Functional Neural Networks for Parametric Image Restoration Problems

Anonymous Author(s)
Affiliation
Address
email

Abstract

Almost every single image restoration problem has a closely related parameter, such as the scale factor in super-resolution, the noise level in image denoising, and the quality factor in JPEG deblocking. Although recent studies on image restoration problems have achieved great success due to the development of deep neural networks, they handle the parameter involved in an unsophisticated way. Most previous researchers either treat problems with different parameter levels as independent tasks, and train a specific model for each parameter level; or simply ignore the parameter, and train a single model for all parameter levels. The two popular approaches have their own shortcomings. The former is inefficient in computing and the latter is ineffective in performance. In this work, we propose a novel system called functional neural network (FuncNet) to solve a parametric image restoration problem with a single model. Unlike a plain neural network, the smallest conceptual element of our FuncNet is no longer a floating-point variable, but a function of the parameter of the problem. This feature makes it both efficient and effective for a parametric problem. We apply FuncNet to super-resolution, image denoising, and JPEG deblocking. The experimental results show the superiority of our FuncNet on all three parametric image restoration tasks over the state of the arts.

1 Introduction

Image restoration [34] is a classical yet still active topic in low-level computer vision, which estimates the original image from a degraded measurement. For example, single image super-resolution [14] estimates the high-resolution image from a downsampled one, image denoising [4] estimates the clean image from a noisy one, and JPEG deblocking [7] estimates the original image from a compressed one. It is a challenging ill-posed inverse problem which aims to recover the information lost to the image degradation process [3], and it is also important since it is an essential step in various image processing and computer vision applications [60, 48, 2, 28, 40, 26].

Almost every single image restoration problem has a closely related parameter, such as the scale factor in super-resolution, the noise level in image denoising, and the quality factor in JPEG deblocking. The parameter in an image restoration problem tends to have a strong connection with the image degradation process. In super-resolution problem, the blur kernel of the downsampling process is determined by the scale factor [51]. In image denoising problem, the standard deviation of the additive white Gaussian noise is determined by the noise level [9]. In JPEG deblocking problem, the quantization table for DCT coefficients is determined by the quality factor [36]. When we try to restore the clean image from a corrupt one, we might know the value of the corresponding parameter for various reasons. In super-resolution problem, the scale factor is specified by users [23]. In image denoising problem, the noise level could be measured by other devices [37, 32]. In JPEG deblocking

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

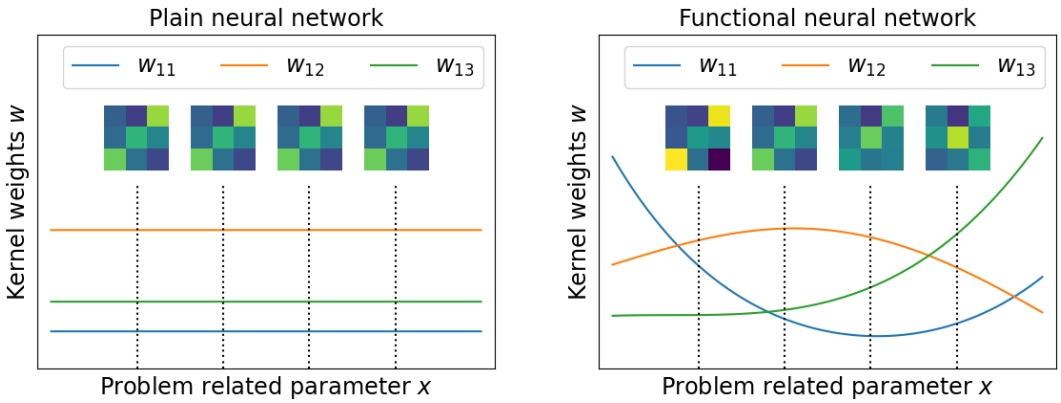

Figure 1: The difference between plain neural network and our functional neural network (FuncNet). The left and right figure visualize a 3×3 convolution kernel in a plain neural network and its counterpart in a FuncNet respectively. For the kernel in a plain network, its weights remain unchanged for different problem related parameter levels, so the network only has a limited adaptability to parametric image restoration problems. Unlike a plain network, the smallest conceptual element of our FuncNet is no longer a floating-point variable, but a function of the problem related parameter. In other words, the kernel weights of our FuncNet can change for different situations and make our FuncNet perform better for parametric image restoration problems.

problem, the quality factor could be derived from the header of the JPEG file [8]. Therefore, it is very important to use the known parameter well in such a parametric image restoration problem.

Recently, deep convolutional neural network based methods are widely used to tackle the image restoration tasks, including super-resolution [12, 23, 39, 29, 41, 44, 57, 56, 10, 35], image denoising [20, 5, 53, 27, 42, 52, 54, 50, 1], and JPEG deblocking [31, 11, 45, 16, 6, 15, 30, 55, 13]. They have achieved significant improvements over conventional image restoration methods due to their powerful learning ability. However, they have not been paying attention to the parameter involved in an image restoration problem, and handled it in an unsophisticated way. Most previous researchers either treat problems with different parameter levels as independent tasks, and train a specific model for each parameter level [12, 39, 56, 52, 55]; or simply ignore the parameter, and train a single model for all parameter levels [23, 50]. The two popular approaches have their own shortcomings. The former is inefficient in computing, because they may have to train and store dozens of models for different parameter levels. The latter is ineffective in performance, since they ignore some important information that could have helped the restoration process.

