Explaining the Implicit Neural Canvas: Connecting Pixels to Neurons by Tracing their Contributions

Namitha Padmanabhan* Matthew Gwilliam* Pulkit Kumar Shishira R Maiya
Max Ehrlich Abhinav Shrivastava

University of Maryland, College Park

https://namithap10.github.io/xinc

Abstract

The many variations of Implicit Neural Representations (INRs), where a neural network is trained as a continuous representation of a signal, have tremendous practical utility for downstream tasks including novel view synthesis, video compression, and image super-resolution. Unfortunately, the inner workings of these networks are seriously understudied. Our work, eXplaining the Implicit Neural Canvas (XINC), is a unified framework for explaining properties of INRs by examining the strength of each neuron's contribution to each output pixel. We call the aggregate of these contribution maps the Implicit Neural Canvas and we use this concept to demonstrate that the INRs we study learn to "see" the frames they represent in surprising ways. For example, INRs tend to have highly distributed representations. While lacking high-level object semantics, they have a significant bias for color and edges, and are almost entirely space-agnostic. We arrive at our conclusions by examining how objects are represented across time in video INRs, using clustering to visualize similar neurons across layers and architectures, and show that this is dominated by motion. These insights demonstrate the general usefulness of our analysis framework.

1. Introduction

Leveraging neural networks to represent data, where the network computes feature maps and embeddings for various visual inputs, is central to computer vision. Recently, implicit neural representations (INRs) have emerged as an exciting, radically different approach where a multilayer perceptron (MLP) or convolutional neural network (CNN) is overfit on an image as a generative network, becoming a continuous approximation of the discrete visual data [39, 40, 44, 48, 49]. The most well-known type of INRs are Neural Radiance Fields, where a network learns to represent a scene, allowing for continuous scene representation for novel view synthesis, editing, *etc.* [17, 35, 42, 54, 56]. However, INRs are useful for many other tasks as well, in particular, visual data compression [5, 7–9, 13, 14, 19, 30, 32, 50, 57, 58].

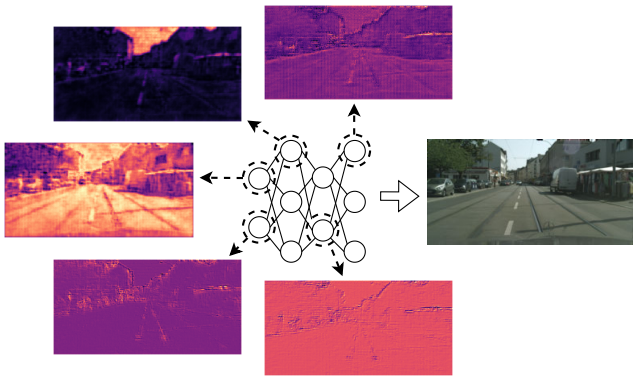

Figure 1. How do INRs "see" the images they represent? We propose XINC, which we use to show what parts of a learned visual signal are important to each neuron of an INR. Here, we take a sample from the neural canvas, with contribution maps for 5 neurons sampled from the last layer of a NeRV trained on a Cityscapes [11] video. Some neurons attend to colors and textures, while others focus on low-level features like edges.

In spite of their immense utility, the inner workings of INRs remain relatively under-explored. While some studies shed light on MLP-based INRs [55], NeRF [56], and the impact of hypernetworks [10, 55], the hidden details of non-coordinate methods such as NeRV [7] are quite opaque. This is not completely unique to INRs – explainability and interpretability are still massive challenges for other areas in vision as well [1, 2, 18, 38]. Nevertheless, there is a significant gap for understanding INRs, considering that popular analysis techniques such as NetDissect [3] and Grad-CAM [46] do not transfer to INRs in straightforward ways.

*Equal contribution

Even analysis work for generative networks tends to focus on control and manipulation [23], and the coordinate-to-signal paradigm of INR is distinct enough from the generative noise-to-signal paradigm that most analysis work cannot be trivially extended [37].

While understanding INRs would obviously be beneficial for improving their representation, analyzing with the intent to explain is valuable on its own. As INRs trend towards commercial viability for tasks like compression, explaining their behaviors becomes critical for real-world enterprises to consider them as alternatives to standard codecs. Traditional compression methods are not perfect, but their failure modes are known and easily explainable. Deep INR methods, by contrast, have failure modes that are somewhat unknown and quite opaque. Thus, beyond helping researchers to improve the models' performance, explaining INRs will help with their actual adoption in the real world.

Our work proposes eXplaining the Implicit Neural Canvas (XINC) as a novel framework for understanding INRs. Specifically, we propose a technique for dissecting INRs to create contribution maps for each "neuron" (group of weights), which connect neurons and individual pixels. Together, these contribution maps comprise the "implicit neural canvas" for a given visual signal. XINC can thus be used to help us understand the relationships between INRs and the images and videos they represent.

We analyze MLP-based and CNN-based INRs. While some other analysis works help explain MLP-based INRs [55], our analysis of convolution-based INRs is first of its kind. To demonstrate the utility of XINC we use it to derive novel insights by characterizing INRs in 5 key ways.

Neuron Contributions Correlate with Colors and Edges We show representative examples of contribution maps for MLPs and CNNs, and show how they attend to objects in terms of low-level features like edges and colors, while lacking coordinate proximity-based correlations.

INRs Must Represent to Omit We use sums of contribution maps to show that the representations learned by INRs, while intensity-biased, are not dictated by raw pixel magnitude alone. INRs must learn representations even for missing objects, where the final output colors are near zero.

Representation is Distributed We examine how many pixels each neuron represents, and to what extent, as well as conversely how many neurons are involved in the representation of each pixel. We demonstrate that these representations are quite distributed in nature.

Motion Drives Contribution Change Over Time We show how NeRVs handle objects and motion, demonstrating that contributions to a given object remain fairly constant over time. This is true even with motion, and fluctuations in our neural canvas across time for videos prove that motion drives changes in neuron contributions for NeRV.

We Can Group Similar Neurons We cluster neurons from various INRs to show how we can group neurons with similar representation properties using contribution maps.

