

Qonvolution: Towards Learning High-Frequency Signals with Queried Convolution

Supplementary Material

Contents

A Qualitative Results	16
A.1 Image SR	16
A.2 NVS	16
B Theoretical Results	19
B.1 Benefit of features	19
B.2 Shortcomings of Gaussian-based Representation	20
C Implementation Details	22
C.1 1D Regression	22
C.2 2D Image SR	22
C.3 2D Regression	22
C.4 NVS	22
D Experiments	24
D.1 NVS	24

A QUALITATIVE RESULTS

A.1 IMAGE SR

We provide some qualitative results of image SR experiment with Real-ESRGAN (Wang et al., 2021) as the baseline in Fig. 5. Adding QNN to Real-ESRGAN faithfully reconstructs high-frequency details in various regions and results in higher quality synthesis visually.

A.2 NVS

We now provide qualitative results of NVS task using 3DGS (Kerbl et al., 2023) baseline on multiple datasets in Fig. 6. Adding QNN to 3DGS faithfully reconstructs high-frequency details in various regions and results in higher quality synthesis visually.

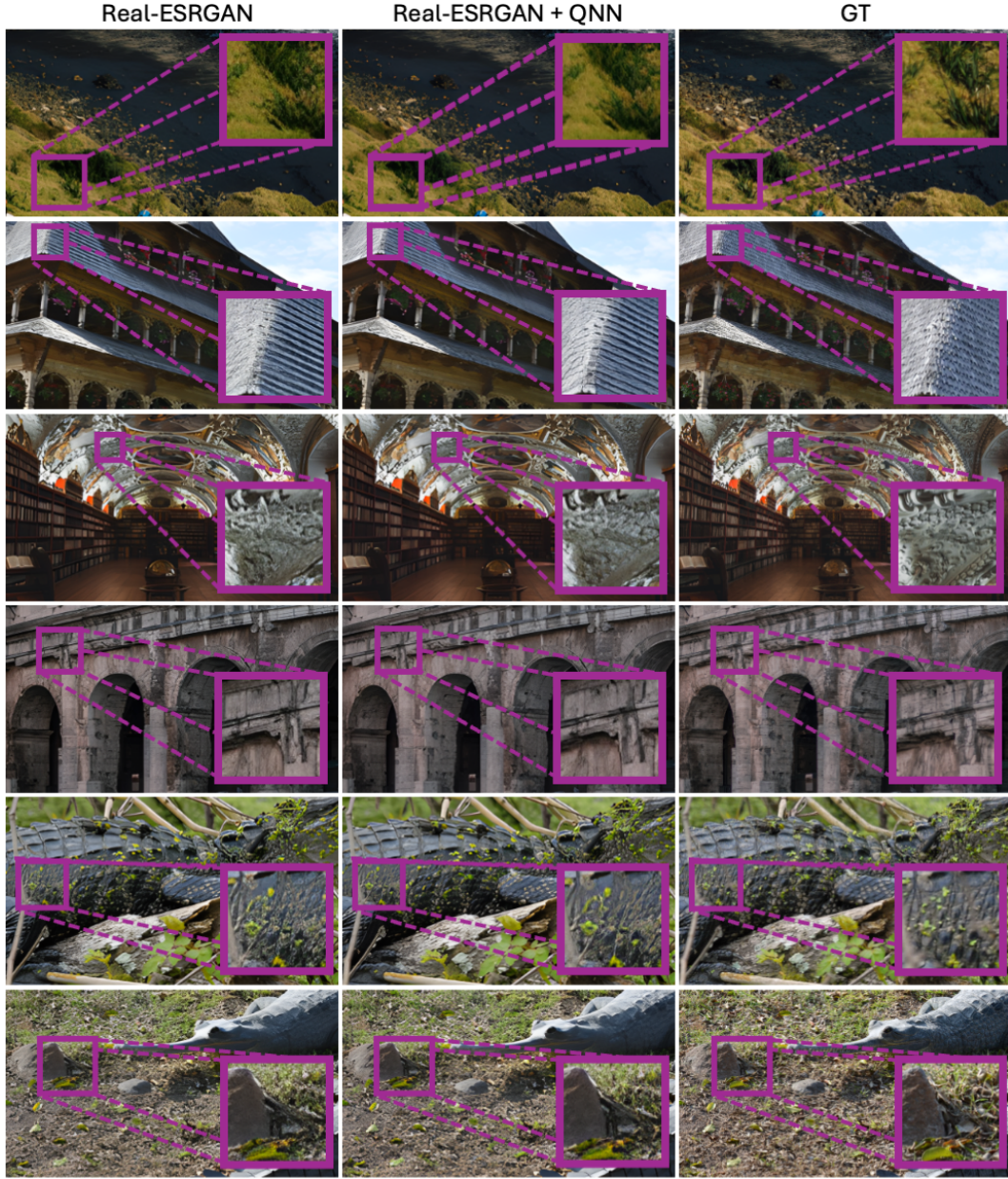


Figure 5: **SR Results** of DIV2K Val images. Adding QNN to Real-ESRGAN faithfully reconstructs high-frequency details in various regions and results in higher quality synthesis visually. We highlight the differences in inset figures.

