Explicit Regularisation in Gaussian Noise Injections

Abstract

We study the regularisation induced in neural networks by Gaussian noise injections (GNIs). Though such injections have been extensively studied when applied to data, there have been few studies on understanding the regularising effect they induce when applied to network activations. Here we derive the explicit regulariser of GNIs, obtained by marginalising out the injected noise, and show that it penalises functions with high-frequency components in the Fourier domain; particularly in layers closer to a neural network’s output. We show analytically and empirically that such regularisation produces calibrated classifiers with large classification margins.

1 Introduction

Noise injections are a family of methods that involve adding or multiplying samples from a noise distribution, typically an isotropic Gaussian, to the weights or activations of a neural network during training. The benefits of such methods are well documented. Models trained with noise often generalise better to unseen data and are less prone to overfitting (Srivastava et al., 2014; Kingma et al., 2015; Poole et al., 2014).

Even though the regularisation conferred by Gaussian noise injections (GNIs) can be observed empirically, and the benefits of noising data are well understood theoretically (Bishop, 1995; Cohen et al., 2019; Webb, 1994), there have been few studies on understanding the benefits of methods that inject noise throughout a network. Here we study the explicit regularisation of such injections, which is a positive term added to the loss function obtained when we marginalise out the noise we have injected.

Concretely our contributions are:

- We derive an analytic form for an explicit regulariser that explains most of GNIs’ regularising effect.
- We show that this regulariser penalises networks that learn functions with high-frequency content in the Fourier domain and most heavily regularises neural network layers that are closer to the output. See Figure[1] for an illustration.
- Finally, we show analytically and empirically that this regularisation induces larger classification margins and better calibration of models.
For first order derivatives this is a matrix, i.e., the form $\mathbf{2.2}$ Sobolev Spaces

to the multiplicative case. Here we focus our analysis on noise

The multiplicative case can be rewritten as an activation-scaled addition:

where $\mathbf{each}$ hidden layer unit. In the additive case, we obtain:

What happens to these activations when we inject noise? First, let

network consisting of dense layers (a.k.a. a multi-layer perceptron: MLP) we have that:

Under stochastic optimisation algorithms, such as Stochastic Gradient Descent (SGD), we estimate

Training a neural network involves optimising network parameters to maximise the marginal likeli-

hood of a set of labels given features via gradient descent. With a training dataset $\mathcal{D}$ composed

of $N$ data-label pairs of the form $(x, y)$ $x \in \mathbb{R}^d$, $y \in \mathbb{R}^m$ and a feed-forward neural network with $M$

parameters divided into $L$ layers: $\theta = \{W_1, ..., W_L\}$, $\theta \in \mathbb{R}^M$, our objective is to minimise the

expected negative log likelihood of labels $y$ given data $x$, $-\log p_\theta(y|x)$, and find the optimal set of

parameters $\theta^*$ satisfying:

$$\theta^* = \arg\min_\theta \mathcal{L}(\mathcal{D}; \theta), \quad \mathcal{L}(\mathcal{D}; \theta) := -\mathbb{E}_{x,y \sim \mathcal{D}} \log p_\theta(y|x).$$  \hspace{1cm} (1)$$

Under stochastic optimisation algorithms, such as Stochastic Gradient Descent (SGD), we estimate

$\mathcal{L}$ by sampling a mini-batch of data-label pairs $\mathcal{B} \subset \mathcal{D}$.

$$\mathcal{L}(\mathcal{B}; \theta) = -\mathbb{E}_{x,y \sim \mathcal{B}} \log p_\theta(y|x) \approx \mathcal{L}(\mathcal{D}; \theta).$$  \hspace{1cm} (2)$$

