Fourier Feature Networks with Periodic Activation Functions

Bobby Alvarez
Computational Modeling and Data Analytics
Virginia Tech
alvarezbobby9@vt.edu

Abstract

My Project uses a Fourier feature mapping as proposed in *Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains*[9] in order to map the images coordinates to the images RGB values. To further improve this model, I will be using periodic activation functions proposed in *Implicit Neural Representations with Periodic Activation Functions*[7]. Combining these two related techniques should give us an advantage over both individual methods. We will test these methods, training our network on sub-sampled image and comparing the results of the individual, and combined models. To evaluate the models we will look at the peak signal to noise ratio for each iteration, as well as the qualitative features of the reconstructed image.

1 Introduction

In order to perform image regression, we are essentially mapping $\mathbb{R}^2 \rightarrow \mathbb{R}^3$. We have a 2 dimensional coordinate input, x, and y, and a 3 dimensional output representing the RGB values for the coordinate input. Traditional multi-layer perceptions struggle with high frequency detail, so no matter how many training iterations we go through, we still cannot get an accurate mapping of the image without doing some kind of transformation. To extract the high frequency details, we will use our Fourier features transformation to map the coordinates to Fourier features. Our network will be a simple MLP with fully connected layers, however to further improve our model we will use sinusoidal activation functions for our fully connected layers. The *Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains* [9] paper implemented ReLU activation functions for its fully connected layers, however in a more recent paper *Implicit Neural Representations with Periodic Activation Functions* [7] it was found that sinusoidal activation layers are better at coordinate mapping than traditional activation layers. The combination of both of these methods should give us a fairly realistic reconstruction of the image without many training iterations. Although there are more interesting use cases for this type of model such as shape regression or frame injection. Image regression is a good analog of the performance and underlying structure of more complex applications. We can quantify the error of our model with a metric called peak signal to noise ratio,

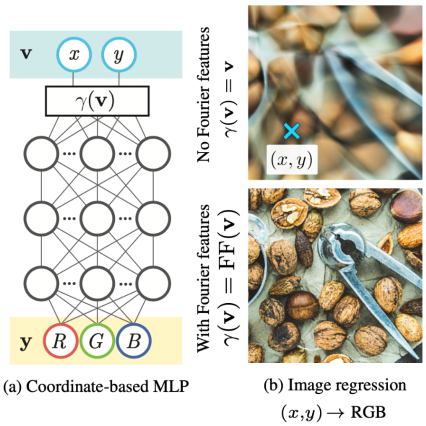

Figure 1: From [9]

we can obtain this value for both the testing data and the training data. This metric is obtained by making a prediction on the training inputs, then calculating the MSE for either the testing or training data, and finally doing a transformation on that value.

2 Related work

In *Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains*[9], the authors proposed the original idea for using a Fourier feature mapping and its relationship with an MLP's neural tangent kernel. The authors used different transformations on the Fourier feature mapping in order to "tune" the neural tangent kernel's spectrum. The paper proposed this example for many different spatial mapping use cases such as, computed tomography, magnetic resonance imaging, as well as image and shape regression. A large focus of the paper was how Fourier feature mapping can overpower the spectral bias [6] of non-mapped MLPs, enabling them to learn higher frequency details. The other paper, *Implicit Neural Representations with Periodic Activation Functions* [7], looked at periodic activation functions for implicit neural representations and introduced sinusoidal representation networks or SIRENs. This paper originally looked to solve similar problems regarding spatial data, as well as SIRENS spatio-temporal applications as well. Interestingly the derivative of a SIREN is a SIREN, because the derivative of a sine function is a cosine, or a phase shifted sine. Without knowing about the paper I did try to use a sine activation function by itself. But that did not work great, however the paper goes over optimal weight initialization, which greatly improved performance. In a follow up paper to *Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains* named *Learned Initializations for Optimizing Coordinate-Based Neural Representations* [8], the authors used meta-learning techniques to improve model training time with techniques such as model-agnostic meta-learning, or MAML [3], and the improved technique, Reptile [5].

3 Dataset and Features

For this project, we used the DIV2K [2] dataset. We specifically looked at training image one, an image of a starfish. For the uses of our project we down-scaled the image by a factor of four. For the rest of our data we used an image of a fox to compare the methods, as this was the standard for the Fourier features paper and the smaller image let the code run much faster.