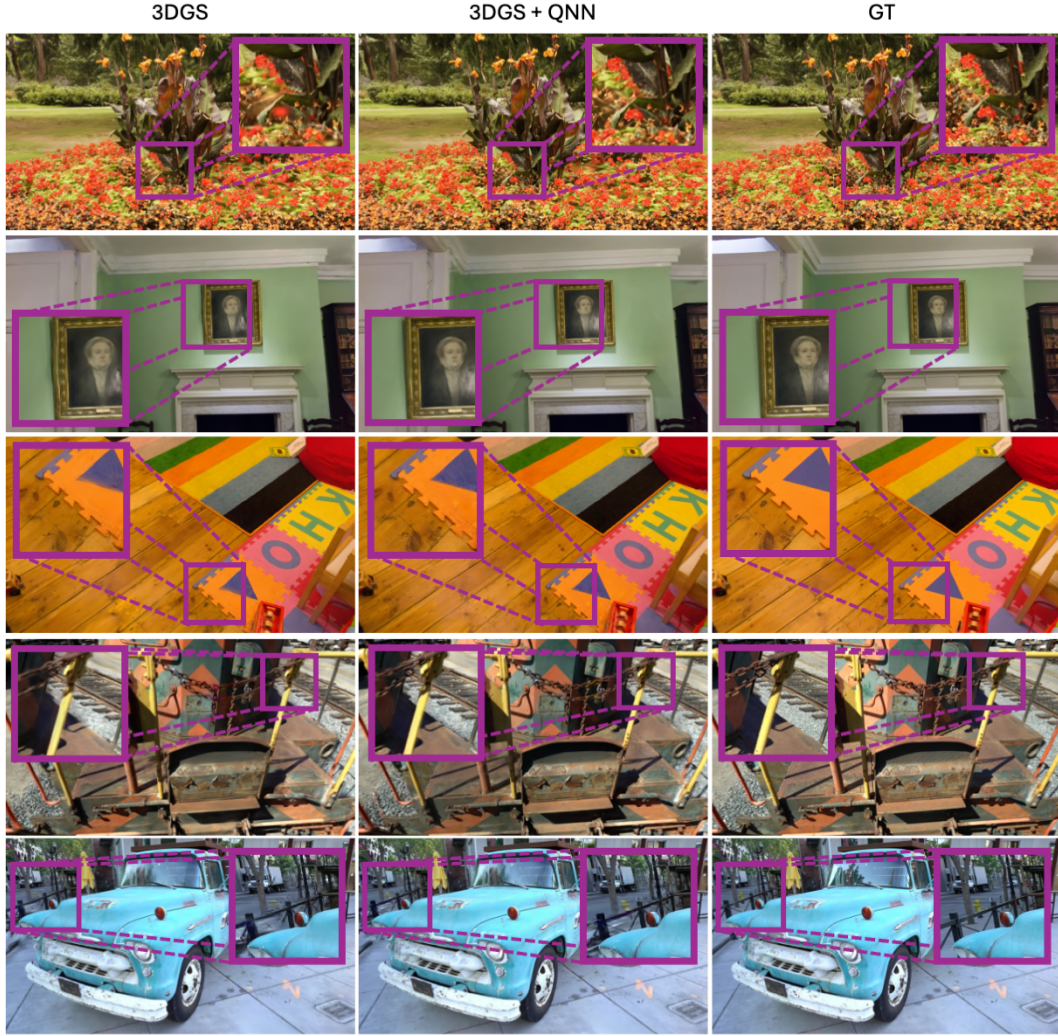


Figure 6: **NVS Results.** We provide examples of NVS task using 3DGS (Kerbl et al., 2023) baseline on multiple datasets. Adding QNN to faithfully reconstructs high-frequency details in various regions and results in higher quality synthesis visually. We highlight the differences in inset figures.

B THEORETICAL RESULTS

We now add some explanations and proofs which we could not put in the main paper because of the space constraints.

B.1 BENEFIT OF FEATURES

We show that adding neighborhood and queries provide the best possible risk.