2. Related Work

Implicit Neural Representation is a way to represent and compactly encode a variety of high resolution signals. The key idea is to map a set of coordinates for a specific signal to a function, by employing a neural network as the continuous approximating function [7, 35, 45, 48, 53]. SIREN [48] utilizes periodic activation functions in MLPs to fit and map complicated signals, including images, 3D shapes and videos using very small networks. COIN [13], the first image-specific INR method, uses implicit neural representation for image compression tasks and COIN++ [14] extends this work to encode multiple images. In the challenging realm of videos, NeRV [7] is the first method to scale video compression using image-wise implicit representation. It uses convolution layers in addition to MLPs and outputs all RGB values of a frame given the positional embedding of frame index t as input. Other methods rely on MLPs [27, 32]. Others [8, 9, 19, 58] compute embeddings on the actual frames by introducing a small encoder, thus improving performance over index-based methods. Meta-learning approaches try to address long per-item fitting times by learning a hypernetwork for predicting network weights [10, 25]. Others propose encoding weights as vectors for a wide variety of downstream tasks [31].

Interpreting and Visualizing Deep Neural Networks is a longstanding field of study in machine learning. A large number of techniques aim to understand the internal representations of convolutional neural networks and deep networks in general. Hoiem et al. analyzes effects of various object characteristics on detection performance. Grad-CAM [46] aims to provide visual explanations for the decisions of CNNs. Using the gradients flowing into the final convolutional layer from a target concept, Grad-CAM and related subsequent works [6, 12, 15] generate a coarse localization map that highlights crucial image regions contributing to the prediction of the concept, and [24] proposes a method for automatically captioning these regions to describe image features. Network Dissection [3] considers each unit in a network as a concept detector and compares their activity with human-interpretable pattern-matching tasks such as the detection of object classes. GAN Dissection [4] extends these ideas to visualize and understand GANs at the unit, object, and scene-level. Other works characterize neuron behavior with natural language [21], by combining NetDissect-style concepts [28, 36, 52], or in terms of how they behave in conjunction with other closely-related neurons [41]. Lindsay and Bau creates a comprehensive test methodology that systematically perturbs model inputs and monitors the effect on model predictions, helping

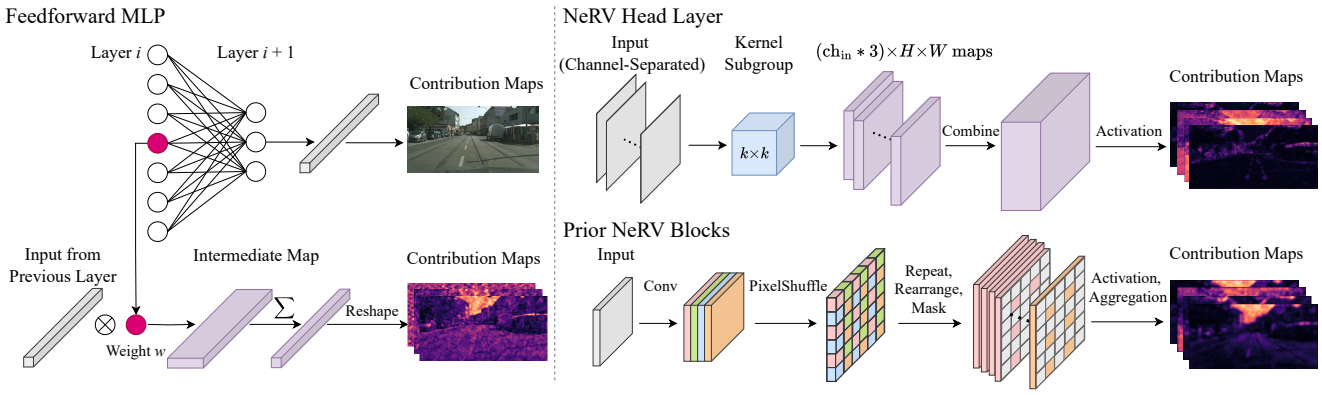

Figure 2. (left) We dissect MLP-based INRs by aggregating their activations (weights multiplied by previous layer outputs) for each pixel at each neuron. (right) We extend this core idea of pixel-to-neuron mapping for the CNN-based INR, NeRV, by computing intermediate feature maps that are not yet summed on the input dimension. For layers prior to the head, we also account for the PixelShuffle and apply an aggregation filter to account for the subsequent layer's kernels, which propagate that neuron's contribution to neighboring pixels. To compute contributions for a given layer, we simply perform the shown steps in sequence for that layer and each subsequent layer.

identify potential instabilities. These works are insightful for their target domains, but do not directly apply to INR. So, we propose to analyze the relationships between various parameters in MLP- and CNN-based INR networks and the spatio-temporal locations of the reconstructed output.

3. Implicit Neural Canvases

In this section, we explain how we compute the contribution maps that comprise the implicit neural canvas. These contribution maps connect neurons to pixels in terms of their activations (weights×inputs) that contribute to each pixel, for INRs. We first explain how we do this computation for MLP-based INRs in Section 3.1. We then explain how we extend this for CNN-based INRs in Section 3.2.

3.1. Dissecting Multi-Layer Perceptrons

In this work, as a representative method for MLP-based INRs, we use a Fourier Feature Network (FFN) [51] with a Random Fourier Matrix [43] as the positional encoding, followed by linear layers. At each MLP layer, the input pixels are operated upon independently, without any spatial rearrangement or sharing of information between the pixels. Thus, we can obtain a mapping from the neurons in a layer to the $h \times w$ spatial locations (pixels) in the output image.

For each neuron, η, we compute a mapping of how its output contributes to all neurons in the next layer (or the output channels, for the last layer). Consider a layer, l, with a weight matrix W of shape $m \times n$. We use i and j to denote the weight connecting i-th neuron in the current layer to j-th neuron in the next layer, which are also the indices of the current neuron of interest, η_i, and some neuron in the next layer, η_j, respectively. When we map a neuron, η_i, to the MLP INR's output, we are thus forming a mapping from the weights $\{w_{i1}, w_{i2}, \ldots, w_{in}\}$ to the output. Now consider the image the INR represents, I, of shape $h \times w \times 3$. Let us consider that the INR takes pixel location x_k, y_k as input, and neuron η_i produces an output scalar, o. We compute its contribution to pixel x_k, y_k as $\sum_{j=1}^{n} o \cdot w_{ij}$. When we compute this for all pixels, we obtain a map of shape $h \times w$ for η_i, which we refer to as the contribution map for η_i.