4 Methods

For traditional image regression, the training data fed into out MLP is simply the full-sized image coordinates, and corresponding RGB values. However for our method, we will be training our network with a half-sampled image, where we take every other element throughout the rows and columns. Next we need to construct our network, where the first layer layer is the Fourier feature map, followed by a set amount of fully connected layers. To apply our Fourier feature map, we have v as an input coordinate. Thus our Fourier feature map is given by:

$$\gamma(v) = [cos(2\pi Bv) \quad sin(2\pi Bv)]$$

where $B \in \mathbb{R}^{mxd}$ is a Gaussian matrix distributed by $\mathcal{N}(0, \sigma)$. We apply this map to every point in the training data. Note that when we apply the mapping, we are essentially doubling the input size. This means that our first fully connected layer takes two times the number of Fourier features, and outputs how ever many `out_features` is specified. The rest of the fully connected layers takes in an outputs a size of `out_features`. The final layer takes in `out_features` for the number of input features, and outputs three output features. For our fully connected layers, we will use periodic activations from the SIREN paper where each input is multiplied by a weight matrix and has a bias applied to it. That transformation is then inputted into a sinusoidal function for that given layer. This is essentially a normal MLP, but instead of using something like a ReLU activation function, we apply periodic non-linearity with a sine function. In order for this to work well, we need to carefully choose our initial weights. Our weight matrix will be uniformly distributed with $\mathcal{U}(-\sqrt{c/n}, \sqrt{c/n})$. For the first layer we will multiply our weight and bias transformation by thirty as the referenced paper claimed these initialization were the most optimal. For the loss function, mean square error between the model output and the RGB training data. We then update the network and repeat for

the number of training iterations. For our metric peak signal to noise ratio, we will use this formula: $PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right)$, where MAX_I is the maximum possible pixel value of the image.

5 Results

For our performance tests, we decided to use the fox image [1] as our training time was relatively fast and we got sufficient data from our methods. Using a larger image may have obscured the results. In order to compare results we trained three separate networks, one with only Fourier feature maps, one with only SIREN, and finally one with both Fourier feature maps and SIREN. We trained each network on a down-scaled 344×344 fox image. We had three fully connected hidden layer for all networks, each with 344 input and output features. Each network took in 2 input features, x and y coordinates, and outputted 3 features representing RGB values. For the networks that used Fourier feature mapping, the scaling factor for the Gaussian matrix was chosen to be 16 as that gave us the most optimal bias versus variance trade off. We used the Adam optimizer for our training processes, and mean-squared error for our loss functions. Each network was trained for 500 iterations. The model with only Fourier feature mapping took 16 seconds to train, the model with just SIREN took only 14 seconds to train, and the combined model took 22 seconds to train.

