An Open Publishing Platform of the Organization for Human Brain Mapping
APERTURE NEURO

ORIGINAL RESEARCH ARTICLE

Improving the Interpretability of fMRI Decoding Using Deep Neural Networks and Adversarial Robustness

Patrick McClure[a,c], Dustin Moraczewski[b], Ka Chun Lam[a], Adam Thomas[b], and Francisco Pereira[a]

[a] *Machine Learning Team, Functional Magnetic Resonance Imaging Facility, National Institute of Mental Health, Bethesda, MD 20892, USA*
[b] *Data Science and Sharing Team, Functional Magnetic Resonance Imaging Facility, National Institute of Mental Health, Bethesda, MD 20892, USA*
[c] *Computer Science Department, Naval Postgraduate School, Monterey, CA 93943, USA*

ABSTRACT

Deep neural networks (DNNs) are being increasingly used to make predictions from functional magnetic resonance imaging (fMRI) data. However, they are widely seen as uninterpretable "black boxes," as it can be difficult to discover what input information is used by the DNN in the process, something important in both cognitive neuroscience and clinical applications. A saliency map is a common approach for producing interpretable visualizations of the relative importance of input features for a prediction. However, methods for creating maps often fail due to DNNs being sensitive to input noise or by focusing too much on the input and too little on the model. It is also challenging to evaluate how well saliency maps correspond to the truly relevant input information, as ground truth is not always available. In this paper, we review a variety of methods for producing gradient-based saliency maps and present an adversarial training method we developed to make DNNs robust to input noise, with the goal of improving the quality of DNN saliency maps. We introduce two quantitative evaluation procedures for saliency map methods in fMRI, applicable whenever a DNN or linear model is being trained to decode some information from imaging data. We evaluate the procedures using synthetic data sets, where the complex activation structure is known, and on fMRI data from the Human Connectome Project using saliency maps produced for linear models and DNNs doing task decoding in both settings. Our key finding is that saliency maps produced with different methods vary widely in quality in both synthetic and Human Connectome Project fMRI data, even when those methods have similar prediction performance. We found that training both linear and non-linear models using adversarial noise increased the quality of their saliency maps. We also found that, while some linear models generate good saliency maps in the highly controlled, synthetic data, non-linear DNN methods generate better saliency maps in real-world fMRI data. Finally, our experiments give evidence that combining adversarial training with a complex, non-linear model can improve saliency map quality when compared to several methods commonly used in the literature.

Keywords: Deep neural network, fMRI, Decoding, Interpretability, Adversarial robustness
Corresponding author: Patrick McClure (patrick.mcclure@nps.edu)
Date Received: Dec 17, 2020
Date Accepted: July 5, 2022

INTRODUCTION

Machine learning prediction models are becoming increasingly used in cognitive and computational neuroscience for a variety of classification and regression tasks, including diagnosing diseases and predicting participant traits from functional or structural magnetic resonance imaging (fMRI, sMRI) data (1, 2). In these applications, however, accurate predictions on new participants are not enough. It is also necessary to have interpretability, i.e. knowing what information in the input space (e.g. activation, connectivity between regions of interest (ROIs) or networks, measures derived from brain structure) is indicative of the target variable being predicted, from the architecture and parameters of the prediction model. A question related to interpretability is whether the information used by the prediction model is all the information present in the data. This is sometimes confounded with interpretability in the informal sense above, as it depends both on aspects of the model (e.g. regularization or network structure) and what in the data differs between classes. In neuroimaging, all of these are

This is an open access article distributed under the terms of the Creative Commons Attribution 4.0 International License (CC BY 4.0), which permits authors to copy and redistribute the material in any medium or format, remix, transform and build upon material, for any purpose, even commercially.

OPEN ACCESS

crucial to developing a deeper understanding of the relationship between brain function, how it manifests in the data, and how it is related to a target variable of interest. This also matters for translational applications, such as the use of brain stimulation to modulate brain function.

The recent availability of very large imaging data sets is starting to allow the development of deep neural network (DNN) models that can have better prediction performance than classical machine learning techniques in neuroimaging applications (3, 4). However, their adoption has been slowed, in part, by the perception that such models are inscrutable or, conversely, widely used linear models are more interpretable. For the specific application of *task decoding* – predicting which cognitive state a participant is in – from fMRI activation data, the most commonly used models are linear. To interpret these prediction models, the model weights are generally projected into input space (e.g. for a multinomial regression on activation data, the map for each state label would have the same shape as the inputs, with one weight per input dimension, associated with the location of that input dimension) (5, 6).

There are two main drawbacks to using linear model weights as maps. First, this does not take into consideration that the model compares how similar a test example is to all the class weights, and the prediction is the class with the highest similarity. For instance, if two classes placed high weight on one voxel, it is less informative than if only one class did. This cannot be determined by looking at a single weight map. Second, the weight maps are the *same* for every test example, e.g. informative locations with activation differences across participants will likely have smaller magnitude weights than informative locations with consistent activation across participants. For a particular participant, activation in one location might have played more of a role in prediction than for others, but the weight map will not reflect that. This is not a drawback if we are looking for population effects, common to all participants; it will be when individually analyzing participants.

Regardless of the interpretability method used, a further limitation of using linear models is that, by definition, they can only capture linear relationships between input features and what is being predicted. For activation inputs, for instance, the model can weigh the presence of activation in each voxel as positive or negative evidence for a particular class; the weight does not reflect the activation of other voxels in examples of that class. For this, one would have to use a non-linear prediction model, such as a DNN.

The weight map approach does not work as an interpretability method for DNNs, however, given that they perform multiple layers of computation, with non-linear operations. There have been many attempts to open up the DNN "black box" (7), across different areas of application. One of the most popular approaches is creating *gradient-based saliency maps*, visualizations of the input space highlighting the features that drive the prediction of each class. While superficially similar to weight maps, these gradient-based saliency maps differ by being functions of a prediction model *and* the specific image provided as input. They show, for a given test participant, how changes in activation lead to changes in prediction. The non-linearities in a DNN make it so that some input features can change how those, and other, input features affect the prediction, in a participant-specific manner. This is in contrast with linear models, where the input features are used in the same way for every participant.

There are several different methods for generating gradient-based saliency maps, which we review and contrast in the Methods to Compute Gradient-based Saliency Maps section. Both weight maps and gradient-based saliency maps – which we will jointly describe as saliency maps – have been used in the neuroimaging literature, as we discuss in the Computing Saliency Maps section. But how should someone analyzing neuroimaging data decide on what type of saliency map to use? Intuitively, a saliency map is good if it tells us what information in the input features is indicative of the prediction target. For a given model, this is a function of the data, the specific prediction the model is trying to make (e.g. which tasks are being distinguished), and the way the model uses information in the data. In the context of a DNN for fMRI decoding, the saliency map for a particular task should show the locations where the presence of activation is associated with that *specific* task alone and the locations where activation is *shared* between that and some (but not all) of the other tasks. Additionally, the saliency map for a task should negatively indicate the locations where activation provides evidence of some other task.

A quantitative assessment of the quality of saliency maps – or a comparison of different methods for obtaining them – is more challenging, given the absence of ground truth. Different methods for generating saliency maps have been shown to have different failure modes, as we review in the Computing Saliency Maps section. In the neuroimaging literature, and specifically when using activation data, such assessments generally rely on a voxelwise comparison of the saliency map to reference images or activation databases (e.g. (8) compared to a general linear model (GLM) voxelwise activation image for the tasks and (9) compared to NeuroSynth relevance maps (10) for the different tasks). From that perspective, a saliency map is good if it identifies activation similar to activation present in other studies. Note that, in the case of activation maps, comparing to reference images does not provide an indication of where adding activation would decrease the probability of the correct class. Furthermore, this comparison ignores the fact that DNN gradients are input-specific (i.e. a function of each *individual* input image and the prediction task), instead of population-wide (i.e. a function of the prediction task alone) as is the case if using the weights from linear models. Finally, comparing to reference activation implicitly assumes that the activation present in them contains all of the indicative information.

 CC By 4.0: © McClure et al.

What should the criteria be for judging a saliency map, then? If applying a classification model to multi-class data (e.g. task decoding), researchers expect there to exist very informative class-specific activation (present for one class, and no others), somewhat informative activation (shared between some, but not all, of the classes), non-informative activation present in all classes, and activation due to noise. The model should identify and combine these appropriately in order to predict each class. Note that, as the number of classes increases, there might be less or no class-specific activation. Furthermore, activation that plays the same functional role in different participants (e.g. represents the same information or is active as a result of the same mental process) might also be in somewhat different locations, and only partially overlap across those participants (e.g. functionally defined ROIs would be an instance of this, as described in (11)). Taken together, this suggests that saliency maps produced for each class should assign weights that reflect the informativeness of activation. Activation that pushes the model toward predicting some of the classes – is informative for them – will be positive in their respective saliency maps and negative in the saliency maps for the other classes. Activation present in all participants should be more positive than that present in just some of them.

Under this scenario, and given ground truth, we propose two intuitive criteria for evaluating the quality of a class saliency map for a specific test participant input. The first is to compute how similar the map is to where and how informative the activation in each voxel is for that class. This reflects what activation changes would make that input more like one of a given class, in the light of other classes. The second is to compute, for any two classes, whether the saliency map with respect to a target class is similar to what it would take to transform the example into an example of that target class. This criterion is more useful for situations where one wants to understand what activation would need to change to switch a participant between classes (e.g. as a result of neurostimulation). This reflects the model's decision boundaries between classes, e.g. what activation would switch the model from predicting one class to another for this particular input.

In this paper, we propose these criteria for evaluating the quality of saliency map methods for models in fMRI decoding. We illustrate the criteria by introducing two synthetic data sets that mimic brain activation with known information structure (extremely informative class-specific activation, informative activation shared between some – but not all – classes, and non-informative activation present in all classes). One of the synthetic data sets has the same activation structure across participants, the other introduces participant-specific variability in the location of activation. We then review commonly used types of prediction models, in a general multi-class situation, and review methods for obtaining saliency maps, both weight and gradient based.

We also introduce an adversarial training method that is computationally cheap, focuses on participant-specific adversarial noise, and has not been used on neuroimaging data before. Using the synthetic data sets as known ground truth, we show that, under our quantitative criteria, saliency maps produced with different methods vary widely in quality, even if the prediction models they were generated from have comparable decoding performance. Finally, we also apply these saliency methods to linear and DNN models trained to perform fMRI task decoding using the Human Connectome Project (HCP) data set (12). HCP data allow us to estimate ground truth saliency maps for each participant, using a similar approach to that used for the synthetic data set with participant variability in activation location. This enables us to test the degree to which each method can learn to utilize participant-specific input information, in the context of all of the participants in the training set. In HCP data, decoding performance for a variety of prediction models is close to the performance ceiling. However, the quality of the saliency maps produced, as measured by our criteria, varies widely. Our experimental results indicate that making DNNs robust to small input noise, via adversarial training, can greatly improve the quality of saliency maps, without adversely affecting decoding performance. We provide code for reproducing our results in synthetic and real data sets,[1] although standalone implementations of each method and prediction model combination are beyond the scope of this paper.

