Journal of Machine Learning Research 23 (2022) 1-33 Submitted 7/21; Revised 12/21; Published 1/22

Analytically Tractable Hidden-States Inference in Bayesian Neural Networks

Luong-Ha Nguyen[*] LUONGHA.NGUYEN@GMAIL.COM
James-A. Goulet[*] JAMES.GOULET@POLYMTL.CA
Department of Civil Engineering, Polytechnique Montréal, Montréal, Canada

Editor: Philipp Hennig

Abstract

With few exceptions, neural networks have been relying on backpropagation and gradient descent as the inference engine in order to learn the model parameters, because closed-form Bayesian inference for neural networks has been considered to be intractable. In this paper, we show how we can leverage the *tractable approximate Gaussian inference*'s (TAGI) capabilities to infer hidden states, rather than only using it for inferring the network's parameters. One novel aspect is that it allows inferring hidden states through the imposition of constraints designed to achieve specific objectives, as illustrated through three examples: (1) the generation of adversarial-attack examples, (2) the usage of a neural network as a black-box optimization method, and (3) the application of inference on continuous-action reinforcement learning. In these three examples, the constrains are in (1), a target label chosen to fool a neural network, and in (2 & 3) the derivative of the network with respect to its input that is set to zero in order to infer the optimal input values that are either maximizing or minimizing it. These applications showcase how tasks that were previously reserved to gradient-based optimization approaches can now be approached with analytically tractable inference.

Keywords: Bayesian, neural networks, TAGI, Gaussian inference, approximate inference, adversarial attack, optimization, reinforcement learning

1. Introduction

With few exceptions, neural networks have been relying on backpropagation (Rumelhart et al., 1986) and gradient descent as the inference engine in order to learn the model parameters. In such a case, the inference can be seen as approximating the posterior by a point solution minimizing a loss function. In addition to learning the model parameters, one may be interested in inferring the values of hidden states in a neural network. Note that we are not interested here in cases such as variational auto-encoders (Kingma and Welling, 2014), or in generative adversarial networks (Goodfellow et al., 2014; Chen et al., 2016) where dedicated latent variables are added; we are rather interested in inferring the value of hidden states from single observation instances. A first example is the case of adversarial attacks (AA), where images can be tailored in order to fool a neural network into performing incorrect classifications with high certainty (Goodfellow et al., 2015). In the context of white-box AA, images that seem realistic for a human observer, are generated by inferring

[*] Equal contribution

perturbations that can be added to the input layer of a neural network in order to fool it. A second example of hidden state inference involves the definition of policy networks in reinforcement learning (RL) with methods such as advanced actor critic (A2C) (Mnih et al., 2016) and proximal policy optimization (PPO) (Schulman et al., 2017). For such cases, current methods relying on backpropagation use gradient ascent in order to infer the optimal actions that are maximizing an action-value function (Sutton and Barto, 2018).

The closed-form Bayesian inference for neural networks has long been considered to be intractable, both in terms of its parameters (Goodfellow et al., 2016) or hidden states (Ardizzone et al., 2019; Kruse et al., 2021). Recently, the *tractable approximate Gaussian inference* (TAGI) (Goulet et al., 2021) method was shown to either be on par or exceed the performance of neural networks trained with backpropagation in fully connected architectures (Goulet et al., 2021), for convolutional (CNN) and generative ones (Nguyen and Goulet, 2021b), as well as for deep reinforcement learning with categorical actions (Nguyen and Goulet, 2021a). This paper shows how we can leverage TAGI's probabilistic inference capabilities to infer hidden states, rather than only using it for inferring the network's parameters. One novel aspect introduced is the capacity to infer hidden states through the imposition of constraints designed to attain specific objectives, as illustrated in this paper through three examples: (1) the generation of adversarial-attack examples, (2) the usage of a neural network as a black-box optimization method, and (3) the application of inference on continuous-action reinforcement learning. In these three examples, the constrains are in (1), a target label chosen to fool a neural network, and in (2 & 3) the derivative of the network with respect to its input that is set to zero in order to infer the optimal input values that are either maximizing or minimizing it. The paper is organized such that before diving in the theory and examples for these applications in Sections 3-5, Section 2 reviews the theory behind TAGI. Our goal throughout this paper is to demonstrate that a paradigm other than gradient-based optimization is possible through analytical inference.

2. Tractable Approximate Gaussian Inference

In terms of notation, we use lower cases letter for deterministic variables, upper cases for random variables, and bold fonts to denote either vector or matrices. Consider a feedforward neural network (FNN) where the $\boldsymbol{x} \in \mathbb{R}^{\mathtt{X}}$ is the input layer, $\boldsymbol{z}^{(j)} \in \mathbb{R}^{\mathtt{A}^{(j)}}$ is the j-th layer of $\mathtt{A}^{(j)}$ hidden units, $\phi(\cdot)$ is an activation function so that $\boldsymbol{a}^{(j)} = \phi(\boldsymbol{z}^{(j)}) \in \mathbb{R}^{\mathtt{A}^{(j)}}$ are the activation units from the j-th hidden layer, $\boldsymbol{z}^{(\mathtt{O})} \in \mathbb{R}$ is the output unit, $\boldsymbol{y} \in \mathbb{R}$ are the observations, and where the parameters $\boldsymbol{\theta}$ are the weights and biases defining the relationships between layers.

TAGI (Goulet et al., 2021) assumes that the joint distribution between the observations and a neural network's parameters is approximated by a multivariate Gaussian distribution,

$$f\begin{pmatrix}\boldsymbol{\theta}\\ \boldsymbol{y}\end{pmatrix} = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\theta}\\ \boldsymbol{y}\end{bmatrix};\begin{bmatrix}\boldsymbol{\mu}_{\boldsymbol{\theta}}\\ \boldsymbol{\mu}_{\boldsymbol{Y}}\end{bmatrix},\begin{bmatrix}\boldsymbol{\Sigma}_{\boldsymbol{\theta}} & \boldsymbol{\Sigma}_{\boldsymbol{Y\theta}}^{\intercal}\\ \boldsymbol{\Sigma}_{\boldsymbol{Y\theta}} & \boldsymbol{\Sigma}_{\boldsymbol{Y}}\end{bmatrix}\right),$$

so that the parameter inference can build upon the Gaussian conditional equation describing the probability density function (PDF) of $\boldsymbol{\theta}$ conditional on observations $\boldsymbol{y}$,

$$\begin{array}{rcl} f(\boldsymbol{\theta}|\boldsymbol{y}) & = & \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}_{\boldsymbol{\theta}|\boldsymbol{y}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}|\boldsymbol{y}}) \\ \boldsymbol{\mu}_{\boldsymbol{\theta}|\boldsymbol{y}} & = & \boldsymbol{\mu}_{\boldsymbol{\theta}} + \boldsymbol{\Sigma}_{\boldsymbol{Y\theta}}^{\intercal} \boldsymbol{\Sigma}_{\boldsymbol{Y}}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}_{\boldsymbol{Y}}) \\ \boldsymbol{\Sigma}_{\boldsymbol{\theta}|\boldsymbol{y}} & = & \boldsymbol{\Sigma}_{\boldsymbol{\theta}} - \boldsymbol{\Sigma}_{\boldsymbol{Y\theta}}^{\intercal} \boldsymbol{\Sigma}_{\boldsymbol{Y}}^{-1} \boldsymbol{\Sigma}_{\boldsymbol{Y\theta}}. \end{array}$$

The approach is inherently divided in two steps; first propagate uncertainties through the network in order to obtain the joint PDFs between the quantities to be updated (that is, neural network's parameters and hidden state units) and the observations, and then update these quantities using the Gaussian conditional equations. The first key operation to be considered is the propagation of uncertainty from the activation units $\boldsymbol{A}^{(j)} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{A}}^{(j)}, \boldsymbol{\Sigma}_{\boldsymbol{A}}^{(j)})$ of a hidden layer j to a hidden unit $Z_i^{(j+1)}$ on the subsequent layer $j+1$

$$Z_i^{(j+1)} = \sum_{k=1}^{\mathtt{A}} W_{i,k}^{(j)} A_k^{(j)} + B_i^{(j)}, \tag{1}$$

where $W_{i,k}^{(j)}$ are weights and $B_i^{(j)}$ bias parameters that are modelled by Gaussian random variables. In order to maintain the analytical tractability of Equation 1, TAGI approximates the product of any pair of weight and activation unit by a Gaussian random variable $WA \approx \mathcal{N}(\mu_{WA}, \sigma_{WA}^2)$, for which the exact moments are computed analytically using *Gaussian multiplicative approximation* (GMA), see Appendix B.2 for details. The second key operation is the propagation of uncertainty through non-linear activation functions

$$A_i^{(j+1)} = \phi\left(Z_i^{(j+1)}\right), \tag{2}$$

where, in order to maintain the analytical tractability, TAGI locally linearize $\phi(\cdot)$ at the expected value of the hidden units $\mu_{Z_i}^{(j+1)}$. Maintaining the computational tractability of equations 1 and 2 requires assuming diagonal covariance structures for the hidden units among a same layer $\boldsymbol{\Sigma}_{\boldsymbol{Z}}^{(j)}$, and for the parameters $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$.

The update step, that is, Gaussian conditional inference step, is performed using a recursive layer-wise procedure; Using the short-hand notation $\{\boldsymbol{\theta}^{+}, \boldsymbol{Z}^{+}\} \equiv \{\boldsymbol{\theta}^{(j+1)}, \boldsymbol{Z}^{(j+1)}\}$ and $\{\boldsymbol{\theta}, \boldsymbol{Z}\} \equiv \{\boldsymbol{\theta}^{(j)}, \boldsymbol{Z}^{(j)}\}$, the posteriors for the parameters and hidden states are computed following

$$\begin{array}{rlrl} f(\boldsymbol{Z}|\boldsymbol{y}) & = \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}_{\boldsymbol{Z}|\boldsymbol{y}}, \boldsymbol{\Sigma}_{\boldsymbol{Z}|\boldsymbol{y}}) & f(\boldsymbol{\theta}|\boldsymbol{y}) & = \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}_{\boldsymbol{\theta}|\boldsymbol{y}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}|\boldsymbol{y}}) \\ \boldsymbol{\mu}_{\boldsymbol{Z}|\boldsymbol{y}} & = \boldsymbol{\mu}_{\boldsymbol{Z}} + \mathbf{J}_{\boldsymbol{Z}}\left(\boldsymbol{\mu}_{\boldsymbol{Z}^{+}|\boldsymbol{y}} - \boldsymbol{\mu}_{\boldsymbol{Z}^{+}}\right) & \boldsymbol{\mu}_{\boldsymbol{\theta}|\boldsymbol{y}} & = \boldsymbol{\mu}_{\boldsymbol{\theta}} + \mathbf{J}_{\boldsymbol{\theta}}\left(\boldsymbol{\mu}_{\boldsymbol{Z}^{+}|\boldsymbol{y}} - \boldsymbol{\mu}_{\boldsymbol{Z}^{+}}\right) \\ \boldsymbol{\Sigma}_{\boldsymbol{Z}|\boldsymbol{y}} & = \boldsymbol{\Sigma}_{\boldsymbol{Z}} + \mathbf{J}_{\boldsymbol{Z}}\left(\boldsymbol{\Sigma}_{\boldsymbol{Z}^{+}|\boldsymbol{y}} - \boldsymbol{\Sigma}_{\boldsymbol{Z}^{+}}\right)\mathbf{J}_{\boldsymbol{Z}}^{\intercal} & \boldsymbol{\Sigma}_{\boldsymbol{\theta}|\boldsymbol{y}} & = \boldsymbol{\Sigma}_{\boldsymbol{\theta}} + \mathbf{J}_{\boldsymbol{\theta}}\left(\boldsymbol{\Sigma}_{\boldsymbol{Z}^{+}|\boldsymbol{y}} - \boldsymbol{\Sigma}_{\boldsymbol{Z}^{+}}\right)\mathbf{J}_{\boldsymbol{\theta}}^{\intercal} \\ \mathbf{J}_{\boldsymbol{Z}} & = \boldsymbol{\Sigma}_{\boldsymbol{Z}\boldsymbol{Z}^{+}} \boldsymbol{\Sigma}_{\boldsymbol{Z}^{+}}^{-1}, & \mathbf{J}_{\boldsymbol{\theta}} & = \boldsymbol{\Sigma}_{\boldsymbol{\theta}\boldsymbol{Z}^{+}} \boldsymbol{\Sigma}_{\boldsymbol{Z}^{+}}^{-1}. \end{array} \tag{3}$$

Note that the layer-wise recursive procedure defined in equations 3 only requires the storage of the joint prior PDFs for pairs of subsequent hidden layers and pairs of hidden layers and the parameters directly connecting into them. This allows maintaining the computational tractability of the uncertainty propagation and inference steps which scale linearly with respect to the number of weight parameters. Despite the approximations mentioned above, TAGI was shown to match the performance of FNN trained with backpropagation for regression and classification benchmarks.

The fundamental mathematical operations in convolutional neural networks (that is, additions, multiplications and non-linear activations) are no different from those in feedforward neural networks (Nguyen and Goulet, 2021b). Therefore, the mathematical formulation presented in §2 for defining the joint PDF for observations and parameters in FNN can be readily employed in deep convolutional architectures, with the minor addition of a *pooling* and *noise decay* formulation. Existing methods such as the stochastic max pooling introduced by Peters and Welling (2018) can be directly be applied with TAGI. In addition to this method, Nguyen and Goulet (2021b) have adapted an *average* pooling approach to the context of TAGI, where, as its name indicates, the output is the average of the activation units in the pooling kernel. For a $\mathtt{K}$-elements pooling kernel, the output Gaussian random variable is defined as

$$A^{\text{pool}} = \tfrac{1}{\mathtt{K}}(A_1 + A_2 + \cdots + A_{\mathtt{K}}). \tag{4}$$

Moreover, in the original TAGI formulation for FNN, the observation errors v standard deviation parameters σ_V were considered to be constant during the training. However, throughout empirical experimentation with CNNs, we noticed that the performance was improved when using a decay equation

$$\sigma_V^e = \eta \cdot \sigma_V^{e-1}, \tag{5}$$

where e is the current epoch and $\eta \in (0, 1]$ is the decay factor hyperparameter that needs to be learned outside of the TAGI analytic inference procedure. This approach is similar to what is done in standard deep neural networks trained with backpropagation where noise is added to the gradient with a decay schedule (Neelakantan et al., 2015). In the case of gradient-based learning, this noise consists in discrete samples added to the gradient itself whereas for TAGI, it consists in additional variance on the output layer so that the update during the inference step will put more weight on the prior rather than on the likelihood, with this effect diminishing with time.

In the experimental setups explored so far, the inference capacity of TAGI was employed to learn the neural network parameters whereas the updated knowledge regarding the hidden units is discarded each time new training observations become available. In the current setup, we are not only interested in using TAGI to infer the network's parameters, but also the hidden units at specific location within the network. The appeal of TAGI is that it can inherently do so, without requiring any modifications to its formulation. In the following subsections, we will present how the novel inference capacity from the TAGI method can be leveraged in order to provide new solutions to existing challenges such as adversarial-attack generation, black-box optimization, and continuous-action reinforcement learning.

