A PROOFS AND DERIVATIONS

We begin by stating the general condition under which an MFN can ultimately be expressed as a linear combination of the original filter function: the fact that elementwise multiplication of the bases results in a *sum* of the same bases with different parameters.

Definition 1. *A filter $g(x;\theta)$ satisfies the multiplicative sum property if for some $\theta^{(1)}$ and $\theta^{(2)}$ we have that*

$$g(x;\theta^{(1)}) \circ g(x;\theta^{(2)}) = \sum_{i=1}^{N} \beta_i g(x;\bar{\theta}^{(i)}) \tag{10}$$

for some alternative set of parameters $\bar{\theta}^{(1:N)}$ and coefficients $\beta_{1:N}$.

We now state the general theorem that if the filter used in an MFN satisfies the multiplicative sum property, then the function represented by the MFN can also be expressed as a linear combination of the filter bases applies to the input.

Theorem 3. *Let f by an MFN defined by the recurrence*

$$\begin{aligned} z^{(1)} &= g\left(x;\theta^{(1)}\right) \\ z^{(i+1)} &= \left(W^{(i)}z^{(i)} + b^{(i)}\right) \circ g\left(x;\theta^{(i+1)}\right),\ i = 1,\dots,k-1 \\ f(x) &= W^{(k)}z^{(k)} + b^{(k)} \end{aligned}$$

with a filter g that satisfies the multiplicative sum property. Then f can be expressed as

$$f_j(x) = \sum_{t=1}^{T} \alpha_t g(x;\tau_j^{(t)}) + c. \tag{11}$$

where we use the notation $g(x;\tau_j^{(t)})$ to denote a scalar-valued application of the filter, specific to the jth coordinate.

Proof. The proof follows by induction on i. The base case of $z^{(1)}$ follows immediately from the definition (and in fact without any bias term b). Now suppose that for layer $i-1$, for each $z_j^{(i-1)}$ we have

$$z_j^{(i-1)} = \sum_{t=1}^{N} \alpha_t^j g(x;\tau_j^{(t)}) \tag{12}$$

Then we have that

$$z_j^{(i)} = \left(\sum_{p=1}^{d_i} W_{jp}^{(i)} z_p^{(i-1)} + b^{(i)}\right) g(x;\theta_j^{(i)}) \tag{13}$$

$$= \left(\sum_{p=1}^{d_i} W_{jp}^{(i)} \sum_{t=1}^{T} \alpha_t^p g(x;\tau_p^{(t)}) + b_j^{(i)}\right) g(x;\theta_j^{(i)}) \tag{14}$$

$$= \sum_{p=1}^{d_i}\sum_{t=1}^{T} W_{jp}^{(i)}\alpha_t^p g(x;\tau_p^{(t)}) g(x;\theta_j^{(i)}) + b_j^{(i)} g(x;\theta_j^{(i)}) \tag{15}$$

$$= \sum_{p=1}^{d_i}\sum_{t=1}^{T} W_{jp}^{(i)}\alpha_t^p \sum_{q=1}^{N} \beta_q g(x;\bar{\tau}_p^{(t,q)}) g(x;\theta_j^{(i)}) + b_j^{(i)} g(x;\theta_j^{(i)}) \tag{16}$$

$$= \sum_{t=1}^{T'} \bar{\alpha}_t g(x;\bar{\tau}_j^{(t)}) \tag{17}$$

where the line (15) to (16) follows from the multiplicative sum instruction. The final output $f(x)$ is simply a linear function of the final hidden layer $z^{(k)}$, and thus also satisfies the condition of the theorem. □

We now prove the multiplicative sum property for both the Fourier and Gabor filters. This, together with the theorem above, proves the results in the main text, stated for the FourierNet and GaborNet functions explicitly.