METHODS

Quantitative Criteria for Evaluation of Interpretability

In this section, we propose two quantitative evaluation criteria for the interpretability of saliency maps for a classification model. We will use them to compare a variety of methods for producing saliency maps, which will be introduced in the Computing Saliency Maps section. In the Experiments section, results are presented for both synthetic and real data. Given that it will be easier to describe the evaluation criteria with reference to illustrative examples, we introduce the basic synthetic data set at this point. The synthetic data set comprises 2D images that are simplified analogs of brain activation maps. The "brain" in these maps can be in one of three classes: A, B, or C. Each class has a characteristic template of activation, made up of nine activation regions, shown in Figure 1a (top), together with the average of the three templates. Each class template (e.g. A) has an active region that is completely *specific* to it and appears in no other class. In addition, it has two other active regions

[1] https://github.com/patrick-mcclure/adversarial_training_for_fmri_decoding.

 CC By 4.0: © McClure et al.

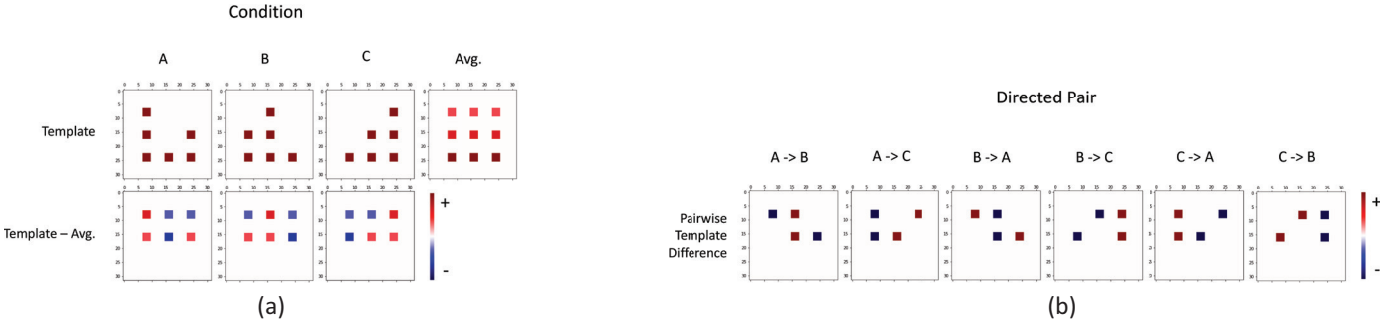

Fig. 1. Synthetic 2D fMRI ground truth for the two quantitative evaluations we introduce. a (top): The template images for activation locations in each of three conditions – A, B, C – as well as the average activation across conditions. **a (bottom):** The "template-average" images used as ground truth for class informative activation maps for each condition. **b:** The "pairwise template difference" images used as ground truth for pairwise class activation difference maps.

that are *shared* with one other class (e.g. AB and AC). Finally, it has three active regions that are present in all classes (e.g. ABC). In the first synthetic data set, the class template is the same for all the participants, with participant-specific noise. In the second synthetic data set, each region is shifted horizontally and vertically by some random amount, independently for each participant.

Our first evaluation criterion aims to answer the question, "What information in the input space allows a prediction model to make a correct prediction?" For activation maps, it is whatever activation departs from the average activation across all classes. Intuitively, this means that any asymmetry in the presence of activation for different classes provides information that the prediction model can exploit. We quantify this by computing the pixelwise correlation between a saliency map for the correct class of an image and the "template-average" image (i.e. the difference between the correct class template and the average of the three templates) for each test image. We will call this "template-average" image a *class informative activation (CIA) map*; it is shown in Figure 1a (bottom). The values in this map should also reflect the relative importance of class-specific versus shared activation, in terms of carrying information about the class. This map will be the ground truth for the gradient-based saliency maps produced for each class by the various methods we consider.

Our second evaluation criterion aims to answer the question, "What information in the input space corresponds to the important differences between any two classes?" The answer to this, given an image of a particular class (e.g. A), are the changes in activation that would be needed to transform it into an image of another class (e.g. B). We will call this the *pairwise class activation difference (CAD) map*, shown in Figure 2b. The saliency map for predicting class B, with input from class A, should approximate this map and can be calculated using the fact that a gradient-based saliency map depends on both the input image and the prediction target of interest. We quantify this by computing the pixelwise correlation between a saliency map for a class (e.g. B), given an image of another class (e.g. A), and the pairwise CAD map (e.g. B-A). This map will be the ground truth for the gradient-based saliency maps produced for each class transformation, by the various methods we consider.

Prediction Models

Notation For all models, an example $\mathbf{x} = [x_1, \ldots, x_V]$ is a vector of input features; in our experiments, these are activation levels across V voxels. The class label y corresponds to the experimental condition or task present as the example was acquired. Almost all the models we consider use a one-hot encoding of the label, i.e. it will be a vector $\mathbf{y}$ such that entry i is 1 if the condition $y = i$, and 0 otherwise.

Multinomial Regression In multinomial regression, each class i has a vector of weights for each input feature, resulting in a weight matrix W where row i contains the weight vector $\mathbf{w}_i$ for class i. These weights are multiplied by the input features, $\mathbf{x}$, to yield a vector $\mathbf{z} = W\mathbf{x}$, with one logit z_i per class i. A softmax function, $\sigma(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$, is then applied to the resulting vector, computing $\sigma(W\mathbf{x})$ The output of the softmax is a probability distribution over classes, $p_W(y \mid \mathbf{x})$, that is parameterized by W. When incorporating L2 regularization, training becomes maximum a posteriori estimation, $\text{argmax}_W\, p_W(y \mid \mathbf{x}) p(W)$, with a Gaussian weight prior, $p(W)$. We include this model because it is often deemed the easiest to interpret, given the mapping between each class and the input feature space.

Correlation Classifier A correlation classifier compares a test example $\mathbf{x}$ to class prototypes $\mathbf{c}_i$, by computing the voxelwise correlations between the example and each of them. It predicts the condition i whose prototype is most similar to the example. A class prototype is usually the average of the training set examples with that class label. We include this model because the CIA interpretability measure described earlier is based on similarity to class information images. This model can be seen as a special case of the multinomial regression model if we consider z-scored versions of the test example and the

 CC By 4.0: © McClure et al.

class prototypes. In that situation, the vector of weights for each class is simply the z-scored class prototype. Likewise, we compute the prediction by passing all class similarities through a softmax function to yield a probability for each class. This allows us to produce saliency maps using exactly the same approach as for multinomial regression.

Neural Networks In many cases, a multinomial regression model operating on input features cannot separate all of the classes. Neural networks (NNs) are models that can learn to compute new features that are non-linear functions of the original input features, yielding a new feature space where classes can be separated. NNs are built up of functions (i.e. "layers"). A layer, l, takes the form of multiplying the input to the layer, $\mathbf{x}_l$, by a weight matrix, W_l, and then applying an activation, f_l, resulting in new features $f_l(W_1\mathbf{x}_1)$. In NNs, a non-linear activation function is used, such as a rectified linear unit (ReLU) (13). Layers are applied sequentially, with the first layer operating on the input image, the second layer operating on the output of the first, and so on until the layer before the last. For multi-class classification, the final layer, often called "output layer," usually computes a softmax non-linearity, similar to multinomial regression, producing $p_W(y \mid \mathbf{x})$. DNNs are NNs that have more than one non-output layer.

Linear Networks Linear networks (LNs) are similar to NNs, except that the activation function is the identity function. In LNs, the weight matrices of the different layers can be multiplied to create a single linear layer that computes the same function as sequentially applying the different layers. For every neural network model in our experiments, we also build a linear network with the same connectivity structure, to examine how non-linear activation functions affect interpretability.

3D Convolutional Networks 3D convolutional networks are NNs that are built using 3D convolutional layers. A 3D convolutional layer takes a 3D volume, X, as an input. Each location in that 3D volume corresponds to a feature vector with c channels. The weights of the layer correspond to 3D filters of shape (d, h, w, l), where d is the number of channels of the input to the layer, h is the height of the filter, w is the width of the filter, and l is the length of the filter. When using a stride of one, as done in this paper, each filter is applied at each location throughout the input 3D volume. This results in outputting another 3D volume, with a feature vector containing the output of each filter at each possible location. Mathematically, convolving the n_{th} filter, W_n, can be written as

$$Y_n[a,b,c] = \sum_{t=1}^{d} \sum_{i=-\lfloor h/2 \rfloor}^{\lfloor h/2 \rfloor} \sum_{j=-\lfloor \omega/2 \rfloor}^{\lfloor \omega/2 \rfloor} \sum_{k=-\lfloor \ell/2 \rfloor}^{\lfloor \ell/2 \rfloor} X[t, a-i, b-j, c-k] W_n[t, i, +\lfloor h/2 \rfloor + 1, j + \lfloor \omega/2 \rfloor + 1, \lfloor \ell/2 \rfloor + 1]$$

The output for a single filter becomes a channel of the output of the convolutional layer. In a convolutional neural network (CNN), a non-linearity is applied to the output of the convolutional layer; this is not the case in a convolutional linear network (CLN).

Computing Saliency Maps

Gradient-Based Saliency Maps

As proposed in the introduction, one intuitive definition of the interpretability of a prediction model is the degree to which it is possible to understand the relationship between the input and output variables learned by that model. For linear regression, this is usually done by looking at the learned weights, as they directly reflect how much changing an input variable would change the output variable (i.e. the gradient of the output with respect to the input). However, linear regression weights are not always straightforward to interpret, as discussed in (14). This is even more the case in multinomial regression, given that the weights for one class affect the prediction through a softmax function that takes other classes into account.

