CORRELATING BRAINS AND DNNS FOR IMAGES AND COLORS

Ayushi Arora, Deeksha Vijay, Diksha Banka, Dravya Marwaha, Gaurvika Kapoor, Praveen Prabhat, Twinkle Arora
IIT Kanpur

July 29, 2021

1 Objective and Overview

Convolutional neural networks (CNNs) have made great advances in the computer vision fields. In this analysis, we study representation of colors in CNNs and in brains using the Python toolbox DNNBrain. We started with a literature survey, studying papers on how colors are perceived by humans and macaque brains. Then we moved to the DNNBrain toolbox, using its functionality to study how CNNs encode images as well as colors. We carry out a number of experiments (provided by DNNBrain) to do this. Finally, we carry out a classification task on color classes. Here we analyse two methods:

1. Transfer learning
2. Learning from scratch

Our hypothesis is that transfer learning will give better classification accuracy since features are extracted from CNNs which already contain some information of the input picture, as seen from the experiments carried out.

2 Overview of the toolbox, DNN and the dataset

2.1 DNNBrain[1]

DNNBrain is a modular python toolbox to integrate DNN software packages and brain mapping tools. It integrates representations of DNNs and the neural representations of the brain. It assembles stimuli, artificial activity data, and biological neural activity data together, helping in comparing representations of DNNs and brains.

2.2 AlexNet[2]

AlexNet model was primarily used as our DNN model which consists of eight layers and learnable parameters. The architecture is composed of five convolutional layers and three fully connected layers that receives inputs from all units in previous layer followed by a 1000 way softmax classifier.

2.3 BOLD5000[3]

BOLD5000 is a large-scale dataset of practical study of human FMRI with 5000 real-world images as stimuli. It overlaps with standard computer vision datasets by incorporating images from the Scene Understanding (SUN), Common Objects in Context (COCO), and ImageNet datasets.

3 Experiments on the CNNs

We carried out a number of experiments to analyse how CNNs encode images and colors. These experiments are based on the documentation of the DNNBrain toolbox. Some of them (marked *) are extended to color inputs.

3.1 Scan DNN*

Here, we extract and display feature maps of three images for each convolutional layer after ReLU. We extract the maximum activation channels for each image in each of the convolutional layers of AlexNet. We then display the activations for these channels to understand how each convolutional layer sees the initial image. We observed that the DNN representation of the image became more abstract along the depth of the layers. Following are the DNN representations we got for the 3 images.

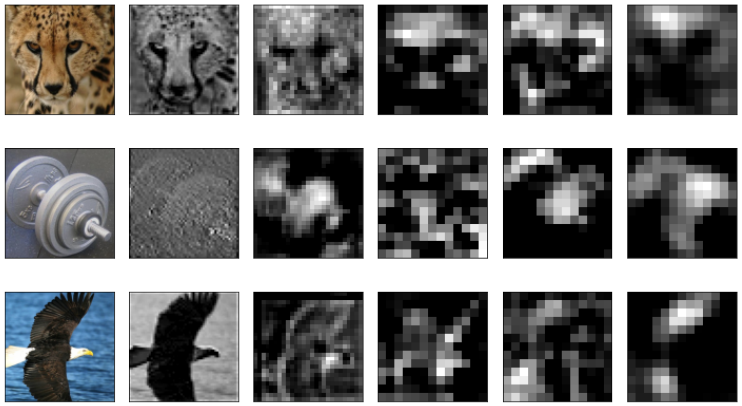


Figure 1: Feature maps for different layers of AlexNet

Now, to understand the perception of colors by the DNN, we passed plain color images in the DNN and observed the representations in different convolutional layers. Following are the resulting images. As it is evident from these images, the initial layers perceive the colors more or less the same but as we go deeper, it starts differentiating the colors.