A.1 FOURIER NETWORKS

Theorem 4. *The Fourier basis function*

$$g(x;\theta^{(i)}) = \sin(\omega^{(i)}x + \phi^{(i)}) \tag{18}$$

satisfies the multiplicative sum property with β *and* $\bar{\theta}^{(1:2)} = \left\{\bar{\omega}^{(1:2)}, \bar{\phi}^{(1:2)}\right\}$ *given bvy*

$$\begin{aligned}
\beta_1 &= \beta_2 = \frac{1}{2} \\
\bar{\omega}^{(1)} &= \omega^{(1)} - \omega^{(2)} \\
\bar{\omega}^{(2)} &= \omega^{(1)} + \omega^{(2)} \\
\bar{\phi}^{(1)} &= \phi^{(1)} - \phi^{(2)} - \frac{\pi}{2} \\
\bar{\phi}^{(2)} &= \phi^{(1)} + \phi^{(2)} + \frac{\pi}{2}.
\end{aligned} \tag{19}$$

Proof. This fact follows from simple rules for sum of sinusoidal function, and was also highlighted in the main text

$$\begin{aligned}
g(x;\theta^{(1)}) \circ g(x;\theta^{(2)}) &= \sin(\omega^{(1)}x + \phi^{(1)}) \circ \sin(\omega^{(2)}x + \phi^{(2)}) \\
&= \frac{1}{2}\cos\left((\omega^{(1)} - \omega^{(2)})x + \phi^{(1)} - \phi^{(1)}\right) - \\
&\quad \frac{1}{2}\cos\left((\omega^{(1)} + \omega^{(2)})x + \phi^{(1)} + \phi^{(1)}\right) \\
&= \frac{1}{2}\sin\left((\omega^{(1)} - \omega^{(2)})x + \phi^{(1)} - \phi^{(1)} - \frac{\pi}{2}\right) + \\
&\quad \frac{1}{2}\sin\left((\omega^{(1)} + \omega^{(2)})x + \phi^{(1)} + \phi^{(1)} + \frac{\pi}{2}\right) \\
&= \beta_1 g(x;\bar{\theta}^{(1)}) + \beta_2 g(x;\bar{\theta}^{(2)})
\end{aligned} \tag{20}$$

with the parameters defined as in the theorem statement. □

An inspection of the two theorems above directly leads to the explicit expansion of the terms in the sequence as well, which we restate here.

Corollary 2. *Let* $i_1, i_2, \ldots, i_{k-1}$ *range over all* $\prod_{j=1}^{k-1} d_j$ *possible indices of each hidden unit of each layer of an MFN, and let* $s_2, \ldots, s_k \in \{-1, +1\}$ *range over all* 2^{k-1} *possible binary signs; then the expansion of* $z_{i_k}^{(k)}$ *given is given by all the terms*

$$\begin{aligned}
\bar{\alpha} &= \left\{\frac{1}{2^{k-1}} W_{i_k,i_{k-1}}^{(k-1)} \ldots W_{i_3,i_2}^{(2)} W_{i_2,i_1}^{(1)}\right\} \\
\bar{\omega} &= \left\{s_k\omega_{i_k}^{(k)} + \ldots + s_2\omega_{i_2}^{(2)} + \omega_{i_1}^{(1)}\right\} \\
\bar{\phi} &= \left\{s_k\phi_{i_k}^{(k)} + \ldots + s_2\phi_{i_2}^{(2)} + \phi_{i_1}^{(1)} + \frac{\pi}{2}\sum_{i=2}^{k} s_k\right\}.
\end{aligned} \tag{21}$$

with a similar form for terms that begin at the $i > 1$ *layer, multiplied by the corresponding* $b_{i_j}^{(j)}$ *term.*

A.2 GABOR NETWORKS

We now prove similar properties for the Gabor Network.

Theorem 5. *The Gabor basis function*

$$g_j(x;\theta^{(i)}) = \exp\left(-\frac{\gamma_j^{(i)}}{2}\left\|x-\mu_j^{(i)}\right\|_2^2\right)\sin\left(\omega_j^{(i)}x+\phi_j^{(i)}\right) \tag{22}$$

satisfies the multiplicative sum property with β and $\bar{\theta}^{(1:2)} = \left\{\bar{\gamma}^{(1:2)}, \bar{\mu}^{(1:2)}, \bar{\omega}^{(1:2)}, \bar{\phi}^{(1:2)}\right\}$ given by

