Adventures in 2D coordinate networks

Akash Garg

Version 1.0 - January 2, 2023

Coordinate networks are pretty interesting where a small neural network replicates the given input. If a conditioning latent vector provided then the same network to learn / compress lots of data [RHW24]. Coordinate networks also form the basis of the new hotness, NeRFs. A simple setup we will explore:

$$f_\theta(x, y) \mapsto (r, g, b).$$

where f_θ is our neural network, the input coordinates are the pixel coordinates in a given image and the network learns to output the colo values (r, g, b). Following the work of Tancik [Tan23], the basic network is composed of 4 hidden FC layers, each with a width of 128 and ReLU activation function. In code it would look something like this:

```
nn.Sequential(
  nn.Linear(2, 128),
  nn.ReLu(),
  nn.Linear(128, 128),
  nn.ReLu(),
  nn.Linear(128, 128),
  nn.ReLu(),
  nn.Linear(128, 128),
  nn.ReLu(),
  nn.Linear(128, 128),
  nn.ReLu(),
  nn.Linear(128, 3))
  nn.Sigmoid()
```

A standard test image in Figure 1 is used for for all experiments. The image is 512x512 pixels, thus the "dataset" contains 262144 samples. During training, a batch size of 1024 is used (about 256 batches) and training is done for about 200 epochs). The more challenging part of all this was finding a suitable "learning rate" for training, a value of $1.e-3$ seems to work well.

Figure 1: A classical (although controversial) test image used the the source.

The first experiment (Figure 2) normalizes input coordinates between $[0, 1]$, uses a the sigmoid activation function for the output layer to constrain the RGB values to lie between $[0, 1]$.

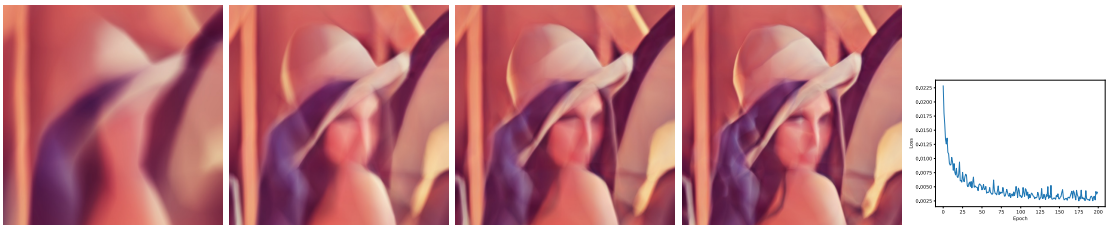

Figure 2: ReLU activation with sigmoid for the final output layer. Left to right: epoch 10, 50, 100 and 200 and loss as a function of epoch. Final accuracy: 0.00403014

One common pitfall during regression tasks is the use of the sigmodi activation function on the output value. Even though output values are RGB and thus restricted to $[0, 1]$, it is common practice to elide the activation function and the netowrk learns the appropriate output range. The sigmoid also squashes it's gradients so we can run into the vanishing gradients problem when using sigmoid. Furthermore, if the ground truth values tend to zero or one, the output of the network would have to be large negative or positive values in order to output zero or one after sigmoid is applied. Such large values can be hard for the network to learn. However, in the current setup, the ground truth values generally don't tend towards those extremes and make little difference if sigmoid is used, see Figure 3.

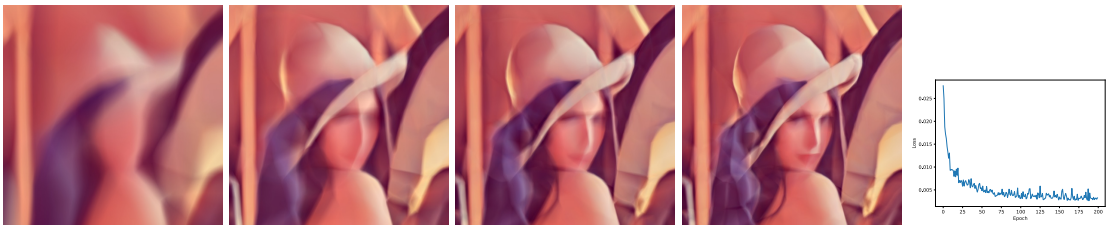

Figure 3: ReLU activations without sigmoid. Left to right: epoch 10, 50, 100 and 200 and loss as a function of epoch. Final accuracy: 0.003259

