free methods however are much less studied in the literature and used predominantly for estimating aleatoric uncertainty [43].

2.2.1 Bayesian Neural Networks and Variational Inference

Early work for uncertainty estimation in neural networks came with Bayesian Neural Network (BNNs) during the 90s with both Mackay's [29] and Neal's [34] PhD theses covering these models. The structure of a BNN enables one to capture uncertainty by replacing the deterministic weights of a standard neural network instead with a prior distribution over the same parameters. Since we are dealing with neural networks I will amend the notation for parameters with weights $\mathbf{W}$ and split a training set $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$ into inputs $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and outputs $\mathbf{Y} = \{y_1, \ldots, y_N\}$. Instead of optimizing the networks weights directly, one performs Bayesian inference to compute the posterior over weights given data $p(\mathbf{W}|\mathbf{X}, \mathbf{Y})$. Attaining this posterior encapsulates information about the set of possible parameters given the data. This is simple to describe in theory however usually unobtainable in practice due to the intractability of the models. Decomposing the posterior using Bayes',

$$p(\mathbf{W}|\mathbf{X}, \mathbf{Y}) = \frac{p(\mathbf{Y}|\mathbf{X}, \mathbf{W})p(\mathbf{W})}{p(\mathbf{Y}|\mathbf{X})} \tag{2.3}$$

computing the normaliser $p(\mathbf{Y}|\mathbf{X}) = \int p(\mathbf{Y}|\mathbf{X}, \mathbf{W})p(\mathbf{W})d\mathbf{W}$ requires explicit integration which can also be referred to as marginalising the likelihood over $\mathbf{W}$ [10]; which in most interesting cases cannot be done and requires approximation methods. These approximation methods fall under the realm of variational inference, where the idea for handling an intractable posterior $p(\mathbf{W}|\mathbf{X}, \mathbf{Y})$ is to approximate using a simpler distribution $q_\theta(\mathbf{W})$ parameterised by θ. The task of marginalising over the weights in the true posterior is replaced with an optimization task to optimize the parameters θ of the simpler distribution, such that the approximating distribution is as close to the true posterior as possible.

The Kullback-Leibler divergence exists as an information-theoretic measure of the closeness between two distributions and is minimized when the two are equal. Observering the KL between the variational

approximation the true posterior

$$\mathbf{KL}(q_\theta(\mathbf{W})||p(\mathbf{W}|\mathbf{X},\mathbf{Y})) = \int q_\theta(\mathbf{W}) \log \frac{q_\theta(\mathbf{W})}{p(\mathbf{W}|\mathbf{X},\mathbf{Y})} d\mathbf{W} \tag{2.4}$$

the minimum is found when $q_\theta(\mathbf{W}) \approx p(\mathbf{W}|\mathbf{X},\mathbf{Y})$. This minimization objective of the KL divergence is usually intractable however decomposing the terms,

$$\begin{aligned}
\mathbf{KL}(q_\theta(\mathbf{W})||p(\mathbf{W}|\mathbf{X},\mathbf{Y})) &= \int q_\theta(\mathbf{W}) \log q_\theta(\mathbf{W}) d\mathbf{W} - \int q_\theta(\mathbf{W}) \log p(\mathbf{W}|\mathbf{X},\mathbf{Y}) d\mathbf{W} \\
&= \int q_\theta(\mathbf{W}) \log q_\theta(\mathbf{W}) d\mathbf{W} - \int q_\theta(\mathbf{W}) \log \frac{p(\mathbf{W},\mathbf{Y}|\mathbf{X})}{p(\mathbf{Y}|\mathbf{X})} d\mathbf{W} \\
&= \mathbb{E}_{q_\theta(\mathbf{W})}\big[\log q_\theta(\mathbf{W})\big] - \mathbb{E}_{q_\theta(\mathbf{W})}\big[\log p(\mathbf{W},\mathbf{Y}|\mathbf{X})\big] + \log p(\mathbf{Y}|\mathbf{X})
\end{aligned}$$

the first two terms correspond to expectations with respect to the variational distribution. While the final term displays a dependency to compute the evidence $p(\mathbf{Y}|\mathbf{X})$. Considering the log evidence term we can derive an equivalent and most importantly a tractable optimization objective,

$$\begin{aligned}
\log p(\mathbf{Y}|\mathbf{X}) &= \log \int p(\mathbf{W},\mathbf{Y}|\mathbf{X}) d\mathbf{W} \\
&= \log \int q_\theta(\mathbf{W}) \frac{p(\mathbf{W},\mathbf{Y}|\mathbf{X})}{q_\theta(\mathbf{W})} d\mathbf{W} \\
&\geq \int q_\theta(\mathbf{W}) \log \frac{p(\mathbf{W},\mathbf{Y}|\mathbf{X})}{q_\theta(\mathbf{W})} d\mathbf{W} \\
&= \int q_\theta(\mathbf{W}) \log \frac{p(\mathbf{Y}|\mathbf{X},\mathbf{W})p(\mathbf{W})}{q_\theta(\mathbf{W})} d\mathbf{W} \\
&= \int q_\theta(\mathbf{W}) \log p(\mathbf{Y}|\mathbf{X},\mathbf{W}) d\mathbf{W} + \int q_\theta(\mathbf{W}) \log \frac{p(\mathbf{W})}{q_\theta(\mathbf{W})} d\mathbf{W} \\
&= \mathbb{E}_{q_\theta(\mathbf{W})}\big[\log p(\mathbf{Y}|\mathbf{X},\mathbf{W})\big] - \mathbf{KL}(q_\theta(\mathbf{W})||p(\mathbf{W})) := \text{ELBO}
\end{aligned}$$

which is referred to as the evidence lower bound and more commonly just the ELBO. In Gal's PhD thesis [10], he makes the observation that in this formulation the first term, the expected log likelihood under q encourages a good data fit whilst the second term, the KL divergence encourages the approximating posterior to be as close to the prior as possible and in effect acts as a regulariser by penalizing complexity. Rewriting the above,

$$\begin{aligned}
\text{ELBO} &= \int q_\theta(\mathbf{W}) \log \frac{p(\mathbf{W},\mathbf{Y}|\mathbf{X})}{q_\theta(\mathbf{W})} d\mathbf{W} \\
&= \int q_\theta(\mathbf{W}) \log \frac{p(\mathbf{W}|\mathbf{X},\mathbf{Y})p(\mathbf{Y}|\mathbf{X})}{q_\theta(\mathbf{W})} d\mathbf{W} \\
&= \int q_\theta(\mathbf{W}) \log \frac{p(\mathbf{W}|\mathbf{X},\mathbf{Y})}{q_\theta(\mathbf{W})} d\mathbf{W} + \log p(\mathbf{Y}|\mathbf{X}) \\
&= -\mathbf{KL}(q_\theta(\mathbf{W})||p(\mathbf{W}|\mathbf{X},\mathbf{Y})) + \log p(\mathbf{Y}|\mathbf{X})
\end{aligned}$$

and so the ELBO can be rewritten as minus the KL divergence between the true posterior and the variational approximation plus the addition of the log evidence which is constant with respect to θ. Hence

maximizing the evidence lower bound is equivalent to minimizing the KL divergence [4]. This general procedure is known as variational inference and replaces marginalization using integrals with an equivalent optimization procedure. This technique improves tractability for larger complex models however scales poorly with data.

2.2.2 Monte-Carlo Dropout

Monte-Carlo Dropout is a practical approach to performing approximate inference in deep learning models introduced in work by Gal and Ghahramani [11]. They observed that modifying networks by inserting dropout layers before every weight layer and including dropout at test time introduces stochasticity which allows effective sampling from the approximate posterior with each forward pass. This procedure of computing stochastic forward passes is known as Monte Carlo Dropout.

Gal showed formally that training using dropout layers and with L2 normalization is equivalent to maximizing the ELBO. That is finding a simpler distribution, from a tractable family of distributions such that it minimizes the KL divergence with the true model posterior $p(\mathbf{W}|\mathbf{X}, \mathbf{Y})$. Denoting the output of a Bayesian Neural Network by $f(\mathbf{x}, \mathbf{W})$ and the likelihood $p(\mathbf{y}|f(\mathbf{x}, \mathbf{W}))$ it assumed in the case of regression that the likelihood is modelled by a Gaussian with mean given by the model output and small variance σ^2: $p(\mathbf{y}|f(\mathbf{x}, \mathbf{W})) = \mathcal{N}(f(\mathbf{x}, \mathbf{W}), \sigma^2)$. In the case of classification, the model output is squashed using a softmax function and the resulting vector is used to sample from: $p(\mathbf{y}|f(\mathbf{x}, \mathbf{W})) = \text{Softmax}(f(\mathbf{x}, \mathbf{W}))$. Furthermore, modelling the approximating distribution by a mixture of two Gaussians, each with small variance σ^2 and with one of the means fixed at 0, the objective function to minimize becomes:

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N} \log p(\mathbf{y}_i|f(\mathbf{x}_i, \widehat{\mathbf{W}}_i)) + \frac{1 - p_{\text{dropout}}}{2N}\|\theta\|^2 \tag{2.5}$$

for N data points and where $\widehat{\mathbf{W}}_i \sim q_\theta(\mathbf{W})$ are the drawn model weights after applying dropout with probability p_{dropout}.

Kendall and Gal [21] showed the expected value of the variational distribution can be approximated using Monte Carlo, sampling T sets of weights as follows:

$$E(\mathbf{y}) \approx \frac{1}{T}\sum_{t=1}^{T} f(\mathbf{x}, \widehat{\mathbf{W}}_t)) \tag{2.6}$$

hence the process of averaging T forward passes from a network with dropout enabled is called Monte Carlo Dropout.

In the regression setting, the epistemic uncertainty is captured by the predictive variance which is approximated by:

$$\mathbf{Var}(\mathbf{y}) \approx \sigma^2 + \frac{1}{T}\sum_{t=1}^{T} f(\mathbf{x}, \widehat{\mathbf{W}}_t))^T f(\mathbf{x}, \widehat{\mathbf{W}}_t)) - E(\mathbf{y})^T E(\mathbf{y}) \tag{2.7}$$

Decomposing the above, the first term σ^2 represents the constant noise in the data, the homoscedatic aleatoric uncertainty while the second part represents the model uncertainty in its predictions.

For classification, the predictive distribution can be approximated using

$$p(y = c|\mathbf{x}, \mathcal{D}) \approx \frac{1}{T}\sum_{t=1}^{T} \text{Softmax}(f(\mathbf{x}, \widehat{\mathbf{W}}_t)) \tag{2.8}$$

however for classification one should consider methods other than predictive variance as the uncertainty measure such as predictive entropy and mutual information [10]. The output model in classification is a conditional distribution over class labels, predicitve entropy $\mathbb{H}(\mathbf{p}) = -\sum_c p_c \log p_c$ provides a measure of total uncertainty however does not distinguish between model and data uncertainty. Considering the

information gain between model parameters and data can help distinguish between epistemic and aleatoric examples [39]. Given a training set $\mathcal{D}_{\text{train}}$ the information gain in model parameters from receiving the label y for a new data point $\mathbf{x}$ is given by the mutual information:

$$\mathbb{I}[y, \mathbf{W}|\mathbf{x}, \mathcal{D}_{\text{train}}] := \mathbb{H}[p(y|\mathbf{x}, \mathcal{D}_{\text{train}})] - \mathbb{E}_{p(\mathbf{W}|\mathcal{D}_{\text{train}})}\big[\mathbb{H}[p(y|\mathbf{x}, \mathbf{W})]\big] \tag{2.9}$$

Mutual information thus provides a measure of epistemic uncertainty. Since learning the label for data the model is uncertain about would result in greater information gain than for data which the model already explains well.

In order to estimate the aleatoric uncertainty (homoscedastic) in classification, Kendall and Gal modify the model output $f(\mathbf{x}, \mathbf{W})$ by placing a Gaussian distribution on network output such that $\hat{\mathbf{y}} = f(\mathbf{x}, \mathbf{W}) + \mathcal{N}(0, \sigma^2)$, where σ^2 is a learned diagonal covariance. This vector is then squashed as before using a Softmax operator. Full details of heteroscedastic case for aleatoric uncertainty in regression and classification have been omitted and can be found in the referenced paper.

2.2.3 Deep Ensemble

As an alternative to the above work by Gal and Ghahramani [11], a non-Bayesian alternative to uncertainty estimation is considered using ensembles of deep neural networks [24]. The authors note that in practice Bayesian Neural Networks are often harder to implement and computationally more intensive to train than deterministic networks. Hence they seek a more general solution but which can match Monte Carlo Dropout in terms of performance for predictive uncertainty estimation. In addition they aim to match the simplicity of Monte Carlo Dropout which can easily be implemented to existing models by adding dropout layers into the model. Interpreting Dropout [40] as an ensemble method; since predictions are averaged over an ensemble of multiple networks with shared parameters. Coupled with the observation ensemble methods are known to improve performance the authors are motivated to explore ensemble methods for estimating uncertainty. Their work aims to provide a simple and scalable method for estimating uncertainty using the notion of proper scoring methods, adversarial training and the use of ensembles.

Formally the training procedure is described as following. For a given training set $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^N$ of N data points. Creating an ensemble of M neural networks with parameters $\{\theta_m\}_{m=1}^M$ such that each θ parameterises a distribution over outputs $p_\theta(y|\mathbf{x})$. Initializing the parameters randomly each network in the ensemble is trained in parallel on data batches drawn at random from the training set $\mathcal{D}$. As per the original paper, for a given epoch and for a particular network, a data point n_m is sampled at random and an adversarial example is generated using $\mathbf{x}'_{n_m} = \mathbf{x}_{n_m} + \epsilon \cdot \text{sign}(\nabla_{\mathbf{x}_{n_m}} \ell(\theta_m, \mathbf{x}_{n_m}, y_{n_m}))$. The objective is therefore given by:

$$\text{Minimize } \ell(\theta_m, \mathbf{x}_{n_m}, y_{n_m})) + \ell(\theta_m, \mathbf{x}'_{n_m}, y_{n_m})) \text{ w.r.t. } \theta_m \tag{2.10}$$

where $\ell(\theta, \mathbf{x}, y)$ is a "proper scoring rule". The paper advises the use of scoring rules as a measure of predictive uncertainty. Whereby a scoring function $S(p_\theta, (y, x))$ assigns a numerical score to $p_\theta(y|x) \equiv p(y|x, \theta)$ relative to $y|x \sim q(y|x)$ with $q(x, y)$ the true distribution on such pairs; naturally higher scores denoting better calibrated predictions. With this notion, the expected scoring rule is $S(p_\theta, q) = \int q(y, x) S(p_\theta, (y, x)) dy dx$ and a proper scoring rule is defined to be one such that $S(p_\theta, q) \leq S(q, q)$ with equality only when $p \equiv q$. This formalism then allows the training of networks is done so to encourage calibration of uncertainty by minimizing $-S(p_\theta, q)$.