$$\begin{aligned}
\beta_1 = \beta_2 &= \frac{1}{2}\exp\left(\frac{-\gamma^{(1)}\gamma^{(2)}\|\mu^{(1)}-\mu^{(2)}\|^2}{2(\gamma^{(1)}+\gamma^{(2)})}\right)\\
\bar{\gamma}^{(1)} = \bar{\gamma}^{(2)} &= \gamma^{(1)}+\gamma^{(2)}\\
\bar{\mu}^{(1)} = \bar{\mu}^{(2)} &= \frac{\gamma^{(1)}\mu^{(1)}+\gamma^{(2)}\mu^{(2)}}{\gamma^{(1)}+\gamma^{(2)}}\\
\bar{\omega}^{(1)} &= \omega^{(1)}-\omega^{(2)}\\
\bar{\omega}^{(2)} &= \omega^{(1)}+\omega^{(2)}\\
\bar{\phi}^{(1)} &= \phi^{(1)}-\phi^{(2)}-\frac{\pi}{2}\\
\bar{\phi}^{(2)} &= \phi^{(1)}+\phi^{(2)}+\frac{\pi}{2}.
\end{aligned} \tag{23}$$

Proof. The sinusoidal terms in the Gabor filter proceed in the exact same manner as for the Fourier filter. The exponential terms follow from the fact, well known from Gaussian distributions, that

$$\begin{aligned}
&\exp\left(-\frac{1}{2}\gamma^{(1)}\|x-\mu^{(1)}\|_2^2\right)\exp\left(-\frac{1}{2}\gamma^{(2)}\|x-\mu^{(2)}\|_2^2\right) =\\
&\exp\left(\frac{-\gamma^{(1)}\gamma^{(2)}\|\mu^{(1)}-\mu^{(2)}\|^2}{2(\gamma^{(1)}+\gamma^{(2)})}\right)\exp\left(-\frac{1}{2}(\gamma^{(1)}+\gamma^{(2)})\left\|x-\frac{\gamma^{(1)}\mu^{(1)}+\gamma^{(2)}\mu^{(2)}}{\gamma^{(1)}+\gamma^{(2)}}\right\|_2^2\right)
\end{aligned} \tag{24}$$

The first term is a constant, and so can be folded into the β terms, while the second is the exponential term of a Gabor filter with the parameters given above. □

Finally, just as for the FourierNet, we can also provide precise characterizations of each term in the final Gabor filter expansion. This corollary follows immediately from the proof above, plus some algebraic simplification.

Corollary 3. *Let $i_1, i_2, \ldots, i_{k-1}$ range over all $\prod_{j=1}^{k-1} d_j$ possible indices of each hidden unit of each layer of an MFN, and let $s_2, \ldots, s_k \in \{-1, +1\}$ range over all 2^{k-1} possible binary signs; then the expansion of $z_{i_k}^{(k)}$ given is given by all the terms*

$$\begin{aligned}
\bar{\alpha} &= \left\{\frac{1}{2^{k-1}}W_{i_k,i_{k-1}}^{(k-1)}\ldots W_{i_3,i_2}^{(2)}W_{i_2,i_1}^{(1)}\exp\left(-\frac{\sum_{p=1,q\neq p}^{k}\gamma_{i_q}^{(q)}\gamma_{i_p}^{(p)}\|\mu_{i_p}^{(p)}-\mu_{i_q}^{(q)}\|_2^2}{2\sum_{p=1}^{k}\gamma_{i_p}^{(p)}}\right)\right\}\\
\bar{\gamma} &= \left\{\gamma_{i_k}^{(k)}+\ldots+\gamma_{i_2}^{(2)}+\gamma_{i_1}^{(1)}\right\}\\
\bar{\mu} &= \left\{\frac{\gamma_{i_k}^{(k)}\mu_{i_k}^{(k)}+\ldots+\gamma_{i_2}^{(2)}\mu_{i_2}^{(2)}+\gamma_{i_1}^{(1)}\mu_{i_1}^{(1)}}{\gamma_{i_k}^{(k)}+\ldots+\gamma_{i_2}^{(2)}+\gamma_{i_1}^{(1)}}\right\}\\
\bar{\omega} &= \left\{s_k\omega_{i_k}^{(k)}+\ldots+s_2\omega_{i_2}^{(2)}+\omega_{i_1}^{(1)}\right\}\\
\bar{\phi} &= \left\{s_k\phi_{i_k}^{(k)}+\ldots+s_2\phi_{i_2}^{(2)}+\phi_{i_1}^{(1)}+\frac{\pi}{2}\sum_{i=2}^{k}s_k\right\}.
\end{aligned} \tag{25}$$

with a similar form for terms that begin at the $i > 1$ layer, multiplied by the corresponding $b_{i_j}^{(j)}$ term.

