Runtime Monitoring of Deep Neural Networks Using Top-Down Context Models Inspired by Predictive Processing and Dual Process Theory

Anirban Roy, [1] **Adam Cobb,** [1] **Nathaniel D. Bastian,** [2] **Brian Jalaian,** [3] **Susmit Jha**[1]

[1] Neuro-Symbolic Computing and Intelligence, Computer Science Laboratory, SRI International
[2] Intelligent Cyber-Systems and Analytics Research Lab, Army Cyber Institute, U.S. Military Academy
[3] Joint Artificial Intelligence Center, U.S. Department of Defense
anirban.roy@sri.com, adam.cobb@sri.com, nathaniel.bastian@westpoint.edu, brian.a.jalaian.civ@mail.mil, susmit.jha@sri.com

Abstract

Deep neural networks (DNNs) have achieved near-human-level accuracy on many datasets across different domains. But they are known to produce incorrect predictions with high confidence on inputs far from the training distribution. This challenge of lack of calibration of DNNs has limited the adoption of deep learning models in high-assurance systems such as autonomous driving, air traffic management, cybersecurity, and medical diagnosis. The problem of detecting when an input is outside the training distribution of a machine learning model, and hence, its prediction on this input cannot be trusted, has received significant attention recently. Several techniques based on statistical, geometric, topological, or relational signatures have been developed to detect the out-of-distribution (OOD) or novel inputs. In this paper, we present a runtime monitor based on predictive processing and dual process theory. We posit that the bottom-up deep neural networks can be monitored using top-down context models comprising two layers. The first layer is a feature density model that learns the joint distribution of the original DNN's inputs, outputs, and the model's explanation for its decisions. The second layer is a graph Markov neural network that captures an even broader context. We demonstrate the efficacy of our monitoring architecture in recognizing out-of-distribution and out-of-context inputs on the image classification and object detection tasks.

Introduction

Machine learning models, such as those used in autonomous self-driving cars, need to be capable of independent decision-making in possibly new environments which can be different from their training environment. It is very difficult to measure such a generalization capability and predict the behavior of machine learning models in novel scenarios. The successes of deep neural networks (DNNs) on standard datasets in many domains such as computer vision (Gkioxari, Girshick, and Malik 2015), speech recognition (Hannun et al. 2014), and text analysis (Majumder et al. 2017), has not been representative of their performance in the open-world, where inputs might not belong to the training distribution on which the DNN was trained. Consequently, this has inhibited their deployment in the safety-critical systems such as self-driving vehicles (Bojarski et al. 2016), aircraft collision avoidance (Julian and Kochenderfer 2017), battlefields (Abdelzaher et al. 2018) and medical diagnoses (De Fauw et al. 2018). This brittleness and the resulting lack of trust of DNN-based artificial intelligence (AI) systems is exacerbated by the high confidence in predictions of DNNs even on inputs which are out of distribution and the predictions are usually incorrect. This high confidence on incorrect predictions on OOD input has been widely reported in literature (Guo et al. 2017a; Hendrycks and Gimpel 2016) and is attributed to the overfitting of the model in the negative log likelihood space. The responsible deployment of DNN models in high-assurance applications necessitates the detection of those inputs and scenarios where the DNN cannot be trusted and, hence, must abstain from making decisions. The question then is: *can we locate these machine learning models in a monitoring architecture where their failures can be detected and masked or tolerated?*

We posit that we have identified such a candidate architecture: it is one in which we build a predictive context model and rather than use the output of deep learning models directly, we first validate and fuse them with the context model to detect whether the inputs present a surprise to the model. This may seem like an exercise in semantics — even the usual machine learning models typically "fuse" interpretations from different sensors that compose the input to the model, and collate them over time — but we contend that our proposed monitoring architecture amounts to a shift in focus and brings new techniques with it, as we will illustrate in this paper. We suggest that a better approach is to evaluate inputs against a context model: the model is the accumulation of everything we have learned and come to trust and it makes more sense to evaluate new inputs against this than only predict on the input in isolation. This is the basis of the approach that we recommend, but we locate it in a model of perception known as *predictive processing* (PP) (Rao and Ballard 1999) complemented with the dual process theories of reasoning (Evans 2010).

We make the following contributions in this paper:

- We propose a new two-layered runtime monitoring architecture for DNNs motivated by insights from cognitive science, in the form of predictive processing and dual process theory.
- We provide a candidate implementation of this run-

time monitoring architecture using feature density modeling to implement the first monitoring layer, and graph Markov neural networks to implement the second layer.

- We demonstrate the value of this approach on the image classification and object detection tasks and show how different layers of the runtime monitor can be used to detect OODs, novel classes, and out of context inputs.

Related Work

Recent approaches for outlier and out of distribution detection consider different statistical, geometric or topological signatures in data that differentiate OODs from the training distribution. The softmax score can be viewed as representing the prediction uncertainty of a DNN, and hence, the changes in the softmax scores due to input perturbations and temperature scaling have been used to detect OODs (Hendrycks and Gimpel 2016; Liang, Li, and Srikant 2017; Guo et al. 2017b). Another line of work uses the conformance among the predictions made by a machine learning models on the nearest neighbors (Papernot and McDaniel 2018). Cosine similarity (Tack et al. 2020) to the nearest training sample (or a subset of these kept as memorized prototype examples during monitoring) has also been used for the detection of OODs. The Mahalanobis distance of an input from the in-distribution data (Lee et al. 2018) has also been proposed to detect OOD inputs. Several other metrics such as reconstruction error (An and Cho 2015), likelihood-ratio between the in-distribution and OOD samples (Ren et al. 2019), trust scores (ratio of the distance to the nearest class different from the predicted class and the distance to the predicted class) (Jiang et al. 2018), density function (Liu et al. 2020b; Hendrycks, Mazeika, and Dietterich 2019), probability distribution of the softmax scores (Lee et al. 2017; Hendrycks et al. 2019; Tack et al. 2020; Hendrycks, Mazeika, and Dietterich 2019) have also been used to detect OODs. These methods can be viewed as runtime monitors. The first layer of our runtime monitoring framework can leverage these approaches. We adopt feature density modeling to implement the first layer. One key distinction from these approaches is our use of attributions (explanations) for individual decisions made by a machine learning model. This distinguishes our approach from existing literature which focus on learning the joint distribution of input and output, which is independent of the model. By incorporating the attribution for a decision, our first layer models not just data but the model's response to the data, which along with the output serves as the simplest context for the input.

