Analytic Expressions for Probabilistic Moments of PL-DNN with Gaussian Input

Adel Bibi[*] Modar Alfadly[*] and Bernard Ghanem
King Abdullah University of Science and Technology (KAUST), Saudi Arabia
{adel.bibi,modar.alfadly,bernard.ghanem}@kaust.edu.sa

Abstract

*The outstanding performance of deep neural networks (DNNs), for the visual recognition task in particular, has been demonstrated on several large-scale benchmarks. This performance has immensely strengthened the line of research that aims to understand and analyze the driving reasons behind the effectiveness of these networks. One important aspect of this analysis has recently gained much attention, namely the reaction of a DNN to noisy input. This has spawned research on developing adversarial input attacks as well as training strategies that make DNNs more robust against these attacks. To this end, we derive in this paper exact analytic expressions for the first and second moments (mean and variance) of a small piecewise linear (PL) network (Affine, ReLU, Affine) subject to general Gaussian input. We experimentally show that these expressions are tight under simple linearizations of deeper PL-DNNs, especially popular architectures in the literature (*e.g. *LeNet and AlexNet). Extensive experiments on image classification show that these expressions can be used to study the behaviour of the output mean of the logits for each class, the interclass confusion and the pixel-level spatial noise sensitivity of the network. Moreover, we show how these expressions can be used to systematically construct targeted and non-targeted adversarial attacks.*

1. Introduction

Deep neural networks (DNNs) have demonstrated impressive performance over the years in many fields of research. The applications include object classification [14, 16], semantic segmentation [19], activity recognition/detection [7, 32], speech recognition [15] and bioinformatics [4] to name a few. Despite this success, DNNs still exhibit uncouth behaviour under certain circumstances. Some of these nuisances in performance became apparent when adversarial perturbations were first introduced [31]. For instance, despite their excellent visual recognition performance, there are various simple routines that can tailor

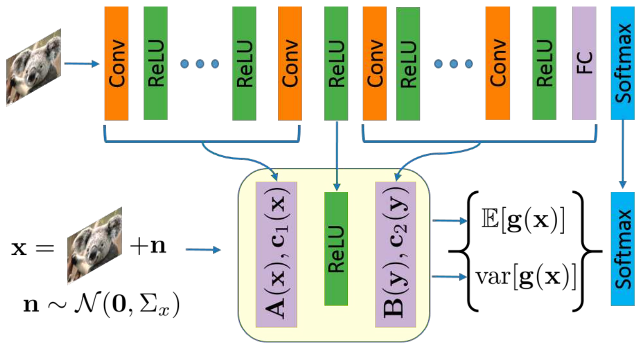
Conv
ReLU
ReLU
Conv
ReLU
Conv
ReLU
Conv
ReLU
FC
Softmax
$\mathbf{x} =$ $+\mathbf{n}$
$\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \Sigma_x)$
$\mathbf{A}(\mathbf{x}), \mathbf{c}_1(\mathbf{x})$
ReLU
$\mathbf{B}(\mathbf{y}), \mathbf{c}_2(\mathbf{y})$
$\mathbb{E}[\mathbf{g}(\mathbf{x})]$
$\text{var}[\mathbf{g}(\mathbf{x})]$
Softmax

Figure 1. Any PL-DNN can be linearized before and after a given ReLU through a two-stage linearization truncating it into a (Affine, ReLU, Affine) network, whose 1^{st} and 2^{nd} moments can be derived analytically when it is exposed to Gaussian input noise. We show that these moments are helpful in predicting how PL-DNNs react to noise and constructing adversarial Gaussian input attacks.

small imperceptible perturbations to the input that result in a drastic negative effect on classification performance [12, 24, 31]. More interestingly, some of these perturbations can be both image and network agnostic [23]. In light of these revelations, only little progress has been made to systematically understand the reasons behind such behaviour. However, there are some recent approaches that do try to deal with these nuisances. For instance and as a direct approach, it has been shown that augmenting the training data with these perturbations can in fact enhance the robustness of networks [12, 24]. Unfortunately, this brute force solution does not provide much insight on the reasons behind such behaviour. In addition to increasing the training set size and training time, such a strategy might still not be sufficient, since the space of all possible adversarial perturbations is possibly too large for augmentation to help.

Thus, in this paper, we derive expressions of the first and second moments (mean and consequently the variance) of a small piecewise linear (PL) network in the form of (Affine, ReLU, Affine) under Gaussian input. These two expressions provide a powerful tool for analyzing other larger PL-DNNs, which are by far the most popularly used networks in recent literature, through means of two-stage linearization (as shown in Figure 1). We show that these expressions provide much insight in understanding the behaviour

[*]Both authors contributed equally to this work.

of DNNs under adversarial attacks and in generating Gaussian adversarial attacks.

2. Related Work

Understanding and analyzing DNNs has a long rich history. In this section, we gather the most related work to ours, which generally touches upon the following four concepts.

Visual Representation. A natural means of analyzing DNNs is by examining their visual representations. There has been several works that have carried this out through visualization tools [6, 21, 33]. Moreover, there has been extensive empirical studies that attempt to explain DNN behaviour by investigating the effect of transferring features among different vision tasks [2]. Interestingly, Soatto *et al.* [29] has nicely linked the visual representation of a DNN with the underlying task (*e.g.* classification).

Robustness. There has been a wide range of studies analyzing the robustness of classifiers in general and DNNs in specific to the presence of noise [10, 11]. Different types of noise have been examined from additive to geometric transformations. For instance, Fawzi *et al.* [9] propose a generic probabilistic framework for analyzing classifier robustness to different nuisance factors. Moreover, the work of [8] proposes a method to assess classifier robustness to input undergoing geometric transformations.

Moreover, and most related to our work, analyzing and understanding DNNs undergoing additive input noise has been well studied in the literature [1, 3, 13, 31]. The literature here can in general be divided into two parts.

Noise Injection During Training. Another approach to analyzing DNNs is by injecting noise into various layers during backpropagation, which has been shown to improve performance significantly [13]. This phenomenon has been well founded since the work of [3], which shows that adding noise is equivalent to regularizing the loss with a term involving only first order derivatives of the DNN. Other experimental studies [20] show the effect on backgpropagation and performance of adding Gaussian, Binomial and Gamma noise to both the input and hidden layers.

Adversarial Perturbations. On the other hand of the spectrum and most recently, it has been demonstrated that one can tailor noise (perturbations) to the DNN input leading to severe reduction in classification performance [12, 24, 31]. For instance, Szegedy *et al.* [31] show that such perturbations can be constructed by maximizing the network's prediction error through a penalized optimization. The work of [23] proposes a much simpler approach by solving a projection problem with simple constraints and a closed form solution. These adversarial perturbations have been empirically shown to be doubly-universal, *i.e.* agnostic of both the input and DNN architecture [23]. Even more surprisingly, Su *et al.* [30] recently demonstrate that perturbing a single input pixel of some well-known DNNs can result in a misclassification rate as high as 70% on popular benchmarks. Furthermore, Nguyen *et al.* [25] propose a method to generate completely random images that fool a DNN into believing (with high certainty) that they belong to object classes seen in training. These phenomena are quite serious and can be menacing especially in real-world DNN deployments (*e.g.* self-driving cars). To mitigate their effects, it has been shown that augmenting training data with adversarial perturbations [12, 24] leads to a more robust network.

