Bayesian Graph Representation Learning for Adversarial Patch Detection

Alexander M. Berenbeim[a], Alexander V. Wei[a], Adam Cobb[b], Anirban Roy[b], Susmit Jha[b], and Nathaniel D. Bastian[a]

[a]Army Cyber Institute, United States Military Academy, West Point, NY, USA
[b]Computer Science Laboratory, SRI International, Menlo Park, CA USA

ABSTRACT

Representing context, reasoning within contexts, and providing quantitative assessments of machine learning (ML) model certainty are all tasks of fundamental importance for secure, interpretable, and reliable model development. Recent enthusiasm regarding generative ML models has highlighted the importance of representing context, which is contingent on relevant and contextual features of data and model predictions are unreliable on out-of-context inputs. Herein, we develop the theory of graph representation learning (GRL) to extend to Bayesian Graph Neural Networks and to incorporate various forms of uncertainty quantification to improve model development and application in the presence of adversarial attacks. Within this framework, we approach the challenge of adversarial patch detection using a synthesized dataset consisting of images from the APRICOT and COCO datasets to study various binary classification models for patch detection. We present GRL models with two layers of edge convolution that are capable of detecting patches with up to 93.5% accuracy. Further, we find evidence supporting the use of the certainty and competence framework for model predictions as a tool for detecting patches, particularly when the former is included as a model feature in graph neural networks.

Keywords: Graph Representation Learning, Uncertainty Quantification, Adversarial Patch Detection

1. INTRODUCTION

The recent explosion in the power and performance of generative machine learning models presents profound opportunities and challenges. On one hand, the latent semantic space of large language models can be leveraged to establish priors concerning expert knowledge captured by the training set.[1,2] In particular, from a human-agent point-of-view, accessing this expert knowledge depends critically on the representation and coding of *context*.[3,4]

Importantly, generative models present great challenges to several safety-critical tasks, such as object detection, whose models have been shown to be vulnerable to adversarial attacks.[5,6] Specifically, existing methods for detecting adversarial attacks perform poorly when detecting novel, out-of-distribution attacks.[7,8] Generative models can exacerbate this challenge by in-filling images with adversarial patches.[9,10]

Outside of in-context learning, models scale and fare less well, in part because the transformers and weights of the model are hampered by closed-world assumptions, a lack of a formal logic for abducting and deriving rules, and a lack of uncertainty quantification that may otherwise condition a logic for implementing these models in production environments where the data encountered is often different from that of the in-distribution training data. In this paper, we address the challenge of adversarial patch detection to contextual learning by developing the theory of Bayesian Graph Neural Networks and incorporating some forms of uncertainty quantification in the form of the certainty and competence scoring framework[11,12] to furnish additional node-level features for our GNNs. The proposed extension of deterministic graph representation learning to a Bayesian framework is done so in order to enable other forms uncertainty quantification which allow for out-of-context and out-of-distribution detection. For now, we examine binary discriminative models for direct adversarial patch detection.

We summarize Bayesian approaches to the general framework of graph neural networks (GNNs), which can be understood probabilistically as *Markov random fields* and *Hilbert space embeddings.* While inferring the distribution of latent representations for a Markov random field is almost always computationally intractable, several

Send correspondence to: nathaniel.bastian@westpoint.edu

approximate methods such as Mean-field variational inference (MVI) are equivalent to message passing over a graph.[13] Although various dropout methods can be used for approximating Bayesian sampling of graphs, we focus on generalizing the Markov random field formalism and discuss MVI as well as physics-based approximate methods when generalizing to Bayesian GNNs. In determining the efficacy of using GNNs and the aforementioned certainty framework for patch detection, we examined several different binary classification models, both within deterministic and stochastic paradigms, and found experimental support that graph convolutional neural networks incorporating object classification certainty scores as node-level features are capable of patch detection.

The organization of this paper is structured as follows. Section 2 provides a brief overview of some related literature, while Section 3 highlights some preliminaries. Section 4 provides an overview of our proposed methodology. Results are provided and discussed in Section 5, while Section 6 concludes by summarizing the findings, discussing the implications, and outlining future work.

2. RELATED WORKS

Graph representation learning and graph neural networks have seen a recent explosion in the past decade. Hamilton[13] collects and presents the fundamentals of graph representation learning and the five chief tasks of the graph neural network: node classification, edge classification, edge detection, community detection, and graph classification. GNNs are presented as neural network architectures that take a graph structure $G = (V, E)$ on input, generally in the form of a triple consisting of node features x_v for all nodes $v \in V$, edge features $\mathcal{F}_{(u,v)}$ for all edges set $(u, v) \in E$, and an adjacency matrix $\mathbf{X}$ defined by the set E. GNNs are first presented as extensions of convolutions to graph-structured data, whose outputs can be understood as low-dimensional graph embeddings. The graph embeddings may be understood as latent variables used to explain the observed graph structure of the node and edge features, as well as the adjacency matrix. Hamilton demonstrates how to understand the aggregated information from embeddings of neighboring nodes to update a node embedding as a form of neural message passing. Then following Dai et al.[14] , Hamilton describes how basic GNNs can be understood in relation to probabilistic graph models (PGMs) through message-passing algorithms used for variational inference to infer distributions over latent variables.

Two key intuitions highlighted by Hamilton are that stacking multiple rounds of message passing in a basic GNN is analogous to applying a low-pass convolutional filter, and that the approximate methods used to compute the posteriors of Markov random fields, specifically those satisfying fix-point solutions, amount to message-passing operating over distributions rather than embeddings. This latter insight indicates we can learn an embedding μ_v that could correspond to some PGM as an alternative to trying to specify a concrete probabilistic model, so that GNNs are not only generalizations of convolutions to graph-structured data, but that they are related to the approximate solutions to variational inference in PGMs. Literature generalizing GNNs to the Bayesian setting build on the insight that observed graph data can be understood as a parametric family of random graphs.

At this time in pre-print, Carr and Wingate[15] introduce Graph Neural Processes (GNP) as a conditional and latent neural process operating on graph data for the task of edge classification. Carr and Wingate establish that the convolution operation used in traditional deep learning neural network architectures can be replaced with features used in GNNs, demonstrating how to take sparsely observed context points as input to output a distribution over target points. The authors further highlight the importance of this generalization on the grounds that application of traditional convolutional techniques to graphs are inappropriate because the former rely on assuming that data is Euclidean, while the latter do not lie on regular lattices. Finally, the authors contribute to the subject of uncertainty quantification in Bayesian networks, identifying that uncertainty estimates are either used at the weight level or at the output level of the network.

Zhang et al.[16] proposed Bayesian Graph Convolutional Networks for node classification tasks, viewing the observed graph data as a realization from a parametric family of random graphs. Zhang et al. motivate the Bayesian approach in order to incorporate uncertainty quantification (UQ), as graph data is often noisy, and subject to arbitrary modeling assumptions which may introduce edges spuriously, or otherwise miss fundamental relationships. Their experimental results of their assortative mixed-membership block model demonstrate reliable performance when few labels are available during the training process.

Hasanzadeh et al.[17] identify two major limitations faced by GNNs: 1) over-smoothing and over-fitting prevent deep networks; 2) non-Bayesian GNN frameworks do not provide uncertainty quantification of output predictions. Building off the Bayesian framework introduced by Zhang et al.[16] for graph convolutional networks, which presented an observed graph as a realization from a parametric family of random graphs, Hasanzadeh et al. proposed a unified framework for adaptive connection sampling in graph neural networks generalizing existing stochastic regularization methods, chiefly DropConnect, as Graph DropConnect. Further, Hasanzadeh et al. confirmed experimentally with ablation studies on benchmarks that adaptively learning sample rates for semi-supervised node classification tasks made GNNs less prone to over-fitting and over-smoothing. While their proposed implementation only depends on the graph topology, it cannot consider node features.

Pal et al.[18] provide a Bayesian framework targeting the posterior inference of a graph in order to address issues of inaccurate modeling assumptions and noisy graph data. Specifically, their non-parametric graph model is used to construct the posterior distribution of graph adjacency matrices. Further, they discuss the application of their non-parametric model for the node classification, link prediction, and recommendation tasks (which is a variation of link prediction). Object detection and classification neural networks are particularly important examples of data that are amenable to graph representation learning and sensitive towards adversarial attacks perturbing context. In particular, we take focus on extracting graphs from the Faster R-CNN model,[19] which simultaneously predict object bounds and objectness scores for predicted regions. Trained on the MS COCO data,[20] the Faster R-CNN model is publicly available, and the foundation of several other successful object detection networks, and was thus the most natural candidate to use in our research.