A closely related field of study is the use of deep ensembles for uncertainty quantification (Lakshminarayanan, Pritzel, and Blundell 2016; Blundell et al. 2015; Gal and Ghahramani 2016; Wen, Tran, and Ba 2020; Dusenberry et al. 2020). These methods rely on either explicitly using different neural networks or multiple passes over a stochastic network where the network weights are sampled from a distribution. This leads to significant memory and computation overhead. In a parallel line of work, the density estimates of a DNN's latent representations have been used to quantify epistemic uncertainty (Mandelbaum and Weinshall 2017; Jha et al. 2018a; Oh et al. 2018; van Amersfoort et al. 2020). These methods often require modification to the training method, such as replacing softmax activation with a target-conditional multivariate Gaussian distribution for classification. This ensures that the model learns a density estimate of the final layer, and the log-likelihood of the density model is treated as the epistemic uncertainty useful for OOD detection. Unnormalized density on the softmax logits have also been used for OOD detection (Liu et al. 2020b). Such methods can be improved using contrastive training (Winkens et al. 2020) that improves the feature extractor and reduces feature collapse before estimating the feature-space density. More recently, uncertainty quantification and OOD detection in deterministic single forward-pass DNNs (van Amersfoort et al. 2020; Liu et al. 2020a; Mukhoti et al. 2021; Jha et al. 2018b; Jang, Jha, and Jha 2019; Jha et al. 2019a) have been studied where distance-aware output layers in the form of radial basis functions or Gaussian processes are learned in presence of additional inductive biases that encourage internal feature representation to better reflect epistemic uncertainty. The inductive biases can be in form of Jacobian penalty (Gulrajani et al. 2017) or spectral normalization (Miyato et al. 2018) which ensure that the internal features are smooth (to aid improved in distribution detection) but at the same time, sensitive to avoid feature collapse and detect OOD inputs. These methods require modification to the training of DNNs and also require finding the suitable hyperparameters in the inductive biases.

Yet another related area is the use of contextual cues for object detection and segmentation. Graph-based models provide a flexible way to represent context where nodes represent objects and edges represent pair-wise relations among the objects(Choi et al. 2010; Gould et al. 2008; Zhang and Chen 2012). Among graph-based models, conditional random fields (CRF) are explored extensively where contextual cues are represented by edge potentials. Common contextual cues include co-occurrence, spatial distance, geometric and appearance similarity(Choi et al. 2010; Gould et al. 2008; Divvala et al. 2009; Koller and Friedman 2009; Zhang and Chen 2012). More recently, graphical models are combined with neural networks to exploit data-driven feature learning. Graph convolutional networks (GCN) (Dai, Dai, and Song 2016; Kipf and Welling 2017; Hamilton, Ying, and Leskovec 2017) provide a convolutional implementation of the graphical models combining the power of representation learning of neural networks with the structured representation of graphs. However, standard GCNs do not explicitly capture the contextual relations that are crucial to detect out of context (OOC) objects where the individual objects in a scene are not novel, but the overall composition is unusual. Existing studies have argued the importance of OOC object detection as these affect the performance of object detection for both humans and machines. Choi et al. (Choi, Torralba, and Willsky 2012) define OOC objects that violate common contextual rules (e.g., flying cars) in terms of unusual background, unusual size, etc. They consider a graph-based model to capture such relations among the objects. Some approaches consider context as the entire background with respect to an object and detect OOC objects that are inconsistent with the background. Dvornik et al. (Dvornik, Mairal, and Schmid 2018) identify the usual pair of objects as OOC instances. None of these approaches can capture context broadly enough to incorporate object classes, relative sizes and relative location with other objects in the scene. GCNs provide an end-to-end neural network-based realization of graph models and are shown to be successful in object detection. Inspired by graph Markov neural networks (GMNNs) (Qu, Bengio, and Tang 2019), we capture relations between objects by simultaneously learning two GCNs - one for learning node representations and another for learning label dependency. By combining the graph neural network as the second layer with the feature density models in the first layer, we form an effective implementation of a predictive coding and dual theory inspired runtime monitor. While the layer 1 feature density model serves as the intuitive and associative "System 1", the graph NNs form the deliberative layer 2.

Predictive Processing and Context Modeling

We invite the reader to review the prior work (Jha, Rushby, and Shankar 2020) which describes the assurance of systems with machine learning models based on predictive coding. We sketch the salient points below. We use a simple example of an autonomous car to demonstrate how the context model can be useful in monitoring DNNs. Consider the component of a car's vision system concerned with detecting traffic lanes. A basic method will look for more-or-less straight lines painted on the road, and a bottom-up approach will perform this process as each image frame is processed. But this is inefficient—the traffic lanes in the current image frame are likely to be similar to those in the previous few frames, and we should surely use this to seed the search—and it is fragile—missing or scuffed lane markers might cause lanes to go undetected where they could have been extrapolated from previous images. A better approach builds a model of the road and its traffic lanes and uses this to seed the search for lanes in the current image by predicting their location. There will be some uncertainty in the model and its projection of lanes, so what is sent to the vision system will be the best guess, or perhaps a probability distribution over several such estimates. The vision system will use this to seed its search for lanes in the current image and will send back the difference or "error" between the prediction and its current observation. The error signal is used to refine the model in a way that aims to minimize future prediction errors and thereby bring it closer to reality.

This is an example of "analysis by synthesis" meaning that we formulate hypotheses (i.e., candidate world models) and favor those whose predictions match the input data. In practical applications, we need to consider the level of the "predictions" concerned: do we use the world model to synthesize the raw data (e.g., pixels) that we predict the sensor will detect, or do we target some higher level of its local processing (e.g., objects)?

The significant attribute of this top-down approach is that it focuses on construction and exploitation of the world model (or models: a common arrangement has a hierarchy of models), in contrast to the more common bottom-up machine learning models. We will develop arguments that the top-down approach is effective for the interpretation and assurance of perception in autonomous systems, but it is interesting, and perhaps reassuring, to know that it is widely believed to be the way perception works in human (and other) brains, as first proposed by Helmholtz in the 1860s (von Helmholtz 1867). *Predictive Processing* (PP) (Wiese and Metzinger 2017), also known as predictive *coding* (Clark 2013) and predictive *error minimization* (Hohwy 2013), posits that the brain builds models of its environment and uses these to predict its sensory input, so that much of its activity can be seen as (an approximation to) iterative Bayesian update to minimize prediction error. PP has prior "predictions" flowing from models down to sense organs and Bayesian "corrections" flowing back up that cause the posterior models to track reality.