3.2. Dissecting Convolutional Neural Networks

For CNNs, we consider kernels as neurons, and we use the NeRV architecture [7], which has two types of layers. Except for the last, all layers consist of a learned convolutional layer, followed by a PixelShuffle [47] and a nonlinear activation function. The last layer consists of a convolutional layer and an optional nonlinear activation function. For this network, the construction of the contribution mapping is less straightforward compared to MLP, since convolutions operate over groups of pixels at each stage, and PixelShuffles rearrange contributions. We first explain how to address the head layer, and then how to extend this for earlier layers.

3.2.1 Head Layer

We begin by analyzing the contribution from kernels in the convolutional head layer to the output RGB image pixels. Consider the head layer, l_h, with input v_{in} of shape $h \times w \times \text{ch}_{\text{in}}$ (ch_{in} and ch_{out} are the number of input and output channels, respectively). We thus have a set of $\text{ch}_{\text{in}} * \text{ch}_{\text{out}}$ convolutional kernels, meaning ch_{in} kernels per each R, G, B channel of the output, and thus, $\text{ch}_{\text{in}} * 3$ kernels that can potentially contribute to each output pixel. We obtain the individual contribution of each kernel by passing each kernel in l_h over v_{in} and storing the outputs separately, yielding a set of feature maps, $h \times w \times (\text{ch}_{\text{in}} * \text{ch}_{\text{out}})$. To get the true

output contribution, and thus the final contribution maps for all $\text{ch}_{\text{in}} * 3$ neurons, we pass these feature maps through the l_h activation (*e.g. tanh*).

3.2.2 Prior NeRV Blocks

All NeRV layers except the head layer perform some upsampling using PixelShuffle [47]. PixelShuffle upsamples feature maps by moving elements from the channel dimension into the spatial dimension. Let the number of input channels of the convolution preceding the PixelShuffle in layer i be $\text{ch}_{i,\text{in}}$ and the number of output channels be $\text{ch}_{i,\text{out}}$. For upsampling factor r, the number of output channels from the PixelShuffle, which is the same as the number of input channels to the $(i+1)^{\text{th}}$ layer, is computed as

$$\text{ch}_{i+1,\text{in}} = \frac{\text{ch}_{i,\text{out}}}{r^2} \tag{1}$$

We require a contribution map of shape $h \times w \times (\text{ch}_{i,\text{in}} * \text{ch}_{i,\text{out}})$. To obtain this, we first follow the same approach to the head layer and obtain a mapping of shape $\frac{h}{r} \times \frac{w}{r} \times \text{ch}_{i,\text{in}} \times \text{ch}_{i,\text{out}}$. Considering the penultimate NeRV layer for the sake of simplicity, we can apply PixelShuffle to this downsampled contribution map and obtain the map of size $h \times w \times \text{ch}_{i,\text{in}} \times \text{ch}_{(i+1),\text{in}}$. However, the contributions of distinct output kernels $\text{ch}_{i,\text{out}}$ are now no longer separated since they are moved into the spatial dimension.

To resolve this, we repeat the operation r^2 times, yielding a new map of size $h \times w \times \text{ch}_{i,\text{in}} \times \text{ch}_{(i+1),\text{in}} \times r^2$, which has the same dimensions as a mapping of shape $h \times w \times \text{ch}_{i,\text{in}} \times \text{ch}_{i,\text{out}}$, if we substitute according to Equation 1. While this map has the dimensions we require for further processing by downstream layers, we must take care that each non-overlapping $r \times r$ block contains samples from the same filter in order to preserve each filter's contribution. If nearby samples were from different filters, as in the traditional PixelShuffle, then the contribution of different filters would be mixed and aggregated in future layers. We can conveniently accomplish this with a simple re-arranging of the repeated channels followed by masking such that each $r \times r$ block contains only one non-zero element (to prevent overcounting of the contribution of each kernel). The final result is then passed through the activation layer before proximity correction.

Layers before the penultimate layer differ only slightly. For the layer in question, we treat it exactly like the penultimate layer. However, the result is not at the target output resolution. So, we must perform the subsequent layers' corresponding operations, with one minor adjustment – we use nearest neighbor upsampling, instead of PixelShuffle, for all layers after the initial layer. This is because the spatial correspondences of the PixelShuffle are specific to the feature map structure of the given subsequent layer, and have no meaning for the earlier layer feature maps.

Table 1. **Dataset Statistics.** For these densely annotated datasets we provide the typical number of annotated instances per video, video length, and the portion of frames which are annotated.

Dataset	Videos	Instances	Frames	% Labeled	Domain
Cityscapes-VPS [26]	500	$2-71$	30	$1/6^{\text{th}}$	Streets
VIPSeg [34]	3536	$1-78$	$4-81$	All	Open

3.2.3 Proximity Correction

Layers before the head layer do not contribute directly to pixels. Instead, their outputs are the inputs to some later set of convolution kernels. While we account for the upsampling with how we handle PixelShuffle, we still must address the fact that the subsequent kernels act on $k \times k$ neighborhoods at each filter map. So, we use an aggregation filter to account for the fact that a given point in some intermediate feature map contributes to a $k \times k$ neighborhood of points in the next layer's feature maps. This is a simple $k \times k$ box filter for each layer.

4. Understanding Representations

4.1. Dataset and Training Details

To analyze the relationship between neurons and pixels in encoded signals, we use videos from the Cityscapes-VPS [26] (train and val) and VIPSeg [34] video panoptic segmentation datasets. Table 1 provides the number of instances per frame in each dataset, the percentage of annotated frames and other information. To select a few videos from these datasets, we look for examples containing objects of diverse shapes, sizes and categories, to afford diversity in analysis. Having instance and object category level information enables us to answer questions pertaining to whether INRs understand semantics and instances and also helps analyze the distribution of representations for objects of different types. Further, it would help us determine whether the groups of parameters attending to the same objects remains consistent across frames, whether the model omits representing small objects, and investigate other important properties. We train our FFNs and NeRVs on downsampled images, to keep the model and contribution map sizes small. Specifically, we resize and crop all videos to 128×256 resolution. We choose model sizes and training times such that bits-per-pixel and reconstruction quality are roughly equivalent across models and video frames. We provide more details in the appendix.