Setup. Let \mathcal{X} The input space \mathcal{X} is either a finite discrete set or a subset of \mathbb{R}^d and $P_{\mathcal{X}}$ is a probability distribution on \mathcal{X} . The target function $f : \mathcal{X} \rightarrow [0, 1]$ is any measurable deterministic function. A feature map $\phi : \mathcal{X} \rightarrow \mathcal{Z}$ is a deterministic measurable function, mapping \mathcal{X} to a feature space \mathcal{Z} , resulting in a random variable $Z = \phi(X)$ distributed according to P_Z . The hypothesis class $\mathcal{H}(\phi)$ is the set of all deterministic measurable functions that map from \mathcal{Z} to the output space $[0, 1]$. The risk of a hypothesis $h \in \mathcal{H}(\phi)$ is the expected squared (MSE) loss: $\mathcal{R}(h, \phi, f) = \mathbb{E}_{P_{\mathcal{X}}}[(f(X) - h(\phi(X)))^2]$. And the best achievable risk for a feature map ϕ is $\mathcal{R}^*(\phi, f) = \min_{h \in \mathcal{H}(\phi)} \mathcal{R}(h, \phi, f)$.

Lemma 1 (Best achievable risk). *The lowest achievable risk for a feature map ϕ is $\mathcal{R}^*(\phi, f) = \mathbb{E}_{P_Z}[\mathbb{V}_{P_{X|Z}}(f(X)|Z)]$.*

Proof. Given that we are using the squared loss, and since f is a deterministic function, the optimal predictor is given by (Bishop & Nasrabadi, 2006, Eq. (3.36)):

$$h^*(Z) = \mathbb{E}_{P_{X|Z}}[f(X)|Z].$$

Since f and ϕ are deterministic and measurable, the conditional expectation $\mathbb{E}_{P_{X|Z}}[f(X)|Z]$ exists, is a deterministic function of Z and is measurable in Z . Furthermore, since $f(X) \in [0, 1]$ the conditional expectation also lies in $[0, 1]$. Therefore, h^* there is a deterministic measurable function from \mathcal{Z} to $[0, 1]$, and thus belongs to $\mathcal{H}(\phi)$.

The minimal risk is therefore given by:

$$\mathcal{R}^*(\phi, f) = \mathcal{R}(h^*, \phi, f) = \mathbb{E}_{P_Z}[\mathbb{E}_{P_{X|Z}}[(f(X) - h^*(Z))^2|Z]] = \mathbb{E}_{P_Z}[\mathbb{V}_{P_{X|Z}}[f(X)|Z]]$$

where the last step uses the definition of h^* together with the definition of conditional variance. \square

Lemma 2 (Feature augmentation cannot worsen the best achievable risk). *Define two feature maps ϕ_i and ϕ_j where $\phi_j(X) = (\phi_i(X), \tilde{\phi}(X))$ contains the features of ϕ_i and $\tilde{\phi}$ provides some additional features. Then:*

$$\mathcal{R}^*(\phi_i, f) \geq \mathcal{R}^*(\phi_j, f).$$

Proof. Denote the random variable $F = f(X)$ and the random variables resulting from the feature maps as $Z_i = \phi_i(X)$, $\tilde{Z} = \tilde{\phi}(X)$, $Z_j = \phi_j(X) = (Z_i, \tilde{Z})$. Therefore, there exists a function $\pi(z_i, \tilde{z}_j) = z_i$ which recovers Z_i from Z_j . Thus $F \perp Z_i | Z_j$ (conditioning on Z_j already determines Z_i , so adding Z_i provides no further information). Applying the data processing inequality for Bayes risk (Xu & Raginsky, 2022) gives:

$$\mathcal{R}^*(\phi_i, f) \geq \mathcal{R}^*(\phi_j, f).$$

\square

Lemma 3 (Including coordinates X in the features yields zero optimal risk). *Let $Z = \phi(X) = (X, \tilde{\phi}(X))$ where $\tilde{\phi}$ is some feature map. Then*

$$\mathcal{R}^*(\phi, f) = 0$$

Proof. Using Lemma 1: $\mathcal{R}^*(\phi, f) = \mathbb{E}_{P_Z}[\mathbb{V}_{P_{X|Z}}(f(X)|Z)]$. Since Z contains X , conditioning on Z determines X exactly. As f is a deterministic function of X , we have $\mathbb{V}_{P_{X|Z}}(f(X)|Z) = 0$. Therefore:

$$\mathcal{R}^*(\phi, f) = \mathbb{E}_{P_Z}[\mathbb{V}_{P_{X|Z}}(f(X)|Z)] = \mathbb{E}_{P_Z}[0] = 0.$$