B Additional Experimental Details and Results

B.1 Image Representation

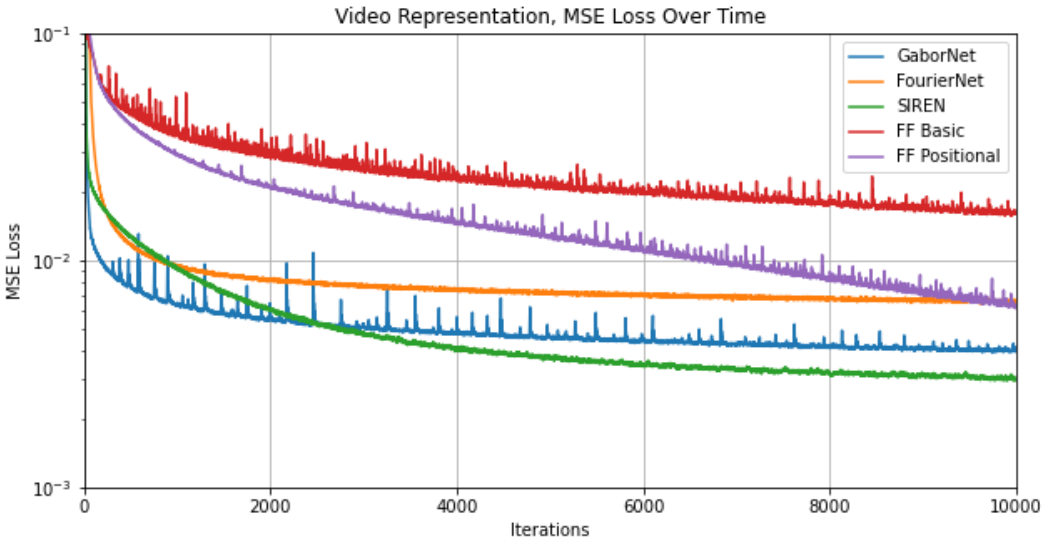

Figure 6: Plot of training loss (mean squared error) over time for models in the video representation task.

In the image representation task, all models used were 3 layers with 256 hidden units per layer, and were trained for 10000 iterations. All models were trained with the Adam optimizer, with a learning rate of 10^{-2} for the two MFNs and a learning rate of 10^{-4} for all other models. Both MFN models used an input scaling factor of 256.

The cat video used in the video representation task is available at the following url:

`https://www.pexels.com/video/the-full-facial-features-of-a-pet-cat-3040808`

Models used for video representation were 3 layers with 1024 hidden units per layer, and were trained for 10000 iterations. We performed an exponentially grid search over learning rates for each model, testing a learning rate of 10^{-i} for $i \in \{2, 3, 4, 5, 6\}$, and found the best performance when using $1e-5$ for SIREN and 10^{-3} for all other models. Both MFN architectures used a scaling factor γ of 256.

B.2 Image Generalization

In the image generalization experiments, we use two datasets (*Natural* and *Text* datasets); each contains 16 images with a size of 512 $\times$ 512 pixels. We use 25% of the pixels resulting 256 $\times$ 256 images to train the models and use the full image size in the evaluation. We construct and train the models separately for each image. The models are trained to map the input of (x, y) coordinates to the corresponding RGB values. We train all the models in 2000 epochs. The final result is computed by averaging the results over the 16 images in each dataset. For each of the MFNs and Fourier feature networks, we construct a 4 hidden layer network with 256 nodes in each hidden layer.

