Thesis Proposal

Daniel Fentham

Supervisors: Mark Ryan & Dave Parker
Thesis Group Members: Flavio Garcia & Hyung Jin Chang

August 28, 2020

Working title
Bayesian Geometric Deep Learning for Cyber Security

Abstract

Geometric Deep Learning has become an important field of research due to its ability to adapt many Machine Learning methods to non-Euclidean domains. This allows us to process huge amounts of data over structures such as social graphs and 3-D medical images which are vital in our increasingly complex analysis needs. However, this method is also susceptible to adversarial examples where a small imperceptible perturbation to the input data can lead to unintended behaviours. Bayesian inference allows us to estimate the uncertainty of predictions made by traditional Machine Learning methods and these have shown to provide good estimates in determining if an input is adversarial. Unfortunately, there is no general method to do this in the space of Geometric Deep Learning. The proposed thesis will look at this inadequacy and attempt to generalise the application of Bayesian methods to a number of Geometric Deep Learning algorithms by leveraging a common technique they use for convolution operations. This will be carried out with the underlying aim of applying Geometric Deep Learning methods to security related situations in the attempt to improve both the performance of the system and the robustness to adversarial examples.

Contents

1 Introduction

The field of adversarial attacks on Deep Neural Networks (DNNs) has caused a huge amount of concern for Artificial Intelligence (AI) applications where safety and security are of paramount importance. Adversarial examples can be defined as calculated perturbations that are made to the input of a DNN which leads to unexpected behaviour which benefits an adversary. Since the seminal paper exposing these insecurities was published [1] attacks have become more powerful and subtle, causing more reliable manipulations of behaviour with less perceptible changes to the input data. Furthermore, the spectrum of DNN architectures found to be vulnerable to this type of attack has widened to include image classification networks [1], reinforcement learning architectures [2] and graph neural networks [3]. The broad applicability of adversarial attacks across a number of different architectures hints at a more systemic issue in the fundamental theory of DNNs which we cannot ignore.

Geometric Deep Learning (GDL) is an umbrella term for Deep Learning over any domain which is not Euclidean in its structure. This includes manifolds and, importantly, graphs which are a very natural structure to operate over in the real world. Since GDL is a broad term applied to many different geometric spaces we may work over there are a number of different algorithms to handle these different spaces. However, a majority of the algorithms share a method of performing convolution in the spectral domain.

Through the work of Zügner et al. [3] the reach of adversarial examples has grown to include Graph Convolution Networks (a specific implementation of GDL). This means that GDL based systems are restricted in their use when it comes to safety and security critical applications in a similar way to traditional DNNs. For a technology which is likely to become more ingrained in systems as we need to process data with increasing structural complexity, this is highly problematic.

A theory surrounding the literature of adversarial examples is that when a perturbation is applied to an input, the perturbed input moves away from the true data manifold [4] into regions where extrapolations about the network knowledge are required. These extrapolations make large assumptions about the data which allows adversarial examples to exist. To combat this theory Bayesian Neural Networks (BNNs) have been successfully implemented to detect these data points which lie off the manifold of natural data by looking at the uncertainty estimates in the model results.

Approaches to implementing Bayesian inference in GDL have been defined by Zhang et al. and focus on graph structures and the explicit calculations graphs use [5]. This means these methods are not generalisable to GDL methods where graphs are not the underlying geometric structure of data. This restricts the application of Bayesian inference in domains that are non-Euclidean in structure.

The proposed thesis attempts to generalise the application of Bayesian methods to multiple GDL algorithms by utilising a shared convolution method. Many

GDL algorithms use the spectral domain to carry out convolution over non-Euclidean structures. If we are able to place a distribution over the space of potential convolution kernels (which is commonly done in Bayesian Convolutional Neural Networks [6]) then we can implement this over all algorithms which use this method of convolution. Realistically, the use of the spectral domain in GDL is limited due to inefficient calculations, therefore, some alterations are made to the method which may complicate the Bayesian implementation. Mixture model networks (MoNet) [7] can also be used which generalises a number of GDL algorithms to a single model. The convolution method used in MoNet could also have a distribution placed over the convolution kernels producing a Bayesian MoNet, which would also generalise to a number of different GDL models while keeping the efficiency benefits.

This will all be carried out in the context of the security of computer systems. The increasing ability of DNNs has led to a huge surge in applications leveraging this technology to process data in an intelligent way. However, due to the existence of adversarial attacks this opens these applications to potential manipulations of the system which benefits an adversary, and limits the use of them in safety and security critical systems.

As an example, the potential danger of adversarial attacks was shown by Eykholt et al. [8] where adversarial stickers were generated which, when placed on traffic signs, caused the signs to be misclassified by an image classifier (for example, changing a stop sign to speed limit sign). This work was recently extended by McAfee researchers who successfully fooled a Tesla speed assist system by placing electrical insulating tape on a traffic sign [9]. Further examples exist in facial recognition systems where Sharif et al. successfully caused a misclassification in off the shelf facial recognition software by developing a pair of glasses with a printed pattern attached [10]. This attack could either cause the facial recognition software to misclassify the user as another random person or as a defined target. More immediate and applicable applications of adversarial attacks to our everyday lives include social networks where the content of uploaded images are checked to ensure that they conform to the terms of service. With the application of adversarial perturbations, these checks can be undermined.

From the examples listed, it is clear to see that adversarial attacks have a broad scope to cause issues in the technology sector. From the manipulation of road signs, autonomous vehicles can be attacked which can lead to injury. Attacking facial recognition systems can authenticate adversaries where they would otherwise be refused. Perturbing images that are uploaded to social networks can lead to the failure of the site to remove the content automatically, allowing harmful content to propagate.

We can also gain insight into the importance of this work by considering the effect on systems based on GDL. Recent work has considered their use in detecting fake news over social networks by looking at the propagation patterns over the network [11]. The adversarial actor in this case may be able to manipulate the propagation patterns to avoid the automatic detection systems which allows

misinformation to spread with little resistance. Similar work was carried out by Liu et al. which used Graph Neural Networks to detect malicious domains that were used to spread malware [12]. The adversary in this case would also benefit from avoiding detection by placing adversaries in the graph which disrupt the detection mechanism.

The complexity of structures we expect Deep Learning to operate over is going to increase as GDL techniques advance and become more capable. This will include many security critical applications. Understanding how adversarial techniques can affect GDL models is vitally important. It must be taken seriously and tackled quickly.

2 Literature Review

The survey of previous work focuses on the key areas I will explore during the development of my thesis. Each topic is handled individually with some common threads being highlighted and discussed as topics intersect and feed into one-another.

I first discuss the generation of adversarial examples (section 2.1) which contains some of the most common and interesting attacks that can be applied to traditional DNNs. This is an important part of the survey since many adversarial example generation techniques build upon the ones specified or leverage them as a core part of an attack scenario. It also allows us to frame realistic attacker scenarios using existing adversarial technology since it has been shown that these simple adversarial attacks can also affect GDL models [3].

I then focus on adversarial defences in section 2.2 which outlines some defence techniques. Within these explanations some theory regarding the existence and possible defence of the adversarial examples is also noted which provides deeper insight into the ideas behind the creation of defences.

The topic then shifts to Bayesian Neural Networks in section 2.3. We first look at how Bayesian Neural Networks are constructed and how we can use this method to extract uncertainty estimates. We then explore the robustness of Bayesian Neural Networks to adversarial examples, and how we can use the uncertainty estimates to detect adversarial inputs (a key feature of Bayesian Neural Networks). This is then extended to applying Bayesian methods to convolution filters which helps to explain the starting point of the core part of the proposed thesis topic.

Geometric Deep Learning is discussed in the last section (2.4). As the foundational part of the thesis, this is a very important section which tackles Geometric Deep Learning at a fairly high level. However, it should be detailed enough to allow the proposed topic to be explained sufficiently. This section also details some applications of Geometric Deep Learning, specifically looking at the domains which have a security focus with the potential for adversarial examples to cause disruption. Existing research on adversarial examples in the Geometric Deep Learning domain is then discussed which connects the core technology of the proposed thesis with a clear application to the adversarial

space we will be working in. The combination of Bayesian methods and GDL is also explored to evaluate the current methods in this area.

2.1 Generating Adversarial Examples

Adversarial examples have made fast progress since the seminal paper by Szegedy et al. was published in 2013 [1]. Their ability to manipulate the predictions of DNNs have made them a particularly interesting topic in the cyber security space. Therefore, how they integrate in to realistic attacker models must be considered if we hope to defend against them.

Many adversarial example generation techniques focus on a white-box scenario. This is where the adversary has complete access to the network (the weights of the trained model, gradients, datasets used to train, validate and test the model, model architecture etc). These do not provide a realistic view of what an adversary would have access to when attacking an external system, but they do provide a *worst case* scenario. Therefore, by proving a model cannot be attacked in a white-box setting provides strong evidence that, when given less knowledge of the model, an adversary will not be able to produce adversarial examples. Most of the adversarial example generation techniques in the following section assume a white-box scenario (sections 2.1.1 – 2.1.5).

The other common scenario is black-box. This assumes that an adversary has no access to any of the internals of the model (weights, gradient, datasets etc) and provides a more realistic attack scenario that an adversary may face. Since there is no data that can be used to generate adversarial examples from, this is usually more difficult. In this situation the property of transferability [1] is relied upon. Transferability tells us that an adversarial example generated for one model is likely to fool another with high probability given they are trained to carry out similar tasks. By leveraging this intriguing property we can create our own surrogate model which carries out a similar task to the target model, then create adversarial examples using white-box methods on the surrogate, the generated examples should then be able to lead to incorrect behaviour in the target model with high probability. There are also attack methods (such as ZOO in section 2.1.6) which can generate examples for black-box scenarios without the need for surrogate models to be created.