It is interesting that the brain seems to work in this way, but there are independent reasons for thinking that PP is a good way to organize the perception system for autonomous systems, as opposed to a largely bottom-up system in which sensor measurements and inputs are interpreted and fused to yield a world model with little feedback from the model back to the sensors and the inputs being collected. The fatal accident between an Uber self-driving car and a pedestrian in Arizona on 18th March 2018 illustrates some deficiencies of such bottom up methods (NTS 2019).

A pure bottom-up system has no recollection even of the immediately previous sensor readings, and this precludes calculation of velocity from position. Consequently, perception systems typically maintain a simple model that will permit this: the object tracker of Lin's vision processing pipeline (Lin et al. 2018) is an example, and the Uber car employed a system of this kind. The Uber car used three sensor systems to build its object tracker model: cameras, radars, and lidar. For each of these sensor systems, its own object detector indicates the position of each detected object and attempts to classify it as, for example, a vehicle, a pedestrian, a bicycle, or other. The object tracker fuses these inputs using a "prioritization schema that promotes certain tracking methods over others, and is also dependent on the recency of the observation" (NTS 2019, page 8). In the case of the Arizona crash, this resulted in a "flickering" identification of the victim as the sensor systems' own classifiers changed their identifications, and as the object tracker preferred first one sensor system, then another, as listed below (NTS 2019, Table 1): 5.6 seconds before impact, victim classified as *vehicle*, by radar; 5.2 seconds before impact, victim classified as *other*, by lidar; 4.2 seconds before impact, victim classified as *vehicle*, by lidar; Between 3.8 and 2.7 seconds before impact, classification alternated between *vehicle* and *other*, by lidar; 2.6 seconds before impact, victim classified as *bicycle*, by lidar; 1.5 seconds before impact, victim classified as *unknown*, by lidar; 1.2 seconds before impact, victim classified as *bicycle*, by lidar.

The deeper harm of this "flickering" identification is that "if the perception model changes the classification of a detected object, the tracking history of that object is no longer considered when generating new trajectories" (NTS 2019, page 8). Consequently, the object tracker never established a trajectory for the victim and the vehicle collided with her even though she had been detected in some form or other for several seconds.

There are two related problems here: one is that the object tracker maintains a rather impoverished model of the world and the context in which decisions are being made, the other is that its method of decision-making on the inputs pays no attention to the context. The goal underlying perception in predictive processing is to build a context model that accurately reflects the world; it therefore encodes a lot more information than an individual input. What we want is a method to measure divergence between the context model and a new input; small divergence should indicate a routine evolution of the world, and can be incorporated as an update to the model; a large divergence requires more attention: does it indicate a new development, or is it possibly a flaw in interpretation of the raw sensor data? In any of the two later cases, we cannot trust the prediction of the machine learning model.

Implementations of the predictive processing approach can employ Bayesian methods (Spratling 2017). The context model represents the various objects in the environment, together with their attributes such as type, trajectory, inferred intent etc., with probability distribution functions (*pdf*s) over some or all of these. An observation updates these *priors* to deliver refined *posterior* estimates. This kind of Bayesian inference typically generates intractable integrals, so predictive processing employs methods known as *Variational Bayes* that turn the problem into iterative optimization of the posterior models to minimize prediction error.

An attractive attribute of predictive processing is that it gives us a systematic way to exploit multiple inputs and sensors, and to fuse and cross-check their information. Suppose we have a context model built from camera data, and we add a proximity sensor. Predictive processing can use the camera-derived model to calculate what the proximity sensor is expected to "see" and this can be seen as a falsifiable test of the model's accuracy. If the prediction is verified, then we have independent confirmation of some aspects of our context model. We say "independent" because it seems plausible that sensors based on different phenomena (e.g., cameras, radars, ultrasound) with completely different interpretation functions and trained on different datasets, will have independent failures. In a fully integrated predictive processing monitor, the context model(s) will combine information from all sources. The context model will update conservatively to reflect this uncertainty, and the monitor will consequently lower its confidence in the machine learning model till the discrepancy is resolved.

Observe that the context model can be quite parsimonious and crude: we do not need a photographic rendition of the scene, merely knowledge of the significant objects in our proximity in sufficient detail to guide safe actions, so discrepancies between the outlines of adjacent vehicles "seen" by cameras and proximity sensors, for example, may be of little account as what we need to know is their presence, position, type, and inferred intent. In fact, as we will discuss later, we can model context at the varying levels of detail. In our implementation discussed in the paper, we model the context at two levels - the first level uses features of the deep neural network, and the second models more high-level spatial and temporal relations between objects in a scene.

Dual Process Models and Hierarchical Context Modeling

Another theory about the organization of the human brain is interesting in this regard; this is the "dual-process" model (Frankish 2010; Evans and Stanovich 2013), popularized by Kahneman as separate "fast and slow" systems of thought (Kahneman 2011). Its utility has been recently demonstrated for computing confidence of machine learning models through a very limited implementation (Jha et al. 2019b,a). *System 1* is unconscious, fast, and specialized for routine tasks; *System 2* is conscious, slow, easily fatigued, and capable of deliberation and reasoning—it is what we mean by "thinking." As with predictive processing, we do not advocate the dual-process model merely because it seems to correspond to the way the brain works, but because it seems, independently, to be a good architecture. Here, we can imagine a feature density model forming a highly automated "System 1," with more deliberative models constituting a "System 2" that gets actively involved when System 1 encounters large prediction errors. System 1 maintains a single world model and the System 2 either embellishes this, or maintains a richer world model of its own. It is believed that humans maintain a hierarchy of models (Frankish 2010; Evans and Stanovich 2013; Kahneman 2011), and this seems a good approach for autonomous systems as well. The idea is that a predictive processing loop operates between each adjacent pair (in the hierarchy) of models, so that the lower level is like a sensor for the upper level, with priority and update frequency determined by the size of prediction errors.

Predictive processing in humans is generally presented as a way to minimize "surprise" or as maintaining "situation awareness." And one way to enhance this would be to increase the use of hypothetical reasoning by System 2 in construction of the world model, so that things not seen but "that might be there" are explicitly represented as "ghosts" or as increased uncertainty on attributes of detected objects. A related idea uses AI to make inferences so that, for example, detection of many brake lights up ahead is used to infer some kind of problem and this will be represented as increased uncertainty in the world model. In this way, what might otherwise be the surprising appearance of an unanticipated situation will instead develop as gradual changes in uncertainty or the resolution of ghosts into real objects. The graph Markov neural networks provide an effective mechanism to both model these relationships and richer context, and to deliberate through counterfactual queries and context-informed prediction. Thus, the dual process theory motivates the two layered predictive coding architecture of our runtime monitor. While these theories aim at explaining human cognition, we use these as a runtime monitor to compute surprise of the underlying model and, hence, detect when the model cannot be trusted due to novel or out of distribution or out of context input.