In our classification setting, interpretability translates into being able to understand what, in the input space, allows the model to distinguish classes (i.e. how changing the input variables changes the probability of predicting each class). Formally, this is the gradient with respect to the *input variables*, $\nabla_{\mathbf{x}} p_W(y \mid \mathbf{x})$. This is different from the gradient with respect to the weights, the *learned parameters* of the model, $\nabla_W p_W(y \mid \mathbf{x})$. The parameter gradient is computed during the training of a DNN, in gradient-based optimization methods, to find parameters that make the training data more likely given the model. Note also that the gradients of the logits with respect to the input for DNNs are dependent on the inputs; this is in contrast to linear regression, which has the same gradients for the logits for *every* input. Note that different models can produce different probabilistic predictions, and therefore different gradient maps, and still have the same accuracy. This is because the accuracy depends only on what class is predicted to be most likely, but does not take into account exactly how likely that, or other, classes are according to a model's prediction.

Most Common Issues with Gradient-Based Saliency Maps

While gradient-based saliency maps have been successfully deployed in a variety of machine learning applications, they have two common failure modes:

1. maps are almost entirely a function of the specific input and are not sensitive to the learned parameters of the model;
2. maps dramatically change in the presence of even extremely small input noise.

 CC By 4.0: © McClure et al.

Both of these failure modes can be illustrated by reference to the situation where a DNN does task decoding based on activation maps, as follows.

The first problem occurs when methods produce saliency maps that mostly reflect the specific input (e.g. they are a function of activation being present or absent), rather than what the model learned (e.g. which presence of activation distinguishes some classes from others). This can be demonstrated by showing that saliency maps are similar for both a trained DNN and one where parameters were randomly initialized, which would indicate that the maps do not reflect the information the model uses to make a prediction. Adebayo et al. (15) show that this is the case for methods such as multiplying the gradient by the input, guided back-propagation, guided GradCAM, and integrated gradients, which all over-focus on the specific input. Note that other methods, such as Layer-wise Relevance Propagation (LRP) and DeepLift, which are used in SHapley Additive exPlanations (SHAP) (16), are equivalent to the gradient times input method, for ReLU networks with zero baseline and no biases (17). Due to this and the popularity of LRP and DeepLift, we will, in all experiments, use the gradient times input method as a representative technique for the methods that over-focus on the input when generating saliency maps.

The methods discussed earlier have been used to study DNNs trained on neuroimaging data, with (8) using guided back-propagation, (9)s using LRP, and (18) using standard gradients.

The second problem occurs when DNNs are susceptible to abrupt changes in prediction (e.g. which task) based on small input changes (e.g. tiny activation changes in a few voxels) (19, 20). This violates the intuition that inputs that are extremely similar should have similar predictions (i.e. the model should learn a smooth function). This will then perturb any estimate of how much a particular input feature is relevant for a prediction (21, 22). Robustness, in this context, means ensuring that small changes in inputs to the DNN will not dramatically change the output. As a result, making DNNs robust to small noise has been proposed as means of increasing interpretability; however, inducing too much smoothness can lead to a decrease in accuracy in some instances (23–25). The main approaches devised to address this rely on either smoothing the input gradients of non-robust DNNs (e.g. SmoothGrad (26)) or making DNNs robust to input noise, namely using random and adversarial input noise during training, which can improve the interpretability of DNNs (22, 24). To the best of our knowledge, adversarial training for improving interpretability has not been previously used with neuroimaging data.

Computing Saliency Maps for All the Prediction Models

In this section, we describe the basic approaches for deriving saliency maps for all the prediction models introduced earlier, both with and without using gradients. We then introduce other approaches that build on the use of gradients, including our own adversarial training one. All the approaches introduced will be compared in the synthetic and real-data experiments reported in the Experiments section.

Multinomial Regression Weights For multinomial regression, the simplest way to produce a saliency map for a class is to use the learned weights for each input feature for that class. We included this approach because it is very commonly used and subjectively deemed most interpretable.

Multinomial Regression, Correlation Classifer, LN, and NN Input Gradients A gradient-based saliency map can be generated for NNs, but also for LN (it is a NN with an identity activation function), multinomial regression (it is a NN without a hidden layer), and correlation classifier (it is a special case of multinomial regression).

NN Gradient Times Input One popular method to create saliency maps for an input is to multiply that input with the input gradient for that input. Several commonly used saliency map methods, such as LRP and DeepLift, which are used in SHAP (16), are equivalent to the gradient times input method, for ReLU networks with zero baseline and no biases (17). As a result, we include saliency maps produced by multiplying the NN gradient-based saliency map by the NN input in all of our experiments.

SmoothGrad Another popular method for dealing with the lack of robustness in NN gradients is SmoothGrad (26), which estimates the gradient $\nabla_{\mathbf{x}} pW(y \mid \mathbf{x})$ around an input $\mathbf{x}$ as $E_{\delta}[\nabla_{\mathbf{x}} pW(y \mid \mathbf{x}+\delta)]$, where $p(\delta) = N\left(\delta; 0, I\sigma_{\delta}^{2}\right)$. In practice, Monte-Carlo sampling is used to estimate this expectation by adding random noise δ – sampled from $p(\delta)$ – to an input, calculating the input gradient for each of these noisy inputs and then averaging over the gradients produced for these different noisy samples. Several papers have reported that this improves the interpretability of NN gradients (15, 26), although there are no formal guarantees on this leading to robust gradients.

Random Noise and Adversarial Training A common way of increasing model robustness to input noise is to add noise to inputs during the training process so that the NN encounters different variations of a given training example, all of which must lead to the same prediction. This makes training produce models that correspond to locally smooth functions. The input noise during training is often random noise such that the produced training example variations are centered at the original training examples, such as zero-mean Gaussian noise or zero-centered spherical noise. We include models trained with random noise in all experiments.

It is possible to train with noise that more effectively increases a NN's robustness by using *adversarial training*,

 CC By 4.0: © McClure et al.

where the noise is determined by calculating the input change that would alter the output the most, rather than randomly. In the original adversarial training papers, the adversarial noise was computed using a gradient step for the input gradient at a real input example, using the fast gradient sign method (19, 20). Recently, projected gradient descent (PGD) has been used to improve adversarial training by performing multiple gradient steps when generating adversarial noise (27). For PGD, the goal is to find the adversarial noise that will either optimally decrease the probability of the correct class of a training input (a.k.a. non-targeted adversarial noise) or optimally increase the probability of an incorrect class (a.k.a. targeted adversarial noise) while staying close to the input example. A common definition of staying close to the input example is restricting the adversarial noise, δ, to have a small L2-norm (i.e. keeping the Euclidean distance between the input example and the adversarial example small). Generating a non-targeted, L2 adversarial example using PGD is achieved by optimizing

$$\text{argmin}_{\delta_{(\mathbf{x},y)}} \; p_W\left(y \mid \mathbf{x} + \delta_{(\mathbf{x},y)}\right) \text{ s.t. } \left\|\delta_{(\mathbf{x},y)}\right\|_2 \leq \in .$$

Adversarial noise can be found by performing this optimization and then adding it to the respective input in order to generate an adversarial example. These adversarial examples can then be used to calculate the parameter gradients for one parameter update step. For relatively small models, such as our synthetic data experiment, this can work very well. However, this is computationally expensive and can be prohibitive if using large DNNs. Recently, Shafahi et al. (25) proposed simultaneous generation of adversarial examples, by optimizing the adversarial noise and updating the model parameters, thus significantly speeding up adversarial training. We developed a version of this method that uses example-specific L2 adversarial noise, instead of universal L_∞ noise (see Algorithm 1). L2 noise was chosen because it has been suggested to have improved interpretability in comparison to L_∞ noise (23) and has less of an accuracy-interpretability tradeoff than L_∞ noise (24).

To evaluate whether the non-linearities used in NNs help to generate better saliency maps, we also trained LNs with adversarial training. We include both NN and LN models trained with our adversarial training method in all experiments.

Algorithm 1 m-step Minibatch Adversarial Training

Require: Set of training minibatches D, Model parameters W, noise bound E, learning rate τ, hop steps m

while not converged **do**
  for minibatch $B \in$ D **do**
    for (x, y) $\in B$ **do**
      $\delta_{(\mathbf{x},y)} \leftarrow \mathbf{0}$ ▷ Initialize adversarial noise
    for i = 1, ..., m **do**
      $g_W \leftarrow \mathbb{E}_{(\mathbf{x},y)\in B}\left[\nabla_W \log p_W\left(y \mid \mathbf{x} + \delta_{(\mathbf{x},y)}\right)\right]$ ▷ Get parameter gradients
      $W \leftarrow W + \tau g_W$ ▷ Update parameters
    for (x, y) $\in B$ **do**
      $g_{adv} \leftarrow \nabla_{\mathbf{x}} p_W\left(y \mid \mathbf{x} + \delta_{(\mathbf{x},y)}\right)$ ▷ Get input gradients
      if $\left\|g_{adv}\right\|_2 > 0$ **then**
        $\upsilon \sim U(0, \varepsilon)$
          ▷ Sample adversarial step size
        $\delta_{(\mathbf{x},y)} \leftarrow \delta_{(\mathbf{x},y)} - \upsilon g_{adv} / \left\|g_{adv}\right\|_2$
          ▷ Update adversarial noise
      if $\left\|\delta_{(\mathbf{x},y)}\right\|_2 > \in$ **then**
        $\delta_{(\mathbf{x},y)} \leftarrow \in \delta_{(\mathbf{x},y)} / \left\|\delta_{(\mathbf{x},y)}\right\|_2$
          ▷ Rescale the adversarial noise

Improving Input Gradients by Decreasing Overconfidence Unlike the derivative of a logit, the derivative of the probability, $\nabla_{\mathbf{x}} p_w(y \mid \mathbf{x})$, is dependent on the confidence of the network. This can be an issue if a model is overconfident, as predicted probabilities very close to 0 or 1 will make the derivatives be very close to 0. To prevent this, we perform temperature scaling for the evaluated models, as suggested by (28). In temperature scaling, a temperature parameter, t, is introduced into the softmax function in all of our models, $\sigma(z)_i = \frac{\exp(z_i / t)}{\Sigma_j \exp(z_j / t)}$. The temperature parameter can then be set by taking an already trained model, fixing its weights, and finding the t that results in the lowest negative log-likelihood on the validation data. This will result in a model whose predicted probabilities better match the data, as the negative log-likelihood penalizes being confident on incorrect predictions.

EXPERIMENTS

Synthetic Data

Synthetic Data without Participant-Specific Activation Locations

Data As described in the Quantitative Evaluation of Interpretability section, the synthetic data set contains 2D images that are simplified analogs of brain activation maps in one of three classes: A, B, or C. Each class activation template (top row of Figure 2a) contains 32 × 32 = 1024 pixels. Each activation region is a 3 × 3 pixel square, so ~7% of the "brain" contains activation. Pixels belonging to an activation region have a value of 1, and background pixels have a value of 0. We generated 1000 examples of each class for train, validation, and test sets, respectively, analogous to 1000 participants in a real data set. Each example was produced by taking the template for the corresponding class and corrupting it with pixel-wise independent Gaussian noise (σ = 0.5).

