Interpret the Evidential Deep Learning

Shengtong Jiang[1]

MSc Robotics and Computation

Simon Julier

Amy Widdicombe

Submission date: 26 September 2022

[1]**Disclaimer:** This report is submitted as part requirement for the MY DEGREE at UCL. It is substantially the result of my own work except where explicitly indicated in the text. *Either:* The report may be freely copied and distributed provided the source is explicitly acknowledged
Or:
The report will be distributed to the internal and external examiners, but thereafter may not be copied or distributed except with permission from the author.

Abstract

As machine learning algorithms have evolved, they have been used more often to solve real-life problems. People are starting to focus on why algorithms get things wrong and how we should avoid them. This poses two challenges. The first challenge is that people want to know how confident the models are in their predictions. This challenge is addressed by a technique called Evidential Deep Learning, which tells us about the uncertainty of the predictions. The second challenge is whether the network uses what we think is the correct data to make the decision. For example, when a model performs an image recognition task, is it looking at the object to be recognized or somewhere else, such as the background of the image? Interpretability techniques were developed to address this challenge. With the help of interpretability techniques, we can know which part of the image has the most significant impact on the prediction, in other words, which part of the image the model is looking at. However, interpretable techniques were developed for conventional networks, and we wanted to know how these techniques explain Evidential deep learning. In this project, we use a widely used interpretability technique, smoothGrad, to explore the properties of Evidential Deep Learning. We qualitatively analyzed smoothGrad maps generated by Evidential Deep Learning and conventional networks and found some differences. We then turned to a secondary network for quantitative analysis, which unfortunately failed to distinguish between high- and low-uncertainty maps. This may be due to the fact that we used an immature EDL or an unsuitable interpretability technique in this project. After that, we conducted a third experiment to find which part of the image had the most significant effect on the EDL uncertainty.

Contents

1 Introduction **2**

1.1 Background and motivation 2

1.2 Aims 3

1.3 Objectives 4

2 Literature Review **5**

2.1 Evidential Deep Learning 5

2.1.1 Theory of Evidential Deep Learning 5

2.1.2 Why Dirichlet 7

2.1.3 Loss functions 8

2.1.4 Perfomance on MNIST dataset 9

2.2 Saliency map 10

2.2.1 Why we care about interpretability 11

2.2.2 What is saliency map 12

2.2.3 Vanilla Gradient and SmoothGrad 13

2.2.4 Sanity Check and Why We Choose SmoothGrad 14

2.2.5 Summary 15

3 Technology selection **17**

3.1 Dataset 17

3.2 Model selection 17

3.3 Other settings 18

4 Interpreting EDL **20**

4.1 Experiment setup 20

4.2 Results 21

5 Prediction of Uncertainty from Saliency Maps **22**

5.1 Experiment setup 22

5.2 Results 23

6 Saliency maps of things that affect uncertainty **25**

6.1 Experiment setup 25

6.2 Results 25

7 Conclusion and Future work **27**

7.1 Summary 27

7.2 Future work 27

References **29**

A Other appendices, e.g. code listing **31**

Chapter 1

Introduction

In this chapter, a brief introduction to the project will be given. We will discuss the background and motivation for this project.

1.1 Background and motivation

Over the years, we have witnessed many discoveries in the architecture of neural networks, which have led to significant improvements in the accuracy of neural network predictions. As machine learning algorithms evolve, people are starting to use them in real systems, such as robots performing tasks and autonomous driving. Decisions in them can have a significant impact. An ML algorithm with good decision-making ability can bring much convenience to the work, but some malicious decisions can bring severe loss of life and property. Therefore, we need a way to see if the decisions of these machine learning models are reasonable and how confident the models are in their predictions, where interpretability and uncertainty are introduced.

The confidence or the uncertainty of a prediction can be further interpreted as how much the model believes in the model's output, for example, whether it is a random guess or whether the model is fairly confident in the outcome. A simple example can be given to illustrate the importance of knowing the confidence of a prediction. Let's say we feed a picture of a boat to a model that is trained to classify between cats and dogs. This model will likely output a random guess, such as a half-and-half likelihood output. What if we input a picture that contains both a cat and a dog[13]? It, too, would probably output a half-and-half likelihood output. However, we cannot tell the difference between these results by only looking at the result coming from the Softmax layer. In practice, we do not want to have inputs that the network has never seen before, such as feeding images of boats into a cat and dog model. To avoid these situations, we need to know the uncertainty of the predictions, and we should be extra careful when high uncertainty predictions are made. Otherwise, it could lead to some devastating situations. Under the robotics scope, a robot may encounter unpredictable situations, for example, where it is faced with an environment that it has never seen before, and therefore it will not be confident about the safety of its actions in that state, then the robot should avoid such situations.

Fortunately, the problem of uncertainty in network prediction was first addressed by the Bayesian neural network in 1992, which uses distributions as weights rather than deterministic weights. By using certain sampling techniques, we can treat the variance of the results as uncertainty. The larger the variance, the greater the uncertainty, and vice versa. However, in this project, we are looking at another newer technique, Evidential Deep Learning(EDL)[8], published in 2018, which can also model uncertainty in prediction, but claims to have better performance. In addition, EDL can assert that the network has not seen the input picture before and therefore assign it to the "uncertain" class. In contrast, a Bayesian neural network will still try to assign the input to one of the existing classes even if it finds it difficult to distinguish what precisely the input picture is. EDL can calculate the corresponding evidence for each class, and the larger the overall evidence, the lower the uncertainty in prediction. There has been little previous work examining how EDLs interpret input images to make EDLs aware of the uncertainty of their prediction, and we will focus primarily on this topic in this project. In this project, we will attempt to interpret the evidence from EDL in the feature space. An interpretability technique called saliency map will be used, by which we can investigate which part of the image has the more significant impact on uncertainty. We will also study if saliency maps contain evidence of how uncertain a network is about its prediction. Further explanation of EDL and saliency maps and other techniques will be provided in the literature review chapter.

1.2 Aims

In a further assumption, we hypothesise that only some specific pattern in the input image helps to predict a particular class and gives uncertainty. However, not much research has been done on what precisely that specific pattern could be. In this project, the aim is to investigate whether the saliency maps can provide insights into how EDL performs and whether the EDL uses those specific patterns inside the image that contribute to the uncertainty of the prediction. To achieve this, we used an interpretability tool called the saliency map. The saliency map tells us where the model pays more attention to the image, and with the help of the saliency map, we can interpret the model's predictions. We also want to know if the saliency maps from the EDL network contain more information than the maps from the regular network. If so, whether this extra information is the specific pattern we are looking for?

1.3 Objectives

In this project, an EDL model will first be trained on the CIFAR-10 dataset, and this model will then be used to generate a series of saliency maps from the original dataset. These generated saliency maps will be grouped into two different categories based on the uncertainty of the predictions for these images. A secondary model will then be trained to classify between these two categories of the saliency map, and if we obtain an accuracy higher than a random guess, i.e. 50%, then we can conclude that these saliency maps contain some information that depends on the uncertainty. In other words, the evidence from the EDL model can address the uncertainty within the image, and then our next step is to find which part of the image has the most significant impact on the uncertainty of the prediction.

The aim of this project is to interpret the EDL network using saliency map, and investigate if the uncertainty information is contained in the saliency map. In this project, the following objectives will be fulfilled in the following chapters:

- In chapter 2, we briefly described what EDL and saliency maps are and why we chose them for our research.
- In chapter 3, we described the technology that will be used in this project.
- In chapter 4, we implemented an EDL network and trained it on the CIFAR-10 dataset. After that, we analyze its significance maps.
- In chapter 5, we used the trained EDL network to generate the saliency map dataset and tested whether the secondary network can learn uncertainty information from the saliency map dataset.
- In chapter 6, we tried to find out which part of the image has the greatest impact on uncertainty.
- In chapter 7, we made the conclusion of this project and discussed what can be done as future work.