For both FourierNet and GaborNet, we set the input scaling factor to 256, with the value of α in GaborNet set to 3.0. We use Adam optimizer with learning rate of 3e-3 and no weight decay. For the Fourier feature models, we use the hyperparameters presented in the code repository published by the authors. The optimizer used in the Fourier feature models is the Adam optimizer with learning rate of 1e-3 and no weight decay. All the Fourier feature models use 256 as the embedding size for the random features. The scaling factor for the Fourier feature networks with positional encoding is set to 6 for the *Natural* dataset and 5 for the *Text* dataset. The scaling factor for the Fourier feature networks with Gaussian random features is set to 10 for the *Natural* dataset

Figure 7: Image generalization samples from *Natural* datasets.

and 14 for the *Text* dataset. As in the code implementation published by (Tancik et al., 2020), we implement our model in Python using JAX as the deep learning framework.

In addition to the experimental results presented in Section 4.1, we show additional generated images for *Natural* dataset in Figure 7 and *Text* dataset in Figure 8. We can similarly see from the figure that the quality of generated images by the MFNs are superior to the baselines in *Text* dataset and competitive in the *Natural* dataset. The Fourier feature networks (especially with basic and positional encoding) failed to generate some parts of the texts in the images. The use of Gaussian random features improves the generated images, but still fails to generate small portions of the texts. In contrast, both FOURIERNET and GABORNET correctly generate all parts of the texts. Figure 8 also highlights the parts of the texts that are failed to be generated by the baselines methods. In the *Natural* dataset, all of the models except the Fourier feature networks using basic encoding performs equally well with very similar visual results.

Comparison with the traditional Fourier transform. In addition to the experiments above, we also run the standard upsampling technique that performs FFT on the 256×256 pixels images and then performs inverse FFT on the extended frequency domains that outputs 512×512 pixels images. We then measure the PSNR on the unseen pixels, similar to the experiments on the MFNs and Fourier feature networks. This traditional FFT upsampling method performs quite well on both *Natural* and *Text* dataset with 73.80 ± 12.20 and 66.78 ± 8.14 PSNRs respectively. However, the standard Fourier analysis on the traditional upsampling methods serves a different purpose compared to the one in the implicit neural representations (the MFNs, SIRENs, and Fourier feature networks). The standard Fourier analysis does not scale well to higher dimensions, or it does not admit to backpropagation through more complex loss functions (e.g., in neural rendering fields). The lack of scalability and non-differentiability hinders the usage of FFT based upsampling methods in an end-to-end manner and hence adds significant barriers to such methods being employed in generic tasks. In contrast, the MFNs (and other implicit neural representations) provides generic methods that apply to generic tasks where we can learn based upon derivatives, as shown in our paper and the related works (Sitzmann et al., 2020; Tancik et al., 2020; Mildenhall et al., 2020).