In this paper, we focus our analysis on PL-DNNs that employ ReLU activations. Unlike previous approaches, we study how the probabilistic moments of the output of a PL-DNN with a Gaussian input can be computed analytically. We do this by first deriving exact expressions for the first and second moments (the mean and consequently variance) of a simple (Affine, ReLU, Affine) network. Then, we extrapolate these expressions to deeper PL-DNNs by employing a simple two-stage linearization step that locally approximates it with a (Affine, ReLU, Affine) network. Since these expressions are a function of the noise parameters, they are particularly useful in analyzing and inferring the behaviour of the original PL-DNN *without* the need to probe the network with inputs sampled from the noise distribution, as done in previous work for data augmentation [12, 24].

Contributions. (i) We provide a new perspective to analyzing PL-DNNs by deriving closed form expressions for the output mean and variance of a network in the form (Affine, ReLU, Affine) in the presence of Gaussian input noise. Since all PL-DNN networks can be locally *linearized* to have this form, we show with extensive experiments that the original network can also be analyzed using the derived expressions, so long as the linearization is reasonable (*i.e.* our expressions are tight). **(ii)** Applying these expressions on popular network architectures lead to interesting insights for the task of image classification. For example, they can reliably predict the fooling rate of well-known adversarial perturbations [23] to AlexNet [16]. **(iii)** By optimizing for the input noise parameters, we show how these expressions can be proactively used to construct targeted and non-targeted Gaussian adversarial attacks, as well as, $\alpha\%$-noise attacks where only $\alpha\%$ of the input pixels are corrupted.

3. Network Moments

In this section, we first analyze networks of the form (Affine, ReLU, Affine) in the presence of Gaussian input noise. The functional form of this particularly shaped network is: $\mathbf{g}(\mathbf{x}) = \mathbf{B}\max(\mathbf{A}\mathbf{x} + \mathbf{c}_1, \mathbf{0}_p) + \mathbf{c}_2$, where $\max(.)$ is an element wise operator. For example, the output of $\mathbf{g}$ can be the output logits of d classes of a shallow network, whereby $\mathbf{g} : \mathbb{R}^n \rightarrow \mathbb{R}^d$. The affine mappings can be of any size, and throughout the paper we assume that $\mathbf{A} \in \mathbb{R}^{p\times n}$ and $\mathbf{B} \in \mathbb{R}^{d\times p}$. Also, note that $\mathbf{g}$ can also include convoltuional layers, since they are also linear mappings with particular structure (circulant or Toeplitz).

Let us now consider $\mathbf{x}$ to be a random Gaussian vector. In this case, the problem is cast as a nonlinear random variable mapping, where we ideally seek the probability density function (PDF) of $\mathbf{g}(\mathbf{x})$ when $\mathbf{x} \sim \mathcal{N}(\mu_x, \Sigma_x)$. Computing this PDF is possible when $\mathbf{B} = \mathbf{I}$; however, the problem is much more difficult for arbitrary $\mathbf{B}$ in general. Thus, we focus instead on deriving probabilistic moments of this unknown distribution. This section is divided into two theorems: one theorem that derives the first moment (mean) of the output $\mathbf{g}(.)$, *i.e.* $\mathbb{E}[\mathbf{g}(\mathbf{x})]$, and another that derives its second moment, *i.e.* $\mathbb{E}[\mathbf{g}^2(\mathbf{x})]$. We provide more details on the derivation of the second moment as it involves an extension to Price's Theorem [22, 27], which provides many insights on analyzing general nonlinear/non-smooth functions acting on multi-variate Gaussian inputs. For simplicity, and throughout the paper, we denote $\mathbf{g}_i(.)$ as the i^{th} function in $\mathbf{g}(.)$, *i.e.* $\mathbf{g}_i(\mathbf{x}) = \mathbf{B}(i,:)\max(\mathbf{A}\mathbf{x} + \mathbf{c}_1, \mathbf{0}_p) + \mathbf{c}_2(i)$.

3.1. Deriving the 1^{st} Output Moment: $\mathbb{E}[\mathbf{g}(\mathbf{x})]$

To derive the first moment of $\mathbf{g}(\mathbf{x})$, we first consider the scalar function $q(x) = ReLU(x) = \max(x, 0)$ acting on a single Gaussian random variable x.

Remark 1. *The PDF of $q(x) = \max(x,0) : \mathbb{R} \to \mathbb{R}$ where $x \sim \mathcal{N}(\mu_x, \sigma_x^2)$ is:*

$$f_q(x) = Q\left(\frac{\mu_x}{\sigma_x}\right)\delta(x) + f_x(x)u(x)$$

where $Q(.)$ is the Gaussian Q-function, $\delta(x)$ is the dirac function, $f_x(x)$ is the Gaussian PDF and $u(.)$ is the unit step function. It follows directly that $\mathbb{E}[q(x)] = \frac{\sigma_x}{\sqrt{2\pi}}$.

Now, we derive the first moment of $\mathbf{g}(\mathbf{x})$.

Theorem 1. *For any function in the form of $\mathbf{g}(\mathbf{x})$ where $\mathbf{x} \sim \mathcal{N}(\mu_x, \Sigma_x)$, we have:*

$$\mathbb{E}[\mathbf{g}_i(\mathbf{x})] = \sum_{v=1}^{p}\mathbf{B}(i,v)\left(\frac{1}{2}\bar{\mu}_v - \frac{1}{2}\bar{\mu}_v erf\left(\frac{-\bar{\mu}_v}{\sqrt{2}\bar{\sigma}_v}\right) + \frac{1}{\sqrt{2\pi}}\bar{\sigma}_v exp\left(-\frac{\bar{\mu}_v^2}{2\bar{\sigma}_v^2}\right)\right) + \mathbf{c}_2(i)$$

where $\bar{\mu}_v = (\mathbf{A}\mu_x + \mathbf{c}_1)(v)$, $\bar{\Sigma} = \mathbf{A}\Sigma_x\mathbf{A}^\top$, $\bar{\sigma}_v^2 = \bar{\Sigma}(v,v)$ and $erf(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$ is the error function.