An interesting addition in this framework is the use of adversarial examples. Adversarial examples, originally proposed by Szegedy [42] can be found by applying ever so slight perturbations unnoticeable to the human eye but such that it can be possible to change the networks prediction. The famous example displayed in [15] by Goodfellow shows a small perturbation on the image of a Panda results in the network predicting instead a Gibbon with extremely high confidence; yet the original image and the perturbed image are completely indistinguishable. Goodfellow proposes the fast gradient sign method for generating adversarial examples adding a small perturbing along the direction that is most likely to increase to loss. Thus in the training of deep ensembles, by training on adversarial examples as well as the original training samples allows the trained model learns to be robust in the face of adversaries.

Predictions are made by combining each prediction within the ensemble uniformly such that $p(y|\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} p_{\theta_m}(y|\mathbf{x}, \theta_m)$. In the classification setting this is just the average of predictions whereas in the regression setting it is a mixture of Gaussians. The paper suggests for computational ease to assume the ensemble prediction follows a Gaussian with mean and variance equal to those of the mixture itself. That is $\frac{1}{M}\sum_{m=1}^{M}\mathcal{N}(\mu_{\theta_m}, \sigma^2_{\theta_m}) = \mathcal{N}(\mu_*(\mathbf{x}), \sigma_*(\mathbf{x})^2)$ where

$$\begin{aligned}\mu_*(\mathbf{x}) &= \frac{1}{M}\sum_{m=1}^{M}\mu_{\theta_m}(\mathbf{x}) \\ \sigma^2_*(\mathbf{x}) &= \frac{1}{M}\sum_{m=1}^{M}\left(\sigma^2_{\theta_m}(\mathbf{x}) + \mu^2_{\theta_m}(\mathbf{x})\right) - \mu^2_*(\mathbf{x})\end{aligned} \tag{2.11}$$

Thus training an ensemble of deep neural networks one can obtain a measure of uncertainty in a non-Bayesian manner by averaging the predictions and calculating the variance from a single input in multiple networks. Noting however an ensemble of multiple networks inherently requires multiple sets of weights need to be stored. This memory overhead may mean limitations in deployment. However utilizing methods such as Distillation [18] it is possible to train smaller networks to learn to mimic the performance of an ensemble of larger cumbersome models.

2.3 Motivation for Real Time

A limitation of both methods described is they require multiple evaluations of each input to estimate uncertainty. In Monte-Carlo Dropout multiple stochastic forward passes through a Bayesian network are required while Deep Ensemble requires evaluating an ensemble of multiple networks to ascertain a good estimate of the uncertainty. Multiple evaluations per detection is impractical for most safety critical applications and hence motivates the need for a real time solution. In the following chapter Subjective Logic and Evidential Deep Learning are introduced. Evidential deep learning utilises subjective logic to modify the output layer of deterministic neural network to provide uncertainty estimation for free in a single forward pass. Thus with minimal modification to existing architectures, real time uncertainty estimation can be achieved without the need for sampling or the large memory overhead of an ensemble.

Chapter 3

Evidential Deep Learning

In classification problems, traditionally the sum of probabilities across all classes sum to one. This forces the estimate to fall in the set of pre-defined classes. Forcing probability assignment to each element in the state space restricts any ability to express uncertainty and awareness of when the model is simply guessing. In typical deep learning models, neural networks generate a distribution over class labels by squashing the network outputs through a softmax layer. This removes any notion of ignorance and capitulates the ability for a network to say "I do not know". Evidential Deep Learning [38] takes inspiration from Subjective Logic and generalises this by proposing that uncertainty plus the class probabilities sum to one instead. This uncertainty measure now models the fact the not all classes are right. This chapter introduces Subjective Logic and provides a practical guide to implementing this framework using Evidential Deep Learning.

3.1 Intoduction to Subjective Logic

Much of this section is influenced by Josang [20] and the purpose here is to provide a self-contained introduction to subjective logic. At a high-level subjective logic extends traditional probabilistic logic to express uncertainty about the probabilities dealt with. It enhances standard logic and probabilistic logic such that degrees of truth can be assigned; thus allowing one to reason in the presence of uncertain or incomplete evidence. Subjective Logic is motivated by the fact probability values are assigned between 0 and 1 based on finite evidence. With finite evidence no proposition can be determined with 100 percent accuracy. Thus modelling uncertainty is paramount for the subjective world in which we reason in. The problem with our current framework also lies with the additivity principle of probability which requires the sum of probabilities for a state space of mutually exclusive disjoint elements to sum to 1. This problem was initially tackled by Dempster who proposed the notion of upper and lower probabilities and was extend by Shafer who proposed the model of belief functions. Dempster-Shafer theory generalizes the Bayesian model to a subjective setting by abandoning the additivity principle and instead assigning belief mass among the subsets of the state space. This framework now allows one to express ignorance in a setting of lacking evidence by assigning belief mass throughout the space. Subjective Logic formalizes the Dempster-Shafer theory, allowing the inclusion of base rates and a correspondence to the Dirichlet distribution.

Subjective logic argues with subjective opinions, these opinions represent the ignorance in our estimations and contain the uncertainty about the probability values. In effect modelling second-order probabilities. Formally a subjective opinion reasons over a state space and is a composite function of the belief masses, uncertainty in the estimates and prior base rates. Given a subjective opinion there exists a correspondence under a particular mapping to either a Beta or a Dirichlet distribution that we will derive. Thus by constructing an opinion given evidence we are building a framework to derive a probability density parameterized by evidence. This will allows us to draw from a distribution of possible probability assignments instead of a single point estimate.

To formalize the above. Given a state space X such that $|X| = K$ denote the power set of X, 2^X

containing every subset of X. In subjective logic belief mass is assigned and distributed throughout subsets of the reduced power set $\mathcal{R}(X) = 2^X \backslash \{\emptyset, X\}$. Subjective logic allows for belief mass distribution over any proper subset of the state space X, either restricted to elements of X or to elements of $\mathcal{R}(X)$.

The distribution of belief mass is driven by the amount of evidence observed. Observing infinite evidence corresponds to an additive belief mass distribution where the total sum of belief mass is one. In the case of finite evidence, the belief mass distribution is said to be sub-additive. That is the sum of belief mass is less than one. To account for sub-additivity, the complement to one is made up by an uncertainty mass. Clearly in the additive case, infinite evidence means there can be no uncertainty.

Define the vector of belief masses $\mathbf{b}_X$ to be the distribution of belief mass over the elements of $\mathcal{R}(X)$ and the corresponding uncertainty mass u_X to be the uncertainty around the expected probability values. Then the inclusion of uncertainty mass for sub-additive belief mass describes a $(K+1)$ dimensional simplex:

$$\left\{ u_X + \sum_{x \in \mathcal{R}(X)} \mathbf{b}_X(x) = 1 \middle| 0 \leq u_X, \mathbf{b}_X(x_1), \ldots, \mathbf{b}_X(x_K) \leq 1 \right\}$$

Subjective opinions express belief and uncertainty about propositions and can be split into three broad types: binomial opinions, multinomial opinions and hyper opinions. For a binary state space the opinion is binomial. For larger state spaces, $|X| > 2$ opinions on singleton elements $x \in X$ are multinomial opinions while opinions on $x \in \mathcal{X}$ are called hyper-opinions. The following focuses on multinomial opinions since in classifications the distribution is over a set of mutually exclusive elements in the state space.

Each multinomial opinion contains a belief vector $\mathbf{b}_X$ of size K, a base rate vector $\mathbf{a}_X$ of size K and the uncertainty scalar u_X. The base rate $\mathbf{a}_X$ represents a distribution over X such that $\mathbf{a}_X(x_k) \in [0, 1]$ and $\sum_k \mathbf{a}_X(x_k) = 1$ and is the tie between the evidential and the probabilistic view [2]. Base rates or priors are central to Bayesian statistics and define how unknown mass should be distributed. In the case of complete ignorance equal distribution of mass among the K classes is desired. That is $\mathbf{a}_X = \frac{1}{K}\mathbf{1}$. Thus each multinomial opinion is made up of $(2K+1)$ parameters. The probability projection for a multinomial opinion can be expressed by

$$P_X(x_k) = \mathbf{b}_X(x_k) + \mathbf{a}_X(x_k) u_X, \qquad \forall x_k \in X \tag{3.1}$$

The probability projection of multinomial opinion thus provides an estimate to the ground truth of probability p_k. The following section provides an overview of the Dirichlet distribution and its link to Subjective Logic.

3.2 The Dirichlet Distribution

The Dirichlet distribution can be viewed as a distribution over distributions and is a suitable prior for classification models since it is the conjugate prior for both the categorical and multinomial distributions [44]. The Dirichlet distribution $D(\mathbf{p}|\boldsymbol{\alpha})$ is parameterized by positive scalars α_k for $k = 1, \ldots, K$ with $K \geq 2$ and defined as

$$D(\mathbf{p}|\boldsymbol{\alpha}) = \begin{cases} \frac{1}{B(\boldsymbol{\alpha})} \prod_{k=1}^{K} p_k^{\alpha_k - 1}, & \text{for } \mathbf{p} \in \mathcal{S}_K, \\ 0 & \text{otherwise,} \end{cases} \tag{3.2}$$

where $\mathcal{S}_K$ is the K-dimensional unit simplex,

$$\mathcal{S}_K = \left\{ \mathbf{p} \middle| \sum_{k=1}^{K} p_k = 1 \text{ and } 0 \leq p_1, \ldots, p_K \leq 1 \right\}$$

with $B(\boldsymbol{\alpha})$ the K-dimensional multinomial beta function. Parameters $\boldsymbol{\alpha}$ represent our a priori knowledge, setting $\alpha_k = 1$ for each $k = 1, \ldots, K$ represents a uniform distribution in the K-dimensional simplex i.e. a non-informative prior.

In low dimensions it is possible to visualize the Dirichlet distribution. For $K = 3$ the distribution can be viewed on the simplex equilateral triangle.

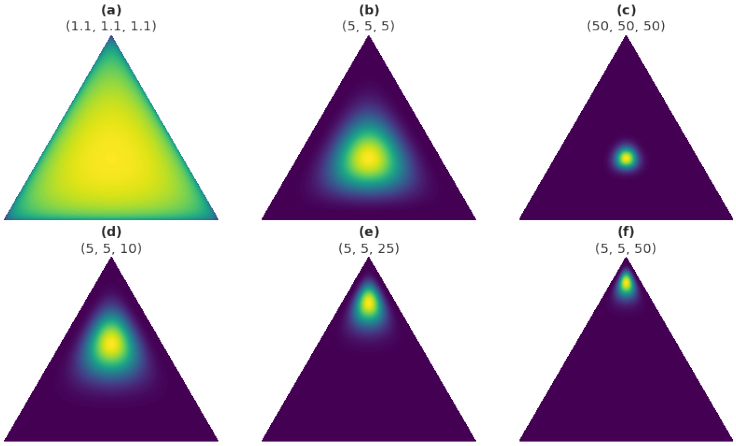


Figure 3.1: Dirichlet distributions on the equilateral triangle ($K = 3$) for varying $\boldsymbol{\alpha}$.

Visually it is clear that the prior parameters and their magnitudes determine the overall shape and peak of the distribution. Based on each simplex we can describe the uncertainty for each predicted distribution induced by $\boldsymbol{\alpha}$. In the top row (a-c) each component α_k has the same magnitude, however the precision increases left to right. In the first case (a), the magnitudes are set near 1 which represents a uniform prior for the prediction. This behaviour represents an out-of-distribution case where the distribution is flat across the simplex indicating epistemic uncertainty. In (b) and (c) the precision increases and the peak concentrates towards the middle of the simplex. This prior represents high data uncertainty from to noise in the data being predicted on. This is not an necessarily out-of-distribution and the predicted distribution is a flat categorical distribution [30]. A highly confident prediction should result in a sharp distribution in the corner of the simplex corresponding to that class. Finally, in the bottom row (d-f) increasing the final parameter component, holding all others constant the distribution gets increasingly sharper and shifts away and into the corner of the last class. Indicating an increasing confident predictive distribution.

For a given distribution defining the Dirichlet strength, or precision to be the sum:

$$\alpha_0 = \sum_{k=1}^{K} \alpha_k \tag{3.3}$$

then the expected probability value for each component can be found by examining the mean of the Dirichlet:

$$\begin{aligned} \mu_k &= \mathbb{E}[D(\mathbf{p}|\boldsymbol{\alpha})]_k \\ &= \frac{\alpha_k}{\alpha_0} \end{aligned} \tag{3.4}$$

3.3 Equivalence of Subjective Opinions and the Dirichlet PDF

In this section we show the equivalence between a Dirichlet pdf and subjective opinions on the same state space and define a bijective mapping between the two. To clarify notion, first let $X = \{x_1, \ldots, x_K\}$ be the state space. Define the tuple $\omega_X = (b_1, \ldots, b_K, u, \mathbf{a})$ to be a multinomial subjective opinion containing the belief masses, the uncertainty and a vector of base rates about X and define $D(\mathbf{p}|\boldsymbol{\alpha})$ a Dirichlet pdf over X.

The parameters of the Dirichlet $\boldsymbol{\alpha}$ can be viewed as the total evidence for each singleton and contain a priori knowledge as well observed evidence. Defining a non-informative prior weight constant W. This weight is then distributed over all possible outcomes as a function of the base rates. Therefore given prior probabilities $\mathbf{a}$, collected evidence $\mathbf{e}$ and non-informative prior weight constant the total evidence can be expressed as

$$\boldsymbol{\alpha} = \mathbf{e} + W\mathbf{a} \tag{3.5}$$

where $e_k \geq 0, \alpha_k \in [0,1]$ for $k = 1, \ldots, K$ and $\sum_k a_k = 1, W \geq 2$. Therefore the Dirichlet pdf over K singleton states can be paramterized by the base rate, collected evidence and a non-informative weight constant. Observing equations (3.3)+(3.4), we can write the expected probability value for any singleton as

$$\mu_k = \frac{e_k + Wa_k}{\sum_k (e_k + Wa_k)} = \frac{e_k + Wa_k}{W + \sum_k e_k} \tag{3.6}$$

Hence a Dirichlet pdf can directly map observed evidence into probabilities over the state space. This also shows observational evidence and base rates can be combined to form subjective opinions. It is possible now to define a bijective mapping between the Dirichlet pdf and multinomial opinions over the same state space. Firstly it is required in expectation probabilities for all $x_k \in X$ are equal: i.e. with a slight abuse of notion,

$$\begin{aligned} P_X(x_k) &= \mathbb{E}[\omega_X(\mathbf{b}, u, \mathbf{a})]_k \\ &= \mathbb{E}[D(\mathbf{p}|\boldsymbol{\alpha})]_k \\ &= \mu_k \\ \iff b_k + ua_k &= \frac{e_k + Wa_k}{W + \sum_k e_k} \\ &= \frac{e_k}{W + \sum_k e_k} + \frac{Wa_k}{W + \sum_k e_k} \end{aligned}$$

Additionally we require that each belief mass b_k is an increasing function in observed evidence e_k and such that the uncertainty scalar u is a decreasing function in total observed evidence $\sum_k e_k$. That is,

$$\lim_{u \to 0} \sum_k b_k = 1 \quad \text{and} \quad \lim_{u \to 0} \sum_k e_k = \infty$$

Therefore we can formally establish the mapping between a multinomial opinion $\omega_X = (\mathbf{b}, u, \mathbf{a})$ and a Dirichlet pdf $D(\mathbf{p}|\mathbf{e}, \mathbf{a})$ over the state space X of size K, for each $x_k \in X$. Distinguishing between the cases of certainty and uncertainty,

$$b_k = \frac{e_k}{W + \sum_k e_k} \quad \text{and} \quad u = \frac{W}{W + \sum_k e_k} \text{ for } k = 1, \ldots, K$$

$$\iff \begin{cases} u + \sum_k b_k = 1 & \text{and} \quad \mathbf{e} = \frac{W}{u}\mathbf{b} & \text{if } u \neq 0, \\ \sum_k b_k = 1 & \text{and} \quad \sum_k e_k = \infty & \text{if } u = 0. \end{cases}$$

The certainty constraint requiring at least one of the observed evidence parameters e_k be infinite. This equivalence allows one to use evidential theory to quantify belief mass and uncertainty in a sound theoretical framework using Dirichlet pdfs. This framework will be applied to the context of neural networks where the output of our network constitutes evidence for mutually exclusive singletons which we use to construct the notion of uncertainty and construct Dirichlet priors to reason classification.