In contrast with the MS COCO dataset, we relied on the APRICOT dataset,[5] to furnish images with adversarial patches. APRICOT is a collection of over 1000 annoted photographs with printed adversarial patches in public locations that target several object categories for three COCO-trained detection models. These particular patches are designed to fool object detection systems by perturbing context, and the variety of patches represent natural variations in position, distance, lighting conditions, and viewing angle. Braunegg et al.[5] found that adversarial patches can be effectively flagged, both in a high-knowledge, attack-specific scenario, and in an unsupervised setting where patches are detected as anomalies in natural images. Our fundamental hypothesis is that patches may be detected through a combination of UQ and graph representation learning by assessing when object detection and classification models are uncertain of individual objects, and whether those individual objects effectively act on the surrounding certainties of their neighbors, measurably transforming them.

Finally, we considered several approaches to UQ that would be relevant for adversarial patch detection. Zhang et al.[21] examined several scores and approaches for detecting adversarial data from the MNIST,[22] CIFAR-10,[23] and ImageNet[24] datasets, among the most effective scores were the predictive confidence scores (PCS) and variation rate from original (VRO). Zhang et al.[21] found that PCS could be used to distinguish between benign and adversarial examples in general, and in Bayesian settings, the VRO was most effective when comparing prediction stability between original model and multiple dropout-enabled predictions. Similarly, Berenbeim et al.[11] and Wong et al.[12] developed the certainty and competence framework to assess the quality of discriminative model predictions. In particular, Wong et al.[12] applied this framework to network traffic data to separate true positive classifications from false positive classifications, as well as inputs that are out-of-distribution relative to training data. Although the certainty score defined in Berenbeim et al. and Wong et al. is definitionally identical to PCS, the certainty of a model prediction is an invertible matrix and is defined as an intrinsic multi-dimensional property of any probability vector. We offer an additional hypothesis that the certainty derived from identified adversarial patches will be distributed differently from the certainties for COCO related objects.

3. PRELIMINARIES

3.1 (Bayesian) Neural Networks

A *statistical learning machine* is a function $f(\mathbf{x}, \omega)$ taking features $\mathbf{x}$ from some real-valued feature space $\mathcal{F}$ and parametrized by real-valued parameters ω. In general, the ℓ^{th} *neural network layer* $f^{(\ell)}(\mathbf{x}^{(\ell-1)}, \omega^{(\ell-1)})$ defined in Equation 1 is the composition of a (typically non-linear) activation function with a linear function combining known input features $\mathbf{x}$ with previously defined layers $\mathbf{z}^{(\ell-1)}$.

$$f^{(\ell)}(\mathbf{x}, \mathbf{z}^{(\ell)}, \omega^{(\ell-1)}) = \textsc{Activation}(\textsc{Message}(\mathbf{x}, \mathbf{z}^{(\ell-1)}, \omega^{\ell})) \tag{1}$$

A common rendering of Equation 1 would substitute Activation with an activation function like tanh or ReLU, or possibly the SoftMax function, denoted by σ below, and explicitly representing the message in linear terms

$$f^{(\ell)}(\mathbf{x}, \mathbf{z}^{(\ell-1)}, \mathbf{W}^{(\ell)}) = \sigma(\mathbf{W}_{\mathbf{x}}^{(\ell)}\mathbf{x} + \mathbf{W}_{\mathbf{z}^{(\ell-1)}}^{(\ell)}\mathbf{z}^{(\ell-1)} + \mathbf{b}^{(\ell)}) \tag{2}$$

where $\mathbf{z}^{(\ell-1)}$ is the output of $f^{(\ell-1)}$, the parameter ω is rendered explicitly in terms of parameter weights $\mathbf{W}$ given to the input features and the hidden (latent) layers $\mathbf{z}^{(\ell-1)}$, and $\mathbf{b}$ denotes a bias term. Weights are primarily determined through backpropagation and with respect to some given loss function $\mathcal{L}$. Without loss of generality, we denote by f the neural network and by $\mathbf{z}^{(\ell)}$ the ℓ^{th} layer of f. We consider a *deep neural network* to be a function $f(\mathbf{x}, \mathbf{W})$ formed by at least two neural network layers.

An attractive feature of Bayesian Neural Networks (BNNs) is that they can capture the model uncertainty of their frequentist counterpart DNNs. This is done in practice by placing prior distributions over the model parameters $\mathbf{W}$, so that the posterior updates while training the DNN. Hasanzadeh et al.[17] furnish additional examples where unlike traditional DNNs, the Bayesian extension is robust to overfitting.

In the following subsections, we motivate and discuss several approaches to approximate Bayesian inference which is in general computationally intractable. We rely on exponential families of density functions:

$$p(\mathbf{x};\vartheta) = \exp(\vartheta^\top\phi(\mathbf{x}) - A(\vartheta))\mu(\mathbf{x}) \tag{3}$$

where ϑ are the canonical parameters specifying the distribution, $\gamma(\mathbf{x})$ is a function on $\mathbf{x}$, $\phi(\mathbf{x})$ denotes the linearly independent sufficient statistics of $\mathbf{x}$, $\mu(x)$ a measure on the underlying sample space Ω and $A(\vartheta)$ is a finite log partition function,

$$A(\vartheta) = \log\int \exp(\vartheta^\top\phi(\mathbf{x}))\mu(d\mathbf{x}) < \infty.$$

We motivate our focus on using exponential families because: (1) the connection between combining Hilbert space embeddings, sufficient statistics, and graph neural message passing (see Section 3.3) with the Pitman-Koopman-Darmois theorem; (2) exponential families have nice conjugate priors distributions, which helps facilitate Bayesian analysis; (3) they furnish us with a general space of energy functions to consider for the general graph learning tasks we will consider.

3.2 Graph Neural Networks

Definition 3.1 (Graphs). *A graph is defined as the pair $G = (V, E)$, where V is the set of nodes and E is set of edges. Let $v_i \in V$ denote a node and $e_{ij} = (v_i, v_j) \in E$ denote an edge between node v_i and node v_j. A graph is complete if there is an edge for every pair of nodes. A clique of G is a either a single node, or a complete subgraph of G. Let $\mathcal{C}$ denote the set of cliques for a graph G.*

The degree of a node describes the number of edges shared by a node. An undirected graph is one where each edge corresponds to an unordered pair of nodes, and a directed graph is one where each edge corresponds to an ordered pair of nodes. $\boldsymbol{A} \in \{0,1\}^{n\times n}$ represents the adjacency matrix of graph G, where $n = |V|$ is the total number of nodes. When $\boldsymbol{A}_{ij} = 1$, there exists an edge from node v_i to node v_j, otherwise $\boldsymbol{A}_{ij} = 0$. When G is undirected, $\mathbf{A}$ is always symmetric. A k-path in G is an ordered sequence of nodes n_i such that each pair $(n_i, n_{i+1}) \in E$. For disjoint triples $X, Y, Z \subset V$, we say that Z separate X for Y if any path from a node in X to Y passes through at least one node in Z.

The neighborhood of node v_i as $N(v_i) := \{v_j \in V \mid (v_i, v_j) \in E\}$. The k^{th}-hop neighborhood is iteratively defined as $N_1(v_i) = N(v-i)$ and

$$N_k(v_i) := N_{k-1}(v_i) \cup \{v_k \in V \backslash N_{k-1}(v_i) \mid \exists v_j \in N_{k-1}(v_i).(v_k, v_j) \in E\}.$$

For graph data with node features, we use $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ to denote the node feature matrix where d is the number of node features. For graph data with edge features, we use $\boldsymbol{\mathcal{E}}$ to denote the $N \times N \times \tau$ tensor of edge features, where τ is the dimension of the edge features. Specifically, $\mathcal{E}_{ij\cdot}$ is the τ-dimensional feature vector of an edge between node v_i and node v_j, and without loss of generality $\mathcal{E}_{ij\cdot} = \mathbf{0}$ if and only if there is no edge between v_i

and v_j, so that we may assume that $\mathcal{E}_{\cdot\cdot k}$ are square, non-negative real matrices. In practice, $\mathcal{E}$ is formed directly from the adjacency matrix $\mathbf{A}$ and the feature vectors from each edge. However, following Gong et al.,[25] in order to stabilize the learning process when iteratively learning over edge features, we may assume that $\mathcal{E}$ has been normalized from the raw features $\widehat{\mathcal{E}}$ via

$$\widetilde{\mathcal{E}_{ijk}} = \frac{\widehat{\mathcal{E}_{ijk}}}{\sum_{\ell=1}^{N} \widehat{\mathcal{E}_{i\ell k}}} \qquad \textit{and} \qquad \mathcal{E}_{ijk} = \sum_{\ell=1}^{N} \frac{\widetilde{\mathcal{E}_{i\ell k}}\widetilde{\mathcal{E}_{j\ell k}}}{\sum_{m=1}^{N} \widetilde{\mathcal{E}_{m\ell k}}}.$$

The normalization for the edge features proposed in the above definition produces a stationary finite Markov chain with double stochastic transition matrices for each $\mathcal{E}_{\cdot\cdot k}$, so that there will be a uniform stationary distribution for each $\mathcal{E}_{\cdot\cdot k}$. Gong et al.[25] demonstrate the utility of this normalization procedure as it generalizes both graph convolutional networks and graph attention networks.

