Neural Networks, All of Whose Derivatives are Bidrectional Self-Normalizing Neural Networks

Background and the Core Idea

Sitzman et al., in *Implicit Neural Representations With Periodic Activation Functions* propose the use of periodic activation functions, specifically the use of the sine in the fitting of neural networks to, for example, images and signed distance functions. They prove that if the weights are uniformly distributed in $[-c, c]$ where $c = \sqrt{6/f}$ where f is the fan in, then the pre-activations are always standard normal distributed irrespective of the depth of the network. Most importantly, the derivatives of these networks are networks of the same type, ensuring that these guarantees are applicable also to derivatives of networks of this type.

However, this provides no guarantees about the derivatives with respect to any parameter.

Lu et al., in *Bidirectional Self-Normalizing Neural Networks* prove a similar result: provided that the layers are orthogonal linear transformations that are uniformly distributed on the orthogonal group in the Haar sense followed by activation functions that are Gaussian-Poincaré normalized, meaning that the activation function f satisfies $E[f(Z)^2] = 1$ and $E[f'(Z)^2] = 1$ where Z is the standard normal distribution, and the input vector is thin-shell concentrated in the sense that for all $\epsilon > 0$ $\mathbb{P}\left\{|\frac{1}{n}\|x\|_2^2 - 1| > \epsilon\right\} \to 0$ as $n \to \infty$ then the squared magnitude of the input to each layer is n and the derivative of any loss function E with respect to the input to any layer is the same provided that the layers are wide.

The guarantee that a thin-shell concentrated vector has its norm preserved under forward propagation in a BSNN is comparable to the guarantee that in a SIREN the pre-activations are standard normal distributed, but that the derivative of a loss function with respect to the input to any layer always has the same magnitude is an additional guarantee, which when taken together with the first becomes a much stronger assurance with regard to the trainability of deep BSNNs.

Seeing as deeper networks generally have more representational power it is reasonable to hope that such deeper networks could be trained faster and have fewer parameters.

In order to provide these guarantees also for derivatives of the network it is necessary to ensure that $E[f^{(n)}(Z)^2] = 1$ for all n. One solution is $x \mapsto \sqrt{2}\sin(\frac{\pi}{4} + x)$.

This particular solution is special, indeed it can be seen that if W has the same distribution as $-W$, then $E[f^{(n)}(W)^2] = 1$ for all n.

Theorem 1. *Let W and $-W$ have the same distribution and $f(x) = = \sqrt{2}\sin(\frac{\pi}{4} + x)$, then $E[f^{(n)}(W)] = 1$ for all n.*

Proof. If g is an odd function, i.e. $g(x) = -g(-x)$ then $\mathbb{E}[g(W)] = = \mathbb{E}[-g(-W)] = -\mathbb{E}[g(-W)]$. Since $-W$ has the same distribution as W it follows that $\mathbb{E}[g(W)] = -\mathbb{E}[g(W)] = 0$.

Because $f'' = -f$, it is sufficient to check that $\mathbb{E}[f(W)^2] = \mathbb{E}[f'(W)^2] = 1$ to ensure that $\mathbb{E}[f^{(n)}(W)] = 1$ for all n.

For the first equality $\mathbb{E}[(\sin(Z) + \cos(Z))^2] = E[\sin^2(Z) + 2\sin(Z)\cos(Z) + \cos(Z)^2] = \mathbb{E}[1 + \sin(2Z)] = 1$, since the sine is odd. For second equality $\mathbb{E}[(\cos(Z) - \sin(Z))^2] = \mathbb{E}[\cos(Z)^2 - 2\sin(Z)\cos(Z) + \sin^2(Z)] = \mathbb{E}[1 - \sin(2Z)] = 1$, again since the sine is odd. □

Using an activation function with this property provides a stronger assurance with regard to the stability of forward propagation than that from that $\mathbb{E}[f^{(n)}(Z)] = 1$ where Z is a standard normal distribution. The proof used by Lu et al. to obtain their guarantees uses the fact that the pre-activations are approximately Gaussian when the network is wide. By not relying on this as strongly and instead ensuring that the the output of the activation function has its required properties whenever the pre-activations are symmetric distribution about zero we can we can expect better properties for actual networks and especially for narrow networks, since a distribution will be approximately symmetric long before it is approximately Gaussian.

