FEED-FORWARD PROPAGATION IN PROBABILISTIC NEURAL NETWORKS WITH CATEGORICAL AND MAX LAYERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Probabilistic Neural Networks take into account various sources of stochasticity: input noise, dropout, stochastic neurons, parameter uncertainties modeled as random variables. In this paper we revisit the feed-forward propagation method that allows one to estimate for each neuron its mean and variance w.r.t. mentioned sources of stochasticity. In contrast, standard NNs propagate only point estimates, discarding the uncertainty. Methods propagating also the variance have been proposed by several authors in different context. The presented view attempts to clarify the assumptions and derivation behind such methods, relate it to classical NNs and broaden the scope of its applicability. The main technical innovations are new posterior approximations for argmax and max-related transforms, that allows for applicability in networks with softmax and max-pooling layers as well as leaky ReLU activations. We evaluate the accuracy of the approximation and suggest a simple calibration. Applying the method to networks with dropout allows for faster training and gives improved test likelihoods without the need of sampling.

1 INTRODUCTION

Despite the massive success of Neural Networks (NNs) considered as deterministic predictors, there are many scenarios where a probabilistic treatment is highly desirable. One of the best known techniques to improve the generalization is *dropout* (Srivastava et al., 2014), which introduces multiplicative Bernoulli noise in the network. At test time, however, it is commonly approximated by substituting the mean value of the noise variables. Computing the expectation by Monte Carlo (MC) sampling instead leads to improved test likelihood and accuracy (Srivastava et al., 2014; Gal & Ghahramani, 2015) but is computationally expensive. A challenging problem in NNs is the sensitivity of the output to the perturbations of the input, in particular random and adversarial perturbations (Moosavi-Dezfooli et al., 2017; Fawzi et al., 2016; Rodner et al., 2016). In Fig. 1 we illustrate the point that the average of the network output under noisy input differs from propagating the clean input. It is therefore desirable to estimate the output uncertainty resulting from the uncertainty of the input. In classification networks, propagating the uncertainty of the input can impact the confidence of the classifier and its robustness as shown by Astudillo & da Silva Neto (2011). Ideally, we would like that a classifier is not overconfident when making errors, however such high confidences of wrong predictions are typically observed in NNs. Similarly, when predicting real values (*e.g.* optical flow estimation), it is desirable to estimate also confidences of such predictions. Taking into account uncertainties from input or dropout allows to predict output uncertainties well correlated with the test error (Kendall & Gal, 2017; Gast & Roth, 2018; Schoenholz et al., 2016). Another important problem is overfitting. Bayesian learning is a sound way of dealing with a finite training set: the parameters are considered as random variables and are determined up to an uncertainty implied by the training data. This uncertainty needs then to be propagated to predictions at the test-time.

The above scenarios motivate considering NNs with different sources of stochasticity as Bayesian networks, a class of directed probabilistic graphical models. We focus on the *inference problem* that consists in estimating the probability of hidden units and the outputs given the network input. While there exist elaborate inference methods for Bayesian networks (variational, mean field, Gibbs

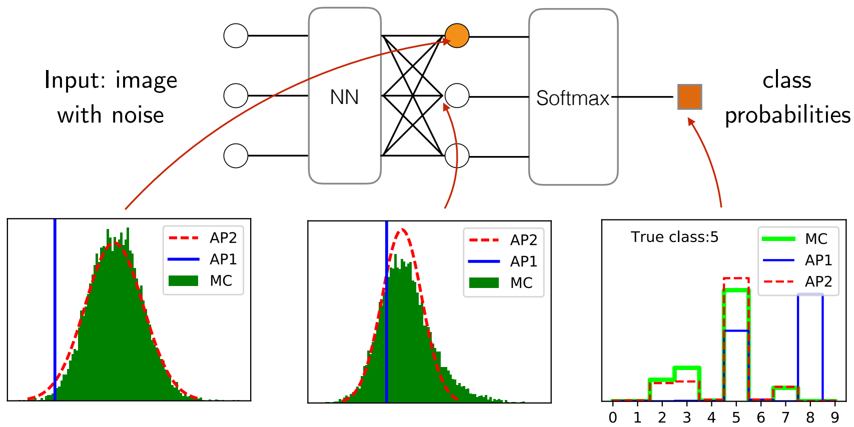

Figure 1: Illustrative example of propagating an input perturbed with Gaussian noise $\mathcal{N}(0, 0.1)$ through a fully trained LeNet. When the same image is perturbed with different samples of noise, we observe on the output empirical distributions shown as Monte Carlo (MC) histograms. Propagating the clean image results in the estimate denoted AP1 which may be away from the MC mean. Propagating means and variances results in a posterior Gaussian distribution denoted AP2. For the final class probabilities we approximate the expected value of the softmax. The methods AP1 and AP2 are formally defined in § 2. A quantitative evaluation of this experiment is given in § 5.

sampling, *etc*.), they are computationally demanding and can hardly be applied at the same scale as state-of-the-art NNs.

Contribution and Related Work We revisit feed-forward propagation methods that perform an approximate inference analytically by propagating means and variances of neurons through all layers of a NN, ensuring computational efficiency and differentiability. This type of propagation has been proposed by several authors under different names: *uncertainty propagation* (Astudillo & da Silva Neto, 2011) in a very limited setting with no learning, *fast dropout training* (Wang & Manning, 2013), *probabilistic backpropagation* (Hernández-Lobato & Adams, 2015) in the context of Bayesian learning, *assumed density filtering* Gast & Roth (2018). Perhaps the most general form is considered by Wang et al. (2016) and termed *natural parameter networks*. The *local reparametrization trick* (Kingma et al., 2015) can be viewed as application of the variance propagation method through one layer only and then sampling from the approximate posterior.

In these preceding works, for propagation through softmax, sampling or point-wise estimates were used while max-pooling was avoided. Ghosh et al. (2016) proposed an analytic approximation for softmax using two inequalities, but resorted to sampling noting that the approximation was not accurate. Gast & Roth (2018) introduced Dirichlet posterior to overcome the difficulty with softmax, however, the softmax is still used in the model internally. Furthermore, typically used expressions for ReLU activations involve differences of error functions and may be unstable.

We propose a latent variable view of probabilistic NNs that links them closer to their deterministic counterparts and allows us to develop better approximations. Our technical contribution includes the development of numerically suitable approximations for propagating means and variances through "multivariate" activation functions such as $\mathrm{softmax}$ for categorical variables and other $\max$-related non-linearities: $\max$-pooling and leaky ReLU. This makes the whole framework practically operational and applicable to a wider class of problems.

Experimentally, we verify the accuracy of the proposed propagation in approximating the true posterior and compare it to the standard propagation by NN, which has not been questioned before. This verification shows that the proposed scheme has better accuracy than standard propagation in all tested scenarios. We further demonstrate its potential utility in the end-to-end learning with dropout.

2 Probabilistic NNs and Feed-forward Expectation Propagation

In probabilistic NNs, all units are considered to be random variables. In a typical network, units are organized by layers. There are l layers of hidden random vectors X^k, $k = 1, \dots l$ and X^0 is the input layer. Each vector X^k has n_k components (layer units) denoted X^k_i. The network is modeled as a conditional *Bayesian network* (aka belief network, Neal (1992)) defined by the pdf

$$p(X^{1,\dots l} \mid X^0) = \textstyle\prod_{k=1}^{l} p(X^k \mid X^{k-1}). \tag{1}$$

We further assume that the conditional distribution $p(X^k \mid X^{k-1})$ factorizes and depends on a linear combination of the random vector X^{k-1}, $p(X^k \mid X^{k-1}) = \prod_{i=1}^{n_k} p(X^k_i \mid A^k_i)$, where $A^k_i = (W^k X^{k-1})_i$ are *activations*. We will denote values of r.v. X^k by x^k, so that the event $X^k = x^k$ can be unambiguously denoted just by x^k. Notice also that we consider biases of the units implicitly via an additional input fixed to value one. The posterior distribution of each layer $k > 0$, given the observations x^0, recurrently expresses as

$$p(X^k \mid x^0) = \mathbb{E}_{X^{k-1} \mid x^0}\big[p(X^k \mid X^{k-1})\big] = \int p(X^k \mid x^{k-1}) p(x^{k-1} \mid x^0)\, dx^{k-1}. \tag{2}$$

The posterior distribution of the last layer, $p(X^l \mid x^0)$ is the prediction of the model.

We now explain how the standard NNs with injected noises give rise to the Bayesian networks of the form (1). Consider a deterministic nonlinear mapping applied to a "noised" activation:

$$X^k = f(A^k - Z^k), \tag{3}$$

where $f \colon \mathbb{R} \to \mathbb{R}$ is applied component-wise and Z^k_i are independent real-valued random variables with a known distribution (such as the standard normal distribution). From representation (3) we can recover the conditional cdf of the belief network $F_{X^k \mid X^{k-1}}(u) = \mathbb{E}[\![f(W^k X^{k-1} - Z^k) \leq u \mid X^{k-1}]\!]$ and the respective conditional density.