We summarize the general GNN construction found in Hamilton[13] and Battaglia et al.[26] A basic GNN $\mathbf{f}(G,\omega)$ is a real-valued vector function that aggregates information from local neighborhoods to construct an embedding representing nodes, edges, or the graph as a whole. In keeping with the embedding output notation, we denote by $\mathbf{z}_v$ a node embedding for vertex v, $\mathbf{z}_{(u,v)}$ an edge-embedding for an edge (u,v), and $\mathbf{z}_G$ for the graph-level embedding output for graph G after applying a pooling function.

In general, loss functions are defined with respect to the embedding outputs $\mathbf{z}.$, and it will be assumed that the gradient of the loss will be backpropagated through the parameters of the GNN using stochastic gradient descent. The iteration of basic message-passing GNNs captures the *structural* information and *features* of a graph. In practice, $\mathbf{z}.^{(\ell)}$ corresponds to hidden embeddings for all layers ℓ except for $\ell = 0$, which corresponds to the input features, and $\ell = L$, the final layer of the GNN. At each iteration ℓ of the GNN, messages are passed through the network first by aggregating neighborhood information, and then updating the message accordingly.

Hamilton presents the basic GNN for a node embedding in the form of

$$\mathbf{z}_v^{(\ell)} = \sigma\left(\mathbf{W}_v^{(\ell)}\mathbf{x}_v + \mathbf{W}_{N(v)}^{(\ell)} \sum_{u\in N(v)} \mathbf{z}_u^{(\ell-1)} + \mathbf{b}^{(\ell)}\right). \tag{4}$$

Equation 4 describes the ℓ-layer of a GNN, with $\mathbf{W}_v$ the trainable parameter matrices for nodes v, $\mathbf{W}_{N(v)}$ the trainable parameter matrices for the neighborhoods of v, $\mathbf{b}$ a bias term, and σ an elementwise nonlinearity.

This construction can be generalized with differentiable functions Aggregate and Update. The former function captures the structural information of the ℓ-hop neighborhood for each vertex u, generalizing the term $\sum_{u\in N(v)} \mathbf{z}_u^{(\ell-1)}$ with the *message*

$$\mathbf{m}_{N(u)}^{(\ell-1)} = \textsc{Aggregate}(\{\mathbf{z}_u^{(\ell-1)} \mid u \in N(v)\}).$$

The latter function collects the aggregated information and node features so that we may express

$$\mathbf{z}_v^{(\ell)} = \sigma(\textsc{Update}(\mathbf{z}_v^{(\ell-1)}, \mathbf{m}_{N(v)}^{(\ell-1)}; \mathbf{W}_v^{(\ell)}, \mathbf{W}_{N(v)}^{(\ell)})).$$

For the sake of readability, we will always assume that Update functions will have corresponding parameters $\mathbf{W}$ for the features invoked in argument, and may thus omit them.

Similarly to the node message-passing model in Equation 4, the construction of the basic GNN message-passing model at the graph level for constructing graph-embeddings is given by:

$$\mathbf{z}_G^{(\ell)} = \sigma(\mathbf{A}\mathbf{z}_G^{(\ell-1)}\mathbf{W}_{neigh}^{(\ell)} + \mathbf{z}_G^{(\ell-1)}\mathbf{W}_{self}^{(\ell)} + \mathbf{b}). \tag{5}$$

Each $\mathbf{z}_G^{(\ell-1)} \in \mathbb{R}^{|V|\times d}$ in Equation 5 denotes the matrix of node representations at layer $\ell - 1$.

We generalize the construction of arbitrary embeddings defined by recursing through the following four relations:

$$\mathbf{z}_{(u,v)}^{(\ell)} = \textsc{Update}_{edge}\left(\mathbf{z}_{(u,v)}^{\ell-1}, \mathbf{z}_u^{(\ell-1)}, \mathbf{z}_v^{(\ell-1)}, \mathbf{z}_G^{(\ell-1)}\right) \tag{6}$$

$$\mathbf{m}_{N(v)}^{(\ell)} = \textsc{Aggregate}_{node}\left(\{\mathbf{z}_{(u,v)}^{(\ell)} \mid u \in N(v)\}\right) \tag{7}$$

$$\mathbf{z}_v^{(\ell)} = \textsc{Update}_{node}\left(\mathbf{z}_v^{(\ell-1)}, \mathbf{m}_{N(v)}^{\ell}, \mathbf{z}_G^{(\ell-1)}\right) \tag{8}$$

$$\mathbf{z}_G^{(\ell)} = \textsc{Update}_{graph}\left(\mathbf{z}_G^{(\ell-1)}, \{\mathbf{z}_v^{(\ell)} \mid v \in V\}, \{\mathbf{z}_e^{(\ell)} \mid e \in E\}\right) \tag{9}$$

Equation 6 describes the general construction for edge features and GNN message passing in general when we start with $\mathcal{E}$. $\textsc{Update}_{edge}$ is a function that updates edge embeddings based on the embeddings of incident nodes. The message being passed by the individual neighborhood of v is then updated with Equation 7, after which we update the corresponding node embeddings in Equation 8. Finally, the ℓ-layer Graph embedding is defined by Equation 9.

For general tractability, we introduce a basis $\mathcal{B} = (\mathbf{B}_1, \ldots, \mathbf{B}_b)$, so that the message-updating structure can be represented by

$$\mathbf{m}_{N(v)}^{(\ell)} = \sum_{k=1}^{\tau} \sum_{u \in N_i(v)} \frac{\alpha_i \otimes \mathcal{B} \otimes \mathbf{z}_v^{(\ell)}}{f_n(N(u), N(v))} \tag{10}$$

where $\alpha_i = [\alpha_{1,i}, \ldots, \alpha_{b,i}]^\top$ is a vector of basis combination weights for the edge feature/relation i, and f_n are normalization functions. This implementation can allows us to compactly decompose $\mathbf{W}_i$, the parameter weight matrices for relations i, so that we may learn parameters for relation i as expressed by $\mathbf{W}_i = \sum_{j=1}^{b} \alpha_{j,i}\mathbf{B}_j$.

With these definitions in mind, *graph neural networks* refer to the subclass of neural networks distinguished by (1) taking graph structured data as an input, and (2) using a generalized, differentiable operations on non-Euclidean spaces for the purpose of message-passing. The five primary tasks for which GNNs are trained are: (1) Node Classification; (2) Edge Classification; (3) Graph Classification; (4) Edge Detection; (5) Community Detection. Effectively, there are two meta-tasks: Classification and Detection, which in turn involve different formalisms and algorithmic implementations. However, *edge detection* (alternatively, *link prediction*) and *edge classification* can both be encompassed by *relation prediction*, where the task is to identify if a relationship between two nodes exists (detection) and what kind (classification), the latter which may amount to predicting the corresponding edge features.

3.2.1 Graphs and Markov random fields

Definition 3.2 (Random Fields). *Let X be drawn from a given probability space $(\Omega, \mathcal{F}, \mathbb{P})$. A random field on X is a stochastic process f indexed by a parameter space T with a given topology such that for each $t \in T$, f_t takes a value $X \in \Omega$.*

Let G be a random graph drawn from a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ where Ω is a class of undirected graph. Further let $\mathbf{f}$ denote a random field on an undirected graph G such that we assign a random variable f_i to each each node $i \in V$ so that we may consider the joint probability function $\mathbb{P}\{\mathbf{f}\} = \mathbb{P}\{f_1, \ldots, f_{|V|}\}$. For every disjoint triple $X, Y, Z \subset V$, let $\mathbf{f}_X, \mathbf{f}_Y, \mathbf{f}_Z$ denote the projection of $\mathbf{f}$ onto the corresponding three disjoint subsets. If whenever Z separates X from Y,

$$\mathbb{P}\{\mathbf{f}_X, \mathbf{f}_Y \mid \mathbf{f}_Z\} = \mathbb{P}\{\mathbf{f}_X \mid \mathbf{f}_Z\}\mathbb{P}\{\mathbf{f}_Y \mid \mathbf{f}_Z\}. \tag{11}$$

then $\mathbb{P}\{\mathbf{f}\}$ has the global Markov property.

We say $\mathbf{f}$ has the pair-wise Markov property if for each $(u, v) \notin E$, f_i and f_j are independent whenever conditioned on the remaining variables.