Consider an $L$ layer network with no noise injections and a non-linearity $\phi$ at each layer. We obtain

the activations $h = \{h_0, ..., h_L\}$, where $h_0 = x$ is the input data before any noise is injected. For a

network consisting of dense layers (a.k.a. a multi-layer perceptron: MLP) we have that:

$$h_k(x) = \phi(W_k h_{k-1}(x))$$  \hspace{1cm} (3)$$

What happens to these activations when we inject noise? First, let $\epsilon$ be the set of noise injections at

each layer: $\epsilon = \{\epsilon_0, ..., \epsilon_L\}$. When performing a noise injection procedure, the value of the next

layer’s activations depends on the noised value of the previous layer. We denote the intermediate,

soon-to-be-noised value of an activation as $\hat{h}_k$ and the subsequently noised value as $h_k$:

$$h_k(x) = \phi\left(W_k \hat{h}_{k-1}(x)\right), \quad \hat{h}_k(x) = h_k(x) \circ \epsilon_k,$$  \hspace{1cm} (4)$$

where $\circ$ is some element-wise operation. We can, for example, add or multiply Gaussian noise to

each hidden layer unit. In the additive case, we obtain:

$$\hat{h}_k(x) = \tilde{h}_k(x) + \epsilon_k, \quad \epsilon_k \sim \mathcal{N}(0, \sigma^2_k I).$$  \hspace{1cm} (5)$$

The multiplicative case can be rewritten as an activation-scaled addition:

$$\tilde{h}_k(x) = h_k(x) + \epsilon_k, \quad \epsilon_k \sim \mathcal{N}\left(0, h_k^2(x)\sigma^2_k I\right).$$  \hspace{1cm} (6)$$

Here we focus our analysis on noise additions, but through equation (6) we can translate our results to

the multiplicative case.

$\textbf{2.2} \hspace{1cm} \text{Sobolev Spaces}$

To define a Sobolev Space we use the generalisation of the derivative for vector-valued functions of

the form $g : \mathbb{R}^d \rightarrow \mathbb{R}^m$. We denote the $\alpha^{th}$ derivative of $g$ with respect to its input $x$ as: $D^\alpha g(x)$.

For first order derivatives this is a matrix, i.e $Dg(x) \in \mathbb{R}^{m \times d}$.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure1.png}
\caption{Here we illustrate the effect of GNIs injected throughout a network’s activations. Each coloured dot represents a neuron’s activations. We add GNIs, represented as circles, to each layer’s activations bar the output layer. GNIs induce a network for which each layer learns a progressively lower frequency function, represented as a sinusoid matching in colour to its corresponding layer.}
\end{figure}
Proposition 1. Consider an $L$ layer neural network experiencing isotropic GNIs at each layer $k \in [0, \ldots, L - 1]$ of dimensionality $d_k$. We denote this added noise as $\epsilon = \{\epsilon_0, \ldots, \epsilon_{L-1}\}$. We

\[ D^\alpha \mathbf{h}_k(x) \]

As in Section 2.2, we use the generalisation of the derivative for vector-valued functions. For example, $D^\alpha \mathbf{h}_k(\mathbf{h}_{k-1}(x))$ denotes the $\alpha$th derivative of the non-noised $k$th layer activations $\mathbf{h}_k(x)$ with respect to the preceding layer’s activations $\mathbf{h}_{k-1}(x)$ and $D^\alpha \mathcal{L}(\mathbf{h}_k(x), y)$ denotes the $\alpha$th derivative of the loss with respect to the non-noised activations $\mathbf{h}_k(x)$.