To overcome these weaknesses, we propose a novel system called functional neural network (FuncNet) to solve a parametric image restoration problem with a single model. The difference between plain neural network and our FuncNet is shown in Figure 1. Unlike a plain neural network, the smallest conceptual element of our FuncNet is no longer a floating-point weight, but a function of the parameter of the problem. When we train a FuncNet, we gradually change these functions to reduce the value of the loss function. When we use the trained FuncNet to restore a degraded image with a given parameter, we first evaluate those functions with the parameter, use the evaluation values as weights of the network, and then do inference as normal. This feature makes it both efficient and effective for a parametric problem. By this way, we neatly blend the parameter information into a neural network model, use it to help the restoration process, and only increase a negligible amount of computation as we will demonstrate later. We apply FuncNet to super-resolution, image denoising, and JPEG deblocking. The experimental results show the superiority of our FuncNet on all three parametric image restoration tasks over the state of the arts.

2 Related Work

Neural networks for parametric problems. To the best of our knowledge, there are six ways to solve a parametric problem with neural network based method. We list them below roughly in the order of popularity, and discuss their advantages and disadvantages.

The first method treats problems with different parameter levels as independent tasks, and trains a specific model for each parameter level [12, 39, 56, 52, 55]. The overwhelming majority of previous papers use this approach. This method is easy to understand, and generally has good performance since the parameter level is implied in a model. But it is inefficient in computing, because we may have to train and store dozens of models for different parameter levels.

The second method simply ignores the parameter, and trains a single model for all parameter levels [23, 50]. This method is also easy to understand, and is very efficient since we only need to train and store a single model. But its performance is typically lower than the first method, since we ignore some important information that could have helped the restoration process.

The third method trains a model with a shared backbone for all parameter levels and multiple specific blocks for each parameter levels [29]. It is a compromise between the first and the second method, and has acceptable performance and efficiency. But we may still have to store dozens of specific blocks for different parameter levels. And if the capacity of a specific block is not large enough, the block cannot take full advantage of the parameter information.

The fourth method converts the parameter scalar into a parameter map, and treats the map as an additional channel of the input degraded image [54]. It is another way to blend the parameter information into a neural network model. However, the performance of this method is only marginally higher than the second method, and it is still not as good as the first method. Due to the huge semantic difference between the corrupt image and the parameter map, it is hard to make much use of the parameter information for the model.

The fifth method trains a model with a relatively shallow backbone network, and each filter of the backbone network is generated by a different filter generating network [25, 21, 22, 13]. The filter generating networks are usually multilayer perceptrons, and they take the parameter as input. Since the total size of a model is limited, assigning each filter a unique complex network severely limits the size of the backbone network. Such a shallow backbone network only leads to a mediocre performance. Considering the universal approximation ability of the multilayer perceptron, this method is not that different from training a unique shallow model for each parameter level.

The sixth method searches in the latent space of a generative model, and returns the most probable result which is not contradicting the degradation model with the parameter [47, 33]. It is a general image restoration method which can solve various image restoration problems with a single model, as long as the degradation model is continuously differentiable. However, this method is slower than a feedforward neural network based method, since it requires multiple forward and backward passes in a search. And due to the limited representation ability of the generative model, the performance of this method is also worse than a discriminative model based method.

Deep CNN for Single Image Super-Resolution. The first convolutional neural network for single image super-resolution is proposed by Dong et al. [12] called SRCNN, and it achieved superior performance against previous works. Shi et al. [39] firstly proposed a real-time super-resolution algorithm ESPCNN by proposing the sub-pixel convolution layer. Lim et al. [29] removed batch normalization layers in the residual blocks, and greatly improved the SR effect. Zhang et al. [56] introduced the residual channel attention to the SR framework. Hu et al. [18] proposed the Meta-Upscale Module to replace the traditional upscale module.

Deep CNN for Image Denoising. Zhang et al. [52] proposed DnCNN, a plain denoising CNN method which achieves state-of-the-art denoising performance. They showed that residual learning and batch normalization are particularly useful for the success of denoising. Tai et al. [42] proposed MemNet, a very deep persistent memory network by introducing a memory block to mine persistent memory through an adaptive learning process. Zhang et al. [54] proposed FFDNet, a fast and flexible denoising convolutional neural network, with a tunable noise level map as the input.

Deep CNN for JPEG deblocking. Dong et al. [11] proposed ARCNN, a compact and efficient network for seamless attenuation of different compression artifacts. Guo and Chao [16] proposed a highly accurate approach to remove artifacts of JPEG-compressed images, which jointly learned a very deep convolutional network in both DCT and pixel domains. Zhang et al. [55] proposed DMCNN, a Dual-domain Multi-scale CNN to take full advantage of redundancies on both the pixel and DCT domains. Ehrlich et al. [13] proposed a novel architecture which is parameterized by the JPEG files quantization matrix.

3 Functional Neural Network (FuncNet)

In this section, we describe the proposed FuncNet model. To transform a plain neural network into a FuncNet, we replace every trainable variable in a plain neural network by a specific function, such as weights and biases in convolution layers or fully connected layers, affine parameters in Batch Normalization layers [19], and slopes in PReLU activation layers [17]; and keep other layers without trainable variables unchanged, such as pooling layers, identity layers, and pixel shuffle layers [39]. We first describe the method to specify the functions in our FuncNet models, then we describe the initialization, training and inference method for these functions, next we describe the network architectures to contain those functions, and finally we analyse the storage and computational efficiency of our FuncNet models.