\square

Proof of Theorem 1

Proof. The different feature maps are $\phi_1(x) = (g(x))$; $\phi_2(x) = (\phi_1(x - \Delta), \phi_1(x), \phi_1(x + \Delta))$ for a fixed $\Delta > 0$; $\phi_3(x) = (\phi_2(x), x - \Delta, x, x + \Delta)$. Applying Lemma 2 twice, first with $\phi_i = \phi_1, \phi_j = \phi_2$ and then with $\phi_i = \phi_2, \phi_j = \phi_3$ results in the non-strict inequalities:

$$\mathcal{R}^*(\phi_1, f) \geq \mathcal{R}^*(\phi_2, f) \geq \mathcal{R}^*(\phi_3, f).$$

Applying Lemma 3 for $\phi = \phi_3$ shows that $\mathcal{R}^*(\phi_3, f) = 0$. Combining these results proves the theorem:

$$\mathcal{R}^*(\phi_1, f) \geq \mathcal{R}^*(\phi_2, f) \geq \mathcal{R}^*(\phi_3, f) = 0$$

□

B.2 SHORTCOMINGS OF GAUSSIAN-BASED REPRESENTATION

We next present a theoretical result on the shortcoming of the gaussian-based representations. We will show that increasing image-fidelity (increasing PSNR or decreasing MSE) for gaussian-based representation requires exponential increase in the number of gaussians.

Theorem 2 (Approximation Error with Wavelets, Theorem 1, (Maierov & Meir, 2002)). *Let f be a class of measurable real-valued functions defined on a compact domain in \mathbb{R}^d , and let $Y_p^{r,d}$ denote a Sobolev space function with pseudo-dimension r for some $r > 0$. Next, consider the set of n -term ψ wavelet approximations:*

$$f_n^l(\psi) = \sum_{j=1}^n c_j \psi(\mathbf{A}_j x + \mathbf{b}_j), \quad (4)$$

For $r > 1$ and $1 \leq p, q < \infty$ satisfying $\frac{r}{d} > \left(\frac{1}{p} - \frac{1}{q}\right)_+$ and integers $1 \leq n, d < \infty$, the approximation error in L_q norm is bounded from below by

$$L_q(Y_p^{r,d}, f_n^l(\psi)) \geq \frac{c}{(n \log n)^{r/d}}, \quad (5)$$

where c is a constant independent of n .

Corollary 2.1 (Number of Gaussians). *Let a 2D image be a measurable real-valued function whose first derivative exists at all points. Then, the minimum number of 2D splatted gaussians, n , required to achieve an approximation L_2 error of ϵ is bounded below by:*

$$n \geq \exp \left[W \left(\frac{c}{\epsilon} \right)^2 \right], \quad (6)$$

where W denotes the Lambert W function and c is a constant independent of ϵ .

Corollary 2.1 says that the number of gaussians n required increases exponentially as the desired error ϵ decreases.

Proof. Let ϵ_q denote the approximation error, defined by the LHS of Eq. (5) in Theorem 2. Then, we have the following inequality:

$$\epsilon_q \geq \frac{c}{(n \log n)^{r/d}}$$

To find the minimum number of gaussians n , we rearrange the inequality to get

$$n \log n \geq \left(\frac{c}{\epsilon_q} \right)^{d/r} \quad (7)$$

To solve for number of gaussians n , we use the Lambert W function, which is defined as the inverse function of $f(w) = we^w$. The Lambert W function is the multi-valued inverse. In our case, since the number of gaussians n must be a positive integer, we only consider the principal branch of the

function, $W_0(w)$ for $w \geq 0$. We rewrite the inequality in a form that allows us to apply this function. Consider the equality case:

$$n \log n = \left(\frac{c}{\epsilon_q} \right)^{d/r} \quad (8)$$

Let $u = \log(n)$. Then, $n = \exp(u)$. Substituting, we have:

$$\begin{aligned} \exp(u) u &= \left(\frac{c}{\epsilon_q} \right)^{d/r} \\ \implies u &= W \left(\frac{c}{\epsilon_q} \right)^{d/r} \end{aligned} \quad (9)$$

Substituting back $u = \log n$ and writing for original inequality, we have:

$$n \geq \exp \left[W \left(\frac{c}{\epsilon_q} \right)^{d/r} \right]. \quad (10)$$

For splatted 3D gaussians in 2D, the domain dimension $d=2$ and the wavelet ψ is the gaussian wavelet. Additionally, the terms c_j , \mathbf{A}_j and \mathbf{b}_j in Eq. (4) corresponds to the opacity, projected 3D covariance and projected 3D mean respectively. Also, the target image function has its first derivatives defined at all points, and so, it belongs to a Sobolev space with pseudo-dimension $r=1$. Substituting these values and choose $q = 3$, $p > 6/5$ into the above inequality gives the bound:

$$n \geq \exp \left[W \left(\frac{c}{\epsilon_3} \right)^2 \right] \quad (11)$$

Let ϵ denote the MSE. Using Jensen's inequality, the norms are related by $\epsilon_3 \leq \epsilon$. Hence, we write:

$$n \geq \exp \left[W \left(\frac{c}{\epsilon} \right)^2 \right] \quad (12)$$

Thus, the number of gaussians n required increases exponentially for decreasing the MSE ϵ .