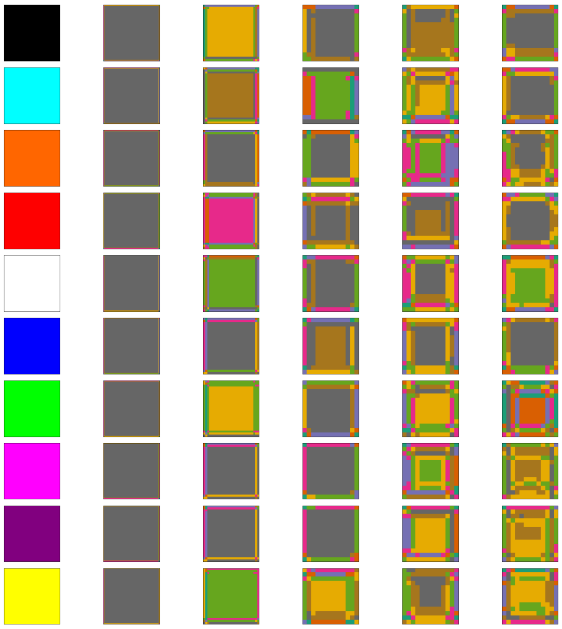


Figure 2: Feature maps for different layers of AlexNet for color inputs

3.2 Probe DNN

The objective of this experiment is to study whether specific stimuli attribute are explicitly encoded in a certain layer of DNN. The BOLD5000 stimulus images were sorted into binary categories - 2,547 animate images and 2,369 inanimate images, according to salient objects located in each image. We examine how animate information is explicitly encoded in AlexNet.
Firstly, we extract representations of these images in layers we are interested in.. To avoid overfitting, the dimension (i.e. the number of units) of the representations from each layer is reduced by PCA to retain the top 100 components. Then, we train a logistic regression model on the artificial representation from each Conv layer of AlexNet to decode the stimulus category. The accuracy of the model is evaluated with a 10-fold cross validation. Finally we summarize and plot the results.

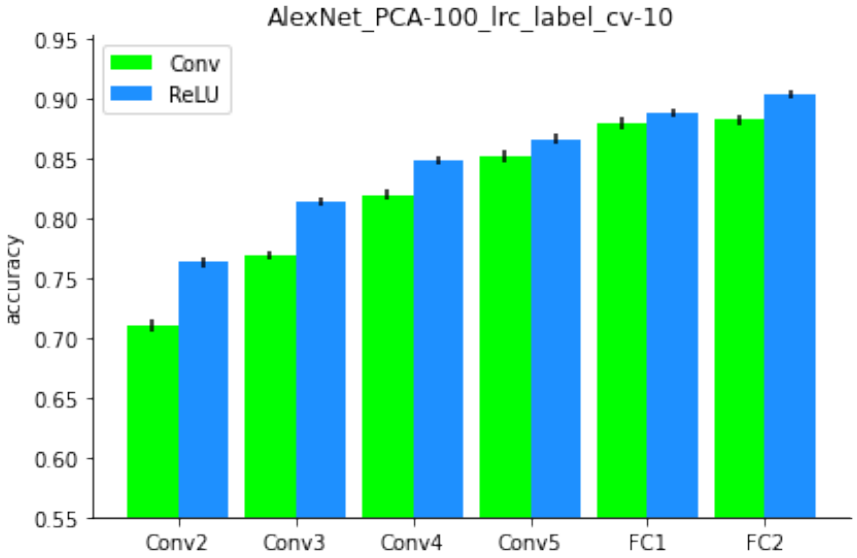


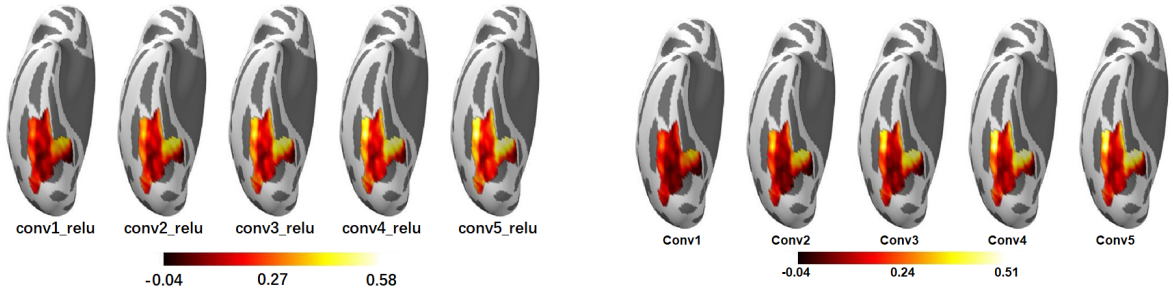
Figure 3: Accuracy for the different layers of AlexNet