4.2. Contributions Correlate with Colors and Edges

Figure 3 shows some example contribution maps for neurons at different layers for both MLP-based and CNN-based INRs. FFN and NeRV head layer neurons seem to learn some form of whole scene representation, and the CNN

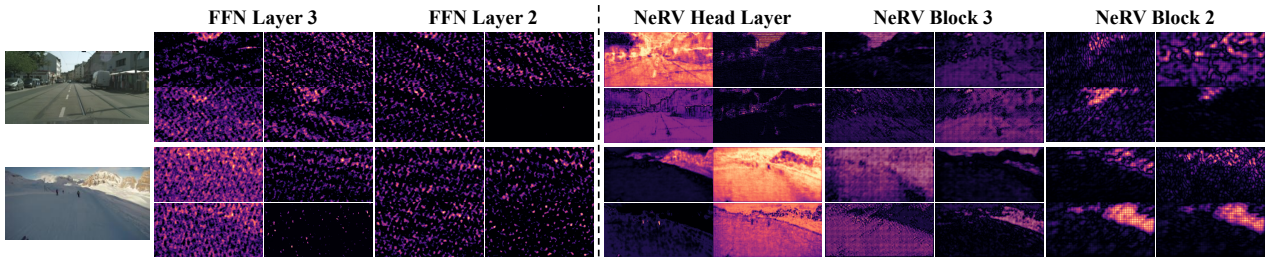

Figure 3. **The implicit neural canvas.** We show representative example contribution maps for various layers of FFN [51] and NeRV [7]. Notice how early layer FFN neurons manifest strong Fourier patterns, and how the last layers NeRV tend to resemble the image, with NeRV head layer neurons being reminiscent of classical image processing filters.

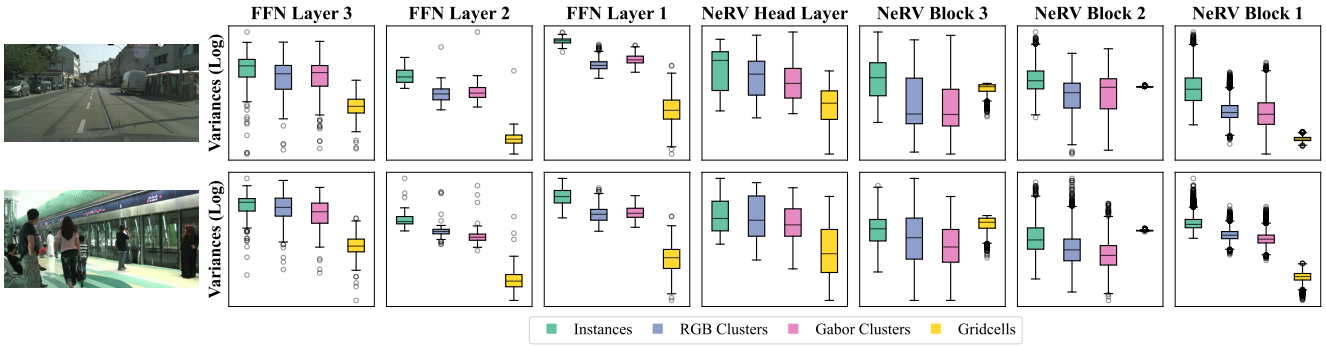

Figure 4. **Grouping contributions.** We compute the variance of the difference from expected contribution for different groupings of contributions - Instances of objects and background, RGB color-based clusters, Gabor filter-based clusters and regular gridcells. These results suggest that INRs ignore space while preferring instances, color, and edge features.

penultimate layer has similar characteristics. The contribution maps of the FFN layers suggest a progression from representing Fourier features in early layers, to representing more of the image in question, with Fourier artifacts, in the last layer. For the NeRV layers, there seem to be a variety of features captured by various neurons, including edges, textures, colors, and depth. The behavior of a neuron, in both networks, is not simply to represent some pixels, instead, it learns some low-level scene attributes.

We explore these behaviors quantitatively by aggregating neuron contributions to pixels, first in terms of instances, and then also by clustering pixels by color value (RGB Clusters), clustering on Gabor filter features [16, 33] (Gabor Clusters, details in appendix), and clustering pixels spatially (taking equal size rectangular gridcells). We hypothesize that for a type of pixel clustering to be meaningful for neurons, each neuron should tend to have high activation in some clusters, and little to no activation in others. Quantitatively, the variance of contributions across clusters would be high. However, using raw variance of contributions would give massive bias to clusterings that form large and small regions, where some contributions would be trivially small. So instead, we first compute the expected contribution for each cluster, which is the contribution over the whole image, normalized by size of the given cluster.

$$\text{cont}_{\text{expected}} = \left(\frac{\text{area}_{\text{cluster}}}{\text{area}_{\text{image}}}\right) * \text{cont}_{\text{image}} \tag{2}$$

We then obtain the actual contributions to each cluster, $\text{cont}_{\text{actual}}$ by summing the contributions of a neuron to the pixel values in the cluster, and contribution deltas are taken as $\delta_{\text{cont}} = \text{cont}_{\text{expected}} - \text{cont}_{\text{actual}}$. We then normalize each δ_{cont} by dividing by $\text{cont}_{\text{expected}}$ such that they are percentage differences between actual and expected neuron contributions to each cluster. Finally, we compute the variance (standard deviation) of the normalized percentage differences. The above is repeated for all neurons in a layer.