3.1 Specification of Functions

The functions used in our FuncNet model should be simple enough. Let us consider the following failure case as a negative example. Suppose the number of the parameter levels is large but still finite, and we use polynomial functions in our FuncNet. If the polynomial function is too complex, and its degree is greater than or equal to the number of the parameter levels minus one, then our FuncNet is no different from training a specific model for each parameter level. In this case, the failing FuncNet takes as much or even more storage space than multiple independent models, and its inference speed is also slightly slower than a same size plain neural network. This negative example demonstrates the necessity of choosing simple functions for our FuncNet.

We choose the simplest and the most basic kind of function, the linear function, as the functions used in our FuncNet model. In this case, the FuncNet model takes exactly double storage space than a same size plain neural network, and it is still more efficient than storing dozens of models for different parameter levels. And it only increases a negligible amount of computations than a same size plain neural network, as we will demonstrate later. Choosing such a simple function does not lead to a poor performance of the final FuncNet model. With multiple activation layers, the final FuncNet model retains the power of nonlinear fitting. The linear function used in our FuncNet model can be defined as:

$$G(x; \theta_a, \theta_b) = \frac{x - x_a}{x_b - x_a}(\theta_b - \theta_a) + \theta_a \tag{1}$$

where x is the parameter of the problem, x_a and x_b are lower and upper bound of the support of the parameter distribution respectively, θ_a and θ_b are trainable variables, and $G(x; \theta_a, \theta_b)$ is the function used in our FuncNet model to generate variables for different parameter levels.

The parameters have different properties for different problems, and we can make the linear function 1 to suit different problems better by replacing x with $H(x)$, where $H(x)$ is a problem-related function. Then the linear function 1 will become:

$$G(H(x); \theta_a, \theta_b) = \frac{H(x) - H(x_a)}{H(x_b) - H(x_a)}(\theta_b - \theta_a) + \theta_a \tag{2}$$

The chosen $H(x)$ should have a physical interpretation related to the problem, and of course should make the final FuncNet model perform well. In super-resolution problem, we use $H(x) = 1/x$ because the reciprocal of the scale factor is the rescaled length on a low-resolution image from a unit length on a high-resolution image. In image denoising problem, we use $H(x) = x$ because the noise level is equal to the standard deviation of the additive white Gaussian noise. In JPEG deblocking problem, we use $H(x) = 5000/x$ for $x \leq 50$ and $H(x) = 200 - 2x$ for $x > 50$. This is the formula used in JPEG standard [36], and it transforms the quality factor into a scale factor of the quantization table for DCT coefficients. The choices of $H(x)$ for different problems are still empirical, just like when people determine the depth, the width or other configuration for a neural network. But we hope that we can determine $H(x)$ automatically in the future, just like what people do in Neural Architecture Search right now [59].

3.2 Initialization, Training and Inference

Proper initialization is crucial for training a neural network. Even with modern structures and normalization layers, a bad initialization can still hamper the learning of the highly nonlinear system.

The goal of initialization for a plain neural network is to set the value of every trainable variable in a proper range, and to avoid reducing or magnifying the magnitude of input signals exponentially. To properly initialize a function in FuncNet, we have to guarantee that all possible output values of the function are in a proper range. Suppose $H(x)$ in function 2 is a monotonic function, then what we need to do is to use initialization algorithm for a plain neural network [17] to initialize θ_a and θ_b independently. In this way, all possible output values of the function 2 lie somewhere between θ_a and θ_b, and must also be in a proper range if θ_a and θ_b are well set. If θ_a and θ_b are both sampled from a zero-mean distribution whose standard deviation is σ, then for all possible output values of the function 2, their expected values are still zero, and their standard deviations are between $\sigma/\sqrt{2}$ and σ. Experiments have shown that such a small deviation is acceptable for training.

Training a FuncNet is not very different from training a plain neural network. In every iteration, we first sample a fixed number of parameter levels uniformly, use them to construct a minibatch, and then perform stochastic gradient based optimization as normal. As suggested in [58], we train our FuncNet using L1 loss. During the training, we gradually change the values of trainable variables θ_a and θ_b to reduce the value of the loss function. The problem can be formulated as

$$\min_{\theta_a,\theta_b} \mathbb{E}_{I^O,I^D,x} \| F(I^D; G(H(x); \theta_a, \theta_b)) - I^O \|_1 \tag{3}$$

where I^D is the degraded version of its original counterpart I^O, and F is our FuncNet model.

When we use the trained FuncNet to restore a degraded image with a given parameter x, we first evaluate function 2 with x, trained θ_a and θ_b, use the evaluation values as variables of the corresponding plain neural network, and then do inference with the generated plain neural network as normal.

3.3 Network Architectures

The requirements of the networks for the three image restoration problems are very different. For super-resolution problem, the degradation is deterministic and relatively mild, and the network can concentrate on a relatively small area. For image denoising problem, the degradation is random and relatively severe, and the network need to pay attention to a larger area. For JPEG deblocking problem, the degradation occurs in the DCT domain, and the network should have ability to utilize the information in the DCT domain. So we use individually designed architectures for the three image restoration problems to meet their own requirements.

We directly use architectures of the state-of-the-art plain neural networks for the three tasks as the architectures of our FuncNet models, and we only make essential modifications to them. For super-resolution problem, we use the architecture of RCAN [56], and replace the upscale module for integer scale factors [39] with the meta upscale module for non-integer scale factors [18]. For image denoising problem, we apply a modified U-net [38] structure as the backbone and use RCAB [56] as residual blocks. For JPEG deblocking problem, we use the architecture of DMCNN [55] for reference. For the DCT domain branch of our JPEG deblocking model, we use frequency component rearrangement to get a more meaningful DCT representation as suggested in [13]; and for the pixel domain branch, we use the same architecture of our image denoising model. More detailed information can be found in the supplementary material.