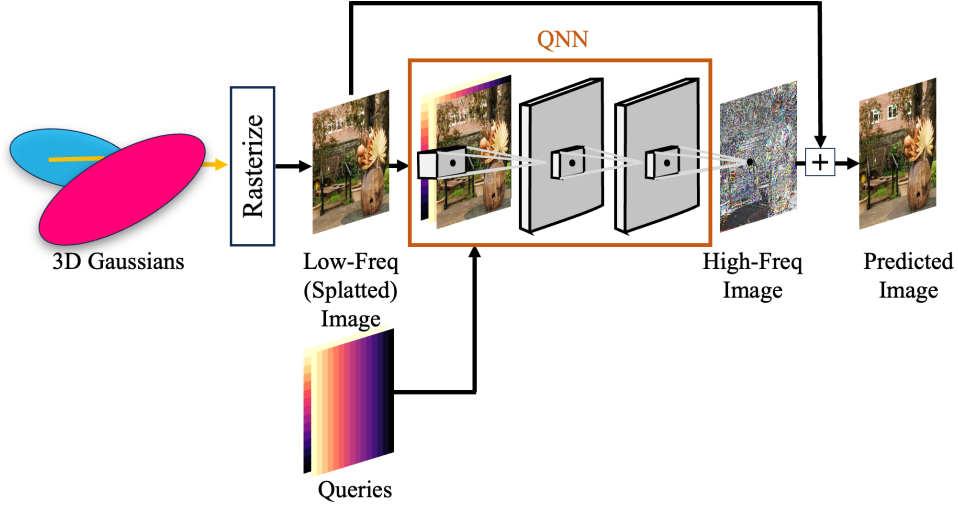


Figure 7: **GS+QNN Architecture.** QNN takes the 2D low-frequency splatted image and queries as input to produce the high-frequency image, which is then added to the splatted image to output the predicted novel view image.

C IMPLEMENTATION DETAILS

We now provide some additional implementation details.

C.1 1D REGRESSION

Dataset. The authors generate this 1D signal by sampling from a standard i.i.d. Gaussian vector of length N , scaling its j th entry by $1/j^\alpha$, and then taking the real component of its inverse Fourier transform. To fully evaluate the networks’ ability to learn high-frequency signals, we remove the signal’s bandlimitedness.

Implementation. To ensure a fair comparison, we reduce the input channels of CNN and QNN architectures by a factor of $\sqrt{3}$ to match the parameter count with vanilla MLP, as they use a 1D convolution kernel size of 3. Since all these networks predict the high-frequency signal, we add the low-frequency component to these signal to obtain the predicted signal, which we then use for metric calculation.

C.2 2D IMAGE SR

Implementation. For the SR experiments, we directly predict the desired signal without adding the low-frequency signal since the backbones contain residual connections.

C.3 2D REGRESSION

Implementation. To ensure a fair comparison, we reduce the input channels of QNN architectures by a factor of 3 to match the parameter count with vanilla MLP, as QNN uses a 2D convolution kernel size of 3×3 . The Fourier encodings use the default setting of 256-dimensional cosines and sines each randomly sampled from the standard deviation of 10. Since all these networks predict the high-frequency signal, we add the low-frequency component to these signal to obtain the predicted signal, which we then use for metric calculation.

C.4 NVS

Data Preprocessing. For data preprocessing, we follow established practices. We downsample indoor Mip-NeRF 360 scenes by a factor of two and outdoor scenes by four (Kerbl et al., 2023). We

downsample the OMMO scene #01 images by a factor of four as in [Kheradmand et al. \(2024\)](#), while other scenes and datasets use their original image resolutions.

Initialization. For initialization, we use the provided Structure-from-Motion (SfM) data for Mip-NeRF 360, Tank & Temples, Deep Blending and OMMO datasets. We use random initialization for Shelly and Syn-NeRF as they do not provide SfM points.

Architecture. We show the architecture of combining Gaussian-Splatting with QNN in Fig. 7. QNN takes the 2D low-frequency splatted image and queries as input to produce the high-frequency image, which is then added to the low-frequency splatted image to output the predicted novel view image. Unlike NeRF-based methods, every pixel in QNN needs exactly one query and does not need any volumetric rendering which does not hamper the training times.