We plot the result of performing this computation for instances, RGB, Gabor, and gridcell clusters, across all neurons, in Figure 4. This computation reveals quantitatively what Figure 3 hints at – there is relatively little meaningful correlation between contributions and space, as seen in the low variances of gridcell representations. Instead there is in general a surprisingly high variance for neurons when we group their contributions using instance masks, which suggests that they learn some type of object semantics. Given that the variances for RGB and Gabor clusters are also quite high, we suggest that these are low-level semantics, a combination of color and edge features. There is also an inter-

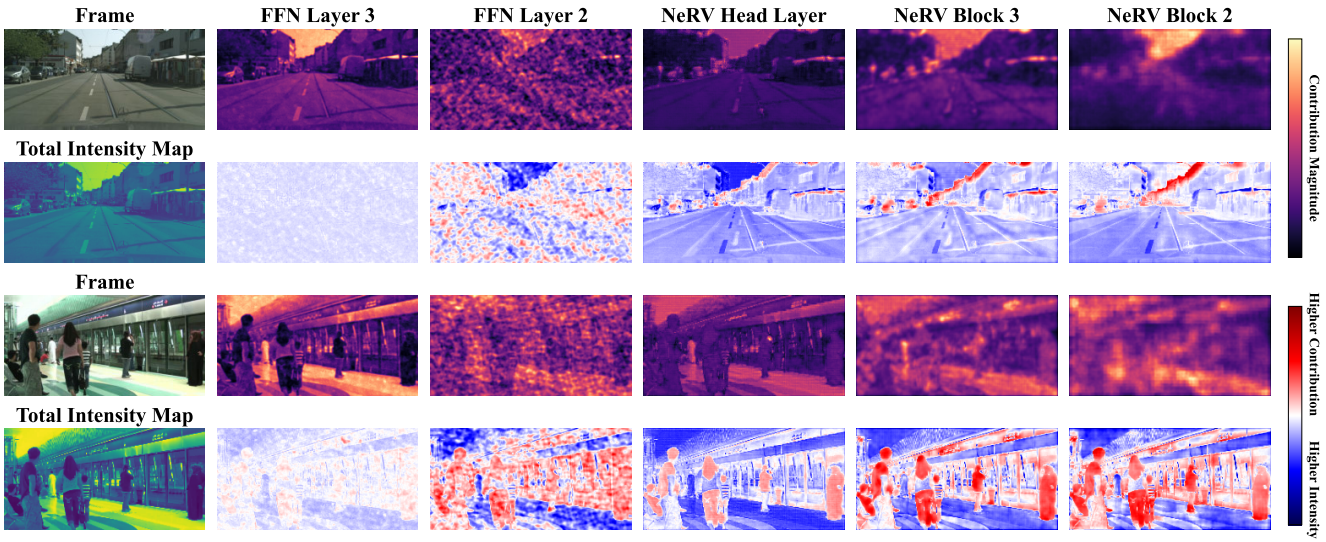

Figure 5. **Contribution vs. intensity.** We compare contribution and intensity in alternating rows. In the top row, we sum all contribution maps for the indicated layer. In the next row, we show the difference between this, and the raw image intensity (sum of all color channels) to show when contribution does and does not correlate with intensity. The next two rows repeat this for a frame from another video.

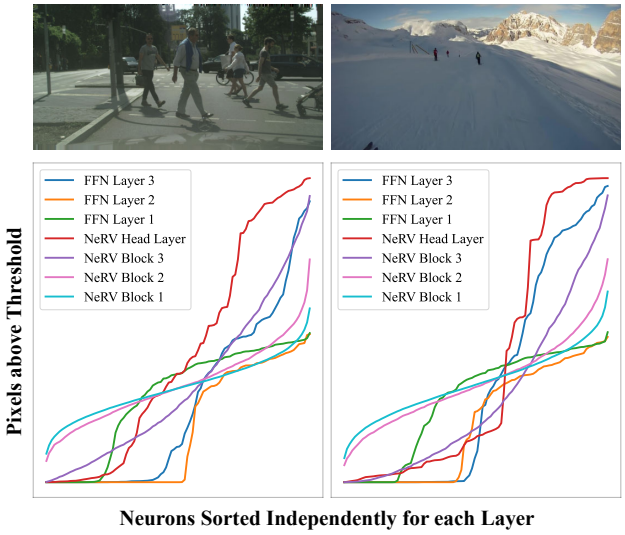

Figure 6. **Pixels per neuron.** We compute the pixels activated for each neuron in each layer, with each layer sorted independently by order of increasing number of pixels, for frames from two videos.

esting trend with the NeRV Block 3, where it seems to have some meaningful correlation with gridcells. However, we note that the top 25% of variances for Instances, RGB, and Gabor, are almost all higher than even the highest gridcell variance, and therefore we believe the intuition holds.

4.3. Contribution is not Intensity

Furthermore, we observe an interesting property that not all pixels require equal representation. Figure 5 shows the sum of absolute contributions over all neurons for distinct layers of the FFN and NeRV. We compare these aggregated, layer-level contribution maps to the total intensity (summing over color channels) of the image they correspond to. The heatmaps reveal a correspondence between the areas of large contributions and the areas of high intensity in the input image. This suggests that the neurons of an INR largely attend to higher intensities in the image.

While there is naturally some intensity bias, as a result of the pixelwise loss functions used by these networks, this does not fully explain the behavior of the representation. To discover other concepts that the INR potentially pays attention to, we subtract the intensity image from the contribution heat map. In Figure 5, this reveals that in addition to intensity, the INR layers also pay attention to other low-level information such as edges and textures. We note that the NeRV layers and the FFN layer 2 have a tendency to over-represent the people in the second image, relative to intensity, while the FFN layer 3 contributions are largely consistent with intensity. Additionally, not all edges are treated equally, with NeRV having large contributions for some edges, and small contributions for others.