3. Adversarial Attack through Inference

In the first example, we are interested in white-box adversarial attacks (Akhtar and Mian, 2018) where we have access to the network structure and its parameters. Current white-box attacks are typically formulated as an optimization problem where one uses gradient descent and backpropagation in order to find optimal perturbations to be applied on the input layer in order to fool the network into making wrongful classifications.

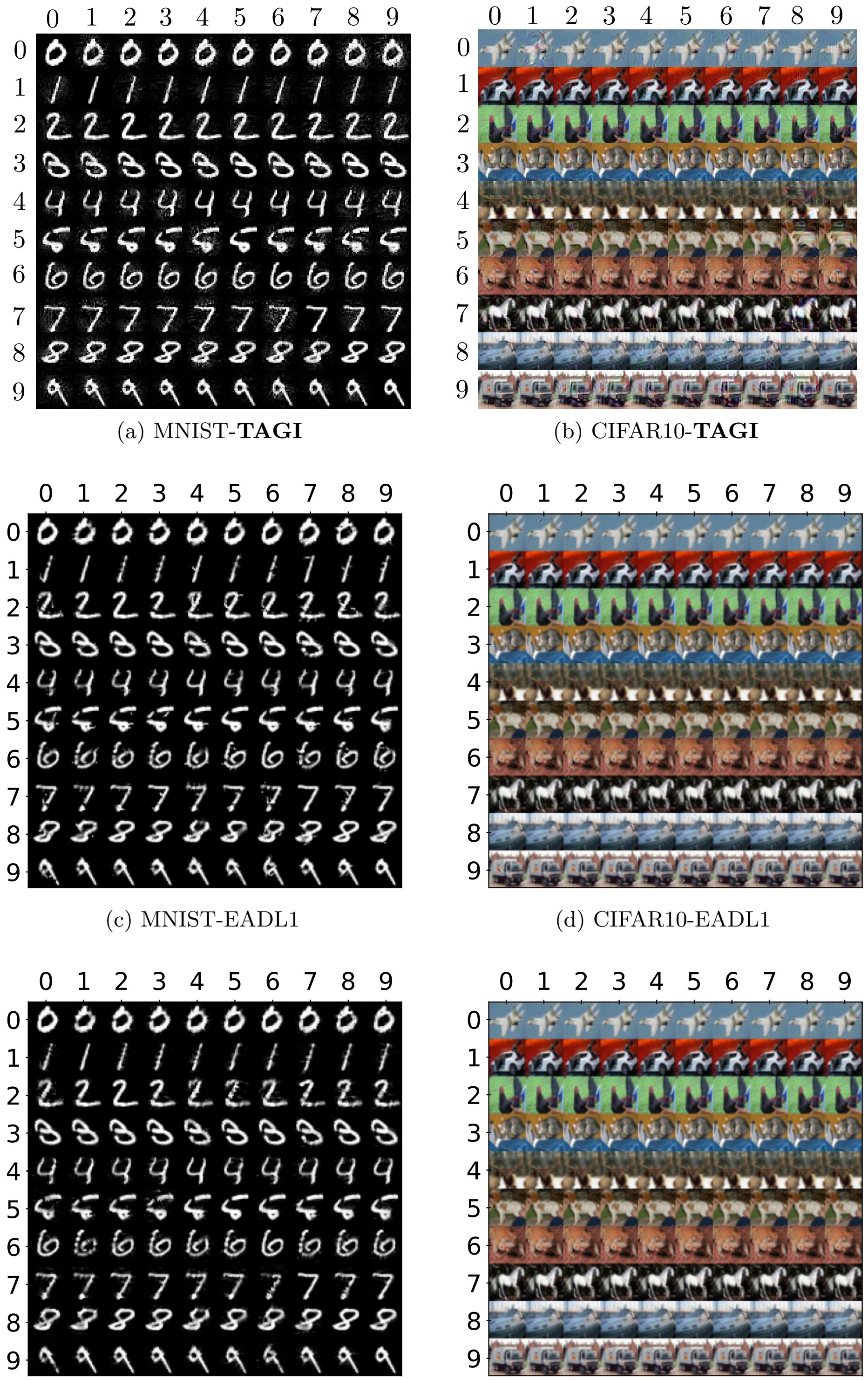
With TAGI, the generation of adversarial-attack images can be done analytically, without relying on an optimization process. We start with the assumption that we have a pre-trained neural network; Then, from a deterministic target image $\boldsymbol{x}$ of size $\mathtt{M} \times \mathtt{N}$, for which we want to obtain a corrupt label, we define the prior knowledge on the input layer by the mean vector corresponding to the deterministic image $\boldsymbol{\mu}_{\boldsymbol{X}} = \boldsymbol{x} \in \mathbb{R}^{\mathtt{M}\times\mathtt{N}}$, and a diagonal covariance $\boldsymbol{\Sigma}_{\boldsymbol{X}} = \sigma_X^2 \cdot \boldsymbol{I}$. Here, the amount of change that TAGI will apply on the original image during the inference procedure is controlled by the input layer's standard deviation parameter σ_X, an hyperparameter that need to be tuned in order to reflect the amount of change desired in the original image. This prior knowledge about the target image is propagated forward through the network analogously to the procedure presented in §2. The adversarial attack is then perform in the inference step where we use the target label $\tilde{y}$ chosen for the attack as an observation on the output layer. After performing the inference step analogously to the procedure presented in §2, the initial image defined by its updated mean vector $\boldsymbol{\mu}_{\boldsymbol{X}|\tilde{y}}$ and covariance $\boldsymbol{\Sigma}_{\boldsymbol{X}|\tilde{y}}$ is now modified in order to trigger the class $\tilde{y}$. In order to improve the quality of the attack, the process is repeated recursively over multiple iterations, where the inferred values $\{\boldsymbol{\mu}^{(i)}_{\boldsymbol{X}|\tilde{y}}, \boldsymbol{\Sigma}^{(i)}_{\boldsymbol{X}|\tilde{y}}\}$ at iteration i are used as the prior's hyper-parameters at the next iteration $i+1$.

We compare the performance of TAGI with EADL1 (Chen et al., 2018), PGDL2 (Madry et al., 2017), and CWL2 (Carlini and Wagner, 2017) on the MNIST (LeCun et al., 1998) and CIFAR10 (Krizhevsky et al., 2009) data sets. We employ the same network architectures for all approaches and the details of these architectures are presented in Appendix A. For TAGI, we set $\sigma_X = 0.03$ with a maximal number of epochs $\mathtt{E} = 100$. The hyperparameter σ_X was defined through experimentation as it lead to successful yet almost unnoticeable attacks across all the datatasets tested. For the gradient-based approaches, we use the implementation from Ding et al. (2019). The hyper-parameter values are provided in Table **??**. Figure 6 presents adversarial examples generated from TAGI, EADL1, and CWL2 approaches and the examples obtained using the PGDL2 approach are presented in Appendix A.1.3. From a visual standpoint the adversarial attacks generated with TAGI are on par with EADL1, CWL2 and PGDL2.

Table 1 compares the error rates obtained for each method: without attack, with targeted attacks where a specific class is seeked, and with non-targeted attacks where the goal is simply to fool the network. These results obtained for convolutional architectures confirm that TAGI can, without relying on an optimization scheme, infer adversarial-attack examples that are visually indistinguishable from the original and that match the performance of existing approaches.

4. Optimization through Inference

This section presents how we can leverage TAGI's inference capabilities to find the local maxima or minima of a function. Take for example a feedforward neural network used for approximating a function $g(\cdot)$ such that $z^{(\mathtt{O})} = g(x; \boldsymbol{\theta})$, where $x \in \mathbb{R}$ is a vector of covariates, $z^{(\mathtt{O})} \in \mathbb{R}$ is the neural network output, and $\boldsymbol{\theta}$ is a vector of parameters defining the weights $\boldsymbol{w}$ and biases $\boldsymbol{b}$ from the neural network. Figure 2 presents an example of a directed acyclic



(a) MNIST-**TAGI**

(b) CIFAR10-**TAGI**

(c) MNIST-EADL1

(d) CIFAR10-EADL1

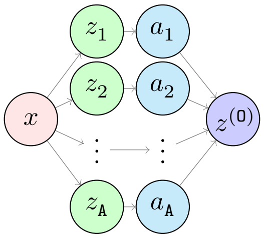
(e) MNIST-CWL2

(f) CIFAR10-CWL2

Figure 1: Examples of images subjected to adversarial attacks with different target labels $\tilde{y}$. Columns represent the different target labels $\tilde{y}$ and rows are the true label images.

			Error Rate [%]		
Data set	Model	Method	No attack	Targeted attack	Non-targeted attack
MNIST	2 conv.	**TAGI**	0.64	99.8	99.9
		EADL1	0.62	100	100
		PGDL2	0.62	96.9	97.5
		CWL2	0.62	99.2	99.2
CIFAR10	3 conv.	**TAGI**	22	99.6	99.9
		EADL1	22	100	100
		PGDL2	22	100	100
		CWL2	22	100	100

Table 1: Quantitative performance evaluation of TAGI's adversarial attacks on MNIST and CIFAR10.



$$\begin{array}{rcll} z^{(\mathtt{O})} & = & g(x;\boldsymbol{\theta}) = (\boldsymbol{w}^{(1)})^{\intercal}\boldsymbol{a} + b^{(\mathtt{O})} & \text{(Output variable)} \\ \boldsymbol{a} & = & \phi(\boldsymbol{z}) & \text{(Activation units)} \\ \boldsymbol{z} & = & \boldsymbol{w}^{(\mathtt{0})}x + \boldsymbol{b}^{(1)} & \text{(Hidden units)} \\ x & \in & \mathbb{R} & \text{(Input covariate)} \\ \boldsymbol{\theta} & = & \{\boldsymbol{w}^{(\mathtt{0})}, \boldsymbol{w}^{(1)}, \boldsymbol{b}^{(1)}, \boldsymbol{b}^{(\mathtt{O})}\} & \text{(Weight and bias parameters)} \end{array}$$

Figure 2: Graphical representation of a single hidden-layer neural network with one input x and one output $z^{(\mathtt{O})}$.

graph (DAG) describing the dependency between the different components of such a FNN having a single layer of hidden units $\boldsymbol{z} \in \mathbb{R}^{\mathtt{A}}$.

One feature of neural networks is that when its parameters $\boldsymbol{\theta}$ are known and the activation functions are differentiable up to the n-th order, we have access to both $g(x;\boldsymbol{\theta})$ as well as its derivatives $g^n(x;\boldsymbol{\theta})$. For instance, the first derivative of the network illustrated in Figure 2 is express using the chain rule of derivation as

$$z'^{(\mathtt{O})} = g^1(x;\boldsymbol{\theta}) = \frac{dz^{(\mathtt{O})}}{d\boldsymbol{a}}\frac{d\boldsymbol{a}}{d\boldsymbol{z}}\frac{d\boldsymbol{z}}{dx} = \left(\boldsymbol{w}^{(1)} \odot \phi'(\boldsymbol{z})\right)^{\intercal} \boldsymbol{w}^{(\mathtt{0})},$$

and because this relation only involves summation and product operations, we can directly use TAGI's forward uncertainty propagation method presented in §2 in order to compute the covariance $\text{cov}(Z'^{(\mathtt{O})}, \boldsymbol{X})$. The details regarding the analytical calculation of partial derivatives' expected values, variances, and covariances using TAGI are presented in Appendix B for the general case $z'^{(\mathtt{O})} = g^1(\boldsymbol{x};\boldsymbol{\theta})$.

In the context of an optimization problem, the goal is to identify the input $\boldsymbol{x}$ that either maximizes or minimizes $z^{(\mathtt{O})}$, at which location the first derivative of the function approximation is equal to zero, that is, $g^1(\boldsymbol{x};\boldsymbol{\theta}) = z'^{(\mathtt{O})} = 0$. Using the same inference procedure presented in §2, we can infer analytically the probability density function $f(\boldsymbol{x}|z'^{(\mathtt{O})} = 0) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_{\boldsymbol{X}|z'}, \boldsymbol{\Sigma}_{\boldsymbol{X}|z'})$. For that purpose, we first define the prior knowledge for the vector of covariates $\boldsymbol{x}$ so that $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_{\boldsymbol{X}}, \boldsymbol{\Sigma}_{\boldsymbol{X}})$. Then, analogously to the update step in Equation 3, the expected value $\boldsymbol{\mu}_{\boldsymbol{X}|z'}$ and variance $\boldsymbol{\Sigma}_{\boldsymbol{X}|z'}$ are computed following

$$\begin{array}{rcl} \boldsymbol{\mu}_{\boldsymbol{X}|z'} & = & \boldsymbol{\mu}_{\boldsymbol{X}} - \boldsymbol{\Sigma}^{\intercal}_{Z'^{(\mathtt{0})}\boldsymbol{X}} \left(\sigma^{(\mathtt{0})}_{Z'}\right)^{-2} \mu^{(\mathtt{0})}_{Z'} \\ \boldsymbol{\Sigma}_{\boldsymbol{X}|z'} & = & \boldsymbol{\Sigma}_{\boldsymbol{X}} - \boldsymbol{\Sigma}^{\intercal}_{Z'^{(\mathtt{0})}\boldsymbol{X}} \left(\sigma^{(\mathtt{0})}_{Z'}\right)^{-2} \boldsymbol{\Sigma}_{Z'^{(\mathtt{0})}\boldsymbol{X}}, \end{array} \tag{6}$$

where the expected value $\mathbb{E}[Z'^{(\mathtt{0})}] = \mu^{(\mathtt{0})}_{Z'}$, variance $\mathrm{var}[Z'^{(\mathtt{0})}] = (\sigma^{(\mathtt{0})}_{Z'})^2$, and covariance $\boldsymbol{\Sigma}_{Z'^{(\mathtt{0})}\boldsymbol{X}} = \mathrm{cov}(Z'^{(\mathtt{0})}, \boldsymbol{X})$ are obtained using the forward propagation of uncertainty defined for TAGI. In order to ensure that the inferred values for $\boldsymbol{x}$ correspond to either a minimum or a maximum, we need to rely on the sign of the first derivative to control the direction of the expected value update step. Moreover, in order to prevent a wrongly learn sign for $\boldsymbol{\Sigma}_{Z'^{(\mathtt{0})}\boldsymbol{X}}$ from interfering with the sign of the gradient $\mu^{(\mathtt{0})}_{Z'}$, the expected value in Equation 6 is thus reformulated as

$$\boldsymbol{\mu}_{\boldsymbol{X}|z'} = \boldsymbol{\mu}_{\boldsymbol{X}} + \alpha \cdot \mathrm{sign}\left(\mu^{(\mathtt{0})}_{Z'}\right) \left| \boldsymbol{\Sigma}^{\intercal}_{Z'^{(\mathtt{0})}\boldsymbol{X}} \left(\sigma^{(\mathtt{0})}_{Z'}\right)^{-2} \mu^{(\mathtt{0})}_{Z'} \right|, \tag{7}$$

where $\alpha = 1$ when seeking a maximum, and $\alpha = -1$ for a minimum.