Runtime Monitoring Architecture

The overall architecture of the proposed runtime monitoring for deep learning models is presented in Figure 1. As illustrated, the architecture has two levels (motivated by the Dual Process theory). In the first level, we use a feature density model that learns a joint distribution of the input, predicted class output and the explanation provided by the model. In the second level, we use the graph Markov neural network to learn spatial and temporal relationships between the objects for an object detection task (more generally, components of an input). In both of these layers, our focus is on runtime monitoring and not developing a cognition system in its own right, hence, the surprise detected by both these layers is used by the monitor to identify when the underlying machine learning model cannot be trusted.

Layer 1: Feature Density Model The joint distribution of the features and inputs can be learned and have been investigated for OOD detection with mixed results (Kirichenko, Izmailov, and Wilson 2020; Zisselman and Tamar 2020). We adopt a relative simpler approach of modeling the joint distribution as a Gaussian mixture model (McLachlan and Basford 1988) but complement features with attribution over the features. We use integrated gradient, which was found to produce better results empirically. For the DNN to be monitored, we learn a Gaussian model (mean μ_y and covariance Σ_y) for the features $f(\boldsymbol{x})$ of each class y of the DNN. This is achieved in a single pass by recording the features $f(\boldsymbol{x})$ for all the samples $\boldsymbol{x}$ in the training set. The Mahalanobis distance is then computed as $d = \sqrt{(f(\boldsymbol{x}) - \mu_y)\Sigma_y^{-1}(f(\boldsymbol{x}) - \mu_y)}$. Using the Mahalanobis distance is akin to the use of Gaussian discriminant analysis, where a Gaussian mixture model with a single Gaussian component per class is used to fit the features. A high Mahalanobis distance implies a low feature-space density and a high epistemic uncertainty.

While Mahalanobis distance estimation with tied covariance are used by the existing state-of-the-art OOD detection methods (Liang, Li, and Srikant 2017), we extend the distribution to also incorporate the attribution over the input features, thus, enabling us to model broader context that includes model-specific attribution. Model attributions are specific to a particular DNN and capture properties such as whether an input has concentrated attribution over few features or more dispersed attributions. We accomplish this by simultaneously learning a Gaussian model (mean μ_y^A and covariance Σ_y^A) for the attributions over the features, computed using integrated gradients $IG(f(\boldsymbol{x}), y)$ for the features of samples $f(\boldsymbol{x})$ for each class y of the DNN. This yields another Mahalanobis distance, which measures the likelihood of a given input given the predicted class and the attribution over the input, $d^A = \sqrt{(IG(f(\boldsymbol{x})) - \mu_y^A)\Sigma_y^{A^{-1}}(IG(f(\boldsymbol{x})) - \mu_y^A)}$.

While runtime monitoring, if either of the two Mahalanobis distances, d_A or d, are high for a given input, we can detect that the prediction of the deep learning model cannot be trusted on this input. In our experiments, we use the 95 percentile value of the distances in the training data as the threshold for flagging inputs at runtime. The features used in the density modeling at Layer 1 are provided to the Layer 2 as the node attributes used in the graph convolutional networks in Layer 2.

Layer 2: Graph Markov Neural Networks (Qu, Bengio, and Tang 2019) For layer 2, we use graph Markov neural network that we implement using two graphical models: 1) Representation graph (RepG) that learns the representation for objects at each node by exchanging the representations with its neighbors. RepG relies on these representations to predict object labels at each node independently, ignoring context cues such as label dependency. 2) To complement RepG, we propose a context graph (ConG) that learns context relations across the nodes by sharing the context cues with its neighbors. We consider a graph convolutional networks (GCN) to instantiate both RepG and ConG. We first introduce the GCN framework and then discuss the implementation of RepG and ConG via GCN.

Given an image, we build a graph over the objects in the image. Let's define a graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. We define $X = \{\boldsymbol{x}_i\}$ and $Y = \{y_i\}$ as the feature representation and label of ith node, respectively. Given this definition, the goal is to predict object labels from the feature.

$$H^{l+1} = f^l(W^l, H^l, E),\ p(y_i|X, E) = \mathrm{SM}(H^L), \qquad (1)$$

where $f^l(\cdot)$ is the convolutional function corresponding to layer l iteratively updating the node representations to H^{l+1} from the current representation H^l and the connections between the nodes E. W^l is the parameters for layer l. Note that $H^0 = X$, the initial node representations. The final prediction is made by applying a softmax operation (SM) on the final representation H^L predicting the class distribution. We formulate context-informed label prediction in a conditional random field framework where the conditional distribution over the labels are defined as

$$p(Y|X, E) = \frac{1}{Z(X, E)} \prod_{(i,j)\in E} \phi_{i,i}(y_i, y_j, X), \qquad (2)$$

where $Z(X, E)$ is the partition function over the graph and $\phi_{i,i}(y_i, y_j, X)$ is the potential function over a pair of nodes i, j. Let us denote θ as the graph parameters. Then, learning can be done by maximizing the following conditional log-likelihood:

$$\ell_{Y|X}(\theta) = \log p_\theta(Y|X, \theta). \qquad (3)$$

However, directly maximizing this likelihood function is intractable due to the combinatorial nature of the partition function. Thus, we consider optimizing the approximate likelihood that is shown to be successful in learning similar graphical models.

$$\ell_{Y|X}(\theta) = \sum_i \log p_\theta(y_i|y_{j\in B(i)}, X, \theta), \qquad (4)$$