Chapter 2

Literature Review

In this chapter, a brief background on the used technique is presented. First, we start with Evidential Deep Learning[8]. Then, we will talk about the interpretability tools that are used to explain the network, also known as saliency maps.

2.1 Evidential Deep Learning

In this project, we will focus on analysing the behaviour of Evidential Deep Learning. The EDL was first published by Murat Sensoy and Lance Kaplan in 2018. The EDL aims to measure the uncertainty of the network's predictions.

2.1.1 Theory of Evidential Deep Learning

EDL has been demonstrated in many datasets for its performance and accuracy. The structure of the model does not differ significantly from that of a standard neural network, however, in the final layer, softmax activation is no longer used because we do not want a deterministic probability as a prediction; instead, we model the output of this network as a probability distribution and believe it is subject to a Dirichlet distribution. Thus, the output from the last layer can be considered as Dirichlet parameters. —In addition, a RELU activation will be used to prevent negative outputs as we do not want negative evidence, where the negative evidence does not make sense in this scenario.– Following our formula described above, we can easily calculate the probability of each category and the uncertainty for all classes. In EDL, the Subjective Logic (SL) is introduced to measure uncertainty. SL is a formalization of the Dempster-Shafer theory of evidence on Dirichlet Distribution[7]. The Dempster–Shafer Theory of Evidence, also known as DST, is a special case of Bayesian theory with random sets. It assigns belief masses to subsets of a frame of discernment, which denotes the set of exclusive possible states, e.g., possible K class labels in the dataset, and each class is assigned a belief mass b_k during the inference, K belief masses and an additional uncertainty term sum up to be 1.

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

The belief mass for each class can be thought of as the probability that the network thinks the input will be classified in that class. This belief mass appears to be much like the probability coming out from the softmax layer. However, this additional uncertainty term introduced is interpreted as the degree of confidence the model has in its predictions. The uncertainty term is not generated directly by the network; instead, the network generates the evidence for each class, and the uncertainty is then calculated from the evidence.

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

Where S is the sum of 1 plus evidence is given by,

$$S = \sum_{i=1}^{K} (e_i + 1)$$

To be consistent with the context of subjective logic (SL), in this network, we can also use non-negative Dirichlet parameters to represent the evidence, which means adding 1 to the evidence and the relationship between the Dirichlet parameters and the evidence is given by.

$$\alpha_i = e_i + 1$$

With the help of a Dirichlet distribution parameterised by the evidence, we can generate the probability and uncertainty of each prediction. The uncertainty is calculated by dividing the number of classes by the total number of pieces of evidence. This means that the more evidence the model has for this input, the more confident the model is of the prediction and, therefore, the less uncertainty (u) in this prediction.

In theory, we want the evidence to be representative of the number of times the model sees this type of image, in other words, "we term evidence as a measure of the amount of support collected from data in favor of a sample to be classified into a certain class."[8] To do so, we assume that when an observation on a sample is related to one of the classes, the corresponding Dirichlet parameter is incremented. Taking the CIFAR-10 dataset as an example, each time the model observes a car image, the Dirichlet parameter of the car class will be incremented, so the more car images the model sees from the training dataset, the larger the Dirichlet corresponding to the car class will be, which means that the model has more confidence in this class. When the trained model sees a new car image, it will be very sure that the image is a car because the Dirichlet parameter of the car class is so large that the overall uncertainty is reduced to a small value.

2.1.2 Why Dirichlet

In this project, EDL is discussed under the classification problem setting. In the classification problem, the output will be a K-class label, usually represented in a one-hot encoding format. This output is assumed to be sampled from the Categorical distribution. To model second-order probability and uncertainty, we further assume that the parameters of the above Categorical distribution are sampled from the Dirichlet distribution.

The Dirichlet distribution was chosen as the prior because it forms a conjugate prior on the likelihood function, i.e. the categorical distribution, and is, therefore, easier to compute. The Dirichlet parameter (α) is learned directly by the network during training. According to the Dempster-Shafer theory that is described above, α can be approximately regarded as evidence. Those Dirichlet parameters obtained from the network will be summed to the Dirichlet strength. The higher the Dirichlet strength, the lower the prediction uncertainty is considered to be.

$$y \in \{1, ..., K\}$$

$$y \sim Categorical(p)$$

$$p \sim Dirichlet(\alpha)$$

Intuitively, Dirichlet strength can be thought of as the network's confidence in its predictions or the number of times the network observes similar images during training. It is why we expect the corresponding Dirichlet parameter to increment when some specific pattern is observed during training.

In order to demonstrate EDL's theory more clearly, we need to clarify the relationship between evidence, belief quality and the Dirichlet parameters. EDL assumes that there is evidence in the input images that tells EDL what to predict. The more evidence there is about a particular class, the more the network believes that the image belongs to that class. The more total evidence there is for all of the classes, the less uncertainty EDL has about its predictions. The mathematical background for EDL is Dempster-Shafer's theory of evidence, which assigns belief masses to each class, and we can think it of as amount of evidence in each class. These evidence are computed by the network through forward propagation and are represented by the Dirichlet parameters.

2.1.3 Loss functions

In the original paper, three loss functions were introduced. Before introducing the equations of loss functions, we need to formalize the notation. We use $\mathcal{L}_i(\theta)$ to represent the loss for i_{th} sample, and the θ is the parameter of the network. The y_i is the one-hot vector for i_{th} sample's label. The y_{ij} is the j_{th} element in the one-hot vector y_i. α is the parameter of Dirichlet distribution, α_i is the Dirichlet parameter generated by the EDL network for i_{th} sample, and α_{ij} is the j_{th} element in the Dirichlet parameter vector, corresponding to the j_{th} class out of K classes. The S is the total evidence for all of the classes.

The first loss function comes from the idea of maximum likelihood, where we have:

$$\mathcal{L}_i(\theta) = \sum_{j=1}^{K} y_{ij}(\log(S_i) - \log(\alpha_{ij}))$$

In the second loss function, the Bayes risk with respect to the class predictor is computed. It is described as:

$$\mathcal{L}_i(\theta) = \sum_{j=1}^{K} y_{ij}(\psi(S_i) - \psi(\alpha_{ij}))$$

Where the ψ is the diagamma function.
The third loss function uses the sum of square loss, described as:

$$\mathcal{L}_i(\theta) = \sum_{j=1}^{K}(y_{ij}^2 - 2y_{ij}\mathbb{E}[p_{ij}] + \mathbb{E}[p_{ij}^2])$$

Where p_{ij} is the class assignment probabilities[8]. The third loss function performs better in most implementations of EDL because, according to the original EDL paper[8], it was found empirically to have more consistent performance. In contrast, the first two loss functions tend to produce excessive belief mass. The third loss function also proved to optimise the network and remove misleading evidence. On this basis, the variance of the predictions was also reduced. In order to reduce the evidence of misclassified samples, KL divergence is introduced as a regulariser to improve the loss function. KL divergence penalises the misleading evidence, resulting in better performance. The loss function with a KL divergence term is:

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \mathcal{L}_i(\theta) + \lambda_t \sum_{i=1}^{N} KL[D(p_i|\tilde{\alpha}_i)||D(p_i|1, ..., 1)]$$

Where $\tilde{\alpha}_i = y_i + (1 - y) \odot \alpha_i$. λ_t is a coefficient that grows over the epoch during the training, so in the early epochs, KL divergence is limited in regularising misclassified samples, and by doing so, the model is able to explore the space in the early epoch and be regularised to fit the data in the later epoch to achieve better accuracy. The KL divergence is given by:

$$KL[D\left(p_i|\tilde{\alpha}_i\right)||D\left(p_i|1, ..., 1\right)] = \log\left(\frac{\Gamma\left(\sum_{k=1}^{K} \tilde{\alpha}_{ik}\right)}{\Gamma\left(K\right)\prod_{k=1}^{K} \Gamma\left(\tilde{\alpha}_{ik}\right)}\right) + \sum_{k=1}^{K} \left(\tilde{\alpha}_{ik} - 1\right) \left[\psi\left(\tilde{\alpha}_{ik}\right) - \psi\left(\sum_{j=1}^{K} \tilde{\alpha}_{ij}\right)\right]$$

Where the Γ is the gamma function.

