Exploring feature visualization: how optimized images range across networks and approaches

Dawn Finzi (dfinzi@stanford.edu)

I. Motivation

In the last year or so, a flurry of papers have come out which probe the nature of neural computations in visual cortex by using convolutional neural networks to optimize images for neurons in macaque [1] [2] or mouse visual cortex [3]. An especially exciting example of this line of research found a linear mapping from CNN features to neurons in macaque visual area V4 and then used gradient ascent to synthesize images which optimally drive neuronal responses, achieving remarkable specificity and control of the neuronal sub-populations within this cortical area [1].

Of particular interest is whether such an approach would be feasible for human fMRI to characterize ventral temporal cortex (VTC) which is responsible for high-level visual categorization in humans. For example, to what extent would the images optimized for VTC reveal a modularity, with images of buildings optimized for known place-selective regions or images of body parts optimized for limb-selective regions? Or how might the ideal stimuli differ for the different face-selective areas in VTC? We know from previous research that there are different face-selective areas in ventral cortex that appear to be specialized for different tasks but fully characterizing these differences and their purpose has proved elusive.

However, before instantiating this approach for human fMRI can even be attempted, we need to explore the space of potential visualizations, namely:

1) How do design decisions impact the results? Particularly, how do different loss functions and preprocessing choices alter the end visualization?
2) How reliable across iterations are the images generated by this partly stochastic process?
3) So far neuroscience research has focused on AlexNet [4] but more biologically plausible networks have started to emerge (TNN) [5]. How do the features compare across these two networks, AlexNet and TNN, which have different architectures but are both trained on the same task (ImageNet [6])?

In pursuit of these questions, we implemented and experimented with gradient ascent for two main networks, AlexNet [4] and TNN [5], which were both pretrained on ImageNet [6] and weight frozen.

II. Related work

In addition to the motivational work at the intersection of visual neuroscience and machine learning, there has been substantial research on visualizing network features within the field of machine learning. The concept of visualizing the higher level features of neural networks was introduced in 2009 [7]. The particular approach we will be implementing executes feature visualization by iteratively adjusting input images to optimize a target aspect of a network. This approach has been highly successful in producing dramatic, psychedelic images when used to optimize for entire layers in convolutional neural networks [8] and has also been applied to nearly all aspects (units, channels, layers, class logits and class probabilities) of GoogLeNet [9]. While we will starting from random noise image tensors, a large body of work has focused on incorporating learned priors. This includes approaches such as learning a deep generator network to map a latent space to image data and then optimizing within that latent space instead [10]. This produces highly realistic visualizations, though it obscures what results from the prior vs. the model target.

III. Methodology

A. Model architectures

This project focuses on two model architectures (1) AlexNet [4] and (2) TNN [5]. AlexNet was chosen as a starting point due largely to its widespread use in neuroscience, including the recent work leveraging feature visualization [1]. We use an AlexNet checkpoint pre-trained on ImageNet and focus on optimizing for the five convolutional layers. The second architecture, TNN is a convolutional neural network closely inspired by the layout of the biological visual system and has been shown to achieve a better match to neural activity in the primate visual system [5]. It has ten convolutional layers, with layer 6 providing the best match to primate visual

area V4, and later layers providing a better fit to primate inferior-temporal cortex (IT). In contrast, primate V4 is best fit by AlexNet's 3rd convolutional layer [1].

B. Feature visualization approach

1) General implementation: We implemented feature visualization in TensorFlow [17] for individual units and channels within specific layers of the chosen networks. More specifically, we created a random noise image tensor to optimize and then instantiated the model graph and restored the model weights from a checkpoint for our pre-trained CNN of choice. We then created a loss tensor based on the *negative* of the model output to the image tensor for the layer and channel of interest. Using the negative of the model output allows us to transform a gradient ascent problem into classic gradient descent. We additionally added L2 regularization to the image to penalize high values, as well as a total variation regularizer, which penalizes variance between neighboring pixels and has been shown to reduce the appearance of "checkerboard" artifacts [11]. Our loss function is thus as follows:

$$\mathcal{L}(x, F) = -\frac{1}{m}\sum_{i=1}^{m}(F^{[i]}(x)) + \lambda_1\|x\|_2^2 + \lambda_2 \sum_{i,j}\sqrt{(x_{i,j+1} - x_{i,j})^2 + (x_{i+1,j} - x_{i,j})^2} \quad (1)$$

where F represents the feature activation, x is the input image where $x \in R^{H\times W}$, λ_1 is the regularization constant for the L2 regularization and λ_2 is the regularization constant for the total variation loss. We set $\lambda_1 = 0.0001$ and used a default λ_2 of 1, though we experimented with parametrically varying this regularization hyperparameter. We could then compute the gradients with respect to the image in order to iteratively optimize, for which we used Adam optimization with a learning rate of 0.05 [12].