We say $\mathbf{f}$ *has the local Markov property if for all nodes* i *in* V*,*

$$\mathbb{P}\{f_i, \mathbf{f}_{V\setminus(\{i\}\cup N(i))} \mid \mathbf{f}_{N(i)}\} = \mathbb{P}\{f_i \mid \mathbf{f}_{N(i)}\}\mathbb{P}\{\mathbf{f}_{V\setminus(\{i\}\cup N(i))}\}. \tag{12}$$

If $\mathbf{f}$ *satisfies any of these properties, then* $\mathbf{f}$ *is a Markov random field.*

Finally, whenever $\mathbf{f}$ *is locally Markovian, then* $\mathbb{P}\{\mathbf{f}\}$ *can be defined as a Gibbs distribution*

$$p(\mathbf{f}) = \frac{1}{A}\exp\left\{-\sum_{C\in\mathcal{C}} V_C(\mathbf{f}_C)\right\} \tag{13}$$

where A *is the value of the partition function, and* V_C *are energy functions for each clique* C *defining the clique potential function.*

We specify Markov random fields in terms of energy functions V_C, where each $\mathbf{f}_C = \mathbf{x}_C \cup \mathbf{z}_C$. Random graph models $G = (V, E)$ can be defined as Markov random field over node features and node embeddings that factorize over the graph structure with non-negative clique potential functions χ and ψ for vertices and edges respectively:

$$p(\mathbf{f}) \propto \prod_{v\in V}\chi(x_v, z_v)\prod_{(u,v)\in E}\psi(z_u, z_v). \tag{14}$$

Of note, the MRF described in Equation 14 does not depend on cliques beyond edges from our graph G, i.e. for every clique C such that $|C| > 2$, the corresponding energy $V_C(\cdot) = 0$. This coincides with pairwise interactions only, forming the so-called *auto-model*.

Theorem 3.3. *The node-level representation of a graph neural network with hidden edge features is a Markov random field with potential functions ranging over all cliques, and*

$$p(\mathbf{f}_V, \mathbf{f}_E, \mathbf{f}_G) \propto \left(\prod_{v\in V}\chi(x_v, z_v)\prod_{(u,v)\in E}\psi_V(z_u, z_v)\psi_E(x_{(u,v)}, z_{(u,v)})\prod_{C\in\mathcal{C}:|C|>2}\lambda_V(z_C)\lambda_E(z_{\{(u,v):u,v\in C\}})\right)\zeta(x_G, z_G). \tag{15}$$

In generality, we will consider the graph $G = (V, E)$ as defining a random field $\mathbf{f} = \mathbf{f}_V \cup \mathbf{f}_E \cup \mathbf{f}_G$ where the projection onto $\mathbf{f}_V$ is Markov random field. Here $\mathbf{f}_E$ projects onto the edge features and similarly $\mathbf{f}_G$ onto graph-level features. For cliques $C \in \mathcal{C}$ such that $|C| > 2$, $f_C = z_C$. The extension of Equation 14 to Equation 15 describes the generalization from a node-level graph neural network with hidden edge features to a general Markov random field structure with potential functions ranging over all cliques.

When specifying a Markov random field for the other tasks, this requires considering the joint distributions of the relevant features and latent variables. Inference is then conducted within a hierarchical Bayesian setting with corresponding parameters for the observation model and priors respectively.

3.3 Hilbert space embeddings

Hamilton develops an analogy between GNN message passing algorithms and Hilbert space embeddings of distributions that we pick up on in our work. Specifically, given an arbitrary dimensional feature map $\phi : \mathbb{R}^m \to \mathcal{R}$, the probability density $p(\mathbf{x})$ can be represented by its expected value under ϕ

$$\mu_{\mathbf{x}} = \mathbb{E}[\phi(\mathbf{x})] = \int_{\mathbb{R}^m}\phi(\mathbf{x})p(\mathbf{x})d\mathbf{x} \tag{16}$$

For suitable ϕ, the map $\mu : \mathbb{R}^m \mapsto \mathcal{R}$ induced by Equation 16 will be injective with $\mu_{\mathbf{x}}$ serving as a sufficient statistic for $p(\mathbf{x})$. While the mean-field message passing equations operate over distributions instead of embeddings, when considering mean-field inference, we leverage the Hilbert space embeddings, and the fixed-point iteration for solving the variational equation relation (see Equation 18) exactly corresponds to neural message passing over a graph.

3.3.1 Mean-field Variational Inference

Mean-field variational inference approximates the posterior using density functions q_v based on the assumption that the posterior distribution over the latent variables $\mathbf{z}_V$ factorizes into $|V|$ independent distributions, e.g. $p(\mathbf{z}_V \mid \mathbf{x}_V) \approx q(\mathbf{z}_V) = \prod_{v \in V} q_v(\mathbf{z}_v)$. Optimal approximate q_v are obtained by minimizing the Kullback-Leibler divergence between the approximate and true posterior. Densities $q_v(\mathbf{z}_v)$ which minimize KL divergence satisfy:

$$\log(q_v(\mathbf{z}_v)) = c_v + \log(\chi(\mathbf{x}_v, \mathbf{z}_v)) + \sum_{u \in N(v)} \int_{\mathbb{R}^d} q_u(\mathbf{z}_u) \log(\psi(\mathbf{z}_u, \mathbf{z}_v)) d\mathbf{z}_u \tag{17}$$

The fixed point solutions to mean-field variational inference in Equation 17 are approximated by iteratively computing:

$$\log(q_v^{(\ell)}(\mathbf{z}_v)) = c_v + \log(\chi(\mathbf{x}_v, \mathbf{z}_v)) + \sum_{u \in N(v)} \int_{\mathbb{R}^d} q_u^{(\ell-1)}(\mathbf{z}_u) \log(\psi(\mathbf{z}_u, \mathbf{z}_v)) d\mathbf{z}_u. \tag{18}$$

Hamilton[13] presents the connection between GNN message passing and Hilbert embeddings by noting that given an injective feature map and independent marginal densities $q_v(\mathbf{z}_v)$, the solution to Equation 16 can be rewritten using fixed point iteration as

$$\mu_v^{(\ell)} = \mathbf{c} + f(\mu_v^{(\ell-1)}, \mathbf{x}_v, \{\mu_v \mid u \in N(v)\})$$

where f is formed by composing aggregation and update functions through message passing.

3.3.2 Graph-based Hamiltonian Monte Carlo

One of the most popular inference schemes deployed to learn the posterior distribution of a neural networks model parameters is the Markov Chain Monte Carlo (MCMC) scheme using Hamiltonian Monte Carlo (HMC).[27,28]

HMC sampling is a first-order approximate simulation of Hamiltonian dynamics based on numerical integration, followed by a Metropolis acceptance step for correction. In our case, HMC sampling starts from an unnormalized log posterior defined by the likelihood $p(\mathbf{Y} \mid \mathbf{X}, \omega)$ and prior $p(\omega)$. The likelihood is a function of the parameters, $\omega \in \mathbb{R}^D$, which in our case, will correspond to z_C as the weights of the Bayesian graph neural network models.

We consider the graph-classification task as the first step towards general graph-based HMC. At minimum, this amounts to substituting $\mathbf{Y}$ in the likelihood function $p(\mathbf{Y} \mid \mathbf{X}, \omega)$ with the graph features x_G. For our initial purposes, we consider treating the graph convolution layers as a fixed embedding, i.e. we substitute $\mathbf{X}$ with

$$\left(\prod_{v \in V} \chi(x_v, z_v) \prod_{(u,v) \in E} \psi_V(z_u, z_v) \psi_E(x_{(u,v)}, z_{(u,v)}) \right) \prod_{C \in \mathcal{C} : |C| > 2} \lambda_V(z_C) \lambda_E(z_{\{(u,v) : u,v \in C\}})$$

and z_G with ω from Equation 15. However, we note that for the time being, all clique potentials are set to 1 for cliques with 3 or more vertices.

The Bayesian model function is then transformed into a Hamiltonian system by introducing a momentum variable $\mathbf{p} \in \mathbb{R}^D$, producing a log joint distribution, $\log[p(\omega, \mathbf{p})] = \log[p(\omega | \mathbf{Y}, \mathbf{X})\, p(\mathbf{p})]$, which is proportional to the Hamiltonian, $H(\omega, \mathbf{p})$. If we let $p(\mathbf{p}) = \mathcal{N}(\mathbf{p} | \mathbf{0}, \mathbf{M})$, where the covariance $\mathbf{M}$ denotes the mass matrix, our Hamiltonian can then be written as:

$$H(\omega, \mathbf{p}) = \underbrace{-\log[p(\mathbf{Y}|\mathbf{X}, \omega)\, p(\omega)]}_{\substack{\text{Potential Energy} \\ U(\omega)}} + \underbrace{{}^1\!/_2\ \mathbf{p}^\top \mathbf{M}^{-1} \mathbf{p}.}_{\substack{\text{Quadratic Kinetic Energy} \\ K(\mathbf{p})}}$$

We sample using the leapfrog integration scheme, the details of which can be found in Cobb et al.[28] and Neal.[29]

Finally, we remark that similar approaches can be adopted for the node and edge classification tasks, although present implementation of Graph Convolution in a Bayesian framework requires specifying an a priori upper bound of graph complexity. Although theoretically we may consider samples as subgraphs from the Rado graph, for the sake of computational tractability, random graphs will need to be drawn as subgraphs of some arbitrarily large complete graph K_N. This in turn has implications for edge and community detection, and necessitates further research into techniques for identifying graphs exceeding the maximal specified complexity.