2.1.4 Perfomance on MNIST dataset

In the original EDL paper, the performance of EDL was first tested on the MNIST dataset[17], a dataset of 70,000 handwritten digital images. It was created by "re-mixing" the samples from NIST's original datasets[15][1]. A few examples of this dataset are shown in fig2.1.

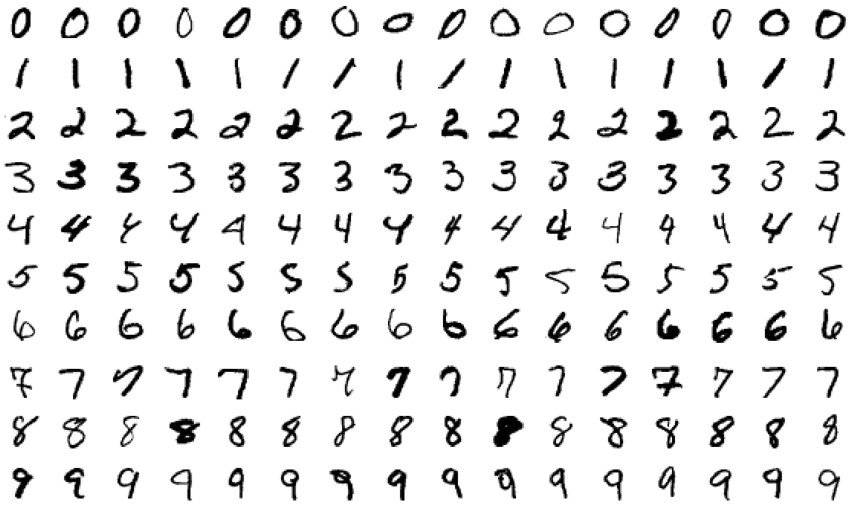
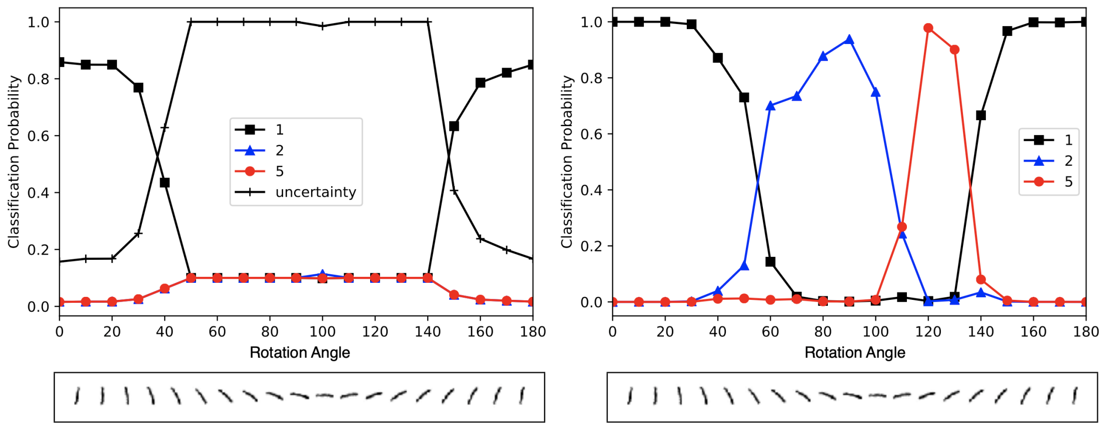


Figure 2.1: Examples of MNIST [15]

The MNIST dataset is an excellent toy example of testing a network. Besides, it is particularly easy to create an "out-of-distribution" input for the EDL network to test examples with low uncertainty. The EDL network was first trained on the MNIST dataset and then tested on some manually rotated digits. As the network never saw these rotated digits, we expected the uncertainty to be high. Fig 2.2 shows the results of testing the EDL network on some rotated images of digit

1. These images of digit 1 were rotated from 0 degrees to 180 degrees. As can be seen from fig 2.2a, the prediction uncertainty starts low between 0 and 30 degrees, as even though it is a little skewed, it is still considered 1, and the model predicts correctly. However, when the number 1 is rotated between 50 and 140 degrees, the uncertainty rises rapidly because the input becomes unrecognisable, and the model has not seen it in the training set before. Yet even if the network now incorrectly predicts the input as 2 or 5, we can know that the model is less certain about the prediction by looking at the high uncertainty and low probability of the prediction. When the number 1 is rotated by 140 degrees, the uncertainty drops again because digit 1 is symmetrical, and the image of the input starts to look like 1 to the model.



(a) Rotating digit 1 feed in to EDL network

(b) Rotating digit 1 feed in to regular network

Figure 2.2: EDL network and regualr network predicting rotated image of digit 1[8]

Predictions between 40 and 140 degrees are dreadful if a regular network is used, as shown in the fig 2.2b, where the probability of the input being classified as a digit 2 or digit 5 is high, but we have no way of knowing how uncertain the model is about the prediction. As a result, the user may incorrectly assume that 2 or 5 is an acceptable prediction because the probability of these false predictions is very high, as shown in the fig 2.2b, at 120 degree the model believes a 100% probability that this rotated digit 1 is actually a 5.

2.2 Saliency map

As mentioned above, the EDL model assumes that when some specific patterns in an image are observed during the training, the corresponding Dirichlet parameter is incremented. In other words, certain parts of the image affect the uncertainty of the prediction. In this project, we will discuss what is the pattern in the input image that has the most significant impact on uncertainty.

2.2.1 Why we care about interpretability

The fact that a deep learning network is a large and complex non-linear function makes it a black box for human users. Model designers often wonder why the network gives such predictions because sometimes the designer is concerned about the features the model predicts based on. These features are often related to security or ethical issues. More specifically, we care about the underlying logic behind model predictions for the following reasons:

1. The first is due to safety considerations, when machine learning has an increasing number of practical uses in everyday life, some of which involve situations where safety is very much a concern, such as autonomous driving. Models cannot be trained on datasets containing all possible scenarios. Moreover, it is almost impossible for the system to be fully testable. Moreover, failure in such cases can result in serious personal injury. For example, autonomous cars can be trained with a dataset with an enormous amount of data, but no single dataset can contain all possible scenarios. There is no way to guarantee that a autonomous car will not fail and cause an accident, and if an accident occurs, looking at the test cases will not help to find out why the car made the decision that caused the accident. Therefore, interpreting the model's decisions can help one to identify problems in advance.

2. The second is related to ethical issues. Today, ethical issues in the field of machine learning are widely discussed. The development of machine learning algorithms has greatly improved the accuracy of predictions. The fundamental problem is that the network is trained on data. Therefore, it will model and amplify the biases in the dataset that we want to eliminate. In other word, predictions may be based on unethical biases in the data. Humans may want to guard against certain kinds of discrimination, and their notion of fairness may be too abstract to be completely encoded into the system[4]. The network may predict based on features that discriminate the race or gender. Even if the data might be preprocessed to eliminate the biased feature, there may also be indirectly biased features, which can be difficult to find at the outset. (e.g., one may not build gender-biased word embeddings on purpose, but it was a pattern in data that became apparent only after the fact)[4]. For example, when a model is trained on a dataset containing personal information, height and weight data may indirectly lead to gender-biased prediction.