As seen in the plot, the classification accuracy progressed with the depth of Conv layers, indicating higher layers encode more animate information than lower layers. Moreover, the ReLU operation within each convolutional layer plays a significant role in improving the representation capacities for animate information.

3.3 Map between DNN and Brain

In this experiment, we examined how well the representation from each layer of a CNN predict the response of a voxel in the human ventral temporal cortex (VTC). DNNBrain supports two kinds of analyses to link DNN representations to brain representations: encoding model (EM) and Representational Similarity Ananlysis (RSA).

- **Univariate Encoding :** We trained generalized linear models to map representations of each layer to each voxel within right VTC. The encoding scores were evaluated by Pearson correlation between the measured responses and the predicted responses using a 10-fold cross validation procedure. At each run of the cross validation before the generalized linear model, a PCA transformer was fitted on the DNN representations splitted as training set, and then we transformed DNN representations both in training and testing set to keep the top 100 components. The encoding score maps of each layer are shown as Figure 4 (a). The overall encoding score of the VTC gradually increased for the hierarchical layers of AlexNet, indicating that as the complexity of the visual representations increase along the DNN hierarchy, the representations become increasingly VTC-like.
- **Multivariate Encoding :** We built a PLS model to map representations of each layer to the whole right VTC. The encoding scores were evaluated by Pearson correlation between the measured responses and the predicted responses using a 10-fold cross validation procedure. The encoding score maps of each layer are shown as Figure 4 (b). The results are similar as the univariate encoding model, indicating that interactions between different voxels encode little representation information from each DNN Conv layer.