The space in between the white-box and black-box scenarios is grey-box. This assumes that the attacker has some access or knowledge of the model, but also some unknowns. The scope of what the attacker has access to or knows about the model is often stated in papers which handle adversarial attacks or defences.

2.1.1 L-BFGS

Adversarial examples have caused a great disturbance in the field of DNN research since the initial paper was published [1]. This paper showed that small, imperceptible perturbations to an input image can cause a DNN to misclassify

it even though it has a high accuracy on the test set and generalises well to new images. Szegedy et al. proposed the reason that adversarial attacks exist (citing [13]) is due to the hidden layers encoding a non-local generalisation prior over the input space. This means that portions of the input space with little or no data points can be assigned class probabilities by the output unit. These regions of the input space correspond to natural noise in the data, for instance, an object from a different view point, or a different colour of an object. This brings into question the similarity metric used by DNNs since these images can be far away in terms of pixel similarity yet still share the same label.

The adversarial perturbation that was proposed is simple and forms the basis for a number of more advanced attacks. They define a DNN $f : \mathbb{R}^m \to \{1, \ldots, k\}$ which classifies an input to one of k different classes. They also assume that f has a continuous loss function $loss_f : \mathbb{R}^m \times \{1, \ldots, k\} \to \mathbb{R}^+$. For a given input image $\boldsymbol{x} \in \mathbb{R}^m$ with target label $\ell \in \{1, \ldots, k\}$ they solve a box-constrained optimisation problem:

$$\min_{\delta} \ \|\delta\|_2 \ s.t. \ f(\boldsymbol{x} + \delta) = \ell; \ \boldsymbol{x} + \delta \in [0, 1]^m$$

The perturbation δ is added to the original input image $(\boldsymbol{x} + \delta)$ which is the closest image to $\boldsymbol{x}$ classified as the target label ℓ by the classifier f. The constraint $\boldsymbol{x}+\delta \in [0, 1]^m$ must be observed to ensure the perturbed image is still valid (pixel values are valid between 0 and 1). The most interesting applications of adversarial examples occur in the non-trivial case, therefore, $f(\boldsymbol{x}+\delta) \neq f(\boldsymbol{x})$ which leads to a difficult problem when evaluating the above equation since the minimum must be found. This difficulty leads to an approximation of the optimisation using the box-constrained L-BFGS method as a way of finding the local minimum. The aim is to find the minimum $c > 0$ for which the minimiser δ of the following problem satisfies the condition $f(\boldsymbol{x} + \delta) = \ell$:

$$\min_{\delta} \ c|\delta| + loss_f(\boldsymbol{x} + \delta, \ell) \ s.t. \ \boldsymbol{x} + \delta \in [0, 1]^m$$

This idea forms the basis for a vast majority of the adversarial perturbation calculations that have been defined since this seminal paper.

2.1.2 Fast Gradient Sign Method

In the paper by Szegedy et al. adversarial training is proposed as a defence to the existence of adversarial examples [1](see section 2.2.1 for more details). To improve upon the box-constrained L-BFGS method for generating adversarial examples for this defence Goodfellow et al. proposed the Fast Gradient Sign Method (FGSM) [14]. This method efficiently calculates adversarial examples by exploiting linear behaviour in high dimensions. Practically they optimise the problem:

$$\delta = \varepsilon \text{sign}(\nabla_{\boldsymbol{x}} \mathcal{J}(\boldsymbol{\theta}, \boldsymbol{x}, \ell))$$

In the above equation $\boldsymbol{\theta}$ are the model parameters, $\mathcal{J}(\cdot)$ is the cost function used to train the network, $\nabla_{\boldsymbol{x}}$ is the gradient of the cost function with respect to the input image and ε is a small positive constant which is the size of the step we take in the direction of the gradient. The function "sign" is the sign of the gradient of the cost function (either positive or negative). The larger the value of ε, the more noticeable the perturbation.

This leads to the calculation of adversarial examples defined as:

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} + \delta$$

This is an untargeted attack on the network. This means we do not specify which label the image should be assigned, as long as it is not the true label. We can alter the definition of FGSM slightly to obtain a targeted attack by changing the true label ℓ to the target label then focus on minimising the cost function $\mathcal{J}(\cdot)$.

This attack exploits the linearity Goodfellow et al. believe is the key to adversarial examples. They propose that many networks are encouraged to be as linear as possible to allow for easy optimisation which is usually achieved through the widespread use of the ReLU activation function. This linearity means that the FGSM attack is highly effective, simple to implement and extremely cheap to run. These properties have made it a staple adversarial example generator for many papers in this field and is often used as a benchmark for the effectiveness of adversarial defences against the simple attacks.

2.1.3 Basic Iterative Method

A simple extension to FGSM was defined by Kurakin et al. which applied FGSM multiple times to generate adversarial images [15]. This is the Basic Iterative Method (BIM). The application of FGSM repeatedly changes the adversarial example generation from a one-step attack to a multi-step one with the advantage that we can take many smaller steps with changes in direction which leads to a smaller perturbation (provided the step size is small).

The method starts with:

$$\tilde{\boldsymbol{x}}_0 = \boldsymbol{x}$$

With the recursive definition:

$$\tilde{\boldsymbol{x}}_{n+1} = \text{clip}_{\boldsymbol{x},\varepsilon}\{\tilde{\boldsymbol{x}}_n + \alpha \text{sign}(\nabla_{\boldsymbol{x}} \mathcal{J}(\boldsymbol{\theta}, \boldsymbol{x}_n, \ell))\}$$

$\text{clip}_{\boldsymbol{x},\varepsilon}\{\cdot\}$ clips the pixel values of intermediate results to ensure they are within an ε-neighbourhood of the original image and produce valid images (i.e. pixel values between 0 and 255). Note that α is now the step size.

In a similar way to the FGSM attack, we can alter this to target a specific classification by changing the true label ℓ to the target label and then minimise the cost function.

2.1.4 Universal Adversarial Perturbations

Universal Adversarial Perturbations (UAPs) were defined by Moosavi-Dezfooli et al. and take a slightly different approach to generating adversarial examples [16]. Previous methods have focused on generating an adversarial example using the current data point and the model information (the gradient for example) to generate a unique adversarial perturbation for that specific point. UAP was defined to work in the space of images and generates a single perturbation which can be applied to any image to cause misclassification.

Formally we can write this as:

$$f(\boldsymbol{x} + \delta) \neq f(\boldsymbol{x}) \text{ for } \textit{most } \boldsymbol{x} \sim \mu$$

In the above equation μ is the natural distribution of images in $\mathbb{R}^m$. The adversarial perturbation (δ) is *universal* if it causes misclassification for *most* of the images from the natural distribution.

This method of generating adversarial perturbations focuses on satisfying two constraints:

- $\|\delta\|_p \leq \xi$
- $\mathbb{P}_{\boldsymbol{x} \sim \mu}(f(\boldsymbol{x} + \delta) \neq f(\boldsymbol{x})) \geq 1 - \beta$

The first point tells us that the ℓ_p-norm of δ must be below some specified threshold ξ to encourage perturbations that do not exceed this limit. The second constraint makes sure that all $\boldsymbol{x}$ in the distribution must be misclassified with a probability $1 - \beta$. It can easily be seen that the two constraints are at odds with one another. By decreasing ξ to make the perturbations less perceptible we make it more difficult to satisfy the second constraint and vice-versa.

In practice, a (surprisingly small) set of images is sampled from μ with a minimal perturbation, which sends the current point to the classification boundary, calculated for each element. This perturbation is aggregated with the current universal one. This universal perturbation is projected back to the ℓ_p-ball of radius ξ each time to ensure the perturbation is imperceptible (does not exceed the set threshold ξ). Multiple passes can be done over the set of images until the rate of misclassification $(1 - \beta)$ is satisfied. Readers are directed to the full paper for a more in-depth analysis of this method [16].

2.1.5 Carlini & Wagner

One of the most advanced attacks currently used as a benchmark for adversarial defences on images is the Carlini & Wagner attack (C&W) [17]. This attack was developed in response to the distillation defence developed by Papernot et al. [18] (see section 2.2.2 for further details). The C&W attack focuses on minimising a defined distance metric between the original clean image and the adversarially perturbed one while also ensuring an objective function (g) is satisfied and the perturbed image is valid:

$$\text{minimise} \quad \mathcal{D}(\boldsymbol{x}, \boldsymbol{x}+\delta) \text{ s.t. } g(\boldsymbol{x}+\delta) \leq 0,\ \boldsymbol{x}+\delta \in [0,1]^m$$

The distance metric ($\mathcal{D}$) can be ℓ_0, ℓ_1 or ℓ_∞ depending on how we want the adversarial perturbations to behave:

- ℓ_0-norm – Corresponds to the number of pixels that have been altered in an image. Restricting this limits the number of pixels that can be changed, but not how much they change by.
- ℓ_∞-norm – Measures the maximium change to any coordinate. Restricting this limits the amount any one pixel can change by, but does not limit the number of pixels that can change.
- ℓ_2-norm – The root-mean-square distance between the perturbed and unperturbed image. This balances both the number of pixels that are perturbed and the amount they are changed by.