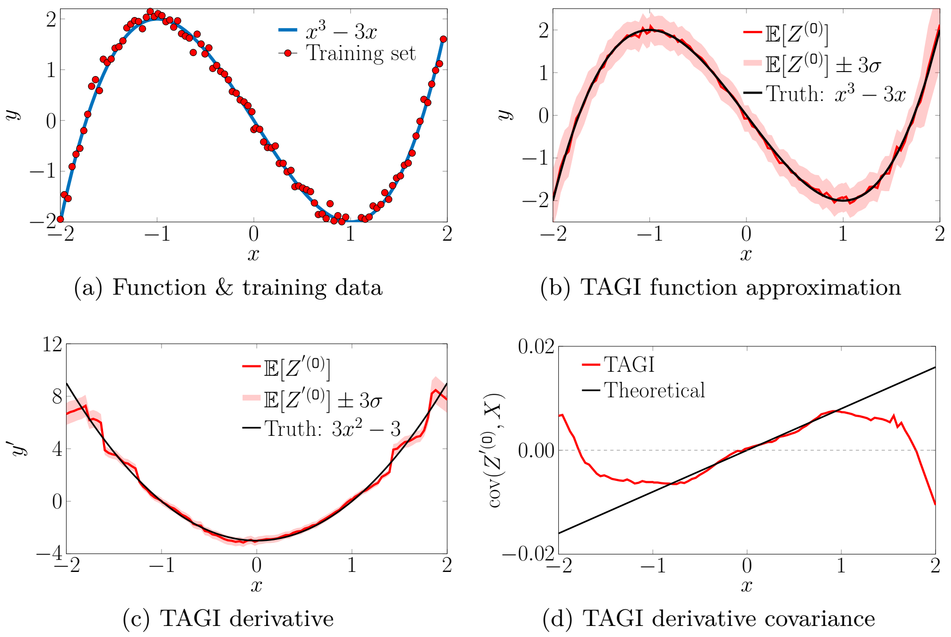
In order to seek the location where the derivative is equal to zero, we repeat the inference multiple times where the inferred values $\{\boldsymbol{\mu}^{(i)}_{\boldsymbol{X}|z'}, \boldsymbol{\Sigma}^{(i)}_{\boldsymbol{X}|z'}\}$ at iteration i are used as the prior's hyper-parameters at the next iteration $i+1$. The algorithm 1 presents an example of the implementation for the optimization of a function using TAGI's inference capacity.

Algorithm 1: Optimization of a function using TAGI

```
Define a neural network $g(\boldsymbol{x};\boldsymbol{\theta})$;
Initialize $\sigma_V$, the prior for $\boldsymbol{\theta}$ and for the covariates $\boldsymbol{X}$;
Given a data set $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}, \forall i = \{1, 2, \ldots, \mathtt{D}\}$;
for epoch $= 1 : \mathtt{E}$ do
    for $i = 1 : \mathtt{D}$ do
        Compute the prediction for a given input $\boldsymbol{x}_i$;
        $\{\mu_Y, \sigma_Y^2\} = g(\boldsymbol{x}_i; \boldsymbol{\theta})$;
        Update $\boldsymbol{\mu}_{\boldsymbol{\theta}|\mathcal{D}}$, $\boldsymbol{\Sigma}_{\boldsymbol{\theta}|\mathcal{D}}$ using TAGI;
        Compute the partial derivative of $g(\boldsymbol{x}; \boldsymbol{\theta})$ w.r.t. $\boldsymbol{x}$;
        Update $\boldsymbol{\mu}_{\boldsymbol{X}|z'}$, $\boldsymbol{\Sigma}_{\boldsymbol{X}|z'}$ using Equation 7;
```

We illustrate the inference-based optimization scheme on a 1D toy problem for $y = x^3 - 3x + v$ as depicted in Figure 3a, where the observation errors $V \sim \mathcal{N}(0, 0.1^2)$. The function approximation obtained using TAGI is presented in Figure 3b and its derivative in Figure 3c. We use this toy problem to illustrate how we use the derivative constraint α in order to reach either the local maximum at $x = -1$ or the local minimum at $x = +1$.

Table 2 presents the optimal location $\mu_{X|z'}$ found by TAGI depending on the starting location $\mu^{\mathtt{0}}_X$ and whether or not a derivative constraint α is employed. Note that for all



(a) Function & training data

(b) TAGI function approximation

(c) TAGI derivative

(d) TAGI derivative covariance

Figure 3: A 1D optimization problem where we show how from training data (a) we can approximate the underlying function (b) as well as its derivative (c). In (d) we compare the theoretical derivative's covariance and the one learned with TAGI.

cases, the initial input variance is set to $\sigma_X = 0.01$ and we performed 100 epochs for a total computational time of 18 seconds. The results show that when no derivative constraint α is employed, the optimal value reached correspond to either a maximum or a minimum, depending on the starting location $\mu_X^{\emptyset}$. More specifically, the inference will lead to the maximum or minimum associated within the region where the sign of the covariance is the same as for the starting location $\mu_X^{\emptyset}$, as depicted in Figure 3d. A positive derivative constrain

$\mu_X^{\emptyset}$	α	Update equation	$\mu_{X\|z'}$
0.25	N/A	Eq. 6	0.965
−0.25	N/A	Eq. 6	−0.992
0.25	+1	Eq. 7	−0.993
−0.25	+1	Eq. 7	−0.992
0.25	−1	Eq. 7	0.965
−0.25	−1	Eq. 7	0.965

Table 2: Comparison of the optimal values obtained $\mu_{X|z'}$ depending on the starting location $\mu_X^{\emptyset}$ and whether or not a derivative constraint α is employed.

α leads to the local minimum whether starting in a region having a positive or negative covariance. On the other hand, a negative constrain leads to the local maximum. Note that whether or not we use a derivative constrain α, TAGI will fail to infer the local maximum at $x = -1$ while starting at a value such as $x = 1.9$, because the sign of the covariance estimated using TAGI is incorrect so that the optimal location inferred will be pushed beyond the value $x = +2$. This example illustrates a limitation of TAGI's inference-based optimization scheme where, like for gradient-based approaches, the starting location $\mu_X^{\mathtt{0}}$ matters.

Although this optimization problem is trivial as it involves only one dimension, it showcases how the inference capability of TAGI can be leveraged in order to solve optimization tasks. The next section will build on that capacity in order to tackle continuous-actions reinforcement learning problems which involves optimization in higher-dimensional spaces.

5. Continuous-Actions RL through Inference

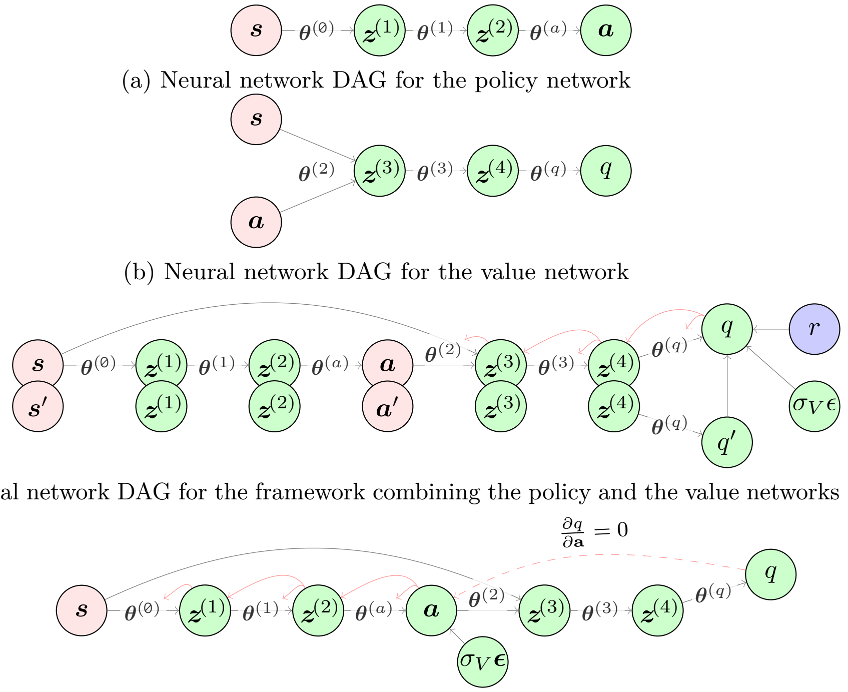
This section presents how to perform continuous-actions reinforcement learning (RL) by leveraging hidden-state inference. For both categorical and continuous actions RL frameworks, an agent's goal is to maximize the expected value conditional on an action a. For categorical actions, this can be achieved through the explicit evaluation of the expected value for each action and the selection of the optimal one. In the case of continuous actions, it is not possible nor desirable to evaluate the expected value associated with all possible actions; one thus face a continuous optimization problem. In deep-RL methods such as advanced actor critic (A2C) (Mnih et al., 2016) and proximal policy optimization (PPO) (Schulman et al., 2017), this optimization is tackled using gradient ascent approaches. Here, we rely instead on the method presented in §4 to identify the optimal action through inference.

For typical RL problems, the environment's state at a time t and $t+1$ are $\{\boldsymbol{s}, \boldsymbol{s'}\} \in \mathbb{R}^{\mathtt{N}^2}$, and the expected utility conditional on the actions $\boldsymbol{a} \in \mathbb{R}^{\mathtt{A}}$ and states $\boldsymbol{s}$ is defined by the action-value function $q(\boldsymbol{s}, \boldsymbol{a}) \in \mathbb{R}^1$. Figure 4a presents the directed acyclic graph describing the interconnectivity in a neural network capable of modelling a policy network, that is, the dependency between the actions $\boldsymbol{a}$ and the states $\boldsymbol{s}$. Figure 4b presents a similar graph for a value network modelling the dependency between the action-value function q, and the actions $\boldsymbol{a}$ and states $\boldsymbol{s}$. Figure 4c presents the combination of the value and policy networks from (a) and (b) in a single network that is analogous to the temporal-difference learning framework by Nguyen and Goulet (2021a), where $\{\boldsymbol{s}, \boldsymbol{a}\}$ are the states and action at a time t and $\{\boldsymbol{s'}, \boldsymbol{a'}\}$ the states and action at a time $t+1$. In this graph, the nodes that have been doubled represent the states $\boldsymbol{s}$ and $\boldsymbol{s'}$ which are both evaluated in a network sharing the same parameters in order to learn from the observation equation

$$q(\boldsymbol{s}, \boldsymbol{a}) = r(\boldsymbol{s}) + \gamma q(\boldsymbol{s'}, \boldsymbol{a'}) + \sigma_V \epsilon, \tag{8}$$

where ϵ is a realization from a standard-normal random variable, $r(\boldsymbol{s})$ is the reward function, and γ is the discount factor.

One particularity in the graph from Figure 4c is that the actions $\{\boldsymbol{a}, \boldsymbol{a'}\}$ are deterministic inputs (red nodes), as the specific actions at a time t are sampled from their current posterior predictive distribution. The red arrows outline the flow of information during the inference procedure for the components belonging to the value network. Note that the policy network cannot be updated directly because the flow of information in Figure 4c is broken by the



(a) Neural network DAG for the policy network

(b) Neural network DAG for the value network

(c) Neural network DAG for the framework combining the policy and the value networks

(d) Inference for the policy network

Figure 4: Graphical representation of a neural network structure for temporal-difference Q-policy learning for continuous actions.

knowledge of the actions. The component belonging to the policy network are thus updated separately as depicted in Figure 4d, where the prior for the actions $\boldsymbol{a}$ is computed from the policy network so that

$$\boldsymbol{a} = \boldsymbol{w}^{(a)}\boldsymbol{z}^{(2)} + \boldsymbol{b}^{(a)} + \sigma_V \boldsymbol{\epsilon}, \tag{9}$$

and where the inference for the actions uses the constrain on the derivative $\boldsymbol{a} : \frac{\partial q}{\partial \boldsymbol{a}} = 0$, as proposed in §4. Algorithm 2 details an example of implementation for the on-policy reinforcement learning in the context of TAGI.

We compare the performance of the TAGI-based on-policy TD reinforcement learning framework for continuous actions with the Proximal Policy Optimization (PPO) (Schulman et al., 2017), Advanced Actor Critic (A2C) (Mnih et al., 2016), Actor Critic using Kronecker-Factored Trust Region (ACKTR)(Wu et al., 2017), and Trust Region Policy Optimization (TRPO) (Schulman et al., 2015). We perform this comparison on the half-cheetah and inverted pendulum problems from the Mujoco environment (Todorov et al., 2012) implemented in OpenAI Gym (Brockman et al., 2016).

For the TAGI-based approach, the Q-value network uses a FNN with three hidden layers of 128 units. The policy network employs a FNN with two hidden layers of 128 units. The standard deviation σ_V in Equation 8 and 9 is initialized at 2 and is decayed each 1024 steps with a decaying factor of 0.9999. The minimal standard deviation is $\sigma_V^{\min} = 0.3$. These hyperparameters were set expirementally in order to offer a good performance for both environments. For the backpropagation-based approach, we use the same model architecture