3.4 Practical Use

In all practical cases we will never be in a situation of infinite evidence nor complete certainty, thus we must consider the case of uncertainty. As above we can directly translate a subjective opinion to Dirichlet parameters using

$$\boldsymbol{\alpha} = \left\langle \frac{W}{u} b_1 + W a_1, \ldots, \frac{W}{u} b_K + W a_K \right\rangle$$

In the absence any evidence, by (3.5) the parameter vector reduces to $\boldsymbol{\alpha} = W\mathbf{a}$. Therefore the evidence for an opinion ω_X is given by

$$\mathbf{e} = \left\langle \frac{W}{u} b_1, \ldots, \frac{W}{u} b_K \right\rangle.$$

We focus on the case of a uniform a priori over the entire frame of cardinality K. This forces $W = K$ with base rates $a_k = 1/K$ which results in

$$\boldsymbol{\alpha} = \left\langle \frac{K}{u} b_1 + 1, \ldots, \frac{K}{u} b_K + 1 \right\rangle = \left\langle e_1 + 1, \ldots, e_K + 1 \right\rangle = \mathbf{e} + 1$$

as desired. Defining the Dirichlet strength, $S = \sum_k (e_k + 1)$ we see given evidence $e_k \geq 0$ for the kth singleton, the belief mass b_k and uncertainty u are computed by

$$b_k = \frac{e_k}{S} \text{ and } u = \frac{K}{S}.$$

Such that all $K + 1$ parameters are non-negative and are additive, i.e.,

$$u + \sum_{k=1}^{K} b_k = 1.$$

Under this construct uncertainty is inversely proportional to the total evidence generated and in the case of no evidence each individual belief mass is zero and the uncertainty is one. Subjective Logic provides a means to take direct observations and form subjective opinions. Given an opinion we have the framework to map to a Dirichlet pdf which we can calculate the expected probability for each singleton in the state space as the mean of the Dirichlet distribution:

$$\hat{p}_k = \frac{\alpha_k}{S}$$

alongside an value of our uncertainty based on the amount of evidence collected. With this framework we can modify standard neural networks to replace point estimates with density's over distributions that can quantify uncertainty.

Considering the output of a network to be an estimate for observed evidence $\mathbf{e}$, since there is no notion of negative valued evidence any negative logits must be removed. A negative output indicates no signal for that state and we should discard this as noise. Replacing the softmax function at the end of a network with a function such as ReLU will discard negative valued signals by returning zero and retain true positive signals. If all logit outputs from the network are negative, then then output after applying ReLU will be a vector of zeros. This zero vector be interpreted as zero total evidence and hence total uncertainty for the classification task.

Formally, given an example $\mathbf{x}$, denote $f(x|\mathbf{W})$ the output of the network with learned weights $\mathbf{W}$. Then $\hat{\mathbf{e}}(x) = \text{ReLU}(f(x|\mathbf{W}))$ represents the output of a network passed through a ReLU function and is a proxy for the estimated evidence vector predicted by the neural network. This vector will parameterise a Dirichlet distribution under $\boldsymbol{\alpha} = \hat{\mathbf{e}}(x) + 1$. The Dirichlet $D(\mathbf{p}|\boldsymbol{\alpha})$ represents a prior distribution to a categorical distribution over K class labels. Given a parameterisation $\boldsymbol{\alpha} =< \alpha_1, \ldots, \alpha_K >$ the precision $\alpha_0 = \sum_j \alpha_j$ is given by the sum of components.

The posterior over class labels is given by the expected distribution under the Dirichlet prior:

$$\begin{aligned} p(y = k|\mathbf{x}, \mathbf{W}) &= \int p(y = k|\mathbf{p}) D(\mathbf{p}|\mathbf{x}, \mathbf{W}) d\mathbf{p} \\ &= \mathbb{E}_{D(\mathbf{p}|\mathbf{x},\mathbf{W})}[p(y = k|\mathbf{p})] \\ &= \frac{\alpha_k}{\sum_{j=1}^{K} \alpha_j} \end{aligned}$$

3.5 EDL Loss Function

Now the framework has been described, a suitable loss function is now required to train networks to learn to form opinions. To restrict what evidence is created, the network should be peanlized for assigning evidence not only the wrong class but also to multiple different classes. In the original Evidential Deep Learning paper by Sensoy et al. [38] three possible loss functions are proposed: type II max likelihood, cross-entropy and sum of squares loss. This work uses cross-entropy due to association with classification loss. In the final part of this chapter, I formally justify this chioice of cross entropy as a suitable loss function by considering the KL divergence between the model and unimodal Dirichlet. The expected cross-entropy loss, or Bayes risk is then stated as

$$\mathcal{L}^{CE}(\mathbf{W}) = \int \left[\sum_{j=1}^{K} -y_j \log(p_j) \right] D(\mathbf{p}|\boldsymbol{\alpha}) d\mathbf{p}$$

In addition to minimizing the expected cross entropy loss it is desirable in the context of subjective logic to minimize evidence generated for incorrect class label. Sensoy explains this by noting when the objective is to simply minimize loss, in the presence of class specific features the model will learn to generate evidence when these features are observed. For instance the roundness of the digit zero which also happens to appear in both an eight and a six. To counter this it is suggested to regularize the model by removing the evidence for these counter examples on the basis that total evidence should tend to zero when the model is uncertain about the true class. Note, a Dirichlet distribution parameterized using zero evidence equates to the uniform distribution. Thus we should regularize with a KL divergence between the model after removing evidence for the correct class and a uniform distribution.

Combining the expected cross-entropy loss with this regularization term gives the EDL or Evidential Deep Learning Loss $\mathcal{L}^{EDL}$ by summing over a training batch of N examples

$$\boxed{\mathcal{L}^{EDL}(\mathbf{W}) = \sum_{i=1}^{N} \mathcal{L}_i^{CE}(\mathbf{W}) + \lambda_t \cdot \mathbf{KL}(D(\mathbf{p}_i|\widetilde{\boldsymbol{\alpha}}_i)||D(\mathbf{p}_i|\mathbf{1}))}$$

where $\lambda_t \in [0, 1]$ represents an annealing coefficient, with t the current training epoch. In the KL regularization term, $\widetilde{\boldsymbol{\alpha}} = \mathbf{y} + (1 - \mathbf{y}) \odot \boldsymbol{\alpha}$ denotes modified parameters of the Dirichlet replacing the truth parameter with the value 1. Increasing this annealing term from 0 to 1 during the course of training thus encourages exploration at the beginning before converging immediately the uniform distribution.

The following two subsections derive analytic expressions for both the cross entropy loss and the KL regularization term.

3.5.1 Cross Entropy Loss

To derive an analytic expression for the expected cross-entropy loss. Swapping the order of intergral and the summation terms in $\mathcal{L}^{CE}(\mathbf{W})$ it is possible to write the expected loss as a weighted sum of Dirichlet

sufficient statistics [33] as follows

$$\begin{aligned}\mathcal{L}^{CE}(\mathbf{W}) &= \int \left[\sum_{j=1}^{K} -y_j \log(p_j)\right] D(\mathbf{p}|\boldsymbol{\alpha}) d\mathbf{p} \\ &= -\sum_{j=1}^{K} y_j \int \log(p_j) D(\mathbf{p}|\boldsymbol{\alpha}) d\mathbf{p} \\ &= -\sum_{j=1}^{K} y_j \mathbb{E}_{D(\mathbf{p}|\boldsymbol{\alpha})}\big[\log p_j\big]\end{aligned} \tag{3.7}$$

The first step is to evaluate the above expression is to explicitly write the expectation as the integral of the Dirichlet and the function $\log p_k$ over the simplex. Since $\sum_{j=1}^{K} p_j = 1$, there are $K-1$ degrees of freedom and therefore we need only integrate over the $(K-1)$th dimensional simplex $\mathcal{S}_{K-1}$. Examining the explicit Dirichlet within the integral since the Beta distribution is independent of the $\mathbf{p}$ this can be factored out of the integrand as follows.

$$\begin{aligned}\mathbb{E}_{D(\mathbf{p}|\boldsymbol{\alpha})}\big[\log p_k\big] &= \int_{\mathcal{S}_{K-1}} D(\mathbf{p}|\boldsymbol{\alpha}) \log p_k d\mathbf{p} \\ &= \int_{\mathcal{S}_{K-1}} \log p_k \frac{1}{B(\boldsymbol{\alpha})} \prod_{j=1}^{K} p_j^{\alpha_j - 1} d\mathbf{p} \\ &= \frac{1}{B(\boldsymbol{\alpha})} \int_{\mathcal{S}_{K-1}} \log p_k \prod_{j=1}^{K} p_j^{\alpha_j - 1} d\mathbf{p}\end{aligned}$$

Making two observations helps this to be solved analytically. Firstly consider for arbitrary x and y the derivative of y with respect to an exponent x,

$$\begin{aligned}\frac{d}{dx}\big(y^x\big) &= \frac{d}{dx}\big(\exp(x \log y)\big) \\ &= \log(y) \cdot \exp(x \log y) \\ &= y^x \log(y)\end{aligned}$$

Thus,

$$\frac{\partial}{\partial \alpha_k} p_k^{\alpha_k} = p_k^{\alpha_k} \log p_k$$

and hence,

$$\begin{aligned}\frac{\partial}{\partial \alpha_k} \prod_j p_j^{\alpha_j - 1} &= \Big(p_k^{-1} \prod_{j \neq k} p_j^{\alpha_j - 1}\Big) \cdot \Big(\frac{\partial}{\partial \alpha_k} p_k^{\alpha_k}\Big) \\ &= \Big(p_k^{-1} \prod_{j \neq k} p_j^{\alpha_j - 1}\Big) \cdot \Big(p_k^{\alpha_k} \log p_k\Big) \\ &= \log p_k \prod_j p_j^{\alpha_j - 1}\end{aligned}$$

which is precisely equal to the integrand we are trying to evaluate. Therefore,

$$\begin{aligned}
\mathbb{E}_{D(\mathbf{p}|\boldsymbol{\alpha})}\big[\log p_k\big] &= \frac{1}{B(\boldsymbol{\alpha})}\int_{\mathcal{S}_{K-1}} \frac{\partial}{\partial \alpha_k}\prod_j p_j^{\alpha_j - 1} d\mathbf{p} \\
&= \frac{1}{B(\boldsymbol{\alpha})}\frac{\partial}{\partial \alpha_k}\int_{\mathcal{S}_{K-1}} \prod_j p_j^{\alpha_j - 1} d\mathbf{p} \\
&= \frac{1}{B(\boldsymbol{\alpha})}\frac{\partial}{\partial \alpha_k}\int_{\mathcal{S}_{K-1}} B(\boldsymbol{\alpha}) D(\mathbf{p}|\boldsymbol{\alpha}) d\mathbf{p} \\
&= \frac{1}{B(\boldsymbol{\alpha})}\frac{\partial}{\partial \alpha_k}\Big(B(\boldsymbol{\alpha})\underbrace{\int_{\mathcal{S}_{K-1}} D(\mathbf{p}|\boldsymbol{\alpha}) d\mathbf{p}}_{=1}\Big) \\
&= \frac{1}{B(\boldsymbol{\alpha})}\frac{\partial}{\partial \alpha_k} B(\boldsymbol{\alpha}) \\
&= \frac{\partial}{\partial \alpha_k}\log\Big(B(\boldsymbol{\alpha})\Big) \\
&= \frac{\partial}{\partial \alpha_k}\log\frac{\prod_j \Gamma(\alpha_j)}{\Gamma(\alpha_0)} \\
&= \frac{\partial}{\partial \alpha_k}\log\prod_j \Gamma(\alpha_j) - \frac{\partial}{\partial \alpha_k}\log\Gamma(\alpha_0) \\
&= \frac{\partial}{\partial \alpha_k}\sum_j \log\Gamma(\alpha_j) - \frac{\partial}{\partial \alpha_k}\log\Gamma(\alpha_0) \\
&= \frac{\partial}{\partial \alpha_k}\log\Gamma(\alpha_k) - \frac{\partial}{\partial \alpha_k}\log\Gamma(\alpha_0) \\
&= \frac{\Gamma'(\alpha_k)}{\Gamma(\alpha_k)} - \frac{\Gamma'(\alpha_0)}{\Gamma(\alpha_0)} \\
&= \psi(\alpha_k) - \psi(\alpha_0)
\end{aligned}$$

where $\alpha_0 = \sum_j \alpha_j$ and $\psi(\cdot)$ is the digamma function. Therefore,

$$\begin{aligned}
\mathcal{L}^{CE}(\mathbf{W}) &= \int \left[\sum_{j=1}^{K} -y_j \log(p_j)\right] D(\mathbf{p}|\boldsymbol{\alpha}) d\mathbf{p} \\
&= \sum_{j=1}^{K} y_j\Big(\psi(\alpha_0) - \psi(\alpha_j)\Big)
\end{aligned} \tag{3.8}$$