The objective function (g) is an optimisation problem which can be solved using existing optimisation algorithms which is included due to the difficult minimisation problem of finding the minimal δ which, when added to the clean image, gives us our target class. The objective function g can take on many different forms including $g(\boldsymbol{x}+\delta) = -\mathcal{J}_{\ell_{target}}(\boldsymbol{x}+\delta)+1$ for instance. The above equation can be expressed more simply as:

$$\text{minimise} \quad \|\delta\|_p + c \cdot g(\boldsymbol{x}+\delta) \text{ s.t. } \boldsymbol{x}+\delta \in [0,1]^m$$

This can be informally read as, given $\boldsymbol{x}$ find δ which solves the above equation where we have a suitably chosen constant c. Carlini et al. select c to be the smallest value for which $g(\tilde{\boldsymbol{x}}) \leq 0$ and is found using a modified binary search.

The three attacks introduced by this paper (ℓ_0, ℓ_2 and ℓ_∞) all follow from the above optimisation problem and allows adversarial examples to be generated which target a particular class and specify the classification accuracy of the image on the target network. This is an incredibly powerful attack.

2.1.6 Zeroth Order Optimisation

The Zeroth Order Optimisation attack (ZOO) [19] is a black-box method for generating adversarial examples which builds off of the C&W attack [17] (see section 2.1.5). The C&W method requires white-box access to the system, limiting its use since the network logits must be extracted and the gradient must be used to optimise the objective function. Since the ZOO technique is black-box, the access to the internals of the target model is not required, in addition we do not need to know anything about the dataset the model was trained on or the architecture of the target model itself which is information generally required for transfer attacks. Unlike transfer attacks ZOO is competitive with state-of-the-art white-box adversarial example generation methods (such as C&W) and does not have any loss in performance when the examples are used on different networks (transfer attacks).

To formulate the adversarial example a derivative-free optimisation method is used. By using results provided by the target model (a zeroth order oracle) we can evaluate two points close to one another to approximate a gradient vector. Classical optimisation algorithms are then used, such as gradient descent, to generate the adversarial example. However, due to the large number of parameters that need to be optimised for large models, this method is not feasible in practice (for example CIFAR images have $32 \times 32 \times 3 = 3072$ parameters). To combat this issue ZOO iteratively updates batches of coordinates (batches of pixel values) with improved efficiency when sampling strategies are used to optimise only the most important pixels first. This reduces the complexity of the optimisation process massively.

For full details of how ZOO can be implemented by leveraging C&W and derivative-free optimisation methods the interested reader is directed to the original paper [19].

2.2 Adversarial Defence

Research into adversarial defences formed an early part of the literature reviewed and gives a good insight into the theories behind the existence of adversarial examples. All of the defences proposed in the literature can be circumvented by an existing attack which paints a pessimistic picture for the security of AI systems. The existence of adversarial examples seems to be a natural occurrence which cannot be avoided. The methods detailed in this section go some way to securing AI systems but they are not perfect and cannot guarantee a perfect defence against a determined and skilled attacker.

2.2.1 Adversarial Training

The seminal paper by Szegedy et al. suggested that the robustness of a network can be improved by providing adversarial examples as training input referred to as adversarial training [1]. They provided some evidence of this through experimental results, however, they could not be rigorously tested due to the inefficient box-constrained L-BFGS method that was used to generate the adversarial examples. This work was carried forward by Goodfellow et al. when they defined the FGSM adversarial example generation technique [14]. With the improved efficiency of adversarial example generation, further experiments could be carried out which confirmed the increased robustness from adversarial training.

The theory behind this defence is as such: By the universal approximator theorem [20] we know that DNNs can at least *represent* functions which are robust to adversarial examples. However, the challenge is to develop a dataset which enables a robust function to be learned. Adversarial training generates adversarial examples from the dataset using FGSM and includes them in the training set in an attempt to include the set of functions which are more robust to adversarial examples as a function that can possibly be learned.

Adversarial training increases the robustness of DNNs to adversarial examples. However, they are not stopped completely. Many adversarial example generation techniques can always find more adversarial examples even after adversarial training has been performed.

2.2.2 Defensive Distillation

Work carried out by Papernot et al. [18] proposes defensive distillation as a possible way to mitigate adversarial examples which is inspired by distillation work carried out by Hinton et al. [21]. The defensive distillation method starts by training an oracle network to carry out a task (such as image classification) to a suitable degree of accuracy. A second network is then created which has the same architecture as the first (this is the divergence from the work by Hinton where usually the secondary network is smaller, so it can be used in more computationally limited environments e.g. phones). To train the second network, a dataset is provided to the oracle network which produces outputs (usually a softmax vector) which are used to label the dataset with *soft labels*. The dataset and soft labels are then provided to the second network which is trained on this dataset.

The aim of this method of training is to perform gradient smoothing which makes gradient based adversarial example generation techniques (such as L-BFGS, FGSM, BIM etc) more difficult to execute. This also has the effect of helping the network to generalise more strongly to samples outside of the training dataset which results in a model which is more robust to adversarial examples.

Unfortunately, this defence method was found to be vulnerable to the C&W attack [17] (see section 2.1.5). This undermines the use of distillation as a defence in situations where the attacker can execute either a C&W attack or the ZOO attack [19] (section 2.1.6) which was inspired by C&W.

2.3 Bayesian Neural Networks

Bayesian Neural Networks (BNNs) are an important area to understand in the development of this thesis since the techniques used here form a core part of the proposed topic. This proposal highlights the ability of BNNs to extract uncertainty from predictions made on new inputs and how uncertainty can be used to detect adversarial examples. In the following few sections the methodology behind these features will be explored which will, hopefully, give some intuitive understanding of how they could be adapted to GDL.

2.3.1 Theory

If we have a dataset $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{Y}\}$ where $\boldsymbol{X} = \{x_1, \ldots, x_N\}$ inputs and $\boldsymbol{Y} = \{y_1, \ldots, y_N\}$ labels, then we aim to find a set of functions $y_i = \boldsymbol{f}(x_i)$. We can place a prior distribution over the space of functions which represents our prior about which functions are likely to have generated our data. The probability

$p(\boldsymbol{Y}|\boldsymbol{f}, \boldsymbol{X})$ tells us which observations are generated given a specific function. We then look for the posterior distribution over the space of functions to determine which functions are most likely given our observed data. This is expressed as $p(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{Y})$. Using this equation we can predict an output given a new input:

$$p(y^*|x^*, \boldsymbol{X}, \boldsymbol{Y}) = \int p(y^*|\boldsymbol{f}^*)p(\boldsymbol{f}^*|x^*, \boldsymbol{X}, \boldsymbol{Y}).\mathrm{d}\boldsymbol{f}^*$$

The integration over all $\boldsymbol{f}$ beautifully implies that we do not base our prediction on the model. All the predictions made only use the data we have observed and consider all possible models. Unfortunately this form of prediction is intractable for many models due to the prior $p(\boldsymbol{f}^*|x^*, \boldsymbol{X}, \boldsymbol{Y})$. Instead, we approximate it by conditioning the model on a set of finite random variables $\boldsymbol{\omega}$ (model parameters), with the modelling assumption that the model depends only on these parameters. With this assumption the prediction of input x^* becomes:

$$p(y^*|x^*, \boldsymbol{X}, \boldsymbol{Y}) = \int p(y^*|\boldsymbol{f}^*)p(\boldsymbol{f}^*|x^*, \boldsymbol{\omega})p(\boldsymbol{\omega}|\boldsymbol{X}, \boldsymbol{Y}).\mathrm{d}\boldsymbol{f}^*.\mathrm{d}\boldsymbol{\omega}$$

The term $p(\boldsymbol{\omega}|\boldsymbol{X}, \boldsymbol{Y})$ is (again) intractable. However, we can get close to the true value by using approximating methods such as Variational Inference, Hamiltonian Monte Carlo or Monte Carlo Dropout [22]. Using this method we can incorporate Bayesian modelling into the Neural Network prediction process.

2.3.2 Uncertainty and Adversarial Examples Bayesian Methods

One hypothesis for the existence of adversarial examples is that they are data points which lie off the manifold of natural data. In these areas the model would have no prior experience to guide its classification, and so it must extrapolate its knowledge making incorrect assumptions about the unfamiliar space. Therefore, to detect adversarial inputs a measure of the distance from an input to the manifold of natural data can be calculated.

Geometric methods of distance calculation in this space can be used but in the case of data such as images, geometric distance is a poor proxy for image similarity, therefore, more reliable measures such as model uncertainty can be used [23]. Due to the distributions placed over the model parameters we can easily extract the uncertainty of the classification of the predicted labels. This uncertainty comes in different forms:

- **Aleatoric** – Uncertainty due to natural noise in the data
- **Epistemic** – Uncertainty due to a lack of knowledge (insufficient data in the training set to make a reliable prediction)

The uncertainties detailed above can be used for adversarial example detection. Inputs to the BNN which have a high epistemic uncertainty indicate adversarial inputs since the model is making a prediction even though it has no

data to support it.

To evaluate these uncertainties a simpler uncertainty can be extracted from a classification which is the entropy of the predictive distribution:

$$H[P(y|x)] = -\sum_{y \in \boldsymbol{Y}} P(y|x) \log P(y|x)$$

However, this entropy gives us no distinction between aleatoric or epistemic uncertainty. Instead, we can consider the information we would gain about ω if we receive label y for a point x given the dataset $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{Y}\}$. This is Mutual Information (MI):

$$I(\boldsymbol{\omega}, y|\mathcal{D}, x) = H[(p(y|x, \mathcal{D}))] - \mathbb{E}_{p(\boldsymbol{\omega}|\mathcal{D})} H[p(y|x, \boldsymbol{\omega})]$$

Considering the different forms of uncertainty, we can easily determine that if we have a new point x' classified as y' with a high MI value, then we have learned a lot from classifying that point. This indicates that we did not previously have knowledge in this area and therefore we can determine that the point x' is potentially adversarial which is a high epistemic uncertainty.