3.4 Storage and Computational Efficiency Analysis

Our FuncNet models have high storage efficiency. As we described in Section 3.1, we use two-degree-of-freedom functions in FuncNet models. This takes only twice as much space as what a plain neural network with the same architecture will take. So storing a FuncNet model is much cheaper than storing dozens of plain networks for different parameter levels. Take super-resolution task as an example. Suppose the scale factor varies from 1.1 to 4 with stride 0.1 as suggested in [18], we can save 93.3% on storage space by using FuncNet rather than plain neural networks.

Our FuncNet models have high computational efficiency as well. For the training phase, we only need to train one FuncNet model rather than to train dozens of plain networks individually. The computational efficiency analysis for this phase is similar to the preceding storage efficiency analysis. For the inference phase, as we described in Section 3.2, we first evaluate functions with the problem related parameter, use the evaluation values as variables of the corresponding plain neural network,

and then do inference with the generated plain neural network as normal. So compared to a plain neural network with the same architecture, our FuncNet model only needs a little extra effort to evaluate functions. This part of computation is directly proportional to the number of parameters in the corresponding plain network, and it is several orders of magnitude smaller than the number of multi-adds for a plain image restoration network. Still take super-resolution task as an example. Suppose we need to double the size of a 360p image, our FuncNet model only needs extra 0.0001% computation than a plain neural network with the same architecture.

4 Experiments

4.1 Training Settings

Training datasets. Following [56, 18, 50], we use DIV2K dataset [43] for training. There are 1000 high-quality images in DIV2K dataset, 800 images for training, 100 images for validation and 100 images for testing. All our three FuncNet models for the three parametric image restoration tasks are trained with DIV2K training images set.

Parametric settings. In all three parametric problems, the problem related parameters are sampled uniformly. In super-resolution problem, the training scale factors vary from 1.1 to 4 with stride 0.1. In image denoising problem, the training noise levels are sampled from the uniform distribution on the interval (0, 75]. In JPEG deblocking problem, the quality factors vary from 10 to 80 with stride 2.

Degradation models. In super-resolution problem, we use the bicubic interpolation by adopting the Matlab function imresize to simulate the LR images. In image denoising problem, we generate the additive white Gaussian noise dynamically by using the Numpy function. In JPEG deblocking problem, we use the Matlab JPEG encoder to generate the JPEG images.

Data augmentations. In all three parametric problems, we use the same data augmentation method. We randomly augment the image patches by flipping horizontally, flipping vertically and rotating 90°.

Optimization settings. In super-resolution problem, we randomly extract 32 LR RGB patches with the size of 40×40 as a batch input. In image denoising problem, we randomly extract 32 RGB patches with the size of 96×96 as a batch input. In JPEG deblocking problem, we randomly extract 32 gray patches with the size of 96×96 as a batch input, and we make sure that the image patches are aligned with boundaries of Minimum Coded Unit blocks. All our three FuncNet models are trained by ADAM optimizor with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The initial learning rate is set to 10^{-4} and then decreases by half for every 2×10^5 iterations of back-propagation. All experiments run in parallel on 4 GPUs.

4.2 Evaluation on Standard Benchmark Datasets

In super-resolution problem, we use B100 dataset for non-integer scale factor testing, and we use five standard benchmark datasets for integer scale factor testing: Set5, Set14, B100, Urban100, and Manga109. The results are evaluated with PSNR and SSIM [46] on Y channel of transformed YCbCr space. In image denoising problem, we use three standard benchmark datasets: CBSD68, Kodak24, and McMaster. The results are evaluated with PSNR and SSIM [46] on RGB channel as suggested in [52]. In JPEG deblocking problem, we use two standard benchmark datasets: LIVE1 and BSDS500. The results are evaluated with PSNR, SSIM [46], and PSNR-B [49] on Y channel.

We compare our results with those of state-of-the-art methods for all three parametric problems. Similar to [29], we also apply self-ensemble strategy to further improve our FuncNet model and denote the self-ensembled one as FuncNet+. The quantitative results are shown in Table 1, 2, 3, and 4. The visual comparisons are shown in Figure 2, 3, and 4. More detailed information can be found in the supplementary material.

4.3 Kernel Visualization and Interpretation

We visualize kernels of our FuncNet model and try to understand and interpret them. The key point of the analysis is to find out how kernels change with the problem related parameter. Here we show samples of kernels from the input and the output layer of the denoising FuncNet, since the denoising problem has the most definite physical meaning among the three image restoration problems. The