Proof. Based on Remark (1) and noting that $(\mathbf{A}\mathbf{x} + \mathbf{c}_1) \sim \mathcal{N}(\bar{\mu}, \bar{\Sigma})$, we have:

$$\begin{aligned}\tilde{\mu}(i) &= \int_0^\infty z\frac{1}{\sqrt{2\pi}\bar{\sigma}_i}\exp\left(-\frac{(z-\bar{\mu}_i)^2}{2\bar{\sigma}_i^2}\right)dz \\ &= \frac{1}{2}\bar{\mu}_i - \frac{1}{2}\bar{\mu}_i\text{erf}\left(\frac{-\bar{\mu}_i}{\sqrt{2}\bar{\sigma}_i}\right) + \frac{1}{\sqrt{2\pi}}\bar{\sigma}_i\exp\left(-\frac{\bar{\mu}_i^2}{2\bar{\sigma}_i^2}\right)\end{aligned}$$

Thus, $\mathbb{E}[\mathbf{g}_i(\mathbf{x})] = \sum_{v=1}^{p}\mathbf{B}(i,v)\tilde{\mu}(v) + \mathbf{c}_2(i)$. □

3.2. Deriving the 2^{nd} Output Moment: $\mathbb{E}[\mathbf{g}^2(\mathbf{x})]$

Here, we need three pre-requisite lemmas: one characterizes the PDF of a squared ReLU (Lemma 1), one extends Price's Theorem (Lemma 2), and one derives the first moment of the product of two ReLU functions. Details of all the derivations are in the **supplementary material**.

Lemma 1. *The PDF of $q^2(x) = \max^2(x,0) : \mathbb{R} \to \mathbb{R}$ where $x \sim \mathcal{N}(0, \sigma_x^2)$ is :*

$$f_{q^2}(x) = \frac{1}{2}\delta(x) + \frac{1}{2\sqrt{x}}f_x(\sqrt{x})u(\sqrt{x})$$

and its first moment is $\mathbb{E}[q^2(x)] = \frac{\sigma_x^2}{2}$.

Proof. The proof is straightforward by using the cumulative distribution function (CDF):

$$F_{q^2}(c) = \mathbb{P}(\max^2(x,0) \leq c) = \frac{1}{2}\delta(c) + F_x(\sqrt{c})u\left(\sqrt{c}\right) \tag{1}$$

F_{q^2} and F_x are the CDFs of q^2 and x, respectively. The proof is complete by differentiating the smooth part of (1) with respect to c. □

Lemma 2. *Let $\mathbf{x} \in \mathbb{R}^n \sim \mathcal{N}(\mu_x, \Sigma_x)$ for any even p, where $\sigma_{ij} = \Sigma_x(i,j)\ \forall i \neq j$. Then, under mild assumptions on the nonlinear map $\Psi : \mathbb{R}^n \to \mathbb{R}$, we have $\frac{\partial^{\frac{p}{2}}\mathbb{E}[\Psi(\mathbf{x})]}{\prod_{\forall odd i}\partial\sigma_{ii+1}} = \mathbb{E}[\frac{\partial^p\Psi(\mathbf{x})}{\partial x_1\ldots\partial x_p}]$.*

Proof. The proof is very similar to the one found in [26] but with n variables and by taking gradients with respect to consecutive covariances $\Sigma(i,j)$. Details of all derivations are in the **supplementary material**. □

Lemma (2) relates the mean of the gradients/subgradients of any nonlinear function to the gradients/subgradients of the mean of that function. This lemma has Price's theorem [27] as a special case when function $\Psi(\mathbf{x}) = \prod_i^n \Psi_i(x_n)$ and $\Sigma(i,i) = 1\ \forall i$. It is worthwhile to note that there is an extension to Price's theorem [22], where the assumptions $\sigma_{ii}^2 = 1\ \forall i$ and $\Psi(\mathbf{x}) = \prod_i^n \Psi_i(x_i)$ are dropped; however, it only holds for the bivariate case (*i.e.* $n = 2$) and thus is a special case of Lemma (2).

Lemma 3. *For any bivariate Gaussian $\mathbf{x} \sim \mathcal{N}(\mathbf{0}_2, \Sigma_x)$, the following holds for $T(x_1,x_2) = \max(x_1,0)\max(x_2,0)$:*

$$\mathbb{E}[T(x_1,x_2)] = \frac{1}{2\pi}\left(\sigma_{12}\sin^{-1}\left(\frac{\sigma_{12}}{\sigma_1\sigma_2}\right) + \sigma_1\sigma_2\sqrt{1 - \frac{\sigma_{12}^2}{\sigma_1^2\sigma_2^2}}\right) + \frac{\sigma_{12}}{4}$$

where $\sigma_{ij} = \Sigma_x(i,j)\ \forall i \neq j$ and $\sigma_i^2 = \Sigma_x(i,i)$.

Proof. Using Lemma (2) with $p = 4$ and choosing σ_{12} to be the covariances at which differentiation happens, we have:

$$\frac{\partial^2 \mathbb{E}[T(x_1,x_2)]}{\partial\sigma_{12}\partial\sigma_{12}} = \mathbb{E}\left[\frac{\partial^4 \mathbb{E}[T(x_1,x_2)]}{\partial x_1 \partial x_2 \partial x_1 \partial x_2}\right] = \mathbb{E}\left[\frac{\partial^4 \mathbb{E}[T(x_1,x_2)]}{\partial x_1^2 \partial x_2^2}\right] = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\frac{\sigma_{12}^2}{\sigma_1\sigma_2}}} \tag{2}$$

To solve the partial differential equation in Eq (2), two boundary conditions are needed. Similar to [27], they can be computed when $\sigma_{12} = 0$, which occurs when x_1 and x_2 are independent random variables. It is easy to show from Remark (1) that $\mathbb{E}[T(x_1,x_2)]|_{\sigma_{12}=0} = \frac{\sigma_1\sigma_2}{2\pi}$ and that,

$$\left.\frac{\partial \mathbb{E}[T(x_1,x_2)]}{\partial\sigma_{12}}\right|_{\sigma_{12}=0} \overset{\text{lemma (2)}}{=} \mathbb{E}[u(x_1)u(x_2)] \overset{\sigma_{12}=0}{=} \mathbb{E}[u(x_1)]\mathbb{E}[u(x_2)] = \frac{1}{4}$$

With these boundary conditions, we compute the integral to complete the proof.

$$\mathbb{E}[T(x_1,x_2)] = \int_0^{\sigma_{12}}\left[\frac{1}{4} + \int_0^y \frac{dc}{2\pi\sqrt{1-\frac{c^2}{\sigma_1^2\sigma_2^2}}}\right]dy + \frac{\sigma_1\sigma_2}{2\pi}$$

□

Theorem 2. *For any function in the form of* $\mathbf{g}(\mathbf{x})$ *where* $\mathbf{x} \sim \mathcal{N}(\mathbf{0}_n, \Sigma_x)$ *and that* $\mathbf{c}_1 = \mathbf{0}_p$ *then:*

$$\mathbb{E}[\mathbf{g}_i^2(\mathbf{x})] = 2\sum_{v_1}^{k}\sum_{v_2}^{v_1-1}\mathbf{B}(i,v_1)\mathbf{B}(i,v_2)\left(\frac{\bar{\sigma}_{v_1v_2}}{2\pi}\sin^{-1}\left(\frac{\bar{\sigma}_{v_1v_2}}{\bar{\sigma}_{v_1}\bar{\sigma}_{v_2}}\right) + \frac{\bar{\sigma}_{v_1}\bar{\sigma}_{v_2}}{2\pi}\sqrt{1-\frac{\bar{\sigma}_{v_1v_2}^2}{\bar{\sigma}_{v_1}^2\bar{\sigma}_{v_2}^2}} + \frac{\bar{\sigma}_{v_1v_2}}{4}\right) + \frac{1}{2}\sum_{r}^{k}\mathbf{B}(i,r)^2\bar{\sigma}_r^2 + \mathbf{c}_2(i)$$