2) Preprocessing: To improve robustness to transformations we preprocessed the image, using the exact preprocessing steps outlined in [13], namely

- Padding by 12 pixels to avoid edge artifacts
- Jittering by 8 pixels
- Randomly scaling by a factor of one of [0.9, 0.92, 0.94, 0.96, 0.98, 1.0, 1.02, 1.04, 1.06, 1.08, 1.1].
- Rotating by an angle of one of [-10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10] degrees
- Jittering for a second time for 4 pixels

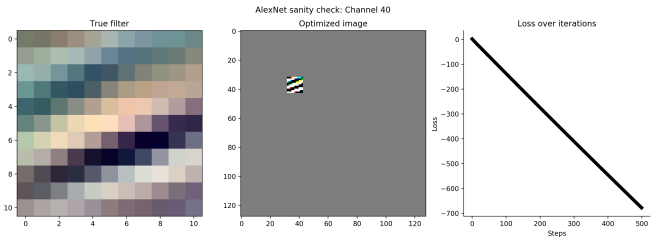

Fig. 1: Image optimization for one unit in AlexNet layer Conv1

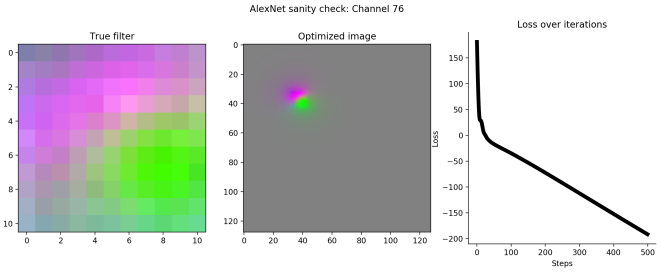

Fig. 2: Image optimization with total variation loss for one unit in AlexNet layer Conv1

IV. Results

A. Benchmarks

While it is difficult to determine a baseline for a target such as feature visualization, we can at least verify that we have implemented gradient ascent correctly. For the first convolutional layer of AlexNet, there are no nonlinearities so there is a closed form solution for the true filter of each unit (and the filters themselves are even included in the original AlexNet paper [4]). As such, we can use optimize for units in the first convolutional layer of AlexNet as a sanity check for the implementation. We see that we can return the original filter and that the loss decreases linearly if the total variation regularization constraint is removed (see Figure 1). Similarly, we return a close approximation of the filter and monotonically decreasing loss when total variation regularization is introduced (see Figure 2).

Additionally, while unfortunately extremely subjective, we can also attempt to verify that our implementation can produce images for later layers that are roughly as "vivid" and complex as state of the art work using similar approaches for other networks [9], which we believe to be the case (see Figure 3 for an example).

B. Comparisons within a network

Our initial experiments focus on the stochasticity within the process as well as the impact of design choices when optimizing for the same channel, layer and network.