**Proposition 1.** Consider an $L$ layer neural network experiencing isotropic GNIs at each layer $k \in [0, \ldots, L - 1]$ of dimensionality $d_k$. We denote this added noise as $\epsilon = \{\epsilon_0, \ldots, \epsilon_{L-1}\}$. We
where the relative importance of to understand the main contributors behind the regularising effect of GNIs, we first want to establish See Appendix B for the proof and for the exact form of any noise is added, as defined in equation where is in as: Given this form for the accumulated noise, we can now define the expected regulariser induced by isotropic GNIs, as: where is drawn from the dataset, are the activations before any noise is added, as defined in equation See Appendix A for the proof. Given this form for the accumulated noise, we can now define the expected regulariser induced by isotropic GNIs, as: For compactness of notation, we denote each layer’s Jacobian as: and demonstrates that for all batches and GNI variances remains dominant in early stages of training as for 6 layer MLPs undergoing GNIs at each layer. Each point corresponds to one of 250 different network initialisation acting on a batch of size 32 for the classification dataset CIFAR10 and regression dataset Boston House Prices (BHP) datasets, such that we test both classification and regression settings. The dotted red line corresponds to and demonstrates that for all batches and GNI variances . In (c,d) we plot the ratio in the first 100 training steps (t) for 10 randomly initialised networks. Shading corresponds to the standard deviation of values over the 10 networks. remains dominant in early stages of training as . Assume is in the class of infinitely differentiable functions. We can define the accumulated noise at layer , as: where is a remainder term of third order and above We assume is in the class of infinitely differentiable functions. We can marginalise out the injected noise to obtain an added regulariser: where is the loss for a pair drawn from the dataset, are the activations before any noise is added, as defined in equation . is a remainder term of third order and above terms in . We assume is in the class of infinitely differentiable functions. We can marginalise out the injected noise to obtain an added regulariser: where is estimated using for a range of . See Appendix B for the proof and for the exact form of . To understand the main contributors behind the regularising effect of GNIs, we first want to establish the relative importance of and . In Figure 2 we show that for a range of noise injection variances, datasets, and activation functions; where is estimated using 1000.
Figure 3: Figure (a) shows the test set loss for convolutional models (CONV) and 4 layer MLPs trained on SVHN with R and GNIs for $\sigma^2 = 0.1$, and no noise (Baseline). Figure (b) shows the trace of the network parameter Hessian for a 2-layer, 32-unit-per-layer MLP where $H_{i,j} = \frac{\partial L}{\partial w_i \partial w_j}$, which is a proxy for the parameters’ location in the loss landscape. All networks use ELU activations. See Appendix [H] for more such results on other datasets and network architectures.

Regularisation in Regression In the case of regression one of the most commonly used loss functions is the mean-squared error (MSE), which is defined for a data label pair $(x, y)$ as:

$$L(x, y) = \frac{1}{2}(y - h_L(x))^2$$

For this loss, the Hessians in Theorem 1 are simply the identity matrix. The explicit regularisation term, guaranteed to be positive is:

$$R = \frac{1}{2}E_{x \sim B} \left[ \sum_{k=0}^{L-1} \sigma_k^2 \|J_k(x)\|_2^2 \right]$$

where $\sigma_k^2$ is the variance of the noise $\epsilon_k$ injected at layer $k$ and $\| \cdot \|_2$ is the Frobenius norm. See Appendix [B.1] for a proof.

Regularisation in Classification In the case of classification, we consider the case of a cross-entropy (CE) loss. Recall that we consider our network outputs $h_L$ to be the pre-softmax of the logits of the final layer. We denote $p(x) = \text{softmax}(h_L(x))$. For a pair $(x, y)$ we have:

$$L(x, y) = -\sum_{c=0}^{C} y_c \log(p(x))_c,$$

where $c$ indexes over $C$ possible classes. The hessian $H_L(\cdot)$ no longer depends on $y$:

$$H_L(x)_{i,j} = \begin{cases} p(x)_i(1 - p(x)_j) & i = j \\ -p(x)_i p(x)_j & i \neq j \end{cases}$$

This Hessian is positive-semi-definite and $R$, guaranteed to be positive, can be written as:

$$R = \frac{1}{2}E_{x \sim B} \left[ \sum_{k=0}^{L-1} \sigma_k^2 \sum_{i,j} (\text{diag}(H_L(x))^T J_k^2(x))_{i,j} \right]$$

where as before $\sigma_k^2$ is the variance of the noise $\epsilon_k$ injected at layer $k$. See Appendix [B.2] for a detailed demonstration of this.
To test our derived regularisers, we want to ascertain that models trained with $R$ have similar training profiles to models trained with GNIs. In Figure 3, we show that models trained with $R$ and GNIs have similar test-set loss and parameter Hessians throughout training. This means that models trained with $R$ and GNIs have almost identical trajectories through the loss landscape and that $R$ is a good descriptor of the effect of GNIs.

Using the derived analytic forms for $R$ in both classification and regression, our aim now is to understand the mechanism underpinning the effect of GNIs. As we show, $R$ has a connection to the Fourier domain, and it penalises neural networks that parameterize functions with higher frequencies in the Fourier domain; offering a novel lens under which to study GNIs.