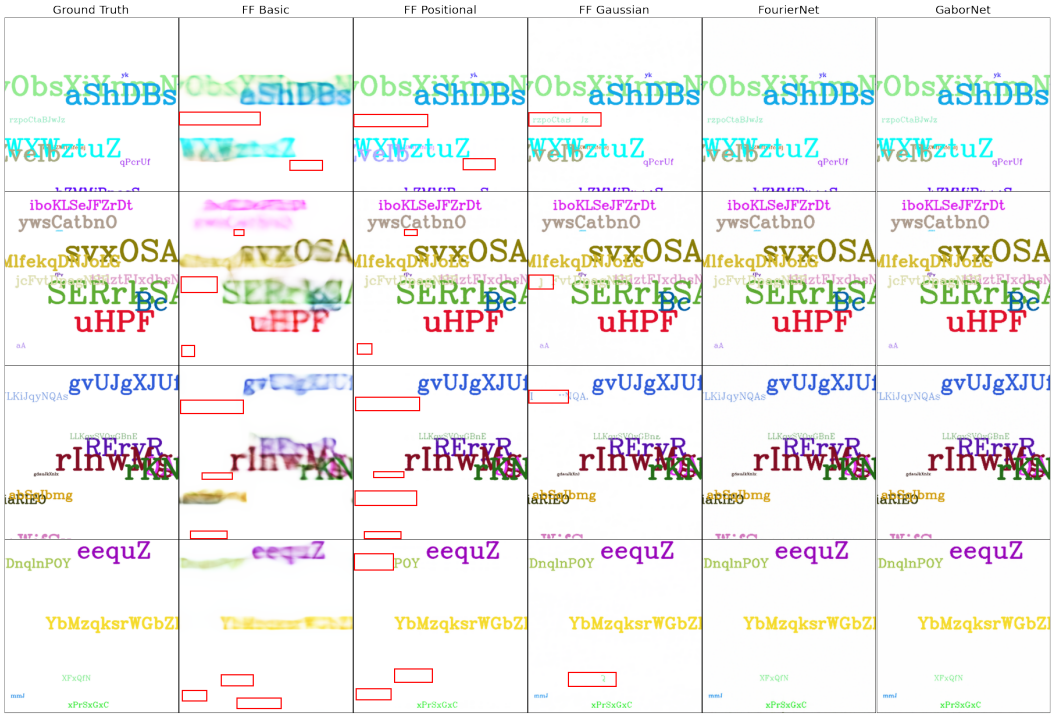

Figure 8: Image generalization samples from *Text* datasets.

B.3 Differential Equations

B.3.1 Poisson Equation

For the image reconstruction tasks using the supervision of gradients and Laplacians of the ground truth, we use 5 layer networks with a hidden size of 256. The networks were trained using Adam Optimizer. For the image reconstruction task using gradients, the learning rate chosen for FourierNet and GaborNet was 1e-3, which was chosen by grid search. The batch size was chosen to be 16384, as in SIREN. For SIREN, we used the reference implementation of it, which also uses 5 layer networks with a hidden size of 256. The same architecture was also used for ReLU MLP. All the networks were trained for 10000 iterations. For the image reconstruction task using Laplacians, the learning rates chosen for FourierNet and GaborNet were 5e-3 and 3.5e-3 respectively, which was chosen by grid search. The batch size was chosen to be 16384, same as SIREN. For SIREN, we used the reference implementation of it. All the networks were trained for 10000 iterations.

B.3.2 Helhmholtz and Wave Equation

For the single source inversion task corresponding to the Helmholtz equation, we use 5 layer networks for both FourierNet and GaborNet with a hidden size of 256. The networks pertaining to SIREN, ReLU MLP, FourierNet and GaborNet are trained using randomly sampled points from a ℓ_∞ball $\in \mathbb{R}^2$ of radius 1. The networks were trained using Adam optimizer. The learning rates chosen for FourierNet and GaborNet were 2.5e-4 and 5e-4 respectively, which was chosen by grid search. The batch size was chosen to be 32, same as SIREN. For SIREN, we used the reference implementation of it, which also uses 5 layer networks with a hidden size of 256. The same architecture was also used for ReLU MLP. All the networks were trained for 50000 iterations. For the wave equation, we use 5 layer networks for both FourierNet and GaborNet with a hidden size of 512. The networks were trained using Adam optimizer. The learning rates is chosen for FourierNet and GaborNet were 1e-4 and 2e-4 respectively, which was chosen by grid search. The batch size was chosen to be 32, same as SIREN. For SIREN, we used the reference implementation of it, which also uses 5 layer networks with a hidden size of 512. All the networks were trained for 10000 iterations.

B.4 SHAPE REPRESENTATION AND SDF FITTING

The 3D room oriented point cloud used in our shape representation is freely available at

`http://www.turbosquid.com`

The loss function used to train models on this task is identical to the one presented in Sitzmann et al. (2020); namely, for a model N training on points $x \in \Omega$ in an oriented point cloud with surface Ω_0, we use

$$L = \sum_{x \in \Omega} \lambda_1 \left\| 1 - |\nabla_x N(x)| \right\| + \sum_{x \in \Omega_0} \lambda_2 \|N(x)\| + \lambda_3 \left(1 - \langle \nabla_x N(x), n(x) \rangle\right) + \sum_{x \notin \Omega_0} \lambda_4 \exp(-\alpha |N(x)|) \tag{26}$$

where $n(x)$ is the normal of the SDF f (i.e. $n(x) = \nabla_x f(x)$, and we use the following prescribed values for the scaling factors of the various terms: $\lambda_1 = 50$, $\lambda_2 = 3000$, $\lambda_3 = 100$, $\lambda_4 = 3000$, and $\alpha = 100$.