Proof. The proof follows naturally after considering the much simpler scalar function that is in the form $\tilde{g}(\mathbf{z}) = \sum_t^d \alpha_t \max(z_t, 0)$ where $\mathbf{z} \in \mathbb{R}^d \sim \mathcal{N}(\mathbf{0}_d, \Sigma_z)$. Therefore, we have $\mathbb{E}[\tilde{g}^2(\mathbf{z})] = \sum_r^d \alpha_r^2 \mathbb{E}[\max^2(z_r, 0)] + 2\sum_{v_2 \leq v_1}\alpha_{v_1}\alpha_{v_2}\mathbb{E}[\max(z_{v_1},0)\max(z_{v_2},0)]$. Note that $\tilde{g}(\mathbf{z})$ is only a special case of $\mathbf{g}_i(\mathbf{x})$, where $\mathbf{z} = \mathbf{A}\mathbf{x}$ and $\alpha_t = \mathbf{B}(i,t)$. It is also clear from $\mathbb{E}[\tilde{g}(\mathbf{z})]$ that it is sufficient to analyze $\mathbb{E}[\tilde{g}(\mathbf{z})]$ in the bivariate case. Thus, the function we are interested in is $\mathbb{E}[\tilde{g}^2(z_1,z_2)] = \alpha_1^2\mathbb{E}[\max^2(z_1,0)] + \alpha_2^2\mathbb{E}[\max^2(z_2,0)] + 2\alpha_1\alpha_2\mathbb{E}[\max(z_1,0)\max(z_2,0)]$. Using Lemmas (1) and (3), the proof is complete. □

Lastly, the variance of $g(\mathbf{x})$ can be directly derived: $\text{var}(\mathbf{g}_i(\mathbf{x})) = \mathbb{E}[\mathbf{g}_i^2(\mathbf{x})] - \mathbb{E}[\mathbf{g}_i(\mathbf{x})]^2|_{\mu_x=\bar{\mathbf{0}}_k}$. Refer to the **supplementary material** for the compact analytic form.

3.3. Comments

It is important to note that **(i)** the 2nd moment is derived under the assumptions $\mu_x = \mathbf{0}_n$ and $\mathbf{c}_1 = \mathbf{0}_p$ as the analysis gets intractable otherwise. Motivated with the fact that the variance of a random variable is translation invariant, we later show experimentally that these assumptions are not very strict resulting in a linear relation between the true variance and our analytically derived one. **(ii)** Lemma (2) is particularly useful when the function $\Psi(\mathbf{x})$ is PL. Since the choice of p is arbitrary, one can choose p to be big enough so that all the subgradients $\frac{\partial^p}{\partial x_1 \dots \partial x_p}$ reduce to dirac functions, thus, simplifying the evaluation of an integral such that $\mathbb{E}[\partial^p/\partial x_1 \dots \partial x_p]$ is a direct evaluation of the Gaussian PDF [27]. As such, the ReLU activations in PL-DNNs reduce to dirac functions by taking two gradients.

3.4. Extending to Deeper PL-DNNs

To extend the previous results to deeper DNNs that are not in the form (Affine, ReLU, Affine), we first denote the larger DNN as $\mathbf{R}(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}^d$ (*e.g.* a mapping of the input to the logits of d classes). By choosing the l^{th} ReLU layer, any $\mathbf{R}(.)$ can be decomposed into: $\mathbf{R}(\mathbf{x}) = \mathbf{R}_{l+1}(\text{ReLU}_l(\mathbf{R}_{l-1}(\mathbf{x})))$. In this paper, we employ a simple two-stage (bottom and top) linearization based on Taylor series approximation to cast $\mathbf{R}(.)$ into the form (Affine, ReLU, Affine). For example, we can linearize it around points $\mathbf{x}$ and $\mathbf{y} = \text{ReLU}_l(\mathbf{R}_{l-1}(\mathbf{x}))$, such that $\mathbf{R}_{l-1}(\mathbf{x}) \approx \mathbf{A}\mathbf{x} + \mathbf{c}_1$ and $\mathbf{R}_{l+1}(\mathbf{y}) \approx \mathbf{B}\mathbf{y} + \mathbf{c}_2$. The resulting function after linearization is $\mathbf{R}(\mathbf{x}) \approx \mathbf{B}\max(\mathbf{A}\mathbf{x} + \mathbf{c}_1, \mathbf{0}_k) + \mathbf{c}_2$. Figure 1 illustrates this two-stage linearization. The strategy of choosing the points of linearization ($\mathbf{x}$ and $\mathbf{y}$) and layer l is left to Section 4.

4. Experiments

In this section, we discuss a variety of experiments to showcase: **(i)** the tightness of our derived network moment expressions across network architectures and input variance levels on both synthetic and real networks; **(ii)** their tightness across different choices of linearization; and **(iii)** how we can use our expressions to construct an adversarial Gaussian input attack that confuses a PL-DNN in various ways (*e.g.* a targeted, non-targeted, or sparse pixel attack).

4.1. Tightness of Network Moments

Although our derived expressions for the output mean and variance in Theorems (1) and (2) are tight for a (Affine, ReLU, Affine) network, it is conceivable that the two-stage linearization might impact the tightness of these expressions when applied to deeper PL-DNNs. Here, we empirically study their tightness by comparing them against Monte Carlo estimates for both synthetic and real networks.