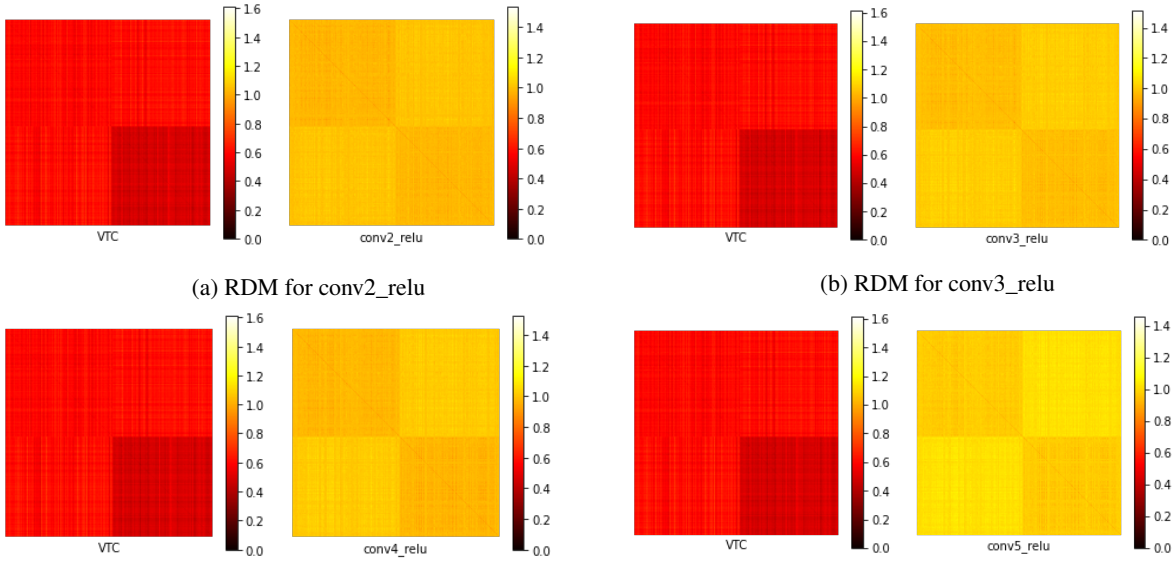


(a) Encoding score maps of each layer for Univariate Encoding

(b) Encoding score maps of each layer for Multivariate Encoding

Figure 4: Encoding Model analysis

- **Representation Similarity Analysis:** In this, we compared the representations of the DNN and that of the brain using a representational dissimilarity matrix (RDM) as a bridge. Firstly we made the dimension (i.e. the number of units) of the representations from each layer reduced by PCA to retain the top 100 components, in order to reduce the computation load. Then, we created RDMs created to measure how similar the response patterns are for every pair of stimuli using the multivariate response patterns from the DNN and the brain, respectively. The RDMs are displayed in Figure 5. Finally, we calculated representation similarity between the DNN and the brain as correlation between their RDMs.



(a) RDM for conv2_relu

(b) RDM for conv3_relu

(c) RDM for conv4_relu

(d) RDM for conv5_relu

Figure 5: Representation similarity analysis for the different layers of AlexNet

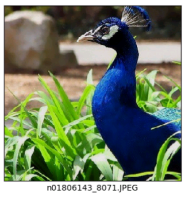
3.4 Top Stimulus

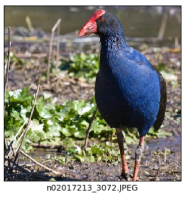
Top Stimulus is a visualization approach to identify top 'k' images with the highest activation for a specific unit of DNN from a large image collection. We select top3 stimuli for 10th, 85th and 131st channels of fc3 layer of AlexNet model. For each unit, it generates information of the top3 images. The corresponding images are shown in figure 6.

3.5 Saliency Image*

Saliency image is acquired by computing gradients on the input images relative to the target unit by a backpropagation algorithm. It highlights pixels of the image that increase the unit's activation most when its value changes.The purpose of the saliency map is to find the regions which are prominent or noticeable at every location in the visual field and to

(a) fc3_chn10_top3



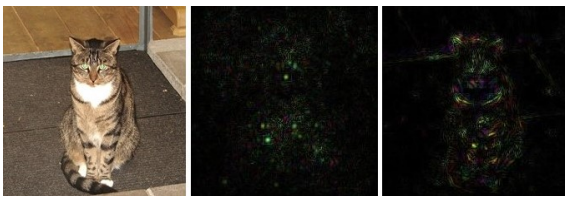


(b) fc3_chn85_top3

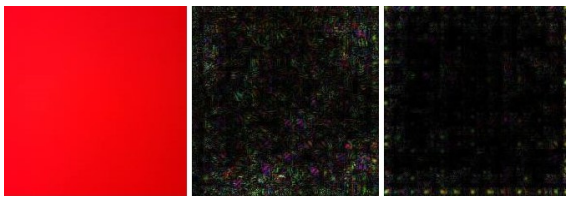
(c) fc3_chn131_top3

Figure 6: Top3 Images for channels 10, 85 and 131

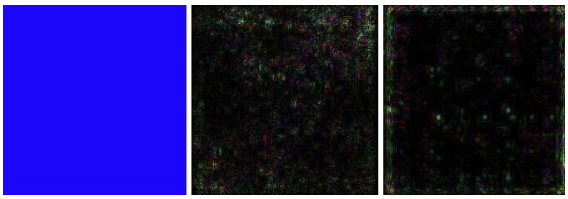
guide the selection of attended locations, based on the spatial distribution of saliency. We used two different algorithms to obtain saliency images - Vanilla Saliency Images and Guided Saliency Images, using built in libraries in DNNbrain. The tutorial was first tested on a random image and then extended to three color inputs. Results are in figure 7.