4.4. Representation is Distributed

When training an INR, one might wonder, since there are more pixels than neurons, how many pixels are represented by each neuron, and if this representation tends to be distributed. To analyze the manner in which neuron-to-pixel contributions are distributed, we compute the ratio of a neuron's contribution to a pixel to the total contribution across neurons at a pixel. We threshold on this ratio to determine

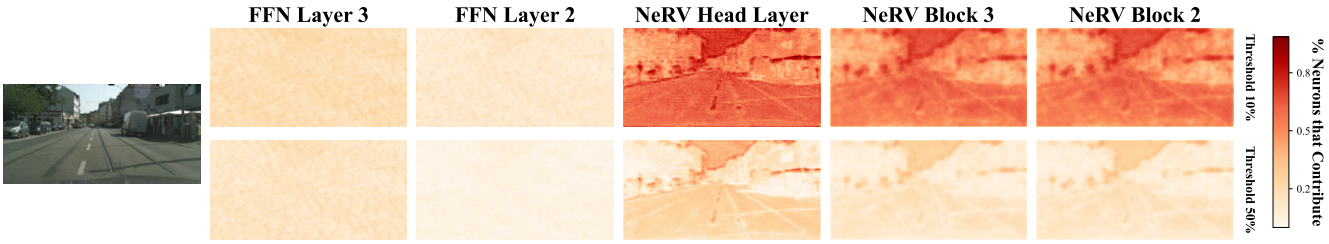

Figure 7. **Neurons per pixel.** We show how many neurons represent significant portions of each pixel, as a percentage of the total neurons in the indicated layer, at two different thresholds for "activation."

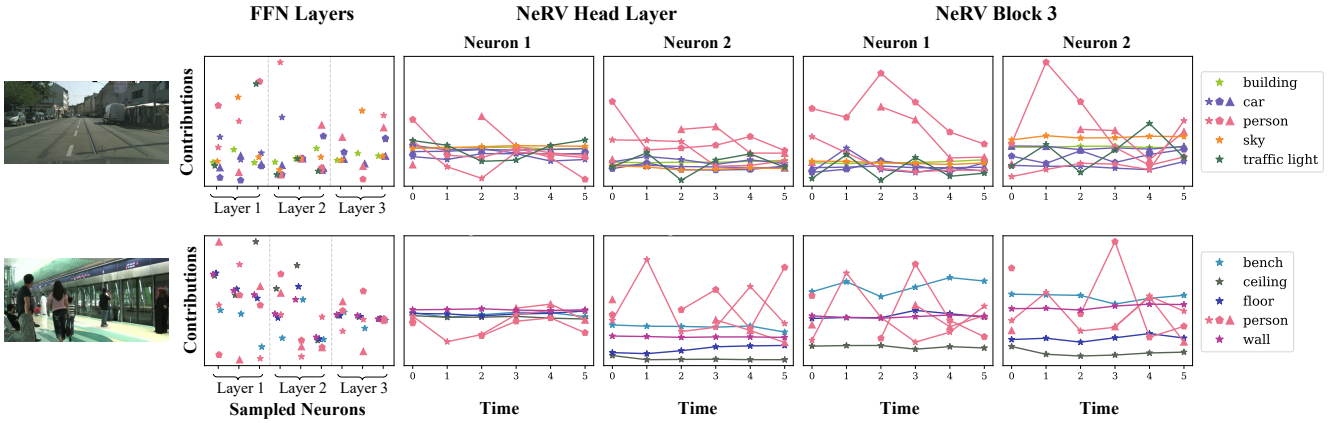

Figure 8. **Neuron contributions to things and stuff.** We aggregate the contributions of neurons to the instances (things) and background (stuff) in a frame from 2 videos. For FFN, we sample 3 neurons from each layer and for NeRV, we sample 2 neurons each from the last 2 layers. Instances belonging to the same category are depicted by distinct markers of the same color. For FFNs, we show contributions only for the given frames, whereas for NeRVs, we show these contributions over time for 6 frames from the source video.

whether a pixel receives a significant portion of its contribution from a specific neuron or not. The threshold used is a simple reciprocal of the number of neurons in a layer. We then compute the total number of pixels that a specific neuron contributes to meaningfully and sort this across neurons, repeating this analysis individually for different layers of the FFN and NeRV.

We show these contribution counts in Figure 6. Since different layers have different numbers of neurons, we resample the neurons of each layer to a common length using linear interpolation to preserve their trends. Notice how the outermost layers of the FFN and NeRV (layer 3 and head layer respectively) show the highest trends with a decreasing trend towards the earlier layers of both networks. Note that nearly 50% of the neurons in FFN Layer 2 have almost no contributions, producing a flat line in the first half. The presence of these dead neurons in this and other layers points to the suitability of these networks for data compression, and helps explain why extensive model pruning can be done with limited impact on reconstruction quality [7].

To further understand how the learned INRs distribute the image representations, we compute at each spatial image location, the density of neurons that contribute to it above a threshold, τ. To set τ for a certain layer, consider the sorted raw contributions of all neurons across locations of the image. We find that a large portion of the contributions are small and the number of larger contributions is fewer in comparison. For an illustration of this, see Figure 18. We obtain the raw contribution value that lies at the x-th percentile of this curve, i.e., the contribution of the neuron to the left of which lies $x\%$ of the area under the curve. The value corresponding to this percentile is chosen as the threshold for all neuron contribution maps. We show this for the 10^{th} and 50^{th} percentile thresholds in Figure 7, which shows the regions of the image with a higher density of dedicated neurons.

High intensity and highly textured areas not only have high contributions as seen earlier in Figure 5, but they also have a large portion of neurons dedicated to them. Increasing the threshold highlights the image areas with the highest neuron density, and this reveals an overwhelming bias for edges with NeRV. However, all these trends are greatly subdued for the FFN at both its head and penultimate layers, suggesting that while the magnitude of its contributions correlates almost perfectly with the image intensity, its actual representation is more evenly distributed by comparison.

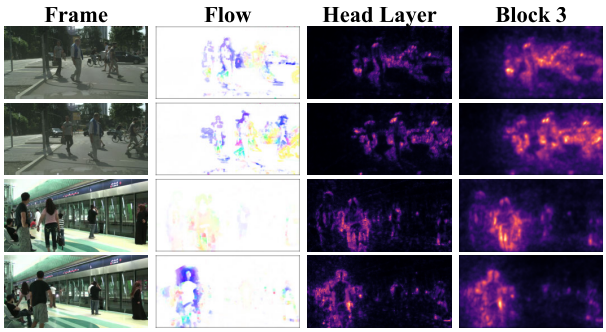

Figure 9. **Neurons and motion.** We show the correlation between motion and changes in neuron contribution over time by computing optical flow between two frames and the difference in contribution maps between those same frames. We plot both of these. Fluctuation is driven by motion, and it seems the areas revealed by motion matter equally to the objects that are actually moving.