In most cases, the use of interpretable tools is about gaining knowledge. We usually obtain explanations and knowledge by asking experts, who can now be trained machine learning models. Using interpretability tools is a way to get explanations from the network. In this project, we will ask for explanations from the EDL model [see above], which can generate uncertainty of prediction.

Specifically, we want to know which part of the input image tells the network the uncertainty of the prediction. We will use an interpretability tool called a saliency map on the EDL network, which generates a heat map showing which pixels receive more attention from the network.

2.2.2 What is saliency map

The interpretability tool we choose is called saliency map. A saliency map means a map that indicates salience or importance. The human brain has a built-in saliency map that helps to draw attention to their field of vision, thus avoiding potential dangers. Inspired by this mechanism, machine learning scientists have used the properties of saliency maps to help inform their models. In the context of machine learning, saliency maps reveal to us which part of the image, at that intensity, has the biggest (infinitesimal) impact on the network prediction. It is one of the most widely used interpretability techniques to date, helping machine learning scientists understand what features their networks are looking at and make predictions accordingly. Saliency methods have emerged as a popular tool to highlight features in an input deemed relevant for the prediction of a learned model[6].

The principle of the saliency map is to look at changes in prediction after perturbing pixels of the image. In other words, the saliency map tells us how the predicted probability of a class will change if the value of certain pixels is increased. If the probability decreases when the pixel value increases, then the corresponding value in the saliency map will be positive, and vice versa. The larger the value in the saliency map, the greater the effect on the prediction of changing the corresponding pixel in the image. Those pixels with larger values in the saliency map are considered to be the more important areas of the image. By looking at the saliency map, we can know which patterns have a greater impact on the predictions and are therefore considered more important regions. Here is an example of the saliency map for a dog in fig 2.3.

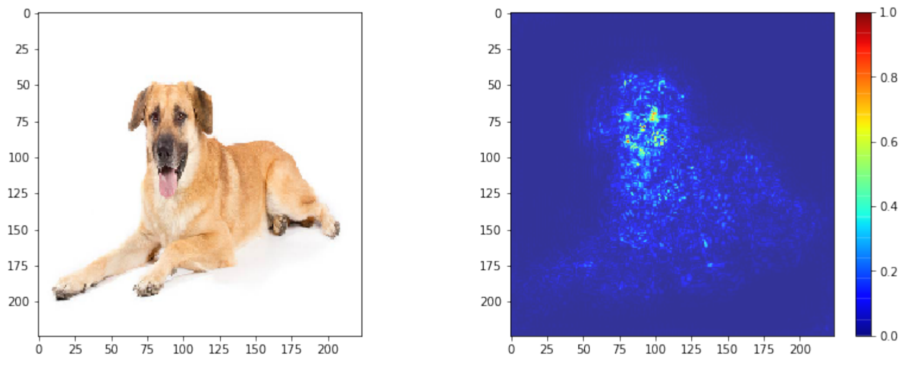


Figure 2.3: Example of dog saliency map using Gradient method. [11]

As we can see from the example in fig 2.3, the network pays more attention to the dog's face than the rest of the image. The dog's eyes and mouth are the key patterns in the input image, leading the network to classify the input image as a dog.Some may be wondering if dog tails and paws are also unique features of dogs that can be used to classify dogs, however not being highlighted in the saliency map. It is important to note that the saliency map can only represent the judgement criteria of the network, which is highly dependent on how the network is trained and may vary with factors such as network structure and hyperparameters.

2.2.3 Vanilla Gradient and SmoothGrad

The values in the saliency map are calculated in several ways, most of which are calculated using gradient methods. In this project, the SmoothGrad were used. However, before delving into SmoothGrad, we will first introduce its predecessor, the Vanilla Gradient method.

The Vanilla Gradient method is one of the most widely used varieties of saliency maps, and it is also considered to be the standard method. Its theory is fairly straightforward and simple. We only need to calculate the gradient of the probability of the prediction class with respect to the input. In the formal setup, the input is represented as a vector $x \in \mathbb{R}^d$. A model is represented as a function $S : \mathbb{R}^d \to \mathbb{R}^C$, where C is the number of class of the labels. Therefore, the prediction of the model S can be represented as $S(x)$. The saliency method is represented by the mapping $E : \mathbb{R}^d \to \mathbb{R}^d$[6]. The Vanilla Gradient quantifies how much a change in each input dimension would a change the predictions $S(x)$ in a small neighborhood around the input[6]. Therefore, the equation[6] of Vanilla Gradient method can be described as:

$$E_{grad}(x) = \frac{\partial S}{\partial x}$$

SmoothGrad produces clearer saliency maps than Vanilla gradient does[5], and is therefore a better choice at rendering saliency maps in this project. smoothGrad can be seen as an extension of the Vanilla Gradient method. It is very similar to the Vanilla Gradient method in which the gradient of the probability of the predicted class with respect to the input is taken. However, before the input image is passed to the saliency function, multiple copies of the input image are generated, and noise is added to them. Multiple saliency maps are then computed, and the final saliency map is simply the average of those saliency maps. With the same notation as we used for Vanilla

Gradient, the equation[6] of SmoothGrad can be described as:

$$E_{sg} = \frac{1}{N}\sum_{i=1}^{N} E(x + g_i)$$

where E is the Vanilla Gradient mapping, and g_i is the noise sampled from normal distribution:

$$g_i \sim \mathcal{N}(0, \sigma^2)$$

2.2.4 Sanity Check and Why We Choose SmoothGrad

There have been lots of saliency methods proposed. However, some of them appear to be often misleading, and some of these interpretability methods are not actually explaining a model in the way we want them to. After an experiment called sanity checks[6] is designed, Some of the saliency methods are proved to be independent of both the model and the data generating process. Therefore, they are inadequate for tasks that are sensitive to either data or model[6], and are nothing but an edge detector. A brief description of the above experiments is listed below. The sanity check consists of two sub-experiments, one for the dependence of the saliency map on the model and the other for the dependence of the saliency map on the data.

The first experiment against the model compares the output of a saliency method on a trained model with the output of the saliency method on a randomly initialized untrained network of the same architecture[6]. If the saliency maps generated on the two models look similar, we can conclude that this saliency method is independent of the model properties. This means that if this saliency approach is used, we will not be able to study the EDL model properties. In fig 2.4, some saliency methods are tested with the experiment described above. Those saliency methods are tested with cascading randomized model parameters from the top layer to bottom layer in the network. We can observe that some saliency methods, such as Guided BackProp and Guided GradCAM, show no sensitivity to model degradation[6]. This means those two saliency methods are independent of the model properties.
The second experiment against data compares a given saliency method applied to a model trained on a labelled data set with the method applied to the same model architecture but trained on a copy of the data set in which we randomly permuted all labels[6]. If the saliency maps generated on these two model looks similar, we can conclude that this saliency method is independent of the data properties. The experiment is conducted on MNIST dataset for some saliency method using network with CNN architecture, as shown in fig 2.5. In fig 2.5, we can observe that the Gradient method and Gradient-SG method are extremely sensitive to data randomization. For the rest of

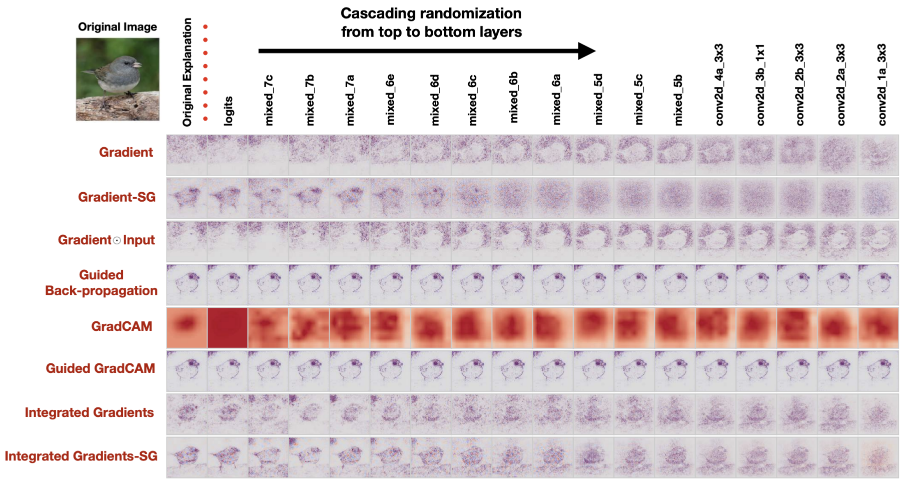


Figure 2.4: Cascading randomization [6]

the method, although we can see some tiny visual changes, the structure still remains after the label of the dataset is randomized. Therefore, we can conclude that only the Gradient methods and Gradient-SG method are capable of finding the properties of data.