(a) Cat



(b) Red



(c) **Blue**

(d) Yellow

Figure 7: Saliency images for an image of a cat and color inputs

3.6 Synthesis Image

In this section, we synthesized an image to extract the highest activation of target unit. The tutorial gave the optimal image of the target channel 131 in layer fc3, which is responsible for flamingo category. We computed the activation loss function of the channel using activation and regularization constraint that deals with the outliers in the pixels. The image was processed in every iteration using gaussian blur to mitigate high frequency. To smoothen the gradient of activation at every iteration we applied Fourier filter. Finally, we synthesized the image after setting optimal metric parameters with 150 iterations and a learning rate of 0.1. The synthesized image is shown in figure 8.

3.7 Minimal Image

Minimal image is a way to simplify a stimulus into a minimal part which could cause almost equivalent activation as the original stimulus using DNN. Here, we have extracted the minimal image of a tiger and extended the same to three color inputs. Since CNNs focus on shapes and edges in an image, it is interesting to note that the minimal images for colors show some sort of grid patterns as shown in figure 9.

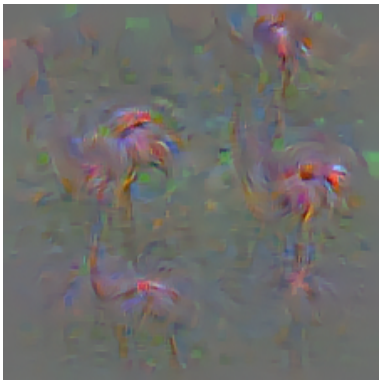


Figure 8: Synthesized optimal image of flamingo

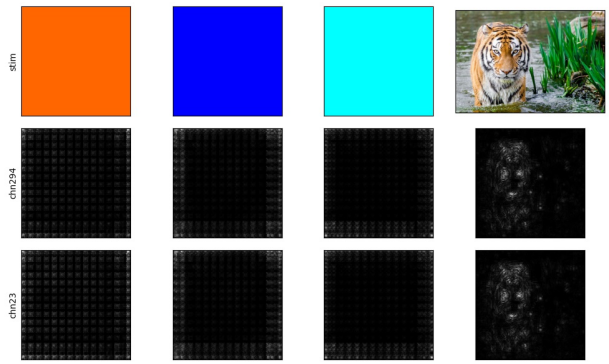


Figure 9: Minimal Image

3.8 Occluder Map

Occluder map is a way to visualize which regions of the image lead to the high unit activations. First, we use a small occluder with certain size and move it in a dense grid with certain stride for each image. When this occluder moving from top left to bottom right, it will generate thousands of images per original image. Then these "occluder images" are fed into the network and record their activations. If the occluded patch is important, there will be a large discrepancy in the corresponding area. Finally we average the output feature maps and get the discrepancy occluder map.

3.9 Upsampling Map

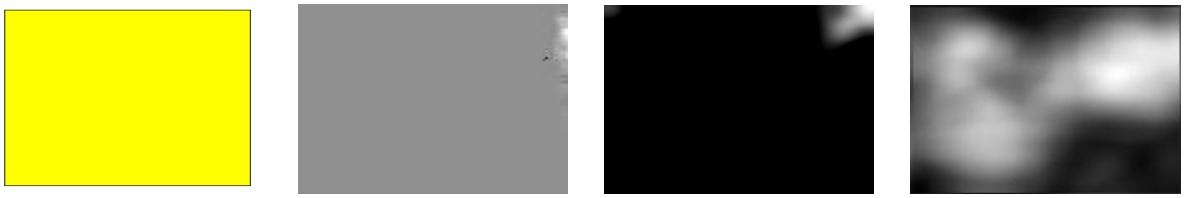
Upsampling map is a way to visualize which regions of the image lead to the high unit activations. First, we fed the image into the network and got its feature map at the given channel. Then we use some interpolate method to upsample the feature map into the original image space. After setting the threshold to filter the map, we finally get the upsampling map.

3.10 Empirical Map

Empirical map is a way to visualize a set of images' receptive field. While generating the empirical map, we first assigned an engine to get regions of the image that lead to the high unit activations(occluding or upsampling). After acquiring these regions, we averaged the patch and got the final empirical map.

4 Classification Task

A classification task is carried out for a color dataset. Our hypothesis is that transfer learning will give better accuracies than the model which learns from scratch. We test this hypothesis.