Example 1. *Stochastic binary unit (Williams, 1992).* Let Y be a binary valued r.v. given by $Y = \Theta(A - Z)$, where Θ is the Heaviside step function and Z is noise with cdf F_Z. Then $\mathbb{P}(Y{=}1 \mid A) = F_Z(A)$. This is easily seen from

$$\mathbb{P}(Y{=}1 \mid A) = \mathbb{P}(\Theta(A - Z) = 1 \bigm| A) = \mathbb{P}(Z \leq A | A) = F_Z(A). \tag{4}$$

If, for instance, Z is distributed with standard logistic distribution, then $\mathbb{P}(Y{=}1 \mid A) = \mathcal{S}(A)$, where $\mathcal{S}$ is the *logistic sigmoid* function $\mathcal{S}(a) = (1 + e^{-a})^{-1}$. □

In general, the expectation (2) is intractable to compute and the resulting posterior can have a combinatorial number of modes. However, in many cases of interest it is suitable to approximate the posterior $p(X^k \mid x^0)$ for a given x^0 with a factorized distribution $q(X^k) = \prod_i q(X^k_i)$. We expect that in many recognition problems, given the input image, the hidden states and the final prediction are concentrated around some specific values (unlike in generative problems, where the posterior distributions are typically multi-modal). A similar factorized approximation is made for the activations. The exact shape of distributions $q(X^k_i)$ and $q(A^k_i)$ can be chosen appropriately depending on the unit type: *e.g.*, a Bernoulli distribution for binary X^k_i a Gaussian or Logistic distribution for real-valued activations A^k_i. We will rely on the fact that the mean and variance are sufficient statistics for such approximating distributions. Then, as long as we can calculate these sufficient statistics for the layer of interest, the exact shape of distributions for the intermediate outputs need not be assumed.

The information-theoretic optimal factorized approximation to the posterior $p(X^k \mid x^0)$, minimizing the forward KL divergence $KL(p(X^k \mid x^0) \| q(X^k))$, is given by marginals $\prod_i p(X^k_i \mid x^0)$. Furthermore, in the case when $q(X^k_i)$ is from to the exponential family, the optimal approximation is given by matching the moments of $q(X^k_i)$ to $p(X^k_i \mid x^0)$. The factorized approximation then can be computed layer-by-layer, assuming that the preceding layer was already approximated. Substituting $q(X^{k-1})$ for $p(X^{k-1} \mid x^0)$ in (2) results in the procedure

$$q(X^k_i) = \mathbb{E}_{q(X^{k-1})}\big[p(X^k_i \mid X^{k-1})\big] = \int p(X^k_i \mid x^{k-1}) \prod_i q(x^{k-1}_i)\, dx^{k-1}. \tag{5}$$

Thus we need to propagate the factorized approximation layer-by-layer, by the marginalization update (5) until we get the approximate posterior output $q(X^l)$. This method is closely related to the *assumed density filtering* (see Minka, 2001), in which, in the context of learning, one chooses a family of distributions that is easy to work with and "projects" the true posterior onto the family after each measurement update. Here, the projection takes place after propagating each layer for the purpose of the inference.

3 Propagation in Basic Layers

We now consider a single layer at a time and detail how (5) is computed (approximately) for a layer consisting of a linear mapping $A = w^{\mathsf{T}} X$ (scalar output, for clarity) and a non-linear noisy activation $Y = f(A - Z)$.

Linear Mapping Activation A in a typical deep network is a sum of hundreds of stochastic inputs X (from the previous layer). This justifies the assumption that $A - Z$ (where Z is a smoothly distributed injected noise) can be approximated by a uni-modal distribution fully specified by mean and variance as *e.g.* normal or logistic distribution[1]. Knowing the statistics of Z, it is therefore sufficient to estimate the mean and the variance of the activation A given by

$$\mu' = \mathbb{E}[A] = w^{\mathsf{T}}\mathbb{E}[X] = w^{\mathsf{T}}\mu, \tag{6a}$$

$$\sigma'^2 = \textstyle\sum_{ij} w_i w_j \operatorname{Cov}[X]_{ij} \approx \sum_i w_i^2 \sigma_i^2, \tag{6b}$$

where μ is the mean and $\operatorname{Cov}[X]$ is the covariance matrix of X. The approximation of the covariance matrix by its diagonal is implied by the factorization assumption for the activations A.

Nonlinear Coordinate-wise Mappings Let A be a scalar r.v. with statistics μ, σ^2 and let $Y = f(A-Z)$ with independent noise Z. Assuming that $\widetilde{A} = A-Z$ is distributed normally or logistically with statistics $\tilde{\mu}, \tilde{\sigma}^2$, we can approximate the expectation and variance of $Y = f(\widetilde{A})$

$$\mu'_i = \mathbb{E}_{q(\widetilde{A})}[f(\widetilde{A})], \qquad \sigma'^2_i = \mathbb{E}_{q(\widetilde{A})}[f^2(\widetilde{A})] - \mu'^2_i \tag{7}$$

by analytic expressions for most of the commonly used non-linearities. For binary variables, occurring in networks with Heaviside nonlinearities, the distribution $q(Y)$ is fully described by one parameter $\mu_i = \mathbb{E}[Y]$, and the propagation rule (5) becomes

$$\mu'_i = \mathbb{E}_{q(A)}\big[p(Y{=}1 \,|\, A^k)\big], \qquad \sigma'^2_i = \mu'_i(1-\mu'_i), \tag{8}$$

where the variance is dependent but will be needed in propagation through other layers.

Example 2. *Heaviside Nonlinearity with Noise.* Consider the model $Y = \Theta(A - Z)$, where Z is logistic noise. The statistics of $\widetilde{A} = A - Z$ are given by $\tilde{\mu} = \mu$ and $\tilde{\sigma}^2 = \sigma^2 + \sigma_S^2$, where $\sigma_S^2 = \pi^2/3$ is the variance of Z. Assuming noisy activations $\widetilde{A}$ to have logistic distribution, we obtain the mean of Y as:

$$\mu' = \mathbb{E}[\Theta(\tilde{A})] = \mathbb{P}(\widetilde{A} \geq 0) = \mathbb{P}\Big(\frac{\widetilde{A}-\tilde{\mu}}{\tilde{\sigma}/\sigma_S} \geq \frac{-\tilde{\mu}}{\tilde{\sigma}/\sigma_S}\Big) \doteq \mathcal{S}\Big(\frac{\tilde{\mu}}{\tilde{\sigma}/\sigma_S}\Big) = \mathcal{S}\Big(\frac{\mu}{\sqrt{\sigma^2/\sigma_S^2+1}}\Big), \tag{9}$$

where the dotted equality is due to that $-(\tilde{A}-\tilde{\mu})\frac{\sigma_S}{\tilde{\sigma}}$ has standard logistic distribution and that the sigmoid function $\mathcal{S}$ is its cdf. The variance of Y is expressed as in (8).

Example 3. *Rectified Linear Unit (*ReLU*)* Assuming the activation A to be normally distributed, the mean of $Y = \max(0, A)$ expresses as $\mu' = \int_{-\infty}^{\infty} \max(0,a)p(a)\mathrm{d}a = \int_0^{\infty} ap(a)\mathrm{d}a = \mu\Phi(\mu/\sigma) + \sigma\phi(\mu/\sigma)$, *i.e.*, expresses analytically using the pdf ϕ and cdf Φ of the standard normal distribution. The variance can be expressed as well. These expressions, used by Frey & Hinton (1999); Hernández-Lobato & Adams (2015) rely on function Φ, which has limited numerical accuracy and may lead to negative output variances. In § 4.4 we propose an approximation for leaky ReLU, which is numerically stable and is suitable for ReLU as well.

Fig. 2 shows the approximations for propagation through Heaviside, ReLU and leaky ReLU nonlinearities. Note that all expectations over a smoothly distributed A result in smooth propagation functions regardless the smoothness (or lack thereof) of the original function.

[1]Note, the prior works assumes that A alone approaches Gaussian, which is a stronger assumption, considering for example binary input X.

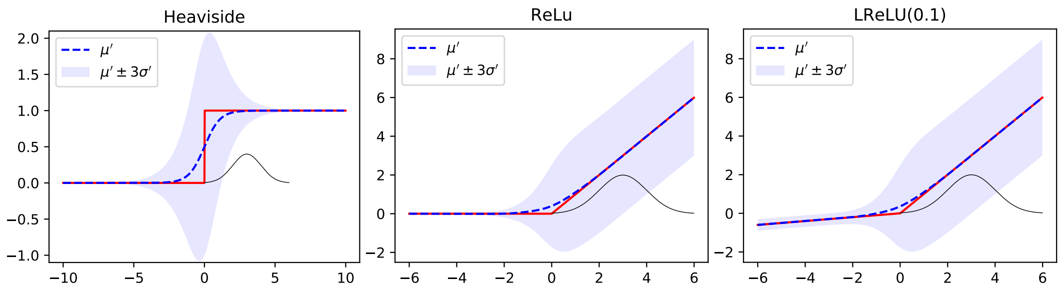

Figure 2: Propagation for the Heaviside function: $Y = [\![A{\geq}0]\!]$, ReLU: $Y = \max(0, A)$ and leaky ReLU: $Y = \max(\alpha A, A)$. Red: activation function. Black: an exemplary input distribution with mean $\mu = 3$, variance $\sigma^2 = 1$ shown with support $\mu \pm 3\sigma$. Dashed blue: the approximate mean μ' of the output versus the input mean μ. The variance of the output is shown as blue shaded area $\mu' \pm 3\sigma'$.