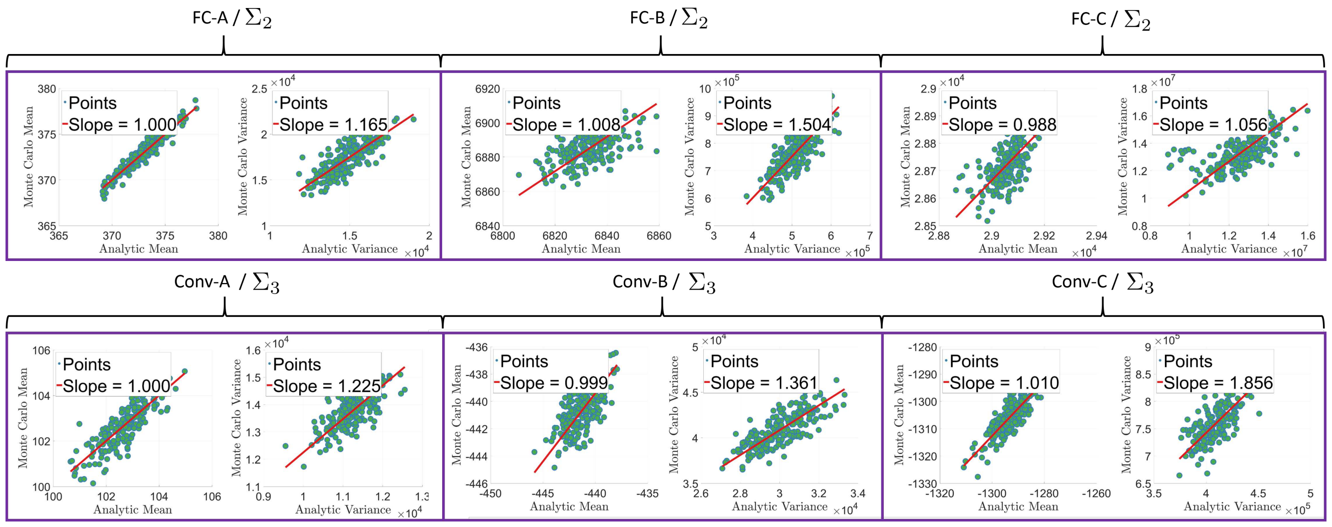

Figure 2. Shows the tightness between the analytic expressions against the Monte Carlo estimates with random Gaussian inputs. The experiment is conducted across different types of networks, different depths and different input variance level. To compare the tightness, we report the slope of the fitted line in the legend. The closer the slope to 1.0 the tighter the expressions to the Monte Carlo estimates.

Synthetic Networks on Synthetic Data. We consider two types of synthetic networks each with a varying number of hidden layers and subject them to various levels of input zero-mean Gaussian noise. The first type only consists of fully connected layers separated by ReLU activations with input size $\mathbb{R}^{100}$. We construct three networks of this type with 2, 3 and 4 layers, denoted as FC-A, FC-B and FC-C, respectively. The other network type consists of convolutional layers separated by ReLU activations with input size $\mathbb{R}^{20\times 20}$. Networks of this type have depths of 2, 3 and 4, denoted as Conv-A, Conv-B and Conv-C, respectively. For details about these networks' configurations, refer to the **supplementary material**. The weights of all networks are randomly generated as *i.i.d.* Gaussian. Note that all networks have a fully connected layer at the end to map the output to a single scalar. To study the effect of input noise level, we use three input noise regimes (low, medium and high) corresponding to three base random covariance matrices $(\Sigma_1, \Sigma_2, \Sigma_3)$, whose maximum variance levels are 10^{-4}, 1 and 200, respectively.

In each experiment, the noise covariances in the three regimes are generated randomly from each of the base covariances Σ_1, Σ_2 and Σ_3. For each Monte Carlo simulation, we randomly generate a set of 7.5×10^4 noisy input samples from each of the three noise regimes, perform forward passes through the networks, and compute the empirical mean and variance of the output. For our proposed analytic mean and variance, the two-stage linearization is done such that $\mathbf{x} = \mathbf{0}$, $\mathbf{y} = \mathbf{R}_{l-1}(\mathbf{x})$, and $l = 2$. The experiment is repeated such that we sample 200 random covariances similar to each of the base covariances Σ_1, Σ_2 and Σ_3.

To compare the tightness of our analytic results to the Monte Carlo estimates, we plot the results of the 200 experiments in Figure 2. Due to space constraints, we only consider the medium noise regime for FC-A, FC-B and FC-C, and the high noise regime for Conv-A, Conv-B and Conv-C. In each setup, we fit a line through the analytic and Monte Carlo results of the output mean and variance. The closer the slope of the fitted line is to 1.0, the tighter the analytic expressions. The expressions for the mean are tight across all networks, different network depths, and all input variance levels. Moreover, the analytic variances seem to be very accurate in general, but tend to be less tight with deeper networks (*e.g.* Conv-C). This is probably due to the assumption that $\mathbf{c}_1 = \mathbf{0}_p$ in our derivation, which does not exactly hold after the linearization. However, the analytic variance is unaffected by various input noise levels. A complete comparative study on each network for all three noise regimes can be found in the **supplementary material**.

LeNet on MNIST. In this section, we validate the tightness of the expressions in Theorems (1) and (2) on the well-known LeNet architecture [18] pretrained on the MNIST digit dataset [17]. Note that LeNet has a total of four layers, two of which are convolutional with max pooling and the other two are fully connected. The input to the network is $\mathbb{R}^{28\times 28}$ with 10 output logits (*i.e.* $d = 10$). Similar to the synthetic experiments, the number of Monte Carlo samples is 7.5×10^4 and the experiment is repeated 200 times. In this case, the two-stage linearization is done such that $\mathbf{x} = \mathbf{M}$, $\mathbf{y} = \mathbf{R}_{l-1}(\mathbf{x})$ and, for memory efficiency, $l = 3$. Here, $\mathbf{M}$ is an image randomly selected from the MNIST validation set, and for space constraints, we report the results for the medium noise regime only. As for the other noise regimes, refer to the **supplementary material**. Thus, the input is $\mathbf{x} \sim \mathcal{N}(\mathbf{M}, \Sigma_2)$. Since the LeNet architecture has $d = 10$, we will be reporting the tightness of the analytic mean and variance for $\mathbf{g}_i(\mathbf{x})\ \forall i$. As for the metric,

Table 1. Reports the average and standard deviation of the ratio between the Monte Carlo estimates of the mean and variance and their counterparts from the analytic expressions across all labels on MNIST. Refer to text for details.

Logits	$\mathbb{E}[\mathbb{E}_{ratio}]$	$\sigma(\mathbb{E}_{ratio})$	$\mathbb{E}[\text{var}_{ratio}]$	$\sigma(\text{var}_{ratio})$
$\mathbf{g}_0(\mathbf{x})$	1.007	0	0.5431	0.021
$\mathbf{g}_1(\mathbf{x})$	1.043	0.001	0.6192	0.021
$\mathbf{g}_2(\mathbf{x})$	0.916	0.001	0.3681	0.014
$\mathbf{g}_3(\mathbf{x})$	1.061	0.001	0.517	0.019
$\mathbf{g}_4(\mathbf{x})$	0.581	0.006	0.4878	0.016
$\mathbf{g}_5(\mathbf{x})$	0.699	0.004	0.4267	0.015
$\mathbf{g}_6(\mathbf{x})$	0.999	0	0.4795	0.017
$\mathbf{g}_7(\mathbf{x})$	0.990	0	0.446	0.014
$\mathbf{g}_8(\mathbf{x})$	1.001	0	0.4476	0.014
$\mathbf{g}_9(\mathbf{x})$	1.164	0.002	0.5931	0.020

we report the average ratio of the Monte Carlo estimates for the mean and variance to their counterparts from the analytic expressions such that: $\mathbb{E}[\mathbb{E}_{ratio}] = \mathbb{E}\left[\mathbb{E}_{MC}/\mathbb{E}[\mathbf{g}_i(\mathbf{x})]\right]$ and $\mathbb{E}[\text{var}_{ratio}] = \mathbb{E}\left[\text{var}_{MC}/\text{var}(g_i(\mathbf{x}))\right]$, respectively over the 200 runs. We also report the standard deviation of both quantities, which we denote as $\sigma(\mathbb{E}_{ratio})$ and $\sigma(\text{var}_{ratio})$.