Implementation. Our implementation uses the *gsplat* library⁵ ([Ye et al., 2024](#)) and PyTorch ([Paszke et al., 2019](#)). While MLPs take coordinate queries to output high-frequency three-dimensional RGB signal, QNNs convolves both 2D coordinate queries and the low-frequency splatted image to output the RGB signal. This signal is then added to the low-frequency 2D splatted image to produce the predicted NVS output.

All MLP architectures use 4 linear layers with 256 dimensions and ReLUs. To ensure a fair comparison, we reduce the input channels of QNN by a factor of 4. Thus, we implement QNN with 4 2D convolutional layers, a kernel size of 3×3 , 64 channels, ReLUs and vanilla encodings. We experiment with other encodings in Sec. 5.5: Encodings section.

Optimization. For QNN models, the QNN parameters are trained with AdamW optimizer with a learning rate of $1e-4$. We train with all architectures in an end-to-end learning framework without changing the total iterations or loss functions.

⁵<https://github.com/nerfstudio-project/gsplat>

Table 6: **Comparison of adding sky model** and adding QNN each to the 3DGS baseline (Kerbl et al., 2023) on the Mip-NeRF 360 dataset. Adding **QNN outperforms** sky modeling. [Key: **Best**]

Method	Val			Train		
	PSNR (↑)	SSIM (↑)	LPIPS (↓)	PSNR (↑)	SSIM (↑)	LPIPS (↓)
3DGS (Kerbl et al., 2023)	27.67	0.82	0.20	29.71	0.90	0.16
3DGS + Vanilla Sky (Rematas et al., 2022)	27.63	0.82	0.20	29.67	0.89	0.16
3DGS + Fourier Sky	27.69	0.82	0.20	29.67	0.90	0.16
3DGS + QNN	27.96	0.83	0.20	30.11	0.90	0.16

Table 7: **Impact of initialization** of baseline and QNN models on the Mip-NeRF 360 dataset. Adding **QNN outperforms** the baselines on both val and train images across initializations.

Method	Initialization	QNN	Val			Train		
			PSNR (↑)	SSIM (↑)	LPIPS (↓)	PSNR (↑)	SSIM (↑)	LPIPS (↓)
3DGS	SfM	×	27.67	0.82	0.20	29.71	0.90	0.16
		✓	27.96 (+0.29)	0.83 (+0.01)	0.20 (+0.00)	30.11 (+0.41)	0.90 (+0.00)	0.16 (+0.00)
	Random	×	26.28	0.77	0.26	28.19	0.85	0.21
		✓	26.82 (+0.54)	0.79 (+0.02)	0.23 (+0.03)	29.85 (+1.66)	0.89 (+0.04)	0.17 (+0.04)
MCMC	SfM	×	28.26	0.84	0.17	30.31	0.91	0.14
		✓	28.58 (+0.32)	0.84 (+0.00)	0.16 (+0.01)	30.69 (+0.38)	0.91 (+0.00)	0.14 (+0.00)
	Random	×	27.28	0.81	0.20	30.07	0.90	0.15
		✓	27.51 (+0.23)	0.81 (+0.00)	0.20 (+0.00)	30.38 (+0.31)	0.91 (+0.01)	0.15 (+0.00)

Table 8: **Adding extra parameters to 3D gaussian** primitives to the 3DGS baseline (Kerbl et al., 2023) on the Mip-NeRF 360 dataset. Increasing parameters of gaussians in 3D space **overfits** on the training data, and does not generalize. **QNN maintains** a good balance of generalization and fitting performance. [Key: **Best**]

Method	Val			Train		
	PSNR (↑)	SSIM (↑)	LPIPS (↓)	PSNR (↑)	SSIM (↑)	LPIPS (↓)
3DGS (Kerbl et al., 2023)	27.67	0.82	0.20	29.71	0.90	0.16
3DGS + 1-layer CNN	27.47	0.82	0.21	30.12	0.91	0.15
3DGS + 2-layer CNN	26.83	0.81	0.23	30.56	0.92	0.14
3DGS + 4-layer CNN	27.42	0.81	0.23	31.23	0.92	0.14
3DGS + QNN	27.96	0.83	0.20	30.11	0.90	0.16

D EXPERIMENTS

We next provide additional details and results of the experiments evaluating QNN’s performance.

D.1 NVS

Sky Model. Another related baseline is the sky modelling (Rematas et al., 2022; Sun et al., 2024) where the MLP network maps view directions to a residual color with a MLP and sigmoid at the end. We compare QNN with vanilla sky and Fourier encoded sky in Tab. 6. Tab. 6 shows that QNN outperforms sky models as well.