Prediction Models and Results We trained an L2-penalized multinomial linear regression, an L2-regularized NN with a 20-unit fully connected layer with

 CC By 4.0: © McClure et al.

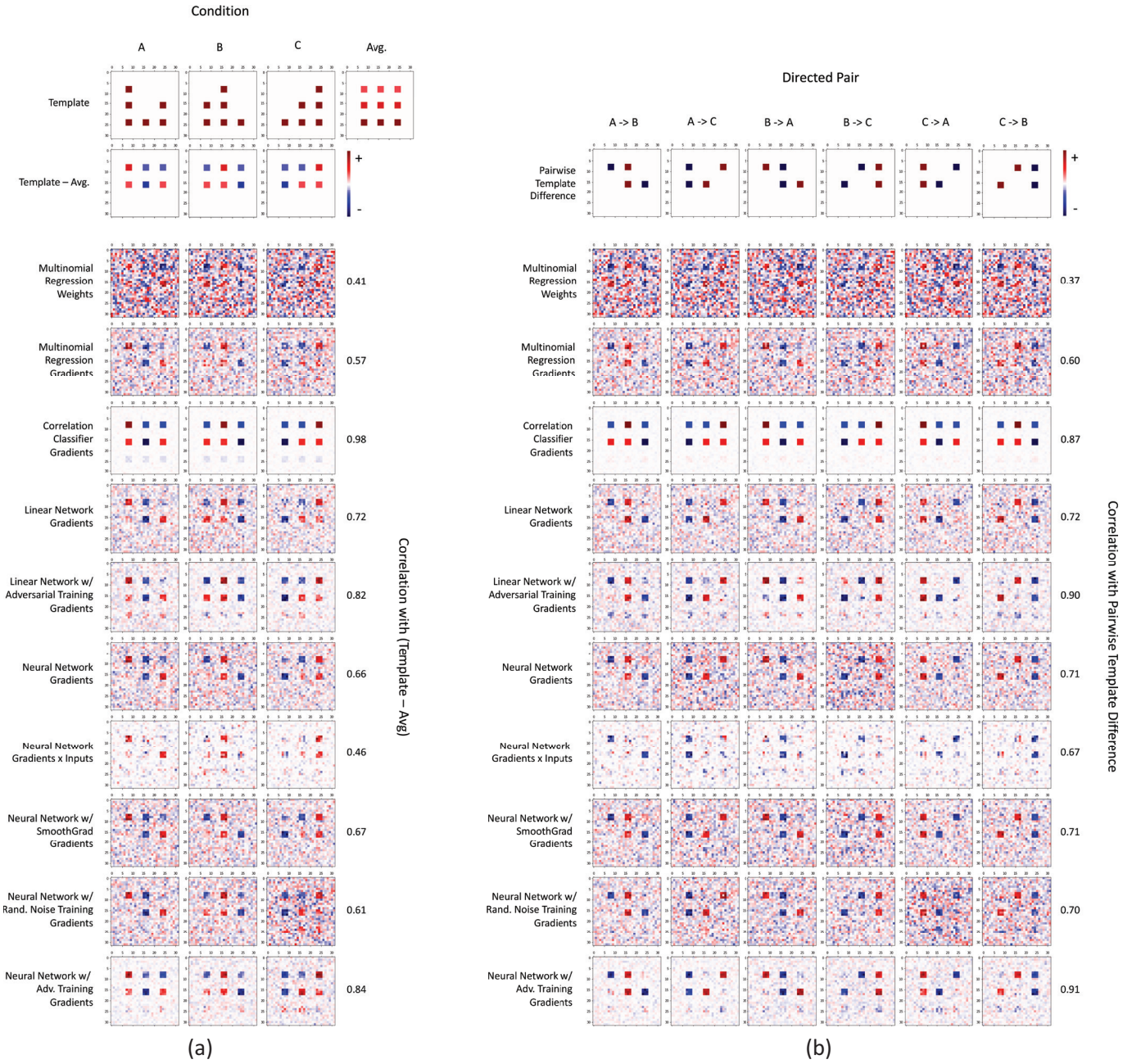

Fig. 2. **Visualization of saliency maps for the predictions for the synthetic test data without participant-specific ROI locations, for the two quantitative evaluations we introduce. a: (CIA)** The saliency maps for the correct class and their Pearson's correlations with the corresponding "template-average" map. **b: (CAD)** The saliency maps for other classes and their correlations with the corresponding "pairwise template difference" map.

ReLU non-linearities and a 7-unit fully connected layer with a softmax non-linearity, and an L2-regularized LN with the same architecture as the NN, except without the ReLU non-linearities, to predict which of the three classes (i.e. A, B, or C) generated an image. Models were trained with full-batch Adam. An NN and an LN were trained with and without 3-step PGD adversarial training with $E = 1$. To test if adversarial noise was particularly helpful, we also trained an NN with additive uniform random noise within the same E-sphere as used in adversarial training. For SmoothGrad, $\sigma_\delta = 0.3$, and was set to maximize interpretability on the validation data. Each model was able to perfectly classify the test data.

Class Informative Activation (CIA) The first criterion introduced in the Quantitative Evaluation of Interpretability section evaluates how well a gradient-based saliency map for a class, generated from an input image with that class, identifies CIA. The ground truth for this is the CIA map (the difference between the class template and the average template across all classes), shown in row 2 of Figure 2a. Each saliency map method we consider produces, for each example, a map that depends on its class. These maps are shown in rows 3–12 of Figure 2a for the examples for a test synthetic participant. The evaluation measure for each test example map is its pixelwise correlation with the CIA map, yielding a sample

 CC By 4.0: © McClure et al.

of correlation value. We use a two-sided paired *t*-test across test examples to compare whether the average correlation for any two methods is different, with Bonferroni correction to account for all of the pairwise comparisons being carried out. The results of the paired *t*-tests are shown in Figure 3a. Overall, the correlation classifier performs best according to this measure, given that it compares the test participant image to the average image for each class across participants in the training set. As our synthesis process generates each of the latter by adding pixelwise, zero-mean gaussian noise, the average image cancels that noise and is, in essence, the class template. Otherwise, both linear and non-linear NNs with adversarial training are generally better than the other methods. There is little difference between them, given that there is no non-linear interaction between the appearance of activation spots across conditions; nor are there differences between participants.

Pairwise Class Activation Differences (CAD) The second criterion introduced in the Quantitative Evaluation of Interpretability section evaluates how well the saliency map for a class B, given an input from class A, identifies the changes in activation that would be needed to transform the input from class A to class B. The ground truth for this, in the synthetic data, is the pairwise CAD map, the difference between the template images for classes B and A. This is shown, for every possible transition, in the first row of Figure 2b.

We applied this approach to the saliency maps produced by all of the methods, as shown in rows 2–8 in Figure 2b. For each test example, each method produces a saliency map indicating what would be required to transform the example into another class. Each of those saliency maps is then correlated with the corresponding pairwise CAD map, yielding an average correlation value across possible classes; across test examples, this results in a sample of correlation values. For any two methods, we used a two-sided paired *t*-test to determine if their average correlation was different. We used Bonferroni correction to account for all of the pairwise comparisons. The results of the paired *t*-tests are shown in Figure 3b. As before, the neural network with adversarial training outperforms the other methods. We also found that the CAD maps for the correlation classifier, e.g. in going from class A to class B, were very similar to the CIA maps for the target class B. This suggests that the saliency map indicating how to transform an example does not reflect where that example is in the input space.

Synthetic Data with Participant-Specific Activation Locations

Data The second type of synthetic example was generated as described in the Quantitative Evaluation of Interpretability section, except that the class template for each condition was unique for each example (analogous to a participant in a real data set). For each of the nine ROIs, the location of its center was obtained by

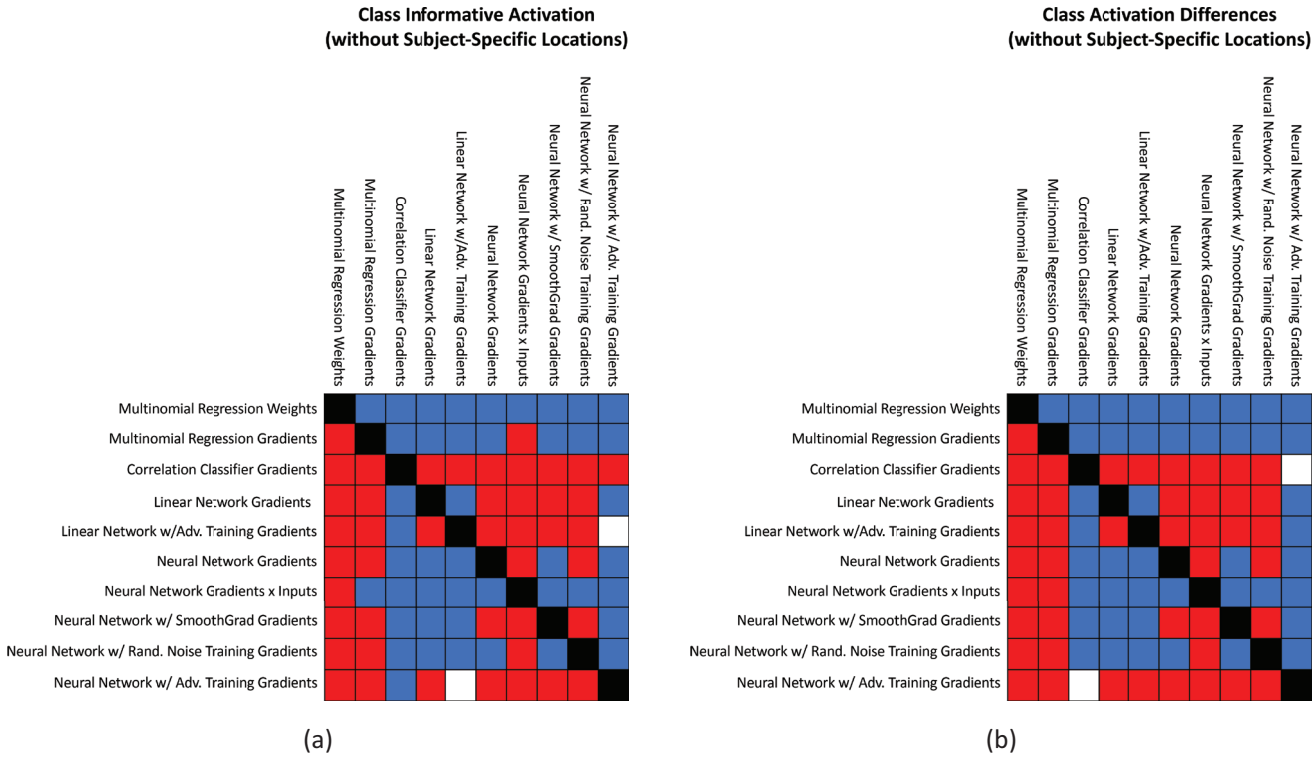

Fig. 3. The results of the pairwise paired *t*-tests for the CIA and CAD correlations, across test examples, for the synthetic data without participant-specific locations. Blue indicates that a method is significantly different ($p < 0.05$, corrected) from another method and it has a smaller mean correlation. White indicates that the two methods being compared do not have significantly different ($p < 0.05$, corrected) correlations. Red indicates that a method is significantly different ($p < 0.05$, corrected) from another method and it has a larger mean correlation. The comparisons should be read left to right rowwise.