If instead, the MI value is low for x' classified as y' then we do not gain much new information which leads to a low epistemic uncertainty. Therefore, we can conclude that MI is a measure of the epistemic uncertainty of a model.

This method for finding the epistemic uncertainty of the classification is made tractable if the Monte Carlo Dropout approximation of sampling the posterior is used. For further information of how to implement this the reader is directed to [23, 22].

2.3.3 Robustness to Adversarial Examples

Work carried out by Carlini et al. [24] looked at the performance of Bayesian Neural Network uncertainty against adversarial examples. The BNN uncertainty was taken from a paper by Feinman et al. [4] where dropout is used as the source of randomness. Feinman defined the uncertainty of a classification as:

$$U(\boldsymbol{x}) = \left(\frac{1}{N} \sum_{n=1}^{N} \|f_r(\boldsymbol{x})\| \right) - \left\| \frac{1}{N} \sum_{n=1}^{N} f_r(\boldsymbol{x}) \right\|$$

The result $U(\cdot)$ is calculated by running the BNN N times on the same input. f_r in the equation is the randomly selected Neural Network for that particular iteration n and $\|\cdot\|$ is the ℓ_2 squared norm (equivalent to $y^\top y$). Since the output ($f_r(\boldsymbol{x})$) is a vector the ℓ_2 squared norm is the length of this vector. If the average vector length for all the sampled networks is equal to the ℓ_2 squared norm of the average of the sum of all of the vectors, then the uncertainty is small. A parameter τ is then defined which is the threshold above which we declare an input as adversarial, for instance if $U(\boldsymbol{x}) > \tau$ the input is deemed adversarial.

Carlini et al. generated many adversarial examples using different generation techniques and passed them to the BNN. For the simple attacks FGSM and JSMA [25] on the MNIST dataset 96% of the adversarial examples were detected with a false positive rate below 1% when $\tau = 0.02$ and $N = 30$. They continued to test the BNN with more advanced attacks:

Zero-Knowledge – The zero-knowledge attack uses the C&W method for adversarial example generation where the attacker has no knowledge of the defence that is in place. In this test 75% of the adversarial examples were detected for MNIST and 95% were detected for CIFAR.

Perfect-Knowledge – This is an adaptive white-box attack where we have knowledge of the model and defence in place. New loss functions for the C&W method can be implemented to perform the strongest possible attack. K different deterministic networks are chosen at random with the loss calculated for each. The loss is then averaged and used with the C&W attack. This produces examples which easily fool the BNN with 98% success (only 2% of examples are detected) on the MNIST and CIFAR datasets. However, the perturbation required for this fooling rate is much larger on both.

Limited-Knowledge – This attack gives the adversary the knowledge of the type of defence in place, but no access to the model is provided. This attack relies on the property of transferability discussed in section 2.1. Two models were constructed with adversarial examples being generated from the first, and transferability evaluated on the second. For MNIST, transferability occurred with 90% success (which can be increased to 98% with a small increase in perturbation strength). Transferability attacks can be achieved with CIFAR, but the perturbations required to fool the network become perceptible.

While this report by Carlini et al. shows that BNNs are still susceptible to adversarial examples the creation of white-box attacks has become nearly as hard as generating black-box attacks and the BNN has significantly better performance than some other defence mechanisms. For a full analysis of how BNNs compare to other defences, the reader is directed towards the original paper [24]. This evaluation indicates an inherent strength of Bayesian methods against adversarial examples.

2.3.4 Bayesian Convolution Layers

The Bayesian interpretation can also be extended to the convolution operation. Since a key part of the proposed thesis is concerned with placing distributions over convolution operators in a GDL setting, this is a particularly important part of the literature review.

When implementing Bayesian Neural Networks, dropout is performed during training after each layer with an approximating distribution, then evaluate the predictive posterior at test time [22]. Practically this means we include a dropout layer after each layer of the Neural Network which includes weights. In the work by Gal and Ghahramani since we want to integrate over the convolution kernels as well, then dropout is applied after each convolution layer

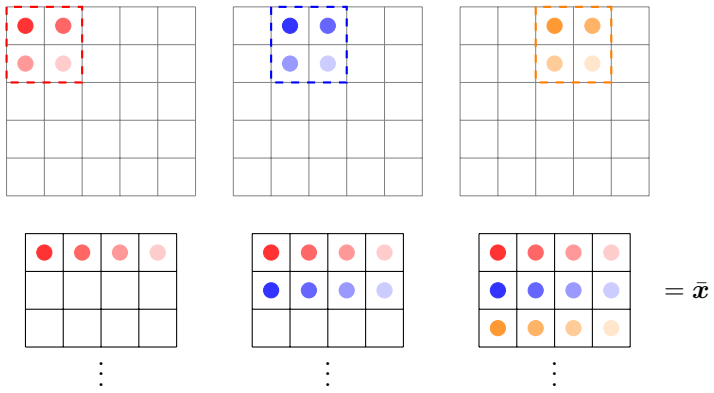

Figure 1: Visual example of the linearisation of extracted *patches*

[6].

To integrate over the kernels the convolution operator is adapted to be a linear operator (inner product). This is carried out by extracting all of the *patches* that we would convolve with as the kernel moves over the input space. Then convolution is just the matrix product of the kernel with these patches. These patches are vectorised and placed in the rows of a matrix to form a new input $\bar{\boldsymbol{x}}$ (see figure 1 for a visual representation of this). The kernels are also vectorised to form the columns of a weight matrix ($\mathbf{W}_i$ for network layer i). Convolution is then reduced to the matrix product $\bar{\boldsymbol{x}}\mathbf{W}_i$. Dropout on the convolution layer is then implemented by randomly setting a weight column to $\mathbf{0}$ by sampling a Bernoulli random variable.

The open question is whether this can be applied to the GDL domain. The transformation of convolution to a linear operator may not be possible in GDL due to the different structure the convolution operator must act over. This problem is one which forms a fundamental part of the proposed thesis.

2.4 Geometric Deep Learning

The nascent field of Geometric Deep Learning aims to extend the traditional Deep Learning methods to domains that are non-Euclidean in nature. This has led to the application of Deep Learning to graph structures, such as social networks [26], citation networks [27] and computer networks [12] and manifolds, such as the spheres [28] and 3-D models [29]. GDL forms an important part of the thesis proposed in this document therefore it is essential to outline the current work in this area.

2.4.1 Theory

Convolution layers provide a large amount of expressive power to traditional DL applications while being incredibly efficient. The aim and biggest challenge in GDL is the generalisation of convolution to non-Euclidean domains. The key properties of convolution are:

- **Convolutional Filters** (provide translation invariance) – Resulting from the shift invariance of the convolution operation
- **Multiple Layers** (provides compositionality) – Simple filters can be composed into more complex ones.
- **Spatially Localised Filters** (provides locality) – Resulting from small filters
- **Weight Sharing** (provides self-similarity) – Resulting from the application of the same filter to multiple parts of the input

Therefore, it is the aim of GDL to provide these advantages to non-Euclidean structures in the most general form possible.

A primer on GDL can be found in appendix A.

2.4.2 Graph Methods

GDL over graph structures was formalised by Kipf and Welling [27] with Graph Convolutional Networks (GCN) which introduced efficient convolution over graph structures. This work focused on classifying nodes on a citation graph by leveraging the graph structure and information about neighbouring nodes. This method can be summarised as a layer-wise propagation rule which is very reminiscent of standard Deep Learning layers:

$$H^{(l+1)} = \sigma(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)})$$

The above equation includes the adjacency matrix A of graph $\mathcal{G}$ with $\tilde{A} = A + I_N$ which is the adjacency matrix with added self connections. $\tilde{D}$ is the degree matrix $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ which is the sum of all edges incident on a node. $\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$ is a normalisation of the graph structure. $H^{(l)}$ is the activation of the previous layer ($H^{(0)}$ is the input) and $W^{(l)}$ is the weight matrix. Similar to traditional DNN notation, σ is the non-linear activation function. This form also includes the convolution of the graph with a filter. For proof of this the reader is directed to the original paper [27].

2.4.3 Mixture Model CNN

Another GDL method proposed by Monti et al. is the Mixture Model CNN (MoNet) [7]. MoNet attempts to generalise the application of convolution to a

Method	**Pseudo-coordinates**	$\mathbf{u}(x,y)$	**Weight function**
CNN [30]	Local Euclidean	$\mathbf{X}(x,y) = \mathbf{X}(y) - \mathbf{X}(x)$	$\delta(\mathbf{u} - \bar{\mathbf{u}}_j)$
GCNN [31]	Local polar geodesic	$\rho(x,y),\ \theta(x,y)$	$\exp\left(-\frac{1}{2}(\mathbf{u} - \bar{\mathbf{u}}_j)^\top \begin{pmatrix} \bar{\alpha}_\rho^2 & \\ & \bar{\alpha}_\theta^2 \end{pmatrix}^{-1} (\mathbf{u} - \bar{\mathbf{u}}_j)\right)$
ACNN [29]	Local polar geodesic	$\rho(x,y),\ \theta(x,y)$	$\exp\left(-\frac{1}{2}\mathbf{u}^\top \mathbf{R}_{\bar{\theta}_j} \begin{pmatrix} \bar{\alpha} & \\ & 1 \end{pmatrix} \mathbf{R}_{\bar{\theta}_j}^\top \mathbf{u}\right)$
GCN [27]	Vertex degree	$\deg(x),\ \deg(y)$	$\left(1 - \lvert 1 - \frac{1}{\sqrt{u_1}}\rvert\right)\left(1 - \lvert 1 - \frac{1}{\sqrt{u_2}}\rvert\right)$
DCNN [32]	Transition probability in r hops	$p^0(x,y), \ldots, p^{r-1}(x,y)$	$\mathrm{id}(u_j)$

Figure 2: Pseudo-coordinate and weight functions required to recover specified models from MoNet

number of different GDL algorithms while improving their efficiency and generalisability. The key to this construction is a *patch operator* which is a local charting procedure in geodesic polar coordinates.