Another common convention is to use an input range $[-1, 1]$ instead of $[0, 1]$. There could be several good reasons for this:

- Many activation functions, such as tanh or sin, are symmetric around zero. When inputs are normalized to $[-1, 1]$, they better leverage this symmetry, allowing the network to utilize the full dynamic range of the activation functions.
- $[-1, 1]$ can provide a more intuitive representation of space where the center of our domain is at the origin. This can lead to more efficient learning for tasks requiring symmetric or centered representations?
- Neural networks are typically initialized with weights distributed symmetrically around zero. Thus $[-1, 1]$ input range naturally aligns with this initialization.

In practice however, normalization range of the input seems to make little difference.

We can now experiment with different activation functions. Note that teh learning rate is decayed by 50% after every 50 epochs. This creates a much smoother learning curve as indicated when plotting the loss

against epochs. Figure 4 uses the GeLU, Figure 5 uses ELU, and Figure 6 uses tanh. Comparing results for different activations see Figure 7.

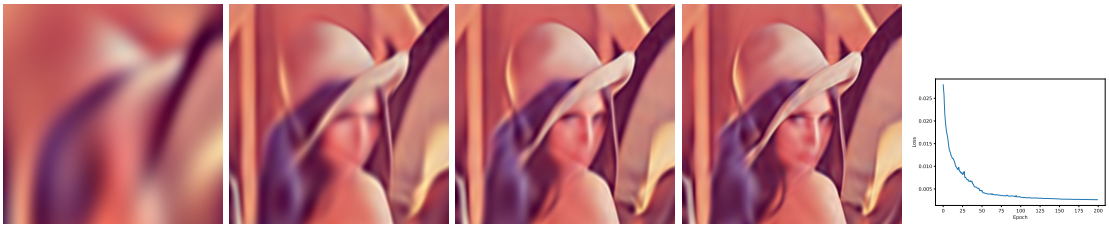

Figure 4: GeLU activations. Left to right: epoch 10, 50, 100 and 200 and loss as a function of epoch. Final accuracy: 0.0026303

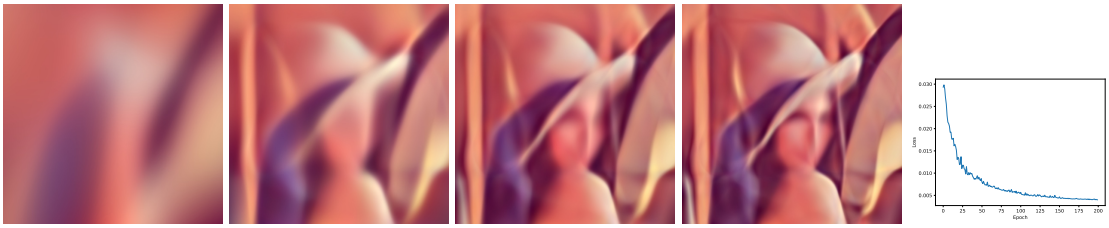

Figure 5: ELU activations. Left to right: epoch 10, 50, 100 and 200 and loss as a function of epoch. Final accuracy: 0.00400449

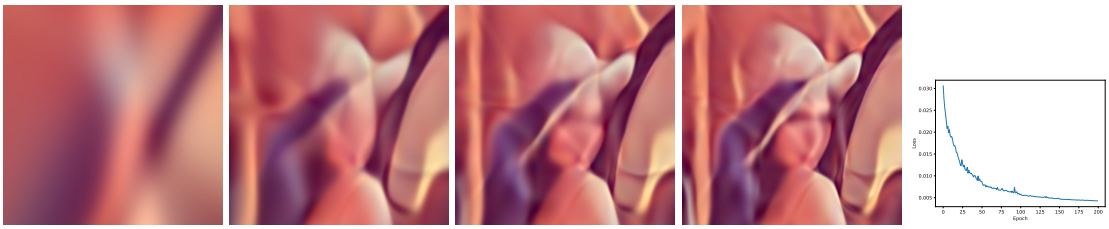

Figure 6: tanh avtivations. Left to right: epoch 10, 50, 100 and 200 and loss as a function of epoch. Final accuracy: 0.00426288

Finally, as Tancik laid out in his thesis [Tan23] we can use a fourier encoding to overcome the spectral bias of neural networks. Using the positional encoding popularized by Vasvani et. al. [Vas+23] the results seem much closer to the original input. Note that we increase the input dimensions from 2 to 20, using 10 frequencies for each x and y coordinate. Results are shown in Figure 8.