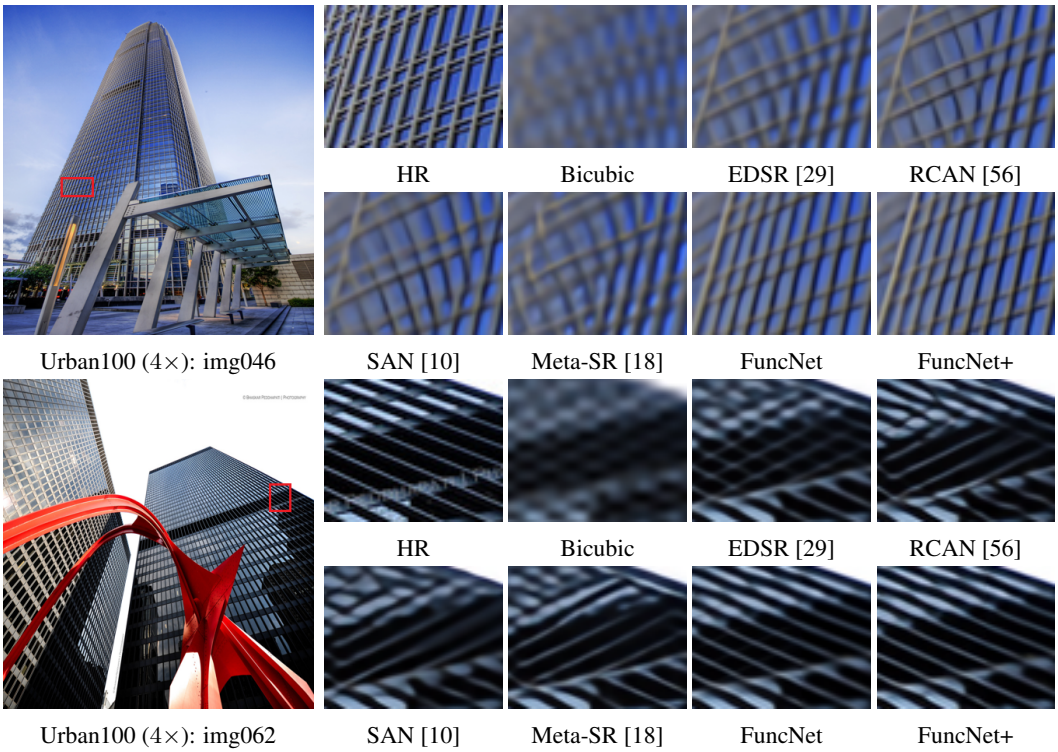

Figure 2: Visual comparison between different super-resolution methods

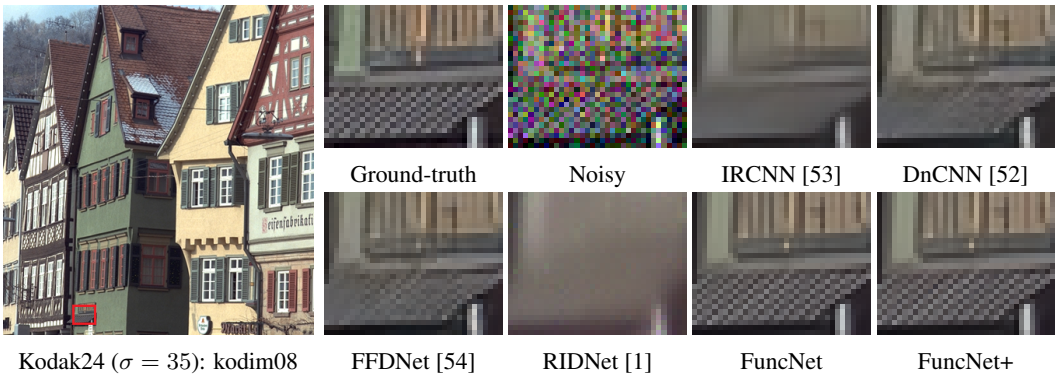

Figure 3: Visual comparison between different image denoising methods

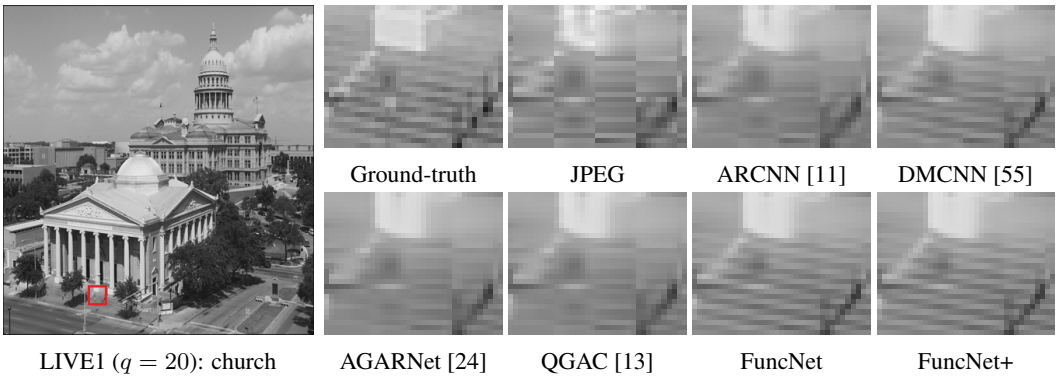

Figure 4: Visual comparison between different JPEG deblocking methods

Table 1: Results of decimal upscale SR on B100. Best and second best results are **highlighted** and underlined