(a) Original (b) Occluder Map (c) Upsampling Map (d) Empirical Map

Figure 10: Original Image along with the maps

4.1 Generating the dataset

To check the performance of both the models, we generated a dataset by creating images using different RGB values and creating numpy arrays using these RGB values. We generated 11 Basic Color Categories for Classification. These Basic Colors included Red, Green, Blue, Yellow, Orange, Pink, Purple, Brown, Grey, Black, and White. So we labeled around 5000 RGB Colors for the Classification Task.

4.2 Learning from scratch

We defined a DNN which consisted of one convolutional layer and two fully connected layers along with ReLU activation and 2D Max-Pooling. This model is designed to receive a tensor with shape as (nsample, 3, 224, 224), and do the classification. For each image channel, we calculate the mean and standard deviation among all training images to be used in data normalization. We derived a subclass from DNN class that transforms training and testing images to input tensors. The transformed test data was used during the evaluation of model's generalization on validation data at each epoch, which was set as 10.

4.3 Transfer learning

We used two major transfer learning scenarios: **finetuning** and **fixed feature extractor**.

- **Fixed Feature Extractor :** Take a ConvNet pretrained on ImageNet, remove the last fully-connected layer (this layer's outputs are the 1000 class scores for a different task like ImageNet), then treat the rest of the ConvNet as a fixed feature extractor for the new dataset. In an AlexNet, this would compute a 4096-D vector for every image that contains the activations of the hidden layer immediately before the classifier. We call these features CNN codes. For better performance, we applied ReLU activation on them. Once we extracted the 4096-D codes for all images, we trained a linear classifier for the new dataset.
- **Fine-tuning :** The strategy here is to not only replace and retrain the classifier on top of the ConvNet on the new dataset, but to also fine-tune the weights of the pretrained network by continuing the backpropagation. Due to overfitting concerns, instead of fine-tuning all the layers of the ConvNet, we kept some of the earlier layers fixed and only fine-tuned some higher-level portion of the network.

Both of the two scenarios use a pretrained DNN and replace the final fully connected (FC) layer with a new one with random weights to do a new task.

Model	Train split	Train Accuracy	Test split	Test Accuracy
Learning from Scratch	0.85	0.85	0.15	0.83
Transfer Learning (Feature Extractor)	0.85	0.88	0.15	0.87
Transfer Learning (Fine-Tuning)	0.85	0.86	0.15	0.85

Table 1: Accuracy Data from the DNNs

5 Results and Discussion

We carried out an analysis for convolutional neural networks (AlexNet), looking at how images are encoded in different layers with the help of various experiments. We extended these analysis to color inputs, which showed how CNNs don't work quite the same compared to when there are salient parts present in the image. This is because CNNs are based on looking at shapes and edges present in the image. Though this still does not prove whether it does not contain any

information for color images.
To test this, we carried out a classification task for a dataset just containing color images by two methods. Training from scratch and transfer learning. Comparing the accuracy scores, it is clear that transfer learning performed better. This is because the DNN does indeed contain usable information of the color images, which helps in distinguishing one color from the other. And this shows that CNNs do encode information to differentiate between colors.

6 GitHub Repository

`https://github.com/shivigup/Color-DL-Brain`

7 Contributions

Members: Ayushi Arora, Deeksha Vijay, Diksha Banka, Dravya Marwaha, Gaurvika Kapoor, Praveen Prabhat, Twinkle Arora
Mentor: Shivi Gupta

References

[1] Xiayu Chen, Ming Zhou, Zhengxin Gong, Wei Xu, Xingyu Liu, Taicheng Huang, Zonglei Zhen, and Jia Liu. Dnnbrain: A unifying toolbox for mapping deep neural networks and brains. *Frontiers in Computational Neuroscience*, 14:105, 2020.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

[3] Nadine Chang, John A. Pyles, Austin Marcus, Abhinav Gupta, Michael J. Tarr, and Elissa M. Aminoff. Bold5000, a public fmri dataset while viewing 5000 visual images. *Scientific Data*, 6(1), May 2019.