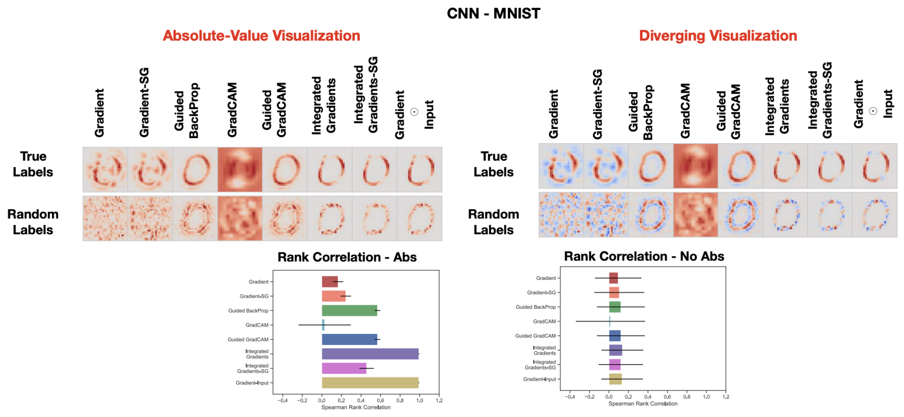


Figure 2.5: Label permutation [6]

The SmoothGrad we chose for this project passed both of these tests, so we can safely use it to analyse the model properties of the EDL.

2.2.5 Summary

This project will use the saliency map method with an EDL model trained on the CIFAR-10 dataset. We want to analyse which regions of the image the EDL model will pay particular attention to and whether it will differ from the standard deterministic model. Based on these saliency maps generated on the EDL network, we will conduct further experiments to discuss whether the saliency maps contain information about uncertainty. In other words, can we tell that an image has high uncertainty just by looking at its saliency maps? We will also try to find out if there are any specific patterns in the image that contribute to the uncertainty. This is different to the normal saliency map, which is generated by obtaining the gradient of the output (evidence). However, in this last experiment, we will take the derivative of the uncertainty of the prediction with respect to the input image.

Chapter 3

Technology selection

In this chapter, we will explain the techniques used in this project, such as the model's architecture. In addition, the datasets and other settings used will be explained.

3.1 Dataset

The dataset used in this project is the CIFAR-10 dataset[16]. It is a very widely used dataset in the field of computer vision. The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images[16]. The 10 classes are airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck, respectively. A few examples of the image in the CIFAR-10 dataset are shown in figxxx.

The main reason for choosing this dataset is that the images in this dataset are accurately annotated. This dataset is well proven in a wide range of applications in machine learning. Another advantage of the CIFAR-10 dataset is that the size of the images is relatively small, with only 32*32 pixels, so even with a complex network structure, the training process will still be quite fast and, therefore, easier for the researcher to debug. The only data pre-processing required is the rescale of the pixel values into a range of 0-1 and the conversion of the labels to a one-hot encoding format, no other feature engineering is required. In each epoch, 10% of the training set was split as the validation set.

3.2 Model selection

The primary network is the EDL network that we will mainly analyse. We did not choose those widely-used architectures, such as ResNet or VGG. On the contrary, to prevent overfitting that could potentially affect the experiment results, the architecture of this model is very simple, some convolutional layers and pooling layers were used. Dropout layers are also introduced to further prevent overfitting. The architecture of this model is borrowed from this website [10], and its structure is shown in fig 3.1. As described in fig 3.1, the network can achieve an accuracy of 86% on the CIFAR-10 dataset. As described in Chapter 2, the last layer of Softmax activation was replaced with ReLU activation to generate non-negative evidence.

The second network is almost identical in structure to the primary network. Except for the last layer, which uses standard Softmax activation to generate probabilities, and because it is trained on grey-scale saliency maps, the input size of the first layer is now 32×32×1 instead of 32×32×32 as in the primary network.

3.3 Other settings

During training, we use the standard Adam optimizer[2] as it provides faster computation time, and its parameter is easier to manipulate. As described in section 2.1.3, the loss function for the primary network is the KL divergence. The loss function for the second network is categorical_crossentropy.