Method \ Scale	×1.1	×1.2	×1.3	×1.4	×1.5	×1.6	×1.7	×1.8	×1.9
Bicubic	36.56	35.01	33.84	32.93	32.14	31.49	30.90	30.38	29.97
Meta-SR [18]	42.82	40.04	38.28	36.95	35.86	34.90	34.13	33.45	32.86
FuncNet	**43.43**	40.41	38.55	37.16	36.02	35.08	34.26	33.60	32.98
FuncNet+	43.36	**40.46**	**38.59**	**37.21**	**36.06**	**35.12**	**34.30**	**33.64**	**33.02**
Method \ Scale	**×2.1**	**×2.2**	**×2.3**	**×2.4**	**×2.5**	**×2.6**	**×2.7**	**×2.8**	**×2.9**
Bicubic	29.18	28.87	28.57	28.31	28.13	27.89	27.66	27.51	27.31
Meta-SR [18]	31.82	31.41	31.06	30.62	30.45	30.13	29.82	29.67	29.40
FuncNet	31.99	31.59	31.23	30.87	30.58	30.30	30.05	29.77	29.59
FuncNet+	**32.02**	**31.62**	**31.26**	**30.90**	**30.62**	**30.34**	**30.09**	**29.81**	**29.63**
Method \ Scale	**×3.1**	**×3.2**	**×3.3**	**×3.4**	**×3.5**	**×3.6**	**×3.7**	**×3.8**	**×3.9**
Bicubic	26.98	26.89	26.59	26.60	26.42	26.35	26.15	26.07	26.01
Meta-SR [18]	28.87	28.79	28.68	28.54	28.32	28.27	28.04	27.92	27.82
FuncNet	29.17	29.02	28.81	28.62	28.46	28.34	28.21	28.06	27.93
FuncNet+	**29.20**	**29.06**	**28.85**	**28.67**	**28.51**	**28.38**	**28.26**	**28.11**	**27.97**

Table 2: Results of integer upscale SR. Best and second best results are **highlighted** and underlined. B100, Urban and Manga represent dataset B100, Urban100, and Manga109 respectively.

Method	Scale = 2			Scale = 3			Scale = 4		
	B100	Urban	Manga	B100	Urban	Manga	B100	Urban	Manga
Bicubic	29.56	26.88	30.80	27.21	24.46	26.95	25.96	23.14	24.89
SRCNN [12]	31.36	29.50	35.60	28.41	26.24	30.48	26.90	24.52	27.58
EDSR [29]	32.32	32.93	39.10	29.25	28.80	34.17	27.71	26.64	31.02
RCAN [56]	32.41	33.34	39.44	29.32	29.09	34.44	27.77	26.82	31.22
SAN [10]	32.42	33.10	39.32	29.33	28.93	34.30	27.78	26.79	31.18
Meta-SR [18]	32.35	-	39.18	29.30	-	34.14	27.75	-	31.03
FuncNet	32.48	33.61	39.73	29.39	29.42	34.90	27.87	27.15	31.71
FuncNet+	**32.51**	**33.78**	**39.87**	**29.43**	**29.57**	**35.10**	**27.90**	**27.29**	**31.97**

Table 3: Results of image denoising. Best and second best results are **highlighted** and underlined. CBSD, Kodak and Mac represent dataset CBSD68, Kodak24 and McMaster respectively.

Method	σ = 15			σ = 35			σ = 75		
	CBSD	Kodak	Mac	CBSD	Kodak	Mac	CBSD	Kodak	Mac
CBM3D [9]	33.52	34.28	34.06	28.89	29.90	29.92	25.74	26.82	26.79
DnCNN [52]	33.89	34.48	33.44	29.58	30.46	30.14	24.47	25.04	25.10
FFDNet [54]	33.87	34.63	34.66	29.58	30.57	30.81	26.24	27.27	27.33
FuncNet	34.26	35.21	35.39	30.02	31.24	31.61	26.72	27.98	28.18
FuncNet+	**34.28**	**35.25**	**35.44**	**30.05**	**31.29**	**31.67**	**26.76**	**28.05**	**28.26**

Table 4: Results of JPEG deblocking. Best and second best results are **highlighted** and underlined. LIVE and BSDS represent dataset LIVE1 and BSDS500 respectively.

Method	Quality = 10		Quality = 20		Quality = 30		Quality = 40	
	LIVE	BSDS	LIVE	BSDS	LIVE	BSDS	LIVE	BSDS
ARCNN [11]	29.13	29.10	31.40	31.28	32.69	32.64	33.63	33.55
DMCNN [55]	29.73	29.67	32.09	31.98	-	-	-	-
QGAC [13]	29.53	29.54	31.86	31.79	33.23	33.12	-	-
FuncNet	29.77	29.68	32.20	32.05	33.63	33.44	34.63	34.41
FuncNet+	**29.81**	**29.71**	**32.23**	**32.07**	**33.66**	**33.47**	**34.66**	**34.44**

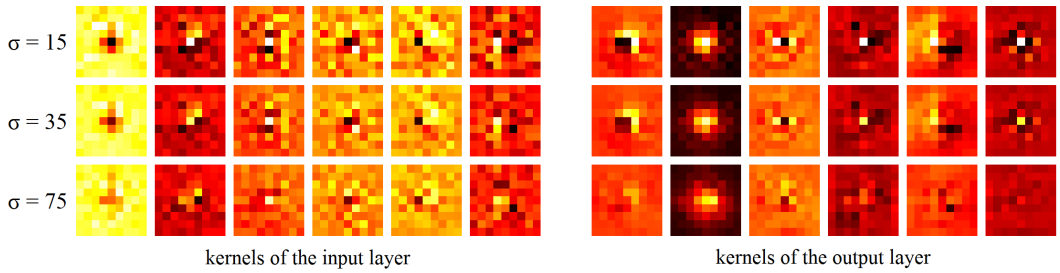

Figure 5: Kernel visualization of the denoising FuncNet. The left part is sampled from the input layer and the right part is sampled from the output layer.

Table 5: Results of the ablation study. Super-resolution, denoising and deblocking are tested on Urban100, Kodak24 and LIVE1 respectively.