3.5.2 KL Regularization

Denoting $\boldsymbol{\alpha} = f(x|\mathbf{W}) + 1$ the parameters of the model and $\mathbf{y}$ a one-hot encoding of the ground truth. Then $\widetilde{\boldsymbol{\alpha}} = \mathbf{y} + (1 - \mathbf{y}) \odot \boldsymbol{\alpha}$ denotes the parameters for of the Dirichlet replacing the truth parameter with the value 1. This is since the ground truth $\mathbf{y} = (0, .., 1, .., 0)$ places a unit mass in the index of the true class, while $1 - \mathbf{y} = (1, .., 0, .., 1)$ places unit mass in all incorrect indices and replaces the true class index with a zero. Now computing the element wise multiplication $(1 - \mathbf{y}) \odot \boldsymbol{\alpha} = (\alpha_1, .., 0, ..\alpha_K)$ and hence $\mathbf{y} + (1 - \mathbf{y}) \odot \boldsymbol{\alpha} = (\alpha_1, .., 1, ..\alpha_K)$. Hence optimizing on this quantity ignores the truth parameter α_k and instead focuses on all parameters relating to misleading evidence. Using the analytic form of the KL divergence between two Dirichlet distributions derived in appendix A.1 results in the desired regularization objective:

$$
\begin{aligned}
\mathbf{KL}(D(\mathbf{p}|\tilde{\boldsymbol{\alpha}})||D(\mathbf{p}|(1,\ldots,1)) &= \log\Gamma(\sum_{k=1}^{K}\tilde{\alpha}_k) - \log\Gamma(\sum_{k=1}^{K}1) \\
&\quad + \sum_{k=1}^{K}\log\Gamma(1) - \sum_{k=1}^{K}\log\Gamma(\tilde{\alpha}_k) \\
&\quad + \sum_{k=1}^{K}(\tilde{\alpha}_k - 1)\Big[\psi(\tilde{\alpha}_k) - \psi(\sum_{j=1}^{K}\tilde{\alpha}_j)\Big] \\
&= \log\Gamma(\sum_{k=1}^{K}\tilde{\alpha}_k) - \log\Gamma(K) - \log\prod_{k=1}^{K}\Gamma(\tilde{\alpha}_k) \\
&\quad + \sum_{k=1}^{K}(\tilde{\alpha}_k - 1)\Big[\psi(\tilde{\alpha}_k) - \psi(\sum_{j=1}^{K}\tilde{\alpha}_j)\Big] \\
&= \log\left(\frac{\Gamma(\sum_{k=1}^{K}\tilde{\alpha}_k)}{\Gamma(K)\prod_{k=1}^{K}\Gamma(\tilde{\alpha}_k)}\right) + \sum_{k=1}^{K}(\tilde{\alpha}_k - 1)\Big[\psi(\tilde{\alpha}_k) - \psi(\sum_{j=1}^{K}\tilde{\alpha}_j)\Big]
\end{aligned}
$$

One potential drawback of this method is we force a uniform distribution on all incorrect classifications. This is not necessarily the best approach for a robust classifier since it is likely some classes are more likely than others. Or rather, not all incorrect classes should be treated the same. For example, classifying a car as a truck should be more likely than as a tree. Regularizing incorrect classifications to a more robust prior distribution should lead to more robust classifications when mistakes are made.

3.6 Justifying Cross-Entropy Using Reverse KL

This section aims to justify the use for the cross-entropy loss which I took for granted in setting up the EDL loss function. Taking inspiration from the work of Malinin and Gales [30, 31] it can be shown calculating the reverse Kullback-Leibler divergence between the model and a target unimodal distribution results in exactly the same objective.

First I provide an overview of the KL divergence and justify why the reverse KL divergence is a suitable measure for the problem. KL divergence was introduced in chapter two as an information-theoretic measure for the closeness between two distributions. It was used to set up an objective to be minimized when approximating analytically intractable distributions with simpler approximating distributions. Kullback-Leibler divergence or simply KL divergence is an asymmetric measure of how two distributions p and q differ. Formally the KL divergence between p and q is computed by,

$$
\mathbf{KL}(p||q) = \sum_{x} p(x)\log\frac{p(x)}{q(x)} \tag{3.9}
$$

KL divergence can be seen as a weighted average of the difference between the distributions over $\mathcal{X}$ however it is important to notice $\mathbf{KL}(p||q) \neq \mathbf{KL}(q||p)$. Denoting the true distribution p and the approximating distribution q then $\mathbf{KL}(p||q)$ is the forward KL divergence and $\mathbf{KL}(q||p)$ is the reverse KL divergence. However it is not immediately clear which direction should be used in optimization.

In the forward KL the weights are set by true distribution p. Thus at points such that $p(x) = 0$ the approximating distribution q is free to differ from the true distribution p since to difference will not contribute to the overall divergence. At points of positive mass, where $p(x) > 0$ the divergence $\log\frac{p(x)}{q(x)}$ will positively contribute to the overall objective term. Thus at these points, the difference between the approximating

distribution q and the true distribution p will aimed to be minimized. Forward KL is commonly said to be zero avoiding for the approximating distribution since $q(x) = 0$ at points when $p(x) > 0$ would result in an infinite contribution to the KL divergence. Thus forward KL forces q to avoid having zero mass when the true distribution is positive. Forward KL tends to over approximate the true distribution over the entire support $\mathcal{X}$ spreading the approximation out by covering the entire distribution. In the case of reverse KL the approximating distribution q determines the weights of the objective function. Points where $q(x) = 0$ have no contribution to the overall divergence even when the true distribution p has positive mass. However points $q(x) > 0$ contribute positive weight so the difference between the true distribution and the approximating distribution is minimized. Reverse KL is said to be zero forcing for the approximating distribution since at points $p(x) = 0$ and $q(x) > 0$ the contribution to the KL divergence is infinite thus q is forced to zero. Reverse KL thus would tend to underestimate the true distribution across the support $\mathcal{X}$. Approximating p well at specific parts, locking onto one mode of the true distribution rather than spreading the entire approximation out as in the case of forward KL.

Malinin and Gales [30, 31] propose a parallel solution to Evidential Deep Learning call Prior Networks. The output of Prior Networks explicitly parameterize a prior distribution whereby $\boldsymbol{\alpha} = f(\mathbf{x}|\mathbf{W})$. The target distribution however differs to the Evidential Deep Learning target by restricting the mass assigned to the correct class label to a finite constant β. Since in Evidential Deep Learning there is no limit to the amount of evidence that is allowed to be placed on the correct label as infinite evidence is the requirement of full certainty. Both methods however target flat masses $\alpha_j = 1$ for $j \neq k$ where k is the true label.

Following the derivation from their paper, the target Dirichlet distribution $p(\mathbf{p}|\boldsymbol{\beta}^{(k)})$ is parameterized by $\boldsymbol{\beta}^{(k)}$ where k is the true class label. The target parameterization being $\beta_j^{(k)} = \beta + 1$ for $j = k$ and $\beta_j^{(k)} = 1$ for $j \neq k$. Hence forcing a flat distribution over incorrect classes and a positive finite mass for the correct label. Then using the forward KL divergence as the optimization procedure between the target distribution aforementioned and the output of the model $p(\boldsymbol{p}|\mathbf{x}, \mathbf{W})$ the following loss can be set up:

$$\mathcal{L}^{\text{Forward}} = \sum_i \mathbf{1}\{y = i\} \cdot \mathbf{KL}(p(\mathbf{p}|\boldsymbol{\beta}^{(i)})||p(\boldsymbol{p}|\mathbf{x}, \mathbf{W})) \tag{3.10}$$

Using the forward KL divergence however sets up the following issue. Given a data set $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^N$ the expected loss is,

$$\begin{aligned}\mathbb{E}_{p(\mathcal{D})}\Big[\mathcal{L}^{\text{Forward}}\Big] &= \mathbb{E}_{p(\mathcal{D})}\Big[\sum_i \mathbf{1}\{y = i\} \cdot \mathbf{KL}(p(\mathbf{p}|\boldsymbol{\beta}^{(k)})||p(\boldsymbol{p}|\mathbf{x}, \mathbf{W}))\Big] \\ &\approx \mathbb{E}_{p(\mathbf{x})}\Big[\mathbf{KL}(\sum_i p(y = i|\mathbf{x}) \cdot p(\mathbf{p}|\boldsymbol{\beta}^{(k)})||p(\boldsymbol{p}|\mathbf{x}, \mathbf{W}))\Big]\end{aligned}$$

This expectation thus calculates the KL between the model prediction and a mixture of Dirichlets [31] each with their own target parameterizations. Hence the mixture has a mode in each corner of the simplex. As discussed, the forward KL is zero avoiding and approximates by spreading the distribution over each mode. The problem with forward KL can be explained by considering uncertainty estimation. In the case of high data uncertainty, rather than spreading out over each mode it is desirable instead to have a unimodal distribution with a single mode in the centre of the simplex. This can be avoided by using the reverse KL divergence in the loss:

$$\mathcal{L}^{\text{Reverse}} = \sum_i \mathbf{1}\{y = i\} \cdot \mathbf{KL}(p(\boldsymbol{p}|\mathbf{x}, \mathbf{W})||p(\mathbf{p}|\boldsymbol{\beta}^{(i)})) \tag{3.11}$$

as before, taking the expectation with respect to the training set:

$$
\begin{aligned}
\mathbb{E}_{p(\mathcal{D})}\Big[\mathcal{L}^{\text{REV}}\Big] &= \mathbb{E}_{p(\mathcal{D})}\Big[\sum_i \mathbf{1}\{y=i\}\cdot \mathbf{KL}(p(\boldsymbol{p}|\mathbf{x},\mathbf{W})||p(\mathbf{p}|\boldsymbol{\beta}^{(i)}))\Big] \\
&\approx \mathbb{E}_{p(\mathbf{x})}\Big[\sum_i p(y=i|\mathbf{x})\cdot \mathbf{KL}(p(\boldsymbol{p}|\mathbf{x},\mathbf{W})||p(\mathbf{p}|\boldsymbol{\beta}^{(i)}))\Big] \\
&= \mathbb{E}_{p(\mathbf{x})}\Big[\sum_i p(y=i|\mathbf{x})\int p(\boldsymbol{p}|\mathbf{x},\mathbf{W})\log\frac{p(\boldsymbol{p}|\mathbf{x},\mathbf{W})}{p(\mathbf{p}|\boldsymbol{\beta}^{(i)})}\Big] \\
&= \mathbb{E}_{p(\mathbf{x})}\Big[\int p(\boldsymbol{p}|\mathbf{x},\mathbf{W})\sum_i p(y=i|\mathbf{x})\log\frac{p(\boldsymbol{p}|\mathbf{x},\mathbf{W})}{p(\mathbf{p}|\boldsymbol{\beta}^{(i)})}\Big] \\
&= \mathbb{E}_{p(\mathbf{x})}\Big[\int p(\boldsymbol{p}|\mathbf{x},\mathbf{W})\big[\log p(\boldsymbol{p}|\mathbf{x},\mathbf{W})\Big(\sum_i p(y=i|\mathbf{x})\Big) \\
&\qquad\qquad - \sum_i p(y=i|\mathbf{x})\log p(\mathbf{p}|\boldsymbol{\beta}^{(i)})\big]\Big] \\
&= \mathbb{E}_{p(\mathbf{x})}\Big[\int p(\boldsymbol{p}|\mathbf{x},\mathbf{W})\Big(\log p(\boldsymbol{p}|\mathbf{x},\mathbf{W}) \\
&\qquad\qquad - \log\prod_i p(\mathbf{p}|\boldsymbol{\beta}^{(i)})^{p(y=i|\mathbf{x})}\Big)\Big] \\
&= \mathbb{E}_{p(\mathbf{x})}\Big[\mathbf{KL}(p(\boldsymbol{p}|\mathbf{x},\mathbf{W})||\prod_i p(\mathbf{p}|\boldsymbol{\beta}^{(i)})^{p(y=i|\mathbf{x})})\Big] \\
&\approx \mathbb{E}_{p(\mathbf{x})}\Big[\mathbf{KL}(p(\boldsymbol{p}|\mathbf{x},\mathbf{W})||p(\mathbf{p}|\tilde{\boldsymbol{\beta}})\Big]
\end{aligned}
$$

with $\tilde{\boldsymbol{\beta}} = \sum_i p(y=i|\mathbf{x})\boldsymbol{\beta}^{(i)}$. This expectation instead calculates the KL between a geometric mixture of Dirichlets which is a standard Dirichlet with parameters set by the arithmetic weightings by each class. Now in the case of low uncertainty, the distribution is a sharp Dirichlet in one particular corner of the simplex. Whereas in the case of high uncertainty, the Dirichlet is set to a single mode distribution set at the centre of the simplex as desired. In both cases of uncertainty using the reverse KL as the objective measure yields a unimodal distribution. Examining the KL term,

$$
\begin{aligned}
\mathbf{KL}(p(\boldsymbol{p}|\mathbf{x},\mathbf{W})||p(\mathbf{p}|\tilde{\boldsymbol{\beta}})) &= \int p(\mathbf{p}|\mathbf{x},\mathbf{W})\log\frac{p(\mathbf{p}|\mathbf{x},\mathbf{W})}{p(\mathbf{p}|\tilde{\boldsymbol{\beta}})} \\
&= \mathbb{E}_{p(\mathbf{p}|\mathbf{x},\mathbf{W})}[-\log p(\mathbf{p}|\tilde{\boldsymbol{\beta}})] - \mathbb{H}[p(\mathbf{p}|\mathbf{x},\mathbf{W})]
\end{aligned}
$$

the first term represents the cross entropy and the second the differential entropy of $p(\mathbf{p}|\mathbf{x},\mathbf{W})$. Examining

the cross entropy and dropping additive constants we see

$$
\begin{aligned}
\mathbb{E}_{p(\mathbf{p}|\mathbf{x},\mathbf{W})}[-\log p(\mathbf{p}|\tilde{\boldsymbol{\beta}})] &= -\mathbb{E}_{p(\mathbf{p}|\mathbf{x},\mathbf{W})}[\log D(\mathbf{p}|\tilde{\boldsymbol{\beta}})] \\
&= -\mathbb{E}_{p(\mathbf{p}|\mathbf{x},\mathbf{W})}[\sum_{i=1}^{K} p(y=i|\mathbf{x}) \log D(\mathbf{p}|\boldsymbol{\beta}^{(i)})] \\
&= -\mathbb{E}_{p(\mathbf{p}|\mathbf{x},\mathbf{W})}\Big[\sum_{i=1}^{K} p(y=i|\mathbf{x}) \log \prod_{j=1}^{K} p_j^{\beta_j^{(i)}-1}\Big] \\
&= -\mathbb{E}_{p(\mathbf{p}|\mathbf{x},\mathbf{W})}\Big[\sum_{i=1}^{K}\sum_{j=1}^{K} p(y=i|\mathbf{x})(\beta_j^{(i)}-1) \log p_j\Big] \\
&= -\sum_{i=1}^{K}\sum_{j=1}^{K} p(y=i|\mathbf{x})(\beta_j^{(i)}-1)\mathbb{E}_{p(\mathbf{p}|\mathbf{x},\mathbf{W})}[\log p_j] \\
&= \beta \cdot \sum_{i=1}^{K} p(y=i|\mathbf{x})(\psi(\alpha_0) - \psi(\alpha_i))
\end{aligned}
$$

Which is exactly the same form as the cross-entropy loss derived in the EDL loss function scaled by the β constant. Recovering this loss function using reverse KL divergence demonstrates the validity of cross entropy loss as a suitable objective for our proposed evidential framework to learn unimodal distributions for handling uncertainty.