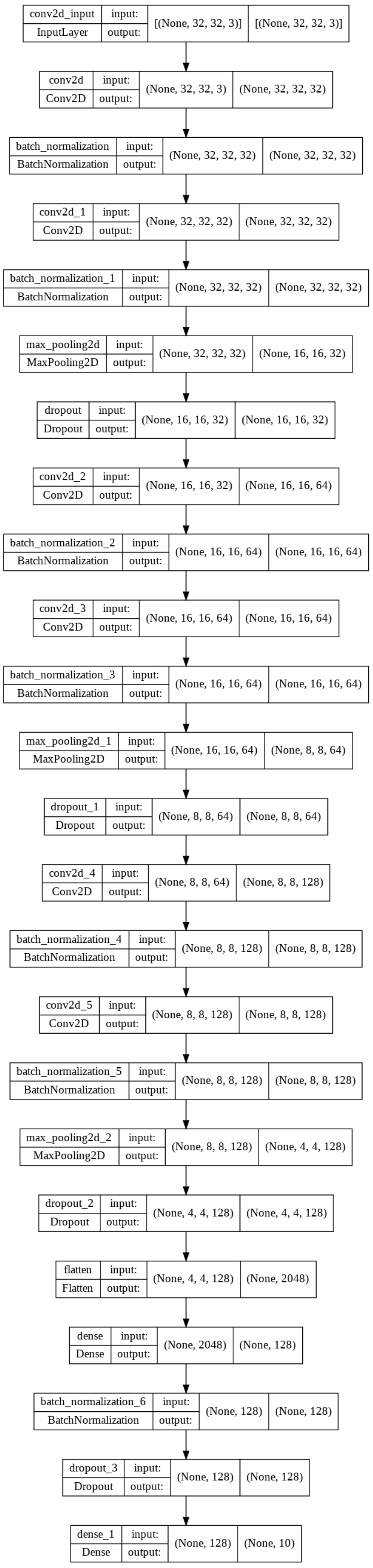

conv2d_input InputLayer input: output: [(None, 32, 32, 3)] [(None, 32, 32, 3)]
conv2d Conv2D input: output: (None, 32, 32, 3) (None, 32, 32, 32)
batch_normalization BatchNormalization input: output: (None, 32, 32, 32) (None, 32, 32, 32)
conv2d_1 Conv2D input: output: (None, 32, 32, 32) (None, 32, 32, 32)
batch_normalization_1 BatchNormalization input: output: (None, 32, 32, 32) (None, 32, 32, 32)
max_pooling2d MaxPooling2D input: output: (None, 32, 32, 32) (None, 16, 16, 32)
dropout Dropout input: output: (None, 16, 16, 32) (None, 16, 16, 32)
conv2d_2 Conv2D input: output: (None, 16, 16, 32) (None, 16, 16, 64)
batch_normalization_2 BatchNormalization input: output: (None, 16, 16, 64) (None, 16, 16, 64)
conv2d_3 Conv2D input: output: (None, 16, 16, 64) (None, 16, 16, 64)
batch_normalization_3 BatchNormalization input: output: (None, 16, 16, 64) (None, 16, 16, 64)
max_pooling2d_1 MaxPooling2D input: output: (None, 16, 16, 64) (None, 8, 8, 64)
dropout_1 Dropout input: output: (None, 8, 8, 64) (None, 8, 8, 64)
conv2d_4 Conv2D input: output: (None, 8, 8, 64) (None, 8, 8, 128)
batch_normalization_4 BatchNormalization input: output: (None, 8, 8, 128) (None, 8, 8, 128)
conv2d_5 Conv2D input: output: (None, 8, 8, 128) (None, 8, 8, 128)
batch_normalization_5 BatchNormalization input: output: (None, 8, 8, 128) (None, 8, 8, 128)
max_pooling2d_2 MaxPooling2D input: output: (None, 8, 8, 128) (None, 4, 4, 128)
dropout_2 Dropout input: output: (None, 4, 4, 128) (None, 4, 4, 128)
flatten Flatten input: output: (None, 4, 4, 128) (None, 2048)
dense Dense input: output: (None, 2048) (None, 128)
batch_normalization_6 BatchNormalization input: output: (None, 128) (None, 128)
dropout_3 Dropout input: output: (None, 128) (None, 128)
dense_1 Dense input: output: (None, 128) (None, 10)

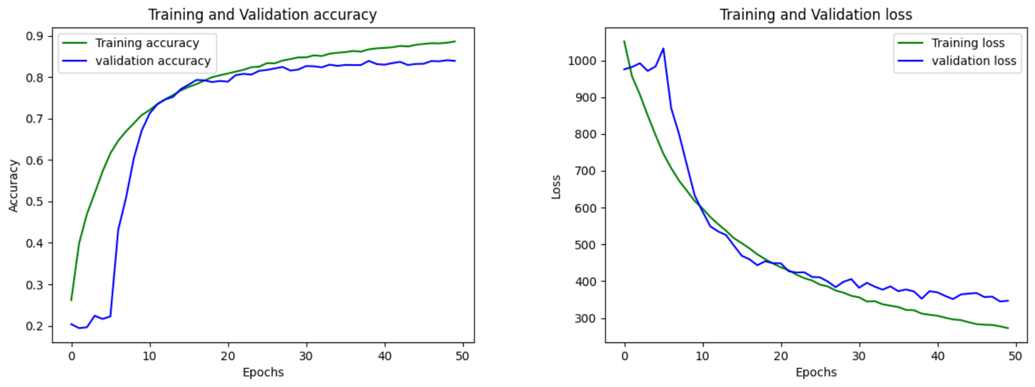
Figure 3.1: Architecture of two models

Chapter 4

Interpreting EDL

4.1 Experiment setup

In order to analyse the EDL, we need to train a network under the EDL setting using the CIFAR-10 dataset. After roughly 50 epochs, the accuracy of the network stabilises. On the CIFAR-10 dataset, the EDL network achieved an accuracy of 89.67% in the 50th epoch. The accuracy of this network over the epochs is shown in fig 4.1a, and the loss of this network over the epochs is shown in fig 4.1b. From this we can see that EDL performs very well on the CIFAR-10 dataset, even if we only look at accuracy.



(a) The training and validation accuracy of the EDL network on CIFAR-10 dataset

(b) The training and validation loss of the EDL network on CIFAR-10 dataset

Figure 4.1: The accuracy and loss of the EDL network on CIFAR-10 dataset

In the first experiment, we investigate which parts of the input images EDL pays more attention to. With the help of an interpretability framework called PAIR[18], we generate saliency maps from the EDL network for the entire dataset. We analysed these generated saliency maps and discuss how these saliency maps generated from EDL differ from those generated from the regular network.

4.2 Results

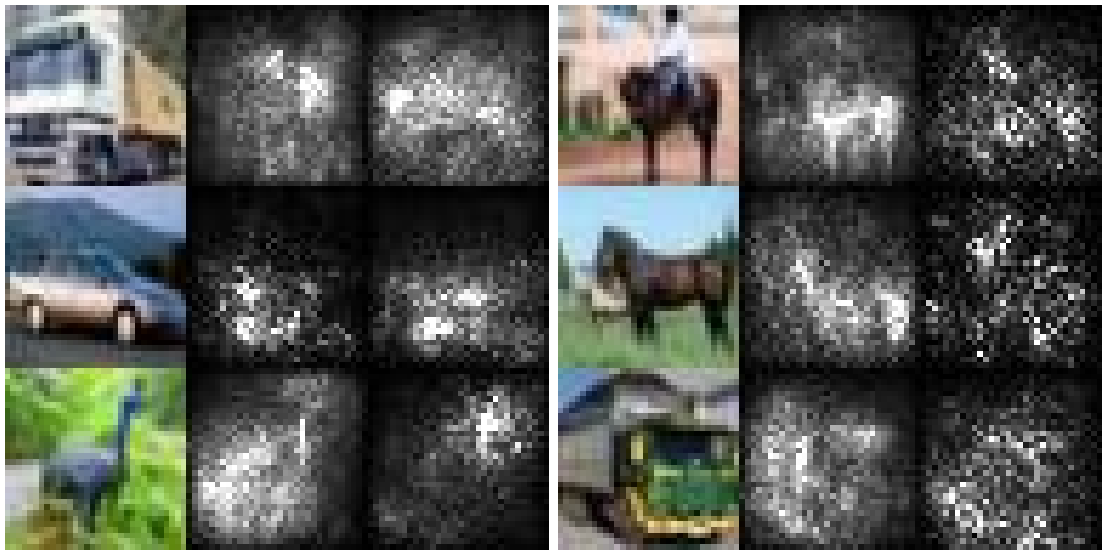


Figure 4.2: The smoothGrad saliency maps generated by EDL model and comparison model. The comparison model is the model with a similar architecture but has a conventional softmax layer. The first column is the original input images, the second column is the smoothGrad saliency maps generated by the EDL network, and the third column is the smoothGrad saliency maps generated by the comparison network. The labels of the original input image are truck, automobile, bird, horse, horse, and truck, respectively.

Some examples of smoothGrad saliency maps generated by the EDL model are shown in figure 4.2. As a comparison, we also added smoothGrad saliency maps generated by the comparison model to the side of the EDL-generated smoothGrad saliency map. The architecture of this comparison model is basically the same as that of the EDL model, except that the last layer uses Softmax activation. The first column of the plot in fig 4.2 is the original input image, the second column is the smoothGrad saliency map generated by the EDL model, and the third column is the smoothGrad saliency map generated by the comparison model. As shown in fig 4.2, we notice some differences between the saliency maps generated by the EDL model and those generated by the comparison model. For example, the highlighted part of the saliency maps generated by the EDL model is usually more concentrated. In contrast, the highlighted part of the saliency maps generated by the comparison model is more dispersed. Both types of saliency maps can describe the contours of the images. However, beyond that, we do not get more information from the naked eye. In other words, we cannot analyze them quantitatively. Therefore, the second experiment was conducted to quantify the analysis.