Summarizing, we can represent the approximate inference in networks with binary and continuous variables as a feed-forward moment propagation: given the approximate moments of $X^{k-1} \,|\, x^0$, the moments of $X^k_i \,|\, x^0$ are estimated via (8), (7) ignoring dependencies between $X^{k-1}_j \,|\, x^0$ on each step (as implied by the factorized approximation).

AP1 and AP2 The standard NN can be viewed as a further simplification of the proposed method: it makes the same factorization assumption but does not compute variances of the activations (6b) and propagates only the means. Consequently, a zero variance is assumed in propagation through non-linearities. In this case the expected values of mappings such as $\Theta(A)$ and $\mathrm{ReLU}(A)$ are just these functions evaluated at μ. For injected noise models we obtain smoothed versions: *e.g.*, substituting $\sigma = 0$ in the noisy Heaviside function (9) recovers the standard sigmoid function. We thus can view standard NNs as making a simpler from of factorized inference in the same Bayesian NN model. We designate this simplification (in figures and experiments) by AP1 and the method using variances by AP2 ("AP" stands for approximation).

4 Propagation in Categorical and Max Layers

In this section we present our main technical contribution: propagation rules for argmax, $\mathrm{softmax}$ and $\max$ mappings, that are non-linear and multivariate. Similar to how sigmoid function is obtained as the expectation of the Heaviside function with injected noise in Example 2, we observe that $\mathrm{softmax}$ layer is the expectation of argmax with injected noise. It will follow that the standard NN with $\mathrm{softmax}$ layer can be viewed as AP1 approximation of argmax layer with injected noise. We propose a new approximation for the argmax posterior probability that takes into account uncertainty (variances) of the activations and enables propagation through argmax and $\mathrm{softmax}$ layers. Next, we observe that the maximum of several variables (used in max-pooling) can be expressed through argmax. This gives a new one-shot approximation of the expected maximum using argmax probabilities. Finally, we consider the case of leaky ReLU, which is a maximum of two correlated variables. The proposed approximations are relatively easy to compute and differentiable, which facilitates their usage in NNs.

4.1 Argmax and Softmax

The $\mathrm{softmax}$ function, most commonly used to model a categorical distribution, ubiquitous in classification, is defined as $p(Y{=}y|x) = e^{x_y} / \sum_k e^{x_k}$, where y is the class index. We explore the following latent variable representation known in the theory of discrete choice: $p(Y{=}y|x) = \mathbb{E}[\overline{Y}_y]$, where $\overline{Y} \in \{0, 1\}^n$ is the indicator of the noisy argmax: $\overline{Y}_y = [\![\mathrm{argmax}_k(X_k + \Gamma_k) = y]\!]$ and Γ_k follows the standard Gumbel distribution. Standard NN implements the AP1 approximation of this latent model: conditioned on $X = x$, the expectation over latent noises Γ is the $\mathrm{softmax}(x)$.

For the AP2 approximation we need to compute the expectation w.r.t. both: X and Γ, or, what is the same, to compute the expectation of $\mathrm{softmax}(X)$ over X. This task is difficult, particularly because variances of X_i may differ across components. First, we derive an approximation for the expectation of argmax indicator without injected noise:

$$\overline{Y}_y = [\![\operatorname*{argmax}_k X_k = y]\!]. \tag{10}$$

The injected noise case can be treated by simply increasing the variance of each X_i by the variance of standard Gumbel distribution.

Let X_k , $k = 1, \dots, n$ be independent, with mean μ_k and variance σ_k^2. We need to estimate

$$\mathbb{E}[\overline{Y}_y] = \mathbb{E}_X [\![X_y - X_k \geq 0 \ \forall k \neq y]\!], \tag{11}$$

The vector U with components $U_k = X_y - X_k$ for $k \neq y$ is from $\mathbb{R}^{n-1}$ with component means $\tilde{\mu}_k = \mu_y - \mu_k$ and component variances $\tilde{\sigma}_k^2 = \sigma_y^2 + \sigma_k^2$. Note the components of U are not independent.

We approximate the distribution of U by the multivariate logistic distribution defined by Malik & Abraham (1973). This choice is motivated by the extrapolation of the case with two input variables. The approximation is made by shifting and rescaling the distribution in order to match the means and marginal variances. The marginal distributions of standard multivariate logistic distribution are standard logistic with zero mean and variance σ_S. Thus the approximation assumes that $(U_k - \tilde{\mu}_k)\sigma_{\mathcal{S}}/\tilde{\sigma}_k$ is standard $(n-1)$-variate logistic with the cdf given by $\mathcal{S}_{n-1}(u) = \frac{1}{1+\sum_k e^{-u_k}}$ (Malik & Abraham, 1973, eq. 2.5). It allows us to evaluate the necessary probability:

$$q(y) = \mathbb{E}[\overline{Y}_y] = \mathbb{P}(U \geq 0) = \mathbb{P}\Big(\frac{U_k - \tilde{\mu}_k}{\tilde{\sigma}_k/\sigma_{\mathcal{S}}} \geq \frac{-\tilde{\mu}_k}{\tilde{\sigma}_k/\sigma_{\mathcal{S}}} \ \forall k \neq y\Big) = \mathcal{S}_{n-1}\Big(\frac{-\tilde{\mu}_k}{\tilde{\sigma}_k/\sigma_{\mathcal{S}}}\Big). \tag{12}$$

Expanding $\tilde{\mu}$, $\tilde{\sigma}^2$ and noting that $\mu_k - \mu_y = 0$ for $y = k$, we obtain the approximation

$$q(y) = \Big(\sum_k \exp\Big\{\frac{\mu_k - \mu_y}{\sqrt{(\sigma_k^2 + \sigma_y^2)/\sigma_{\mathcal{S}}^2}}\Big\}\Big)^{-1}. \tag{13}$$

This approximation has linear memory complexity but requires quadratic time in the number of inputs, which may be prohibitive for applications in NNs. We can simplify it further as follows. The expression (13) simplifies when we can approximate

$$\frac{\mu_k - \mu_y}{\sqrt{(\sigma_k^2 + \sigma_y^2)/\sigma_S^2}} \approx a_k - a_y \tag{14}$$

with some choice of a_k for all k. In this case we obtain $q(y) = (\operatorname{softmax}(a))_y$. We therefore propose the approximation

$$q = \operatorname{softmax}(a) \ \text{ with } \ a_k = \mu_k / \sqrt{\big(\sigma_k^2 + \frac{n\bar{\sigma}^2 - \sigma_k^2}{n-1}\big)/\sigma_S^2}, \tag{15}$$

where $\bar{\sigma}^2 = \frac{1}{n}\sum_k \sigma_k^2$ is the average variance.

Importantly, the approximation is consistent with the already obtained results for the following special cases. In the case of two input variables, for the simplified approximation with a_k set as (15) we have $a_k = \mu_k / \sqrt{(\sigma_1^2 + \sigma_2^2)/\sigma_{\mathcal{S}}^2}$, i.e. (14) holds as equality, and we obtain

$$q(y{=}1) = \operatorname{softmax}(a_1, a_2)_1 = \frac{e^{a_1}}{e^{a_1} + e^{a_2}} = \frac{1}{1 + e^{a_2 - a_1}} = \mathcal{S}(a_2 - a_1) = \mathcal{S}\Big(\frac{\tilde{\mu}}{\tilde{\sigma}/\sigma_S}\Big), \tag{16}$$

which matches the approximation of the Heaviside posterior with input $X_1 - X_2$ with mean $\tilde{\mu}$ and variance $\tilde{\sigma}^2$. As a consequence expectation of $\operatorname{softmax}$ (argmax indicator with injected noise) matches the expectation of sigmoid (Heaviside function with injected noise) given by (9).

In the case when all variances σ_k^2 are equal: $\sigma_k = \sigma$, the approximation (15) results in

$$q = \operatorname{softmax}\Big(\frac{\mu_k}{\sqrt{2}\sigma/\sigma_S}\Big). \tag{17}$$

More specifically, when $X_k = \mu_k + \Gamma_k$, where Γ_k is standard Gumbel (with variance $\pi^2/6 = \sigma_S^2/2$) we obtain that $q = \operatorname{softmax}(\mu_k)$, *i.e.* recover the exact expectation of noisy argmax with deterministic inputs used by AP1.