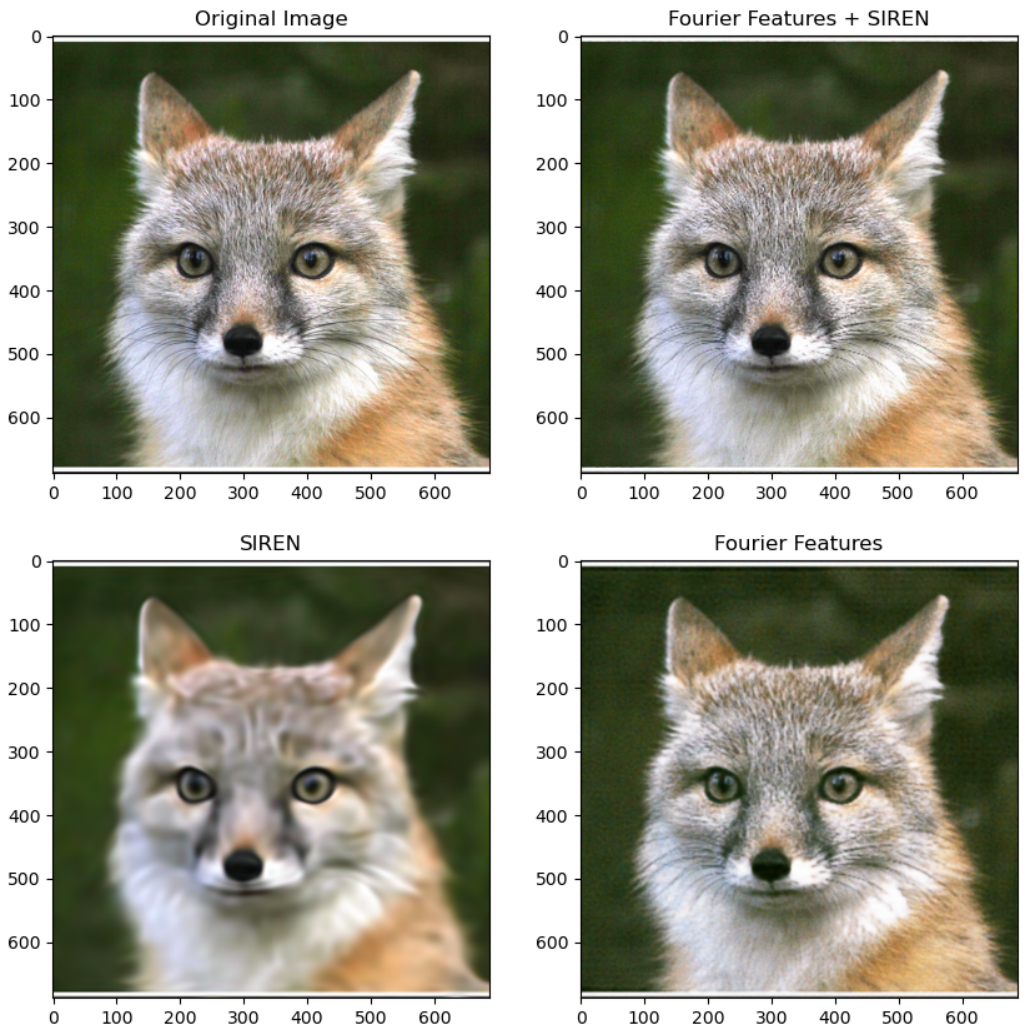

Figure 2: Comparing Qualitative Performance on 500 Iterations

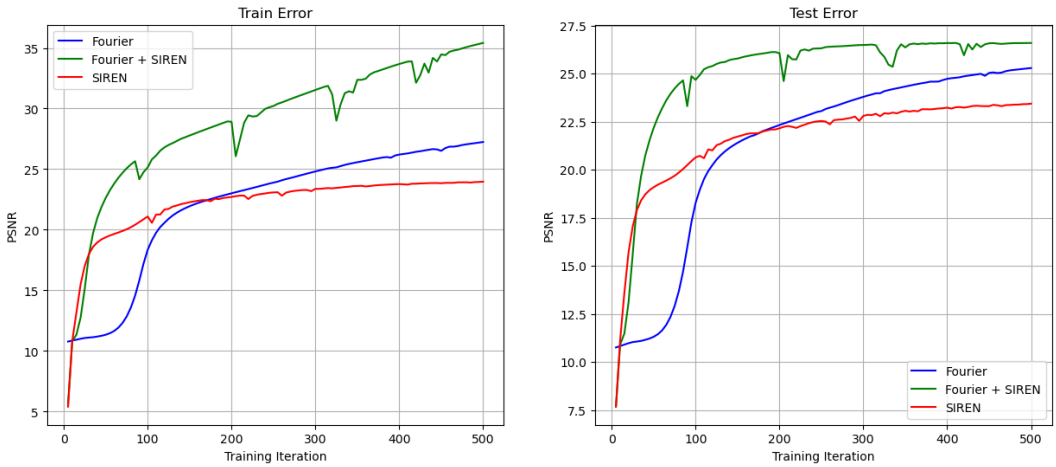

Figure 3: Comparing Network Error on 500 Iterations

For our example with the DIV2K starfish image, we decided to heavily down-sample the image, by a factor of 8 to really stretch the model's capabilities. We used the same three fully connected hidden layer model as before, but this time with 300 input and output channels for the hidden layers. We also used the same scale size of 16, and the same amount of training iterations at 500.