Algorithm 2: Continuous-action reinforcement learning with TAGI

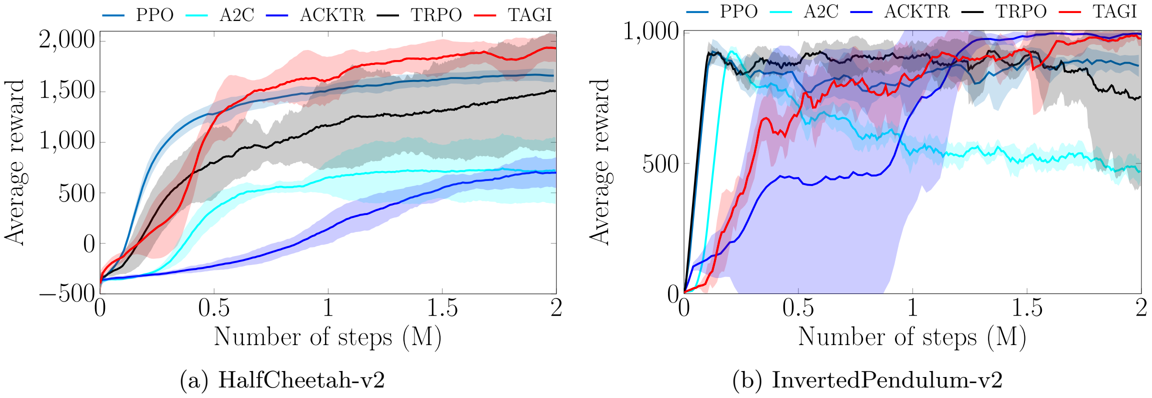
```
Define policy network $\mathcal{P}\left(\mathbf{s};\boldsymbol{\theta}^{\mathcal{P}}\right)$, value network $\mathcal{Q}\left(\mathbf{s},\mathbf{a};\boldsymbol{\theta}^{\mathcal{Q}}\right)$;
Initialize $\boldsymbol{\theta}^{\mathcal{P}}$, $\boldsymbol{\theta}^{\mathcal{Q}}$, $\sigma_V$, horizon $\mathtt{H}$, memory $\mathcal{R}$ to capacity $\mathtt{H}$
steps = 0;
for $episode = 1:\mathtt{E}$ do
    Reset environment $\mathbf{s}_1$;
    for $t = 1:\mathtt{T}$ do
        steps = steps + 1;
        $\left\{\boldsymbol{\mu}_t^{\mathbf{A}}, \boldsymbol{\Sigma}_t^{\mathbf{A}}\right\} = \mathcal{P}\left(\mathbf{s}_t;\boldsymbol{\theta}^{\mathcal{P}}\right)$ ;
        $\mathbf{a}_t : \mathbf{A}_t \sim \mathcal{N}(\boldsymbol{\mu}_t^{\mathbf{A}}, \boldsymbol{\Sigma}_t^{\mathbf{A}})$;
        $\mathbf{s}_{t+1}, r_t = \text{enviroment}(\mathbf{a}_t)$;
        Store $\{\mathbf{s}_t, \mathbf{a}_t, r_t\}$ in $\mathcal{R}$;
        if $steps\ mod\ \mathtt{H} == 0$ then
            $\left\{\boldsymbol{\mu}_{t+1}^{\mathbf{A}}, \boldsymbol{\Sigma}_{t+1}^{\mathbf{A}}\right\} = \mathcal{P}\left(\mathbf{s}_{t+1};\boldsymbol{\theta}^{\mathcal{P}}\right)$ ;
            $\mathbf{a}_{t+1} : \mathbf{A}_{t+1} \sim \mathcal{N}(\boldsymbol{\mu}_{t+1}^{\mathbf{A}}, \boldsymbol{\Sigma}_{t+1}^{\mathbf{A}})$;
            $\left\{\mu_{t+1}^{Q}, (\sigma_{t+1}^{Q})^2\right\} = \mathcal{Q}\left(\mathbf{s}_{t+1}, \mathbf{a}_{t+1};\boldsymbol{\theta}^{\mathcal{Q}}\right)$;
            Take $\mathtt{H}$ samples of $\{\mathbf{s}, \mathbf{a}, r\}$ from $\mathcal{R}$;
            $\mu_{\mathtt{H}}^{Y} = \mu_{t+1}^{Q}; \sigma_{\mathtt{H}}^{Y} = \sigma_{t+1}^{Q}$;
            for $j = \mathtt{H}-1:1$ do
                $\mu_j^Y = r_j + \gamma\mu_{j+1}^{Y}; (\sigma_j^Y)^2 = \gamma^2(\sigma_{j+1}^{Y})^2 + \sigma_V^2$;
            Update $\boldsymbol{\theta}^{\mathcal{Q}}$ using TAGI;
            Update $\boldsymbol{\theta}^{\mathcal{P}}$ using TAGI and Algorithm 1 with the constraint $\frac{\partial q}{\partial \mathbf{a}} = 0$;
            Initialize memory $\mathcal{R}$ to capacity $\mathtt{H}$;
```

for both the policy and value networks as well as the hyper-parameter values from OpenAI baselines (Dhariwal et al., 2017). During training, all methods except PPO uses a single epoch while PPO employs ten. Figure 5 shows the average reward over 100 episodes with respect to the number of steps for both environments.

Table 3 presents the average reward over the last 100 episodes for both environments. Although PPO initially learns faster, the final results show that TAGI outperforms PPO, A2C, and TRPO on both experiments, whereas ACKTR is on par with TAGI in one experiment. In addition, TAGI requires fewer hyper-parameters than other methods. Note that the goal of this experiment is to demonstrate how can inference be leveraged for solving

	Method				
Experiment	**TAGI**	A2C	ACKTR	PPO	TRPO
HalfCheetah-v2	**1934 ± 131**	722 ± 331	700 ± 151	1649 ± 48	1519 ± 478
InvertedPendulim-v2	983 ± 30	467 ± 34	**996 ± 6**	887 ± 42	756 ± 346

Table 3: Average reward over the last 100 episodes of five random runs for the half-cheetah and inverted pendulum experiments.



(a) HalfCheetah-v2

(b) InvertedPendulum-v2

Figure 5: Comparison of the average reward over 100 episodes of five random runs for two millions time steps. (TAGI: Tractable Approximate Gaussian Inference; PPO: Proximal Policy Optimization; A2C: Advanced Actor Critic; ACKTR: Actor Critic using Kronecker-Factored Trust Region; TRPO: Trust Region Policy Optimization.)

existing problems with a novel approach; The application of TAGI to RL problems is in its early days and it is foreseeable that if more time is invested in exploring new architectures and network configurations, the framework could further exceed the current performance.

6. Conclusion

TAGI provides a novel capacity to perform inference in neural networks. Its application to adversarial attacks, optimization, and continuous-action reinforcement learning showcases how these tasks, which previously relied on gradient-based optimization methods, can now be approached with analytically tractable inference. The applications presented in this paper are only a subset from the variety of problems that can take advantage of inference, either through the adaptation of existing architectures or through the development of new ones.

Acknowledgments

The first author was financially supported by research grants from Hydro-Quebec/IREQ, and the Natural Sciences and Engineering Research Council of Canada (NSERC). We would like to thank Magali Goulet and Prof. Mélina Mailhot for having reviewed the equations employed for computing the derivatives.

Appendix A. Model Architecture and Hyper-parameters

This appendix contains the specifications for each model architecture in the experiment section. D refers to a layer depth; W refers to a layer width; H refers to the layer height in case of convolutional or pooling layers; K refers to the kernel size; P refers to the convolutional kernel padding; S refers to the convolution stride; ϕ refers to the activation function type; ReLU refers to rectified linear unit;

Hyper-parameters	EADL1	PGDL2	CWL2
learning rate	0.01	-	0.01
binary search steps	9		9
number of steps	300	300	300
initial constrain	0.001		0.001
clip min	0	0	0
clip max	1	1	1
ϵ	3	-	-
attack step size	0.1	-	-

Table 4: Hyper-parameters for the gradient-based adversarial attacks applied to the MNIST and CIFAR10.

A.1 Adversarial Attack

For the classification tasks trained with backpropagation, we employ the same training setup for both data sets in which the learning rate is 0.003, the number of epochs is 50, the batch size is 64, and the optimizer is Adam.

A.1.1 MNIST

A.1.2 Cifar10

A.1.3 Additional Results

A.2 Optimization

Layer	$D \times W \times H$	$K \times K$	P	S	ϕ
Input	$1 \times 28 \times 28$	-	-	-	-
Convolutional	$32 \times 27 \times 27$	4×4	1	1	ReLU
Pooling	$32 \times 13 \times 13$	3×3	0	2	-
Convolutional	$64 \times 9 \times 9$	5×5	0	1	ReLU
Pooling	$64 \times 4 \times 4$	3×3	0	2	-
Fully connected	$150 \times 1 \times 1$	-	-	-	ReLU
Output	$11 \times 1 \times 1$	-	-	-	-

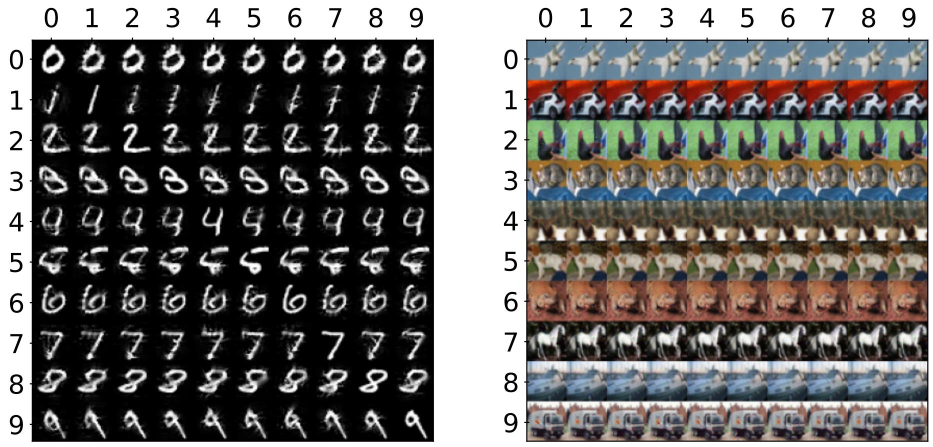
Table 5: Configuration details for the CNN applied to the MNIST adversarial attack.

Layer	$D \times W \times H$	$K \times K$	P	S	ϕ
Input	$3 \times 32 \times 32$	-	-	-	-
Convolutional	$32 \times 32 \times 32$	5×5	2	1	ReLU
Batch normalization	-	-	-	-	
Pooling	$32 \times 16 \times 16$	3×3	1	2	-
Convolutional	$32 \times 16 \times 16$	5×5	2	1	ReLU
Batch normalization	-	-	-	-	
Average pooling	$32 \times 8 \times 8$	3×3	1	2	-
Convolutional	$64 \times 8 \times 8$	5×5	2	1	ReLU
Batch normalization	-	-	-	-	
Average pooling	$64 \times 4 \times 4$	3×3	1	2	-
Fully connected	$64 \times 1 \times 1$	-	-	-	ReLU
Output	$11 \times 1 \times 1$	-	-	-	-

Table 6: Configuration details for the CNN applied to the Cifar10 adversarial attack.

A.3 Continuous-Action Reinforcement Learning

For the half-cheatah environment, the number of states N_s is 17 and the number of actions N_a is 6. For the inverted pendulum environment, the number of states N_s is 4 and the number of actions N_a is 1.



(a) MNIST-PGDL2

(b) CIFAR10-PGDL2

Figure 6: Examples of images subjected to adversarial attacks with different target labels $\tilde{y}$. Columns represent the different target labels $\tilde{y}$ and rows are the true label images.

Layer	$D \times W \times H$	$K \times K$	P	S	ϕ
Input	$1 \times 1 \times 1$	-	-	-	-
Fully connected	$64 \times 1 \times 1$	-	-	-	Tanh
Fully connected	$64 \times 1 \times 1$	-	-	-	ReLU
Output	$1 \times 1 \times 1$	-	-	-	-

Table 7: Configuration details for the feedforward neural network applied to 1D example.

Layer	$D \times W \times H$	$K \times K$	P	S	ϕ
Input	$N_s \times 1 \times 1$	-	-	-	-
Fully connected	$128 \times 1 \times 1$	-	-	-	ReLU
Fully connected	$128 \times 1 \times 1$	-	-	-	ReLU
Output	$N_a \times 1 \times 1$	-	-	-	Tanh

Table 8: Configuration details for the policy network. N_s is the number of states; N_a is the number of actions.

Appendix B. Partial Derivative in TAGI Neural Networks

B.1 TAGI Neural Networks

In a feedforward neural network, the hidden state at a given layer $l+1$ is defined as

$$Z_i^{(l+1)} = \sum_k W_{ik}^{(l)} \phi(Z_k^{(l)}) + B_i^{(l)},\ \forall i \in [1, \mathtt{A}^{(l+1)}],\ \forall k \in [1, \mathtt{A}^{(l)}], \forall l \in [1, \mathtt{L}] \tag{10}$$

where $\phi(.)$ is the activation function, $\{W, B\}$ are the unkown parameters of the neural network, that is, weight and bias, $\mathtt{A}^{(l)}$, is the number of hidden units in layer l and $\mathtt{L}$ is the number of hidden layers. We define the activation unit $A = \phi(Z)$. In the context of TAGI,

Layer	$D \times W \times H$	$K \times K$	P	S	ϕ
Input	$N_s \times 1 \times 1$	-	-	-	-
Fully connected	$128 \times 1 \times 1$	-	-	-	Tanh
Fully connected	$128 \times 1 \times 1$	-	-	-	ReLU
Fully connected	$128 \times 1 \times 1$	-	-	-	ReLU
Output	$1 \times 1 \times 1$	-	-	-	-

Table 9: Configuration details for the Q-value network. N_s is the number of states.

Method	#	Hyperparameter	Value
TAGI	1	Horizon	1024
	2	Initial standard deviation for the value function (σ_V)	2
	3	Decay factor (η)	0.9999
	4	Minimal standard deviation for the value function ($\sigma_V^{\min}$)	0.3
	5	Batch size	16
	6	Number of epochs	1
	7	Discount (γ)	0.99

Table 10: Hyper-parameters for half-cheetah and inverted pendulum problems.

Z, W, and B are assumed to be Gaussian random variables and

$$\begin{array}{rcl} W_{ik}^{(l)} & \perp\!\!\!\perp & W_{np}^{(m)} \perp\!\!\!\perp B_i^{(l)} \perp\!\!\!\perp B_n^{(m)},\ \forall m \in [1, \mathtt{L}], \forall n \in [1, \mathtt{A}^{(m+1)}], \forall p \in [1, \mathtt{A}^{(m)}] \\ Z_t^{(l-1)} & \perp\!\!\!\perp & Z_i^{(l+1)},\ \forall t \in [1, \mathtt{A}^{(l-1)}] \\ Z_k^{(l)} & \perp\!\!\!\perp & Z_q^{(l)},\ \forall q \in [1, \mathtt{A}^{(l)}] \text{ and } k \neq q \\ Z_k^{(l)} & \perp\!\!\!\perp & W_{ik}^{(l)} \perp\!\!\!\perp B_i^{(l)}. \end{array} \tag{11}$$

In addition, we apply the locally linearized activation function $\tilde{\phi}(.)$ to the hidden state in order to obtain the probability density function for the output of $\phi(.)$ so that

$$\phi(Z_k^{(l)}) = J_k^{(l)} \left(Z_k^{(l)} - \mathbb{E}\left[Z_k^{(l)} \right] \right) + \phi\left(\mathbb{E}\left[Z_k^{(l)} \right] \right), \tag{12}$$

where $J_k^{(l)} = \nabla_z \phi\left(\mathbb{E}\left[Z_k^{(l)} \right] \right)$.