 CC By 4.0: © McClure et al.

adding a random shift – −1, 0, or 1 – in vertical and horizontal directions, relative to the location in the previous scenario. The resulting activation centers were then used to create participant-specific templates for classes A, B, and C, and examples of each class by adding pixelwise independent Gaussian noise (σ = 0.5) as before. This gives us the activation patterns for each class, for a given synthetic participant, shown in the first row of Figure 4a. Note the displacements of the activation centers in each ROI. Having different activation patterns across synthetic participants makes it possible to see how variations in the location of certain informative activation, as present in neuroimaging data, manifest itself across the methods studied.

Prediction Models and Results We trained the same type of models on the second synthetic data set as we did on the first synthetic data set. Each model was able to perfectly classify the test data.

Class Informative Activation (CIA) In the presence of participant-specific ROI locations, we compute CIA using participant-specific ground truths, as shown in rows 1 and 2 of Figure 4a. Note that the participant-specific templates shown are the examples for that participant in the different classes; this is unlike the previous synthetic data set, where the templates were common to all participants. Each gradient-based saliency map method we consider produces, for each example, a map that depends on its

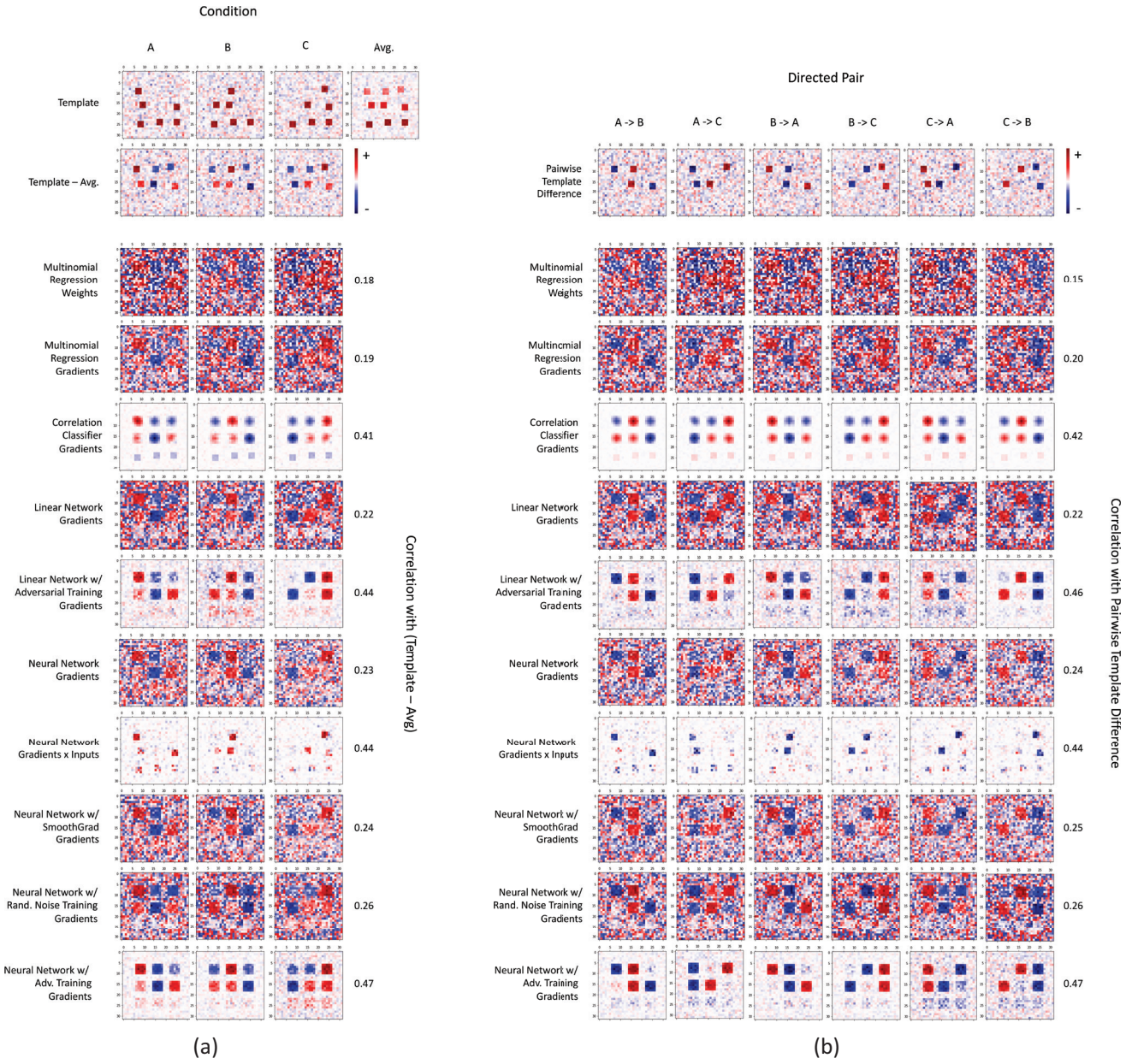

Fig. 4. Visualization of saliency maps for the predictions for the examples for a synthetic test participant with participant-specific ROI locations, for the two quantitative evaluations we introduce. **a: (CIA)** The saliency maps for the correct class and their Pearson's correlations with the corresponding "template-average" map. **b: (CAD)** The saliency maps for other classes and their correlations with the corresponding "pairwise template difference" map.

 CC By 4.0: © McClure et al.

class. These maps, for the examples for a test synthetic participant, are shown in rows 3–12 of Figure 4a. The evaluation measure for each test example map is its pixelwise correlation with the CIA map, yielding a sample of the correlation value. We use a two-sided paired *t*-test to compare whether the average correlation for any two methods is different, with Bonferroni correction to account for all of the pairwise comparisons being carried out. The results of the paired *t*-tests are shown in Figure 5a. Given that activation can now appear in a broader spatial context, relative to the previous synthetic data set, most methods display larger regions with positive and negative gradients. One exception is NN gradients times inputs, which is heavily driven by the input. In this instance, the LN and NN with adversarial training outperform the other methods. This indicates that adversarial training helps create models that identify all of the places where the appearance of activation would make a given participant image more likely to have each class. The LN slightly outperforms the NN, indicating that, for this setup, adding non-linearities to the LN did not improve the CIA measure. The correlation classifier now performs somewhat worse, mainly because there is less activation overlapping across all subjects in the average image for each class, taken across training set participants.

Pairwise Class Activation Differences (CAD) In the presence of participant-specific ROI locations, we compute CAD using participant-specific ground truths, as shown in rows 1 and 2 of Figure 4b. Note that the participant-specific templates are the examples for that participant for the different classes; this is unlike the previous synthetic data set, where the templates were common to all participants. We applied this approach to the saliency maps produced by all of the methods, as shown in rows 2–11 in Figure 4b. For each test example, each method produces a saliency map indicating what it would require to transform the example to a given class. Each of those saliency maps is then correlated with the corresponding pairwise CAD map for that participant (see row 1 of 4b), yielding an average correlation value across possible classes; across test examples, this results in a sample of correlation values. For any two methods, we used a two-sided paired *t*-test to determine if their average correlation was different. We used Bonferroni correction to account for all of the pairwise comparisons. The results of the paired *t*-tests are shown in Figure 5b. In this instance, the CNN with adversarial training outperforms the other methods. As in the prior data set, the input gradient maps for the correlation classifier are not sensitive to the class of the input example.

fMRI Data

Data For our main evaluation, we used volumetric task fMRI activation maps from the Human Connectome Project 1200 data set (HCP1200) distribution, for the 965 participants for which they were produced. We randomly