4.2 Maximum of Two Variables

Let us consider the function $\max(X_1, X_2)$, which is important for leaky ReLU and maxOut. In this case, exact expressions for the moments for the maximum of two Gaussian random variables X_1, X_2 are known (Nadarajah & Kotz, 2008). Denoting $s = (\sigma_1^2 + \sigma_2^2 - 2\,\mathrm{Cov}[X_1, X_2])^{\frac{1}{2}}$ and $a = (\mu_1 - \mu_2)/s$, the mean and variance of $\max(X_1, X_2)$ can be expressed as:

$$\mu' = \mu_1 \Phi(a) + \mu_2 \Phi(-a) + s\phi(a), \tag{18a}$$

$$\sigma'^2 = (\sigma_1^2 + \mu_1^2)\Phi(x) + (\sigma_2^2 + \mu_2^2)\Phi(-a) + (\mu_1 + \mu_2)s\phi(a) - \mu'^2. \tag{18b}$$

These expressions involving the normal cdf Φ, will not be used directly. We simplify them in the case of leaky ReLU and use as a reference for maximum of multiple variables. The variance can be further expressed as

$$\sigma'^2 = \sigma_1^2 \Phi(a) + \sigma_2^2 \Phi(-a) + s^2(a^2\Phi(a) + a\phi(a) - (a\Phi(a) + \phi(a))^2). \tag{19}$$

We observe that the function of one variable $a^2\Phi(a) + a\phi(a) - (a\Phi(a) + \phi(a))^2$ is always negative, quickly vanishes with $|a|$ increasing and is above -0.16. By neglecting it, we obtain a rather tight upper bound $\sigma'^2 \leq \sigma_1^2\Phi(a) + \sigma_2^2(1 - \Phi(a))$. Note that $\Phi(a)$, which serves as interpolating coefficient between σ_1^2 and σ_2^2, is precisely the probability of the event $X_1 > X_2$. This suggests the idea of estimating mean and variance of $\max$ from the argmax probabilities in the multivariate case.

4.3 Maximum of Several Variables

Let X_k , $k = 1, \dots, n$ be independent, with mean μ_k and variance σ_k^2. The moments of the maximum $Y = \max_k X_k$, assuming the distributions of X_k are known, can be computed by integration with the CDF of Y (Ross, 2010) given by $F_Y(y) = \mathbb{P}(X_k \leq y \;\; \forall k) = \prod_k F_{X_k}(y)$. However, this requires numerical 1D integration. We seek a simpler approximation. One option is to compose the maximum of $n > 2$ variables hierarchically using maximum of two variables § 4.2 and assume that the intermediate results are distributed normally.

We propose a new non-trivial one-shot approximations for the mean and variance assuming that the argmax probabilities $q_k = \mathbb{P}(X_k \geq X_j \; \forall j)$ are already estimated. The derivation of these approximations and proofs of their accuracy are given in § A.

Proposition 1. Assuming X_k are logistic (μ_k, σ_k^2), the mean of $Y = \max_k X_k$ can be approximated (upper bounded) by

$$\mu' \approx \sum_k q_k \hat{\mu}_k, \quad \text{where} \;\; \hat{\mu}_k = \mu_k + \frac{\sigma_k}{q_k \sigma_S} H(q_k), \tag{20}$$

where $H(q_k)$ is the entropy of the Bernoulli distribution with probabilities q_k. The variance of Y can be approximated as

$$\sigma'^2 \approx \sum_k \sigma_k^2 \mathcal{S}(a + b\mathcal{S}^{-1}(q_k)) + \sum_k q_k(\hat{\mu}_k - \mu')^2, \tag{21}$$

where $a = -1.33751$ and $b = 0.886763$ are coefficients originating from a Taylor expansion.

Notice the similarity to the expressions (18a) and (19) (identifying q_1, q_2 with argmax probabilities $\Phi(a)$,$\Phi(-a)$, resp.). Also notice that the entropy is non-negative, and thus increases μ' when the argmax is ambiguous, as expected in the extreme value theory.

4.4 Leaky ReLU

`LReLU` is a popular max-related function defined as: $Y = \max(\alpha X, X)$. We use the exact expressions for the case of two correlated normal variables (18a) and (19). Assume that $\alpha < 1$, let $X_2 = \alpha X_1$ and denote $\mu = \mu_1$ and $\sigma^2 = \sigma_1^2$. Then $\mu_2 = \alpha\mu$, $\sigma_2^2 = \alpha^2\sigma^2$ and $\mathrm{Cov}[X_1, X_2] = \mathrm{Cov}[X_1, \alpha X_1] = \alpha\sigma^2$. We have $s = \sigma(1 - \alpha)$ and $a = (\mu_1 - \mu_2)/s = \mu(1 - \alpha)/s = \mu/\sigma$. The mean μ' expresses as

$$\mu' = \mu(\alpha + (1 - \alpha)\Phi(a)) + \sigma(1 - \alpha)\phi(a). \tag{22}$$

The variance σ'^2 expresses as

$$\sigma^2\Big(\Phi(a) + \alpha^2(1-\Phi(a)) + (1-\alpha)^2\big(a^2\Phi(a) + a\phi(a) - (a\Phi(a)+\phi(a))^2\big)\Big) \tag{23}$$

$$= \sigma^2(\alpha^2 + 2\alpha(1-\alpha)\Phi(a) + (1-\alpha)^2\mathcal{R}(a)), \tag{24}$$

where $\mathcal{R}(a) = a\phi(a) + (a^2+1)\Phi(a) - (a\Phi(a)+\phi(a))^2$ is a sigmoid-shape function of one variable. In practice we approximate σ'^2 with the simpler function

$$\sigma'^2 \approx \sigma^2(\alpha^2 + (1-\alpha^2)\mathcal{S}(a/t)), \tag{25}$$

where $t = 0.3758$ is set by fitting the approximation. The approximation is shown in Fig. 2.

5 Experiments

In the experiments we evaluate the accuracy of the proposed approximation and compare it to the standard propagation. We also test the method in the end-to-end learning and show that with a simple calibration it achieves better test likelihoods than the state-of-the-art. Full details of the implementation, training protocols, used datasets and networks are given in § B. The running time of AP2 is 2× more for a forward pass and 2-3× more for a forward-backward pass than that of AP1.

5.1 Approximation Accuracy

We conduct two experiments: how well the proposed method approximates the real posterior of neurons, w.r.t. noise in the network input and w.r.t. dropout. The first case (illustrated in Fig. 1) is studied on the LeNet5 model of Lecun et al. (2001), a 5-layer net with max pooling detailed in § B.4, trained on MNIST dataset using standard methods. We set LReLU activations with $\alpha = 0.01$ to test the proposed approximations. We estimate the ground truth statistics μ^*, σ^* of all neurons by the Monte Carlo (MC) method: drawing 1000 samples of noise per input image and collecting sample-based statistics for each neuron. Then we apply AP1 to compute μ_1 and AP2 to compute μ_2 and σ_2 for each unit from the clean input and known noise variance σ_0. The error measure of the means ε_μ is the average $|\mu - \mu^*|$ relative to the average σ^*. The averages are taken over all units in the layer and over input images. The error of the standard deviation ε_σ is the geometric mean of σ/σ^*, representing the error as a factor from the true value (*e.g.*, 1.0 is exact, 0.9 is under-estimating and 1.1 is over-estimating). Table 1 shows average errors per layer and points the main observation: that AP2 is more accurate than AP1 but both methods suffer from the factorization assumption. The variance computed by AP2 provides a good estimate and the estimated categorical distribution by propagating the variance through softmax is much closer to the MC estimate.

Next, we study a widely used ALL-CNN network § B.4 by Springenberg et al. (2015) trained with standard dropout on CIFAR-10. Bernoulli dropout noise with dropout rate 0.2 is applied after each activation. The accuracies of estimated statistics w.r.t. dropout noises are shown in Table 2. Here, each layer receives uncertainty propagated from preceding layers, but also new noises are mixed-in in each layer, which works in favor of the factorization assumption. The results are shown in Table 2. Observe that GT noise variance σ^* changes significantly across layers, up to 1-2 orders and AP2 gives a useful estimate. Furthermore, having estimated the average factors suggests a simple calibration.

Calibration We divide the variance in the last layer by the average factor σ/σ^* estimated on the validation set. With this method, denoted AP2 calibrated, we get significantly better test likelihoods in the end-to-end learning experiment.