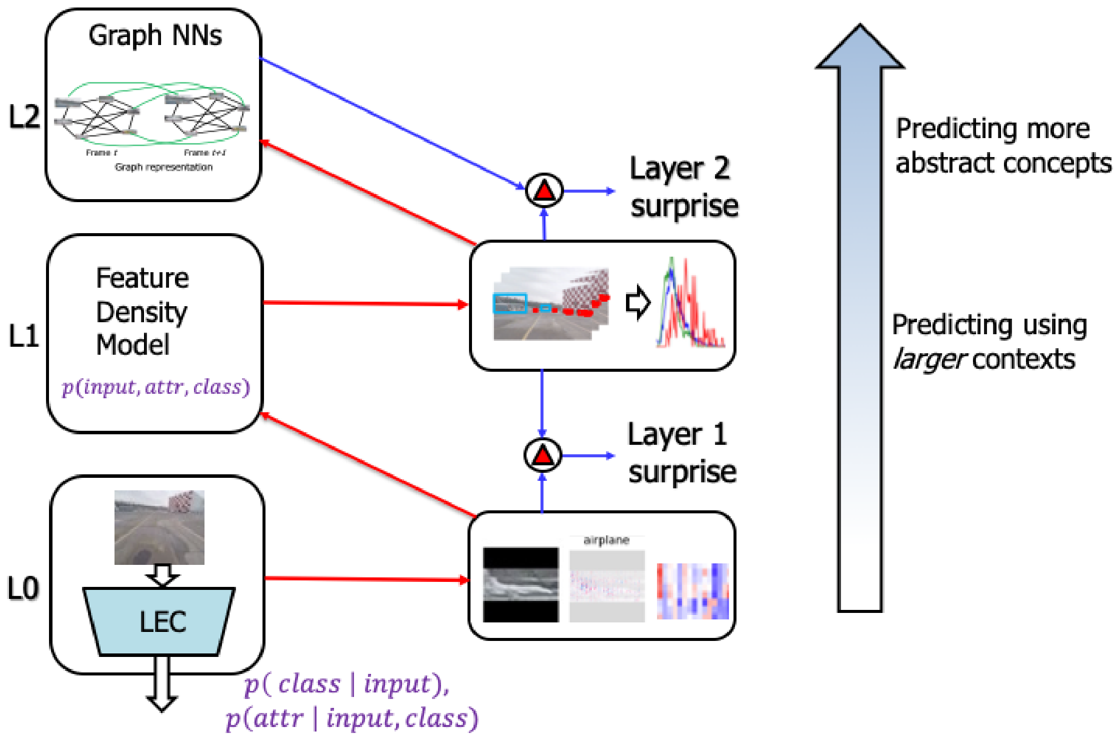

Figure 1: The level L0 represents the original deep neural network, which given an input x makes a decision or prediction d on it. For a classification model, the predicted output would be the class. In addition to the output, we can also compute the explanation of the decision e by the model using attribution methods (Simonyan, Vedaldi, and Zisserman 2013; Selvaraju et al. 2017; Sundararajan, Taly, and Yan 2017; Lundberg and Lee 2017). The predicted class and these attributions over the features (the layer before softmax) form the context $c_1 = (d, e)$ in layer 1 and a density model $p_1(x, d, e)$ is learned to represent the joint distribution of this context and the input using a trusted training set. When the original DNN makes a prediction at runtime, we use the predicted output class and the attribution as context and the learned L1 density model to compute the likelihood $p_1(x|c_1) = p_1(x|d, e)$ of the input. If this likelihood is low, it implies that the runtime monitor has detected that the underlying machine learning model has been surprised, and hence, cannot be trusted. The second layer uses graph Markov neural networks to learn the spatial and temporal distributions of the objects, which is useful in the object detection task. If the input x is a scene, we can decompose it into the individual objects x_i and the relationships between them, r_{ij} and the graph Markov neural networks learn the distribution $p(x_1, \ldots, x_j, \ldots, r_{ij} \ldots)$ using a trusted training set. In level L2, we can perform deliberation at runtime by treating all but one (say, x_k) objects as a context and then evaluating the likelihood of the predicted class of the output for a new input $p_2(x_k|c_2) = p_2(x_k|x_1, \ldots, x_j, \ldots, r_{ij} \ldots$. If this likelihood is low, the level L2 of the runtime monitor has detected a surprise and the underlying machine learning model cannot be trusted. Each layer of the runtime monitor uses larger and more abstract concepts to learn the context, and then uses top down prediction to determine whether the input is likely in the given context or not.

where $B(i)$ is the set of neighboring nodes of i. Intuitively, with this approximation, we only consider the context dependencies of the neighboring nodes. However, as we consider GCN to capture these dependencies, a few iterations of GCN allows capturing log-range dependencies by iterative message passing. Note that even in Eqn. 4, the label of a node y_i is conditioned on both the neighboring context $(y_{j \in B(i)})$ and the feature representation X. We avoid this dependency, we propose an iterative optimization where in one iteration we optimize to learn only representation and context dependency in another. This allows each iteration to be optimized efficiently is a GCN framework.

In the first iteration, we consider a mean-field approximation to remove context dependency and learn only the representation with RepG. Thus, the label distribution is defined as

$$p_{\theta_R}(Y|X, E) = \prod_i p_{\theta_R}(y_i|X, E), \quad (5)$$

where θ_R is the parameters for the representation graph RepG. Our representation graph is implemented by a GCN, where object features are used as node features.

$$H_R^{l+1} = f_R^l(W_R^l, H_R^l, E), \; p(y_i|X_R, E) = \text{SM}(H_R^L), \quad (6)$$

where X_R denote the node features and W_R denotes the GCN parameters. We consider bounding box features as the node features. Specifically, we consider ResNet features as X_R. It is evident from Eqn. 1 and Eqn. 5 that neighboring nodes share feature representations by message passing, but context related to object labels are not shared.

In the second iteration, we aim to predict labels from the context using ConG. This is also realized by a context GCN as follows.

$$H_C^{l+1} = f_C^l(W_C^l, H_C^l, E),\ p(y_i|X_C, E) = \text{SM}(H_C^L),\quad (7)$$

where X_C denote the node features and W_C denotes the GCN parameters. We consider context features as the node features including labels, position, size of neighboring nodes.

Thus, we learn this Layer 2 model iteratively using the expectation-maximization (EM) framework. In the M-step, we learn the representation graph based on the predicted labels of the context graph and in the E-step, keeping the representation fixed, we update the context graph based on the ground truth labels. Specifically, in E-step, we start by a pre-trained RepG to predict node labels. Then the labels are updated by a fixed ConG by aggregating context from the neighboring nodes. Finally. the RepG is updated to match ConG's predictions. In the M-step, we update ConG based on the ground-truth labels. We continue these iterations until convergence, that is, the difference of the predictions between RepG and ConG is constant. For runtime monitoring, we can now make deliberative queries to this learned model where we can fix the labels of any k objects and compute the likelihood of the other labels being correct. In our experiments, we restrict ourselves to querying the likelihood of one object at a time and the total surprise is computed as the sum of the log-likelihoods.