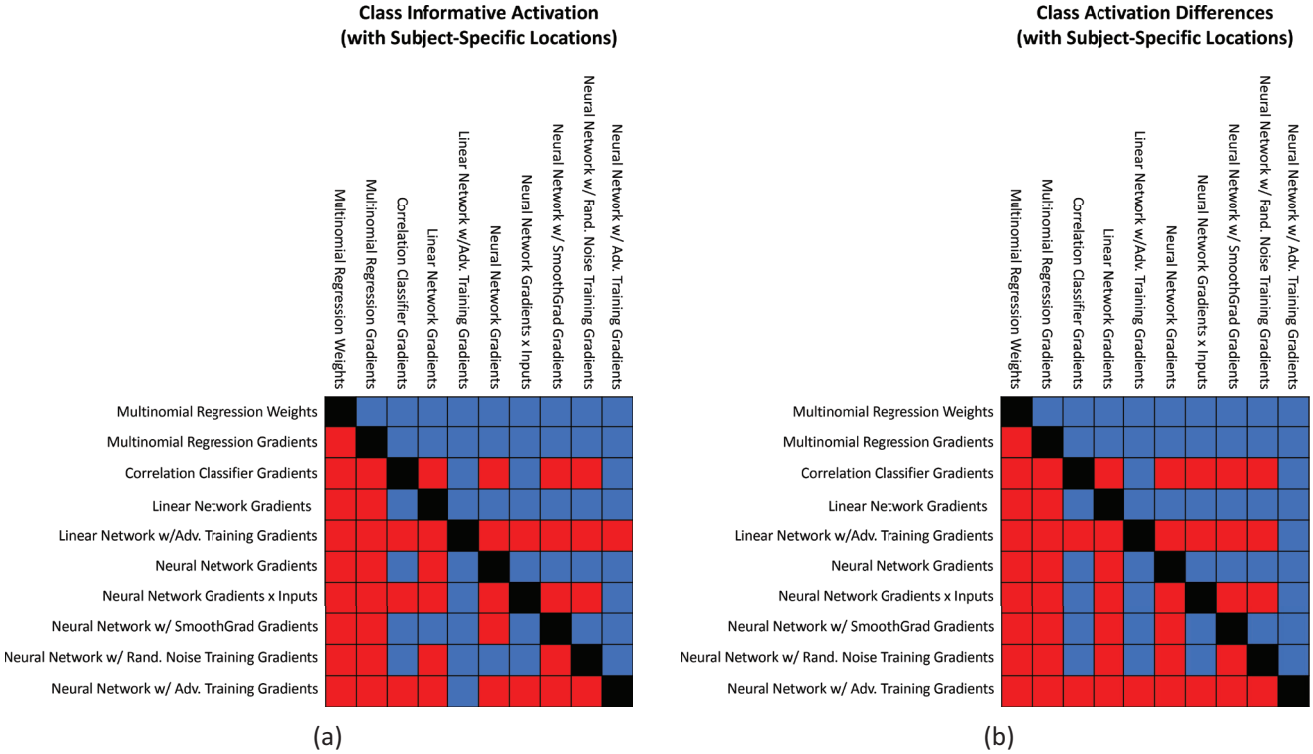

Fig. 5. The results of the pairwise paired t-tests for the class informative activation and class activation differences correlations, across test examples, for the synthetic data with participant-specific locations. Blue indicates that a method is significantly different ($p < 0.05$, corrected) from another method and it has a smaller mean correlation. White indicates that the two methods being compared do not have significantly different ($p < 0.05$, corrected) correlations. Red indicates that a method is significantly different ($p < 0.05$, corrected) from another method, and it has a larger mean correlation. The comparisons should be read left to right rowwise.

 CC By 4.0: © McClure et al.

split participants into groups of 788 (training), 92 (validation), and 95 (test), taking care to keep siblings within the same set, and assigning the largest families to the training set. For each participant, HCP1200 contains activation maps for several conditions within each of seven tasks (described in (29)): working memory (WM), gambling, motor, language, social, relational, and emotion. In our experiments, each task is one class, with the example being the activation map contrasting the main task condition with the control condition (see (29) and Table 1, exceptions were made for gambling, given that activation in the two conditions was extremely similar, and motor, as there was no clear control). The input **x** and output *y* pairs used in training/testing were created by individually z-scoring these activation maps and labeling them with their respective task.

Prediction Models and Results For classifying tasks from HCP 3D activation maps, we trained an L2-penalized multinomial linear regression model, an L2-penalized 3D CNN, an L2-penalized 3D CLN with the same architecture as the 3D CNN, and a 3D CNN and a 3D CLN with 4-step minibatch adversarial training (Algorithm 1). The CNN uses 3 × 3 × 3 filters, with five convolutional layers (32, 32, 64, 64, 64 filters and ReLU non-linearities) and one final softmax layer (Table 2). These were all trained using Adam (30) and a batch size of 32. The CNNs and CLNs were also trained with weight normalization (31). Hyperparameters, such as the L2 coefficient and adversarial noise bound, were set using the validation data. For SmoothGrad, $\sigma_\delta = 2.0$ was set to maximize interpretability on the validation data. For adversarial training, the adversarial noise bound, *E*, was set to 95, which corresponds to a noise of 0.1 (i.e. 10% of a standard deviation) per voxel.

Table 1. The contrast to task mapping used to create labels for the HCP data. Note that, in the absence of a relevant control condition, we used the average of the condition versus baseline contrasts.

HCP Task	Contrasts or Conditions Used
Working memory	2-BACK versus 0-BACK contrast
Gambling	Average of REWARD and PUNISH versus
Baseline Motor	Average of LF, LH, RF, RH, T versus baseline
Language	MATH versus STORY contrast
Social	TOM versus RANDOM contrast
Relational	REL versus MATCH contrast
Emotion	FACES versus SHAPES contrast

Table 2. The 3D convolutional neural network architecture used for HCP fMRI task decoding.

Layer	Filter	Stride	Padding	Non-linearity	Pooling
1	32x3x3x3	1	1	ReLU	2x2x2 Avg.
2	32x3x3x3	1	1	ReLU	2x2x2 Avg.
3	64x3x3x3	1	1	ReLU	2x2x2 Avg.
4	64x3x3x3	1	1	ReLU	2x2x2 Avg.
5	64x3x3x3	1	1	ReLU	2x2x2 Avg.
6	7x57768256	–	–	Softmax	–

The test set accuracies were 97.43% (multinomial regression model), 91.33% (correlation classifier), 95.66% (CLN), 99.19% (CLN with adversarial training), 98.39% (CNN), 98.88% (CNN trained with additive random noise), and 98.07% (CNN with adversarial training). The correlation classifier had a significantly lower accuracy than all of the other methods ($p < 0.05$, corrected), per paired *t*-tests across test participants. The CLN has a significantly lower accuracy than the CLN with adversarial training, the CNN, and the CNN with additive random noise during training ($p < 0.05$, corrected). All the other pairwise comparisons resulted in no significant difference ($p < 0.05$, corrected).

Class Informative Activation (CIA) We computed an approximate ground truth CIA map for each decoding class *in each participant*, by subtracting a participant's average example across classes from the participant's average example of each class. This is approximate in that each example is intrinsically noisy, even though it is already the product of a GLM applied to data from several trials within the corresponding condition. In order to evaluate each saliency map method, we computed the voxelwise correlation between the map produced for each input example, with the gradient for its correct class, and the CIA map. This yields a sample of correlation values for each method. The average correlations for each method are shown in Figure 6a, and the results of the pairwise paired *t*-tests comparing them are shown in Figure 7a.

Overall, the adversarially trained DNN has significantly higher CIA than the other methods. An important note is that the correlation classifier has lower CIA than any of the other methods. This classifier works by comparing each test example to the average example of each class in the training set and, as shown earlier in synthetic data, is unable to consider activation in some locations in the light of activation elsewhere. This is necessary in order to accommodate participant-specific variations in activation location. In Figure 8a, we show the average example across participants for three classes (row 1), and the corresponding CIA maps (row 2). The saliency maps produced with the various methods are shown for one class (language) in column 1 of Figure 8b.

Pairwise Class Activation Differences (CAD) We computed ground truth class pairwise activation difference maps for each pair of classes *and each participant*, by subtracting a participant's average activation map, across conditions, for each class from the average maps of all other classes. We applied this approach to the gradient-based saliency maps for each possible class produced for each test example, using each of the evaluated methods. For each method, and across all test examples, we correlated the pairwise gradient-based saliency maps with the corresponding pairwise CAD maps, yielding an average correlation value across incorrect classes; across test examples, this results in a sample

 CC By 4.0: © McClure et al.

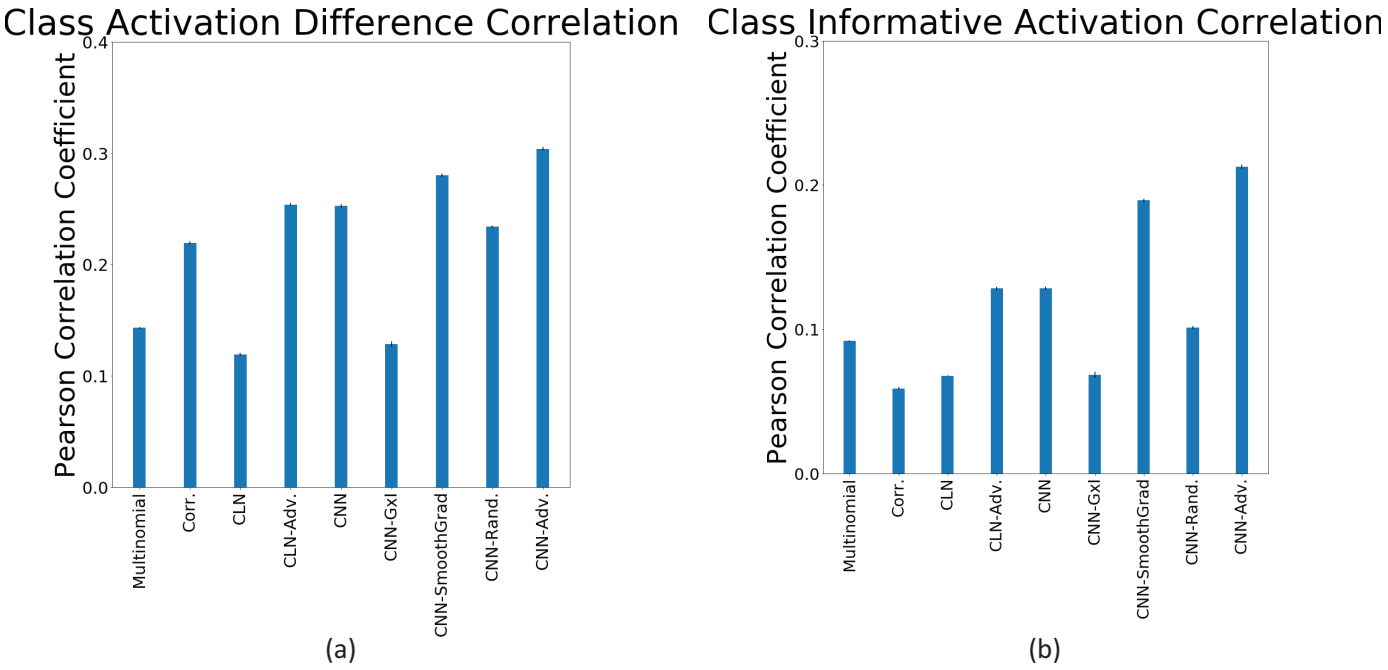

Fig. 6. The average interpretability scores (with standard errors) for saliency maps on the HCP test set, according to our two quantitative evaluation criteria. The linear multinomial regression model gradients, the correlation classifier gradients (Corr.), the convolutional linear network (CLN) gradients, the CLN with adversarial training gradients (CLN-Adv.), the convolutional neural network (CNN) gradients, the CNN gradients times inputs (CNN-GxI), the CNN with SmoothGrad (CNN-SmoothGrad) gradients, the CNN trained with additive random noise (CNN-Rand.) gradients, and CNN with adversarial training (CNN-Adv.) gradients. **a:** The average correlations versus the task informative activation map, across all test participants, with standard error bars. **b:** The average correlations versus the pairwise class activation difference map, across all tasks and test participants, with standard error bars.