One possible disadvantage of the Prior Networks model however is the fixed constant β forces a fixed prediction confidence for all correct examples. This is not suitable for most training sets since all classes are not equally distinguishable. Examples which the model is certain about should not have the same confidence as classes of examples seen once. In the Evidential Deep Learning the belief mass for the true class is an unconstrained free parameter. In theory this mass can tend towards infinity as uncertainty tends to zero and we achieve a state of absolute certainty.

Chapter 4

Object Detection

Understanding our surrounding environment is a fundamental and effortless task easily taken for granted. Visual perception of what's around us dictates how we operate and the decisions we make and so automating this process is an extremely exciting challenge in computer vision. Object detection is the computer vision problem of locating and identifying relevant objects in images and video. Object detection algorithms until recently relied on hand crafted features using classical techniques. However recent advances with deep learning have led to real-time detection systems with state of the art detection accuracy. This chapter begins with an overview of convolutional neural networks. Following this a look at object detection systems and the two main methodologies, region proposal frameworks and then unified regression and classification frameworks.

4.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) have demonstrated breakthrough practical success across many applications in both image and video processing as well as with speech and audio. CNNs operate best on grid based data such as images and video frames which form 2D pixel grids but also 1D grid structures such as sequential time series. CNNs are a special type of neural network, due to use of the convolution operation employed instead of standard matrix multiplication in certain layers [16]. The idea of convolutional layers is that applying convolution to local regions of pixels, the newly formed image will better represent region features. Since each pixel in the new image will encapsulate regional information and reflect the presence of features in each particular area. Considering the problem of image classification or dealing with specific objects in an image; the points of interest will tend to occupy specific local regions of the overall image. Additionally, objects tend to have similar appearance independent of location in the overall frame. More formally, objects tend to have local spatial support and tend to be invariant to translation. To construct this as a framework, fully connected layers are replaced with locally connected layers whereby each neuron connects instead to a local region called a receptive field. Convolution is applied on each $n \times n$ receptive field, with n the convolution size. Whether or not these regions overlap is determined by the stride of the convolution. Finally to account for invariance in translation, each neuron must share the same weights as all other neurons. Combining local connections and shared weight parameters defines the convolution layer and hence the name Convolutional Neural Network.

The convolution operation can be formally defined for both continuous and discrete settings. In most machine learning applications where data will be discretized. In the one-dimensional case we define convolution of two functions f and g as the infinite summation

$$(f * g)(t) = \sum_{x=-\infty}^{\infty} f(x)g(t-x) \tag{4.1}$$

In practice and when using CNNs convolution will be applied over multiple axes at a time. If processing a 2D image I with a 2D kernel K then two-dimensional convolution can be implemented by

$$(I * K)(i, j) = \sum_m \sum_n I(m, n)K(i - m, j - n) \tag{4.2}$$

Applying each convolution layer, the resultant output is commonly known as a feature map or an activation map. Typically a normalization procedure is applied in the form of an activation function. Usually the activation functions is some non-linearity for example the ReLU function to remove any negative values. Finally in this process is the pooling or down-sampling layer. The pooling function reduces the spatial dimension of a feature map output by returning a summary statistic over a local region of outputs. Example pooling functions are average or max pooling which respectively return the average or maximum among each regions. Pooling can be viewed as an infinitely strong prior which allows the layer to be approximately invariant of small translations [16] which is useful when the presence of an object is more important than where it is found. With these three component layers, hierarchical models for feature extraction can be formed by creating deep stacks of convolutions, non-linearities and pooling layers. The resultant CNN is then able to learn multiple levels of representations automatically in the input data from pixel-level to high level semantic features.

The origins of the CNN began in 1980 with "Neocognitron" by Fukushima and later in 1989 with "LeNet" by LeCun. Yann Le Cunn's paper [26] was the first to train CNNs with the backpropagation algorithm and showed they could be applied to the task of classifying handwritten digits. The breakthrough moment for CNNs however came in 2012 with AlexNet [23]. The paper was seen as a major turning point in the computer vision community and propelled the adoption of deep learning over classical methods. AlexNet developed by Alex Krizhevsky used convolutional neural networks to nearly halve the best seen error rate in the ImageNet contest. The ImageNet contest being an image classification competition of 1.4 million images with 1000 unique classes and was seen as the major benchmark within the computer vision and deep learning community running from 2010 to 2017. The best methods pre-2012 relied on the use of hand crafted features however all subsequent winners made use of deep CNNs, with network getting deeper each year.

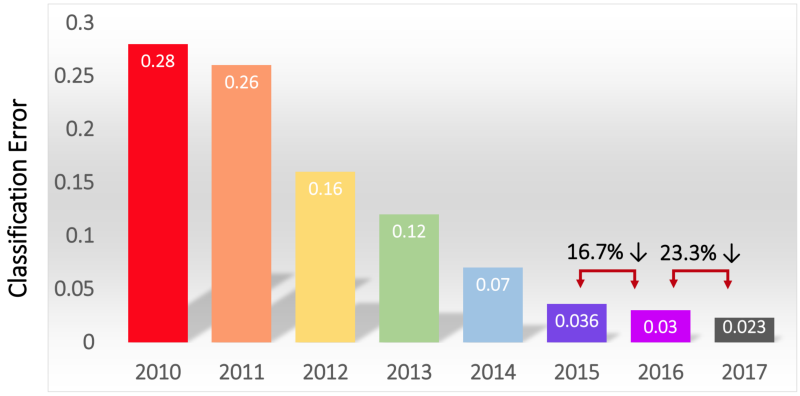


Figure 4.1: Graphic released by the ImageNet team showing the progress between 2010 and 2017. AlexNet in 2012 improved the classification error from 0.26 to 0.16 and propelled the use of deep CNNs in the computer vision community.

4.2 An Overview of Object Detection Methods

Object detection algorithms aim to solve the problem of detecting objects of interest and knowing where they are located. Methods for object detection can be split between classical machine learning techniques and modern deep learning techniques. In the classical approach, features used are typically hand-crafted whereas deep learning methods use neural networks like CNNs to infer features from data. Naively done, one could use a sliding window to divide an image into a grid and slide a classifier over each region. In theory each positive classification provides the location and class of each object in the image. The idea seems natural however it is not obvious how big the windows should be to account for different object sizes and aspect ratios. Using a pyramid representation one can re-size an image to different scales and apply a fixed-size detector on all patches or alternatively scanning an image with multi-scaled sliding windows could resolve this. These methods however are clearly inefficient and would propose extensive lists of mainly redundant candidate windows - thus we seek an alternative solution for region proposal.

Object detection can be split into two main methodologies: region proposal frameworks and unified regression and classification frameworks. Region proposal frameworks are typically two stage networks which separate the process of region proposal and detection. Whereas unified frameworks map the image directly into class probabilities and their bounding boxes. In region proposal networks the first step is to identify and propose regions of interest and second is then to classify within the region and regress the boundaries for each object. Unified frameworks instead compose the intermediate stages of region proposal, feature extraction and finally classification and regression into single step detector. Unified methods have the advantage of faster inference however are generally less accurate than two-stage methods.

4.2.1 Region Proposal Detectors

R-CNN [13] (2014) was an early, post AlexNet proposal for region extraction which makes use of an algorithm called Selective Search. Selective Search produces around 2000 proposals for regions of interest per image and uses cues such as texture and pixel intensity to reduce over the search space. R-CNN then uses a CNN for feature extraction over the optimized subset of regions before passing the extracted features into a pre-trained classifier such as an support vector machine to predict the main class for each region. Object detection algorithms typically result in multiple bounding boxes predictions per object detected. Non-maximum suppression is a method for choosing between overlapping bounding boxes based on choosing greedily with respect to their score. Applying non-maximum suppression in R-CNN results in a final set of detections with a single bounding box per detection. This method of automatic region proposal is a clear improvement over the naive pyramid approach.

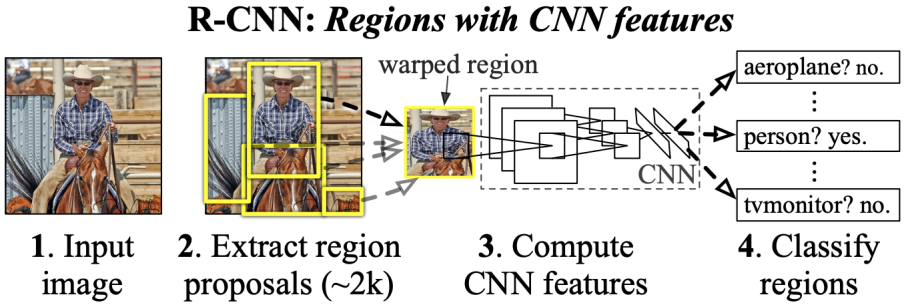


Figure 4.2: Overview of R-CNN method. 1. Takes an input image, 2. uses selective-search to extract 2000 region proposals, 3. The regions are then warped such that a CNN is able to compute features before finally 4. classify each region with a class specific SVM. Image taken from original paper [13]

An issue with R-CNN is that due to fully connected layers in the CNN, R-CNN warps proposed regions into squares without maintaining the aspect ratio. This leads to undesired distortions and likely reduced

recognition accuracy [45]. Spatial Pyramid Pooling [17] introduced in **SPPnet** handles this by generating a fixed length representations regardless of image size and scale, hence is robust to deformation. A second issue of R-CNN is the inefficiency in evaluating a CNN on circa 2000 region proposals suggested from selective search. SPPnet instead computes feature maps first using the entire image before pooling features into regions called sub-images. It does this by using the convolutional outputs to project different region proposals of arbitrary sizes into fixed-length features; this final layer after the convolutional layers is referred to as the spatial pyramid layer.

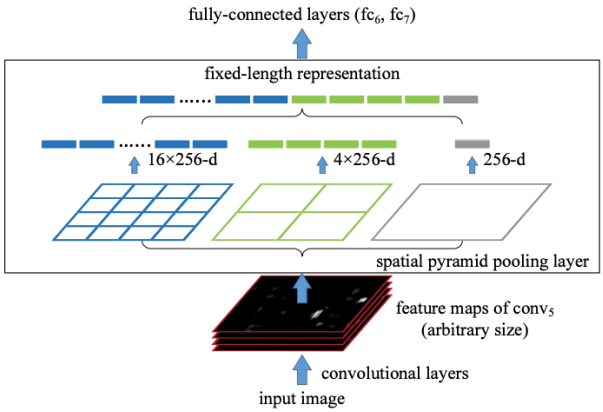


Figure 4.3: Image taken from Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition paper [17]

Girshick in 2015 published **Fast R-CNN** [12] to combine and improve on the ideas of R-CNN and SPP-net. The paper notes that the notable draw backs of the prior methods is that they both suffer from being multi-stage pipelines; first fine tuning the feature extractors, then classifying with SVMs on the region proposals before finally training bounding box regressors. In the case of SPP-net, a limiting factor was the spatial layer did not allow for efficient backpropagation. Fast R-CNN introduced an end-to-end training method solving for backpropagation in the spatial pooling layer and combining the bounding box regression into the overall neural network training using a multi-task loss function. By using a multi-task loss function Fast R-CNN jointly optimizes both the classification task and the regression task and allowed for tuning in all layers. Fast R-CNN saw reduced training time, reduced memory requirements with overall improved detection accuracy.

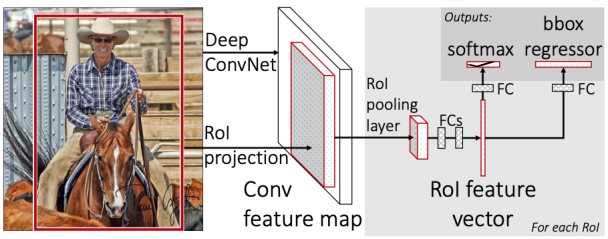


Figure 4.4: Image taken from Fast R-CNN paper [12]

While Fast R-CNN and SPPnet made vast advances in accuracy they still suffered from the bottleneck of the regional proposal algorithms used such as selective search. Region Proposal Networks (RPNs) [37] are fully convolutional networks able to simultaneously predict the object boundaries with object scores at each location. The **Faster R-CNN** framework uses anchor boxes to handle for different scalings and aspect

ratios of objects. At each location in the image, an RPN is used to assign for each anchor box the probability of it containing an object or whether to ignore it as background. Running the RPN on each anchor box, fixed length vectors are then passed to the classifier and regression layers to provide the scores and bounding boxes. Hence exposing convolutional feature representations used at location specific anchor boxes with the detection network, regional proposals come for free. This method makes Faster R-CNN circa 10x faster than Fast R-CNN.

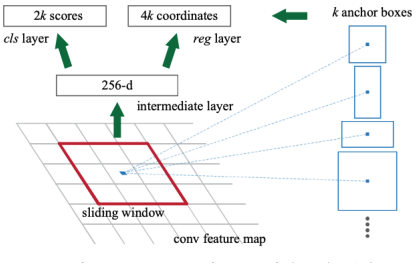


Figure 4.5: Image taken from Faster R-CNN paper [37] displaying the Region Proposal Network used.

Comparing modern object detectors is a trade off between speed and accuracy. As this dissertation motivates the use of real time uncertainty estimation we opt for a real time detection system. Figure 4.6 below shows the spread of accuracy vs speed, measured in mean average precision and frames per second respectively of different detectors. The YOLO [36] and YOLOv2 [35] frameworks which I focus on were built to be unified real time detection systems. The original YOLO which I describe below is almost twice as fast as the best detector left of the dotted line however suffers from a drop in accuracy. YOLOv2 was released as an improvement with the blue dots representing different resolution rates. Single Shot MultiBox Detector (SSD) [28] is another popular unified method which retains high speed and high accuracy however for this work we use the YOLO framework.

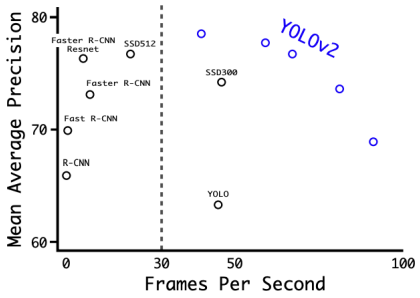


Figure 4.6: Accuracy and speed of various detectors on the VOC 2007 dataset. Image taken from YOLOv2 paper [35].

4.2.2 You Only Look Once (YOLO)

YOLO [36] was designed, as the paper title suggests to be a "Unified, Real-Time Object Detection" system. The YOLO approach is to treat object detection as a regression problem over both the bounding boxes and respective class probabilities. YOLO presents a single neural network solution able to predict both localization and classification for full images in a single evaluation. It does so by extracting features for the entire image and using these to predict the bounding boxes across all classes simultaneously. The YOLO design means training can be done end-to-end and detections made in real time.