If we have a point x on the manifold (or graph node depending on context) then $y \in \mathcal{N}(x)$ is a neighbour of x. For each y we have a d-dimensional vector of pseudo-coordinates $\mathbf{u}(x,y)$. The pseudo-coordinates can be seen as a way of defining the position of y in relation to x. A weighting function is defined using the pseudo-coordinates:

$$\mathbf{w}_{\mathbf{\Theta}}(\mathbf{u}) = (w_1(\mathbf{u}), \ldots, w_J(\mathbf{u}))$$

This equation is parameterised by learnable parameters $\mathbf{\Theta}$. The patch operator can then be defined as:

$$D_j(x)f = \sum_{y \in \mathcal{N}(x)} w_j(\mathbf{u}(x,y))f(y), \quad j = 1, \ldots, J$$

This summation can be interpreted as an integral if we are considering a continuous manifold with the value J representing the dimensionality of the extracted patch. It essentially evaluates the function f at the neighbours of x and determines how much they impact the value of x by using the given weight function.

Using the patch operator, we can now provide a spatial generalisation for convolution using a template matching procedure:

$$(f \star g)(x) = \sum_{j=1}^{J} g_j D_j(x) f$$

The convolution sums over all dimensions of the patch operator with the corresponding dimension of the convolution kernel.

MoNet generalises many existing GDL architectures by changing the pseudo-coordinates $\mathbf{u}$ and the weight function $\mathbf{w}$. The different architectures that can be achieved can be seen in figure 2.

2.4.4 Applications of GDL

The work on GDL has been expanded and adapted to fit a number of different applications. Many of the domains I will consider during the thesis will have applications to GDL, therefore the exploration of them will be important to consider. The simplest benchmark for GDL algorithms that is commonly used in the literature is on citation based networks (Cora [33] and CiteSeer [34]) [27]. This evaluates the ability of a GDL method (focused on the domain of graphs) to classify nodes of the graph by looking at the classification neighbouring nodes, node attributes and edge types to make an informed prediction about the class of the node under scrutiny. However, far more interesting (and security specific) applications also exist.

Malicious Domains: One such use is detecting malicious domains over a graph structure. Networks were created of nodes as domains and the relationships between domains (shared clients) as edges [12]. This paper was able to successfully detect malicious domains with a higher accuracy than the Euclidean counterparts.

Malware Detection: In a similar security related application, Yan et al. consider the use of Deep Graph Convolutional Neural Networks (DGCNNs) in detecting malware from the Control Flow Graphs (CFGs) of the program binaries [35]. The paper was able to generate attributed CFGs from the program binaries (with attributes such as numeric constants, arithmetic instructions etc) then run a DGCNN over the attributed CFG. This was able to achieve comparable performance to existing state-of-the-art methods with a more natural definition.

Fake News: An experimental use which has particular relevance in society today is the application of GDL methods to fake news detection on social media. The work conducted by Monti et al. used Graph CNNs to analyse the propagation of information over a social network [11]. From prior work in this area it was found that fake news had a characteristic spreading pattern which can be detected and used as an identifier for fake news. Furthermore, they claim that since the technique only looks at the way information spreads through the graph it is language independent, which gives a significant advantage over traditional Natural Language Processing methods.

Epidemic Model: Another paper by Shah et al. focused on expanding the ability of GNNs in the temporal domain in an effort to identify the first infected person (patient zero) in an epidemic model [36]. Their paper incorporated time models into the graph analysis to analyse the progress of the epidemic, and importantly, step back through time to find the first infected patient. This method could have a huge impact on the way epidemics can be modelled and analysed.

2.4.5 Adversarial Examples and GDL

In a paper by Zügner et al. the application of adversarial attacks to Graph Neural Networks is explored [3]. Since a Graph Neural Network is the target

architecture they create attacks which are effective against node classifications in networks (nodes are classified based on the class of neighbours and the features of the node itself). They describe two forms of attack. The first is an attack where the structure of the graph is changed (by altering the Adjacency matrix) with the aim of causing misclassification of a target node. The second is where the features of a node are changed with the intention of causing misclassification of a target node. This can be carried out as a *direct attack* where we alter the structure or the features at the node we wish to misclassify, or an *influencer attack* where we leave the target node unperturbed and alter the structure of the graph or the feature vectors of the neighbouring nodes. The attack they define (Nettack) is successful at causing adversarial perturbations of the graph and lead the Graph Neural Network to classify the target node incorrectly in a majority of the tests they carried out. This was the first piece of work to which extended adversarial attacks to operate explicitly on a graph structure.

In the same paper as Nettack by Zügner et al. [3] part of the analysis looked at applying traditional adversarial example techniques to the graph domain. They applied the FGSM attack [14] to the graph network due to the lack of competitors in this space. The gradient based method of FGSM performed very well on the graph and managed to produce adversarial examples. This finding means that existing adversarial example techniques defined for traditional architectures may be able to produce reliable attacks for GDL models. It would be interesting to explore this fact and consider small alterations to make effective attacks against GDL.

In an attempt to defend against traditional adversarial attacks on traditional Convolutional Neural Networks Svoboda et al. introduced PeerNets which leverage graphs of images to support classifications [37]. This method includes Peer Regularised layers in the traditional convolution architecture which compares the features extracted from an input image by convolution layers to those from a number of images from the training set. This then contributes towards the classification. The method defined in this paper makes the network more robust to adversarial examples, increases the strength of the perturbation that is required to fool the network and makes the perturbations more structured.

2.4.6 Bayesian Graph Convolution Networks

Recent work by Zhang et al. has applied Bayesian methods to GCNNs [5]. They view the observed graph as a realisation of a parametric family of random graphs where distributions are placed over the weights (W), the graph ($\mathcal{G}$) and the parameters that define the family of graphs (λ):

$$p(\mathbf{Z}|\mathbf{Y}_{\mathcal{L}}, \mathbf{X}, \mathcal{G}_{obs}) = \int p(\mathbf{Z}|W, \mathcal{G}, \mathbf{X}) p(W|\mathbf{Y}_{\mathcal{L}}, \mathbf{X}, \mathcal{G}) p(\mathcal{G}|\lambda) p(\lambda|\mathcal{G}_{obs}) . \mathrm{d}W . \mathrm{d}\mathcal{G} . \mathrm{d}\lambda$$

This then requires a number of approximations to finally achieve the out-

put ($\mathbf{Z}$) due to the intractable integral. The term $p(\mathbf{Z}|W, \mathcal{G}, \mathbf{X})$ is modelled by using a categorical distribution which is the softmax function on the output of the model. Variational inference can be used to approximate the posterior of the weights $p(W|\mathbf{Y}_{\mathcal{L}}, \mathbf{X}, \mathcal{G})$. Random graph generation models (e.g. stochastic block model) can be used to approximate $p(\lambda|\mathcal{G}_{obs})$. Full details of this methodology can be seen in the original paper [5].

The results of this method show that for citation networks where we must classify the nodes, this Bayesian approach outperforms previous methods by a significant amount when only a small number of nodes are labelled for us. This reinforces the advantage that Bayesian methods perform well even in situations where we have a small amount of training data. Furthermore, when they performed adversarial attacks on the citation network graphs outlined in work by Zügner et al. [3], they found that the Bayesian version of the GCN was significantly more robust to the adversarial examples generated. This again reinforces the use of Bayesian methods as a defence to adversarial examples. However, no evaluation was done on the uncertainty of classifications in this adversarial setting which would have been interesting to see.

3 Thesis Proposal

The proposed thesis expands on the work carried out by Zhang et al. [5] in the application of Bayesian methods to Graph Neural Networks by asking the question of whether Bayesian methods can be incorporated into Geometric Deep Learning at a point which is general between many different algorithms. This is all with the aim of making applications where Geometric Deep Learning is used more secure in realistic systems. The focus on the point of generality between many algorithms would potentially allow us to apply a Bayesian method across multiple domains without having to worry about the specific implementation any one algorithm has. This defined goal can be split into two separate parts:

1. Application of Bayesian methods to Geometric Deep Learning
2. The use of Bayesian Geometric Deep Learning in a realistic attacker scenario

These two parts can be broken down further, and require more exploration individually to fully understand the work that is required to carry them out.

3.1 Part 1: Bayesian Geometric Deep Learning

The primary question in this section is: *How can we place a distribution over convolution kernels in GDL?* The convolution operation in the spectral domain (see appendix A.3) can be expressed as:

$$f \star g = \sum_{k \geq 1} \langle f, \phi_k \rangle_{L^2(\mathcal{X})} \langle g, \phi_k \rangle_{L^2(\mathcal{X})} \phi_k$$

The convolution kernel (g) could have a distribution placed over it, but how can this be achieved? One potential solution is that Bayesian methods could be implemented in a similar way to Bayesian CNNs detailed in section 2.3.4. That is, we may be able to make the convolution operation, which is applied to the non-Euclidean domain, linear. This means that we can just apply dropout after the layer to get a Monte Carlo approximation while making Bayesian inference easy to implement and efficient. However, this is reliant on the fact that the convolution operator in this space can be made linear.