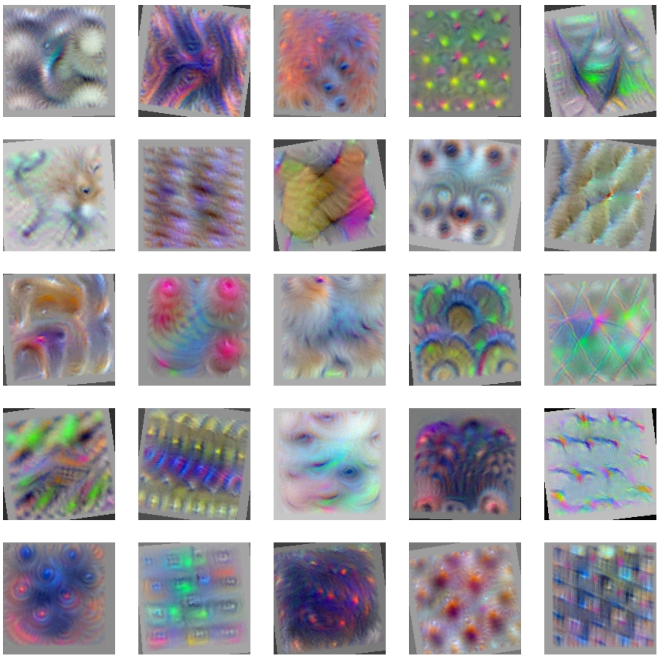

Fig. 3: Example optimized images for 25 channels in AlexNet conv5

1) Reliability across repeated optimizations: To begin with we wanted to determine how similar the optimized images would be across iterations, i.e. repeatedly optimizing for the same channel, layer and network with the exact same parameters but different random noise initializations. To test this, we ran 25 optimizations for channel 6 in layer conv5 of AlexNet. Qualitatively, the results are very similar (see Figure 4a). However, one issue we grappled without throughout this project was how to quantitatively evaluate the images, which we see as an important future area of exploration. For this study, we decided to leverage the convolutional neural networks at the heart of this project and run the optimized images back through ImageNet-trained AlexNet. We could then look at the output of the fc8 layer and compare the logits across repeats, both visually and by calculating Spearman's rank correlation coefficient (ρ). We also applied a softmax function to the logits as follows:

$$\phi_i = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}} \tag{2}$$

where ϕ_i represents the resulting probability for that particular class, x represents the logits and $k = 1000$ for the number of classes in ImageNet. This allowed us to determine the top 1 and top 5 classification labels for the images.

We found that the classification label was reassuringly similar across repetitions, with all 25 iterations for channel 6 in layer conv5 being classified as a "tick".

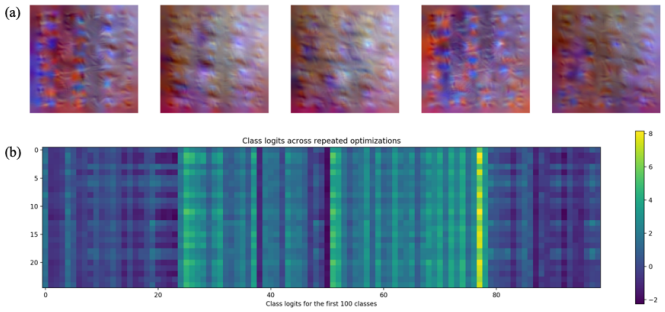

Fig. 4: (a) Five examples of repeated optimizations of a channel in AlexNet conv5 (b) Class logits for the first 100 classes across repeated optimizations

Additionally, of the 125 top 5 labels (25 repeats x 5 labels), only 14 were unique to a repetition and 89% were shared across the repetitions. Looking back at the logits, we can clearly see the similarity between repeats across a range of classes and not just the top few labels (Figure 4b). Furthermore, taking the first two optimizations as a sample, we find that the rank correlation coefficient of logits is significant ($\rho = 0.11; p < .001$).

2) Effects of preprocessing: We then examined the effects of preprocessing on our results. Preprocessing the image plays a critical role in whether the optimized images are vivid, colorful and meaningful to the human eye (though whether this is a desiratum depends on your end goal). This is particularly true if total variation loss also is not included (see Figure 5).

We also directly compared an image optimized including total variation loss for a channel in conv4 of AlexNet with and without preprocessing (see Figure 6). Interestingly, this image was classified by the network as a roundworm regardless of preprocessing, and while the rank correlation coefficient between two was smaller than for the exact repetitions, it was still significant ($\rho = 0.07; p = .02$), suggesting that at least CNNs 'see' these images similarly. It is an open question to what degree this is also true of the brain.

