Q: *Why does Table 2 not have "tuned PWC-Net w/ [1] warping"?*
Reply: One cannot simply take [1] and plug it into a deep learning setting since it, among other concerns, is unclear how one would implement its outside-in filling as an efficient and differentiable operation. Indeed, one of our contributions is to show how to perform [1] better and in a differentiable manner, which enables end-to-end fine-tuning of the underlying flow estimator.

Q: *Why does the approach yield state-of-the-art results at high resolutions like like on Xiph-4K but not lower ones like on Vimeo-90k?*
Reply: A synthesis approach like ours that solely relies on splatting will never be able to surpass an equivalent version that also utilizes a subsequent refinement network. As such, while our computational efficiency is unmatched, we consider the quantitative performance of our proposed interpolation pipeline as "good" but not "state-of-the-art" at low resolutions. The only reason we are able to claim state-of-the-art results at high resolutions is due to our proposed iterative upsampling, but other methods could equally make use of this technique to likewise boost the quality of the interpolation results at high resolutions.

Q: *Why upsampling the flow instead of the result?*
Reply: Downsampling the input frames to $1/4$ resolution and then performing the video frame interpolation at that resolution before upsampling the results back to the original resolution only yields a PSNR of 29.54 on the Xiph-2K dataset, which is as staggering 5.27 dB lower than what we propose.

Q: *Why is there no comparison to other upsampling techniques?*
Reply: We do not claim state-of-the-art optical flow upsampling, we only demonstrate that our simple iterative scheme in the context of our frame interpolation framework is beneficial in terms of quality and speed at high resolutions. Furthermore, using spatially-varying upsampling kernels [6], normalized convolution upsampling [2], or self-guided upsampling [3] would be computationally more demanding which counteracts the underlying efficiency goal.

Q: *Why only include SepConv++[5] and SoftSplat [4] in Figure 1?*
Reply: We include SepConv++[5] as a representative of fixed-time interpolation techniques and SoftSplat [4] as a representative of arbitrary-time interpolation techniques. Even if we included all other interpolation methods that we compare to, this figure would still tell the same story that the higher the interpolation factor the better our proposed approach performs in comparison.

Q: *What values do α_p, α_f, α_v for M^{splat} and M^{merge} converge to?*
Reply: They converge to $\alpha_{\mathrm{p}}^{\mathrm{s}} = 2.35$, $\alpha_{\mathrm{f}}^{\mathrm{s}} = 5.89$, and $\alpha_{\mathrm{v}}^{\mathrm{s}} = 1.24$ for M^{splat} as well as $\alpha_{\mathrm{p}}^{\mathrm{m}} = 164.96$, $\alpha_{\mathrm{f}}^{\mathrm{m}} = 108.04$, and $\alpha_{\mathrm{v}}^{\mathrm{m}} = 1.18$ for M^{merge}. As such, the photometric consistency ψ_{photo} is less important than the flow consistency ψ_{flow} for the splatting metric and vice versa for the merging metric.

Q: *Why not use an outside-in strategy like in [1] to fill holes after splatting?*
Reply: We did not experiment with outside-in filling to close holes after splatting since it is unclear how to make this operation sufficiently fast. Specifically, one has to call the outside-in filling potentially many times since each invocation only fills a single slice of pixels at the boundary of the holes.

Q: *Which standard deviation was used for the 4×4 Gaussian splatting kernel?*
Reply: We use a $\sigma = 0.6$ which we determined through a hyperparameter sweep, which roughly equals the commonly used $(\mathrm{ksize} - 1)/6$ formula.

Q: *Why is softmax splatting with $Z = 0$ essentially average splatting?*
Reply: One needs to avoid divisions by zero when normalizing the splatted values by dividing by the splatted importance metric. As such, one needs to add an ϵ in the denominator. However, if Z consists of sufficiently small values, the ϵ in the denominator would reduce the overall brightness of the splatted result. As such, a correctly implemented softmax splatting operator should add an ϵ to the importance metric Z before using it to avoid divisions by zero while also making sure that the overall brightness is maintained.

Q: *How does the proposed method differ from SoftSplat [4]?*
Reply: First, [4] employs a synthesis network to obtain the interpolation result from splatted feature pyramids. However, splatting these pyramids and invoking the synthesis network requires 800 ms on a V100 to yield a 4K output. In contrast, we warp colors and combine the warped inputs using a merge metric (Equation 1) which only takes 36 ms (on the same hardware and resolution). This is the key benefit of our proposed splatting-only synthesis.

Second, [4] extracts a feature pyramid from each input frame and then splats them to the desired interpolation instant. However, this is challenging at high resolutions since at 4K, each of the four feature pyramids requires 1.7 GB of memory and the synthesis network implemented in PyTorch requires another 23.0 GB. In contrast, we splat inverse flow vectors and then use these to gather colors from the input frames (Figure 5) which at 4K requires 0.5 GB.

Third, [4] computes photo-consistency and refines it through a neural network to obtain the importance metric Z that is used to disambiguate pixels that splat to the same location. However, invoking this network at 4K takes 226 ms. Our splatting metric M^{splat}, which combines photo- with flow-consistency and flow variance, is learnable (Equation 5) and only takes 17 ms at 4K.

Fourth, [4] estimates bidirectional optical flow at the input resolution. However, this is slow (467 ms at 4K) and also error-prone as existing flow methods are typically not trained at such resolutions. Our new iterative flow upsampling lets us estimate the flow at, for example, 1K before iteratively upsampling it to 4K. This not only makes the optical flow estimation more likely to succeed and leads to better results (Table 3), but it is also much faster (157 ms).

Fifth, [4] proposes softmax splatting but their implementation is not numerically stable. In contrast, we show how to stabilize this operation which leads to subtle but consistent improvements in the interpolation quality (Table 5). As such, related work that makes use of softmax splatting [16, 17, 23, 32, 69] could likewise get better results if they were to adopt our stable softmax splatting.

Sixth, [4] utilizes a bilinear 2×2 splatting kernel, which leads to small holes in the result where the flow diverges locally (Figure 8). Our 4×4 Gaussian splatting kernel is less prone to such errors due to its larger footprint.

Consequently, we are not only 33× faster than [4] on XTEST-4k (Table 6) due to (1, 2, 3) but also 5.86 dB better due to (4). Furthermore, by only relying on splatting we are able to perform real-time interpolation from given optical flow estimates as demonstrated in the supplementary "visualization.html" which performs our splatting-based synthesis implemented in Javascript without GPU acceleration on the fly. This would not have been possible with [4].

References

[1] Simon Baker, Daniel Scharstein, J. P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A Database and Evaluation Methodology for Optical Flow. *International Journal of Computer Vision*, 92(1):1–31, 2011.

[2] Abdelrahman Eldesokey and Michael Felsberg. Normalized Convolution Upsampling for Refined Optical Flow Estimation. *arXiv/2102.06979*, 2021.

[3] Kunming Luo, Chuan Wang, Shuaicheng Liu, Haoqiang Fan, Jue Wang, and Jian Sun. UPFlow: Upsampling Pyramid for Unsupervised Optical Flow Learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[4] Simon Niklaus and Feng Liu. Softmax Splatting for Video Frame Interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[5] Simon Niklaus, Long Mai, and Oliver Wang. Revisiting Adaptive Convolutions for Video Frame Interpolation. In *IEEE Winter Conference on Applications of Computer Vision*, 2021.

[6] Zachary Teed and Jia Deng. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. In *European Conference on Computer Vision*, 2020.