Chapter 5

Prediction of Uncertainty from Saliency Maps

5.1 Experiment setup

As in the previous chapter, after qualitatively analysing the saliency maps generated by EDL, we then discussed whether the saliency maps contained information about the uncertainty of the predictions. As described in the previous chapter, we conducted a second experiment to analyse the saliency maps generated by the EDL network quantitatively. Since it is difficult for the human eye to gather useful information from the saliency map, we must resort to a second network to help us to analyse the saliency maps. We let the second network learn the uncertainty information from the generated saliency maps. Before using the second network to learn the uncertainty information, we have to create the training dataset. Therefore, in the following experiments, we will divide the generated saliency maps into two classes based on their uncertainty in the inference process. The two classes of the map are the saliency maps for images with high uncertainty and those with low uncertainty during the prediction. After several trials, we decided to place those maps with uncertainty higher than 60% in the high uncertainty class and those lower than 40% in the low uncertainty class. Furthermore, all maps with uncertainty in between were cut out to make the two classes of images more distinguishable. In practice, we usually got more saliency maps with low uncertainty. To ensure a balanced data set, we must discard those redundant data to make it a balanced set. Otherwise, the model may always predict with the class that has a larger amount to achieve higher accuracy.

We will train a second network on this saliency map dataset. The second network has structure similar to EDL network, as described in section 3.2 If the second network ends up making better predictions than random guesses, that is, with an accuracy higher than 50%, we can conclude that we know that the map contains some (though not human-interpretable) representation of network uncertainty. Otherwise, the saliency map does not contain information about how the EDL network learns about the uncertainty of the prediction. The above experiment methodology is inspired by the paper Saliency Maps Contain Network "Fingerprints"[14] by Amy Widdicombe and Simon Julier.

It is worth noting that during training, we will only generate saliency maps from those images that are correctly predicted and then use it as the dataset in the second experiment. Otherwise, the secondary network might learn how to classify between maps with correct predictions and maps with incorrect predictions rather than classify between maps with high uncertainty and maps with low uncertainty. The secondary network might learn unwanted knowledge because there is a correlation between uncertainty and prediction accuracy, and the network might mistakenly treat the correlation as causality. For example, suppose a low-uncertainty prediction is more likely to be correct than a low-uncertainty prediction. In this case, what the network may actually learn is how to distinguish whether this is the correct prediction by looking at the saliency maps. By discarding those incorrectly predicted saliency maps, we control the unique variable in the experiment.

5.2 Results

A dataset with 6396 samples was generated in which there were an equal number of saliency maps with high uncertainty and low uncertainty. Some examples of saliency maps in the dataset is shown in fig 5.1.

Unfortunately, the training accuracy of the secondary network on this dataset fluctuated around 50%, as shown in fig 5.2. At the end of the 50th epoch, the training accuracy was 50.5%. The variation of accuracy and loss of secondary network with the epochs is shown in fig 5.2. As we can see from the fig 5.2a, both training accuracy and validation accuracy perform poorly. In other words, the model essentially made a random guess on the classification. This means that the saliency map generated by the high uncertainty prediction and the saliency map generated by the low uncertainty prediction are indistinguishable for the secondary network. That is, the secondary network cannot find any information about the uncertainty from the saliency map.

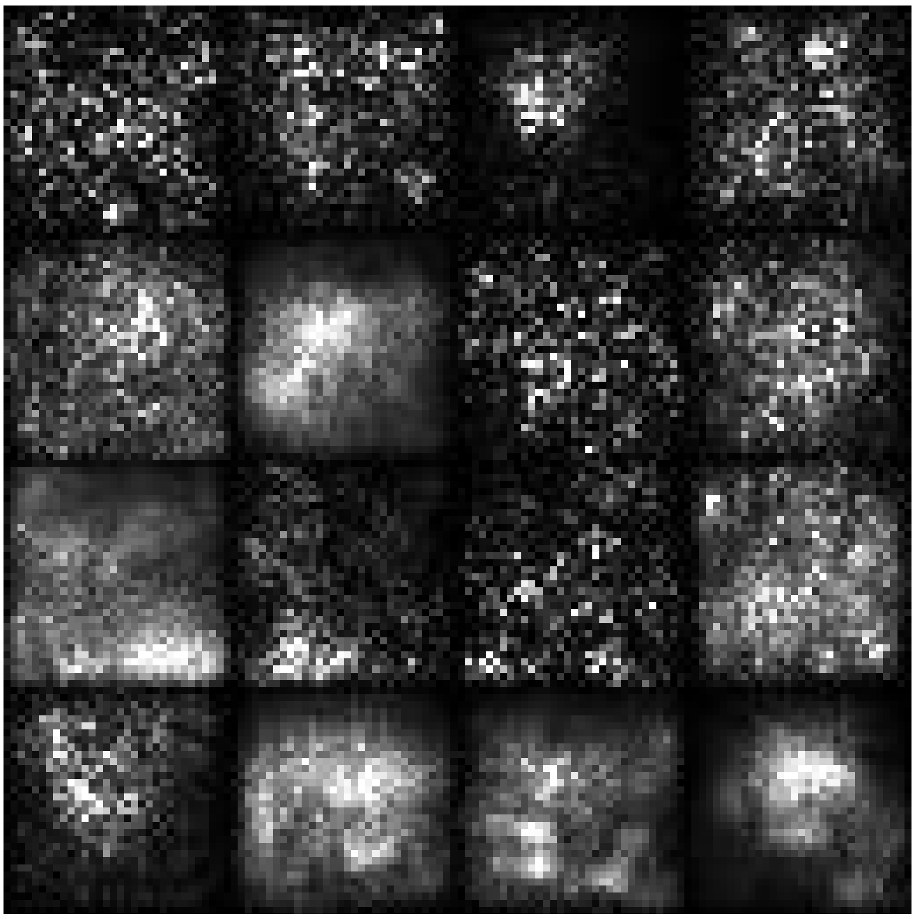
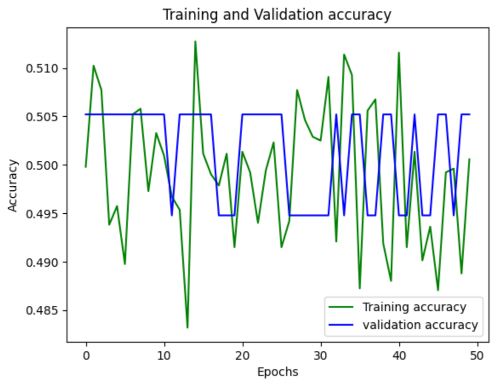
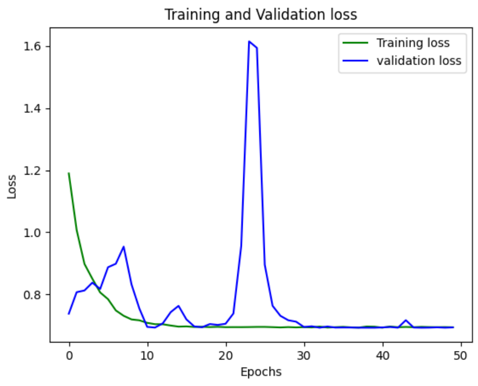


Figure 5.1: Example of samples in the generated saliency maps dataset.



(a) The training and validation accuracy of the secondary network on smoothGrad saliency maps dataset



(b) The training and validation loss of the secondary network on smoothGrad saliency maps dataset

Figure 5.2: The accuracy and loss of the secondary network on smoothGrad saliency maps dataset

Chapter 6

Saliency maps of things that affect uncertainty

6.1 Experiment setup

The last question we want to know is which part of the image determines the uncertainty of the prediction. To do this, we still use saliency maps to help us. However, unlike the regular saliency map, we used in the previous experiment, which takes the gradient of the output with respect to the input, here we take the gradient of the uncertainty with respect to the input, where the uncertainty can be computed during inference. We can thus know which part of the image has the most significant effect on the gradient and, therefore, which part of the image has the most significant effect on the uncertainty.