Experimental Evaluation

In order to evaluate our runtime monitoring framework, we test the effectiveness of each of the two layers separately. First, we evaluate how layer 1 can detect out of distribution inputs and novel classes in the object classification task. Then, we evaluate how layer 2 can detect novel contexts on an out of context dataset.

Layer 1 We evaluate Layer 1 on CIFAR10 (Krizhevsky, Hinton et al. 2009) and SVHN (Netzer et al. 2011) datasets.To demonstrate that the proposed approach generalizes across network architectures, we consider a wide range of DNN models such as ResNet (He et al. 2016) which consists of two convolutional layers and two fully connected layers, WideResNet (Zagoruyko and Komodakis 2016) which consists of a series of residual convolutional blocks followed by fully connected layers, and DenseNet (Huang et al. 2017) where outputs of all previous layers are directly connected to the output of a current layer. With CIFAR10 as in-distribution, we consider SVHN (Netzer et al. 2011) and Tiny-Imagenet (Deng et al. 2009) as the OOD datasets. For CIFAR10, we consider two DNNs: ResNet50, and WideResNet. Table 1 shows the results. With SVHN as in-distribution, we consider CIFAR10 (Krizhevsky, Hinton et al. 2009) and Imagenet (Deng et al. 2009) as the OOD datasets. For SVHN, we use the DenseNet classifier. We consider standard metrics (Hendrycks and Gimpel 2016) such as the true negative rate (TNR) at 95% true positive rate (TPR), the area under the receiver operating characteristic curve (AUROC), area under precision recall curve (AUPR), and the detection accuracy (DTACC) to evaluate our performance. We compare our approach with the three SOTA approaches: SPB (Hendrycks and Gimpel 2016), ODIN (Liang, Li, and Srikant 2017), and Mahalanobis (Lee et al. 2018).

Our runtime monitoring does not require any additional generation of auxiliary data for training the feature density model or exposure to OOD samples. Level 1 itself is competitive to the state of the art OOD detection methods.

Layer 2 In order to evaluate the effectiveness of the Layer 2 in capturing relational context and use it to detect inputs where the objects of known classes appear in a new context for object detection task, we created a new dataset where COCO objects (Lin et al. 2014) are put in surprising novel contexts. Figure 2 shows one example of such an image. Our dataset has 106060 such out of context (OOC) images generated using instances of COCO objects. We call this dataset COCO-OOC dataset. When using Layer 2 to detect OOC objects, we use both the relational context network ConG and the representation network RepG. Both of these are trained via expectation maximization, and they are expected to have similar accuracy for the in distribution inputs. We evaluate whether either of these are better at detecting out of context inputs.

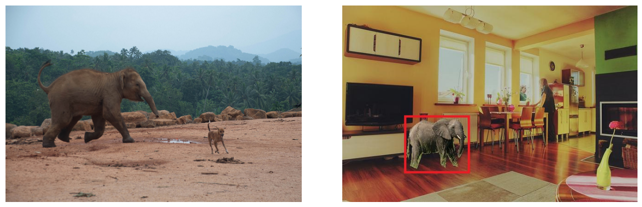

Figure 2: Left: Objects appear in context. Right: The 'elephant' object is out-of-context. Layer 2 of our runtime monitor is expected to detect such out of context inputs where the relation between objects of known classes is unusual.

Following the work on detecting OODs in image classification, we consider the predicted softmax score from the graph convolution networks as a simple baseline. The assumption is that for OOC objects, the softmax score is expected to be lower than the in-context objects. Thus, the softmax score can also be used to identify OOC objects. This softmax score can be from either of the two networks - representation graph network RepG and context graph network ConG. The AUC-ROC score for the Level 2 of our runtime monitor along with the softmax score baseline for both of the graph networks is presented in Table 2.

While our Layer 2 is able to detect the out of context inputs, it also serves as an effective object detector, achieving high accuracy on in-distribution objects. It naturally has low accuracy on out-of-context inputs and thus, the surprise reported from Layer 2 correlates well with the accuracy of the object detector.

Table 1: Level 1 Evaluation: Comparison of TNR, AUROC, DTACC with SPB, ODIN and Mahalanobis methods. Our runtime monitoring does not require any additional input pre-processing, generation of auxiliary data for training the feature density model or exposure to OOD samples. Level 1 itself is still competitive to the state of the art OOD detection methods.

In-dist (model)	**OOD Dataset**	**Method**	**TNR**	**AUROC**	**DTACC**
CIFAR10 (ResNet50)	SVHN	SPB	44.69	**97.31**	86.36
		ODIN	63.57	93.53	86.36
		Mahalanobis	72.89	91.53	85.39
		Ours	**83.68**	93.15	**88.14**
	Imagenet	SPB	42.06	90.8	84.36
		ODIN	79.48	96.25	90.07
		Mahalanobis	94.26	**97.41**	95.16
		Ours	**94.58**	97.14	**95.81**
CIFAR10 (WideResNet)	SVHN	SPB	45.46	90.10	82.91
		ODIN	57.14	89.30	81.14
		Mahalanobis	85.86	**97.21**	91.87
		Ours	**86.86**	94.47	**92.41**
SVHN (DenseNet)	Imagenet	SPB	79.79	94.78	90.21
		ODIN	79.8	94.8	90.2
		Mahalanobis	**99.85**	**99.88**	**98.87**
		Ours	97.62	98.25	96.74
	CIFAR10	SPB	69.31	91.9	86.61
		ODIN	69.3	91.9	86.6
		Mahalanobis	**97.03**	**98.92**	**96.11**
		Ours	93.54	96.68	95.36

Table 2: AUC-ROC score of detection of out of context inputs in the COCO-OOC dataset (higher is better)

Approach	AUC score
Softmax confidence	0.043
RepG	0.589
RepG + ConG	0.980

Table 3: Accuracy on COCO-OOC dataset. For OOC instances, **lower accuracy** is consistent with measured surprise of runtime monitor, and for non-OOC instances **higher** accuracy reflects well-trained model.