Impact of Initialization. 3DGS methods are sensitive to initialization (Kerbl et al., 2023). We therefore analyze the impact of SfM and random initialization on all nine scenes of the Mip-NeRF 360 dataset. We experiment with both 3DGS (Kerbl et al., 2023) and MCMC (Kheradmand et al., 2024) baselines in Tab. 7. Tab. 7 results show that QNN remains effective for both these initializations.

Adding Parameters to Gaussians. QNN design adds network in splatted (2D) space. An alternative is to add extra 32 parameters to the 3D gaussians in 3D space similar to Textured-GS (Chao et al., 2025), splatting these parameters, passing through a CNN with varying convolutional layers and then adding as a residual in Tab. 8. Note that this does not change the number of gaussians but only changes the number of parameters per gaussian. The results in Tab. 8 show that adding parameters to 3D gaussian overfits to the training data and does not generalize.

Increasing Number of Gaussians. Another potential alternative to improve high-frequency details is by increasing the number of 3D gaussians. Since the original 3DGS baseline does not restrict the number of 3D gaussians, we go with the MCMC baseline (Kheradmand et al., 2024) which lets us control the final number of gaussians. We find that this approach suffers from optimization difficulties since the training performance improves but the val performance saturates. This is expected

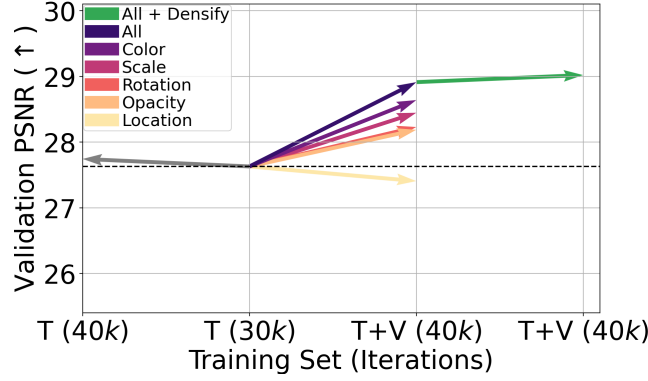


Figure 8: **Oracle Analysis** on the Mip-NeRF 360 dataset finetuning one or all of the parameters. **Color (SH) coefficients are the most important parameter** for improving visual fidelity. The figure shows that extra training time [T(30k)→T(40k)] does not significantly change the validation PSNR, while densification also does not help if we have finetuned everything on the validation images.

Table 9: **Increasing number of gaussians** of the MCMC baseline (Kheradmand et al., 2024) on the Mip-NeRF 360 dataset. **QNN outperforms** increasing number of gaussians. [Key: Best]

Method	#Gaussians (M)	Val			Train		
		PSNR (↑)	SSIM (↑)	LPIPS (↓)	PSNR (↑)	SSIM (↑)	LPIPS (↓)
MCMC (Kheradmand et al., 2024)	1.00	28.26	0.84	0.17	30.31	0.91	0.14
MCMC	1.25	28.23	0.84	0.17	30.51	0.91	0.13
MCMC	1.50	28.25	0.84	0.17	30.66	0.92	0.13
MCMC	1.75	28.23	0.84	0.17	30.78	0.92	0.13
MCMC + QNN	1.00	28.58	0.84	0.16	30.69	0.91	0.14

Table 10: **Number of layers** in QNN on the NVS task with 3DGS (Kerbl et al., 2023) on the Mip-NeRF 360 dataset. QNN uses 4 layers. [Key: Best]

Layers	Val			Train		
	PSNR (↑)	SSIM (↑)	LPIPS (↓)	PSNR (↑)	SSIM (↑)	LPIPS (↓)
2	27.81	0.82	0.20	29.85	0.90	0.16
3	27.89	0.83	0.20	30.01	0.90	0.16
4	27.96	0.83	0.20	30.11	0.90	0.16
5	27.91	0.83	0.20	30.13	0.90	0.16
6	27.90	0.83	0.20	30.12	0.90	0.16

since Corollary 2.1 analytically shows that decreasing MSE (increasing PSNR) needs increasing the number of gaussians exponentially. This makes increasing number of gaussians unscalable.

Number of Convolution Layers. Finally, we experiment with the number of convolution layers to pinpoint the optimal QNN depth in Tab. 10. We found that reducing the number of layers below four negatively impacted performance, while increasing the count beyond four did not provide any additional benefit.