Such research in this direction should explore the Spike and Slab prior method explored by Chen et al.[30] The 'Spike' and 'Slab' prior is a mixture model that consists of a point mass spike at zero using the Dirac δ function, and widely spread distribution (the slab):

$$p(\vartheta_\alpha) = (1 - p_0)\delta(\vartheta_\alpha) + p_0\mathcal{N}(\vartheta_\alpha; 0, \sigma_0^2) \tag{19}$$

where $p_0 \in [0, 1]$. Following Chen et al., the Bayesian MRF with a spike and slab prior is formulated by letting α be an enumeration of the cliques in a graph, so that

$$p(\mathbf{x} \mid \vartheta) \propto \exp(\sum_\alpha \vartheta_\alpha f_\alpha(\mathbf{x}_\alpha))$$

$$\vartheta_\alpha = Y_\alpha G_\alpha$$

$$Y_\alpha \sim \text{Bern}(p_0) \qquad p_0 \sim \text{Beta}(a, b)$$

$$G_\alpha \sim \mathcal{N}(0, \sigma_0^2) \qquad \sigma_0^{-2} \sim \Gamma(c, d).$$

Of particular relevance to future work for Bayesian modeling for edge and community detection, Y_α is a binary random variable representing edges in the clique variable $\mathbf{x}_\alpha$, while G_α is the actual value of the parameter ϑ_α when the edges are instantiated. Chen et al. applied an approximate MCMC method combining Langevin dynamics and reversible jump MCMC to perform inference (LMC) for approximation when the probability distribution has a generic log-linear parametrization

$$p(\mathbf{x} \mid \vartheta) = \frac{1}{A(\vartheta)} \exp\left(\sum_\alpha \vartheta_\alpha f_\alpha(\mathbf{x}_\alpha)\right), \tag{20}$$

where f_α in Equation 20 is a feature function of a set of variables $\mathbf{x}_\alpha$ and associated parameter ϑ_α. Chen et al. identify that learning the parameters of the log-linear MRF described in Equation 20 allows for learning the entire graph structure if some parameters are allowed to tend to zero. In particular, this approach is ideally suited for learning sparse structures, so future experimentation comparing the performance of LMC against HMC is worth investigation when extending to graphs with multiple high valence nodes.

4. METHODOLOGY

In this section, we first describe the two datasets used for studying adversarial patch detection. We then describe the various graph networks and uncertainty metrics we used to perform adversarial patch detection. Finally, we conduct an investigation of these architectures and metrics for adversarial patch detection

4.1 Subject Dataset and Data Preparation

We gathered publicly available data from the COCO[20] and APRICOT[5] datasets for evaluation.

COCO, the Common Objects in Context dataset, is a popular dataset designed for the goal of object detection and scene understanding. Images consist of multiple objects, labeled with precise per-instance segmentations in the form of bounding box data, across 91 object types, with layered annotations. There are a total of 2.5 million labeled instances across 328,000 images, which vary in pixel size and number of bounding boxes.

APRICOT is a popular dataset of over 1000 images derived from the COCO dataset with printed adversarial patches in public locations for the purpose of studying and maintaining adversarial robustness in uncontrolled settings. The patches in the APRICOT dataset have been designed to target several object categories for multiple COCO-trained detection models, with variations across position, lighting conditions, angle, and distance determined to attack the context learning capabilities of those models.

We formed a dataset of 2437 images from the COCO and APRICOT datasets, supplementing them with additional features. These features included a binary label to indicate that there is a patch in the image, and the graph output features derived from applying the Faster R-CNN model to each image.[19] These outputs included predicted bounding boxes, predicted categories, predicted probabilities of the top category, and the full-image convolutional features for each bounding box.

4.2 Network Architectures and Metrics

We approached the challenge of adversarial patch detection with tools of graph representation learning and uncertainty quantification by first determining how to form our graphs, and studying various points at which uncertainty could be quantified and used for out-of-context detection. Graphs were formed first by applying Faster R-CNN on our dataset,[19] where each node was identified with a predicted bounding box, and each edge was inferred whenever bounding boxes overlapped. The features selected for each node varied by experiment.

The output features from Faster R-CNN include: (1) height of bounding box, (2) width of bounding box, (3) area of bounding box, and (4) the 256-length visual-feature vector obtained through max-pooling of the Region Proposal Network (RPN) region-head outputs over the dimension of proposed categories. In particular, we use the 256-length visual feature vector obtained through max-pooling of the RPNG region-head outputs as a node-level feature for some of our experimental adversarial patch detection models. Edge-level features that we considered included solely (1) the Euclidean distance between centers of bounding boxes in R^2, however we did not use this as a feature for any of our experimental adversarial patch detector models.

We utilized torchvision's implementation of the Faster R-CNN along with pre-trained weights that are available for the COCO dataset. We constructed a pipeline which generated R-CNN predictions for each image in the form of bounding boxes, predicted labels, and scores. These were then converted into edge-wise sparse matrices, coordinate-format with respect to pairs of nodes, and finally graphs using the Python Deep Graph Library (DGL). The edge and node feature vectors are generated for all pairs of bounding boxes, and these are then pruned according to overlap of bounding boxes to generate the adjacency matrices for each sampled graph.

For our experimental detection models, we consider two kinds of binary classification schemes: graph classification and logistic regression. We consider both schemes in both deterministic and Bayesian settings. We explored two graph classification architectures and three logistic regression models.

Our graph classification model consisted of two layers of Graph Convolutional Neural Network layers, followed by an average pooling of the features, before feeding into two fully connected neural network layers. We considered two graph classification models varying only on the activation function between the first and second fully connected layer. The baseline setting used ReLU and the second setting used Randomized Leaky ReLU, with a lower bound of .001 and an upper bound of .1. We implemented our GCNs in `torch_geometric`.

For the Bayesian variant of these two graph convolutional networks, we split the GCNs at the Average Pooling stage into a transformation function to be applied to our input graph objects, and a Bayesian neural network consisting of the two fully connected layers. Recalling Equation 15, we are treating

$$\left(\prod_{v\in V}\chi(x_v,z_v)\prod_{(u,v)\in E}\psi_V(z_u,z_v)\psi_E(x_{(u,v)},z_{(u,v)})\right)$$

as an embedded features, and sampling $\zeta(x_G,z_G)$.

In addition to using the score given to each node detected by the Faster R-CNN model, we also considered using the *certainty* and *certainty score* of each image. Introduced in Berenbeim et al.[11] and developed for out-of-distribution detection in Wong et al.,[12] *certainty* and *certainty score* are defined with respect to a discriminative models pseudo-probability vector output and describes the relative certainties between alternative categories. In particular, the certainty is defined by Equation 21 whereas the certainty score (alternatively, predictive confidence score) is defined by Equation 22. With $\mathbf{p}$ denoting a pseudo-probability vector and $[n]$ denoting a set of n labels,

$$\mathbf{C}(\mathbf{p}) = \mathbf{1}\mathbf{p}^\top - \mathbf{p}\mathbf{1}^\top + \mathbf{I} \tag{21}$$

$$\varsigma(\mathbf{p}) = \max_{i \in [n]} \mathbf{p}_i - \max_{j \in [n] \setminus \{i\}} \mathbf{p}_j \tag{22}$$

The torchvision Faster R-CNN model exposes the score of the final predicted category for each bounding box. This is obtained through the direct external application of the public RPN/RoIHeads component on images in the training dataset. However, the scores that belong to the remaining predictions are not accessible without its careful re-implementation. Due to the prevalent use of protected methods and encapsulation in the torchvision Faster R-CNN model, we opted to train an auxiliary ResNet50 model to match the Faster R-CNN model outputs instead of re-implementing the model to extract the pseudo-probability score. The choice to use an auxiliary model trained on the proposed bounding boxes from the Faster R-CNN model in order to extract certainty has two immediate drawbacks: (1) it limits our ability to describe the Faster R-CNN model competence, and (2) we can only indirectly assess the relative decline of certainty of non-patch objects neighboring a patch, since the presence of a patch can impair the quality of an RPN.