Approach	OOC$\downarrow$	non-OOC$\uparrow$	Overall
RepG	0.69	0.77	0.76
RepG + ConG	0.30	0.98	0.93

Conclusion

In this paper, we presented a two-layered runtime monitoring architecture for DNNs using top-down context models inspired by predictive processing and dual process theory. Within this architecture, the first layer consisted of a feature density model that learns the joint distribution of the original DNN's inputs, predicted class outputs and the model's explanation for its decisions. The second layer consisted of a graph Markov neural network to learn spatial and temporal relationships between components of an input. In our experimental evaluation, we demonstrated the high performance (accuracy) of our monitoring architecture in detecting out of distribution inputs in an object classification task (layer 1) and detecting novel context on out of context inputs in an object detection task (layer 2). While runtime monitoring can facilitate the integration of machine learning models into high-assurance applications where traditional back-up systems can take over decision-making when the machine learning models cannot be trusted, we are also investigating methods that exploit runtime monitoring for continuous lifelong learning.

Acknowledgements

We are thankful to Dr. John Rushby and Dr. Natarajan Shankar for discussions on assurance principles. This work was supported in part by the U.S. Army Combat Capabilities Development Command (DEVCOM) Army Research Laboratory under Cooperative Agreement W911NF-17-2-0196, Support Agreement No. USMA21050 and the U.S. National Science Foundation grant #1740079. The views expressed in this paper are those of the authors and do not reflect the official policy or position of the United States Army, the United States Department of Defense, or the United States Government.

References

2019. *Vehicle Automation Report; Tempe, AZ.* National Transportation Safety Board. HWY18MH010.

Abdelzaher, T.; Ayanian, N.; Basar, T.; Diggavi, S.; Diesner, J.; Ganesan, D.; Govindan, R.; Jha, S.; Lepoint, T.; Marlin, B.; et al. 2018. Toward an internet of battlefield things: A resilience perspective. *Computer*, 51(11): 24–36.

An, J.; and Cho, S. 2015. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2(1): 1–18.

Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; and Wierstra, D. 2015. Weight uncertainty in neural network. In *International Conference on Machine Learning*, 1613–1622. PMLR.

Bojarski, M.; Del Testa, D.; Dworakowski, D.; Firner, B.; Flepp, B.; Goyal, P.; Jackel, L. D.; Monfort, M.; Muller, U.; Zhang, J.; et al. 2016. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

Choi, M. J.; Lim, J. J.; Torralba, A.; and Willsky, A. S. 2010. Exploiting Hierarchical Context on a Large Database of Object Categories. In *IEEE Conference on Computer VIsion and Pattern Recognition (CVPR)*.

Choi, M. J.; Torralba, A.; and Willsky, A. S. 2012. Context models and out-of-context objects. *Pattern Recognition Letters*, 33(7): 853–862.

Clark, A. 2013. Whatever Next? Predictive Brains, Situated Agents, and the Future of Cognitive Science. *Behavioral and Brain Sciences*, 36(3): 181–204.

Dai, H.; Dai, B.; and Song, L. 2016. Discriminative embeddings of latent variable models for structured data. In *International conference on machine learning*, 2702–2711. PMLR.

De Fauw, J.; Ledsam, J. R.; Romera-Paredes, B.; Nikolov, S.; Tomasev, N.; Blackwell, S.; Askham, H.; Glorot, X.; O'Donoghue, B.; Visentin, D.; et al. 2018. Clinically applicable deep learning for diagnosis and referral in retinal disease. *Nature medicine*, 24(9): 1342–1350.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Divvala, S. K.; Hoiem, D.; Hays, J. H.; Efros, A. A.; and Hebert, M. 2009. An empirical study of context in object detection. In *2009 IEEE Conference on computer vision and Pattern Recognition*, 1271–1278. IEEE.

Dusenberry, M.; Jerfel, G.; Wen, Y.; Ma, Y.; Snoek, J.; Heller, K.; Lakshminarayanan, B.; and Tran, D. 2020. Efficient and scalable bayesian neural nets with rank-1 factors. In *International conference on machine learning*, 2782–2792. PMLR.

Dvornik, N.; Mairal, J.; and Schmid, C. 2018. Modeling visual context is key to augmenting object detection datasets. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 364–380.

Evans, J. S. B. 2010. Intuition and reasoning: A dual-process perspective. *Psychological Inquiry*, 21(4): 313–326.

Evans, J. S. B. T.; and Stanovich, K. E. 2013. Dual-Process Theories of Higher Cognition: Advancing the Debate. *Perspectives on Psychological Science*, 8(3): 223–241.

Frankish, K. 2010. Dual-Process and Dual-System Theories of Reasoning. *Philosophy Compass*, 5(10): 914–926.

Gal, Y.; and Ghahramani, Z. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, 1050–1059. PMLR.

Gkioxari, G.; Girshick, R.; and Malik, J. 2015. Contextual action recognition with r* cnn. In *Proceedings of the IEEE international conference on computer vision*, 1080–1088.

Gould, S.; Rodgers, J.; Cohen, D.; Elidan, G.; and Koller, D. 2008. Multi-class segmentation with relative location prior. *International journal of computer vision*, 80(3): 300–316.

Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; and Courville, A. 2017. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*.

Guo, C.; Pleiss, G.; Sun, Y.; and Weinberger, K. Q. 2017a. On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*.

Guo, C.; Pleiss, G.; Sun, Y.; and Weinberger, K. Q. 2017b. On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 1025–1035.

Hannun, A.; Case, C.; Casper, J.; Catanzaro, B.; Diamos, G.; Elsen, E.; Prenger, R.; Satheesh, S.; Sengupta, S.; Coates, A.; et al. 2014. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hendrycks, D.; and Gimpel, K. 2016. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*.

Hendrycks, D.; Mazeika, M.; and Dietterich, T. 2019. Deep anomaly detection with outlier exposure. *In International Conference on Learning Representations*.

Hendrycks, D.; Mazeika, M.; Kadavath, S.; and Song, D. 2019. Using self-supervised learning can improve model robustness and uncertainty. In *Advances in Neural Information Processing Systems*, 15663–15674.

Hohwy, J. 2013. *The Predictive Mind*. Oxford University Press.

Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4700–4708.

Jang, U.; Jha, S.; and Jha, S. 2019. On Need for Topology Awareness of Generative Models. *arXiv preprint arXiv:1909.03334*.

Jha, S.; Jang, U.; Jha, S.; and Jalaian, B. 2018a. Detecting adversarial examples using data manifolds. In *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*, 547–552. IEEE.

Jha, S.; Jang, U.; Jha, S.; and Jalaian, B. 2018b. Detecting adversarial examples using data manifolds. In *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*, 547–552. IEEE.

Jha, S.; Raj, S.; Fernandes, S.; Jha, S. K.; Jha, S.; Jalaian, B.; Verma, G.; and Swami, A. 2019a. Attribution-Based Confidence Metric For Deep Neural Networks. In *Advances in Neural Information Processing Systems*, 11826–11837.