5.2 Analytic Normalization

The AP2 method can be used to approximate neuron statistics w.r.t. the input chosen at random from the training dataset as was proposed by Shekhovtsov & Flach (2018). Instead of propagating sample instances, the method takes the dataset statistics $(\mu^0, (\sigma^0)^2)$ and propagates them once through all network layers, averaging over spatial dimensions. The obtained neuron mean and variance are then used to normalize the output the same way as in batch normalization (Ioffe & Szegedy, 2015). This normalization leads to a better conditioned initialization and training and is batch-independent. We

	Conv	LReLU	MaxPool	Conv	LReLU	MaxPool	FC	LReLU	FC	LReLU	FC	Softmax
	Noisy input $\mathcal{N}(0, 10^{-4})$											
σ^*	0.03	0.02	0.02	0.06	0.03	0.03	0.09	0.05	0.10	0.05	0.11	
ε_{μ_1}	0.02	0.19	0.37	0.84	0.43	0.52	1.20	0.66	1.16	0.62	1.25	KL 3.5e-4
ε_{μ_2}	0.02	0.02	0.13	0.29	0.13	0.17	0.37	0.21	0.36	0.20	0.39	KL 3.3e-5
ε_{σ_2}	1.00	1.05	1.25	1.06	1.06	1.12	1.09	1.10	1.03	1.04	0.96	
	Noisy input $\mathcal{N}(0, 0.01)$											
σ^*	0.3	0.16	0.20	0.58	0.24	0.27	0.79	0.47	0.86	0.42	0.92	
ε_{μ_1}	0.02	0.24	0.53	1.46	0.58	0.70	1.44	0.85	1.40	0.79	1.57	KL 0.36
ε_{μ_2}	0.02	0.02	0.21	0.65	0.21	0.31	0.61	0.37	0.67	0.34	0.72	KL 0.05
ε_{σ_2}	1.00	1.10	1.15	1.17	1.22	1.42	1.37	1.59	1.31	1.47	1.23	

Table 1: Accuracy of approximation of mean and variance statistics for each layer in a fully trained LeNet5 (MNIST) tested with noisy input. Observe the following: MC variance σ^* is growing significantly from the input to the output; both AP1 and AP2 have a significant drop of accuracy at linear (fc and conv) layers, due to factorized approximation assumption; AP2 approximation of the standard deviation is within a factor close to one, and makes a meaningful estimate, although degrading with depth; AP2 approximation of the mean is more accurate than AP1; the KL divergence from the MC class posterior is improved with AP2.

	C	A	C	A	C	A	C	A	C	A	C	A	C	A	C	A	C	P	Softmax
σ^*	0	0.26	0.31	0.46	0.86	0.77	1.1	0.78	1.7	0.97	2.2	1.3	1.5	0.89	2	0.74	16	2.8	
ε_{μ_1}	-	0.01	0.02	0.03	0.07	0.06	0.17	0.09	0.19	0.10	0.25	0.11	0.22	0.11	0.21	0.12	0.17	0.38	KL 0.11
ε_{μ_2}	-	0.01	0.02	0.01	0.02	0.02	0.05	0.02	0.06	0.03	0.07	0.04	0.08	0.04	0.09	0.04	0.05	0.14	KL 0.04
ε_{σ_2}	-	1.00	1.00	1.02	0.88	0.89	0.90	0.95	0.84	0.87	0.77	0.77	0.82	0.85	0.88	0.92	0.69	0.45	

Table 2: Accuracy of approximation of mean and variance statistics for each layer in All-CNN (CIFAR-10) trained and tested with dropout. The table shows accuracies after all layers (C-convolution, A-activation, P-average pooling) and the final KL divergence. A similar effect to propagating input noise is observed: the MC variance σ^* grows with depth; a significant drop of accuracy is observed in conv and pooling layers which exploit the independence assumption.

verify the efficiency of this method for a network that includes the proposed approximations for LReLU and max pooling layers in § B.5 and use it in the end-to-end learning experiment below.

5.3 End-to-end learning with Analytic Dropout

In this experiment we approximate the dropout analytically at training time similar to Wang & Manning (2013) but including the new approximations for LReLU and softmax layers. We compare training All-CNN network on CIFAR-10 without dropout, with standard dropout (Srivastava et al., 2014) and analytic (AP2) dropout. All three cases use exactly the same initialization, AP2 normalization as discussed above and the same learning setup. Only the learning rate is optimized individually per method § B.3. The dropout layers with dropout rate 0.2 are applied after every activation and there is no input dropout. Fig. 3 shows the progress of the three methods. The analytic dropout is efficient as a regularizer (reduces overfitting in the validation likelihood), is non-stochastic and progresses faster than standard stochastic dropout. While latter slows the training down due to increased stochasticity of the gradient, the analytic dropout smoothes the loss function and speeds the training up. This is especially visible on the training loss plot Fig. B.3. Furthermore, analytic dropout can be applied as the test-time inference method in a network trained with any variant of dropout. Table 3 shows that AP2, calibrated as proposed above, achieves the best test likelihood, significantly improving SOTA results for this network. Some additional results are given in § B.7. Differently from Wang & Manning (2013), we find that when trained with standard dropout, all test methods achieve approximately the same accuracy and only differ in likelihoods. We believe this is due to the deep CNN in our case that achieves 100% training accuracy.

We also attempted comparison with other approaches. Gaussian dropout (Srivastava et al., 2014) performed similarly to standard Bernoulli dropout. Variational dropout Kingma et al. (2015) in our implementation for convolutional networks has diverged or has not reached the accuracy of the baseline without dropout (we tried correlated and uncorrelated versions with or without local reparametrization trick and with different KL divergence factors $1, 0.1, 0.01, 0.001$).

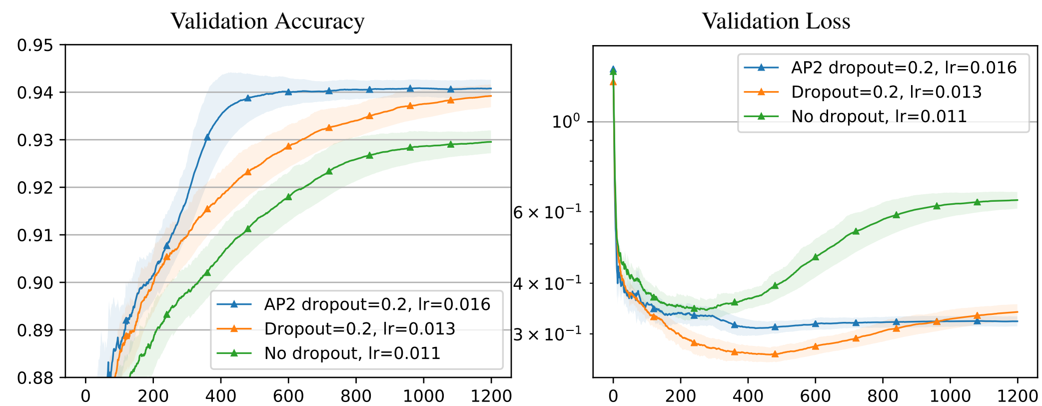

Figure 3: Comparison of analytic AP2 dropout with baselines. All methods use AP2 normalization during training. Analytic dropout converges to similar values of stochastic dropout and is faster per iteration. Both methods are efficient in preventing overfitting as seen in the right plot.

SOTA results (Gast & Roth, 2018)		
Method	NLL	Acc.
Dropout MC-30	0.327	90.88
ProbOut	0.37	91.9

Standard dropout		
Test method	NLL	Acc.
AP1	0.434	0.938
AP2	0.311	0.936
AP2 calibrated	0.214	0.937
MC-10	0.264	0.935
MC-100	0.217	0.937
MC-1000	0.210	0.937

Analytic dropout		
Test method	NLL	Acc.
AP1	1.86	**0.940**
AP2	0.363	**0.940**
AP2 calibrated	**0.194**	**0.940**
MC-10	0.546	0.919
MC-100	0.281	0.925
MC-1000	0.243	0.926

Table 3: Results for All-CNN on CIFAR-10 *test* set: negative log likelihood (NLL) and accuracy. *Left:* state of the art results for this network (Gast & Roth, 2018, table 3). *Middle:* All-CNN trained with standard dropout (our learning schedule and analytic normalization) evaluated using different test-time methods. Observe that "AP2 calibrated" well approximates dropout: the test likelihood is better than MC-100. *Right:* All-CNN trained with analytic dropout (same schedule and normalization). Observe that "AP2 calibrated" achieves the best likelihood and accuracy.

6 Conclusion

We have revisited the method for approximate inference in probabilistic neural networks that takes into account all sources of stochasticity analytically. The latent variable interpretation allows a transparent interpretation of standard propagation in NNs as the simplest approximation and the development of variance propagating approximations. We proposed new approximations to LReLU $\max$ and argmax functions. This allows analytic propagation in max pooling layers and softmax layer.

We measured the quality of the approximation of posterior. The accuracy is improved compared to standard propagation and is sufficient for several use cases such as estimating statistics over the dataset (normalization) and dropout training, where we report improved test likelihoods. We identified that the weak point of the approximation is the factorization assumption. While modeling correlations is possible (*e.g.* Rezende & Mohamed, 2015), it is also more expensive and we showed that a calibration of the cheap methods can give a significant improvement and is a direction for further research. Except as a final layer, argmax and $\operatorname{softmax}$ may occur also inside the network, in models such as capsules (Sabour et al., 2017) or multiple hypothesis (Ilg et al., 2018), *etc.* Further applications of the developed technique may include generative and semi-supervised learning and Bayesian model estimation.

References

Ramn Fernndez Astudillo and Joo Paulo da Silva Neto. Propagation of uncertainty through multi-layer perceptrons for robust automatic speech recognition. In *INTERSPEECH*, 2011.

Anirban DasGupta, S.N. Lahiri, and Jordan Stoyanov. Sharp fixed n bounds and asymptotic expansions for the mean and the median of a Gaussian sample maximum, and applications to the DonohoJin model. *Statistical Methodology*, 20:40–62, 2014.

Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Robustness of classifiers: from adversarial to random noise. In *NIPS*, pp. 1632–1640. 2016.