In Table 1, we report the tightness results across all logit functions $\mathbf{g}_i(\mathbf{x}) \forall i$. These results verify the tightness of the analytic mean expression of Theorem (1), since the average ratios $\mathbb{E}[\mathbb{E}_{ratio}] \approx 1$ with a small $\sigma(\mathbb{E}_{ratio})$. On the other hand, and similar to the synthetic results, the variance estimate is slightly less accurate, since its analytic expression assumes that $\mathbf{c}_1 = \mathbf{0}_p$ and $\mu_x = \mathbf{0}_n$. However, despite this assumption, the ratio between the Monte Carlo estimates and the analytic expression holds to a nearly constant factor across all labels $\mathbb{E}[\text{var}_{ratio}] \approx \frac{1}{2}$. We will show later that $\mathbb{E}[\text{var}_{ratio}] \geq 0.2$. As compared to the mean estimates, an accurate estimate of the variance is not as important since, in many use cases, the variance expression can be used to design noise for adversarial attacks or to build DNNs that are robust and this can be achieved by either maximizing or minimizing the output variance, respectively. Thus, a scaled version of the output variance might be sufficient.

4.2. Studying Adversarial Noise on AlexNet

We now demonstrate the tightness of the analytic expression of the output mean using AlexNet [16] on the 1000 classes of ImageNet [5] (*i.e.* the 1000 logit functions before the final softmax layer). As shown earlier, the estimates of the analytic mean may exhibit minor errors due to linearization; however, we show here that this error marginally affects the score ordering of the output logits.

First, we show that the analytic expression for the mean can predict the fooling rate of a given input once a universal perturbation noise is added to it. More formally, we compute the average fooling rate [24] for all input images $\mathbf{X}_i + \mathbf{N}\ \forall i$ (or XN for short), where $\mathbf{X}_i$ is an image from the ImageNet validation set (50K images) and $\mathbf{N}$ is VGG16's [28] universal adversarial noise [23]. Note that AlexNet

Table 2. Shows ordering accuracy of the top-k classes of the analytic expression compared to the real predictions.

top-1 acc	top-2 acc	top-3 acc	top 4-acc	top-5 acc
98.3%	95.308%	91.156%	86.088%	80.514%

is considered fooled by $\mathbf{N}$ at image $\mathbf{X}_i$, if the class of the maximum output score is different when $\mathbf{N}$ is not present. To use our expressions, we replace the above deterministic setup with an equivalent probabilistic one by assuming the presence of small zero-mean Gaussian noise. Formally, we compute the average fooling rate of $\mathbf{X}_i + \mathbf{N} + \mathbf{n}$ (or XNn for short), where $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \Sigma_1)$ using our analytic expression for the mean of the d=1000 dimensional output. The number of Monte Carlo samples is left as before 7.5×10^4. The two-stage linearization of AlexNet is done such that $\mathbf{x} = \mathbf{X}_i + \mathbf{N}$, $\mathbf{y} = \mathbf{R}_{l-1}(\mathbf{x})$ and, for memory efficiency, $l = 7$. The fooling rate of XN is computed to be 38.70%, while that of XNn is 38.62%. This demonstrates that our moment expressions can very accurately predict the fooling rate of this universal adversarial attack.

To analyze this further, we conduct a second experiment, where we strictly compare the ordering accuracy of the top-k logit scores between the reference XN and the analytic expressions estimates of XNn with $k \in \{1, 2, 3, 4, 5\}$. The result is averaged over the complete ImageNet validation set (refer to Table 2). Here, an ordering score of 100% at a particular value of k concludes that the top-k elements of the 1000 logit scores generated by XN and XNn are exactly identical across all the validation set. These results show that the ordering score up to $k = 5$ is very high, indicating that our analytic expression for the mean (despite the linearization) is to a certain extent order preserving.

4.3. Sensitivity to Linearization

In all previous experiments, we validated the tightness of our derived network moments using different network architectures, input variance levels, and different datasets. However, in those experiments, the point at which two-stage linearization is done was restricted to be the input image. Obviously, this strategy is not scalable. So, we choose a set of representative input images, at which the two-stage linearization parameters $\mathbf{A}, \mathbf{B}, \mathbf{c}_1$ and $\mathbf{c}_2$ are computed once. Now, to evaluate the network moments for a new input, we simply use the two-stage linearization parameters of the *closest* linearization point to this input.

In this experiment, we want to study the tightness of our expressions under this more relaxed linearization strategy using LeNet [18] on the MNIST validation set (5K images) [17]. We cluster the images in the validation dataset using k-means on the image intensity space with different values of k. We use the cluster centers as the linearization points. Table 3 summarizes the tightness of the expressions for $k \in \{250, 500, 5000\}$ and compares them against a weak baseline, where the linearization point is set to be the *farthest* image in each cluster from the cluster center with

Table 3. Demonstrates the tightness of the analytic expressions on LeNet under varying number of linearization points (*i.e.* k-means cluster centers). As for the Baseline experiment, the linearization points are set to be the furthest instances from the clusters' centers.

Logits	k=5000		k=500		k=250		k=250 (Baseline)	
	$\mathbb{E}[\mathbb{E}_{ratio}]$	$\mathbb{E}[\text{var}_{ratio}]$	$\mathbb{E}[\mathbb{E}_{ratio}]$	$\mathbb{E}[\text{var}_{ratio}]$	$\mathbb{E}[\mathbb{E}_{ratio}]$	$\mathbb{E}[\text{var}_{ratio}]$	$\mathbb{E}[\mathbb{E}_{ratio}]$	$\mathbb{E}[\text{var}_{ratio}]$
$\mathbf{g}_0(\mathbf{x})$	0.9998	0.2725	0.752	0.2851	1.1406	0.2849	1.4044	0.2818
$\mathbf{g}_1(\mathbf{x})$	0.9983	0.3782	1.0235	0.3668	1.0587	0.3668	1.2616	0.3702
$\mathbf{g}_2(\mathbf{x})$	0.992	0.2298	1.6994	0.2366	1.8291	0.2499	-3.8096	0.2469
$\mathbf{g}_3(\mathbf{x})$	0.9959	0.2604	1.2021	0.2673	1.3191	0.2582	1.7718	0.2875
$\mathbf{g}_4(\mathbf{x})$	0.9932	0.2761	1.2705	0.3012	1.5868	0.2945	2.6598	0.308
$\mathbf{g}_5(\mathbf{x})$	0.9936	0.2441	0.9686	0.2523	1.5272	0.2429	1.3677	0.2663
$\mathbf{g}_6(\mathbf{x})$	0.991	0.3592	1.1909	0.3739	1.257	0.3539	1.8849	0.3916
$\mathbf{g}_7(\mathbf{x})$	0.9844	0.213	2.5221	0.2156	1.3419	0.2169	2.9314	0.2258
$\mathbf{g}_8(\mathbf{x})$	0.9906	0.2554	1.1877	0.25	1.9278	0.2541	1.5456	0.2571
$\mathbf{g}_9(\mathbf{x})$	0.9917	0.2685	1.3469	0.2796	1.9592	0.2832	2.1028	0.2926