4.5. Objects, Categories, and Motion

To analyze whether INRs have meaningful representations in terms of objects, we aggregate neuron contributions over pixels in each instance's segmentation map. Consider the two sample videos shown in Figure 8. Each has a different set of objects with some categories having multiple instances. Instances may move, disappear and reappear over time. For the FFN, we sample three neurons each from the three layers and depict each neuron's normalized contributions to every instance (as a percentage of contributions to all instances) in the first frame of the video. Notice how contributions to instances of the same category are not grouped together. This is related to our findings in Figure 4, and motivates our claim that these networks learn low-level object features, but lack class-level semantics.

For NeRV, we sample two neurons each from the head layer Block 3, and track the normalized per-instance contributions over multiple frames of the video. We see that a neuron's contributions to an instance remains relatively static over time. When there is fluctuation, we notice that this is often correlated with instances that move, such as persons. Another thing to note is how contributions to high intensity instances such as sky are higher, which is in agreement with our earlier observations.

Further delving into how neurons in NeRV respond to motion, we visualize the optical flow map and the fluctuation of total contribution in the last 2 layers of a NeRV. Figure 9 depicts this for two different time steps in each video (the flow and contribution fluctuation are computed between a specific frame and the immediately preceding frame). We see that the changes in contributions are focused more in areas around the regions of flow. It appears that when a moving object reveals new background, this causes fluctuations in the spatially proximal pixels, even though they technically contain no motion themselves. Thus, we see that in spite of its lack of high-level object semantics, the changes in how a NeRV represents a video across time are dominated by the motion of the entities in the video.

4.6. Neurons can be Clustered by Contributions

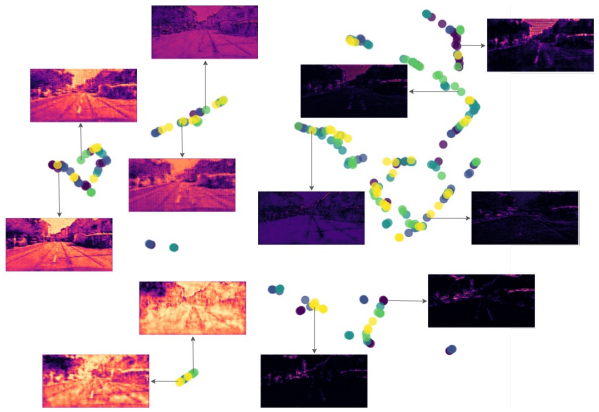

Figure 10. **UMAP for INRs.** We plot the neurons from the head layers of NeRVs with different seeds, and show the contribution maps corresponding to the neurons indicated.

We propose a unique approach for computing a vector representation of INR neurons. Inspired by the relatively high variance for Gabor clusters in Figure 4, we take the pixels in each frame of each video, and cluster them according to Gabor filter features with k clusters. We then aggregate the contribution map values according to the clusters, yielding a k dimensional vector representation of every neuron. We can then project these representations with UMAP to allow for plotting them in 2D space, which we do in Figure 10. As this figure shows, there are many different "types" of neurons when we separate them according to edge contributions. This representation mechanism we develop with XINC provides another powerful method for visualizing and examining the internals of INR.

5. Conclusion

In this paper, we propose XINC, a novel framework for analyzing INRs for image and video representation. Using XINC, we find interesting trends with CNNs and MLPs, such as how while they lack high-level object semantics, they do have biases that mimic object representation, particularly with edges and colors. We also show how fluctuations in contribution for NeRV are driven by motion. We show how our framework can help us identify interesting groups of related neurons. We provide both the framework and these findings to help drive future research for INRs.

Acknowledgements. We would like to thank Soumik Mukhopadhyay for his helpful feedback while we prepared the manuscript. This project was partially funded by NSF CAREER Award (#2238769) to AS.

References

[1] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: a survey on explainable artificial intelligence (xai). *IEEE access*, 6:52138–52160, 2018. 1

[2] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58:82–115, 2020. 1

[3] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6541–6549, 2017. 1, 2

[4] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B Tenenbaum, William T Freeman, and Antonio Torralba. Gan dissection: Visualizing and understanding generative adversarial networks. *arXiv preprint arXiv:1811.10597*, 2018. 2

[5] Lorenzo Catania and Dario Allegra. Nif: A fast implicit image compression with bottleneck layers and modulated sinusoidal activations. In *Proceedings of the 31st ACM International Conference on Multimedia*, page 9022–9031, New York, NY, USA, 2023. Association for Computing Machinery. 1

[6] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-CAM++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2018. 2

[7] Hao Chen, Bo He, Hanyu Wang, Yixuan Ren, Ser Nam Lim, and Abhinav Shrivastava. Nerv: Neural representations for videos. *Advances in Neural Information Processing Systems*, 34:21557–21568, 2021. 1, 2, 3, 5, 7, 12

[8] Hao Chen, Matt Gwilliam, Bo He, Ser-Nam Lim, and Abhinav Shrivastava. Cnerv: Content-adaptive neural representation for visual data, 2022. 2

[9] Hao Chen, Matthew Gwilliam, Ser-Nam Lim, and Abhinav Shrivastava. Hnerv: A hybrid neural representation for videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10270–10279, 2023. 1, 2, 12

[10] Yinbo Chen and Xiaolong Wang. Transformers as meta-learners for implicit neural representations, 2022. 1, 2

[11] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding, 2016. 1

[12] Rachel Lea Draelos and Lawrence Carin. Use hirescam instead of grad-cam for faithful explanations of convolutional neural networks, 2021. 2

[13] Emilien Dupont, Adam Goliński, Milad Alizadeh, Yee Whye Teh, and Arnaud Doucet. Coin: Compression with implicit neural representations. *arXiv preprint arXiv:2103.03123*, 2021. 1, 2

[14] Emilien Dupont, Hrushikesh Loya, Milad Alizadeh, Adam Golinski, Yee Whye Teh, and Arnaud Doucet. Coin++: Data agnostic neural compression. *arXiv preprint arXiv:2201.12904*, 1(2):4, 2022. 1, 2