For each image, YOLO divides input image into a $S \times S$ grid whereby each grid cell predicts one object. Cell ownership of an object is determined by which cell the center of that object falls into. Each grid cell

predicts a fixed number B of bounding boxes and confidence scores for each box. The box confidence score is a reflection of how confident the model is that an object is contained in the box and how accurate the box is. Formally box confidence is computed as $\mathbf{Pr}(\text{Object}) * \text{IOU}(\text{pred, truth})$ where $\mathbf{Pr}(\text{Object})$ represents the probability that the box contains an object while IOU(pred, truth) is intersection over union defined above between the predicted box and the true box. If no object belongs in the cell, then the box confidence should be 0, else the confidence score should be the IOU between the predicted box and the true box. Each cell also predicts a set of C conditional class probabilities, one for each class conditioned on there being an object present in the cell $\mathbf{Pr}(\text{class}_c|\text{Object})$. Each bounding box prediction requires five components: four (x, y, w, h) spatial coordinates plus one confidence score. YOLO was designed as a CNN that for each image returns a single predicted vector encoding both object classifications and bounding boxes. Thus each YOLO prediction will be encoded in a tensor of length $S \times S \times (B * 5 + C)$.

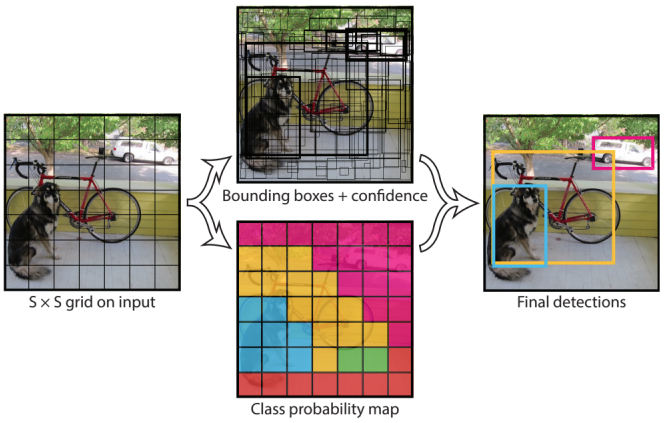


Figure 4.7: YOLO treats detection as a regression problem; dividing each input image into a $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence scores and C class probabilities. Image taken from original YOLO paper [36].

The YOLO detection network has 24 convolutional layers followed by 2 fully connected layers. The network is optimized using a multi-part loss function with three component parts: **coordinate loss**, **confidence loss** and **classification loss**:

$$\underbrace{\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right]}_{\textbf{COORDINATE}}$$

$$\underbrace{+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} (C_i - \hat{C}_i)^2}_{\textbf{CONFIDENCE}}$$

$$\underbrace{+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{obj} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2}_{\textbf{CLASSIFICATION}}$$

with $\mathbb{1}_{ij}^{obj}$ denoting if the jth bounding box in cell i is responsible for an object and $\mathbb{1}_{i}^{obj}$ denoting if there is an object in cell i. Breaking down this loss function into its component parts.**Coordinate loss** sums the squared errors in the predicted box locations and dimensions, counting only the boxes that are responsible for object detections. To to reflect that not all errors are equal for large and small boxes, since small errors in larger boxes is less important than in small ones. The model predicts the square root of the box height and width instead. **Confidence loss** is also a two part loss which sums the squared error in confidence scores $\hat{C}$ for box locations with an object and locations with no object. Since most boxes do not contain objects in practice, to handle for class imbalance the authors weight the no object sum using a factor λ_{noobj}. Finally **classification loss** measures the squared error in the conditional class probabilities for each class in the cells containing an object.

YOLOv2 [35] was the successor model to the original with the aim of being faster and more accurate. In this updated version the authors make use of various methods such as batch normalization [19] to regularize the model and fine tune using different image resolutions. Full details of the modifications can be found in the paper however I will focus on two specific changes which will be important for this model that this dissertation uses. In the original YOLO bounding boxes are initially guessed at random which led to unstable gradients during training. YOLOv2 instead introduces anchor box priors and predicts offsets on these boxes instead of predicting the dimensions outright. Since the network will then learn to adjust the boxes if given suitable priors. Rather the choosing the anchor priors by hand, k-means clustering is ran on the bounding boxes from the training set to automatically infer good priors. Five is chosen by the authors as the optimal number of bounding boxes when trading off accuracy and complexity. Now instead of predicting five random bounding boxes as in original YOLO, five sets of offsets to prior boxes are predicted instead. By constraining the offset values, the predictions remain diverse yet retain closeness to the prior shape. For each bounding box, the network predicts coordinates (t_x, t_y, t_w, t_h). Knowing the cell location (c_x, c_y) and the width and height of the anchor prior (p_w, p_h), the bounding box dimensions (b_x, b_y, b_w, b_h) can be recovered from the equations in figure 4.8.

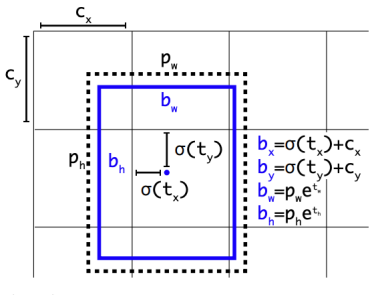


Figure 4.8: Bounding boxes are predicted as offsets to anchor box priors. The priors are found from running k-means clustering on the training set bounding boxes. Image taken from YOLOv2 paper [35].

The network is also modified removing the fully connected layers previously used to predict bounding boxes, instead becoming a fully convolution network. The output tensor is also modified to now predict classification probabilities for each boundary box instead of just each cell. Each bounding box prediction now includes the 4 bounding parameters, a single confidence score and C class probabilities. Now each YOLOv2 prediction will return a tensor of length $S \times S \times B \times (5 + C)$.

Chapter 5

Evidential YOLO

5.1 Proposed Model

This chapter ties together Evidential Deep Learning and object detection to build uncertainty aware detection systems. Chapter 3 showed the practical application of Subjective Logic to modify existing neural networks to enhance class probabilities with an estimate for the epistemic uncertainty. In Chapter 4, the YOLOv2 [35] framework was outlined as a model for detection. Making bounding box predictions in each grid cell based on a set of anchor box priors. Each bounding box prediction includes 4 bounding parameters, a single confidence score and C class probabilities. Evidential YOLO (EDL-YOLO) is the proposed model in this work to accompany each bounding box prediction corresponding class probabilities and uncertainty scores. However it should be clear that any deep learning detection framework which makes use of a softmax layer could be used. In YOLOv2 the logits relating to class labels are passed through a softmax layer to give a set of normalised class probabilities. As shown in figure 5.1, bounding box confidence and coordinates are kept constant, but the class logits are instead fed into the EDL framework to calculate class probabilites and an uncertainty score.

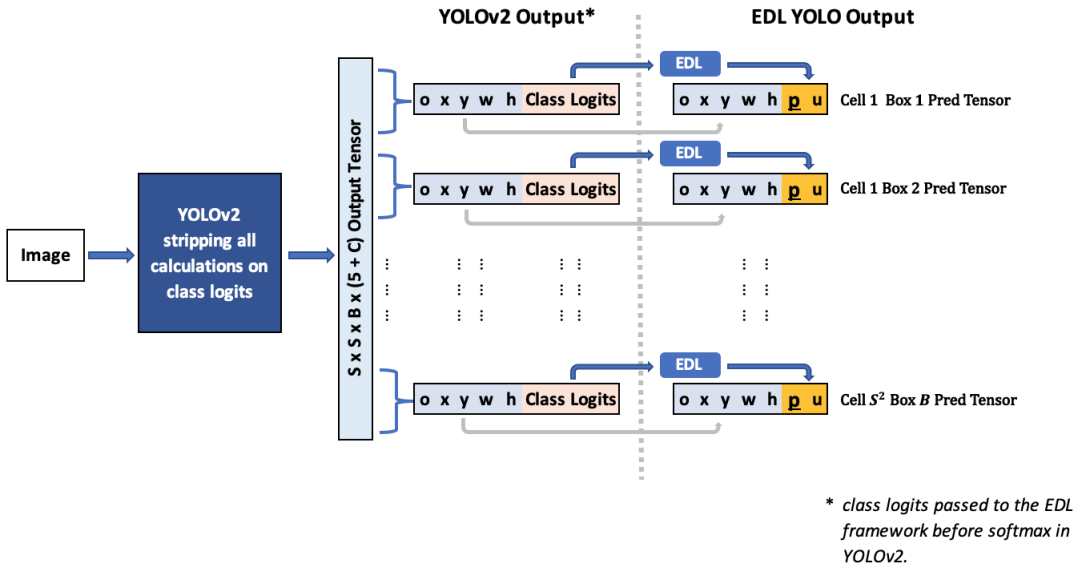


Figure 5.1: Evidential YOLO Framework

5.1.1 Inference

During inference, EDL-YOLO now predicts B bounding boxes in each of the S^2 grid cells per image. Each bounding box prediction $(o, x, y, w, h, \mathbf{p}, u)$ contains the box confidence score o, the four bounding box parameters: centre coordinates (x, y), width w and height h. The EDL components $(\mathbf{p}, u)$ as defined in Chapter 3 and are determined by the logit outputs from the neural network. Rather than passing the logits through a softmax function, using a ReLU or an clipped-exponential function will remove any negative mass propagating forward. The resultant tensor is termed the evidence $\mathbf{e}$ and the Dirichlet parameters $\boldsymbol{\alpha}$ are simply $\boldsymbol{\alpha} = \Big\langle e_1 + 1, \ldots, e_K + 1 \Big\rangle = \mathbf{e} + 1$. The class probability vector is given by the expected value of the Dirichlet $\mathbf{p} = \alpha_0^{-1}\boldsymbol{\alpha}$ with $\alpha_0 = \sum_j \alpha_j$. The classification uncertainty is calculated as $u = \alpha_0^{-1} C$. Each bounding box now has two separate confidence metrics for detection: object detection confidence o and classification probability u. Detection confidence o corresponds to a measure of how confident the model is an object is present. The classification uncertainty u however is a proxy for the epistemic uncertainty, or the models knowledge in classifying the sub image in the bounding box. In each forward pass, EDL-YOLO predicts multiple bounding boxes across for the entire image and multiple detections for each object. Non-maximum suppression is then used to handle multiple detections by choosing greedily with respect some confidence metric. We thus have two models to choose from, one which selects overlapping boxes with respect to object confidence o and another which respect to the classification uncertainty u.

5.1.2 Modified Loss Function

As the modification focuses on classification probabilities, EDL-YOLO retains the coordinate and confidence components from the original YOLO. The classification loss however changes to the EDL loss function proposed in Chapter 3. The model should only be penalized in cells containing an object and so the proposal is the sum over all S^2 cells multiplying the EDL loss component by $\mathbb{1}_i^{obj}$ to reflect this. The EDL classification loss is comprised of the expected cross entropy loss and the annealed KL regularization term. The EDL loss component is scaled by λ_{EDL} and experimentally set to 5 saw the most stable results. Combining all loss components presents the new modifed loss function:

$$
\begin{aligned}
&\underbrace{\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \Big[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \Big]}_{} \\
&\underbrace{\quad + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \Big[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \Big]}_{\textbf{COORDINATE}} \\
&\underbrace{\quad + \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} (C_i - \hat{C}_i)^2}_{\textbf{CONFIDENCE}} \\
&\underbrace{\quad + \lambda_{\text{EDL}} \sum_{i=0}^{S^2} \mathbb{1}_i^{obj} \Big[\mathcal{L}_i^{CE}(\mathbf{W}) + \lambda_t \cdot \mathbf{KL}(D(\mathbf{p}_i | \widetilde{\boldsymbol{\alpha}}_i) || D(\mathbf{p}_i | \mathbf{1})) \Big]}_{\textbf{EDL Classification Loss}}
\end{aligned}
$$

5.2 Implementation

The implementation of EDL-YOLO was built upon YAD2K: Yet Another Darknet 2 Keras, an open source Python implementation of the YOLOv2 architecture using 90% Keras [6] and 10% Tensorflow [32]. The full YAD2K project can be found at: https://github.com/allanzelener/YAD2K. All of the implementation was done in Google Colabaratory (Colab), a free to use Jupyter notebook environment that runs entirely in the cloud. Colab was released as part of efforts to widen machine learning education and research and offers a runtime configured for deep learning with a free to use GPU [5]. The use of the GPU was vital for fast inference times and for handling the vast amount of computation required during the training leg of the project. Creating the EDL-YOLO framework as per figure 5.1 requires no modification of the core YOLO architecture, only that the network output is used for a different inference protocol. A new loss function was also programmed so that the weights were re-optimized to the new objective function. Initially off-the-shelf YOLO weights, available on the authors website https://pjreddie.com/darknet/yolo/ were downloaded and loaded into the model.

In the experiments, the YAD2K framework requires fixed size 416×416 input images. The algorithm splits the image into 13×13 grids with each grid predicting 5 bounding boxes. In ascerting positive detections in the loss function, only bounding boxes with an intersection over union (IOU) of $> 50\%$ are deemed valid detections. The dataset used was from the PASCAL Visual Object Challenge (VOC) [8]. Making use of the entire VOC2007 and VOC2012 training and validation provided $16,551$ images in total to train on. Then using the VOC2012 testing set provided $4,952$ images to validate on. The goal of the VOC challenge was a series of computer vision challenges, with the main goal to recognise objects from realistic scenes from twenty possible classes. YAD2K's framework was trained using the Microsoft's COCO dataset [27] which has 80 possible classes. To handle for this, the final layer from the original YOLO is removed and a new final 2D convolutional layer is added with $\#\text{Boxes} * (\#\text{Num Classes} + 5) = 5 * (20 + 5)$ filters.

Inside the YOLO model as per fig 5.1 the softmax layer is removed and class logits are passed through a set of EDL calculations. In addition, the loss function within YAD2K had to be overwritten to calculate the new modified EDL-YOLO loss. In addition other metrics were coded to track the evidential progress: namely monitoring uncertainty estimation and evidence generation for correct classifications vs incorrect classification with each epoch. If the model is behaving, as the overall loss decreases, training accuracy should increase and the total evidence for correct classifications should increase. This is while evidence for incorrect classifications should tend to zero. The idea being in the presence of out-of-distribution image frames, objects would be confidently detected and localized, however if they are occluded, unknown classes or different aspect ratios, the classifications would be assigned high uncertainty with detection. Classification accuracy is also determined by comparing the predicted class for each object detection with the ground truth class in each cell containing an object and taking the mean.