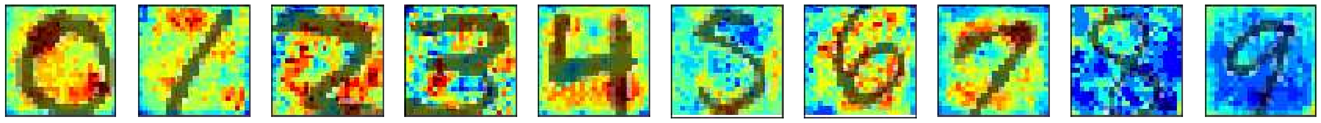
Figure 3. Shows an image of each class label overlaid with a heat map indicating pixel locations contribution to the output fooling rate as computed using the analytic expression of the output mean. The fooling rate increases as the colors transition from blue to red.

$k = 250$. It is clear that the analytic mean remains very close to the Monte Carlo estimate, even when the number of linearization points is as low as $k = 250$, *i.e.* only 5% of the validation set. Interestingly, even though our mean is a much looser approximation of the Monte Carlo estimate for the weak baseline, it does a decent job at predicting this estimate for many of the classes. As for the analytic variance, its tightness remains in an acceptable range with $\mathbb{E}[\text{var}_{ratio}] \in [0.2, 0.4]$ across all linearization strategies. Therefore, we conjecture that translating the Gaussian at the input by either having $\mu_x \neq \mathbf{0}_n$ and/or by having $\mathbf{c}_1 \neq \mathbf{0}_p$ may not significantly impact the output variance.

4.4. Localized Spatial Noise

Now that the tightness of our network moments has been established, we show how they can be utilized in predicting the effects of spatially localized adversarial attacks. Here, we revisit LeNet and the MNIST validation dataset. Instead of adding noise to each MNIST image across all its pixels and measuring how many images are fooled with this noise, we study the effect of spatially localized noise. We model this noise as a Gaussian zero-mean input $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \Sigma_3)$ constructed to only act on a window of 7×7 pixels. In this case, Σ_3 has zero rows and columns corresponding to the pixels that are not to be corrupted. To investigate the spatial sensitivity to noise, we slide this window with no padding and a stride of 1×1 across all images of MNIST and measure the fooling rate at every location. In this way, we create a spatial *fooling map* based on the analytic expression of the mean. In Figure 3, we visualize the average fooling map across all images belonging to the same class label. Arguably, the fooling rate is highest (red) at locations where

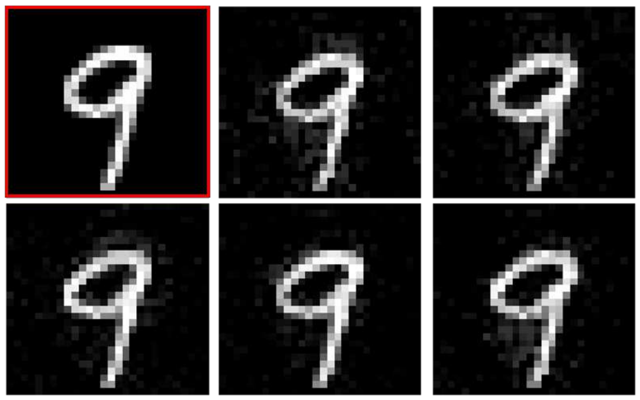
Figure 4. Shows noisy images that fool LeNet. The images from top left to bottom right are the original image from the validation set and the noisy versions classified as 2,3,4,7 and 8, respectively.

the noise can make the image (*e.g.* 1 and 2) look similar to images from another label (*e.g.* 7 and 0, respectively).

4.5. Noise Construction

Apart from being able to predict the effect of pre-defined input noise on a network, another important feature of this work is that the derived output mean and variance expressions can be used to construct noise with certain properties (*e.g.* by embedding them in an optimization for maximum fooling). In this experiment, we show how this can be done for targeted and non-targeted adversarial attacks.

Given an image $\mathbf{M}$ whose predicted class label is i, we want to add noise $\mathbf{x} \sim \mathcal{N}(\mu_x, \Sigma_x)$ such that the expected value of the network's prediction of $\mathbf{M} + \mathbf{x}$ is $j \neq i$. If such noise exists, we say the network is fooled in expectation. To keep the optimization and the number of variables manageable, we assume independent noise across pixels with the same variance level, *i.e.* $\Sigma_x = \sigma^2 \mathbf{I}_n$. Also, we define $\mathcal{E}_i^{\mathbf{M}}(\mu_x, \sigma^2) \equiv \mathbb{E}[\mathbf{g}_i(\mathbf{M} + \mathbf{x}_{(\mu_x, \sigma^2 \mathbf{I}_n)})]$ to avoid text clutter. In the following experiments, the two-stage linearization is carried around $\mathbf{M}$ at $l = 3$ for LeNet and $l = 7$ for AlexNet.

Targeted Attack. On the MNIST dataset, we specify a target class label j and we construct a noise that can fool LeNet in expectation by solving the following optimization:

$$\begin{aligned} \arg\min_{\mu_x, \sigma} & \left(\max_{i \neq j} \left(\mathcal{E}_i^{\mathbf{M}}(\mu_x, \sigma^2) \right) - \mathcal{E}_j^{\mathbf{M}}(\mu_x, \sigma^2) \right) \\ \text{s.t. } & 0 < \sigma^2 \leq 2, \quad -\beta \mathbf{1}_n \leq \mu_x \leq \beta \mathbf{1}_n \end{aligned} \tag{3}$$

In this experiment, we set $\beta = 30$ and solve Eq (3) using an interior-point method. Note that the range of pixel values of MNIST images is $[-127.5, 127.5]$. Figure 4 shows examples of noisy versions of an image from class label 9 that fool LeNet in expectation with multiple target class labels (*i.e.* $j \in \{2, 3, 4, 7, 8\}$). Not every target class label is easily achieved with small β because of the distance in their logits scores. We verify that the constructed noise actually fools the network by sampling from the learned distribution, passing each noisy input through LeNet, and verifying that the predicted label flips from 9 to the target class j.