Positional Encoders

We now have a concrete class of neural networks that can be trained: networks with linear layers of orthogonal linear transformations followed by this special sine activation function but, with the pre-condition that input vectors have squared magnitude equal to their dimension. This, and providing a good positional encoding, is the task of the positional encoder.

Mildenhall et al. in their paper *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis* describe a positional encoder assigning to each co-ordinate the vector $(\sin(2^0\pi p), \cos(2^0\pi p), ..., \sin(2^{L-1}\pi p), \cos(2^{L-1}\pi p))$ where p is the co-ordinate. These vectors have squared magnitude equal to one half their dimension, and by scaling them by $\sqrt{2}$ we would have a vector of the kind we need.

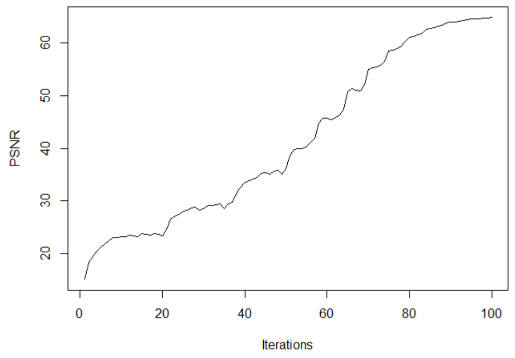
This method was originally applied to five-dimensional input, but when applied to two-dimensional input it introduces obvious patterns, likely present also in five dimensions, in the form of correlations between pixels along lines where one co-ordinate is constant.

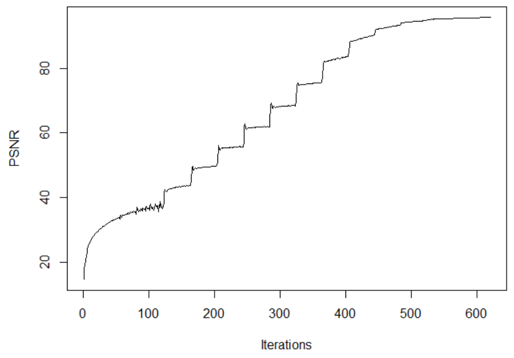
These patterns can be removed by a slight modification of the encoder of Mildenhall et al.: taking the input co-ordinate pair $(x, y)^T$ we construct two additional co-ordinate pairs by rotating co-ordinate pair by one third and two-thirds a turn, to obtain two more $D_{2\pi/3}(x, y)^T$, $D_{4\pi/3}(x, y)^T$ where D_θ denotes a rotation in the x-y plane by θ, and applying the encoder of Mildenhall et al. to each and concatenating the three outputs. Use of this positional encoder avoids line artifacts early in the network fitting and can be seen to improve final mean squared error.

Application to Fitting of Images

Networks of this type, when used with the improved positional encoder can bring MSE between the image and the network output to below 10^{-6} in around 100 iterations and to below 10^{-8} in around 400 iterations, compared to the 10^{-5} achieved by Sitzman et al. in their paper, which required 10000 iterations and 90 minutes on reasonably powerful GPU.

Below are plots of PSNR's, which in this case are $-10\log_{10}(\text{MSE})$ during two training runs, the first is a training run in which the learning rate is reduced aggressively and the second involves less aggressive learning rate reduction.





Source Code and Video

Source code for the experiments and a video showing the fitting process can be found on https://github.com/mlaang/Neural-Networks–All-of-Whose-Derivatives-are-Self-Normalizing-Neural-Networks. The video can also be viewed directly on https://www.youtube.com/watch?v=XYz6ayaKG_g.