[15] Ruigang Fu, Qingyong Hu, Xiaohu Dong, Yulan Guo, Yinghui Gao, and Biao Li. Axiom-based grad-cam: Towards accurate visualization and explanation of cnns, 2020. 2

[16] D. Gabor. Theory of communication. part 1: The analysis of information. *Journal of the Institution of Electrical Engineers - Part III: Radio and Communication Engineering*, 93: 429–441(12), 1946. 5, 12

[17] Kyle Gao, Yina Gao, Hongjie He, Dening Lu, Linlin Xu, and Jonathan Li. Nerf: Neural radiance field in 3d vision, a comprehensive review, 2023. 1

[18] Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pages 80–89. IEEE, 2018. 1

[19] Bo He, Xitong Yang, Hanyu Wang, Zuxuan Wu, Hao Chen, Shuaiyi Huang, Yixuan Ren, Ser-Nam Lim, and Abhinav Shrivastava. Towards scalable neural representation for diverse videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6132–6142, 2023. 1, 2

[20] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023. 12

[21] Evan Hernandez, Sarah Schwettmann, David Bau, Teona Bagashvili, Antonio Torralba, and Jacob Andreas. Natural language descriptions of deep visual features. In *International Conference on Learning Representations*, 2022. 2

[22] Derek Hoiem, Yodsawalai Chodpathumwan, and Qieyun Dai. Diagnosing error in object detectors. In *European conference on computer vision*, pages 340–353. Springer, 2012. 2

[23] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls, 2020. 2

[24] Neha Kalibhat, Shweta Bhardwaj, Bayan Bruss, Hamed Firooz, Maziar Sanjabi, and Soheil Feizi. Identifying interpretable subspaces in image representations, 2023. 2

[25] Chiheon Kim, Doyup Lee, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Generalizable implicit neural representations via instance pattern composers. *arXiv preprint arXiv:2211.13223*, 2022. 2

[26] Dahun Kim, Sanghyun Woo, Joon-Young Lee, and In So Kweon. Video panoptic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 4

[27] Subin Kim, Sihyun Yu, Jaeho Lee, and Jinwoo Shin. Scalable neural video representations with learnable positional features. In *Advances in Neural Information Processing Systems*, 2022. 2

[28] Biagio La Rosa, Leilani H. Gilpin, and Roberto Capobianco. Towards a fuller understanding of neurons with clustered compositional explanations. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 2

[29] Grace W Lindsay and David Bau. Testing methods of neural systems understanding. *Cognitive Systems Research*, 82: 101156, 2023. 2

[30] Shangdong Liu, Puming Cao, Yujian Feng, Yimu Ji, Jiayuan Chen, Xuedong Xie, and Longji Wu. Nrvc: Neural representation for video compression with implicit multiscale fusion network. *Entropy*, 25(8), 2023. 1

[31] Luca De Luigi, Adriano Cardace, Riccardo Spezialetti, Pierluigi Zama Ramirez, Samuele Salti, and Luigi Di Stefano. Deep learning on implicit neural representations of shapes, 2023. 2

[32] Shishira R Maiya, Sharath Girish, Max Ehrlich, Hanyu Wang, Kwot Sin Lee, Patrick Poirson, Pengxiang Wu, Chen Wang, and Abhinav Shrivastava. Nirvana: Neural implicit representations of videos with adaptive networks and autoregressive patch-wise modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14378–14387, 2023. 1, 2

[33] R. Mehrotra, K.R. Namuduri, and N. Ranganathan. Gabor filter-based edge detection. *Pattern Recognition*, 25(12): 1479–1494, 1992. 5, 12

[34] Jiaxu Miao, Yunchao Wei, Yu Wu, Chen Liang, Guangrui Li, and Yi Yang. Vspw: A large-scale dataset for video scene parsing in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4133–4143, 2021. 4

[35] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020. 1, 2

[36] Jesse Mu and Jacob Andreas. Compositional explanations of neurons, 2021. 2

[37] Vineel Nagisetty, Laura Graves, Joseph Scott, and Vijay Ganesh. xai-gan: Enhancing generative adversarial networks via explainable ai systems, 2022. 2

[38] Meike Nauta, Jan Trienes, Shreyasi Pathak, Elisa Nguyen, Michelle Peters, Yasmin Schmitt, Jörg Schlötterer, Maurice van Keulen, and Christin Seifert. From anecdotal evidence to quantitative evaluation methods: A systematic review on evaluating explainable ai. *ACM Computing Surveys*, 55(13s): 1–42, 2023. 1

[39] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision, 2020. 1

[40] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space, 2019. 1

[41] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. https://distill.pub/2020/circuits/zoom-in. 2

[42] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. 1

[43] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007. 3

[44] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization, 2019. 1

[45] Vishwanath Saragadam, Daniel LeJeune, Jasper Tan, Guha Balakrishnan, Ashok Veeraraghavan, and Richard G. Baraniuk. Wire: Wavelet implicit neural representations, 2023. 2

[46] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017. 1, 2

[47] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network, 2016. 3, 4

[48] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020. 1, 2

[49] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations, 2020. 1

[50] Yannick Strümpler, Janis Postels, Ren Yang, Luc Van Gool, and Federico Tombari. Implicit neural representations for image compression. In *ECCV*, 2022. 1

[51] 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. *NeurIPS*, 2020. 3, 5

[52] Andong Wang, Wei-Ning Lee, and Xiaojuan Qi. Hint: Hierarchical neuron concept explainer, 2022. 2

[53] Dejia Xu, Peihao Wang, Yifan Jiang, Zhiwen Fan, and Zhangyang Wang. Signal processing for implicit neural representations, 2022. 2

[54] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021. 1

[55] Gizem Yüce, Guillermo Ortiz-Jiménez, Beril Besbinar, and Pascal Frossard. A structured dictionary perspective on implicit neural representations, 2022. 1, 2

[56] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 1

[57] Yunfan Zhang, Ties van Rozendaal, Johann Brehmer, Markus Nagel, and Taco Cohen. Implicit neural video compression, 2021. 1

[58] Qi Zhao, M. Salman Asif, and Zhan Ma. Dnerv: Modeling inherent dynamics via difference neural representation

for videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2031–2040, 2023. 1, 2