5.3 Training the model

5.3.1 Setup

Once the architecture and data was ready a strategy to train was needed. The problem was deciding which layers of the existing YOLOv2 framework to re-train. I tested a variety of approaches, from retraining the entire model which due to my limited hardware and time compared to the original authors led to low accuracy. To the extreme case of freezing the entire YOLOv2 model apart from the newly initialized final convolutional layer. This however simply overfitted to the training set. The best results were found by freezing the model up to the final pooling layer. Then retraining the final convolutional layers as per figure 5.2 as well as the final convolution layer. All initializations were done using Glorot Initialisation [14]. This method allowed 5 hidden convolutional and a final convolutional layer to be optimized. This mix sufficiently avoid randomnesss to propagate through the entire model and destroy the strong features already learned by YOLOv2. However by freezing the bottom half and learning new top weights allowed new suitable patterns to be generated to predict for twenty classes in the evidential setting. This method maintained strong

performance on the other loss components, namely the coordinate and the confidence loss. The most stable hyperparamaters found were a batch size of 32 and default learning rates for Adam optimization [22]. The loss function parameters used were: $(\lambda_{\text{coord}}, \lambda_{\text{noobj}}, \lambda_{\text{EDL}}) = (1, 5, 5)$

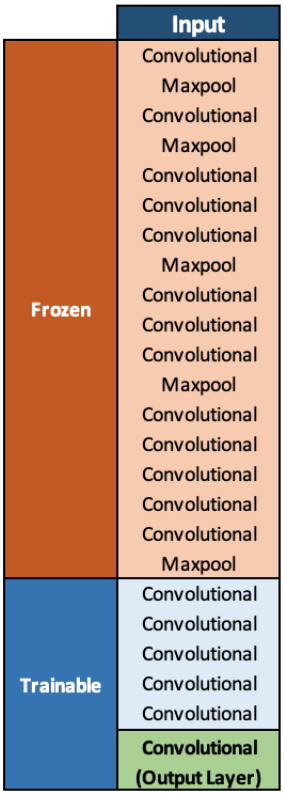


Figure 5.2: EDL-YOLO network. During training, only layers after the final Maxpool are optimized. The final convolutional output is replaced to match the number of classes in PASCAL VOC.

5.3.2 Results

Plotting the EDL analytics in figure 5.11 (as per Sensoy) displays the progress of the evidential metrics. The top row displays the training results and the bottom the test results. The left side displays the estimated evidence predicted for correct vs. incorrect classifications. EDL models should predict high evidence for certain and near zero evidence when uncertain. In both the train and test case, the model generates lots of evidence when correct and near zero when it is wrong as expected. The right hand side displays the estimated uncertainty and overall accuracy in blue. For correct classifications uncertainty should be low as more evidence should be predicted, while when incorrect uncertainty should be high and evidence low.

In this model, the frozen section of YOLOv2 acts as a feature extractor which feeds into the trainable convolutional layers. Training for 20 epochs achieves around just over 80% accuracy in both the training and testing data, however it was found to get to above 90% classification accuracy, the model was simply generating too much evidence. Hence when the model was incorrect, the uncertainty was too low to be meaningful. Figure 5.11 displays this trend: in the top right image, the classification uncertainty is high as desired. But in the bottom right image, the testing misclassification uncertainty tends lower and lower with increasing accuracy - suggesting overfitting.

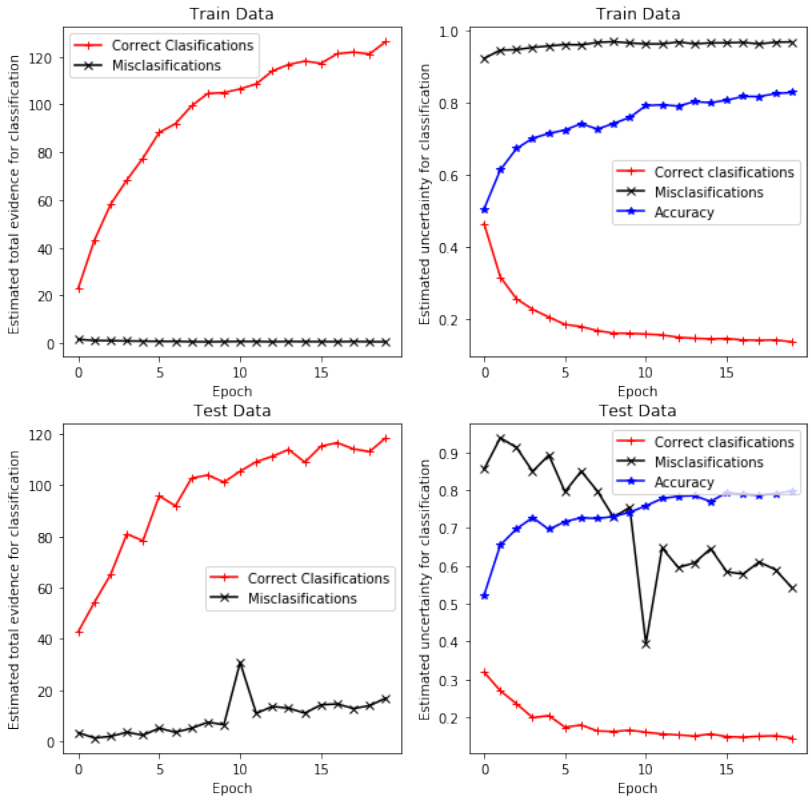


Figure 5.3: EDL Analytics during training. Left plots total evidence estimated for correct and incorrect classifications. Right plots accuracy score and estimated uncertainty.

In original YOLOv2 paper [35] Redmon breaks the training procedure into two parts. First training for classification and then for detection. The classification training was done using the ImageNet 1000 class classification dataset for 160 epochs. The training data was also augmented using random crops, rotations etc. The network is then trained for an additional 160 epochs this time for detection. Replacing the final convolutional layer to predict the bounding boxes as well as classifications. Since EDL-YOLO uses a large section of YOLOv2 as a feature extractor, this likely explains why the evidence for misclassification increases in the test set even as accuracy increases. In the original YOLO loss, classification loss is just the squared error loss. So as class specific features are generated the model generates evidence for that class. As with the example given in chaper 3 for the roundness of the digit zero. This feature also appears for both an eight and six and hence evidence is generated for these classes also. Thus to combat this problem, it is likely the entire feature space needs to be learned from scratch. However the computational resources and time required to retrain on ImageNet for 320 total epochs is not feasible. Hence EDL-YOLO settles for an average test accuracy of 80% on PASCAL VOC2012 with an average estimated 0.5 uncertainty score for misclassifications.

5.3.3 Comparison

Comparing EDL-YOLO to other detectors, mean average precision is a commonly used metric in object detection and explained in detail in appendix A.2. Figure 5.4 displays a summary of average precisions for the different VOC classes for different detection systems tested on VOC2012. EDL-YOLO has a MAP of 63.4 which isn't the greatest however scores higher than YOLOv1. However this is likely due to the powerful base feature extractor used. EDL-YOLO does perform very well on cars and people but overall the MAP is less than most other detectors. In defence of EDL-YOLO, training time and hardware was likely less exhaustive than the published models.

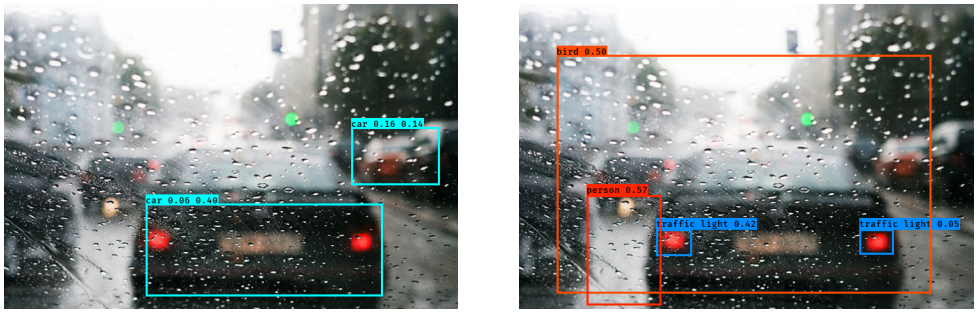
Method	data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast R-CNN	07++12	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
Faster R-CNN	07++12	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
YOLO	07++12	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
SSD300	07++12	72.4	85.6	80.1	70.5	57.6	46.2	79.4	76.1	89.2	53.0	77.0	60.8	87.0	83.1	82.3	79.4	45.9	75.9	69.5	81.9	67.5
SSD512	07++12	74.9	87.4	82.3	75.8	59.0	52.6	81.7	81.5	90.0	55.4	79.0	59.8	88.4	84.3	84.7	83.3	50.2	78.0	66.3	86.3	72.0
ResNet	07++12	73.8	86.5	81.6	77.2	58.0	51.0	78.6	76.6	93.2	48.6	80.4	59.0	92.1	85.3	84.8	80.7	48.1	77.3	66.5	84.7	65.6
YOLOv2 544	07++12	73.4	86.3	82.0	74.8	59.2	51.8	79.8	76.5	90.6	52.1	78.2	58.5	89.3	82.5	83.4	81.3	49.1	77.2	62.4	83.8	68.7
EDL YOLO 416	**07++12**	**63.4**	**83.1**	**66.2**	**68.7**	**53.4**	**43.5**	**72.3**	**79.6**	**72.6**	**41.8**	**56.5**	**38.6**	**60.3**	**80.9**	**70.6**	**77.2**	**41.4**	**63.8**	**45.4**	**84.1**	**67.2**

Figure 5.4: PASCAL VOC2012 test detection results.

5.4 Qualitative Results

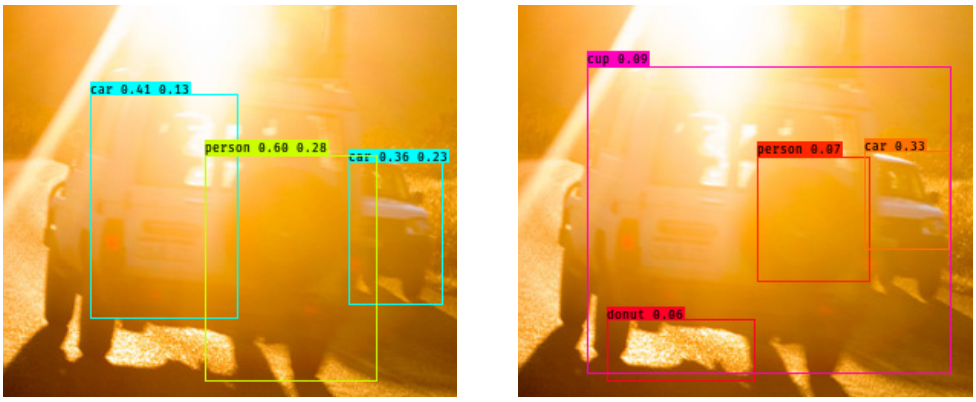
In this final section, we examine EDL-YOLO qualitatively by visualising the predictions. Each prediction is a bounding box labelled with predicted class label, class score and the estimated epistemic uncertainty.

5.4.1 Off-the-shelf YOLOv2 vs. EDL YOLO



(a) EDL-YOLO (b) YOLOv2

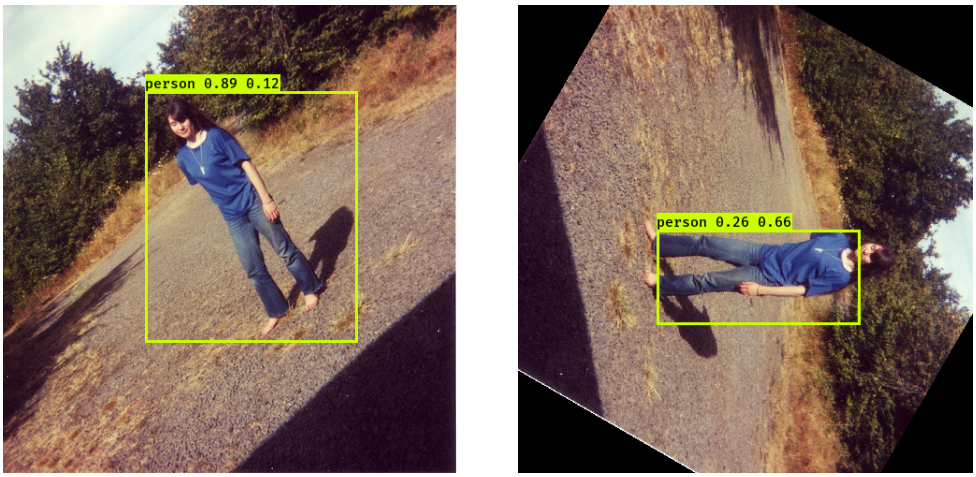
Figure 5.5: Rain blurring the lense: EDL-YOLO correct identifies the car infront and a car in the top of the frame. YOLOv2 fails to detect the main car and associates the brake lights with traffic lights. This example highlights a positive result for EDL-YOLO. The detection finds the car in an out-of-distribution example, predicts a low probability score yet a high uncertainty.



(a) EDL YOLO (b) YOLOv2

Figure 5.6: Intense sunlight imposing the image: EDL-YOLO detects both cars in the image however erroneously detects a person in the shadow of the car with quite high probability. Standard YOLOv2 does detect the overtaking car however detects the main car as a cup, a donut under the bonnet as well as a person in the same location as EDL-YOLO.

5.4.2 Rotation



(a) Upright (b) Rotated

Figure 5.7: This example tests EDL-YOLO's detection during rotation. Here EDL-YOLO correctly detects both the upright and the rotated person. This is another positive result for EDL-YOLO since rotating the image 90 degrees takes the image out-of-distribution. Yet the detection is correct and displays increased uncertainty during rotation.

5.4.3 Other Examples

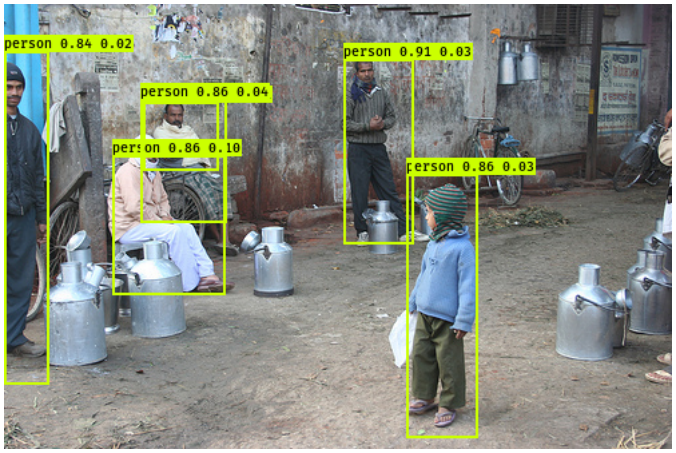


Figure 5.8: Group of people. Each person correctly detected.

Figure 5.9: Birds. EDL-YOLO correctly detects the top bird with flapping wings however wrongly detects the smaller bird with straight wings as an aeroplane.

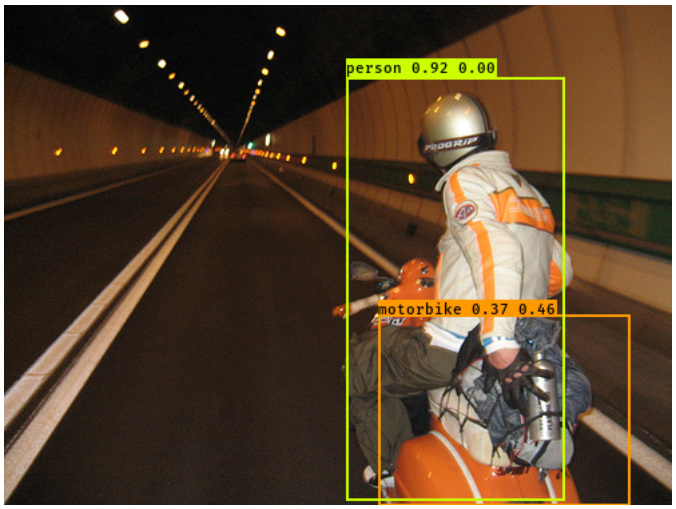


Figure 5.10: Overtaking motorbike. This is a positive result of the model. EDL-YOLO correctly detects the rider as a person with high probability and zero uncertainty. The motorbike which is off angle and occluded by the rider is also correctly detected but with high estimated uncertainty.