B.2 Gaussian Multiplication Approximation (GMA)

Assuming $\boldsymbol{X} = [X_1 \ \dots \ X_4]^\intercal$ are Gaussian random variables, the GMA formulation had been defined by Goulet et al. (2021) as

$$\mathbb{E}[X_1 X_2] = \mu_1 \mu_2 + \operatorname{cov}(X_1, X_2), \tag{13}$$

$$\operatorname{cov}(X_3, X_1 X_2) = \operatorname{cov}(X_1, X_3)\mu_2 + \operatorname{cov}(X_2, X_3)\mu_1, \tag{14}$$

$$\begin{aligned} \operatorname{cov}(X_1 X_2, X_3 X_4) = {} & \operatorname{cov}(X_1, X_3)\operatorname{cov}(X_2, X_4) \\ & + \operatorname{cov}(X_1, X_4)\operatorname{cov}(X_2, X_3) \\ & + \operatorname{cov}(X_1, X_3)\mu_2\mu_4 + \operatorname{cov}(X_1, X_4)\mu_2\mu_3 \\ & + \operatorname{cov}(X_2, X_3)\mu_1\mu_4 + \operatorname{cov}(X_2, X_4)\mu_1\mu_3, \end{aligned} \tag{15}$$

$$\begin{aligned} \operatorname{var}(X_1 X_2) = {} & \sigma_1^2 \sigma_2^2 + \operatorname{cov}(X_1, X_2)^2 \\ & + 2\operatorname{cov}(X_1, X_2)\mu_1\mu_2 + \sigma_1^2 \mu_2^2 + \sigma_2^2 \mu_1^2. \end{aligned} \tag{16}$$

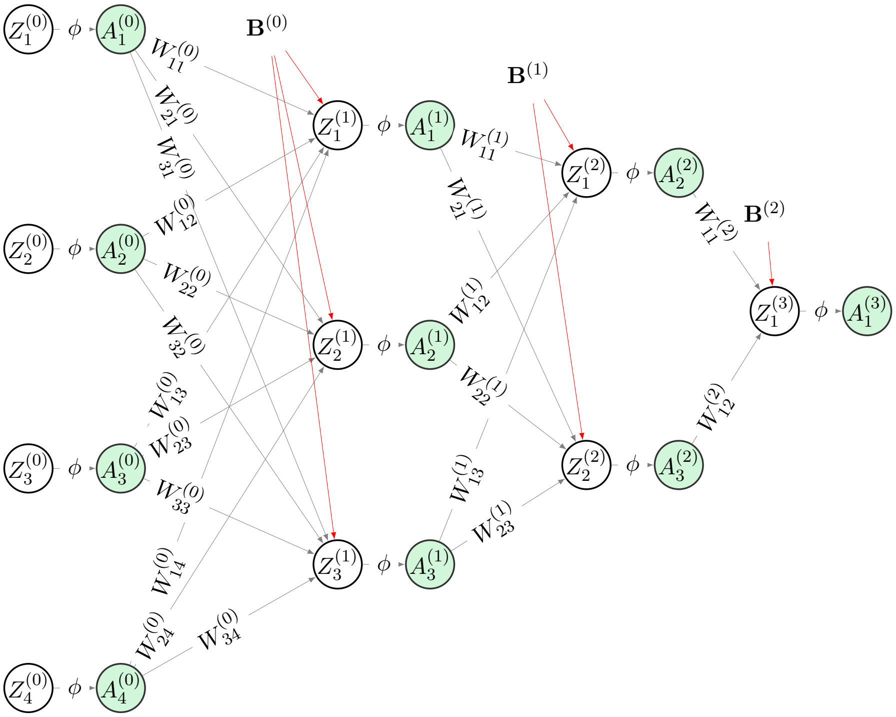


Figure 7: Ilustration of parameters $\boldsymbol{\theta} = \{\mathbf{W}, \mathbf{B}\}$, hidden states $\mathbf{Z}$, and activation units $\mathbf{A}$ associated with a four-layer feedforward neural network.

B.3 Partial Derivative Formulations for A Simple Feedforward Neural Network

This section presents the first-order partial derivative formulations for a feedforward neural network (FNN) of four layers in the context of TAGI. Note that building on this work the formulation for higher order derivatives can be found in Goulet (2021). Figure 7 presents the details of the interconnectivity of the variables associated with a four-layer FNN, Figure 8 describes the partial derivative diagram for the four-layer FNN presented in Figure 7, and Figure 9 shows the partial derivative diagram associated with the parameters and hidden states. The partial derivative diagram allow computing the partial derivative of either a hidden state or an activation unit at any layers with respect to either the hidden state or activation unit from the previous layers. For example, the partial derivative of the first activation unit of layer three, that is, $A_1^{(3)}$ with respect to the first hidden state of layer one, that is, $Z_1^{(1)}$ is the sum of the product of the partial derivatives of two branches relating to this partial derivative, which are identified using the partial derivative diagram in Figure 9. Figure 10 illustrates the computations of this partial derivative.

B.3.1 Partial Derivative $\frac{\partial A_1^{(3)}}{\partial Z_1^{(2)}}$

This section presents the calculations of the partial derivative of $A_1^{(3)}$ with respect to $Z_1^{(2)}$. Figure 11 shows the branch from the partial derivative diagram (Figure 9), that corresponds to this partial derivative. This partial derivative is defined as

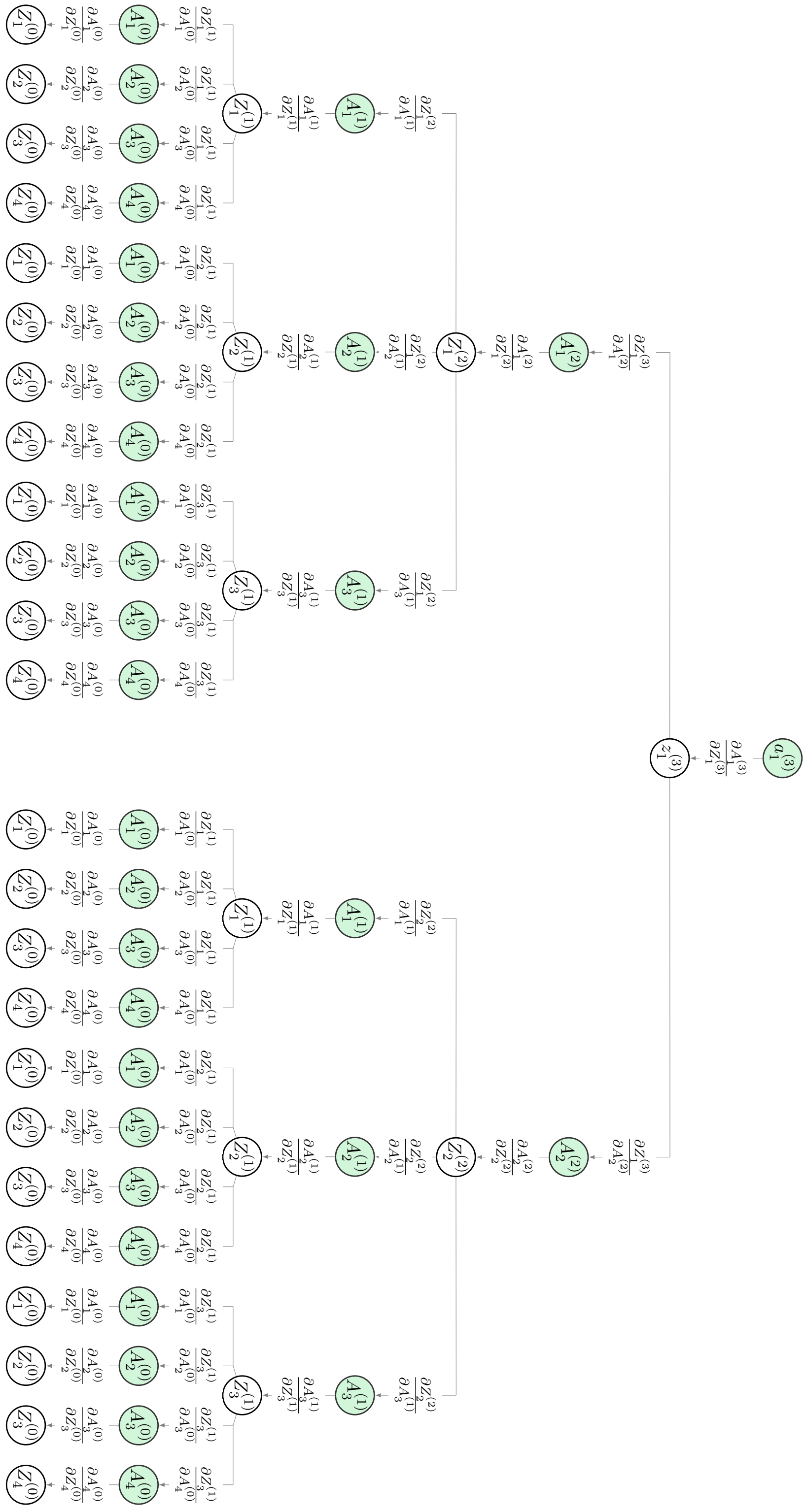

$Z_1^{(0)}$ $\frac{\partial A_1^{(0)}}{\partial Z_1^{(0)}}$ $A_1^{(0)}$ $\frac{\partial Z_1^{(1)}}{\partial A_1^{(0)}}$
$Z_2^{(0)}$ $\frac{\partial A_2^{(0)}}{\partial Z_2^{(0)}}$ $A_2^{(0)}$ $\frac{\partial Z_1^{(1)}}{\partial A_2^{(0)}}$
$Z_1^{(1)}$ $\frac{\partial A_1^{(1)}}{\partial Z_1^{(1)}}$ $A_1^{(1)}$ $\frac{\partial Z_1^{(2)}}{\partial A_1^{(1)}}$
$Z_3^{(0)}$ $\frac{\partial A_3^{(0)}}{\partial Z_3^{(0)}}$ $A_3^{(0)}$ $\frac{\partial Z_1^{(1)}}{\partial A_3^{(0)}}$
$Z_4^{(0)}$ $\frac{\partial A_4^{(0)}}{\partial Z_4^{(0)}}$ $A_4^{(0)}$ $\frac{\partial Z_1^{(1)}}{\partial A_4^{(0)}}$
$Z_1^{(0)}$ $\frac{\partial A_1^{(0)}}{\partial Z_1^{(0)}}$ $A_1^{(0)}$ $\frac{\partial Z_2^{(1)}}{\partial A_1^{(0)}}$
$Z_2^{(0)}$ $\frac{\partial A_2^{(0)}}{\partial Z_2^{(0)}}$ $A_2^{(0)}$ $\frac{\partial Z_2^{(1)}}{\partial A_2^{(0)}}$
$Z_2^{(1)}$ $\frac{\partial A_2^{(1)}}{\partial Z_2^{(1)}}$ $A_2^{(1)}$ $\frac{\partial Z_1^{(2)}}{\partial A_2^{(1)}}$ $Z_1^{(2)}$ $\frac{\partial A_1^{(2)}}{\partial Z_1^{(2)}}$ $A_1^{(2)}$ $\frac{\partial Z_1^{(3)}}{\partial A_1^{(2)}}$
$Z_3^{(0)}$ $\frac{\partial A_3^{(0)}}{\partial Z_3^{(0)}}$ $A_3^{(0)}$ $\frac{\partial Z_2^{(1)}}{\partial A_3^{(0)}}$
$Z_4^{(0)}$ $\frac{\partial A_4^{(0)}}{\partial Z_4^{(0)}}$ $A_4^{(0)}$ $\frac{\partial Z_2^{(1)}}{\partial A_4^{(0)}}$
$Z_1^{(0)}$ $\frac{\partial A_1^{(0)}}{\partial Z_1^{(0)}}$ $A_1^{(0)}$ $\frac{\partial Z_3^{(1)}}{\partial A_1^{(0)}}$
$Z_2^{(0)}$ $\frac{\partial A_2^{(0)}}{\partial Z_2^{(0)}}$ $A_2^{(0)}$ $\frac{\partial Z_3^{(1)}}{\partial A_2^{(0)}}$
$Z_3^{(1)}$ $\frac{\partial A_3^{(1)}}{\partial Z_3^{(1)}}$ $A_3^{(1)}$ $\frac{\partial Z_1^{(2)}}{\partial A_3^{(1)}}$
$Z_3^{(0)}$ $\frac{\partial A_3^{(0)}}{\partial Z_3^{(0)}}$ $A_3^{(0)}$ $\frac{\partial Z_3^{(1)}}{\partial A_3^{(0)}}$
$Z_4^{(0)}$ $\frac{\partial A_4^{(0)}}{\partial Z_4^{(0)}}$ $A_4^{(0)}$ $\frac{\partial Z_3^{(1)}}{\partial A_4^{(0)}}$
$z_1^{(3)}$ $\frac{\partial A_1^{(3)}}{\partial Z_1^{(3)}}$ $a_1^{(3)}$
$Z_1^{(0)}$ $\frac{\partial A_1^{(0)}}{\partial Z_1^{(0)}}$ $A_1^{(0)}$ $\frac{\partial Z_1^{(1)}}{\partial A_1^{(0)}}$
$Z_2^{(0)}$ $\frac{\partial A_2^{(0)}}{\partial Z_2^{(0)}}$ $A_2^{(0)}$ $\frac{\partial Z_1^{(1)}}{\partial A_2^{(0)}}$
$Z_1^{(1)}$ $\frac{\partial A_1^{(1)}}{\partial Z_1^{(1)}}$ $A_1^{(1)}$ $\frac{\partial Z_2^{(2)}}{\partial A_1^{(1)}}$
$Z_3^{(0)}$ $\frac{\partial A_3^{(0)}}{\partial Z_3^{(0)}}$ $A_3^{(0)}$ $\frac{\partial Z_1^{(1)}}{\partial A_3^{(0)}}$
$Z_4^{(0)}$ $\frac{\partial A_4^{(0)}}{\partial Z_4^{(0)}}$ $A_4^{(0)}$ $\frac{\partial Z_1^{(1)}}{\partial A_4^{(0)}}$
$Z_1^{(0)}$ $\frac{\partial A_1^{(0)}}{\partial Z_1^{(0)}}$ $A_1^{(0)}$ $\frac{\partial Z_2^{(1)}}{\partial A_1^{(0)}}$
$Z_2^{(0)}$ $\frac{\partial A_2^{(0)}}{\partial Z_2^{(0)}}$ $A_2^{(0)}$ $\frac{\partial Z_2^{(1)}}{\partial A_2^{(0)}}$
$Z_2^{(1)}$ $\frac{\partial A_2^{(1)}}{\partial Z_2^{(1)}}$ $A_2^{(1)}$ $\frac{\partial Z_2^{(2)}}{\partial A_2^{(1)}}$ $Z_2^{(2)}$ $\frac{\partial A_2^{(2)}}{\partial Z_2^{(2)}}$ $A_2^{(2)}$ $\frac{\partial Z_1^{(3)}}{\partial A_2^{(2)}}$
$Z_3^{(0)}$ $\frac{\partial A_3^{(0)}}{\partial Z_3^{(0)}}$ $A_3^{(0)}$ $\frac{\partial Z_2^{(1)}}{\partial A_3^{(0)}}$
$Z_4^{(0)}$ $\frac{\partial A_4^{(0)}}{\partial Z_4^{(0)}}$ $A_4^{(0)}$ $\frac{\partial Z_2^{(1)}}{\partial A_4^{(0)}}$
$Z_1^{(0)}$ $\frac{\partial A_1^{(0)}}{\partial Z_1^{(0)}}$ $A_1^{(0)}$ $\frac{\partial Z_3^{(1)}}{\partial A_1^{(0)}}$
$Z_2^{(0)}$ $\frac{\partial A_2^{(0)}}{\partial Z_2^{(0)}}$ $A_2^{(0)}$ $\frac{\partial Z_3^{(1)}}{\partial A_2^{(0)}}$
$Z_3^{(1)}$ $\frac{\partial A_3^{(1)}}{\partial Z_3^{(1)}}$ $A_3^{(1)}$ $\frac{\partial Z_2^{(2)}}{\partial A_3^{(1)}}$
$Z_3^{(0)}$ $\frac{\partial A_3^{(0)}}{\partial Z_3^{(0)}}$ $A_3^{(0)}$ $\frac{\partial Z_3^{(1)}}{\partial A_3^{(0)}}$
$Z_4^{(0)}$ $\frac{\partial A_4^{(0)}}{\partial Z_4^{(0)}}$ $A_4^{(0)}$ $\frac{\partial Z_3^{(1)}}{\partial A_4^{(0)}}$

Figure 8: Illustration of partial derivative diagram for a four-layer feedforward neural network.

