**Visual memory representations project overview**

([Project repo](https://github.com/edenjenzohar/visual_memory_representation_final))

This document will explain in a detailed manner the project, from the initial stages of preparing the dataset to running the experiment and analyzing the data. It will be divided into sections:

1. Background
2. Creating the dataset
3. Creating the experimental resources
4. Pavlovia
5. M-Turk
6. Analysis

The purpose of this overview is to guide any future projects that are a continuance of this project. For any further questions you may contact:

* Eden Zohar: 0525207088, [edenzohar04@gmail.com](mailto:edenzohar04@gmail.com)
* Stas Kozak
* Dekel Abeles

**Part 1: Background**

This project was created due to exploration of a different question, regarding the calculation of similarity between images using CNNs. We thought about the fact that when researchers use images in their studies, then choose them in a subjective manner, with no balancing. this can result in tests with different difficulty levels, just from some images being more similar than others, resulting in a harder test and lower accuracy. This creates confounding because the results are created from the datasets not being balanced and not from a real phenomenon.

We understood that there is a need for an objective and supervised method for choosing datasets and creating experimental tests that use images in a balanced manner. Out solution for this problem was the use on CNNs to represent images in a numeric way and quantify the distance (or similarity) between images using a distance matric.

One of the well-known qualities of CNNs is their hierarchal processing stream. The early layers represent low level features of images and the late layers high level features. This hierarchal processing has been investigated in the past few years and a significant correlation has been found to the human visual processing stream. Such that the early layers correlate to V1 and late layers to high order visual processing brain regions.

One of the questions regarding the human visual processing stream is regarding visual memory and how this type of memory changes over time. Some theories (reverse hierarchy theory) state that from the initial stages of encoding, the memory is represented in a high-level manner, such that only the high-level information of the visual stimuli can be retrieved. On the other hand, other theories (xxx) state that the visual representations go thought a transition from low-level features at initial encoding to high-level representation over time. One of the common ways to answer this question was to subjectively define what a low-level vs high-level feature is and to test the memory to this feature though time. What became apparent is the different definitions that existed in the literature for what is each type of feature, and that this definition was prone to subjective interpretation.

We thought that a way to overcome this was to use CNNs as an objective tool to define what is a low-level vs high-level feature of an image, using the features represented in the low layers of the CNN as the low-level features and the features represented in the high layers of the CNN as the high-level features. We used a multi category image set of outdoor scenes (badlands, highways, mountains, playgrounds, golf courses and brides) and computed the similarity between all the images in the dataset. Later, we chose 60 target images, 10 from each category, and picked 2 distractors for each target image. The first distractor was an image that had high similarity to the target image in the low-levels and low-level similarity in the high-levels, this was defined as the low-level distractor. The second distractor was an image that had high similarity to the target image in the high-levels and low-level similarity in the low-levels, this was defined as the high-level distractor. Now, we could test the memory for the target images at different time intervals using the different distractors. For example, if the subject encodes the images and immediately after is asked to choose the image he was previously seen, will it be harder for him to choose between the target image and the low-level distractor? or the target image and the high-level distractor? This way we can understand if it is the high-level features, or the low-level features that he remembers.