4.3 Experimental Design

Our primary aim was to explore the use of the certainty and competence score framework developed in Berenbeim et al.[11] and Wong et al.[12] as features in graph classification and logistic regression models, in comparison with the out-of-distribution detection framework developed in Wong et al.[12] The goal of this exploration was to identify the best performing adversarial patch detection scheme within the Faster R-CNN framework for identifying bounding boxes and labels among the aforementioned architectures, and from direct analysis. Towards this end we ran both deterministic and stochastic variations of three GCN models and three logistic regression models.

We hypothesized that given the role that context plays in classifying objects, and the design of adversarial patches to perturb said context, such perturbations would be evident in the certainties of the classification of the individual objects, and by extension would be informative model features for detecting patches. The first two graph classification model apply the graph level features output by the Faster R-CNN model used to construct the graphs. The third graph classification model uses the certainty from an auxiliary ResNet50 model trained on the COCO dataset applied to each bounding box predicted by the Faster R-CNN model with output values trained to the set of predicted labels from the Faster R-CNN model.

In contrast, the logistic regression models considered collapsed the graph level features for certainty as follows. The first logistic regression considered the arithmetic mean of the predicted object scores from the Faster R-CNN model. The second logistic regression considered the arithmetic mean of the predicted certainty scores derived from the ResNet50 model applied to each bounding box predicted by the Faster R-CNN model. The third logistic regression considered the arithmetic mean of the log-odd with respect to the predicted bounding box score. This final feature was used as a proxy for the certainty score for the pre-trained Faster R-CNN model, as the true certainty score is inaccessible without re-implementing multiple modules within the Faster R-CNN model due to the use of protected methods. However, whereas certainty scores are always non-negative, this value is bounded below by $-\log 90$.

For the Bayesian models, we considered, but ultimately rejected the application of the PCS-VRO test described in Zhang et al.[21] This test consists of dividing the unit square into four quadrants, and identifying non-adversarial benign examples as those with high precision confidence score (aka certainty score) and low VRO, while adversarial example have low PCS and high VRO. Examples with neither are not classified. This test was rejected ultimately due to the reflexive nature of applying the test to the outputs of binary classification models. This test would be substantially more appropriate to either a Bayesian variant of the auxiliary ResNet50 model, or a Bayesian variant of the Faster R-CNN model where we might appropriately extract a VRO score for persistent bounding boxes. In lieu of the PCS-VRO test, we examine the use of the certainty score distributions as found in Wong et al.[12] as a means for out-of-distribution detection in the following settings: first, by comparing the distribution of average certainty scores between the COCO and APRICOT images directly, then by including them as features in graph convolutional networks and logistic regressions, and finally by comparing the certainty scores from each of the six models generated.

We hypothesize the following:

- The convolution of certainty information will be predictive of APRICOT data.

- APRICOT data will have an identifiably lower average score and an identifiably lower average certainty score.
- Binary classification certainty scores will be higher for COCO classification than APRICOT classification.

5. RESULTS AND DISCUSSION

After running our experiments, we answer our three hypotheses as follows:

- The convolution of the Faster R-CNN features and the certainty of the auxiliary ResNet model outputs was predictive of adversarially patched APRICOT data.
- APRICOT data had identifiably lower average scores, but not identifiably lower average certainty scores–further, only the average log-odds score derived from the Faster R-CNN model output scores was relatively predictive of adversarial patches.
- Certainty scores from the auxialiary model were distributed nearly identically, although proxies for certainty scores were from distinct distributions.

	Accuracy	Precision	Recall	F1	AUC-ROC
D:features_rrelu	.935	.994	.805	.889	.974
S:features_rrelu	.462	.020	.014	.016	.085
D:features	.675	.000	.000	.000	.649
S:features	.449	.246	.336	.284	.499
D:certainty	.740	.700	.350	.467	.766
S:certainty	.675	.000	.000	.000	.453
D:avgscore	.675	.000	.000	.000	.775
S:avgscore	.675	.000	.000	.000	.775
D:avgcertscore	.675	.000	.000	.000	.470
S:avgcertscore	.675	.000	.000	.000	.459
D:avglogoddscore	.743	.750	.314	.442	.812
S:avglogoddscore	.675	.000	.000	.000	.812

Table 1. Out-of-Distribution Detection Results By Test According to Model

Table 1 collects the performance of our experimental models. A majority of the models we tested performed poorly at predicting patches, and were biased towards predicting no patches with respect to the given feature. Only the following models were able to detect adversarial patches: the GCNs with Faster R-CNN features as inputs and a random ReLU layer; the stochastic GCN using the Faster R-CNN features as inputs with standard ReLU layer; the GCNs using certainty gathered from the auxiliary ResNet50 model as node features; the deterministic logistic regression model using the avg log-odds score. Among these models, both of the stochastic GCNs using the Faster R-CNN features made Type I and Type II errors, resulting in lower accuracy.

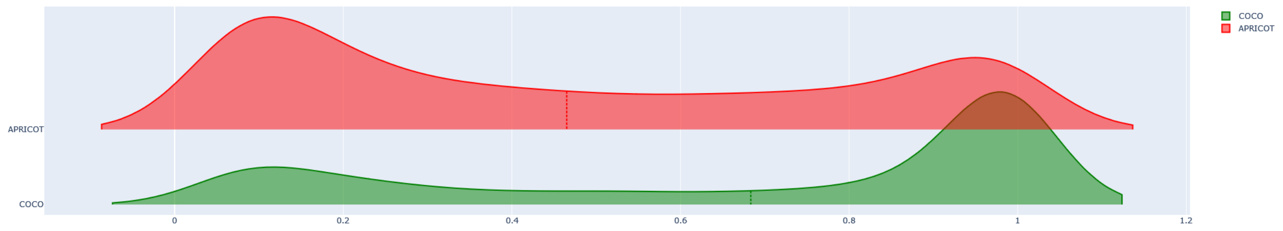

Figure 1. Distribution of the node-level scores by Benign (COCO) and Adversarial (APRICOT) datasets.

We can attribute some of the under performance of our experimental models to the absence of tuning the following hyperparameters: learning rate, number of steps, and number of epochs. However, as seen in Figures

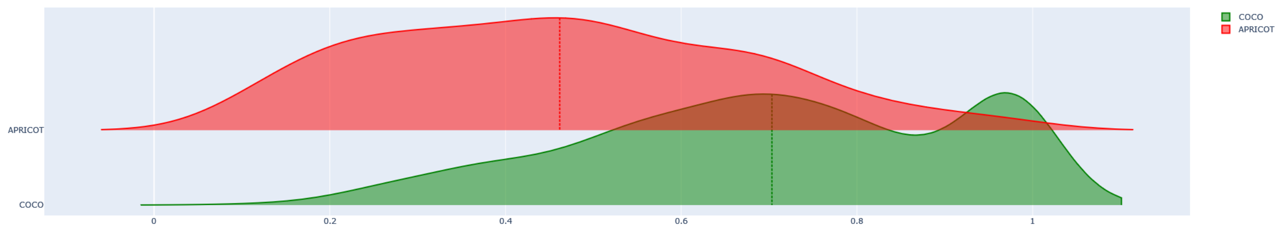

Figure 2. Distribution of the graph-level average scores by Benign (COCO) and Adversarial (APRICOT) datasets.

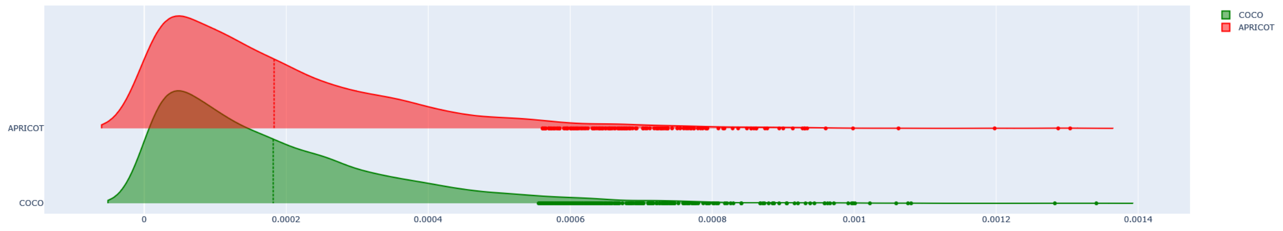

Figure 3. Distribution of the node-level certainty scores by Benign (COCO) and Adversarial (APRICOT) datasets.