3) Effects of regularization: We also experimented with the effects of including the total variation loss term, scaling the regularization constant from 0 to 1 in steps of 0.1 (see Figure 7 for an example). Without total variation loss, the optimized images can often be dominated by high frequency patterns or "checkerboard" artifacts, which may be introduced in the gradient at least in part by strided convolutions and pooling operations [14]. Including the total variation regularization clearly

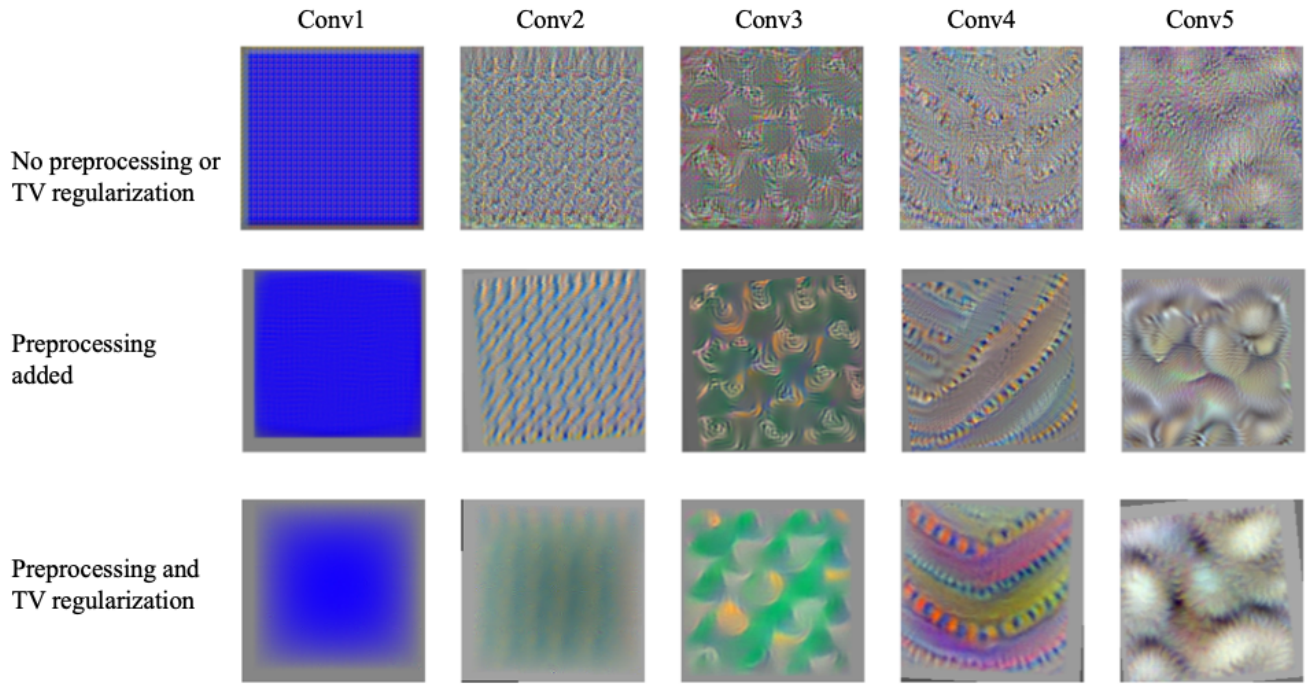

Fig. 5: Optimization for a channel in different layers of AlexNet with and without both preprocessing and total variation loss

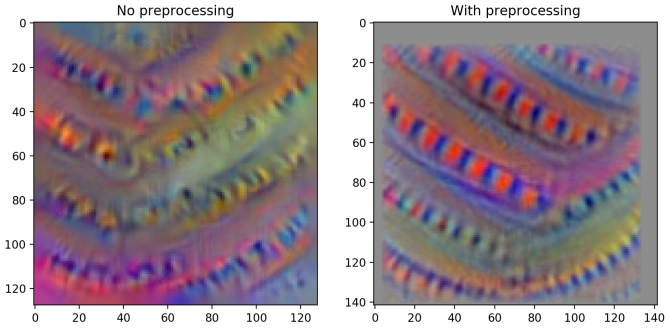

Fig. 6: Effects of preprocessing (AlexNet layer conv4)

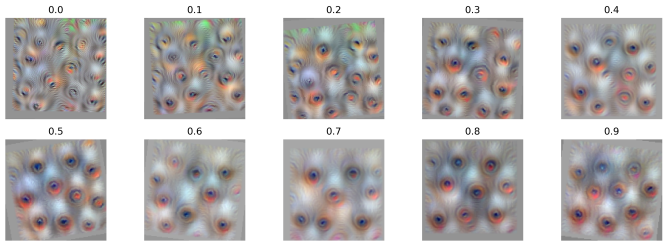

Fig. 7: Effects of different regularization constants for the total variation loss term

succeeds in reducing checkerboard artifacts and high spatial frequencies, effectively smoothing the image, though again the degree to which this is desired is an open question, as some high frequency structure and edges may be important for the integrity of the image.