Jha, S.; Raj, S.; Fernandes, S. L.; Jha, S. K.; Jha, S.; Verma, G.; Jalaian, B.; and Swami, A. 2019b. Attribution-driven Causal Analysis for Detection of Adversarial Examples. *arXiv preprint arXiv:1903.05821*.

Jha, S.; Rushby, J.; and Shankar, N. 2020. Model-Centered Assurance for Autonomous Systems. In *International Conference on Computer Safety, Reliability, and Security*, 228–243. Springer.

Jiang, H.; Kim, B.; Guan, M.; and Gupta, M. 2018. To trust or not to trust a classifier. In *Advances in neural information processing systems*, 5541–5552.

Julian, K. D.; and Kochenderfer, M. J. 2017. Neural network guidance for UAVs. In *AIAA Guidance, Navigation, and Control Conference*, 1743.

Kahneman, D. 2011. *Thinking, Fast and Slow*. Farrar, Straus and Giroux.

Kipf, T. N.; and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Kirichenko, P.; Izmailov, P.; and Wilson, A. G. 2020. Why normalizing flows fail to detect out-of-distribution data. *arXiv preprint arXiv:2006.08545*.

Koller, D.; and Friedman, N. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Lakshminarayanan, B.; Pritzel, A.; and Blundell, C. 2016. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*.

Lee, K.; Lee, H.; Lee, K.; and Shin, J. 2017. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *arXiv preprint arXiv:1711.09325*.

Lee, K.; Lee, K.; Lee, H.; and Shin, J. 2018. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, 7167–7177.

Liang, S.; Li, Y.; and Srikant, R. 2017. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*.

Lin, S.-C.; et al. 2018. The Architectural Implications of Autonomous Driving: Constraints and Acceleration. *ACM SIGPLAN Notices*, 53(2): 751–766. (Proceedings of ASPLOS'18, Williamsburg, VA).

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, 740–755. Springer.

Liu, J. Z.; Lin, Z.; Padhy, S.; Tran, D.; Bedrax-Weiss, T.; and Lakshminarayanan, B. 2020a. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *arXiv preprint arXiv:2006.10108*.

Liu, W.; Wang, X.; Owens, J.; and Li, S. Y. 2020b. Energy-based Out-of-distribution Detection. *Advances in Neural Information Processing Systems*, 33.

Lundberg, S. M.; and Lee, S.-I. 2017. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, 4765–4774.

Majumder, N.; Poria, S.; Gelbukh, A.; and Cambria, E. 2017. Deep learning-based document modeling for personality detection from text. *IEEE Intelligent Systems*, 32(2): 74–79.

Mandelbaum, A.; and Weinshall, D. 2017. Distance-based confidence score for neural network classifiers. *arXiv preprint arXiv:1709.09844*.

McLachlan, G. J.; and Basford, K. E. 1988. *Mixture models: Inference and applications to clustering*, volume 38. M. Dekker New York.

Miyato, T.; Kataoka, T.; Koyama, M.; and Yoshida, Y. 2018. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*.

Mukhoti, J.; Kirsch, A.; van Amersfoort, J.; Torr, P. H.; and Gal, Y. 2021. Deterministic neural networks with appropriate inductive biases capture epistemic and aleatoric uncertainty. *arXiv preprint arXiv:2102.11582*.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; and Ng, A. Y. 2011. Reading digits in natural images with unsupervised feature learning.

Oh, S. J.; Murphy, K.; Pan, J.; Roth, J.; Schroff, F.; and Gallagher, A. 2018. Modeling uncertainty with hedged instance embedding. *arXiv preprint arXiv:1810.00319*.

Papernot, N.; and McDaniel, P. 2018. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *arXiv preprint arXiv:1803.04765*.

Qu, M.; Bengio, Y.; and Tang, J. 2019. Gmnn: Graph markov neural networks. In *International conference on machine learning*, 5241–5250. PMLR.

Rao, R. P.; and Ballard, D. H. 1999. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1): 79–87.

Ren, J.; Liu, P. J.; Fertig, E.; Snoek, J.; Poplin, R.; Depristo, M.; Dillon, J.; and Lakshminarayanan, B. 2019. Likelihood ratios for out-of-distribution detection. In *Advances in Neural Information Processing Systems*, 14707–14718.

Selvaraju, R. R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; and Batra, D. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *CVPR*, 618–626.

Simonyan, K.; Vedaldi, A.; and Zisserman, A. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Spratling, M. W. 2017. A Review of Predictive Coding Algorithms. *Brain and cognition*, 112: 92–97.

Sundararajan, M.; Taly, A.; and Yan, Q. 2017. Axiomatic attribution for deep networks. In *ICML*, 3319–3328. JMLR. org.

Tack, J.; Mo, S.; Jeong, J.; and Shin, J. 2020. Csi: Novelty detection via contrastive learning on distributionally shifted instances. *Advances in Neural Information Processing Systems*, 33.

van Amersfoort, J.; Smith, L.; Teh, Y. W.; and Gal, Y. 2020. Simple and scalable epistemic uncertainty estimation using a single deep deterministic neural network.

von Helmholtz, H. 1867. *Handbuch der Physiologischen Optik III*, volume 9. Leipzig, Germany: Verlag von Leopold Voss.

Wen, Y.; Tran, D.; and Ba, J. 2020. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. *arXiv preprint arXiv:2002.06715*.

Wiese, W.; and Metzinger, T. K. 2017. Vanilla PP for Philosophers: A Primer on Predictive Processing. In Metzinger, T. K.; and Wiese, W., eds., *Philosophy and Predictive Processing*, chapter 1. Frankfurt am Main: MIND Group.

Winkens, J.; Bunel, R.; Roy, A. G.; Stanforth, R.; Natarajan, V.; Ledsam, J. R.; MacWilliams, P.; Kohli, P.; Karthikesalingam, A.; Kohl, S.; et al. 2020. Contrastive training for improved out-of-distribution detection. *arXiv preprint arXiv:2007.05566*.

Zagoruyko, S.; and Komodakis, N. 2016. Wide residual networks. *arXiv preprint arXiv:1605.07146*.

Zhang, Y.; and Chen, T. 2012. Efficient inference for fully-connected CRFs with stationarity. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 582–589. IEEE.

Zisselman, E.; and Tamar, A. 2020. Deep residual flow for out of distribution detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13994–14003.