Method	Super-resolution			Denoising		Deblocking	
	$s = 2$	$s = 3$	$s = 4$	$\sigma = 15$	$\sigma = 35$	$q = 10$	$q = 20$
FuncNet	33.61	29.42	27.15	35.21	31.24	29.77	32.20
Plain net	33.07	28.93	26.70	34.83	30.89	29.58	31.94
FuncNet ($H(x) = x$)	33.48	29.33	27.05	35.21	31.24	29.64	32.16
FuncNet (H is a MLP)	33.60	29.38	27.02	35.19	31.20	29.69	32.17

first and the last layer are also easier to understand. The results are shown in Figure 5. We can find out that the FuncNet use more radical kernel for feature when the noise level is low. By doing so, the FuncNet can get more information. And the FuncNet use more moderate kernel when the noise level is high, so the FuncNet can get less error.

4.4 Ablation Study

As we discussed earlier, using functional kernels instead of numerical kernels is the key to making networks perform better for parametric image restoration problems. To verify the effectiveness of our FuncNet models, we train plain counterparts of our FuncNet models, and compare their evaluation results with FuncNets. And to measure the impact of choice on the problem-related function $H(x)$, we train another two versions of FuncNet. The first one always uses the simplest non-trivial mapping $H(x) = x$, and the second one uses a small multilayer perceptron (MLP) with a hidden layer as a universal function approximator for any possible $H(x)$. We then also compare their evaluation results with FuncNet which uses $H(x)$ with a physical interpretation related to the problem. All the networks for ablation study share same architectures with their corresponding FuncNet models, and all training and evaluation settings remain unchanged. The evaluation results are shown in Table 5.

This ablation study shows that the adaptability of our FuncNet model is important for parametric image restoration problems. Once our FuncNet degenerates into a plain network, its adaptability to different parameter levels disappears, and its performance drops remarkably. The results also prove that both identity function and MLP are acceptable choices for $H(x)$. We can simply use those functions for a problem which is hard to design a $H(x)$ with a physical interpretation.

5 Conclusions

We propose a novel neural network called FuncNet to solve parametric image restoration problems with a single model. To transform a plain neural network into a FuncNet, all trainable variables in the plain network are replaced by functions of the parameter of the problem. Our FuncNet has both high storage efficiency and high computational efficiency, and the experimental results show the superiority of our FuncNet on three common parametric image restoration tasks over the state of the arts.

References

[1] Saeed Anwar and Nick Barnes. Real image denoising with feature attention. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3155–3164, 2019.

[2] Kanwal K Bhatia, Anthony N Price, Wenzhe Shi, Jo V Hajnal, and Daniel Rueckert. Super-resolution reconstruction of cardiac mri using coupled dictionary learning. In *2014 IEEE 11th International Symposium on Biomedical Imaging (ISBI)*, pages 947–950. IEEE, 2014.

[3] José M Bioucas-Dias and Mário AT Figueiredo. A new twist: Two-step iterative shrinkage/thresholding algorithms for image restoration. *IEEE Transactions on Image processing*, 16(12):2992–3004, 2007.

[4] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65. IEEE, 2005.

[5] Harold C Burger, Christian J Schuler, and Stefan Harmeling. Image denoising: Can plain neural networks compete with bm3d? In *2012 IEEE conference on computer vision and pattern recognition*, pages 2392–2399. IEEE, 2012.

[6] Lukas Cavigelli, Pascal Hager, and Luca Benini. Cas-cnn: A deep convolutional neural network for image compression artifact suppression. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 752–759. IEEE, 2017.

[7] Huibin Chang, Michael K Ng, and Tieyong Zeng. Reducing artifacts in jpeg decompression via a learned dictionary. *IEEE transactions on signal processing*, 62(3):718–728, 2013.

[8] Rémi Cogranne. Determining jpeg image standard quality factor from the quantization tables. *arXiv preprint arXiv:1802.00992*, 2018.

[9] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on image processing*, 16(8):2080–2095, 2007.

[10] Tao Dai, Jianrui Cai, Yongbing Zhang, Shu-Tao Xia, and Lei Zhang. Second-order attention network for single image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 11065–11074, 2019.

[11] Chao Dong, Yubin Deng, Chen Change Loy, and Xiaoou Tang. Compression artifacts reduction by a deep convolutional network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 576–584, 2015.

[12] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *European conference on computer vision*, pages 184–199. Springer, 2014.

[13] Max Ehrlich, Ser-Nam Lim, Larry Davis, and Abhinav Shrivastava. Quantization guided jpeg artifact correction. *arXiv*, pages arXiv–2004, 2020.

[14] William T Freeman, Egon C Pasztor, and Owen T Carmichael. Learning low-level vision. *International journal of computer vision*, 40(1):25–47, 2000.

[15] Leonardo Galteri, Lorenzo Seidenari, Marco Bertini, and Alberto Del Bimbo. Deep generative adversarial compression artifact removal. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4826–4835, 2017.

[16] Jun Guo and Hongyang Chao. Building dual-domain representations for compression artifacts reduction. In *European Conference on Computer Vision*, pages 628–644. Springer, 2016.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[18] Xuecai Hu, Haoyuan Mu, Xiangyu Zhang, Zilei Wang, Tieniu Tan, and Jian Sun. Meta-sr: A magnification-arbitrary network for super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1575–1584, 2019.

[19] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[20] Viren Jain and Sebastian Seung. Natural image denoising with convolutional networks. *Advances in neural information processing systems*, 21:769–776, 2008.

[21] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. In *Advances in neural information processing systems*, pages 667–675, 2016.

[22] Di Kang, Debarun Dhar, and Antoni B Chan. Crowd counting by adapting convolutional neural networks with side information. *arXiv preprint arXiv:1611.06748*, 2016.

[23] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1646–1654, 2016.