Non-targeted Attack with $\alpha\%$-pixel Support. Inspired by the findings of some recent work [30], we demonstrate that we can construct additive noise that only corrupts $\alpha\%$ of the pixels in an input image, but still changes the image's class label prediction. Here, we use LeNet on MNIST and AlexNet on ImageNet. In this case, we do not specify the target class label j, so we optimize for the logit score of the correct class label i to be less than the maximum score. The underlying optimization is formulated as follows:

$$\begin{aligned} \arg\min_{\mu_x^\alpha, \sigma} & \left(\mathcal{E}_i^{\mathbf{M}}(\mu_x^\alpha, \sigma^2) - \max_{j \neq i} \left(\mathcal{E}_j^{\mathbf{M}}(\mu_x^\alpha, \sigma^2) \right) \right) \\ \text{s.t. } & 0 < \sigma^2 \leq 2, \quad -\beta \mathbf{1}_{\alpha n} \leq \mu_x^\alpha \leq \beta \mathbf{1}_{\alpha n} \end{aligned} \tag{4}$$

The optimization variable μ_x^α indicates the set of sparse pixels ($\alpha\%$ of the total number of pixels) in μ_x that will be corrupted while the rest of pixels are set to 0. The locations of the corrupted pixels are randomly chosen and fixed before solving the optimization. Two experiments are conducted on few images, one on MNIST and the other on ImageNet. Figures 5 and 6 show examples of noisy images constructed by solving Eq (4) with $\alpha = 4\%$ to fool LeNet and $\alpha = 2\%$ to fool AlexNet. Since there are much fewer pixels to flip the network's prediction and similar to the single pixel attack in [30], we set the permissible range of mean noise by setting $\beta = 550$ for MNIST and $\beta = 75$ for ImageNet.

4.6. Future Work

Motivated by our proposed extension to Price's theorem [27] in Lemma (2), **(i)** a natural extension is to derive these expressions for larger networks beyond only (Affine, ReLU, Affine). This is expected to help minimize the effects of the two-stage linearization on the tightness of the derived moments. Moreover, **(ii)** another extension can be the use of network moments directly in the training process of a PL-DNN. Since the expressions of the mean and variance are differentiable w.r.t. the network weights, they can directly be added as regularizers (*e.g.* adding the expression of the output variance) to the standard loss. This is a more systematic and pragmatic alternative to the commonly used data augmentation-with-noise approach that requires the sampling of a very large number of noisy training samples, which slows down training and is inflexible when robustness to different noise levels is required.

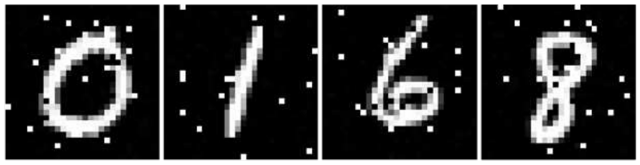
Figure 5. MNIST images with LeNet predicted labels 2, 4, 2, and 2 (from left to right), after adding noise generated by Eq (4)

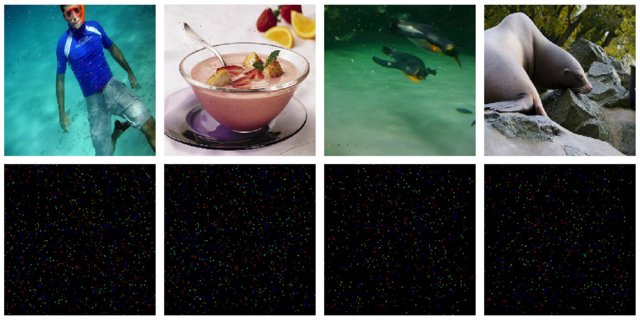
Figure 6. Shows in the top row the noisy images that fool AlexNet as generated by solving Eq (4). The second row shows the generated noise with $\alpha = 2\%$ of the total pixels in the image.

5. Conclusions

In this paper, we derive analytic expressions for the first and second moments of a small piecewise linear (PL) neural network in the form (Affine, ReLU, Affine) with a Gaussian input. Extensive experiments show that these expressions are tight under a reasonable two-stage linearization, particularly useful in analyzing DNN behaviour after exposure to input noise, and facilitate the systematic construction of adversarial input Gaussian attacks. These expressions can provide the community with useful tools to analyze pre-trained PL-DNNs and train them with noise robustness in mind.

Acknowledgments. This work was supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research. We thank Prof. Ahmed S. Salem for his review, recommendations and directions.

References

[1] G. An. The effects of adding noise during backpropagation training on a generalization performance. *Neural computation*, 8(3):643–674, 1996.

[2] A. Bakry, M. Elhoseiny, T. El-Gaaly, and A. Elgammal. Digging deep into the layers of cnns: In search of how cnns achieve view invariance. 2016.

[3] C. M. Bishop. Training with noise is equivalent to tikhonov regularization. *Training*, 7(1), 2008.

[4] D. Chicco, P. Sadowski, and P. Baldi. Deep autoencoder neural networks for gene ontology annotation predictions. In *Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 533–540. ACM, 2014.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[6] A. Dosovitskiy and T. Brox. Inverting visual representations with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4829–4837, 2016.

[7] V. Escorcia, F. C. Heilbron, J. C. Niebles, and B. Ghanem. Daps: Deep action proposals for action understanding. In *European Conference on Computer Vision*, pages 768–784. Springer, 2016.

[8] A. Fawzi and P. Frossard. Manitest: Are classifiers really invariant? In *British Machine Vision Conference (BMVC)*, number EPFL-CONF-210209, 2015.

[9] A. Fawzi and P. Frossard. Measuring the effect of nuisance variables on classifiers. In *British Machine Vision Conference (BMVC)*, number EPFL-CONF-220613, 2016.

[10] A. Fawzi, S.-M. Moosavi-Dezfooli, and P. Frossard. Robustness of classifiers: from adversarial to random noise. In *Advances in Neural Information Processing Systems*, pages 1632–1640, 2016.

[11] A. Fawzi, S. M. Moosavi Dezfooli, and P. Frossard. A geometric perspective on the robustness of deep networks. Technical report, Institute of Electrical and Electronics Engineers, 2017.

[12] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[13] Y. Grandvalet, S. Canu, and S. Boucheron. Noise injection: Theoretical prospects. *Neural Computation*, 9(5):1093–1108, 1997.

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

[15] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[17] Y. LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[18] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio. Object recognition with gradient-based learning. *Shape, contour and grouping in computer vision*, pages 823–823, 1999.

[19] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[20] Y. Luo and F. Yang. Deep learning with noise, 2014.

[21] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015.

[22] E. McMahon. An extension of price's theorem (corresp.). *IEEE Transactions on Information Theory*, 10(2):168–168, 1964.

[23] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. *arXiv preprint arXiv:1610.08401*, 2016.

[24] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016.

[25] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436, 2015.

[26] A. Papoulis. Probability, random variables, and stochastic processes. page 161, 1965.

[27] R. Price. A useful theorem for nonlinear devices having gaussian inputs. *IRE Transactions on Information Theory*, 4(2):69–72, 1958.

[28] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[29] S. Soatto and A. Chiuso. Visual representations: Defining properties and deep approximations. *arXiv preprint arXiv:1411.7676*, 2014.

[30] J. Su, D. V. Vargas, and S. Kouichi. One pixel attack for fooling deep neural networks. *arXiv preprint arXiv:1710.08864*, 2017.

[31] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. 2013.

[32] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[33] C. Vondrick, A. Khosla, T. Malisiewicz, and A. Torralba. Hoggles: Visualizing object detection features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1–8, 2013.