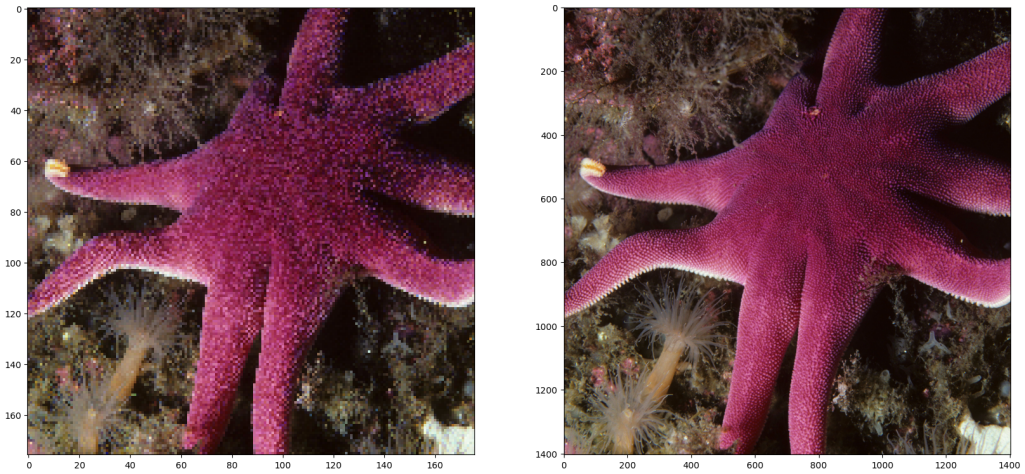

Figure 4: DIV2K Image Example

Figure 5: Reconstructed Images

These tests were done on a Windows Desktop on Python 3.8.8 with IPython 8.12.2 through Jupyter Notebooks. For hardware we used an NVIDIA RTX 3080 10GB GPU, an AMD Ryzen 9 5900X CPU, and 32GB of RAM.

6 Conclusion and discussion

As we can see in figure 2, our model that used Fourier feature mapping and SIREN had the best performance as we expected. The Fourier feature mapping only model also did pretty well, but as we can see in the picture there is still a fair amount of noise left in the approximation. Finally as we can see with the model only using SIREN, we did not have a great fit. This is likely due to a poor high frequency performance with low training iterations. Note that with more training iterations we will get a fit much closer to our combined model. This is the advantage of our combined model, the overall model performance with less training iterations. In figure 3, we can see how our peak signal to noise ratio quantifies what we are seeing in the qualitative plots. For the training error, the Fourier feature network is initialized with high PSNR value, but takes a long time for the PSNR to plateau when compared to the rest of the models. The training error for the SIREN model seems to converge quite quickly, even outpacing the combined model in the beginning, however we converge to a relatively small PSNR compared to the models. For the combined model, we can confirm that we this is the more advantageous model when it comes to the training error. Even at 500 iterations, the model is improving at a rate faster than the other models. Our plots for the testing error show a slightly different, but overall similar picture. The testing error of the Fourier feature network is slow to plateau, but is catching up to testing error of the combined model. For the SIREN model, the testing error converges rather quickly again, but not surprisingly to a lower value. And for the combined model we have very quick convergence.

For the starfish model we wanted to compare this another reconstruction technique known as bi-cubic interpolation. As we can see from figure **??**, the results are quite similar. It seems as if our combined method edges out the bi-cubic interpolation ever so slightly, especially in the high frequency details of the coral behind the starfish.

7 Future Work

In continuation of this work there are many applications to try this method on, this includes but is not limited to: computed tomography, shape regression, MRI reconstruction, and frame injection. Another use case is potentially using this combined method as an up-scaling method instead of using a method like bi-cubic or bi-linear interpolation for a VDSR [4] architecture. We can use methods such as MAML [3] or Reptile [5] in order to significantly decrease the training time, and push the model towards one-shot or few-shot learning. As discussed in the related works section, there is a paper [8] from the same authors as *Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains* [9], that uses meta-learning techniques specifically for spatial mapping model with very compelling results.

References

[1] Image from flickr. Accessed on 5th May 2024.

[2] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[3] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017.

[4] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks, 2016.

[5] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms, 2018.

[6] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks, 2019.

[7] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions, 2020.

[8] Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P. Srinivasan, Jonathan T. Barron, and Ren Ng. Learned initializations for optimizing coordinate-based neural representations, 2021.

[9] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains, 2020.