Models used to fit SDFs were 5 layers with 512 hidden units per layer, and were trained for 50 epochs (roughly 175k iterations). We performed a grid search to tune the learning rate for each model, searching over $\{a \times 10^{-i} \mid a \in \{1, 5\}, i \in \{2, 3, 4, 5, 6\}\}$. This revealed best performance for FOURIERNET at 10^{-4}, GABORNET and ReLU at 5×10^{-4}, and SIREN at 10^{-5}. Input scaling factors used for FOURIERNET and GABORNET were 128 and 256, respectively.

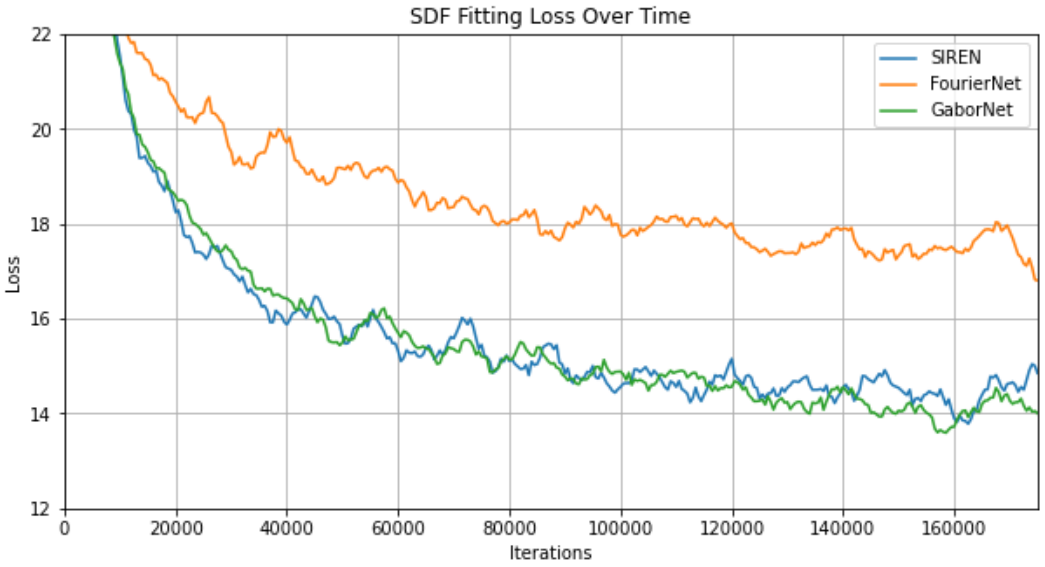

Figure 9: Plot of training losses (Equation (26)) over time for SIREN, GaborNet, and FourierNet.

B.5 3D INVERSE RENDERING FOR VIEW SYNTHESIS

In the view synthesis task, we aim to reconstruct 3D representation from the observed 2D photographs. The input to the models is a 3D coordinate (x, y, z) of a viewpoint. The models need to predict the corresponding 4D vector (r, g, b, v) where (r, g, b) corresponds to the predicted color in RGB format, and v corresponds to the volume density. The outputs of the models are then used as the input for the volumetric rendering procedure, which outputs the final rendered images from the viewpoint. The loss is computed as the mean squared error between the generated images and the observed 2D photograph. Since the rendering procedure is fully differentiable, all the training can be done end-to-end.