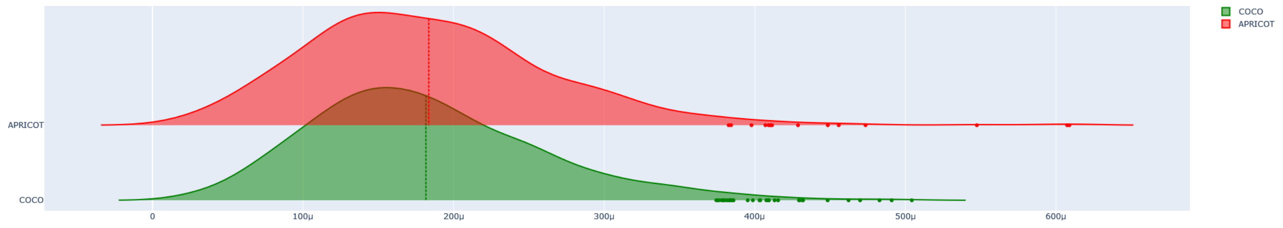

Figure 4. Distribution of the graph-level certainty scores by Benign (COCO) and Adversarial (APRICOT) datasets.

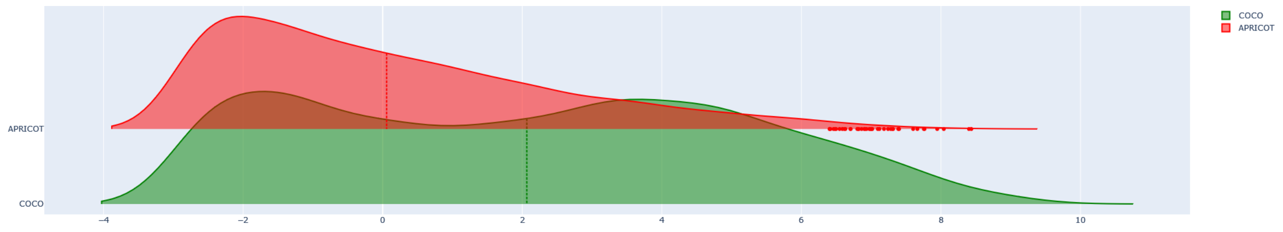

Figure 5. Distribution of the node-level log-odds ratio by Benign (COCO) and Adversarial (APRICOT) datasets.

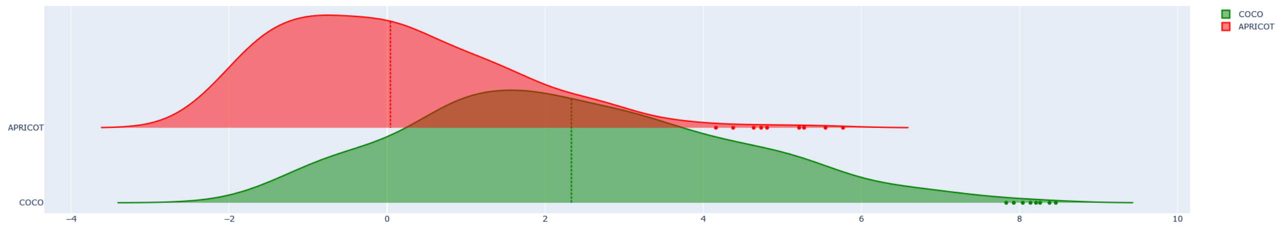

Figure 6. Distribution of the graph-level average log-odds ratio by Benign (COCO) and Adversarial (APRICOT) datasets.

2, 4, and 6, neither the graph level average values of each node's score from the Faster R-CNN model, nor the average certainty score of each image from the auxiliary ResNet50 model were meaningfully separated by the presence of patches. This can readily explain those particular models failure to distinguish APRICOT data from COCO data. In particular, this presents sufficient indication that the auxiliary ResNet model was relatively incompetent, or otherwise, inadequate for adversarial patch detection. In contrast, the sufficient spread between the average log-odds determined by the Faster R-CNN score being sufficiently separated between APRICOT and COCO data is consistent with the observation that the presence of patches negatively impacts the surrounding visual context in each image, resulting in lower scores for surrounding nodes.

Further, as the auxiliary ResNet model did not factor in neighboring bounding box information when training, but was only trained to match the predicted label of the Faster R-CNN model, the certainty scores from these outputs would not necessarily convey any information about neighbors, and in particular would not necessarily convey any structural information about the nodes that corresponded to the patched image relevant to adjacent nodes; instead, the perturbations to context would only be manifest in the output for the Faster R-CNN model which were used to determine the bounding boxes themselves. However, the certainty of each node still contains relevant information for patch detection, as non-patch images have certainties relatively clustered around a few categories while the patches have certainties spread across multiple unrelated categories.

This proved to be the case, as ultimately we found that the best performing graph neural network models composing two GCN layers that convolved the Faster R-CNN features or the certainty features before passing to a two fully connected layers, and that using certainty from the auxiliary model as the node feature led to better performing models over the Faster R-CNN features when using a standard ReLU activation function. Future research in this direction should involve developing a forked variant of the Faster R-CNN model that outputs the full pseudo-probability vector for each identified bounding box, and not just the score. From this vector, we can compute the corresponding node uncertainty.

Although the distribution of certainty scores from our auxiliary ResNet50 model were nearly identically distributed between the COCO and APRICOT datasets, with the Mann Whitney U test p-value of 0.910, the proxies we can use for the certainty score, namely the assigned probability score and the log-odds ratio can be confirmed to belong to different distributions with a Mann Whitney U test p-value of 1.014e-247 for both respective distributions. These results can be visually confirmed in Figures 1, 3, and 5. Ultimately, this can be attributed to the presence of adversarial patches significantly impacting the predictive confidence of neighboring items from the Faster R-CNN model, in contrast with the auxiliary ResNet50 models which did not incorporate the surrounding information in training.

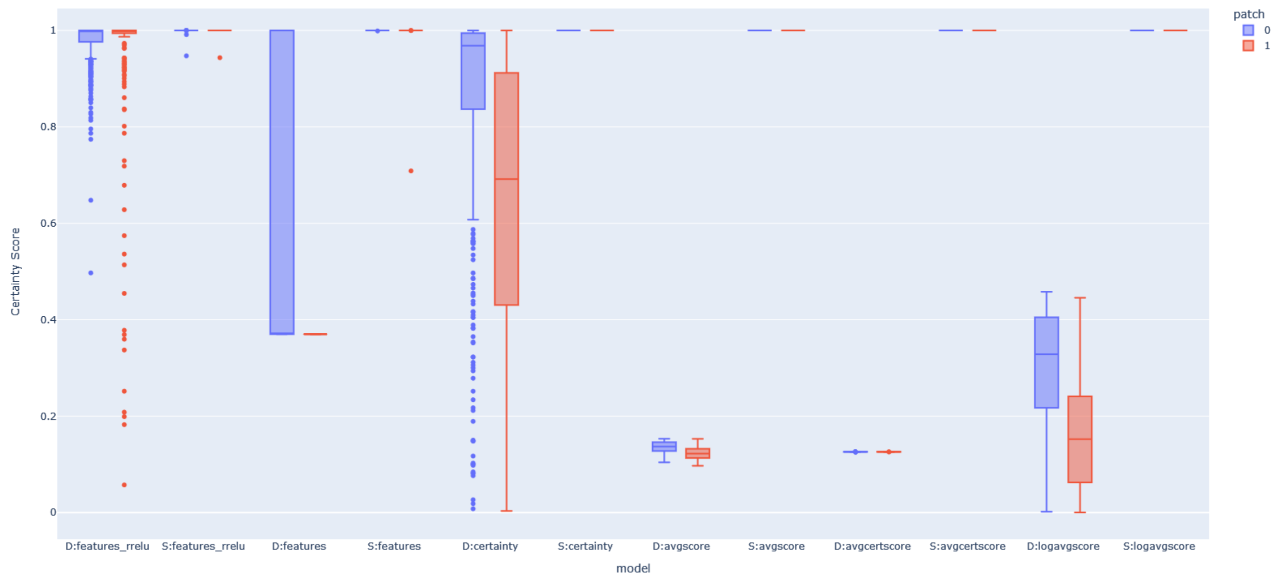

Figure 7. Box-plots of classification model certainty scores by COCO (0) and APRICOT (1) sourced graphs

Finally, we provide a box-plot distribution for the certainty scores of our classification models in Figure 7. In general, most models were as certain in classifying data as originating from the COCO dataset as they were as originating from the APRICOT dataset. We primarily attribute this to a lack of hyperparameter tuning for the graph convolution models, and the lack of a linear relationship between the regressor feature and the presence of a patch in an image for the logistic models.