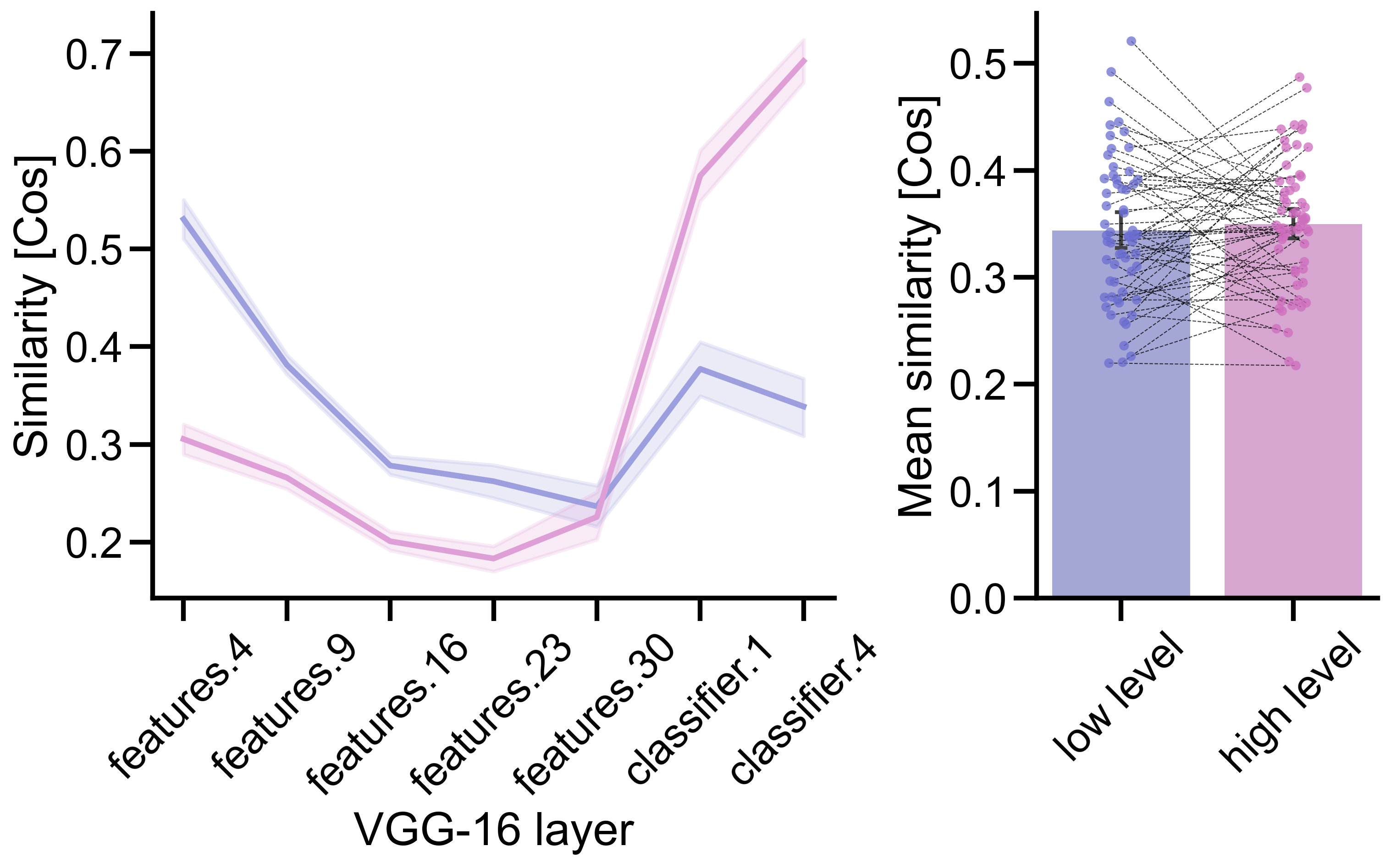
\*\* add from where we took the dataset with link

**Part 2: Creating the dataset.**

We chose to use images of scenes after previous pilots were done on images with an object and background (birds/fungi). We understood that some subjects, when told to “remember the image” look only at the object and not the image as a whole. This created a problem because a CNN calculates the similarity between all the pixels of an image, not only the object.

Next, we wanted to choose 60 target images, 10 from each category that had other images from the dataset that could be chosen as high/low level distractors. After multiple failed attempts (explain why problematic?), we used mixed linear programming to solve this problem (with the help of Moni). This was done using the computational power of google cloud. ([user-managed-notebook-1679582742](https://console.cloud.google.com/vertex-ai/workbench/locations/me-west1-a/user-managed/user-managed-notebook-1679582742?hl=en&project=memory-representations))

This resulted in 60 target images, with each image having a distractor form the low-levels and high-levels and the same time, the mean similarity over all the target and distractor of the low-level vs the mean similarity over all the target and distractor of the high-level was not significantly different.