[24] Yoonsik Kim, Jae Woong Soh, and Nam Ik Cho. Agarnet: adaptively gated jpeg compression artifacts removal network for a wide range quality factor. *IEEE Access*, 8:20160–20170, 2020.

[25] Benjamin Klein, Lior Wolf, and Yehuda Afek. A dynamic convolutional layer for short range weather prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4840–4848, 2015.

[26] Younghee Kwon, Kwang In Kim, James Tompkin, Jin Hyung Kim, and Christian Theobalt. Efficient learning of image super-resolution and compression artifact removal with semi-local gaussian processes. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1792–1805, 2015.

[27] Stamatios Lefkimmiatis. Non-local color image denoising with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3587–3596, 2017.

[28] Shutao Li, Haitao Yin, and Leyuan Fang. Group-sparse representation with dictionary learning for medical image denoising and fusion. *IEEE Transactions on biomedical engineering*, 59(12):3450–3459, 2012.

[29] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 136–144, 2017.

[30] Pengju Liu, Hongzhi Zhang, Kai Zhang, Liang Lin, and Wangmeng Zuo. Multi-level wavelet-cnn for image restoration. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 773–782, 2018.

[31] Xianming Liu, Xiaolin Wu, Jiantao Zhou, and Debin Zhao. Data-driven sparsity-based restoration of jpeg-compressed images in dual transform-pixel domain. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5171–5178, 2015.

[32] Xinhao Liu, Masayuki Tanaka, and Masatoshi Okutomi. Single-image noise level estimation for blind denoising. *IEEE transactions on image processing*, 22(12):5226–5237, 2013.

[33] Fangzhou Luo and Xiaolin Wu. Maximum a posteriori on a submanifold: a general image restoration method with gan. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2020.

[34] Peyman Milanfar. A tour of modern image filtering: New insights and methods, both practical and theoretical. *IEEE signal processing magazine*, 30(1):106–128, 2012.

[35] Ben Niu, Weilei Wen, Wenqi Ren, Xiangde Zhang, Lianping Yang, Shuzhen Wang, Kaihao Zhang, Xiaochun Cao, and Haifeng Shen. Single image super-resolution via a holistic attention network. In *European Conference on Computer Vision*, pages 191–207. Springer, 2020.

[36] William B Pennebaker and Joan L Mitchell. *JPEG: Still image data compression standard*. Springer Science & Business Media, 1992.

[37] Stanislav Pyatykh, Jürgen Hesser, and Lei Zheng. Image noise level estimation by principal component analysis. *IEEE transactions on image processing*, 22(2):687–699, 2012.

[38] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[39] 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. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016.

[40] Vivek Soni, Ashish Kumar Bhandari, Anil Kumar, and Girish Kumar Singh. Improved sub-band adaptive thresholding function for denoising of satellite image based on evolutionary algorithms. *IET Signal Processing*, 7(8):720–730, 2013.

[41] Ying Tai, Jian Yang, and Xiaoming Liu. Image super-resolution via deep recursive residual network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3147–3155, 2017.

[42] Ying Tai, Jian Yang, Xiaoming Liu, and Chunyan Xu. Memnet: A persistent memory network for image restoration. In *Proceedings of the IEEE international conference on computer vision*, pages 4539–4547, 2017.

[43] Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, and Lei Zhang. Ntire 2017 challenge on single image super-resolution: Methods and results. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 114–125, 2017.

[44] Tong Tong, Gen Li, Xiejie Liu, and Qinquan Gao. Image super-resolution using dense skip connections. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4799–4807, 2017.

[45] Zhangyang Wang, Ding Liu, Shiyu Chang, Qing Ling, Yingzhen Yang, and Thomas S Huang. D3: Deep dual-domain based fast restoration of jpeg-compressed images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2764–2772, 2016.

[46] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[47] Raymond A Yeh, Chen Chen, Teck Yian Lim, Alexander G Schwing, Mark Hasegawa-Johnson, and Minh N Do. Semantic image inpainting with deep generative models. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5485–5493, 2017.

[48] Deniz Yıldırım and Oğuz Güngör. A novel image fusion method using ikonos satellite images. *Journal of Geodesy and Geoinformation*, 1(1):75–83, 2012.

[49] Changhoon Yim and Alan Conrad Bovik. Quality assessment of deblocked images. *IEEE Transactions on Image Processing*, 20(1):88–98, 2010.

[50] Songhyun Yu, Bumjun Park, and Jechang Jeong. Deep iterative down-up cnn for image denoising. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[51] Kai Zhang, Luc Van Gool, and Radu Timofte. Deep unfolding network for image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3217–3226, 2020.

[52] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.

[53] Kai Zhang, Wangmeng Zuo, Shuhang Gu, and Lei Zhang. Learning deep cnn denoiser prior for image restoration. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3929–3938, 2017.

[54] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Ffdnet: Toward a fast and flexible solution for cnn-based image denoising. *IEEE Transactions on Image Processing*, 27(9):4608–4622, 2018.

[55] Xiaoshuai Zhang, Wenhan Yang, Yueyu Hu, and Jiaying Liu. Dmcnn: Dual-domain multi-scale convolutional neural network for compression artifacts removal. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 390–394. IEEE, 2018.

[56] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 286–301, 2018.

[57] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2472–2481, 2018.

[58] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss functions for neural networks for image processing. *arXiv preprint arXiv:1511.08861*, 2015.

[59] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[60] Wilman WW Zou and Pong C Yuen. Very low resolution face recognition problem. *IEEE Transactions on image processing*, 21(1):327–340, 2011.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]