Models that had some predictive power differed, with the general tendency being that data without patches was predicted to be without a patch with higher certainty scores. The one exception to this was the Deterministic GCN with Randomized Leaky ReLU using the Faster R-CNN features– this model generally had higher a higher certainty score that images contained patches than the counterpart models. This observation can be explained in part by the nature of the features being convolved, as well as the particular efficacy of using Randomized Leaky

ReLU in learning how to weight the features for identifying patches in training. That the predictive models were otherwise less certain that graphs contained patches comports with the nature of the patches measurably perturb context, even though the graphs have at most one node corresponding to a patch.

Ultimately, we find support that the certainty and competence framework can be applied to detect adversarial patches, either as a feature of a graph convolutional network, or as part of out-of-distribution tests comparing the certainty score distributions derived from a predictive classifying model's output.

6. CONCLUSION

This paper summarized current approaches to GRL, presenting a generalization of the GRL formalism for future development within a Bayesian framework. We used this framework to perform an empirical study using graph representation learning and uncertainty quantification techniques to perform adversarial patch detection.

We first provided a comprehensive overview of the graph representation learning framework and motivated the generalization to a Bayesian graph classification model using deterministic GCN models as embeddings. We experimentally examined the use of several forms of uncertainty quantification as features for the purposes of detecting adversarial patches. We found that graph convolutional networks with two edge convolution layers were able to detect patches with a relatively high degree of accuracy under certain circumstances using both the certainty matrix output of an auxiliary ResNet50 model trained to identify the outputs of the Faster R-CNN model, and with the features output by the Faster R-CNN model when combined with a Random leaky ReLU activation function. These results comport with the nature of the adversarial patches generated for the APRICOT dataset, which were intended to perturb the visual context in ways to impact object recognition of proximate objects within the image, and indicate that convolving the features of the node corresponding to a patch with its neighbors can be used to identify patches. Further, we found little evidence to support using graph-level averages in classical or stochastic logistic regressions beyond the use of the average log-odds ratio.

In total, these results provide a proof-of-concept that graph representation learning incorporating uncertainty quantification at the node level can be used as supplements to models whose outputs are sensitive to context and subject to adversarial attacks that perturb the representation of context. We propose that future research for adversarial patch detection replace the use of an auxiliary ResNet50 model to generate certainty by relying on the full pseudo-probability vector output by the discriminative submodel used for object classification internal to the object detection and classification models. We believe that research in this direction will also enable the extension of the Bayesian sampling from graph level features to node level features, which will lead to more robust and explainable model outcomes, particularly with respect to adversarial patch detection.

ACKNOWLEDGMENTS

This work was supported in part by the U.S. Army Combat Capabilities Development Command (DEVCOM) Army Research Laboratory under Support Agreement No. USMA 21050 and the Defense Advanced Research Projects Agency (DARPA) under Support Agreement No. USMA 23004. The views expressed in this paper are those of the authors and do not reflect the official policy or position of the U.S. Military Academy, the U.S. Army, the U.S. Department of Defense, or the U.S. Government.

REFERENCES

[1] Petroni, F., Rocktäschel, T., Lewis, P., Bakhtin, A., Wu, Y., Miller, A., and Riedel, S., "Language models as knowledge bases?," Association for Computational Linguistics (2019).

[2] Singhal, K., Azizi, S., Tu, T., Mahdavi, S. S., Wei, J., Chung, H. W., Scales, N., Tanwani, A., Cole-Lewis, H., Pfohl, S., et al., "Large language models encode clinical knowledge," *Nature* **620**(7972), 172–180 (2023).

[3] Huang, Q., Ren, H., Chen, P., Kržmanc, G., Zeng, D., Liang, P. S., and Leskovec, J., "Prodigy: Enabling in-context learning over graphs," *Advances in Neural Information Processing Systems* **36** (2024).

[4] Zhang, Y., Zhou, K., and Liu, Z., "What makes good examples for visual in-context learning?," *Advances in Neural Information Processing Systems* **36** (2024).

[5] Braunegg, A., Chakraborty, A., Krumdick, M., Lape, N., Leary, S., Manville, K., Merkhofer, E., Strickhart, L., and Walmer, M., "Apricot: A dataset of physical adversarial attacks on object detection," in [*Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXI 16*], 35–50, Springer (2020).

[6] Puttagunta, M. K., Ravi, S., and Nelson Kennedy Babu, C., "Adversarial examples: attacks and defences on medical deep learning systems," *Multimedia Tools and Applications* **82**(22), 33773–33809 (2023).

[7] Sur, I., Sikka, K., Walmer, M., Koneripalli, K., Roy, A., Lin, X., Divakaran, A., and Jha, S., "Tijo: Trigger inversion with joint optimization for defending multimodal backdoored models," in [*Proceedings of the IEEE/CVF International Conference on Computer Vision*], 165–175 (2023).

[8] Ojaswee, O., Agarwal, A., and Ratha, N., "Benchmarking image classifiers for physical out-of-distribution examples detection," in [*Proceedings of the IEEE/CVF International Conference on Computer Vision*], 4427–4435 (2023).

[9] Zuo, Z., Zhao, L., Li, A., Wang, Z., Zhang, Z., Chen, J., Xing, W., and Lu, D., "Generative image inpainting with segmentation confusion adversarial training and contrastive learning," in [*Proceedings of the AAAI Conference on Artificial Intelligence*], **37**(3), 3888–3896 (2023).

[10] Chen, G., Kang, P., Wu, X., Yang, Z., and Liu, W., "Adaptive visual field multi-scale generative adversarial networks image inpainting base on coordinate-attention," *Neural Processing Letters* **55**(7), 9949–9967 (2023).

[11] Berenbeim, A. M., Cruickshank, I. J., Jha, S., Thomson, R. H., and Bastian, N. D., "Measuring classification decision certainty and doubt," *arXiv preprint arXiv:2303.14568* (2023).

[12] Wong, J. A., Berenbeim, A. M., Bierbrauer, D. A., and Bastian, N. D., "Uncertainty-quantified, robust deep learning for network intrusion detection," in [*2023 Winter Simulation Conference (WSC)*], 2470–2481 (2023).

[13] Hamilton, W., [*Graph Representation Learning*], Synthesis lectures on artificial intelligence and machine learning, Morgan & Claypool Publishers (2020).

[14] Dai, H., Dai, B., and Song, L., "Discriminative embeddings of latent variable models for structured data," in [*International conference on machine learning*], 2702–2711, PMLR (2016).

[15] Carr, A. and Wingate, D., "Graph neural processes: Towards bayesian graph neural networks," *arXiv preprint arXiv:1902.10042* (2019).

[16] Zhang, Y., Pal, S., Coates, M., and Ustebay, D., "Bayesian graph convolutional neural networks for semi-supervised classification," in [*Proceedings of the AAAI conference on artificial intelligence*], **33**(01), 5829–5836 (2019).

[17] Hasanzadeh, A., Hajiramezanali, E., Boluki, S., Zhou, M., Duffield, N., Narayanan, K., and Qian, X., "Bayesian graph neural networks with adaptive connection sampling," in [*International conference on machine learning*], 4094–4104, PMLR (2020).

[18] Pal, S., Malekmohammadi, S., Regol, F., Zhang, Y., Xu, Y., and Coates, M., "Non parametric graph learning for bayesian graph neural networks," in [*Conference on uncertainty in artificial intelligence*], 1318–1327, PMLR (2020).

[19] Ren, S., He, K., Girshick, R., and Sun, J., "Faster r-cnn: Towards real-time object detection with region proposal networks," (2016).

[20] Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., and Dollár, P., "Microsoft coco: Common objects in context," (2015).

[21] Zhang, X., Xie, X., Ma, L., Du, X., Hu, Q., Liu, Y., Zhao, J., and Sun, M., "Towards characterizing adversarial defects of deep learning software from the lens of uncertainty," (2020).

[22] Deng, L., "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine* **29**(6), 141–142 (2012).

[23] Krizhevsky, A., Nair, V., and Hinton, G., "Cifar-10 (canadian institute for advanced research),"

[24] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L., "Imagenet: A large-scale hierarchical image database," in [*2009 IEEE conference on computer vision and pattern recognition*], 248–255, Ieee (2009).

[25] Gong, L. and Cheng, Q., "Exploiting edge features for graph neural networks," in [*Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*], 9211–9219 (2019).

[26] Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al., "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261* (2018).

[27] Neal, R. M., *Bayesian Learning for Neural Networks*, PhD thesis, University of Toronto (1995).

[28] Cobb, A. D. and Jalaian, B., "Scaling Hamiltonian Monte Carlo Inference for Bayesian Neural Networks with Symmetric Splitting," *arXiv preprint arXiv:2010.06772* (2020).

[29] Neal, R. M. et al., "MCMC using Hamiltonian dynamics," *Handbook of Markov chain Monte Carlo* **2**(11), 2 (2011).

[30] Chen, Y. and Welling, M., "Bayesian structure learning for markov random fields with a spike and slab prior," *stat* **1050**, 23 (2012).