Brendan J. Frey and Geoffrey E. Hinton. Variational learning in nonlinear Gaussian belief networks. *Neural Comput.*, 11(1):193–213, January 1999.

Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with Bernoulli approximate variational inference. *arXiv:1506.02158*, 2015.

Jochen Gast and Stefan Roth. Lightweight probabilistic deep networks. *CoRR*, abs/1805.11327, 2018.

Soumya Ghosh, Francesco Maria Delle Fave, and Jonathan S. Yedidia. Assumed density filtering methods for learning Bayesian neural networks. pp. 1589–1595, 2016.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. 2010.

José Miguel Hernández-Lobato and Ryan P. Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *ICML*, pp. 1861–1869, 2015.

Eddy Ilg, Ozgun Cicek, Silvio Galesso, Aaron Klein, Osama Makansi, Frank Hutter, and Thomas Brox. Uncertainty estimates and multi-hypotheses networks for optical flow. In *ECCV*, 2018.

Sergey Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. *CoRR*, abs/1702.03275, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, volume 37, pp. 448–456, 2015.

Alex Kendall and Yarin Gal. What uncertainties do we need in Bayesian deep learning for computer vision? In *NIPS*, 2017.

Diederik P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *NIPS*, pp. 2575–2583. 2015.

Yann Lecun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Intelligent signal processing*, pp. 306–351, 2001.

Henrick J. Malik and Bovas Abraham. Multivariate logistic distributions. *The Annals of Statistics*, 1(3):588–590, 1973.

Thomas P. Minka. Expectation propagation for approximate Bayesian inference. In *Uncertainty in Artificial Intelligence*, pp. 362–369, 2001.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *CVPR*, July 2017.

Saralees Nadarajah and Samuel Kotz. Exact distribution of the max/min of two Gaussian random variables. *IEEE Trans. VLSI Syst.*, 16(2):210–212, 2008.

Radford M. Neal. Connectionist learning of belief networks. *Artif. Intell.*, 56(1):71–113, July 1992.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *ICML*, pp. 1530–1538, 2015.

Erik Rodner, Marcel Simon, Bob Fisher, and Joachim Denzler. Fine-grained recognition in the noisy wild: Sensitivity analysis of convolutional neural networks approaches. In *BMVC*, 2016.

Andrew M. Ross. Computing bounds on the expected maximum of correlated normal variables. *Methodology and Computing in Applied Probability*, 12(1):111–138, Mar 2010.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *NIPS*, pp. 3856–3866. 2017.

Samuel S. Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. *CoRR*, abs/1611.01232, 2016.

Alexander Shekhovtsov and Boris Flach. Normalization of neural networks using analytic variance propagation. In *Computer Vision Winter Workshop*, pp. 45–53, 2018.

J.T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. In *ICLR (workshop track)*, 2015.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15:1929–1958, 2014.

Hao Wang, Xingjian SHI, and Dit-Yan Yeung. Natural-parameter networks: A class of probabilistic neural networks. In *NIPS*, pp. 118–126, 2016.

Sida Wang and Christopher Manning. Fast dropout training. In *ICML*, pp. 118–126, 2013.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, May 1992.

Feed-forward Propagation in Probabilistic Neural Networks with Categorical and Max Layers Appendix

A Maximum of Several Variables

Approximation of the Mean For each k let $A_k \subset \Omega$ denote the event that $X_k > X_j \ \forall j$, *i.e.* that X_k is the maximum of all variables. Let $q_k = \mathbb{P}(A_k)$ be given. Note that events $\{A_k\}_k$ partition the probability space. The expected value of the maximum $Y = \max_k X_k$ can be written as the following total expectation:

$$\mu' = \mathbb{E}\big[Y\big] = \sum_k \mathbb{P}(A_k)\mathbb{E}[Y \,|\, A_k] = \sum_k q_k \mathbb{E}[X_k \,|\, A_k]. \tag{26}$$

In order to compute each conditional expectation, we approximate the conditional density $p(X_k = x_k \,|\, A_k)$, which is the marginal of the joint conditional density $p(X = x \,|\, A_k)$, *i.e.* the distribution of X restricted to the part of the probability space A_k as illustrated in Fig. A.1. The approximation is a simpler conditional density $p(X_k = x_k \,|\, \hat{A}_k)$ where $\hat{A}_k$ is chosen in the form $\hat{A}_k = [\![X_k \geq m_k]\!]$ and the threshold m_k is chosen to satisfy the proportionality:

$$\mathbb{P}(\hat{A}_k) = \mathbb{P}(A_k) = q_k, \tag{27}$$

which implies $m_k = F^{-1}_{X_k}(q_k)$. This can be also seen as the approximation of the conditional probability $\mathbb{P}(A_k \,|\, X_k = r) = \prod_{j \neq k} F_{X_j}(r)$, as a function of r, with the indicator $[\![m_k \leq r]\!]$, *i.e.* the smooth step function given by the product of sigmoid-like functions $F_{X_k}(r)$ with a sharp step function.

Assuming X_k is logistic, we find $m_k = \mu_k + \sigma_k/\sigma_S \log(\frac{1-q_k}{q_k})$. Then the conditional expectation $\hat{\mu}_k = E[X_k \,|\, \hat{A}_k]$ is computed as

$$\hat{\mu}_k = \frac{1}{q_k}\int_{m_k}^{\infty} x p(X_k{=}x) dx = \frac{1}{q_k}\int_{\log(\frac{1-q_k}{q_k})}^{\infty} (\mu_k + a\frac{\sigma_k}{\sigma_S}) p_S(a) da = \mu_k + \frac{1}{q_k}\frac{\sigma_k}{\sigma_S} H(q_k), \tag{28}$$

where p_S is the density of the standard Logistic distribution, $a = \frac{x - \mu_k}{\sigma_k/\sigma_S}$ is the changed variable under the integral and $H(q_k) = -q_k \log(q_k) - (1 - q_k)\log(1 - q_k)$ is the entropy of a Bernoulli variable with probability q_k. This results in the following interesting formula for the mean:

$$\mu' \approx \sum_k q_k \mu_k + \sum_k \frac{\sigma_k}{\sigma_S} H(q_k). \tag{29}$$

Assuming X_k is normal, we obtain the approximation

$$\mu' \approx \sum_k q_k \mu_k + \sum_k \sigma_k \phi(\Phi^{-1}(q_k)). \tag{30}$$

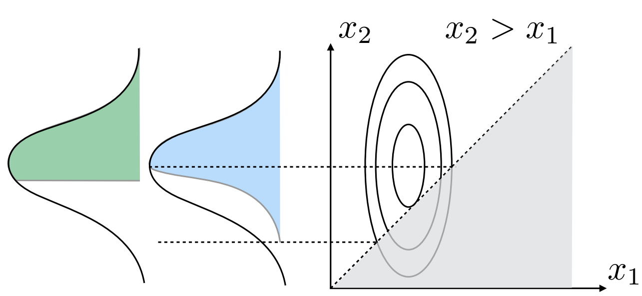

Figure A.1: The joint conditional density $p(X_1 = x_1, X_2 = x_2 \,|\, X_2 > X_1)$, its marginal density $p(X_2 = x_2 \,|\, X_2 > X_1)$ and the approximation $p(X_2 = x_2 \,|\, X_2 > m_2)$, all up to the same normalization factor $\mathbb{P}(X_2 > X_1)$.

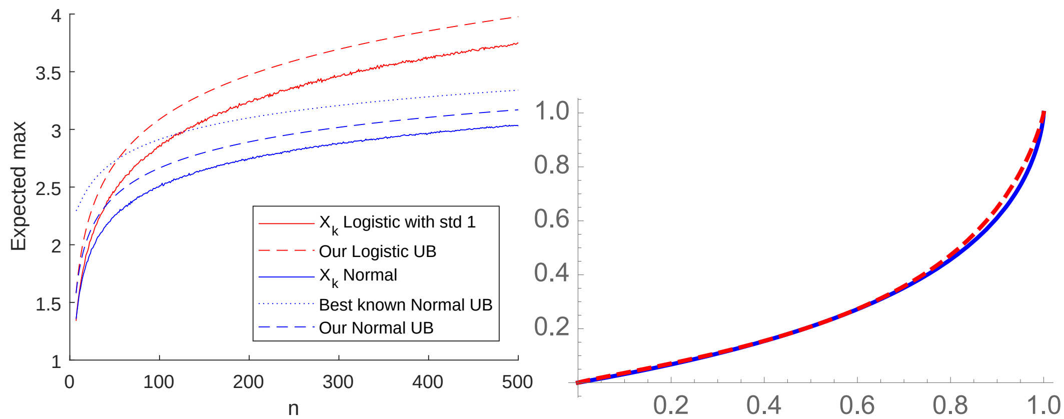

Figure A.2: *Left*: expectation of $Y = \max_k X_k$ for X_k iid logistic or normal, our estimates (dashed) versus sampling-based ground truth (solid) and the best known closed form upper bound for the normal iid case (DasGupta et al., 2014, Theorem 4.1) (dotted). *Right*: the variance scaling function $f(q)$ (35) (solid) and its approximation (36) (dashed).