Oracle Analysis of Gaussian Splatting. To determine which parameters are most critical to the performance of 3DGS, we conduct an “oracle analysis” using all nine scenes from the Mip-NeRF 360 dataset. The 3DGS method represents a scene with a collection of 3D gaussian primitives, each defined by parameters for its location, rotation, scale, opacity, and color (represented by spherical harmonics or SH). The standard training procedure involves training the gaussian representation on a set of training images (T) for 30k iterations. In our oracle study, we extend this by taking the resulting model and finetuning it for an additional 10k iterations on the combined training and validation (T+V) images. To isolate the impact of each component, we finetune either a single parameter type (e.g., only color) or all of them simultaneously. This analysis aims to reveal which parameters contribute most significantly to the PSNR performance. The results, presented in Fig. 8, show that the color coefficients (SH) are the most important parameter for improving the model’s visual fidelity. This is why, QNN predicts the residual colors (Fig. 7) in gaussian splatting experiments of Sec. 5.4, while predicting opacity in VDGS (Malarz et al., 2025) does not improve the PSNR significantly on the Mip-NeRF 360 dataset.

Table 11: **Variance Analysis** for the NVS task on the Mip-NeRF 360 dataset. **QNN outperforming** the baselines is statistically significant.

Method	Seed	Val			Train		
		PSNR (\uparrow)	SSIM (\uparrow)	LPIPS (\downarrow)	PSNR (\uparrow)	SSIM (\uparrow)	LPIPS (\downarrow)
3DGS (Kerbl et al., 2023)	222	27.67	0.82	0.20	29.71	0.90	0.16
	111	27.65	0.82	0.20	29.69	0.90	0.16
	555	27.72	0.82	0.20	29.69	0.90	0.16
	Avg $\pm \sigma$	27.68 \pm 0.03	0.82 \pm 0.00	0.20 \pm 0.00	29.69 \pm 0.02	0.90 \pm 0.00	0.16 \pm 0.00
3DGS + QNN	222	27.96	0.83	0.20	30.11	0.90	0.16
	111	27.92	0.83	0.20	30.11	0.90	0.16
	555	27.93	0.83	0.20	30.12	0.90	0.16
	Avg $\pm \sigma$	27.92 \pm 0.03	0.83 \pm 0.00	0.20 \pm 0.00	30.12 \pm 0.02	0.90 \pm 0.00	0.16 \pm 0.00
MCMC (Kheradmand et al., 2024)	222	28.26	0.84	0.17	30.31	0.91	0.14
	111	28.22	0.84	0.17	30.32	0.91	0.13
	555	28.31	0.84	0.17	30.31	0.91	0.13
	Avg $\pm \sigma$	28.26 \pm 0.03	0.84 \pm 0.00	0.17 \pm 0.00	30.31 \pm 0.02	0.91 \pm 0.00	0.13 \pm 0.00
MCMC + QNN	222	28.58	0.84	0.17	30.69	0.91	0.14
	111	28.53	0.84	0.17	30.67	0.91	0.14
	555	28.54	0.84	0.17	30.69	0.91	0.14
	Avg $\pm \sigma$	28.55 \pm 0.02	0.84 \pm 0.00	0.17 \pm 0.00	30.67 \pm 0.02	0.91 \pm 0.00	0.13 \pm 0.00

Table 12: **Per-scene PSNR Analysis** for the NVS task on the Mip-NeRF 360 dataset. **Larger PSNR improvements are observed in indoor scenes** [Key: **Best**]

Method	Avg	Outdoor	Indoor	garden	bicycle	stump	bonsai	counter	kitchen	room	treehill	flowers
3DGS (Kerbl et al., 2023)	27.68	24.97	31.06	27.66	25.61	26.89	32.16	29.10	31.40	31.59	22.82	21.88
3DGS + QNN	27.92	25.08	31.50	27.89	25.66	27.17	32.57	29.23	31.70	32.51	22.73	21.97
MCMC (Kheradmand et al., 2024)	28.26	25.38	31.96	28.00	25.92	27.22	33.25	29.86	32.46	32.26	23.33	22.24
MCMC + QNN	28.55	25.43	32.44	28.24	25.97	27.53	33.72	29.96	32.74	33.34	23.25	22.17

Variance Analysis. We next check the statistical significance of our results by running the NVS results with three different seeds in Tab. 11. The table shows that integrating QNN into the 3DGS and MCMC baselines yields statistically significant improvements.

Per-scene Analysis. We next investigate the question whether PSNR gains are statistically consistent across scenes, or dominated by a few scenes in Tab. 12. Tab. 12 shows the mean PSNR across three different seeds with both 3DGS and MCMC baselines for NVS on the Mip-NeRF 360 dataset. The PSNR gains are not uniformly distributed across all scenes and larger improvements are observed specifically in indoor scenes.