The method within the GDL domain used is also an open question. The common spectral method many of the algorithms use for convolution (see appendix A.2) is a useful introductory model for intuitive understanding of the GDL process, but it has several issues. These include inefficiency in calculating the eigenfunctions of the space, the lack of shift invariance of the convolution filter and the inability to generalise one filter over other domains (the convolution filter will be specific to a single graph/manifold). Many of these issues are alleviated by more advanced methods, such as the inclusion of Chebyshev polynomials, however, they introduce more complexity into the calculations and may make the linearisation (or other method to integrate Bayesian methods) more difficult. The challenge in this section of work will be to find a common general point between many of the algorithms which allows us to place a distribution over the convolution kernels (either through linearisation followed by dropout or another means) in an efficient way.

A potential solution to this comes in the form of MoNet (section 2.4.3). MoNet provides an abstract GDL algorithm which can be adapted to become a more specific instance of an existing Deep Learning or GDL algorithm. This also has a common convolution operation shared between all of the many models. Therefore, it would be of interest to explore this method to determine if it can be made linear, or another method could be implemented to induce a distribution over the convolution kernels. The adaptation of MoNet would lead to an efficient GDL method, but the challenge in this case is to apply a Bayesian method which does not impact on the efficiency of the original algorithm by a large amount.

3.2 Part 2: Attacker Scenario

The application of Bayesian GDL to a realistic attacker scenario provides a good grounding for why this field of research is interesting and necessary. Bayesian methods used in traditional Deep Learning settings provide uncertainty estimates regarding their outputs, this has been used to either detect adversarial examples autonomously or provide human operators with more information about the certainty of the AI system. The aim is to extend this to the general GDL domain.

The suitable scenario must lend itself well to a non-Euclidean structure, for instance applications in computer networks, CFGs or 3-D facial recognition. These applications provide interesting geometric challenges which require the use of GDL methods and all have clear attack and defence motivations which helps to highlight the necessity of this research.

The attacker model can be separated into two parts:

1. Passive attacks – The attacker does not actively generate adversarial examples but hopes the model cannot distinguish benign data from malicious data well

2. Active attacks – The attacker actively tries to cause a misclassification by providing adversarial examples

The passive attack scenario trains a Bayesian GDL model to classify benign and malicious data. For example, in the case of malware detection using CFGs, the model would need to learn to differentiate malware from goodware by analysing graphs. The Bayesian GDL model is not under any direct attack in this scenario, but it is vital that it performs well, can achieve good accuracy from a small amount of data and has accurate uncertainty estimates on all of the data that forms the natural distribution of data (it should have low uncertainty for all of the provided data).

The active attack scenario places the Bayesian GDL model in a more hostile setting. This time, the model will be provided with adversarial examples where malicious data is manipulated to make it look benign. This includes the generation of adversarial examples, which means (depending on the geometry) a new attack may need to be provided, or an existing attack adapted. The uncertainty estimates provided by the Bayesian GDL method will come under scrutiny in this section since adversarial examples should increase the uncertainty of the classifications.

4 Evaluation

The evaluation of the proposed thesis should provide an opportunity to check that the work being carried out is correct and gives some benefit over existing methods. With this in mind, it makes sense to develop evaluation methods for the two distinct goals outlined in the previous section (3):

1. Application of Bayesian methods to Geometric Deep Learning

2. The use of Bayesian Geometric Deep Learning in a realistic attacker scenario

4.1 Part 1: Evaluation of Bayesian GDL

The evaluation of this point will focus on the correctness and efficiency of the proposed Bayesian GDL method. It is vital that the implementation is both correct and efficient to allow the method to be applied to existing systems with minimal changes to code with assurances that it will work as expected.

To test the correctness of the method, we can compare the properties of the proposed method to characteristic properties of Bayesian Neural Networks:

Less training data required: Bayesian variants can often learn accurate models using less training data than their non-Bayesian counterparts. This is often a huge advantage in situations where data is difficult or expensive to obtain and is one of the most popular reasons for using Bayesian variants of networks. With this in mind, the Bayesian GDL method defined in the thesis must have this property. The evaluation of this is simple to construct. We can compare the performance of two models trained to carry out the same task with increasingly less training data being provided. The expectation is that the Bayesian GDL model should start with similar (slightly worse) performance than the non-Bayesian counterpart, but then as the amount of training data provided decreases, the performance of the Bayesian GDL method should become increasingly better than the non-Bayesian counterpart. On the other hand, it has been noted that Bayesian methods do not scale well to large datasets. This disadvantage can be avoided by carefully selecting appropriate implementations, such as using approximate inference methods for example Monte Carlo Dropout [6, 38].

Uncertainty Estimates: Bayesian methods can provide us with uncertainty estimates of the predictions made by the network. This is a particularly important property for detecting adversarial examples, therefore, the ability of the proposed method to generate accurate uncertainty estimates is vital to the implementation of the proposed thesis. The evaluation of this property can be carried out by looking at the uncertainty estimate of expected data (data which forms a part of the natural distribution of data), which should be low, because if the data is expected then we must have seen similar data before with similar classifications. We can then look at unexpected data (data not from the natural data distribution), which can be data we do not expect the model to classify or even adversarial examples. This data should return a high classification uncertainty since the it acts as a proxy for distance from the natural distribution of data.

In addition to checking the properties of the network, we can also analyse the correctness and efficiency by comparing the Bayesian GDL model to existing Bayesian methods. Since the proposed Bayesian GDL model is an abstraction of many specific implementations we should be able to reduce the proposed model to an already existing Bayesian Neural Network (either Bayesian GCNNs [3] or Bayesian CNNs [6]) which should show similar performance in terms of accuracy of predictions and the efficiency of training and executing. As a formal extension to this, it should be possible to show that the proposed Bayesian GDL method is mathematically equivalent to an existing Bayesian neural network when some restrictions are placed on the Bayesian GDL method.

If successful, the newly defined Bayesian GDL method will introduce reliable uncertainty estimates to GDL models. The designer implementing a system will not need to be concerned about using a specific type of GDL model to extract uncertainties as the implementation will be general enough that it does not factor in to the decision making process.

4.2 Part 2: Evaluation of Attacker Scenarios

The aims of this part of the evaluation revolves around the cyber security applications of this project. The overarching goal in this area is to provide an attack scenario which is realistic and has some real world application. To support this, the chosen attacker scenario must have a clear goal for both the attacker and defender of the system, and the system must be implemented in a real world situation. Further to this, the Bayesian GDL method must have some advantage (e.g. better performance, extraction of uncertainty estimates) which means it performs better than previous models deployed in this area, and that the advantages provide some additional security. Much of the discussion in this part comes from the analysis by Gilmer et al. [39].

As previously mentioned, the attacker/defender goals must be clear in this situation and must be believable in the given system. The attack scenarios can be separated into two parts (in the same way from section 3.2): *passive* attacks and *active* attacks. The attacker must be motivated in both of these attack areas.

Passive attacks: The proposed Bayesian GDL method must be able to accurately classify data in a security setting with minimal false positives and false negatives. In addition, the uncertainty estimates about the predictions it makes must be small in this setting since all of the data it is classifying should be expected and part of the natural distribution of data. The key evaluation metric on passive attack data is the ability of the model to accurately classify the data, and the low uncertainty in those classifications.

Active attacks: The proposed Bayesian GDL method must be able to perform well when attacked actively. This includes the addition of adversarial examples in the testing data. The expectation here is not just to show some more robustness to adversarial examples than non-Bayesian models trained to carry out the same task, but to also provide uncertainty estimates of the prediction which indicates the presence of an adversarial example.

The attack scenario must be chosen carefully and evaluated based on a number of metrics:

- Is there a clear motivation for attack/defence? – Can we identify a reason an adversary may attack this system and a reason it should be defended?
- Is an attack on this system feasible? – Does the adversary need to know too much information about the system to launch an attack? Does the system provide any information about the prediction?
- Is the attack perceptible? – Does it matter if the attack is perceptible? What does a perceptible attack mean in the chosen scenario?
- What are the rules of the game? – Is the scenario static i.e. the Bayesian GDL model is created and trained once or can it be adapted and learn from past mistakes? Further to this, how are predictions with high uncertainty handled?

Motivation: The motivation for attacking the system must be clear to the reader. There must be a clear story regarding why we want to defend a network from attacks, why an adversary would try to undermine the security, and the implications of the security being breached must be obvious to the reader.

Feasible attack: The attack on the system must be feasible, and there must be a reason why it's feasible. Many scenarios assume a white-box model without giving any justification as to why we can make this assumption. It is a good method to determine the worst case attack that can be launched, but it does not provide a realistic attack. In addition, many papers use the target model as an oracle where a surrogate model can be trained using the output of the target with the same input. These systems give a lot of information in the output (in many cases needlessly). Therefore, any information given as output in this system must be justified and useful only to non-adversarial users of the system.

Perceptibility: Many adversarial attacks focus on making the attacks imperceptible. This is a useful constraint when considering image classifiers, but it takes a different form when the data we operate on changes. The proposed scenario must have a clear definition of what perceptible is in this sense and whether perceptibility matters for the adversarial examples generated.

Rules of the game: The scenario must have clear "rules of engagement". This specifies whether the attacker and defender have the ability to change their approach during the process. For instance, the Bayesian GDL model may be able to handle adversarial examples in a different way to expected data. This may include providing less information in the output for potential adversarial examples. This can also include limiting the number of requests a user can make to the system.