4 Fourier Domain Regularisation

To link our derived regularisers to the Fourier domain, we use the connection between neural networks and Sobolev Spaces mentioned above. Recall that by Hornik (1991), we can only assume a sigmoid or piecewise linear neural network parameterises a function in a weighted Sobolev space with measure $\mu$, if we assume that the measure $\mu$ has compact support on a subset $\Omega \subset \mathbb{R}^d$. As such, we equip our space with the probability measure $\mu(x)$, which we assume has compact support on some subset $\Omega \subset \mathbb{R}^d$ where $\mu(\Omega) = 1$. We define it such that $d\mu(x) = p(x)dx$ where $dx$ is the Lebesgue measure and $p(x)$ is the data density function. Given this measure, we can establish a connection between the derivative of functions that lie in the Hilbert-Sobolev space $W^{1,2}(\mathbb{R}^d)$ and the Fourier domain.

**Theorem 2.** Consider a function, $f_\theta : \mathbb{R}^d \rightarrow \mathbb{R}$, with a d-dimensional input and a single output with $f_\theta \in W^{1,2}(\mathbb{R}^d)$ where $\mu$ is a probability measure which we assume has compact support on some subset $\Omega \subset \mathbb{R}^d$ such that $\mu(\Omega) = 1$. Assuming the derivative of $f_\theta$, $Df_\theta$, is in $L^2(\mathbb{R}^d)$; the square of the norm of $Df_\theta$ in $L^2(\mathbb{R}^d)$, the $L^2$ space equipped with measure $\mu$, can be written as:

$$
\|Df_\theta\|^2_{L^2(\mathbb{R}^d)} = \int_{\mathbb{R}^d} \mathcal{G}(\omega) \left[ \mathcal{G}(\omega) \ast \mathcal{P}(\omega) \right] d\omega
$$

where $\mathcal{F}$ is the Fourier transform of $f_\theta$, $\mathcal{P}$ is the Fourier transform or the ‘characteristic function’ of the probability measure $\mu$, $j$ indexes over $\omega = [\omega_1, \ldots, \omega_d]$, $\ast$ is the convolution operator, and $(\cdot)$ is the complex conjugate.

See Appendix [C] for the proof. Note that in the case where the dataset contains finitely many points, the integral for the norm $\|Df_\theta\|^2_{L^2(\mathbb{R}^d)}$ is approximated by sampling a batch from the dataset which is distributed according to the presumed probability measure $\mu(x)$. Expectations over a batch thus approximate integration over $\mathbb{R}^d$ with the measure $\mu(x)$ and this approximation improves as the batch size grows. Using this fact, we can apply Theorem 2 to $R$ and link $R$ to the Fourier domain.

**Regression** Let us begin with the case of regression. Assuming differentiable and continuous activation functions, then the Jacobians within $R$ are equivalent to the derivatives in Definition 2.1. Theorem 2 only holds for functions that have 1-D outputs, but we can decompose the Jacobians $J_k$ as the derivatives of multiple 1-D output functions. We write that $J_{k,i}(\cdot) = Df_{k,i}(\cdot)$, where $f_{k,i}(\cdot)$ is the function from layer $k$ to the $i$th network output, $i = 1, \ldots, d_L$. Using this perspective, and the fact that each $f_{k,i}(\cdot) \in W^{1,2}(\mathbb{R}^d)$ ($d_k$ is the dimensionality of the $k$th layer), if we assume that the probability measure of our space $\mu(x)$ has compact support, we can use Theorem 2 to write:

$$
R = \frac{1}{2} \mathbb{E}_{x \sim B} \left[ \sum_{k=0}^{L-1} \sigma_k^2 \sum_i \| J_{k,i}(x) \|_2^2 \right] = \frac{1}{2} \sum_{k=0}^{L-1} \sigma_k^2 \sum_i \mathbb{E}_{x \sim B} \left[ \| J_{k,i}(x) \|_2^2 \right]
$$