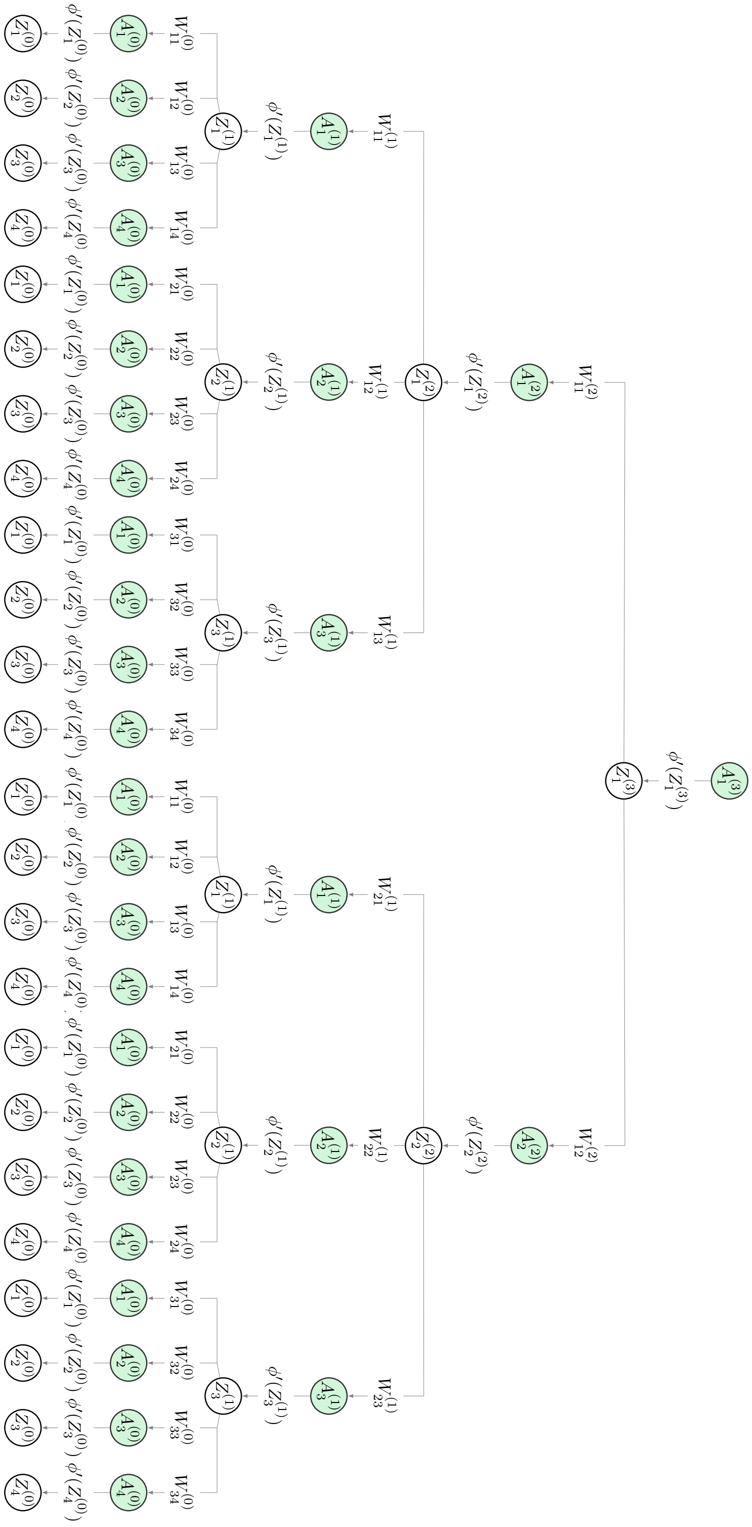


Figure 9: Illustration of partial derivative diagram associated with the parameters and hidden states for a four-layer feedforward neural network.

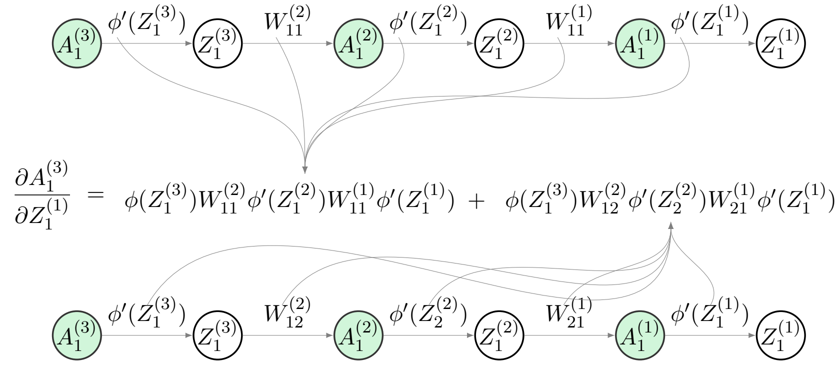


Figure 10: Illustration of the partial derivative of $A_1^{(3)}$ with respect to $Z_1^{(1)}$.

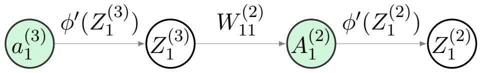


Figure 11: Illustration of a branch of the partial derivative of $A_1^{(3)}$ with respect to $Z_1^{(2)}$.

$$\frac{\partial A_1^{(3)}}{\partial Z_1^{(2)}} = \phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)}). \tag{17}$$

In the context of TAGI, the weights W and hidden states Z are Gaussian random variables, therefore, $\frac{\partial A_1^{(3)}}{\partial Z_1^{(2)}}$ is also approximated by a Gaussian PDF. The expected value is computed using Equation 13 and 11,

$$\begin{aligned}\mathbb{E}\left[\frac{\partial A_1^{(3)}}{\partial Z_1^{(2)}}\right] &= \mathbb{E}\left[\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})\right]\\ &= \mathbb{E}\left[\phi'(Z_1^{(3)})\right]\mathbb{E}\left[W_{11}^{(2)}\phi'(Z_1^{(2)})\right] + \mathrm{cov}\left(\phi'(Z_1^{(3)}), W_{11}^{(2)}\phi'(Z_1^{(2)})\right),\end{aligned} \tag{18}$$

where

$$\mathbb{E}\left[W_{11}^{(2)}\phi'(Z_1^{(2)})\right] = \mathbb{E}\left[W_{11}^{(2)}\right]\mathbb{E}\left[\phi'(Z_1^{(2)})\right] + \underbrace{\cancelto{0}{\mathrm{cov}\left(W_{11}^{(2)}, \phi'(Z_1^{(2)})\right).}}_{\text{Eq. 11}} \tag{19}$$

$$\begin{aligned}\mathrm{cov}\left(\phi'(Z_1^{(3)}),\, W_{11}^{(2)}\phi'(Z_1^{(2)})\right) &= \mathrm{cov}\left(\phi'(Z_1^{(3)}),\, W_{11}^{(2)}\right)\mathbb{E}\left[\phi'(Z_1^{(2)})\right]\\ &+ \mathrm{cov}\left(\phi'(Z_1^{(3)}),\, \phi'(Z_1^{(2)})\right)\mathbb{E}\left[W_{11}^{(2)}\right].\end{aligned} \tag{20}$$

Note that the computations for the covariance $\mathrm{cov}\left(\phi'(Z_1^{(3)}),\, W_{11}^{(2)}\right)$ and $\mathrm{cov}\left(\phi'(Z_1^{(3)}),\, \phi'(Z_1^{(2)})\right)$ depend on the type of the activation function $\phi(.)$ being used for this layer (see §B.5). The variance is computed using Equation 16, 11, 19, and 18,

$$\begin{aligned}
\operatorname{var}\left(\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})\right) &= \operatorname{var}\left(\phi'(Z_1^{(3)})\right)\operatorname{var}\left(W_{11}^{(2)}\phi'(Z_1^{(2)})\right)\\
&+ \operatorname{cov}\left(\phi'(Z_1^{(3)}),W_{11}^{(2)}\phi'(Z_1^{(2)})\right)^2\\
&+ 2\operatorname{cov}\left(\phi'(Z_1^{(3)}),W_{11}^{(2)}\phi'(Z_1^{(2)})\right)\\
&\times \mathbb{E}\left[W_{11}^{(2)}\right]\mathbb{E}\left[W_{11}^{(2)}\phi'(Z_1^{(2)})\right]\\
&+ \operatorname{var}\left(\phi'(Z_1^{(3)})\right)\mathbb{E}\left[W_{11}^{(2)}\phi'(Z_1^{(2)})\right]^2\\
&+ \operatorname{var}\left(W_{11}^{(2)}\phi'(Z_1^{(2)})\right)\mathbb{E}\left[\phi'(Z_1^{(3)})\right]^2,
\end{aligned} \tag{21}$$

where

$$\begin{aligned}
\operatorname{var}\left(W_{11}^{(2)}\phi'(Z_1^{(2)})\right) &= \operatorname{var}\left(W_{11}^{(2)}\right)\operatorname{var}\left(\phi'(Z_1^{(2)})\right) + \underbrace{\cancelto{0}{\operatorname{cov}\left(W_{11}^{(2)},\phi'(Z_1^{(2)})\right)^2}}_{\text{Eq. 11}}\\
&+ 2\cancelto{0}{\operatorname{cov}\left(W_{11}^{(2)},\phi'(Z_1^{(2)})\right)}\mathbb{E}\left[W_{11}^{(2)}\right]\mathbb{E}\left[\phi'(Z_1^{(2)})\right]\\
&+ \operatorname{var}\left(W_{11}^{(2)}\right)\mathbb{E}\left[\phi'(Z_1^{(2)})\right]^2 + \operatorname{var}\left(\phi'(Z_1^{(2)})\right)\mathbb{E}\left[W_{11}^{(2)}\right]^2\\
&= \operatorname{var}\left(W_{11}^{(2)}\right)\operatorname{var}\left(\phi'(Z_1^{(2)})\right) + \operatorname{var}\left(W_{11}^{(2)}\right)\mathbb{E}\left[\phi'(Z_1^{(2)})\right]^2\\
&+ \operatorname{var}\left(\phi'(Z_1^{(2)})\right)\mathbb{E}\left[W_{11}^{(2)}\right]^2.
\end{aligned} \tag{22}$$

B.3.2 Partial Derivative $\frac{\partial A_1^{(3)}}{\partial Z_1^{(1)}}$

This section presents the calculations of the partial derivative of $A_1^{(3)}$ with respect to $Z_1^{(1)}$. According to the partial derivative diagram, there are two branches relating to this partial derivative. The partial derivative is a sum of the product of partial derivatives on these two branches. The rest of this section only presents the computations for one of these two branches (Figure 12). This partial derivative is defined following

$$\frac{\partial A_1^{(3)}}{\partial Z_1^{(1)}} = \phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})W_{11}^{(1)}\phi'(Z_1^{(1)}). \tag{23}$$

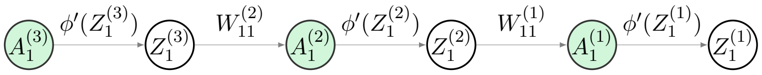


Figure 12: Illustration of a branch of partial derivative of $A_1^{(3)}$ with respect to $Z_1^{(1)}$.

The expected value is computed using Equation 13, 14, 11 and 18,

$$
\begin{aligned}
\mathbb{E}\left[\frac{\partial A_1^{(3)}}{\partial Z_1^{(1)}}\right] &= \mathbb{E}\left[\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})W_{11}^{(1)}\phi'(Z_1^{(1)})\right] \\
&= \mathbb{E}\left[\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})\right]\,\mathbb{E}\left[W_{11}^{(1)}\phi'(Z_1^{(1)})\right] \\
&+ \operatorname{cov}\left(\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)}),\, W_{11}^{(1)}\phi'(Z_1^{(1)})\right),
\end{aligned}
\tag{24}
$$

where

$$
\mathbb{E}\left[W_{11}^{(1)}\phi'(Z_1^{(1)})\right] = \mathbb{E}\left[W_{11}^{(1)}\right]\,\mathbb{E}\left[\phi'(Z_1^{(1)})\right] + \cancelto{0}{\operatorname{cov}\left(W_{11}^{(1)}, \phi'(Z_1^{(1)})\right)}, \tag{25}
$$

$$
\begin{aligned}
&\operatorname{cov}\left(\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)}),\, W_{11}^{(1)}\phi'(Z_1^{(1)})\right) \\
&\qquad= \cancelto{0}{\operatorname{cov}\left(\phi'(Z_1^{(3)}), W_{11}^{(1)}\phi'(Z_1^{(1)})\right)}\,\mathbb{E}\left[W_{11}^{(2)}\phi'(Z_1^{(2)})\right] \\
&\qquad+ \operatorname{cov}\left(W_{11}^{(2)}\phi'(Z_1^{(2)}),\, W_{11}^{(1)}\phi'(Z_1^{(1)})\right)\,\mathbb{E}\left[\phi'(Z_1^{(3)})\right] \\
&\qquad= \Bigg\{\operatorname{cov}\left(\phi'(Z_1^{(2)}),\, W_{11}^{(1)}\phi'(Z_1^{(1)})\right)\,\mathbb{E}\left[W_{11}^{(2)}\right] \\
&\qquad+ \cancelto{0}{\operatorname{cov}\left(W_{11}^{(2)}, W_{11}^{(1)}\phi'(Z_1^{(1)})\right)}\,\mathbb{E}\left[\phi'(Z_1^{(2)})\right]\Bigg\}\,\mathbb{E}\left[\phi'(Z_1^{(3)})\right] \\
&\qquad= \operatorname{cov}\left(\phi'(Z_1^{(2)}),\, W_{11}^{(1)}\phi'(Z_1^{(1)})\right)\,\mathbb{E}\left[W_{11}^{(2)}\right]\,\mathbb{E}\left[\phi'(Z_1^{(3)})\right], \\
&\operatorname{cov}\left(\phi'(Z_1^{(2)}),\, W_{11}^{(1)}\phi'(Z_1^{(1)})\right) \\
&\qquad= \operatorname{cov}\left(\phi'(Z_1^{(2)}),\, W_{11}^{(1)}\right)\,\mathbb{E}\left[\phi'(Z_1^{(1)})\right] \\
&\qquad+ \operatorname{cov}\left(\phi'(Z_1^{(2)}),\, \phi'(Z_1^{(1)})\right)\,\mathbb{E}\left[W_{11}^{(1)}\right].
\end{aligned}
\tag{26}
$$

The variance is computed using Equation 16, 18, 21, 25, and 26,

$$
\begin{aligned}
\operatorname{var}\left(\frac{\partial A_1^{(3)}}{\partial Z_1^{(1)}}\right) &= \operatorname{var}\left(\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})W_{11}^{(1)}\phi'(Z_1^{(1)})\right)\\
&= \operatorname{var}\left(\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})\right)\ \operatorname{var}\left(W_{11}^{(1)}\phi'(Z_1^{(1)})\right)\\
&+ \operatorname{cov}\left(\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)}),\ W_{11}^{(1)}\phi'(Z_1^{(1)})\right)^2\\
&+ 2\operatorname{cov}\left(\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)}),\ W_{11}^{(1)}\phi'(Z_1^{(1)})\right)\\
&\quad \mathbb{E}\left[\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})\right]\ \mathbb{E}\left[W_{11}^{(1)}\phi'(Z_1^{(1)})\right]\\
&+ \operatorname{var}\left(\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})\right)\ \mathbb{E}\left[W_{11}^{(1)}\phi'(Z_1^{(1)})\right]^2\\
&+ \operatorname{var}\left(W_{11}^{(1)}\phi'(Z_1^{(1)})\right)\ \mathbb{E}\left[\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})\right]^2.
\end{aligned}
\tag{27}
$$

The same above steps are repeated for the second branch in order to complete the calculation of the partial derivative of $A_1^{(3)}$ with respect to $Z_1^{(1)}$.