6.2 Results

In the third experiment, we tried to find which part of the image had the most significant effect on uncertainty, and the results are shown in fig 6.1. In fig 6.1, the first column of the image is the original input image, the second column of the image is the saliency map indicating which part of the image has the most influence on the prediction accuracy, and the third column is the saliency map indicating which part of the image has the most influence on the uncertainty. As shown in fig 6.1, we can see that the saliency map of uncertainty has a similar structure to the saliency map of prediction accuracy. This is because uncertainty is always correlated to prediction accuracy, and the part of the image that has a significant impact on uncertainty always has a significant impact on prediction accuracy. For example, images with low prediction accuracy always have high uncertainty and vice versa. In addition, the saliency map of uncertainty is more dispersed than the saliency map of prediction accuracy. We believe this is because the non-central regions of the images also have a strong influence on the uncertainty, e.g., unclear backgrounds may interfere with the network and lead to high uncertainty.

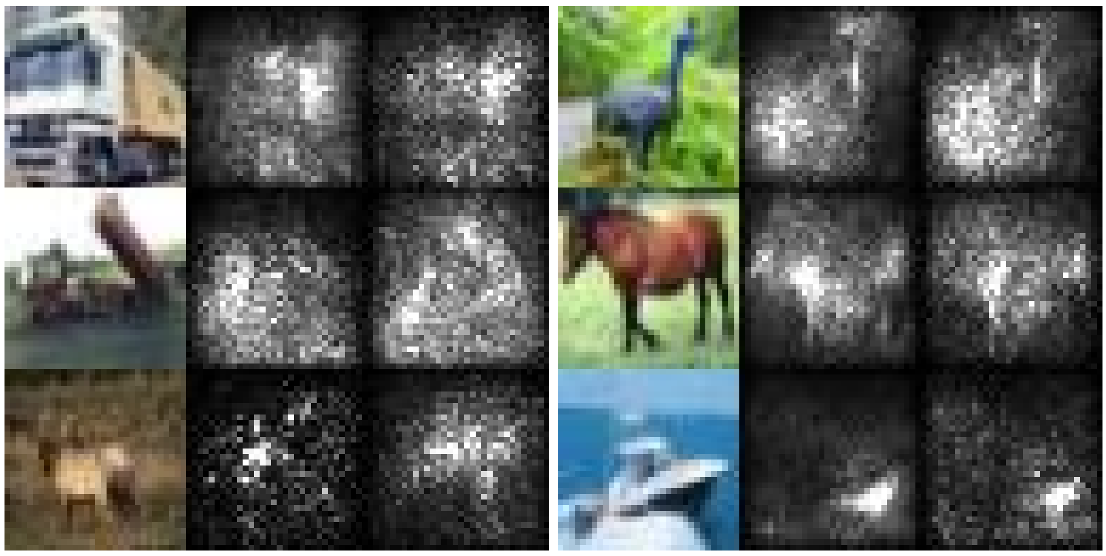


Figure 6.1: The saliency maps generated by EDL model. The first column is the original input images, the second column of the image is the saliency map indicating which part of the image has the most influence on the prediction output, and the third column is the saliency map indicating which part of the image has the most influence on the uncertainty.

Chapter 7

Conclusion and Future work

In this chapter we summarise the experiments we have carried out. On top of that, we give an overview of what we can do in future research.

7.1 Summary

In the first experiment, we examined the saliency maps generated by the EDL network. However, in order to make a quantitative analysis of whether the saliency map generated by the EDL network contains information about the uncertainty, we used a secondary network to find whether it could use the saliency maps to determine whether the predictions were uncertain or not. Unfortunately, the secondary network failed to predict whether or not the saliency map was uncertain. Therefore, we concluded that the saliency map generated by the EDL network did not contain additional information indicating uncertainty. In the third experiment, we investigated which part of the image has a more significant effect on uncertainty. We obtained the result that the part of the image that determines the prediction's accuracy also determines the prediction's uncertainty. In addition, the background of the image also has a significant effect on the uncertainty of the prediction.

7.2 Future work

There might be some work we can do in the future to further dig in this topic. We believe it could be following directions.

1. Alternative network architecture can be chosen. The reason why the saliency maps of current network does not contain uncertainty information may be that the model is not good enough. An updated version of EDL was released in 2020[12], in which the structure was further

changed to improve performance. In this version of the EDL, each category can be considered a hypersphere in the feature space. Samples that do not belong to any of the hyperspheres are considered high uncertainty samples.

2. Alternative interpretability tools can be chosen. The interpretability tool used in this project may not be powerful enough to capture the uncertainty information, while other techniques may be able to do so. For example, adversarial attacks can be combined with saliency maps to interpret the model[9]. Adcersarial attack can probe the internal representations of neural networks to generate interpretable results[3].

References

[1] P. J. Grother, "Nist special database 19," *Handprinted forms and characters database, National Institute of Standards and Technology*, vol. 10, 1995.

[2] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2014. DOI: `10.48550/ARXIV.1412.6980`. [Online]. Available: `https://arxiv.org/abs/1412.6980`.

[3] Y. Dong, H. Su, J. Zhu, and F. Bao, "Towards interpretable deep neural networks by leveraging adversarial examples," *arXiv preprint arXiv:1708.05493*, 2017.

[4] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint arXiv:1702.08608*, 2017.

[5] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, "Smoothgrad: Removing noise by adding noise," *arXiv preprint arXiv:1706.03825*, 2017.

[6] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim, "Sanity checks for saliency maps," *Advances in neural information processing systems*, vol. 31, 2018.

[7] A. JSANG, *Subjective Logic: A formalism for reasoning under uncertainty.* Springer, 2018.

[8] M. Sensoy, L. Kaplan, and M. Kandemir, "Evidential deep learning to quantify classification uncertainty," *Advances in neural information processing systems*, vol. 31, 2018.

[9] C. Zhang, Z. Ye, Y. Wang, and Z. Yang, "Detecting adversarial perturbations with saliency," in *2018 IEEE 3rd International Conference on Signal and Image Processing (ICSIP)*, IEEE, 2018, pp. 271–275.

[10] Brownlee, *How to Develop a CNN From Scratch for CIFAR-10 Photo Classification*, May 2019. [Online]. Available: `https://machinelearningmastery.com/how-to-develop-a-cnn-from-scratch-for-cifar-10-photo-classification/`.

[11] *Saliency Maps in Tensorflow 2.0 · UR Machine Learning Blog*, May 2020. [Online]. Available: `https://usmanr149.github.io/urmlblog/cnn/2020/05/01/Salincy-Maps.html`.

[12] M. Sensoy, L. Kaplan, F. Cerutti, and M. Saleki, "Uncertainty-aware deep classifiers using generative models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 5620–5627.

[13] Alexander Amini, *MIT 6.S191: Evidential Deep Learning and Uncertainty*, Mar. 2021. [Online]. Available: `https://www.youtube.com/watch?v=toTcf7tZK8c`.

[14] A. Widdicombe, S. Julier, and B. Kim, "Saliency maps contain network "fingerprints"," in *ICLR 2022 Workshop on PAIR^2Struct: Privacy, Accountability, Interpretability, Robustness, Reasoning on Structured Data*, 2022. [Online]. Available: `https://openreview.net/forum?id=SWlpXB_bWc`.

[15] Wikipedia contributors, *MNIST database*, Jul. 2022. [Online]. Available: `https://en.wikipedia.org/wiki/MNIST_database`.

[16] *CIFAR-10 and CIFAR-100 datasets.* [Online]. Available: `https://www.cs.toronto.edu/%7Ekriz/cifar.html`.

[17] *MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges.* [Online]. Available: `http://yann.lecun.com/exdb/mnist/`.

[18] *Saliency (PAIR-code).* [Online]. Available: `https://pair-code.github.io/saliency/#home`.

Appendix A

Other appendices, e.g. code listing

The code of this project is published on Github with url `https://github.com/Kouakiradou/MSc-Project`