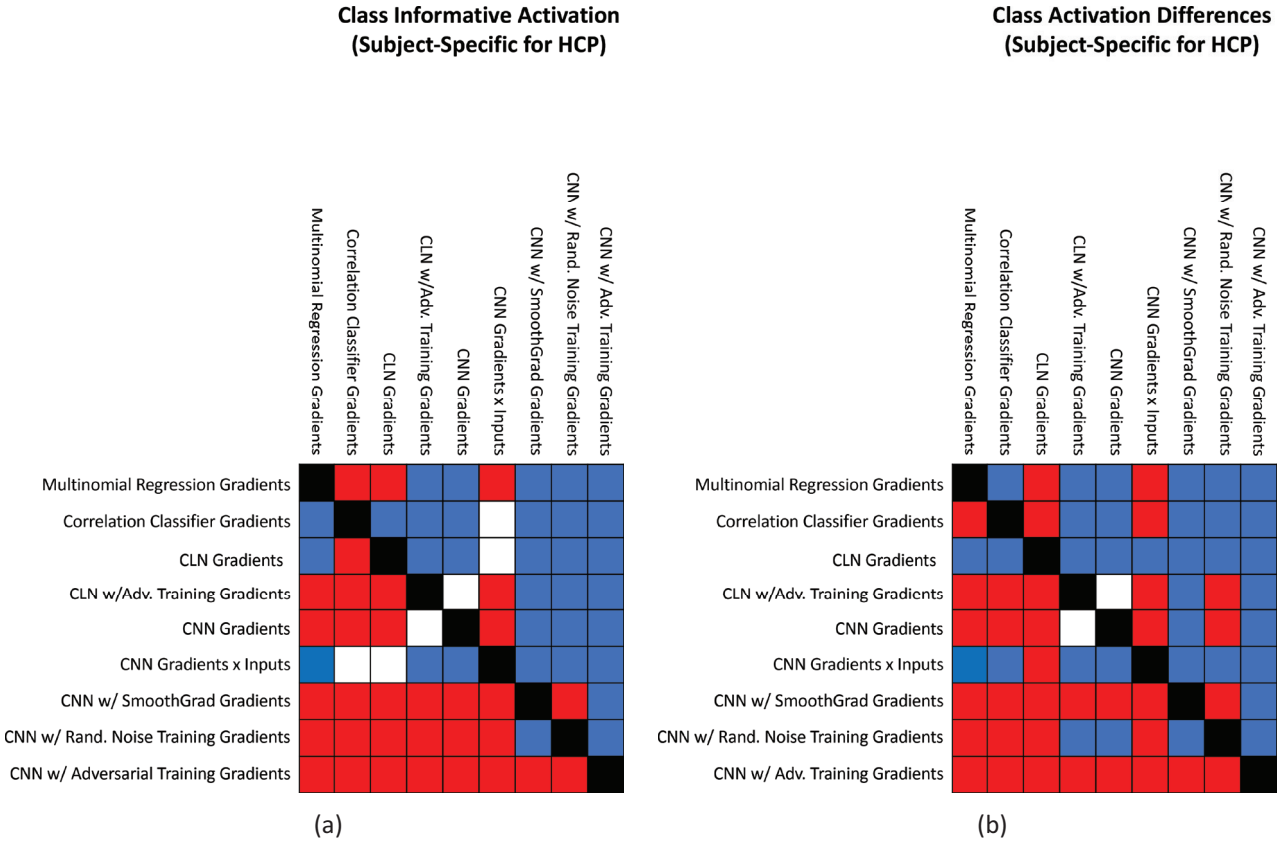

Fig. 7. The results of the pairwise paired *t*-tests for the participant-specific class informative activation and class activation differences correlations, across test examples, for the HCP data. Blue indicates that a method is significantly different ($p < 0.05$, corrected) from another method and it has a smaller mean correlation. White indicates that the two methods being compared do not have significantly different ($p < 0.05$, corrected) correlations. Red indicates that a method is significantly different ($p < 0.05$, corrected) from another method, and it has a larger mean correlation. The comparisons should be read left to right rowwise.

 CC By 4.0: © McClure et al.

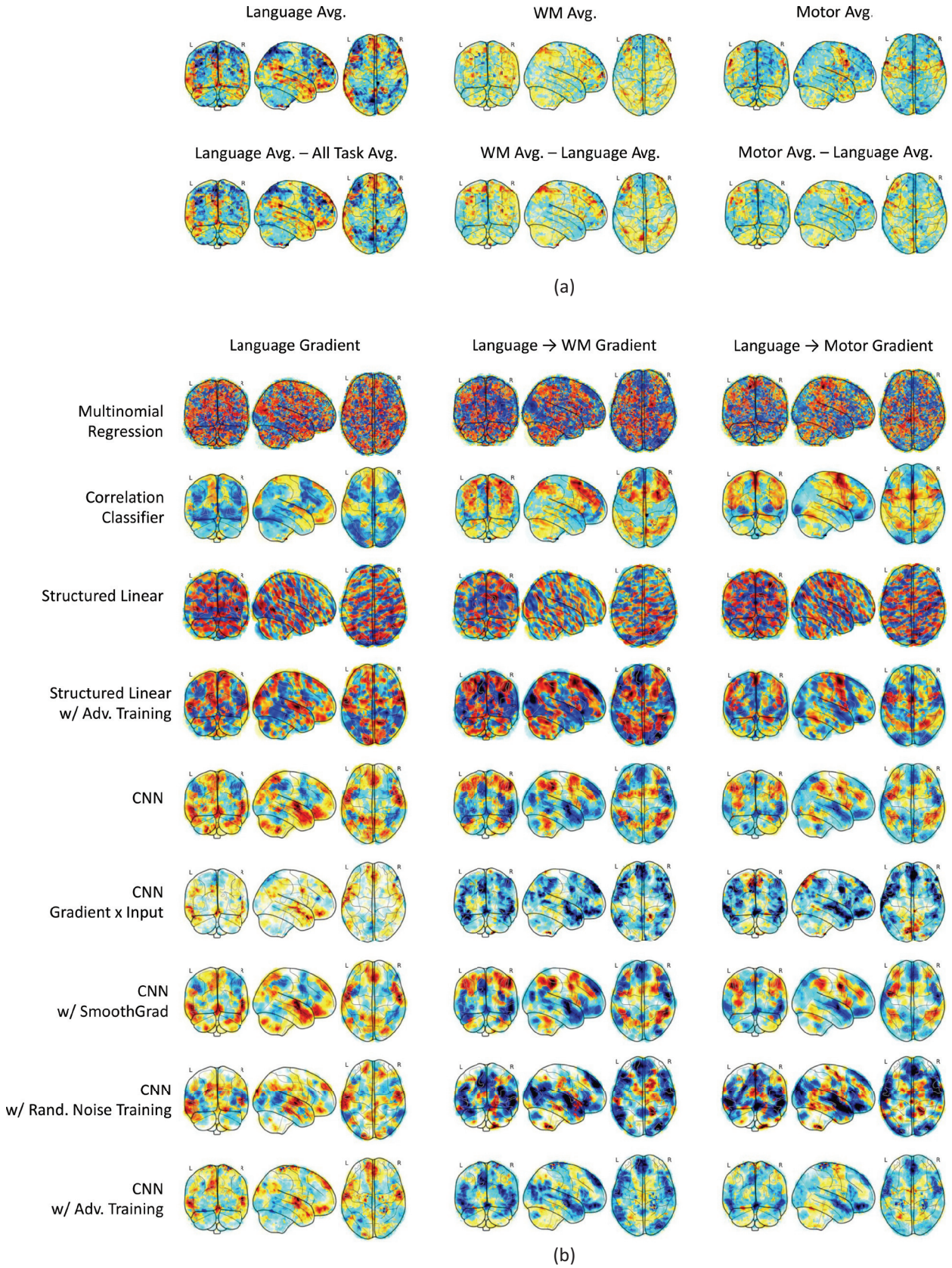

Fig. 8. Visualization of gradient-based saliency maps for a participant from the HCP test data. a: In row 1, the coronal, sagittal, and axial views of "glass brains" of the examples for three classes, Language, Working Memory (WM), and Motor. In row 2, the class informative activation map obtained by subtracting the average example across classes from the Language example (column 1) and the pairwise class activation difference maps for WM (column 2) and Motor (column 3) with respect to Language. **b:** The average gradient-based saliency maps (rows 1–4) for the Language class, applied to the example of the Language (column 1), WM (column 2), and Motor (column 3) classes. (Each "glass brain" was individually scaled, using the 99.9999th percentile of absolute values across the brain to define the extremes.)

 CC By 4.0: © McClure et al.

of correlation values. The average correlations for the different methods are shown in Figure 6b. For any two methods, we used a two-sided paired *t*-test to compare whether their average correlation was different. We used Bonferroni correction to account for all of the pairwise comparisons. The results of this comparison are shown in Figure 7b.

Overall, the adversarially trained DNN has significantly higher CAD than the other methods. In Figure 8b, we show the average gradient-based saliency map produced by each method for the language class, applied to the average example of the language (column 1), WM (column 2), and motor (column 3) classes. Note how these gradient maps remove the most prominent activation in the maps of the correct class (column 1) when transforming to a different class (columns 2 and 3). In particular, note that the gradient times input language to motor gradient map (Figure 8b, column 3, row 3) removes activation where the input activation is high but does not indicate that activation should be added around the central sulcus.

DISCUSSION

In neuroimaging, and many other areas, machine learning models must not only have high prediction performance but also be interpretable, correctly capturing the information in the input space that is relevant to the prediction. Linear methods have been popular in neuroimaging because they have high performance in many prediction problems and are seen as interpretable. However, linear models often learn population-based information and, unlike non-linear models, cannot use different information for different participants. DNNs are being increasingly used to make predictions in cognitive neuroscience and neuroinformatics applications. While successful, in terms of prediction performance, they are widely seen as uninterpretable "black boxes," as it can be difficult to discover what input information is used by a model to make predictions. A common approach to addressing this concern is to produce gradient-based saliency maps, visualizations of the relative importance of input features in prediction. Many methods for generating saliency maps have been introduced in the machine learning literature. However, they are known to be brittle due to focusing too much on the input or being extremely sensitive to small input noise. Finally, it is also challenging to evaluate how well saliency maps correspond to the truly relevant input information, given that ground truth is not always available. There is also no consensus on how to quantitatively evaluate the quality of the resulting maps.