Lemma A.1. The approximation $\hat{\mu}_k$ is an upper bound on $E[X_k|A_k]$.

Proof. We need to show that $E[X_k|A_k] \leq E[X_k|\hat{A}_k]$. Since $\mathbb{P}(A_k) = \mathbb{P}(\hat{A}_k)$, it is sufficient to prove that

$$\int_{A_k} X_k(\omega) d\mathbb{P}(\omega) \leq \int_{\hat{A}_k} X_k(\omega) d\mathbb{P}(\omega). \tag{31}$$

Let us subtract the integral over the common part $A_k \cap \hat{A}_k$. It remains to show

$$\int_{A_k \setminus \hat{A}_k} X_k(\omega) d\mathbb{P}(\omega) \leq \int_{\hat{A}_k \setminus A_k} X_k(\omega) d\mathbb{P}(\omega). \tag{32}$$

In the RHS integral we have $X_k(\omega) \geq m_k$ since $\omega \in \hat{A}_k = \{\omega \,|\, X_k(\omega) \geq m_k\}$. In the LHS integral we have $X_k(\omega) < m_k$ since $\omega \notin \hat{A}_k$. Notice also that $\mathbb{P}(A_k \setminus \hat{A}_k) = \mathbb{P}(\hat{A}_k \setminus A_k)$. The inequality (32) follows.

Corollary 1. The approximations of the expected maximum (29), (30) are upper bounds in the respective cases when X_k are logistic, resp., normal.

Consider the case then all X_k are all iid logistic or normal with $\mu_k = 0$ and $\sigma_k = 1$. We then have $q_k = \frac{1}{n}$. For logistic case $\mu' \approx nH(\frac{1}{n})$, which is asymptotically $\log(n) + 1 - \frac{1}{2n} + O(1/n^2)$. For normal case $\mu' \approx n\phi(\Phi^{-1}(\frac{1}{n}))$. This formulas are compared versus true (sampling-based) values in Fig. A.2.

Approximation of the Variance For the variance we write

$$\sigma'^2 = \mathbb{E}(Y - \mu')^2 = \sum_k q_k \mathbb{E}((X_k - \mu')^2 \,|\, A_k) \approx \sum_k q_k \mathbb{E}((X_k - \mu')^2 \,|\, \hat{A}_k), \tag{33}$$

where the approximation is due to $\hat{A}_k$, and further rewrite the expression as

$$= \sum_k q_k \mathbb{E}(X_k^2 - 2X_k\mu' + \mu'^2 \,|\, \hat{A}_k) \tag{34a}$$

$$= \sum_k q_k \Big(\mathbb{E}(X_k^2 - \hat{\mu}_k^2 \,|\, \hat{A}_k) + (\hat{\mu}_k - \mu')^2 \Big) \tag{34b}$$

$$= \sum_k q_k (\hat{\sigma}_k^2 + (\hat{\mu}_k - \mu')^2) \tag{34c}$$

where $\hat{\sigma}_k^2 = \mathrm{Var}[X_k \mid \hat{A}_k]$. For X_k with logistic density $p(x)$ the variance integral $\hat{\sigma}_k^2 = \int_{m_k}^{\infty} (x - \hat{\mu}')^2 p(x) dx$ expresses as[2]:

$$\hat{\sigma}_k^2 = \frac{1}{q_k} \frac{\sigma_k^2}{\sigma_S^2} \Big(- \frac{\log^2(1-q_k)}{q_k} - 2\,\mathrm{Li}_2(\frac{q_k}{q_k - 1}) \Big) =: \frac{1}{q_k} \sigma_k^2 f(q_k), \tag{35}$$

where Li_2 is dilogarithm. The function f can be well approximated on $[0, 1]$ with

$$\tilde{f}(q) = \mathcal{S}(a + b\mathcal{S}^{-1}(q)), \tag{36}$$

where $a = -1.33751$ and $b = 0.886763$ are obtained from the first order Tailor expansion of $\mathcal{S}^{-1}(f(\mathcal{S}(t)))$ at $t = 0$. This approximation is shown in Fig. A.2 and is in fact an upper bound on f. We thus obtained a rather simple approximation for the variance

$$\sigma'^2 \approx \sum_k \sigma_k^2 \mathcal{S}(a + b\mathcal{S}^{-1}(q_k)) + \sum_k q_k (\hat{\mu}_k - \mu')^2. \tag{37}$$

B Experiment Details

In this section we give all details necessary to ensure reproducibility of results.

B.1 Implementation Details

We implemented our inference and learning in the pytorch[3] framework. The source code will be publicly available. The implementation is modular: with each of the standard layers we can do 3 kinds of propagation: *AP1*: standard propagation in deterministic layers and taking the mean in stochastic layers (*e.g.*, in dropout we need to multiply by the Bernoulli probability), *AP2*: proposed propagation rules with variances and *sample*: by drawing samples of any encountered stochasticity (such as sampling from Bernoulli distribution in dropout). The last method is also essential for computing Monte Carlo (MC) estimates of the statistics we want to approximate. When the training method is *sample*, the test method is assumed to be AP1, which matches the standard practice of dropout training.

In the implementation of AP2 propagation the input and the output of each layer is a pair of mean and variance. At present we use only higher-level pytorch functions to implement AP2 propagation. For example, AP2 propagation for convolutional layer is implemented simply as

```
y.mean = F.conv2d(x.mean, w) + b
y.var = F.conv2d(x.var, w*w)
```

For numerical stability, it was essential that logsumexp is implemented by subtracting the maximum value before exponentiation

```
m, _ = x.max()
m = m.detach() # does not influence gradient
y = m + torch.log(torch.sum(torch.exp(x - m)))
```

The feed-forward propagation with AP2 is about 3 times slower than AP1 or sample. The relative times of a forward-backward computation in our higher-level implementation are as follows:

```
standard training         1
BN                        1.5
inference=AP2             3
inference=AP2-norm=AP2    6
```

Please note that these times hold for unoptimized implementations. In particular, the computational cost of the AP2 normalization, which propagates single pixel statistics, should be more efficient in comparison to propagating a batch of input images.

[2]Computed with the help of Mathematica

[3]http://pytorch.org

B.2 DATASETS

We used MNIST[4] and CIFAR10[5] datasets. Both datasets provide a split into training and test sets. From the training set we split 10 percent (at random) to create a validation set. The validation set is meant for model selection and monitoring the validation loss and accuracy during learning. The test sets were currently used only in the stability tests.

B.3 TRAINING

For the optimization we used batch size 32, SGD optimizer with Nesterov Momentum 0.9 (pytorch default) and the learning rate $lr \cdot \gamma^k$, where k is the epoch number, lr is the initial learning rate, γ is the decrease factor. In all reported results for CIFAR we used γ such that $\gamma^{600} = 0.1$ and 1200 epochs. This is done in order to make sure we are not so much constrained by the performance of the optimization and all methods are given sufficient iterations to converge. The initial learning rate was selected by an automatic numerical search optimizing the training loss in 5 epochs. This is performed individually per training case to take care for the differences introduced by different initializations and training methods.

When not said otherwise, parameters of linear and convolutional layers were initialized using pytorch defaults, *i.e.*, uniformly distributed in $[-1/\sqrt{c},\, 1/\sqrt{c}]$, where c is the number of inputs per one output.

Standard minor data augmentation was applied to the training and validation sets in CIFAR-10, consisting in random translations ± 2 pixels (with zero padding) and horizontal flipping.

When we train with normalization, it is introduced after each convolutional and fully connected layer.

B.4 NETWORK SPECIFICATIONS

The LeNet5 architecture Lecun et al. (2001) is:

```
Conv2d(1, 6, ks=5, st=2), Activation
MaxPooling
Conv2d(6, 16, ks=5, st=2), Activation
MaxPooling
FC(4*4*16, 120), Activation
FC(120, 84), Activation
FC(84, 10), Activation
LogSoftmax
```

Convolutional layer parameters list input channels, output channels, kernel size and stride.

The *All-CNN* network Springenberg et al. (2015) has the following structure of convolutional layers:

```
ksize = [3,  3,  3,  3,   3,   3,   3,   1,   1 ]
stride= [1,  1,  2,  1,   1,   2,   1,   1,   1 ]
depth = [96, 96, 96, 192, 192, 192, 192, 192, 10]
```

each but the last one ending with activation (we used LReLU). The final layers of the network are

```
AdaptiveAvgPool2d, LogSoftmax
```

ConvPool-CNN-C model replaces stride-2 convolutions by stride-1 convolutions of the same shape followed by 2x2 max pooling with stride 2.

B.5 AUXILIARY RESULTS ON NORMALIZATION

We test the analytic normalization method (Shekhovtsov & Flach, 2018) in a network with max pooling and Leaky ReLU layers. We consider the "ConvPool-CNN-C" model of Springenberg et al.