Figure 5.11: Small scale car and person. Both the car and person are correctly detected by EDL-YOLO despite being fairly far and against snow.

Chapter 6

Conclusion

6.1 Discussion

In this work we demonstrate it is possible to modify and train detection systems such as YOLO to become uncertainty aware. Evidential Deep Learning is the proposed model for real-time uncertainty estimation and this work provides a novel implementation of a non-sampling technique for estimating epistemic uncertainty in object detection. Specifically, we use enhance classification probabilities for each bounding box prediction with an epistemic uncertainty measure. Then using a modified non-max suppression algorithm, we select between predicted boxes using classification certainty instead of box confidence. The proposed framework replaces the softmax layer and uses the network output to instead parameterise a Dirichlet distribution over class labels. These outputs (logits) interpreted using subjective logic form the building blocks for belief mass assignment and uncertainty. This work fully justifies and derives the evidential loss function for deep neural networks proposed in the original Evidential Deep Learning paper and is successfully integrated into the YOLO detection loss function.

Evidential YOLO (EDL-YOLO) is the proposed model and we demonstrate robustness during out-of-distribution examples. In the case of intense lighting and rainy conditions which occlude the frame, EDL-YOLO is able to detect key objects of interest while the baseline YOLOv2 model fails to make the key detections. The model also demonstrates robustness for detecting small and distant objects and EDL-YOLO performs particularly well on cars and people. A positive result was shown in the image of two small birds. Our detector successfully classifies the larger bird with flapping wings as a bird with absolute certainty. The second smaller bird with stiff wings is missclassified as a plane, however the detection is accompanied with a higher uncertainty score. This is the desired behaviour of the model, since correct detections are reported with high certainty while erroneous predictions are acknowledged with degrees of uncertainty.

6.2 Future Work

An immediate extension of this work is uncertainty estimation of the bounding box regression and confidence scores. Utilizing regression uncertainty in the non-maximum suppression algorithm would likely improve box accuracy than using classification uncertainty. Additionally, a limitation of EDL is subjective logic estimates only the epistemic uncertainty. Incorporating estimates of aleatoric uncertainty may improve detection for noisy examples where objects are very distant or mostly occluded.

The motivating use case for this work is safety critical object detection systems, for example autonomous vehicles. Object detectors in vehicles trained to suit one environment such as motorways may struggle to detect well in hectic urban settings. Changes in environment will require out-of-distribution detection and hence the systems will need to be tuned. A big challenge when training machine learning models is obtaining suitable labelled data. The premise of active learning is to design a system able to learn using only a small amount of data and one which can choose which data it would like labelled by an oracle. Utilising active

learning results in a vast reduction in the amount of labelled data required since the system asks for only the most informative examples. Evidential Deep Learning may be used as the acquisition method by using epistemic uncertainty for out-of-distribution examples to determine what should be labelled. Thus utilising EDL and active learning in safety critical object detection should improve performance in new environments.

Appendix A

Appendix

A.1 KL Divergence Between Two Dirichlet Distributions

Here I derive the KL divergence between two Dirichlet distributions. First observing the log of a Dirichlet PDF splits up the multiplicative distribution into component parts as follows:

$$\begin{aligned}\log D(\mathbf{p}|\boldsymbol{\alpha}) &= \log\Big(\frac{1}{B(\boldsymbol{\alpha})}\prod_{k=1}^{K} p_k^{\alpha_k-1}\Big) \\ &= \log\Gamma(\alpha_0) - \sum_{k=1}^{K}\log\Gamma(\alpha_k) + \sum_{k=1}^{K}(\alpha_k-1)\log p_k\end{aligned}$$

Then we can explicitly derive the analytic form of the KL

$$\begin{aligned}\mathbf{KL}(D(\mathbf{p}|\boldsymbol{\alpha})||D(\mathbf{p}|\boldsymbol{\beta})) &= \mathbb{E}_{D(\mathbf{p}|\boldsymbol{\alpha})}\Big[\log\frac{D(\mathbf{p}|\boldsymbol{\alpha})}{D(\mathbf{p}|\boldsymbol{\beta})}\Big] \\ &= \big\langle \log D(\mathbf{p}|\boldsymbol{\alpha})) - \log D(\mathbf{p}|\boldsymbol{\beta})\big\rangle_{D(\mathbf{p}|\boldsymbol{\alpha})} \\ &= \Big\langle \log\Gamma(\alpha_0) - \sum_{k=1}^{K}\log\Gamma(\alpha_k) + \sum_{k=1}^{K}(\alpha_k-1)\log p_k \\ &\qquad - \log\Gamma(\beta_0) + \sum_{k=1}^{K}\log\Gamma(\beta_k) - \sum_{k=1}^{K}(\beta_k-1)\log p_k\Big\rangle_{D(\mathbf{p}|\boldsymbol{\alpha})} \\ &= \log\Gamma(\alpha_0) - \log\Gamma(\beta_0) + \sum_{k=1}^{K}\log\Gamma(\beta_k) - \sum_{k=1}^{K}\log\Gamma(\alpha_k) \\ &\qquad + \sum_{k=1}^{K}(\alpha_k-\beta_k)\big\langle \log p_k\big\rangle_{D(\mathbf{p}|\boldsymbol{\alpha})} \\ &= \log\Gamma(\alpha_0) - \log\Gamma(\beta_0) \\ &\qquad + \sum_{k=1}^{K}\log\Gamma(\beta_k) - \sum_{k=1}^{K}\log\Gamma(\alpha_k) \\ &\qquad + \sum_{k=1}^{K}(\alpha_k-\beta_k)\Big[\psi(\alpha_k) - \psi(\alpha_0))\Big]\end{aligned}$$

A.2 Mean Average Precision

A popular metric for measuring the accuracy of object detectors is mAP - mean Average Precision. To explain the methodology some basic definitions first need to be established. The role of a succesful object detector is to identify what objects are in an image and also where the objects are located. The task of localization is usually achieved by placing a bounding box around each detected object and the classifcation task assigns a label to the object. The intersection over union (IOU) is a measure to evaluate the overlap between bounding boxes. If the IOU between the predicted bounding box and the ground truth bounding box exceeds a threshold, then the detection is determined valid. Using notation from the original PASCAL VOC paper, denoting the ground truth bounding box B_{gt} and the prediction B_p, then the IOU is the fraction of the overlap area to the union of areas

$$\text{IOU} = \frac{\text{area}(B_p \cap B_{gt})}{\text{area}(B_p \cup B_{gt})} \tag{A.1}$$

Once the IOU of a detection has been calculated it can be labelled. A detection is a true positive (TP) or a correct detection if the IOU is greater than or equal to a given threshold. This threshold is usually 0.50 with a higher threshold corresponding to a better overlap with the true bounding box. If the IOU falls below the threshold the detection is denoted incorrect and a false positive (FP). If no ground truth exists then the detection is a false negative (FN) while a true negative is ignored for object detection since we can not have a correct misdetection.

With the above notions the precision and recall of detections can be determined. The precision of a model represents how well the model identifies relevant objects. The precision measures how accurate detections made are. Precision is given by the fraction of valid detections among all detections made. The closer a models precision is to 1.0 the more likely it is a positive detection is a relevant one. Recall on the other hand represents how well the model identifies relevant occurrences. Recall is given by the fraction of valid detections to all possible ground truths for the image. The closer a models recall is to 1.0 the more likely all objects that can be detected will be positively detected by the model. There of course comes a trade off between precision and recall and an inverse relationship exists between the two. A sign of a good detector is if precision can remain high as the recall increases.

$$\text{Precision} = \frac{\mathbf{TP}}{\mathbf{TP} + \mathbf{FP}} \text{ , Recall} = \frac{\mathbf{TP}}{\mathbf{TP} + \mathbf{FN}} \tag{A.2}$$

The precision vs. recall curve is commonly used as a way to evaluate object detectors by plotting a curve per object class. Object detectors can then be compared to each other by evaluating the area under the curve of the precision vs recall curve to provide an average precision (AP) score. Different implementations exist however ranging from 11 point interpolation calculates the precision $\rho(r)$ across 11 equally spaced recall values r in the $[0, 1]$ range.

$$\text{AP} = \frac{1}{11} \sum_{r} \rho_{\text{interp}}(r)$$

with

$$\rho_{\text{interp}}(r) = \max_{\tilde{r} \geq r} \rho(\tilde{r})$$

The mAP (mean average predcision) is then given by averaging AP values over all classes. This project implemented mAP estimation following: https://github.com/rafaelpadilla/Object-Detection-Metrics.

Bibliography

[1] *A Tragic Loss*. https://www.tesla.com/en_GB/blog/tragic-loss.

[2] Daniel Bauer, Lars Kuhnert, and Lutz Eckstein. "Deep, spatially coherent Inverse Sensor Models with Uncertainty Incorporation using the evidential Framework". In: *ArXiv* abs/1904.00842 (2019).

[3] Abhijit Bendale and Terrance Boult. "Towards Open World Recognition". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015.

[4] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. "Variational Inference: A Review for Statisticians". In: *Journal of the American Statistical Association* 112.518 (2017), pp. 859–877. DOI: 10.1080/01621459.2017.1285773. eprint: https://doi.org/10.1080/01621459.2017.1285773. URL: https://doi.org/10.1080/01621459.2017.1285773.

[5] Tiago Carneiro et al. "Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications". In: *IEEE Access* 6 (2018), pp. 61677–61685.

[6] François Chollet et al. *Keras*. https://keras.io. 2015.

[7] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[8] Mark Everingham et al. "The pascal visual object classes (voc) challenge". In: *International journal of computer vision* 88.2 (2010), pp. 303–338.

[9] Di Feng, Lars Rosenbaum, and Klaus Dietmayer. "Towards Safe Autonomous Driving: Capture Uncertainty in the Deep Neural Network For Lidar 3D Vehicle Detection". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)* (Nov. 2018). DOI: 10.1109/itsc.2018.8569814. URL: http://dx.doi.org/10.1109/itsc.2018.8569814.

[10] Yarin Gal. "Uncertainty in Deep Learning". PhD thesis. University of Cambridge, 2016.

[11] Yarin Gal and Zoubin Ghahramani. "Dropout As a Bayesian Approximation: Representing Model Uncertainty in Deep Learning". In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. ICML'16. New York, NY, USA: JMLR.org, 2016, pp. 1050–1059. URL: http://dl.acm.org/citation.cfm?id=3045390.3045502.

[12] Ross Girshick. "Fast R-CNN object detection with Caffe". In: *Microsoft Research* (2015).

[13] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[14] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010, pp. 249–256.

[15] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples". In: *arXiv preprint arXiv:1412.6572* (2014).

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[17] K He et al. *September. Spatial pyramid pooling in deep convolutional networks for visual recognition. In european conference on computer vision (pp. 346-361)*. 2014.

[18] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. "Distilling the Knowledge in a Neural Network". In: *NIPS Deep Learning and Representation Learning Workshop*. 2015. URL: http://arxiv.org/abs/1503.02531.

[19] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167* (2015).

[20] Audun Jøsang. *Subjective Logic: A Formalism for Reasoning Under Uncertainty*. 1st. Springer Publishing Company, Incorporated, 2016. ISBN: 3319423355, 9783319423357.

[21] Alex Kendall and Yarin Gal. "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" In: *CoRR* abs/1703.04977 (2017). arXiv: 1703.04977. URL: http://arxiv.org/abs/1703.04977.

[22] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 1097–1105. URL: http://dl.acm.org/citation.cfm?id=2999134.2999257.

[24] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. "Simple and scalable predictive uncertainty estimation using deep ensembles". In: *Advances in Neural Information Processing Systems*. 2017, pp. 6402–6413.

[25] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), p. 436.

[26] Yann LeCun et al. "Handwritten Digit Recognition with a Back-Propagation Network". In: *Advances in Neural Information Processing Systems 2*. Ed. by D. S. Touretzky. Morgan-Kaufmann, 1990, pp. 396–404. URL: http://papers.nips.cc/paper/293-handwritten-digit-recognition-with-a-back-propagation-network.pdf.

[27] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *CoRR* abs/1405.0312 (2014). arXiv: 1405.0312. URL: http://arxiv.org/abs/1405.0312.

[28] Wei Liu et al. "SSD: Single Shot MultiBox Detector". In: To appear. 2016. URL: http://arxiv.org/abs/1512.02325.

[29] David J.C. MacKay. "Bayesian methods for adaptive models". PhD thesis. California Institute of Technology, 1992.

[30] Andrey Malinin and Mark Gales. "Predictive Uncertainty Estimation via Prior Networks". In: *arXiv e-prints*, arXiv:1802.10501 (Feb. 2018), arXiv:1802.10501. arXiv: 1802.10501 [stat.ML].

[31] Andrey Malinin and Mark Gales. "Reverse KL-Divergence Training of Prior Networks: Improved Uncertainty and Adversarial Robustness". In: *arXiv preprint arXiv:1905.13472* (2019).

[32] Martın Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[33] Thomas P. Minka. *Estimating a Dirichlet distribution*. Tech. rep. 2000.

[34] R. M. Neal. "Bayesian Learning for Neural Networks". PhD thesis. Dept. of Computer Science, University of Toronto, 1994.

[35] Joseph Redmon and Ali Farhadi. "YOLO9000: better, faster, stronger". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7263–7271.

[36] J. Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91.

[37] Shaoqing Ren et al. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[38] Murat Sensoy, Melih Kandemir, and Lance M. Kaplan. “Evidential Deep Learning to Quantify Classification Uncertainty”. In: *CoRR* abs/1806.01768 (2018). arXiv: 1806.01768. URL: http://arxiv.org/abs/1806.01768.

[39] L. Smith and Y. Gal. “Understanding Measures of Uncertainty for Adversarial Example Detection”. In: *UAI*. 2018.

[40] Nitish Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[41] Niko Sünderhauf et al. “The limits and potentials of deep learning for robotics”. In: *The International Journal of Robotics Research* 37.4-5 (2018), pp. 405–420. DOI: 10.1177/0278364918770733.

[42] Christian Szegedy et al. “Intriguing properties of neural networks”. In: *arXiv preprint arXiv:1312.6199* (2013).

[43] Michael Truong Le et al. “Uncertainty Estimation for Deep Neural Object Detectors in Safety-Critical Applications”. In: Nov. 2018, pp. 3873–3878. DOI: 10.1109/ITSC.2018.8569637.

[44] Stephen Tu. “The dirichlet-multinomial and dirichlet-categorical models for bayesian inference”. In: *Computer Science Division, UC Berkeley* (2014).

[45] Zhong-Qiu Zhao et al. “Object detection with deep learning: A review”. In: *IEEE transactions on neural networks and learning systems* (2019).