5 Plan for future work

To support the ideas described in the previous two sections (Thesis Proposal in section 3 and Evaluation in section 4) the timetable for this work will be outlined.

In the immediate future I will be focusing on two tracks of work: (1) Implementation of security focused GDL applications and (2) Incorporation of Bayesian methods into GDL. These points are explored further below:

(1) Implementation of security focused GDL applications – Extending on the work by Yan et al. [35] which uses CFGs to classify malware, it would be interesting to explore the use of adversaries in this context. The adversarial examples could be generated through the Graph based adversarial attacks defined by Zügner et al. [3] or even using traditional methods of adversarial example generation such as FGSM [14]. In a related (but initially seperate) area of work, it would also be useful to explore the use of black-box attacks on GDL models (such as ZOO [19]). The evaluation of black-box methods in the GDL domain has not been seen in the literature yet.

(2) Incorporation of Bayesian methods into GDL – As this is a core focus of the proposed thesis this will be a large piece of work which will take a long time to work through. In the immediate future I will focus on becoming more familiar with the most recent and efficient methods used by GDL to understand how convolution is implemented in these. This will also give me the opportunity to determine if the simple Monte Carlo Dropout method of Bayesian approximation will be valid here.

Longer term goals regarding the thesis are difficult to precisely define, but they can be discussed in broad strokes. The goals include making an efficient and correct Bayesian GDL method which will be applied to a concrete, realistic system which improves its security against adversaries.

The timetable for the project is below along with a graphical representation (figure 3):

Time period	Work
09/20	Work on implementing a malware classifier which uses GCNNs to analyse CFGs of program binaries.
10/20	Implement and understand modern GDL methods (MoNet etc.) which will be built on later. Prepare the six month report for sponsor.
11/20	Implement existing Bayesian GDL methods [3] and evaluate on the malware classifier. This would give a good indication about the usefulness of Bayesian methods in this scenario.
12/20 – 02/21	Perform experiments to qualitatively evaluate the use of Monte Carlo Dropout as a potential way of introducing Bayesian methods in GDL. This time will also be used to write up results of malware classification using existing Bayesian methods and GCNNs. Another sponsor report is also due in during this period.
03/21 – 05/21	The next few months will be dedicated to exploring the effectiveness of existing adversarial attacks on GDL, and also looking at potential adaptations that can be made. Work on making an efficient Bayesian GDL method will also start in this time period. RSMG 4 is also due in during this block of time.
06/21 – 08/21	Work will continue on the efficient implementation of Bayesian methods in GDL. The use of graphs in security systems will also be evaluated, ensuring that the security examples which will form a part of the thesis are suitable and meet the evaluation criteria set out in section 4.2.
09/21 – 02/22	Start formally evaluating the newly implemented Bayesian GDL algorithm ensuring it works as expected and exploits the full power of Bayesian methods. Start evaluating the effectiveness of the new Bayesian GDL algorithm in adversarial situations (against both passive and active adversaries).
03/22 – 08/22	Continue with the evaluation of the Bayesian GDL method and its use in adversarial environments. Finalise the security environment that will be used as a primary example in the thesis (with others to support the proposed Bayesian GDL method). Ensuring that the scenario is well defined.
09/22 – 02/23	Write-up the thesis. Run experiments that need to be carried out and ensure all data is available for evaluation.
03/23 – 08/23	Finalise thesis and submit.
09/23	PhD end date

Figure 3: Timetable for Thesis work

Appendix

A Geometric Deep Learning – A primer

This is a very quick introduction to the mathematics of Geometric Deep Learning which intends to provide some intuition about the topic.

The fundamental parts of Geometric Deep Learning are:

- Express the Laplacian in terms of intrinsic operators of the geometry
- Extract eigenfunctions and eigenvalues from the geometry using the Laplacian
- Express any function we want to apply to the geometry in terms of the eigenfunctions using Fourier transform

The last point in the list allows us to apply a convolutional filter to the space. With some further tweaks to the maths we can gain all of the advantages of convolution over Euclidean domains.

A.1 The Laplacian

The Laplacian is the key to any geometry we work over it can be defined mathematically as:

$$\Delta f(x) = -\text{div}(\nabla f(x)) \tag{1}$$

This is (informally) defined as the difference between the average of the function evaluated at infinitesimal displacements around a point and the value of the function at the point itself. Any space that we can define a Laplacian function over can be used with Geometric Deep Learning. To arrive at the Laplacian we need to define the Riemannian manifold first.

We can define a Riemannian geometry as a manifold $\mathcal{X}$, and a tangent plane $T_x\mathcal{X}$ which is the local Euclidean representation of the manifold around a defined point x on the manifold. The tangent plane allows us to define an inner product (Riemannian metric):

$$\langle \cdot, \cdot \rangle_{T_x\mathcal{X}} : T_x\mathcal{X} \times T_x\mathcal{X} \to \mathbb{R}$$

All quantities that can be described by this metric are called *intrinsic*. We also assume that all functions that are defined on the space are square-integrable (L^2) functions, which means the integral of the square of the absolute value is finite (it avoids complications with the inner product definition).

Using the inner product we can define two functions:

- Scalar Field – $f : \mathcal{X} \to \mathbb{R}$

- Vector Field – $F : \mathcal{X} \to T\mathcal{X}$

In the vector field definition $T\mathcal{X}$ is the tangent bundle (the disjoint union of all tangent spaces).

These allow a Hilbert space to be defined which is vital for spectral decomposition. The inner product over the Hilbert space is defined as:

- Scalar Field –

$$\langle f, g \rangle_{L^2(\mathcal{X})} = \int_{\mathcal{X}} f(x)g(x).\mathrm{d}x$$

- Vector Field –

$$\langle F, G \rangle_{L^2(T\mathcal{X})} = \int_{\mathcal{X}} \langle F(x), G(x) \rangle_{T_x\mathcal{X}}.\mathrm{d}x$$

If we consider equation 1 again, we have an intrinsic gradient ∇ applied to a scalar function f evaluated at point x, then we find the negative intrinsic divergence ($-$div) of this. We will first look at finding the intrinsic gradient.

In a Euclidean domain the gradient can be thought of as the direction and magnitude of greatest increase, which is calculated by taking the partial derivative of a function evaluated at a point with respect to each dimension. It makes sense to define the function analogously on the non-Euclidean domain, so the intrinsic gradient over the manifold takes in a scalar function f and returns a tangent vector which tells us the direction of the steepest change of f:

$$\nabla f : L^2(\mathcal{X}) \to L^2(T\mathcal{X})$$

However, we do not have any definition of what a differential is over a non-Euclidean space. We can (again) approach this problem by analogy. The classical notion of a differential in the Euclidean space is the change in output of a function given an infinitesimal change in input to the function. In a similar sense we can define a differential operator on the non-Euclidean domain as:

$$df : T\mathcal{X} \to \mathbb{R}$$

At each point x the differential is defined as a linear functional (one-form) which is a linear map from a vector space to its field of scalars:

$$df(x) = \langle \nabla f(x), \ \cdot \ \rangle_{T_x\mathcal{X}} \tag{2}$$

This acts on tangent vectors $F(x) \in T_x\mathcal{X}$ which model a small displacement around x. The change in the output of the function comes when we apply df to a tangent vector which acts as the displacement (F):

$$df(x)F(x) = \langle \nabla f(x), F(x) \rangle_{T_x\mathcal{X}}$$

This tells us how much f changes at x in the direction of vector $F(x)$.

The linear functional (one-form) from equation 2 can be identified with a vector in the Hilbert space (due to the nature of duality). This vector is the gradient.

We can now focus on finding the divergence (div) which also appears in the definition of the Laplacian. The intrinsic divergence is defined as:

$$\text{div} : L^2(T\mathcal{X}) \to L^2(\mathcal{X})$$

This means we take in a function which acts on a tangent bundle and return a function which acts on a scalar value. This aligns well with the informal idea that the divergence tells us the net flow of a field F at a point x. The intrinsic divergence can be defined as the formal adjoint (transpose) of the gradient:

$$\langle F, \nabla f \rangle_{L^2(T\mathcal{X})} = \langle \nabla^* F, f \rangle_{L^2(\mathcal{X})} = \langle -\text{div } F, f \rangle_{L^2(\mathcal{X})}$$

A.2 Spectral Domain

With the Laplacian defined we can express the geometry in terms of the spectral domain by extracting the eigenfunctions and eigenvalues from the Laplacian:

$$\Delta\phi_k(x) = \lambda_k \phi_k(x), \quad k = 1, 2, \ldots$$

The eigenvalues are non-negative and are taken in ascending order with the corresponding eigenfunctions also taking this order:

$$0 = \lambda_1 \leq \lambda_2 \leq \ldots$$

The eigenfunctions extracted from the Laplacian are real and orthonormal ($\langle \phi_k, \phi_l \rangle_{L^2(\mathcal{X})} = \delta_{kl}$) which means that they can be used as basis vectors (an eigenbasis) for a space which represents the geometry we're working on.

A.3 Fourier transform and function application

Due to the formation of a space with eigenbasis vectors we are able to express any function in terms of this geometry by using Fourier transforms.

If we have a function f we would like to apply to point x on the geometry then we can apply the Fourier transform:

$$f(x) = \sum_{k \geq 1} \int_{\mathcal{X}} f(x') \phi_k(x') . \mathrm{d}x' \phi_k(x)$$

The integral in the equation above is the same as the inner product, so we can write the expression more simply:

$$f(x) = \sum_{k \geq 1} \langle f, \phi_k \rangle_{L^2(\mathcal{X})} \phi_k$$

This also give a better intuitive understanding of the process. The inner product $\langle f, \phi_k \rangle$ is telling us that we're projecting the function f into the space ϕ_k which allows us to represent f in terms of the eigenbasis. We then convert back from the eigenbasis with the multiplication by ϕ_k at the end.