In the light of the increasing usage of saliency map methods in neuroimaging, we believe it is timely to compare their effectiveness. To that effect, we introduced two quantitative criteria to evaluate the interpretability of saliency map methods for fMRI task decoding. These criteria are well-defined whenever a model is being trained to decode some information from imaging; they are, therefore, of broad applicability. We first described the rationale for the criteria using synthetic data sets, where the activation structure is known. There, we also drew the distinction between activation and information (activation present in some tasks but not others), as the latter is what should be captured in a saliency map. In the evaluations of maps in prior neuroimaging work, the ground truth is often a map of known activation for a particular task, so we believe that showing that this distinction matters in practice is a useful contribution. Our quantitative criteria are based on ground truth knowledge, in synthetic data, and an approximation thereof, in real data. The latter differs in many significant ways, e.g. variability in the location of the corresponding activation across participants, noise distribution, and ratio of informative features to total number of features, just to pick a few. Our criteria are based on the notion that activation is informative mainly to the degree that it differs between a few (or most) of the classes being compared.

We provided a brief review of common gradient-based saliency map methods for NNs: plain gradient, gradient times input, SmoothGrad, and gradients with random and adversarial noise during training. We also introduced a new adversarial training method we developed to make DNNs robust to input noise, with the goal of improving saliency map quality. We then evaluated all of these methods with NNs trained to perform task decoding, in both synthetic data and the task activation data in the HCP data set, using both of our quantitative criteria.

Our experiments led to several findings across both synthetic and real data sets. First, we found that gradient-based saliency maps produced with different methods vary widely in interpretability, as measured by our quantitative criteria in synthetic and HCP data. This was the case even for models that had similar prediction performance. Another finding was that using adversarial training improves the quality of the gradient-based saliency maps of both linear and deep NNs, relative to other methods. Those methods fail by being too sensitive to noise or by introducing information from the input image into the final map (e.g. gradient times input). An important test of the basis of our quantitative criteria was the use of a correlation classifier. The correlation classifier makes decisions based on the average class activation maps. This scores high in synthetic data where the activation of each participant is generated from the same spatial template. It does less so in synthetic data where there is participant-specific variation in the location of the corresponding activation. It does worst in the HCP fMRI data where, in addition to participant variation, there are many more features, most of which will be uninformative. There is also more noise, with a far more complex spatial distribution. Why is this important? In general, researchers want to use a prediction model for the purpose

 CC By 4.0: © McClure et al.

of identifying activation that is informative, not just for generating a prediction. The information identified may vary depending on both characteristics of the data such as those described earlier and aspects of the prediction model (e.g. regularization and whether feature selection was used). A saliency map for a particular model combines the information the model gathered in the training set with the specific activation seen in the test participant. Intuitively, this can happen in different ways, for example:

- reducing the effect of noise, both by averaging across training set participants and also downweighting test participant activation that does not look like any in the training set
- accommodating variation, by matching activation in the test participant to activation seen in some but not all training set participants
- contextualizing activation in multiple locations in the test participant (i.e. is activation in one location specific to one class, or to subset of them? Is co-activation of two locations more specific?)

How well methods do on our criteria depends on their capacity to address the factors discussed earlier. We can determine whether they do in synthetic data, given the known ground truth. In the HCP fMRI data, the ground truth is an approximation, given that images are noisy. However, we believe it is still adequate, at least for relative comparison between methods. It is likely that methods that do worse in our measures do so mainly because they are less capable of capturing participant-specific variation in activation location. If so, then the resulting saliency maps will ignore more of the participant-specific activation, something that does not happen (as much) for the adversarially trained DNNs. Overall, we found that using a non-linear model, specifically a NN, in conjunction with adversarial training led to saliency maps that consistently performed well on the synthetic data and performed significantly better than all other methods on the HCP fMRI data. These results indicate that non-linear models and adversarial training can be used to build models that better capture participant-specific information in fMRI decoding. Our results also suggest that adversarial training is a promising approach when compared with other methods and can be carried out in a computationally efficient manner with the procedure we introduce.

With regard to future work, the most obvious direction is to expand this comparison to other large data sets with many more tasks, as the distinction between activation and information will become even more critical. In particular, we could apply the adversarial training procedure to regression problems such as trait and other phenotype measure prediction problems, which are likely to be more clinically relevant. Finally, we would like to address the limitations of using gradients, which are first-order (i.e. linear), local approximations of the optimization landscape, by using input Hessians (i.e. second-order approximations of the optimization landscape). This should increase the interpretability of complex machine learning models and their widespread adoption in the scientific community.

ACKNOWLEDGMENTS

This research was supported by the National Institute of Mental Health Intramural Research Program: ZIC-MH002968 (Patrick McClure, Ka Chun Lam, Francisco Pereira) and ZIC-MH002960 (Dustin Moraczewski, Adam Thomas). Patrick McClure acknowledges the support of the Naval Postgraduate School's Research Initiation Program. This research utilized the computational resources of the NIH HPC Biowulf cluster (http://hpc.nih.gov). Data were provided in part by the Human Connectome Project, WU-Minn Consortium (Principal Investigators: David Van Essen and Kamil Ugurbil; 1U54MH091657) funded by the 16 NIH Institutes and Centers that support the NIH Blueprint for Neuroscience Research, and by the McDonnell Center for Systems Neuroscience at Washington University.

REFERENCES

1. Kietzmann TC, McClure P, Kriegeskorte N. Deep neural networks in computational neuroscience. In: Oxford Research Encyclopedia of Neuroscience. Oxford University Press; 2019. https://doi.org/10.1093/acrefore/9780190264086.013.46
2. Khosla M, Jamison K, Ngo GH, Kuceyeski A, Sabuncu MR. Machine learning in resting-state fMRI analysis. Magnetic Resonance Imaging. 2019;64:101–121.
3. Abrol A, Fu Z, Salman M, Silva R, Du Y, Plis S, et al. Hype versus hope: Deep learning encodes more predictive and robust brain imaging representations than standard machine learning. bioRxiv. 2020. https://doi.org/10.1101/2020.04.14.041582.
4. Khaligh-Razavi SM, Kriegeskorte N. Deep supervised, but not unsupervised, models may explain IT cortical representation. PLoS Computational Biology. 2014;10(11):e1003915.
5. Pereira F, Mitchell T, Botvinick M. Machine learning classifiers and fMRI: A tutorial overview. Neuroimage. 2009;45(1):S199–S209.
6. Mensch A, Mairal J, Bzdok D, Thirion B, Varoquaux G. Learning neural representations of human cognition across many fMRI studies. In: Advances in Neural Information Processing Systems; 2017. p. 5883–5893.
7. Xie N, Ras G, van Gerven M, Doran D. Explainable deep learning: A field guide for the uninitiated. arXiv preprint arXiv:200414545. 2020.
8. Wang X, Liang X, Jiang Z, Nguchu BA, Zhou Y, Wang Y, et al. Decoding and mapping task states of the human brain via deep learning. arXiv preprint arXiv:180109858. 2018.
9. Thomas AW, Heekeren HR, Müller KR, Samek W. Analyzing neuroimaging data through recurrent deep learning models. Frontiers in Neuroscience. 2019;13:1321.
10. Yarkoni T, Poldrack R, Nichols T, Essen D, Van Wager T. NeuroSynth: A new platform for large-scale automated synthesis of human functional neuroimaging data. Frontiers in Neuroinformatics. 2011;5. https://doi.org/10.3389/conf.fninf.2011.08.00058.
11. Nieto-Castañón A, Fedorenko E. Subject-specific functional localizers increase sensitivity and functional resolution of multi-subject analyses. Neuroimage. 2012;63(3):1646–1669.
12. Van Essen DC, Smith SM, Barch DM, Behrens TE, Yacoub E, Ugurbil K, et al. The WU-Minn human connectome project: An overview. Neuroimage. 2013;80:62–79.
13. Nair V, Hinton GE. Rectified linear units improve restricted boltzmann machines. In: ICML, Haifa, Israel; 2010.
14. Kriegeskorte N, Douglas PK. Interpreting encoding and decoding models. Current Opinion in Neurobiology. 2019;55:167–179.

 CC By 4.0: © McClure et al.

15. Adebayo J, Gilmer J, Muelly M, Goodfellow I, Hardt M, Kim B. Sanity checks for saliency maps. In: Advances in Neural Information Processing Systems; 2018. p. 9505–9515.
16. Lundberg SM, Lee SI. A unified approach to interpreting model predictions. In: Advances in Neural Information Processing Systems; 2017. p. 4765–4774.
17. Ancona M, Ceolini E, Öztireli C, Gross M. Towards better understanding of gradient-based attribution methods for Deep Neural Networks. In: International Conference on Learning Representations, Vancouver, BC, Canada; 2018.
18. Ismail AA, Gunady M, Pessoa L, Bravo HC, Feizi S. Input-cell attention reduces vanishing saliency of recurrent neural networks. In: Advances in Neural Information Processing Systems; 2019. p. 10813–10823.
19. Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I, et al. Intriguing properties of neural networks. arXiv preprint arXiv:13126199. 2013.
20. Goodfellow IJ, Shlens J, Szegedy C. Explaining and harnessing adversarial examples. arXiv preprint arXiv:14126572. 2014.
21. Kindermans PJ, Hooker S, Adebayo J, Alber M, Schütt KT, Dähne S, et al. The (un) reliability of saliency methods. In: Explainable AI: Interpreting, Explaining and Visualizing Deep Learning. Springer; 2019. p. 267–280.
22. Etmann C, Lunz S, Maass P, Schönlieb CB. On the connection between adversarial robustness and saliency map interpretability. arXiv preprint arXiv:190504172. 2019.
23. Tsipras D, Santurkar S, Engstrom L, Turner A, Madry A. Robustness may be at odds with accuracy. In: International Conference on Learning Representations, Vancouver, BC, Canada; 2018.
24. Kim B, Seo J, Jeon T. Bridging adversarial robustness and gradient interpretability. arXiv preprint arXiv:190311626. 2019.
25. Shafahi A, Najibi M, Ghiasi MA, Xu Z, Dickerson J, Studer C, et al. Adversarial training for free! In: Advances in Neural Information Processing Systems; 2019. p. 3353–3364.
26. Smilkov D, Thorat N, Kim B, Viégas F, Wattenberg M. Smoothgrad: Removing noise by adding noise. arXiv preprint arXiv:170603825. 2017.
27. Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A. Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:170606083. 2017.
28. Guo C, Pleiss G, Sun Y, Weinberger KQ. On calibration of modern neural networks. In: Proceedings of the 34th International Conference on Machine Learning. vol. 7. JMLR.org; 2017. p. 1321–1330.
29. Barch DM, Burgess GC, Harms MP, Petersen SE, Schlaggar BL, Corbetta M, et al. Function in the human connectome: Task-fMRI and individual differences in behavior. Neuroimage. 2013;80:169–189.
30. Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:14126980. 2014.
31. Salimans T, Kingma DP. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In: Advances in Neural Information Processing Systems; 2016. p. 901–909.

 CC By 4.0: © McClure et al.