The intuitive explanation for why positional encoding fares much better than normal coordinate networks is that the network isn't able to disambiguate between the input coordinate range $[-1, 1]$ very effectively. Positional encoding "spreads" out the input so that the network is able to learn it's features. There is a more theoretical foundation for this using Neural Tangent Kernel [Rah+19], which will need further exploration / understanding. There is also some follow up work like SIREN [Sit+20] and SAPE [Her+21] that is also worth diving into.

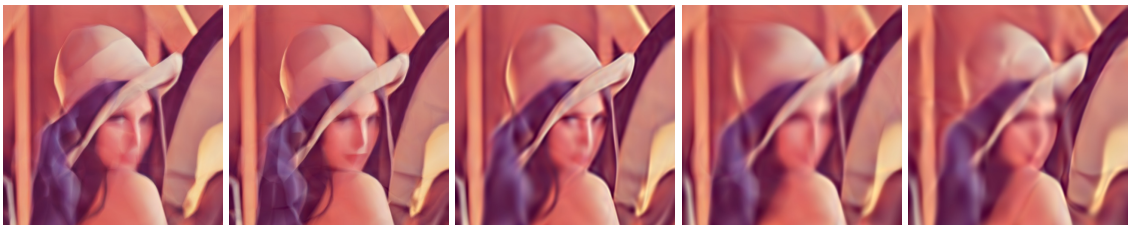

Figure 7: Left to right: ReLU with sigmoid, ReLU without sigmoid, GeLU, ELU, Tanh.

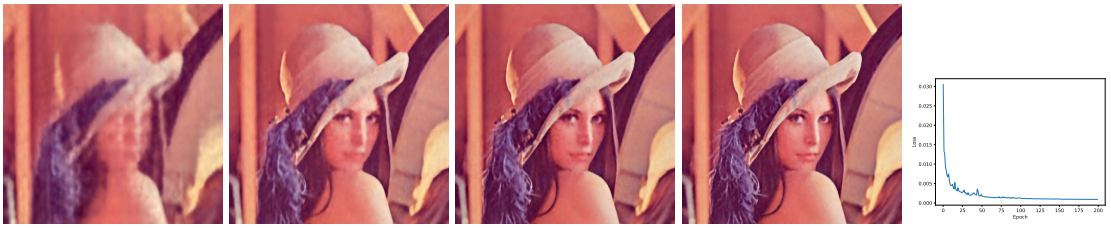

Figure 8: Sinusoidal encoding with ReLU activations. Left to right: epoch 10, 50, 100 and 200 and loss as a function of epoch. Final accuracy: 0.00091497

References

[Her+21] Amir Hertz et al. "SAPE: Spatially-Adaptive Progressive Encoding for Neural Optimization". In: *arXiv preprint arXiv:2104.09125* (2021).

[Rah+19] Nasim Rahaman et al. *On the Spectral Bias of Neural Networks*. 2019. arXiv: 1806.08734 [stat.ML]. URL: https://arxiv.org/abs/1806.08734.

[RHW24] Siyu Ren, Junhui Hou, and Wenping Wang. *NeCGS: Neural Compression for 3D Geometry Sets*. 2024. arXiv: 2405.15034 [cs.CG]. URL: https://arxiv.org/abs/2405.15034.

[Sit+20] Vincent Sitzmann et al. "Implicit Neural Representations with Periodic Activation Functions". In: *Proc. NeurIPS*. 2020.

[Tan23] Matthew Tancik. "Object and Scene Reconstruction using Neural Radiance Fields". PhD thesis. EECS Department, University of California, Berkeley, May 2023. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-128.html.

[Vas+23] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

Revision	Date	Description
1.0	January 2, 2023	Initial draft

Figure 9: Comparing the result of f_θ using positional encoding (left) against our source image (right).