This part has a few codes and figures:

* **Google cloud:**   
  the google cloud [account](https://console.cloud.google.com/welcome?hl=en&project=memory-representations): the output is saved to the google cloud bucket ([link](https://console.cloud.google.com/storage/browser/memory_represetnations_data;tab=objects?forceOnBucketsSortingFiltering=true&hl=en&project=memory-representations&prefix=&forceOnObjectsSortingFiltering=false)) and from their all that’s left is to save the output to your local machine in the directory (*under projects/project\_name/)*the [vertex ai](https://console.cloud.google.com/vertex-ai/workbench/managed?hl=en&project=memory-representations) holds the google collab notebook for running the code. The notebook is located under “user managed notebooks” and is named “[user-managed-notebook-1679582742](https://console.cloud.google.com/vertex-ai/workbench/locations/me-west1-a/user-managed/user-managed-notebook-1679582742?hl=en&project=memory-representations)” .
* **The “show\_selected\_images\_and\_csv” code:**located at: *functions/multicategory memory representation/experiment preperation/show\_selected\_images\_and\_csv.ipynb*

This notebook has two sections, the first creates a csv file “targets\_and\_distractors.csv” with the names of the targets and distractors and

plots the selected targets and distractors and saves a figure “mages\_and\_distractors” and saves it in the “figures and outputs” in the project directory. The second section creates the experimental csv files, this will be explained in the next section.

**Part 3: Creating the experimental resources.**

The experiment needs a few components to run:

1. Csv files: these are the experimental files for each subject. For the same day and 24 hours’ time gaps for each subject there are 2 csv files: encoding and test. These are saved in the:   
   *projects/multicategory/experiment 1/experiment resources/csvs*
2. Images: the images are saved during the “show\_selected\_images\_and\_csv” code at two destinations. The target images in the:   
   *projects/multicategory/experiment 1/experiment resources/images/targets*  
   and all the images used (distractors + targets):   
   *projects/multicategory/experiment 1/experiment resources/all experimental images*
3. Experiment resources: this includes the attention check arrows, the demo encoding and test csv files and the Pavlovia stuff (explained in the Pavlovia section)

**Part 4: Pavlovia**

After the csv and images folders are ready, we can start creating the experiment. To do so, create a new folder on your PC and copy to it the csv files, all the experimental images (60+60\*2 = 180 images) and the experimental resources (arrows, demo images and csv files). Then you can create a new Psychopy experiment or use an existing one, don’t forget to update in the “settings” under “online” the resources by choosing all the files in the folder.

This will create:

* Index file: this is a file with the Psychopy version, name of experiment.
* A JavaScript file: this is the JavaScript file for running the experiment. This is the code you will modify for personal changes on Pavlovia.

After you have these files, go to [Pavlovia](https://gitlab.pavlovia.org/CensorLab) and create a new project:

1. Create a README and then by pressing on “web IDE” you can choose to upload multiple files; this is where you upload all your files from the folder you created.
2. After this is ready, you can active your project by going to “experiments”, choosing your new project and pressing “PILOT”.
3. Create a new [shelf](https://pavlovia.org/dashboard?tab=3) for your experiment.
4. Make sure you turn “save in-complete results” off.
5. When you are sure everything is working, and you included all your personalized code (for interactive csv files loading per subject with shelf etc.) you can press on “RUN” and assign credits.

**Part 5: M-Turk**

* Log into the censorlab account and [create a new project](https://requester.mturk.com/create/projects/new) of “survey link” type. You can use one on the existing experiments as reference for filling in all the details.
* Create a new qualification for this experiment, if you are using the same stimuli as the previous projects, use the same qualifications as used in that experiment, we don’t want the same subjects returning.
* Paste the URL link from the Pavlovia of the new project.

**Part 6: Analysis**

To start the analysis, you need a few files:

* From the Pavlovia: download the results. This will download a “data” folder.
* From M-Turk: go to the manage->[workers](https://requester.mturk.com/workers) and click “download CSV”(this takes time).   
  go to your batch and click “download results”.

Follow these steps:

1. Create a new folder in the relevant time gap period, for example:   
   *projects/multicategory/experiment 1/collected data/pilot\_24\_hours*
2. In this folder create a new batch folder for “batch X”.
3. Copy to this folder the “data” folder downloaded from Pavlovia, make sure the only data in it is from the date you opened your batch (this happens only if you used the same experiment multiple times).
4. Copy the workers data as “batch\_workers.csv”
5. Copy the results as “Batch\_x\_results.csv”
6. Create new .txt file and copy the shelf into it. Name it: “shelf after encoding session closed.txt” / “shelf after test session closed.txt” / “shelf before test session.txt” / “shelf final state.txt”

Now you are ready to run the relevant parsing code.