[4] http://yann.lecun.com/exdb/mnist/

[5] https://www.cs.toronto.edu/~kriz/cifar.html

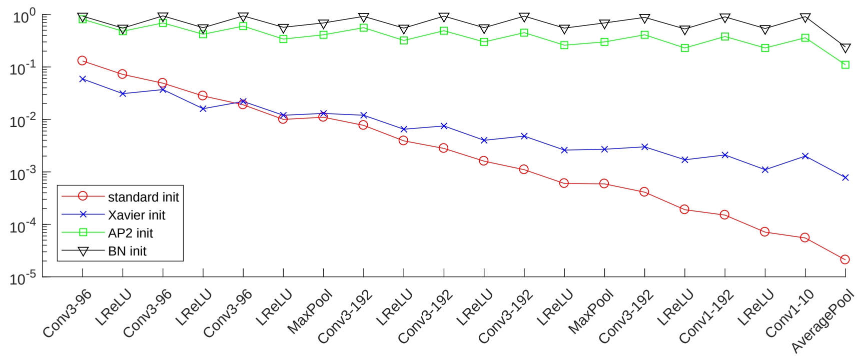

Figure B.1: Standard deviation of neurons in network layers after different initializations. The shown values are averages over all units in each layer (spatial and channel dimensions). With standard random initialization the variances quickly decrease and the network output for the whole dataset collapses nearly to a single point, complicating the training. Xavier init does not fully resolve the problem. Analytic normalization provides standard deviation within a small factor of 1 in all layers, comparable to BN. The zig-zagging effect is observed because the normalization is performed after linear layers only.

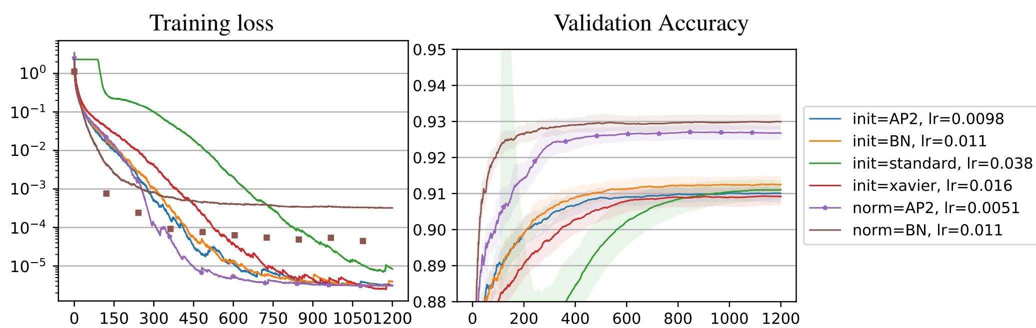

Figure B.2: The effect of initialization/normalization on the progress of training. Observe that the initialization alone significantly influences the automatically chosen initial learning rate (lr) and the "trainability" of the network. Using the normalization during the training further improves performance for both batch and analytic normalization. BN has an additional regularization effect Ioffe (2017), the square markers in the left plot show BN training loss using averaged statistics.

(2015) on CIFAR-10 dataset. It's structure is shown on the x-axis of Fig. B.1. We first apply different initialization methods and compute variances in each layer over the training dataset. Fig. B.1 shows that standard initialization with weights distributed uniformly in $[-1/\sqrt{n_{\rm in}}, 1/\sqrt{n_{\rm in}}]$, where $n_{\rm in}$ is the number of inputs per single output of a linear mapping results in the whole dataset concentrated around one output point with standard deviation 10^{-5}. Initialization of Glorot & Bengio (2010), using statistical arguments, improves this behavior. For the analytic approximation, we take statistics of the dataset itself (μ_0, σ_0) and propagate them through the network, ignoring spatial dimensions of the layers. When normalized by this estimates, the real dataset statistics have variances close to one and means close to zero, *i.e*. the normalization is efficient. For comparison, we also show normalization by the batch statistics with a batch of size 32. Fig. B.2 further demonstrates that the initialization is crucial for efficient learning, and that keeping track of the normalization during training and back propagating through it (denoted norm=AP2 in the figure) performs even better and may be preferable to batch normalization in many scenarios such as recurrent NNs.

B.6 Accuracy with Max pooling

Table B.2 shows accuracy of posterior approximation results for ConvPool-CNN-C, discussed above which includes max pooling layers. The network is trained and evaluated on CIFAR-10 with dropout the same way as in § 5.1.

	C	A	C	A	C	A	M	C	A	C	A	C	A	M	C	A	C	A	C	P	Softmax
σ^*	0	0.17	0.56	0.40	1.5	0.85	0.95	3.9	2.4	10	5.3	25	7.0	8.9	39	21	43	11	26	4.3	
ε_{μ_1}	-	0.00	0.00	0.03	0.21	0.05	0.21	0.96	0.11	0.43	0.11	0.71	0.09	0.18	0.79	0.14	0.37	0.10	0.26	0.97	KL 0.06
ε_{μ_2}	-	0.00	0.00	0.01	0.01	0.01	0.09	0.42	0.05	0.22	0.04	0.19	0.03	0.11	0.59	0.07	0.18	0.08	0.19	0.73	KL 0.03
ε_{σ_2}	-	1.00	1.00	1.02	0.93	0.98	1.08	1.21	1.24	1.00	1.09	0.88	0.99	1.15	1.02	0.97	0.97	1.26	1.00	0.73	

Table B.1: Accuracy of approximation of mean and variance statistics for each layer in a fully trained ConvPool-CNN-C network with dropout. A significant drop of accuracy is observed as well after max pooling, we believe due to the violation of the independence assumption.

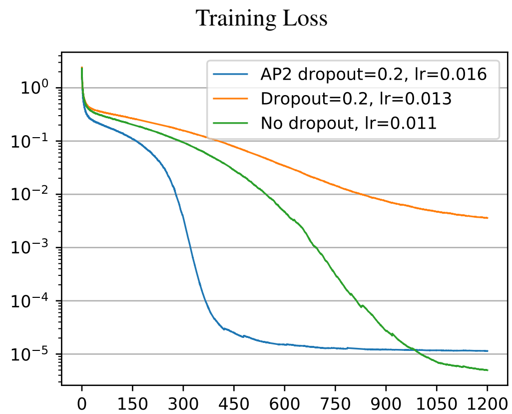

Figure B.3: Training loss corresponding to Fig. 3. While stochastic dropout slows the training down due to increased stochasticity of the gradient, the analytic dropout smoothes the loss function and speeds the training up.

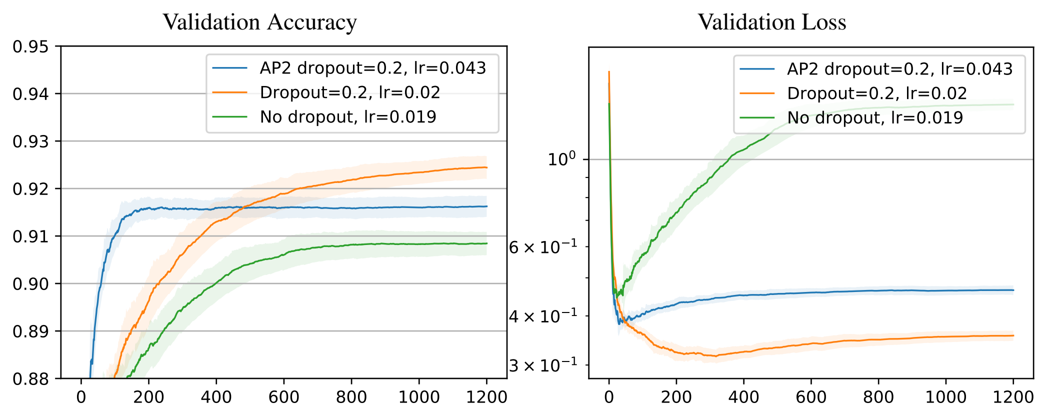

Figure B.4: Comparison of analytic AP2 dropout with baselines. All methods use the same initialization using AP2 statistics and no normalization. Analytic dropout improves over training with no dropout and is faster than sampling dropout but starts slightly overfitting soon.

B.7 Auxiliary Results on Analytic Dropout

Fig. B.4 shows training results, when we use AP2 method only to initialize the network, but switch off the normalization during the training. In this setting we see that AP2 approximate dropout has a significant regularization effect (validation loss) and improves in accuracy over the baseline without dropout. It also performs faster than stochastic dropout, but achieves worse final accuracy in this case. This shows that other regularizer, namely the normalization used in § 5.3 are important as well. Table B.2 confirms that "AP2 calibrated" keeps the good test-time performance for the network trained with stochastic dropout (the best performing network in Fig. B.4).

Standard dropout		
Method	NLL	Acc.
AP1	0.487	0.923
AP2	0.293	0.923
AP2 calibrated	**0.244**	0.923
MC-10	0.312	0.922
MC-100	0.256	0.924
MC-1000	**0.244**	**0.924**

Table B.2: Different test-time propagation methods for a model with dropout. We show test negative log likelihood (NLL) with AP2 and MC posterior estimates for network trained with standard dropout and using AP2 (analytic) dropout. In both cases AP2 results in improved posterior estimates. "AP2 calibrated" rescales the variance in the last layer by the average factor σ/σ^* (see § 5.1) estimated on the validation set.