We can take this one step further and express a convolution operation in a similar way:

$$f \star g = \sum_{k \geq 1} \langle f, \phi_k \rangle_{L^2(\mathcal{X})} \langle g, \phi_k \rangle_{L^2(\mathcal{X})} \phi_k$$

This method allows us to apply convolution operations to any geometry as long as we can represent it using a Laplacian. This conversion to the spectral domain is the key point of Geometric Deep Learning. Unfortunately the representation of convolution above is not great. The convolution kernel (g) is not shift-invariant and the filter coefficients are dependent on the eigenbasis (meaning the filter cannot be used over a different geometry). This has been improved by further work such as MoNet [7], but the representation above is the fundamental basis of these more advanced methods.

A.4 Application to Graphs

Geometric Deep Learning methods are often applied to graphs which have a very similar definition to the manifolds detailed in the previous sections.

We define a graph $\mathcal{G} = (\mathcal{V}, \varepsilon)$ which has vertices $\mathcal{V} = \{1, \ldots, n\}$ and edges $\varepsilon \subseteq \mathcal{V} \times \mathcal{V}$ which are undirected $(i, j) \in \varepsilon \iff (j, i) \in \varepsilon$. All vertices have a weight $a_i \geq 0,\ \forall i \in \mathcal{V}$ and so do all edges $w_{ij} \geq 0,\ \forall (i, j) \in \varepsilon$ and since the graph is undirected $w_{ij} = w_{ji}$.

We can define calculus on the graphs just as we did with manifolds: We have vertex (scalar) fields:

- Vertex (scalar) fields – $f : \mathcal{V} \to \mathbb{R}$
- Edge (vector) fields – $F : \varepsilon \to \mathbb{R}$

We can also define the Hilbert space with inner products:

- Vertex field –
$$\langle f, g \rangle_{L^2(\mathcal{V})} \sum_{i \in \mathcal{V}} a_i f_i g_i$$
- Edge field –
$$\langle F, G \rangle_{L^2(\varepsilon} \sum_{i \in \varepsilon} w_{ij} F_{ij} G_{ij}$$

We can now very easily define a gradient operator

$$\nabla : L^2(\mathcal{V}) \to L^2(\varepsilon)$$

$$(\nabla f)_{ij} = f_i - f_j$$

Which is the difference along an edge.

The divergence operator is defined as:

$$\text{div} : L^2(\varepsilon) \to L^2(\mathcal{V})$$

$$(\text{div } F)_i = \frac{1}{a_i} \sum_{j:(i,j)\in\varepsilon} w_{ij} F_{ij}$$

This is the weighted sum of edges incident on i normalised by a_i.

These allow us to finally reach the Laplacian which is defined as the negative divergence of the gradient. On the graph structure this takes the form:

$$\Delta : L^2(\mathcal{V}) \to L^2(\mathcal{V})$$

$$(\Delta f)_i = \frac{1}{a_i} \sum_{j:(i,j)\in\varepsilon} w_{ij}(f_i - f_j)$$

Which is the difference between f and its local average (the definition of the Laplacian).

The Laplacian for graphs is often represented in matrix form $\mathbf{\Delta}$:

$$\mathbf{\Delta} = \mathbf{A}^{-1}(\mathbf{D} - \mathbf{W})$$

Where $\mathbf{W}$ is the weight matrix, $\mathbf{A}$ is a diagonal matrix of vertex weights and $\mathbf{D}$ is the degree matrix $\text{diag}(\sum_{j\neq i} w_{ij})$ (sum of all incident weights on a vertex).

Just as we can with manifolds, we can use the graph Laplacian to extract the eigenfunctions and eigenvalue from the graph:

$$\mathbf{\Delta\Phi} = \mathbf{\Phi\Lambda}$$

In this expression $\mathbf{\Phi} = (\phi_1, \ldots, \phi_n)$ and $\mathbf{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_n)$. The eigenbasis we can define using the eigenfunction from the graph Laplacian allow us to use the spectral domain for function application in exactly the same way as with manifolds (see section A.3).

References

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, dec 2014.

[2] Y. C. Lin, Z. W. Hong, Y. H. Liao, M. L. Shih, M. Y. Liu, and M. Sun, "Tactics of adversarial attack on deep reinforcement learning agents," in *IJCAI International Joint Conference on Artificial Intelligence*, pp. 3756–3762, mar 2017.

[3] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on neural networks for graph data," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2847–2856, 2018.

[4] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting Adversarial Samples from Artifacts," mar 2017.

[5] Y. Zhang, S. Pal, M. Coates, and D. Ustebay, "Bayesian Graph Convolutional Neural Networks for Semi-Supervised Classification," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 5829–5836, 2019.

[6] Y. Gal and Z. Ghahramani, "Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference," jun 2015.

[7] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model CNNs," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 5425–5434, nov 2017.

[8] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust Physical-World Attacks on Deep Learning Models," jul 2017.

[9] S. Povolny and S. Trivedi, "Model hacking adas to pave safer roads for autonomous vehicles." https://www.mcafee.com/blogs/other-blogs/mcafee-labs/model-hacking-adas-to-pave-safer-roads-for-autonomous-vehicles/, February 2020.

[10] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," in *Proceedings of the ACM Conference on Computer and Communications Security*, vol. 24-28-Octo, pp. 1528–1540, 2016.

[11] F. Monti, F. Frasca, D. Eynard, D. Mannion, and M. M. Bronstein, "Fake News Detection on Social Media using Geometric Deep Learning," feb 2019.

[12] Z. Liu, X. Yun, Y. Zhang, and Y. Wang, *Ringer : systematic mining of malicious domains by graph inference*, vol. 4. Springer International Publishing, 2020.

[13] Y. Bengio, *Learning deep architectures for AI*. Now Publishers Inc, 2009.

[14] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, dec 2015.

[15] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *5th International Conference on Learning Representations, ICLR 2017 - Workshop Track Proceedings*, jul 2019.

[16] S. M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 86–94, oct 2017.

[17] N. Carlini and D. Wagner, "Towards Evaluating the Robustness of Neural Networks," in *Proceedings - IEEE Symposium on Security and Privacy*, pp. 39–57, 2017.

[18] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks," in *Proceedings - 2016 IEEE Symposium on Security and Privacy, SP 2016*, pp. 582–597, nov 2016.

[19] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "ZOO: Zeroth Order Optimization Based Black-Box Attacks to Deep Neural Networks without Training Substitute Models," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, AISec '17, (New York, NY, USA), pp. 15–26, Association for Computing Machinery, 2017.

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

[21] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[22] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *33rd International Conference on Machine Learning, ICML 2016*, vol. 3, pp. 1651–1660, jun 2016.

[23] L. Smith and Y. Gal, "Understanding measures of uncertainty for adversarial example detection," in *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, vol. 2, pp. 560–569, mar 2018.

[24] N. Carlini and D. Wagner, "Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, AISec '17, (New York, NY, USA), pp. 3–14, Association for Computing Machinery, 2017.

[25] N. Papernot, P. Mcdaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proceedings - 2016 IEEE European Symposium on Security and Privacy, EURO S and P 2016*, pp. 372–387, nov 2016.

[26] F. Monti, F. Frasca, D. Eynard, D. Mannion, and M. M. Bronstein, "Fake News Detection on Social Media using Geometric Deep Learning," feb 2019.

[27] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, sep 2017.

[28] T. S. Cohen, M. Weiler, B. Kicanaoglu, and M. Welling, "Gauge equivariant convolutional networks and the icosahedral CNN," in *36th International Conference on Machine Learning, ICML 2019*, vol. 2019-June, pp. 2357–2371, feb 2019.

[29] D. Boscaini, J. Masci, E. Rodolà, and M. Bronstein, "Learning shape correspondence with anisotropic convolutional neural networks," in *Advances in Neural Information Processing Systems*, pp. 3197–3205, may 2016.

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

[31] J. Masci, D. Boscaini, M. M. Bronstein, and P. Vandergheynst, "Geodesic Convolutional Neural Networks on Riemannian Manifolds," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*, dec 2015.

[32] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Advances in neural information processing systems*, pp. 1993–2001, 2016.

[33] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Information Retrieval*, vol. 3, no. 2, pp. 127–163, 2000.

[34] C. L. Giles, K. D. Bollacker, and S. Lawrence, "Citeseer: An automatic citation indexing system," in *Proceedings of the third ACM conference on Digital libraries*, pp. 89–98, 1998.

[35] J. Yan, G. Yan, and D. Jin, "Classifying Malware Represented as Control Flow Graphs using Deep Graph Convolutional Neural Network," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 52–63, jun 2019.

[36] C. Shah, N. Dehmamy, N. Perra, M. Chinazzi, A.-L. Barabási, A. Vespignani, and R. Yu, "Finding Patient Zero: Learning Contagion Source with Graph Neural Networks," jun 2020.

[37] J. Svoboda, J. Masci, F. Monti, M. M. Bronstein, and L. Guibas, "Peernets: Exploiting peer wisdom against adversarial attacks," in *7th International Conference on Learning Representations, ICLR 2019*, may 2019.

[38] Y. Gal, *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

[39] J. Gilmer, R. P. Adams, I. Goodfellow, D. Andersen, and G. E. Dahl, "Motivating the Rules of the Game for Adversarial Example Research," jul 2018.