$$
\approx \frac{1}{2} \sum_{k=0}^{L-1} \sigma_k^2 \sum_i \| Df_{k,i}(x) \|_{L^2(\mathbb{R}^d)}^2 = \frac{1}{2} \sum_{k=0}^{L-1} \sigma_k^2 \sum_i \int_{\mathbb{R}^d} \mathcal{G}_i^k(\omega) \left[ \mathcal{G}_i^k(\omega) \ast \mathcal{P}(\omega) \right] d\omega
$$

where $h_0 = x$, $i$ indexes over output neurons, and $\mathcal{G}_i^k(\omega) = (\sum_j \omega_j) \mathcal{F}_i^k(\omega)$, where $\mathcal{F}_i^k$ is the Fourier transform of the function $f_{k,i}(\cdot)$. The approximation comes from the fact that in SGD, as mentioned above, integration over the dataset is approximated by sampling mini-batches $B$. 

6
Figure 4: As in Rahaman et al. (2019), we train 6-layer deep 256-unit wide ReLU networks trained to regress the function \( \lambda(z) = \sum_i \sin(2\pi r_i z + \phi(i)) \) with \( r_i \in (5, 10, \ldots, 45, 50) \). We train these networks with no noise (Baseline), with GNIs of variance 0.1 injected into each layer except the final layer (Noise), and with the \( R \) for regression in (10) (Exp Reg). The first row shows the Fourier spectrum (x-axis) of the networks (calculated using Lemmas 1 and 2 of Rahaman et al. (2019)) as training progresses (y-axis) averaged over 10 training runs. Colours show each frequency’s amplitude clipped between 0 and 1. The second row shows samples of randomly generated target functions and the function learnt by the networks.

**Classification**

The classification setting requires a bit more work. Recall that our Jacobians are weighted by \( \text{diag}(H_L(x))_i^T \). \( H_L(x) \) is PSD and its diagonal entries are all less than 1 by Equation (12). We can define a new set of measures such that

\[
\mu_i(x) = \text{diag}(H_L(x))_i^T p(x) dx, \quad i = 1 \ldots d_L.
\]

Because this new measure is positive, finite and still has compact support, Theorem 2 still holds for the spaces indexed by \( \mu_i \): \( W_{\mu_i}^{1,2}(\mathbb{R}^d) \). Using these new measures, and the fact that each \( f_{\theta,i}^{k}(\cdot) \in W_{\mu_i}^{1,2}(\mathbb{R}^{d_k}) \), we can use Theorem 2 to write that for classification models:

\[
R = \frac{1}{2} \sum_{k=0}^{L-1} \sigma_k^2 \sum_{i} \mathbb{E}_{x \sim B} \left[ \text{diag}(H_L(x))_i^T \left\| J_{k,i}(x) \right\|_2^2 \right]
\]

\[
\approx \frac{1}{2} \sum_{k=0}^{L-1} \sigma_k^2 \sum_{i} \left\| D f_{\theta,i}^{k}(\omega) \right\|_{L_2(\mathbb{R}^{d_k})}^2 = \frac{1}{2} \sum_{k=0}^{L-1} \sigma_k^2 \sum_{i} \int_{\mathbb{R}^{d_k}} G_i^k(\omega) \left[ G_i^k(\omega) \ast P_i(\omega) \right] d\omega \tag{15}
\]

Here \( P_i \) is the Fourier transform of the \( i \)th measure \( \mu_i \) and as before \( G_i^k(\omega) = (\sum_j \omega_j) F_i^k(\omega) \), where \( F_i^k \) is the Fourier transform of the function \( f_{\theta,i}^{k}(\cdot) \).