B.3.3 Partial Derivative $\frac{\partial A_1^{(3)}}{\partial Z_1^{(0)}}$

This section presents the calculations of the partial derivative of $A_1^{(3)}$ with respect to $Z_1^{(0)}$. From the partial derivative diagram (Figure 9), we identify six branches relating to this partial derivative. Therefore, the partial derivative is equal to the sum of the product of partial derivatives on these six branches. Figure 13 shows the details for one of six branches. The partial derivative relating to this branch is defined following

$$
\frac{\partial A_1^{(3)}}{\partial Z_1^{(0)}} = \phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})W_{11}^{(1)}\phi'(Z_1^{(1)})W_{11}^{(0)}\phi'(Z_1^{(0)}). \tag{28}
$$

The expected value is computed using Equation 13, 11 and 24,

$$
\begin{aligned}
\mathbb{E}\left[\frac{\partial A_1^{(3)}}{\partial Z_1^{(0)}}\right] &= \mathbb{E}\left[\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})W_{11}^{(1)}\phi'(Z_1^{(1)})W_{11}^{(0)}\phi'(Z_1^{(0)})\right]\\
&= \mathbb{E}\left[\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})W_{11}^{(1)}\phi'(Z_1^{(1)})\right]\ \mathbb{E}\left[W_{11}^{(0)}\phi'(Z_1^{(0)})\right]\\
&+ \operatorname{cov}\left(\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})W_{11}^{(1)}\phi'(Z_1^{(1)}),\ W_{11}^{(0)}\phi'(Z_1^{(0)})\right),
\end{aligned}
\tag{29}
$$

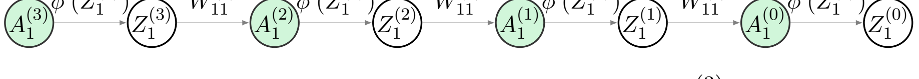


Figure 13: Illustration of a branch of the partial derivative of $A_1^{(3)}$ with respect to $Z_1^{(0)}$.

where

$$\mathbb{E}\left[W_{11}^{(0)}\phi'(Z_1^{(0)})\right] = \mathbb{E}\left[W_{11}^{(0)}\right]\,\mathbb{E}\left[\phi'(Z_0^{(0)})\right] + \cancelto{0}{\text{cov}\left(W_{11}^{(1)},\, \phi'(Z_1^{(0)})\right)}, \tag{30}$$

$$\begin{aligned}
&\text{cov}\left(\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})W_{11}^{(1)}\phi'(Z_1^{(1)}),\ W_{11}^{(0)}\phi'(Z_1^{(0)})\right) \\
&\qquad = \cancelto{0}{\text{cov}\left(\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)}),\ W_{11}^{(0)}\phi'(Z_1^{(0)})\right) \times \mathbb{E}\left[W_{11}^{(0)}\phi'(Z_1^{(0)})\right]} \\
&\qquad + \text{cov}\left(W_{11}^{(1)}\phi'(Z_1^{(1)}),\ W_{11}^{(0)}\phi'(Z_1^{(0)})\right) \\
&\qquad \times \mathbb{E}\left[\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})\right] \\
&\qquad = \text{cov}\left(\phi'(Z_1^{(1)}),\ W_{11}^{(0)}\phi'(Z_1^{(0)})\right) \\
&\qquad \times \mathbb{E}\left[W_{11}^{(1)}\right]\,\underbrace{\mathbb{E}\left[\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})\right]}_{\text{Eq. (18)}} \\
&\text{cov}\left(\phi'(Z_1^{(1)}),\, W_{11}^{(0)}\phi'(Z_1^{(0)})\right) \\
&\qquad = \text{cov}\left(\phi'(Z_1^{(1)}),\, W_{11}^{(0)}\right)\,\mathbb{E}\left[\phi'(Z_1^{(0)})\right] \\
&\qquad + \text{cov}\left(\phi'(Z_1^{(1)}),\, \phi'(Z_1^{(0)})\right)\,\mathbb{E}\left[W_{11}^{(0)}\right].
\end{aligned} \tag{31}$$

The variance is computed using Equation 16, 24, 27, 30 and 31,

$$\begin{aligned}
\text{var}\left(\frac{\partial A_1^{(3)}}{\partial Z_1^{(0)}}\right) &= \text{var}\left(\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})W_{11}^{(1)}\phi'(Z_1^{(1)})W_{11}^{(0)}\phi'(Z_1^{(0)})\right) \\
&= \text{var}\left(\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})W_{11}^{(1)}\phi'(Z_1^{(1)})\right)\,\text{var}\left(W_{11}^{(0)}\phi'(Z_1^{(0)})\right) \\
&+ \text{cov}\left(\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})W_{11}^{(1)}\phi'(Z_1^{(1)}),\ W_{11}^{(0)}\phi'(Z_1^{(0)})\right)^2 \\
&+ 2\text{cov}\left(\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})W_{11}^{(1)}\phi'(Z_1^{(1)}),\ W_{11}^{(0)}\phi'(Z_1^{(0)})\right) \\
&\times \mathbb{E}\left[\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})W_{11}^{(1)}\phi'(Z_1^{(1)})\right]\,\mathbb{E}\left[W_{11}^{(0)}\phi'(Z_1^{(0)})\right] \\
&+ \text{var}\left(\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})W_{11}^{(1)}\phi'(Z_1^{(1)})\right)\,\mathbb{E}\left[W_{11}^{(0)}\phi'(Z_1^{(0)})\right]^2 \\
&+ \text{var}\left(W_{11}^{(0)}\phi'(Z_1^{(0)})\right)\,\mathbb{E}\left[\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})W_{11}^{(1)}\phi'(Z_1^{(1)})\right]^2.
\end{aligned} \tag{32}$$

The same calculations are repeated for the five remaining branches in order to obtain the partial derivative of $A_1^{(3)}$ with respect to $Z_1^{(0)}$.

B.3.4 Covariance between $\frac{\partial A_1^{(3)}}{\partial Z_1^{(0)}}$ and $Z_1^{(0)}$

This section presents the calculations of the covariance for the partial derivative $\frac{\partial A_1^{(3)}}{\partial Z_1^{(0)}}$ and $Z_1^{(0)}$. The following calculations correspond to the branch illustrated in Figure 13,

$$\begin{aligned}
&\operatorname{cov}\left(\frac{\partial A_1^{(3)}}{\partial Z_1^{(0)}},\, Z_1^{(0)}\right)\\
&\quad= \operatorname{cov}\left(\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})W_{11}^{(1)}\phi'(Z_1^{(1)})W_{11}^{(0)}\phi'(Z_1^{(0)}),\ Z_1^{(0)}\right)\\
&\quad= \operatorname{cov}\left(\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})W_{11}^{(1)}\phi'(Z_1^{(1)}),\ Z_1^{(0)}\right)\underbrace{\mathbb{E}\left[W_{11}^{(0)}\phi'(Z_1^{(0)})\right]}_{\text{Eq. (30)}}\\
&\quad+ \operatorname{cov}\left(W_{11}^{(0)}\phi'(Z_1^{(0)}),\ Z_1^{(0)}\right)\underbrace{\mathbb{E}\left[\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})W_{11}^{(1)}\phi'(Z_1^{(1)})\right]}_{\text{Eq. (24)}},
\end{aligned} \tag{33}$$

where

$$\begin{aligned}
&\operatorname{cov}\left(\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})W_{11}^{(1)}\phi'(Z_1^{(1)}),\ Z_1^{(0)}\right)\\
&\quad= \cancelto{0}{\operatorname{cov}\left(\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)}),\ Z_1^{(0)}\right)}\mathbb{E}\left[W_{11}^{(1)}\phi'(Z_1^{(1)})\right]\\
&\quad+ \operatorname{cov}\left(W_{11}^{(1)}\phi'(Z_1^{(1)}),\, Z_1^{(0)}\right)\mathbb{E}\left[\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})\right]\\
&\quad= \left\{\cancelto{0}{\operatorname{cov}\left(W_{11}^{(1)},\, Z_1^{(0)}\right)}\mathbb{E}\left[\phi'(Z_1^{(1)})\right]\right.\\
&\quad+ \left.\operatorname{cov}\left(\phi'(Z_1^{(1)}),\, Z_1^{(0)}\right)\mathbb{E}\left[W_{11}^{(1)}\right]\right\}\mathbb{E}\left[\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})\right]\\
&\quad= \operatorname{cov}\left(\phi'(Z_1^{(1)}),\, Z_1^{(0)}\right)\mathbb{E}\left[W_{11}^{(1)}\right]\underbrace{\mathbb{E}\left[\phi'(Z_1^{(3)})W_{11}^{(2)}\phi'(Z_1^{(2)})\right]}_{\text{Eq. (18)}}
\end{aligned} \tag{34}$$

$$\begin{aligned}
\operatorname{cov}\left(W_{11}^{(0)}\phi'(Z_1^{(0)}),\ Z_1^{(0)}\right) &= \cancelto{0}{\operatorname{cov}\left(W_{11}^{(0)},\, Z_1^{(0)}\right)}\mathbb{E}\left[\phi'(Z_1^{(0)})\right]\\
&+ \operatorname{cov}\left(\phi'(Z_1^{(0)}),\, Z_1^{(0)}\right)\mathbb{E}\left[W_{11}^{(0)}\right].
\end{aligned} \tag{35}$$

Note that the formulations for $\operatorname{cov}\left(\phi'(Z_1^{(1)}),\, Z_1^{(0)}\right)$ and $\operatorname{cov}\left(\phi'(Z_1^{(0)}),\, Z_1^{(0)}\right)$ are provided in §B.5. As mentioned in §B.3.3, there are six branches relating to $\frac{\partial A_1^{(3)}}{\partial Z_1^{(0)}}$. Therefore, we apply the same calculations for the five remaining branches. The final covariance between $\frac{\partial A_1^{(3)}}{\partial Z_1^{(0)}}$ and $Z_1^{(0)}$ is equal to the sum of the covariance of these branches.

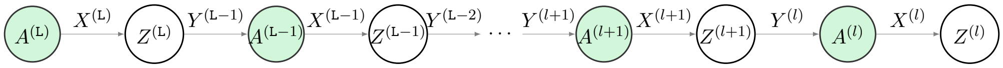


Figure 14: Illustration of a branch in the partial derivative diagram.

B.4 Generalization

This section presents the generalized formulations for a branch of the partial derivative diagram for a feedforward neural networks relating to the partial derivative of an activation unit at layer $\mathtt{L}$, that is, $A^{(\mathtt{L})}$ with respect to a hidden state at layer l, that is, $Z^{(l)}$. Figure 14 shows a branch of the partial derivative diagram for a FNN.

B.4.1 Partial Derivative

$$\frac{\partial a^{(\mathtt{L})}}{\partial z^{(l)}} = X^{(\mathtt{L})}Y^{(\mathtt{L}-1)}X^{(\mathtt{L}-1)}.\cdots Y^{(l+1)}X^{(l+1)}Y^{(l)}X^{(l)} \tag{36}$$

The expected value is computed following

$$\begin{aligned}\mathbb{E}\left[\frac{\partial a^{(\mathtt{L})}}{\partial z^{(l)}}\right] &= \underbrace{\mathbb{E}\left[X^{(\mathtt{L})}Y^{(\mathtt{L}-1)}X^{(\mathtt{L}-1)}\cdots Y^{(l+1)}X^{(l+1)}\right]}_{\mathbb{E}\left[\frac{\partial a^{(\mathtt{L})}}{\partial z^{(l+1)}}\right]}\mathbb{E}\left[Y^{(l)}\right]\mathbb{E}\left[X^{(l)}\right]\\ &+ \operatorname{cov}\left(X^{(\mathtt{L})}Y^{(\mathtt{L}-1)}X^{(\mathtt{L}-1)}\cdots Y^{(l+1)}X^{(l+1)},\ Y^{(l)}X^{(l)}\right),\end{aligned} \tag{37}$$

where

$$\begin{aligned}&\operatorname{cov}\left(X^{(\mathtt{L})}Y^{(\mathtt{L}-1)}X^{(\mathtt{L}-1)}\cdots Y^{(l+1)}X^{(l+1)},\ Y^{(l)}X^{(l)}\right)\\ &\qquad = \left\{\operatorname{cov}\left(X^{(l+1)},Y^{(l)}\right)\mathbb{E}\left[X^{(l)}\right]\right.\\ &\qquad + \left.\operatorname{cov}\left(X^{(l+1)},X^{(l)}\right)\mathbb{E}\left[Y^{(l)}\right]\right\}\\ &\qquad \times\ \mathbb{E}\left[Y^{(l+1)}\right]\underbrace{\mathbb{E}\left[X^{(\mathtt{L})}Y^{(\mathtt{L}-1)}X^{(\mathtt{L}-1)}\right]}_{\mathbb{E}\left[\frac{\partial a^{(\mathtt{L})}}{\partial z^{(\mathtt{L}-1)}}\right]}.\end{aligned} \tag{38}$$