The *Lego* dataset in the "simplified NeRF" task contains 2D photographs from 120 viewpoints downsampled to 400 $\times$ 400 pixel resolution. We split the datasets into 100 training images, 7 validation

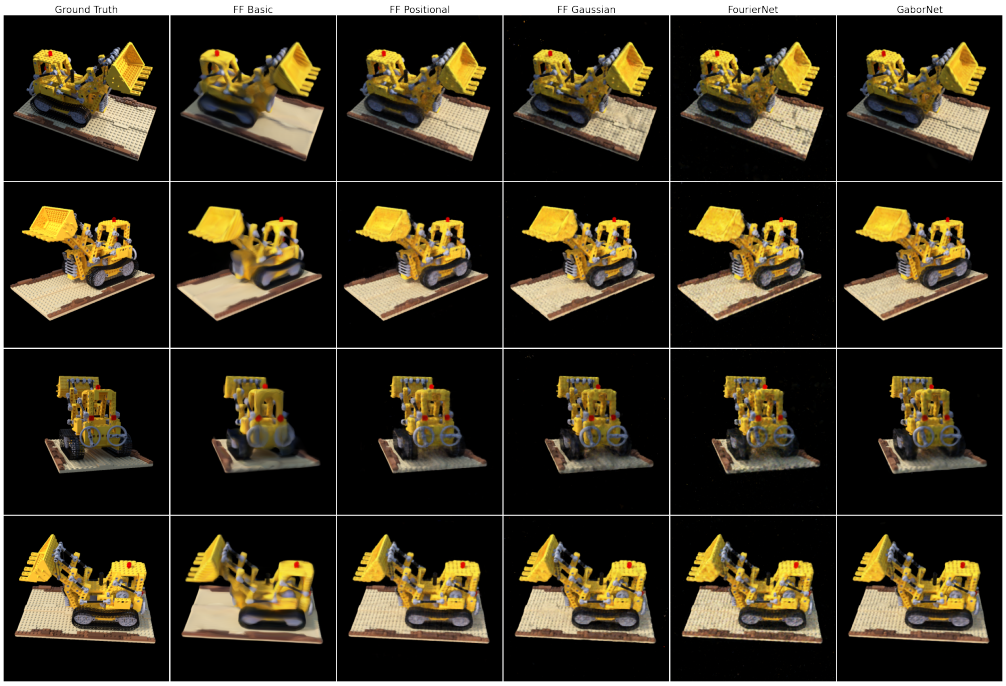

Figure 10: Additional examples of 2D rendered photographs from the *Lego* dataset experiment.

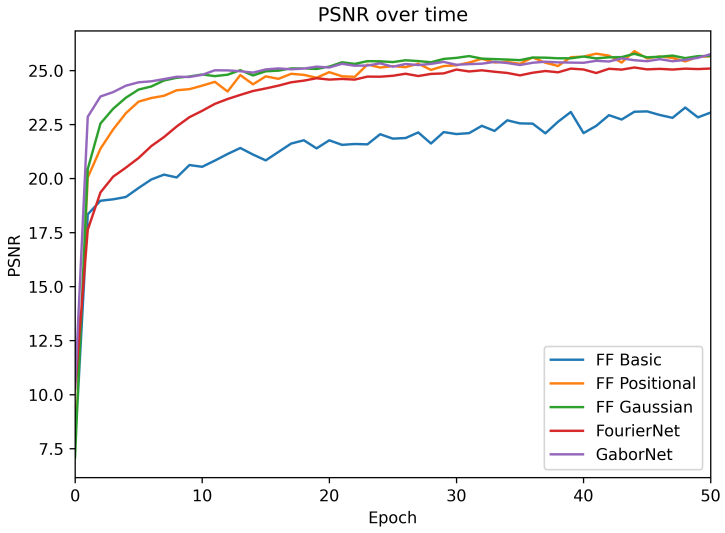

Figure 11: Plot of the PSNR over time for the Fourier feature networks, GaborNet, and FourierNet

images, and 13 test images. All the results in Table 4, rendered images in Figure 5b, as well as additional rendered images in Figure 10 are computed from the test set. In all models, we use a 4 hidden layer network with 256 nodes in each hidden layer. For both FOURIERNET and GABORNET, we set the input scaling factor to 180, with the value of α in GABORNET set to 1.0. We use Adam optimizer with learning rate of 1e-3 and no weight decay. For the Fourier feature models, we use the hyperparameters presented in the code repository published by the authors. The optimizer used in the Fourier feature models is Adam optimizer with learning rate of 5e-4 and no weight decay. All the Fourier feature models use 256 as the embedding size for the random features. All models use 2048 batches of rays, except GABORNET that uses 1440 batches of rays due to memory restriction.