As such for both regression and classification, GNIs, by way of \( R \), induce a prior which favours smooth functions with low-frequency components. This prior is enforced by the terms \( G_i^k(\omega) \) which become large in magnitude when functions have high-frequency components, penalising neural networks that learn such functions. We demonstrate this empirically in Figure 4 where networks trained with GNIs learn functions that don’t overfit; with lower-frequency components relative to their non-noised counterparts. In Appendix D we also show that this penalisation corresponds to Tikhonov regularisation, regularisation methods which penalise a function’s norm in some Hilbert space; in our case the Hilbert-Sobolev space \( W_{\mu_i}^{1,2}(\mathbb{R}^d) \).
domain and sensitivity to noise is quite simple to establish analytically by studying the
2019; Liu et al., 2019; Li et al., 2018). GNIs injected at each network layer, which correspond to a
demonstrating that models trained with noised data are less sensitive to perturbations (Cohen et al.,
quencies in the Fourier domain are less sensitive to noisy data, and there is already ample evidence
to noise. Rahaman et al. (2019) have shown empirically that classifiers biased towards lower fre-
model. As one might intuit there is a link between the Fourier domain and a model's sensitivity
Sensitivity to noise

What does regularisation in the Fourier domain accomplish? In Appendix B we demonstrate that,
4.1 The Benefits of Fourier Penalisation

We measure this layer-wise regularisation in ReLU networks, by measuring $Dh_k(h_{k-1}(x)) = \tilde{W}_k$. $W_k$ is obtained from the original weight matrix $W_k$ by setting its $i^{th}$ column to zero whenever the neuron $i$ of the
layer is inactive. In (a,d) we plot the trace of each layer’s weight matrix $Tr(W_k)$. For GNI models, deeper layers learn functions with a lower frequency, indicating that each successive layer in these
networks learns a function with lower frequency components than the past layer.

We also measure the trace of network weights, which in ReLU networks are indicators of lower frequency functions. The inputs of hidden layers in these
networks, by measuring

$Tr(W_k)$ will have higher frequency components than the next layer’s function $h_k(h_{k-1}(x))$.

Note that there is a recursive structure to the penalisation induced by $R$. Consider the layer-to-layer
functions which map from a layer $k-1$ to $k$, $h_k(h_{k-1}(x))$. $Dh_k(h_{k-1}(x))$ is penalised $k$
times in $R$ as this derivative appears in $J_0, J_1, \ldots J_{k-1}$ due to the chain rule. As such, when training
with GNIs, we can expect the norm of $||Dh_k(h_{k-1}(x))||_2$ to decrease as the layer index $k$
increases (i.e the closer we are to the network output). By Theorem 3, Equation (8), and Equation (15) we
know that larger $||Dh_k(h_{k-1}(x))||_2$ correspond to functions with higher frequency components.
Consequently, we can expect when training with GNIs the function $h_k(h_{k-1}(x))$ will have higher frequency components than the past layer.

We measure this layer-wise regularisation in ReLU networks, by measuring $Dh_k(h_{k-1}(x)) = \tilde{W}_k$. $W_k$ is obtained from the original weight matrix $W_k$ by setting its $i^{th}$ column to zero whenever the neuron $i$ of the
layer is inactive. We also measure the trace of network weights, which in ReLU networks are indicators of lower frequency functions. The inputs of hidden layers in these
networks, by measuring

layerwise ordering and striation of $W_k$ is obtained from the original weight matrix $W_k$. As such, negative weights will be
likely to ‘deactivate’ a ReLU-neuron, inducing sparser $W_k$, smaller $||W_k||_2$, and parameterising
a lower frequency function. As an indicator for the ‘number’ of negative components of a weight
matrix, we can measure its trace. In Figure 5 we demonstrate that $||W_k||_2$ and $Tr(W_k)$ decrease as $k$
increases for ReLU-networks trained with GNIs, indicating that each successive layer in these
networks learns a function with lower frequency components than the past layer.