The variance is computed following

$$\begin{aligned}\mathrm{var}\left(\frac{\partial a^{(\mathtt{L})}}{\partial z^{(l)}}\right) &= \underbrace{\mathrm{var}\left(X^{(\mathtt{L})}Y^{(\mathtt{L}-1)}X^{(\mathtt{L}-1)}\dots Y^{(l+1)}X^{(l+1)}\right)}_{\mathrm{var}\left(\frac{\partial a^{(\mathtt{L})}}{\partial z^{(l+1)}}\right)}\mathrm{var}\left(Y^{(l)}X^{(l)}\right)\\ &+ \mathrm{cov}\left(X^{(\mathtt{L})}Y^{(\mathtt{L}-1)}X^{(\mathtt{L}-1)}\dots Y^{(l+1)}X^{(l+1)},\ Y^{(l)}X^{(l)}\right)^2\\ &+ 2\mathrm{cov}\left(X^{(\mathtt{L})}Y^{(\mathtt{L}-1)}X^{(\mathtt{L}-1)}\dots Y^{(l+1)}X^{(l+1)},\ Y^{(l)}X^{(l)}\right)\\ &\times \underbrace{\mathbb{E}\left[X^{(\mathtt{L})}Y^{(\mathtt{L}-1)}X^{(\mathtt{L}-1)}\dots Y^{(l+1)}X^{(l+1)}\right]}_{\mathbb{E}\left[\frac{\partial a^{(\mathtt{L})}}{\partial z^{(l+1)}}\right]}\mathbb{E}\left[Y^{(l)}\right]\mathbb{E}\left[X^{(l)}\right]\\ &+ \mathrm{var}\left(X^{(\mathtt{L})}Y^{(\mathtt{L}-1)}X^{(\mathtt{L}-1)}\dots Y^{(l+1)}X^{(l+1)}\right)\mathbb{E}\left[Y^{(l)}\right]^2\mathbb{E}\left[X^{(l)}\right]^2\\ &+ \mathrm{var}\left(Y^{(l)}X^{(l)}\right)\mathbb{E}\left[X^{(\mathtt{L})}Y^{(\mathtt{L}-1)}X^{(\mathtt{L}-1)}\dots Y^{(l+1)}X^{(l+1)}\right]^2.\end{aligned} \tag{39}$$

B.4.2 Covariance between Partial Derivative and Hidden State

$$\begin{aligned}&\mathrm{cov}\left(\frac{\partial a^{(\mathtt{L})}}{\partial z^{(l)}},\, z^{(l)}\right)\\ &= \mathrm{cov}\left(X^{(l+1)},\, z^{(l)}\right)\mathbb{E}\left[Y^{(l+1)}\right]\mathbb{E}\left[Y^{(l)}\right]\mathbb{E}\left[X^{(l)}\right]\underbrace{\mathbb{E}\left[X^{(\mathtt{L})}Y^{(\mathtt{L}-1)}X^{(\mathtt{L}-1)}\right]}_{\mathbb{E}\left[\frac{\partial a^{(\mathtt{L})}}{\partial z^{(\mathtt{L}-1)}}\right]}\\ &+ \mathrm{cov}\left(X^{(l)},\, z^{(l)}\right)\mathbb{E}\left[Y^{(l)}\right]\underbrace{\mathbb{E}\left[X^{(\mathtt{L})}Y^{(\mathtt{L}-1)}X^{(\mathtt{L}-1)}\dots Y^{(l+1)}X^{(l+1)}\right]}_{\mathbb{E}\left[\frac{\partial a^{(\mathtt{L})}}{\partial z^{(l+1)}}\right]}.\end{aligned} \tag{40}$$

B.5 Activation Function

B.5.1 Tanh(Z)

The derivative of the function $\phi(Z) = tanh(Z)$ with respect to the hidden state Z is written as

$$\phi'(Z) = \frac{d\phi(z)}{dz} = 1 - \phi(Z)^2. \tag{41}$$

The expected value of $\phi'(Z_j^{(l)})$ is computed using Equation 13 and 41

$$\begin{aligned}\mathbb{E}\left[\phi'(z)\right] &= \mathbb{E}\left[1 - \phi(Z_j^{(l)})^2\right]\\ &= 1 - \mathbb{E}\left[\phi\left(Z_j^{(l)}\right)\right]^2 - \mathrm{var}\left(\phi(Z_j^{(l)})\right).\end{aligned} \tag{42}$$

The variance of $\phi'(Z_j^{(l)})$ is computed using Equation 16

$$\begin{aligned} \operatorname{var}\left(\phi'(Z_j^{(l)})\right) &= \operatorname{var}\left(1-\phi(Z_j^{(l)})^2\right) \\ &= \operatorname{var}\left(\phi(Z_j^{(l)})^2\right) \\ &= 2\operatorname{var}\left(\phi(Z_j^{(l)})\right)\left\{\operatorname{var}\left(Z_j^{(l)}\right)+2\,\mathbb{E}\left[\phi(Z_j^{(l)})\right]^2\right\}. \end{aligned} \tag{43}$$

The covariance between $\phi(Z_i^{(l+1)})$ and $W_{ij}^{(l)}$ is computed using Equation 14

$$\begin{aligned} \operatorname{cov}\left(\phi'(Z_i^{(l+1)}),\,W_{ij}^{(l)}\right) &= \operatorname{cov}\left(1-\phi(Z_i^{(l+1)})^2,\,W_{ij}^{(l)}\right) \\ &= -\operatorname{cov}\left(\phi(Z_i^{(l+1)})^2,\,W_{ij}^{(l)}\right) \\ &= -2\operatorname{cov}\left(\phi(Z_i^{(l+1)}),\,W_{ij}^{(l)}\right)\,\mathbb{E}\left[\phi(Z_i^{(l+1)})\right]. \end{aligned} \tag{44}$$

Using Equation 10 and 12, Equation 44 is rewritten as

$$\begin{aligned} &\operatorname{cov}\left(\phi'(Z_i^{(l+1)}),\,W_{ij}^{(l)}\right) \\ &\quad= -2\operatorname{cov}\left(J_i^{(l+1)}(Z_i^{(l+1)}-\mu_{Z_i}^{(l+1)})+\phi(\mu_{Z_i}^{(l+1)}),\,W_{ij}^{(l)}\right)\,\mathbb{E}\left[\phi(Z_i^{(l+1)})\right] \\ &\quad= -2\,J_i^{(l+1)}\operatorname{cov}\left(Z_i^{(l+1)},W_{ij}^{(l)}\right)\,\mathbb{E}\left[\phi(Z_i^{(l+1)})\right] \\ &\quad= -2\,J_i^{(l+1)}\operatorname{cov}\left(\sum_k W_{ik}^{(l)}\phi(Z_k^{(l)})+B_i^{(l)},\;W_{ij}^{(l)}\right)\,\mathbb{E}\left[\phi(Z_i^{(l+1)})\right] \\ &\quad= -2\,J_i^{(l+1)}\operatorname{cov}\left(W_{ij}^{(l)},W_{ij}^{(l)}\right)\,\mathbb{E}\left[\phi(Z_i^{(l)})\right]\,\mathbb{E}\left[\phi(Z_i^{(l+1)})\right]. \end{aligned} \tag{45}$$

The covariance between $\phi'(Z_j^{(l+1)})$ and $\phi'(Z_i^{(l)})$ is obtained using Equation 15, 10, 11, and 12,

$$\begin{aligned} \operatorname{cov}\left(\phi'(Z_i^{(l+1)}),\,\phi'(Z_j^{(l)})\right) &= \operatorname{cov}\left(1-\phi(Z_i^{(l+1)})^2,\;1-\phi(Z_j^{(l)})^2\right) \\ &= \operatorname{cov}\left(\phi(Z_i^{(l+1)})^2,\;\phi(Z_j^{(l)})^2\right) \\ &= 2\operatorname{cov}\left(\phi(Z_i^{(l+1)}),\;\phi(Z_j^{(l)})\right)^2 \\ &+ 4\operatorname{cov}\left(\phi(Z_i^{(l+1)}),\;\phi(Z_j^{(l)})\right)\,\mathbb{E}\left[\phi(Z_i^{(l+1)})\right]\,\mathbb{E}\left[\phi(Z_j^{(l)})\right], \end{aligned} \tag{46}$$

where

$$\begin{aligned}
\operatorname{cov}\left(\phi(Z_i^{(l+1)}),\ \phi(Z_j^{(l)})\right) &= J_i^{(l+1)}\operatorname{cov}\left(\sum_k W_{ik}^{(l)}\phi(Z_k^{(l)}) + B_i^{(l)},\ \phi(Z_i^{(l)})\right)\\
&= J_i^{(l+1)}\operatorname{cov}\left(W_{ij}^{(l)}\phi(Z_j^{(l)}),\ \phi(Z_i^{(l)})\right)\\
&= J_i^{(l+1)}\operatorname{cov}\left(\phi(Z_j^{(l)}),\ \phi(Z_i^{(l)})\right)\ \mathbb{E}\left[W_{ij}^{(l)}\right]\\
&+ J_i^{(l+1)}\cancelto{0}{\operatorname{cov}\left(W_{ij}^{(l)},\ \phi(Z_i^{(l)})\right)}\ \mathbb{E}\left[\phi(Z_j^{(l)})\right].
\end{aligned}\tag{47}$$

The covariance between $\phi'(Z_i^{(l+1)})$ and $Z_j^{(l)}$ is computed using Equation 14, 10, 11, and 12,

$$\begin{aligned}
\operatorname{cov}\left(\phi'(Z_i^{(l+1)}),\ Z_j^{(l)}\right) &= \operatorname{cov}\left(1-\phi(Z_i^{(l+1)})^2,\ Z_j^{(l)}\right)\\
&= -2\operatorname{cov}\left(\phi(Z_i^{(l+1)}),\ Z_j^{(l)}\right)\ \mathbb{E}\left[Z_j^{(l)}\right]\\
&= -2\,J_i^{(l+1)}\operatorname{cov}\left(\sum_k W_{ik}^{(l)}\phi(Z_k^{(l)}) + B_i^{(l)},\ Z_j^{(l)}\right)\ \mathbb{E}\left[Z_j^{(l)}\right]\\
&= -2\,J_i^{(l+1)}\operatorname{cov}\left(W_{ij}^{(l)}\phi(Z_j^{(l)}),\ Z_j^{(l)}\right)\ \mathbb{E}\left[Z_j^{(l)}\right]\\
&= -2\,J_i^{(l+1)}\operatorname{cov}\left(\phi(Z_j^{(l)}),\ Z_j^{(l)}\right)\ \mathbb{E}\left[W_{ij}^{(l)}\right]\ \mathbb{E}\left[Z_j^{(l)}\right]\\
&- 2\,J_i^{(l+1)}\cancelto{0}{\operatorname{cov}\left(W_{ij}^{(l)},\ Z_j^{(l)}\right)}\mathbb{E}\left[\phi(Z_j^{(l)})\right]\ \mathbb{E}\left[Z_j^{(l)}\right]\\
&= 2\,J_i^{(l+1)}J_j^{(l)}\operatorname{cov}\left(Z_j^{(l)},\ Z_j^{(l)}\right)\ \mathbb{E}\left[W_{ij}^{(l)}\right]\ \mathbb{E}\left[Z_j^{(l)}\right].
\end{aligned}\tag{48}$$

The covariance between $\phi'(Z_j^{(l)})$ and $Z_j^{(l)}$ is computed using Equation 14

$$\begin{aligned}
\operatorname{cov}\left(\phi'(Z_j^{(l)}),\ Z_j^{(l)}\right) &= \operatorname{cov}\left(1-\phi(Z_j^{(l)})^2,\ Z_j^{(l)}\right)\\
&= -2\operatorname{cov}\left(\phi(Z_j^{(l)}), Z_j^{(l)}\right)\ \mathbb{E}\left[\phi(Z_j^{(l)})\right]\\
&= -2\,J_j^{(l)}\operatorname{cov}\left(Z_j^{(l)}, Z_j^{(l)}\right)\ \mathbb{E}\left[\phi(Z_j^{(l)})\right].
\end{aligned}\tag{49}$$

B.5.2 ReLU(Z)

The derivative of the function $\phi(Z) = \mathrm{ReLU}(Z)$ with respect to the hidden state Z and its covariance are formulated following

$$\begin{aligned}
\phi'(Z) &= \begin{cases} 1 & if \quad \mathbb{E}[Z] > 0 \\ 0 & if \quad \mathbb{E}[Z] \leq 0. \end{cases} \\
\mathbb{E}\left[\phi'(Z_j^{(l)})\right] &= 1 \\
\mathrm{var}\left(\phi'(Z_j^{(l)})\right) &= 0 \\
\mathrm{cov}\left(\phi'(Z_i^{(l+1)}),\, W_{ij}^{(l)}\right) &= 0 \\
\mathrm{cov}\left(\phi'(Z_i^{(l+1)}),\, \phi'(Z_j^{(l)})\right) &= 0 \\
\mathrm{cov}\left(\phi'(Z_i^{(l+1)}),\; Z_j^{(l)}\right) &= 0 \\
\mathrm{cov}\left(\phi'(Z_j^{(l)}),\; Z_j^{(l)}\right) &= 0.
\end{aligned} \tag{50}$$

References

N. Akhtar and A. Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.

L. Ardizzone, J. Kruse, C. Rother, and U. Köthe. Analyzing inverse problems with invertible neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

P.-Y. Chen, Y. Sharma, H. Zhang, J. Yi, and C.-J. Hsieh. Ead: elastic-net attacks to deep neural networks via adversarial examples. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Neural Information Processing Systems (NIPS)*, 2016.

P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov. Openai baselines. `https://github.com/openai/baselines`, 2017.

G. W. Ding, L. Wang, and X. Jin. AdverTorch v0.1: An adversarial robustness toolbox based on pytorch. *arXiv preprint arXiv:1902.07623*, 2019.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Neural Information Processing Systems (NIPS)*, volume 27, pages 2672–2680, 2014.

I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT Press, 2016.

J.-A. Goulet, L. H. Nguyen, and S. Amiri. Tractable approximate Gaussian inference for Bayesian neural networks. *Journal of Machine Learning Research*, 22(20-1009):1–23, 2021.

M.-C. Goulet. Neural networks in insurance. Master's thesis, Concordia University, July 2021.

D. P. Kingma and M. Welling. Auto-encoding variational Bayes. *stat*, 1050:1, 2014.

A. Krizhevsky et al. Learning multiple layers of features from tiny images. Master's thesis, University of Toronto, 2009.

J. Kruse, L. Ardizzone, C. Rother, and U. Köthe. Benchmarking invertible architectures on inverse problems. *arXiv*, cs.LG2101.10763, 2021.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *Stat*, 1050:9, 2017.

V. Mnih, Adria P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

A. Neelakantan, L. Vilnis, Q.V. Le, I. Sutskever, L. Kaiser, K. Kurach, and J. Martens. Adding gradient noise improves learning for very deep networks. *CoRR*, abs/1511.06807, 2015.

L. H. Nguyen and J.-A. Goulet. Analytically tractable Bayesian deep Q-learning. *arXiv preprint arXiv:2106.11086*, 2021a.

L. H. Nguyen and J-A. Goulet. Analytically tractable inference in deep neural networks. *arXiv preprint arXiv:2103.05461*, 2021b.

J.W.T Peters and M. Welling. Probabilistic binary neural networks. *arXiv preprint arXiv:1809.03368*, 2018.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533—536, 1986.

J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International conference on machine learning (ICML)*, pages 1889–1897. PMLR, 2015.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT Press, 2nd edition, 2018.

E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

Y. Wu, E. Mansimov, R. B. Grosse, S. Liao, and J. Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in neural information processing systems*, volume 30, pages 5279–5288, 2017.