Sensitivity to noise

A model’s weakness to input perturbations is called the sensitivity of the
model. As one might intuit there is a link between the Fourier domain and a model’s sensitivity
to noise. Rahaman et al. (2019) have shown empirically that classifiers biased towards lower frequ-
cencies in the Fourier domain are less sensitive to noisy data, and there is already ample evidence
demonstrating that models trained with noised data are less sensitive to perturbations (Cohen et al.,
2019; Liu et al., 2019; Li et al., 2018). GNIs injected at each network layer, which correspond to a
greater penalisation in the Fourier domain should induce even less sensitivity to noise than simply
noising data. We demonstrate that this is the case in Figure 6. This connection between the Fourier
domain and sensitivity to noise is quite simple to establish analytically by studying the classification
margins of a model, which we do in Appendix G.
Figure 6: In (a) and (b) a model's sensitivity to noise by adding noise of variance $\sigma^2$ to data and measuring the resulting model accuracy given this corrupted test data. We show this for 2-layer MLPs trained on CIFAR10 (a) and SVHN (b) for models trained with no noise (Baseline), models trained with noise on their inputs (GNI Input), models trained with noise on all their layers (GNI All Layers), and models trained with the $R$ for classification. Noise added during training has variance $\sigma^2 = 0.1$ and confidence intervals are the standard deviation over batches of size 1024. Models trained with noise on all layers, and those trained with $R$, have the slowest decay of performance as $\alpha$ increases, confirming our analysis that such models have larger classification margins.

**Calibration** Networks with lower frequency components are also better calibrated. A perfectly calibrated model is one we can trust. Given a network’s prediction $\hat{y}(x)$ with confidence $\hat{p}(x)$ for a point $x$, perfect calibration consists of being as likely to be correct as you are confident: $p(\hat{y} = y | \hat{p} = r) = r$, $\forall r \in [0,1]$ (Dawid, 1982; DeGroot and Fienberg, 1983).

In Appendix G we show that models that are biased toward lower frequency spectra have lower ‘capacity measures’, measures which attempt to measure model complexity. Guo et al. (2017) show empirically that models with lower capacity are better calibrated and in Figure H.6 we show that this holds true for models trained with GNIs and $R$.

5 Related Work

Many variants of GNIs have been proposed to regularise neural networks. Poole et al. (2014) extend this process to its logical conclusion and apply noise to all computational steps in a neural network layer. Not only is noise applied to the layer input it is applied to the layer output and to the pre-activation function logits. The authors allude to explicit regularisation but only derive a result for a single layer auto-encoder with a single noise injection. Similarly, Bishop (1995) derive an analytic form for the explicit regulariser induced by noise injections on data and show that such injections are equivalent to Tikhonov regularisation in an unspecified function space.

Recently Wei et al. (2020) conducted similar analysis to ours, dividing the effects of Bernoulli dropout into explicit and implicit effects. Their work is built on that of Mele and Altarelli (1993), Helmbold and Long (2015), and Wager et al. (2013) who perform this analysis for linear neural networks. Arora et al. (2020) derive an explicit regulariser for Bernoulli dropout on the final layer of a neural network. Further, recent work by Dieng et al. (2018) shows that noise additions on recurrent network hidden states outperform Bernoulli dropout in terms of performance and bias.

6 Conclusion

In this work, we derived analytic forms for the explicit regularisation induced by Gaussian noise injections. We characterise the explicit regulariser as a form of Tikhonov regularisation which penalises networks with high-frequency content in the Fourier space. Further we show that this regularisation is not distributed evenly within a network, as it disproportionately penalises high-frequency content in layers closer to the network output. Finally we show that such a penalisation induces models that are better calibrated and less sensitive to input perturbations.
Acknowledgments

This research was directly funded by the Alan Turing Institute under Engineering and Physical Sciences Research Council (EPSRC) grant EP/N510129/1. AC was supported by an EPSRC Studentship. MW was supported by EPSRC grant EP/G03706X/1. US was supported by the French National Research Agency (ANR) as a part of the FBIMATRIX (ANR-16-CE23-0014) project. SR gratefully acknowledges support from the UK Royal Academy of Engineering and the Oxford-Man Institute. CH was supported by the Medical Research Council, the Engineering and Physical Sciences Research Council, Health Data Research UK, and the Li Ka Shing Foundation

Impact Statement

This paper uncovers a new mechanism by which a widely used regularisation method operates and paves the way for designing new regularisation methods which take advantage of our findings. Regularisation methods produce models that are not only less likely to overfit, but also have better calibrated predictions that are more robust to distribution shifts. As such improving our understanding of such methods is critical as machine learning models become increasingly ubiquitous and embedded in decision making.

Bibliography


Ben Poole, Jascha Sohl-Dickstein, and Surya Ganguli. Analyzing noise in autoencoders and deep networks. 2014.