C. Comparisons across networks

We then looked to apply these techniques to a second network, TNN, which has twice as many convolutional layers as AlexNet (see Figure 8 for example TNN visualizations). While feature visualization for this network also produced color opponency and Gabor patch type filters for the earliest layer and complex object-like visualizations for the last layer, the progression from simple to complex moved more gradually through the intermediate layers than in AlexNet, a clear result of the additional convolutional layers in TNN. This may more closely mirror the gradual progression and build up of complexity within the human visual system, from the retina to the LGN to the cortical areas V1, V2, V3, V4, VO1, VO2 and then finally to VTC. However, much further work needs to be done to quantify these differences before we can even explore such a claim.

V. Discussion

This project looked to explore feature visualization through optimization for channels within convolutional

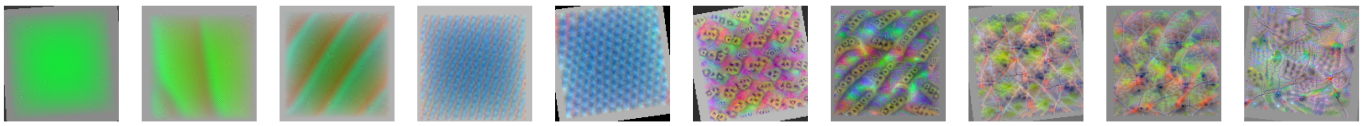

Fig. 8: Example optimized images for a channel in TNN across all ten convolutional layers (going from the first layer to the last, left to right)

neural networks. Focusing mainly on one simpler network, AlexNet, we probed the reliability of the method and the effects of different preprocessing and regularization choices. We found that image transformations and total variation regularization both play a crucial role in producing aesthetically pleasing visualizations that are less like adversarial examples and more like filters we'd expect to see in early layers of the visual system (color patches, Gabor-type filters) for early layers of the network and more complex object-like visualization for later layers of the network. However, despite these differences in how we perceive the images, they are consistent enough to be similarly classified by a network, and interesting open question is which version or optimization approach will be most effective in eventually driving biological neuronal responses.

This method produces vibrant visualizations across networks, as we saw when applying the same approach to a second network, TNN. The visualizations for later layers in both networks appear to resemble real-world objects (see Figure 9; though this is of course highly subjective and speculative) and are highly complex. This is reassuring in that it suggests that these networks are forming somewhat sensible high-order representations. Furthermore, it is a requirement for using a variant of this method to probe neural responses in human category selective cortex, as we know that these regions prefer complex, high contrast images.

An issue throughout has been the difficulty of determining ground truth and what counts as success for these visualizations. We made an initial stab at least quantifying differences across visualizations but feel that in general this is a particularly unsatisfactory aspect of the work. Other researchers have proposed methods for quantifying other facets of these visualizations, such as interpretability [15], which we hope to incorporate in the future.

Finally, we would particularly like to examine more specialized networks, such as FaceNet [16], which was part of the original scope of this project. While we were able to both construct a model graph and restore the weights from a checkpoint for this model, we ran into a continuous stream of errors trying to combine them both and had to jettison that aspect of the project for now (our most recent attempts to implement the method for FaceNet are included in the submitted code). However, we are still extremely interested in how different, more specialized tasks will change the features, particularly in early layers, and plan to pursue this in future work.

VI. Contributions

As this is a one-person project, I am responsible for all contributions to this report. However, note the theoretical inspiration from the cited references as well as the use of python packages (also included in the references) and several base functions which were originally written by other lab members and are marked as such in the code comments. The code can be found at https://drive.google.com/file/d/1v27OumUYwpdZ_Qn34SK-fC_Z9tZaSG1B/view?usp=sharing.

References

[1] Bashivan, P., Kar, K., DiCarlo, J. J. *Neural population control via deep image synthesis.* Science, 364(6439), 2019.

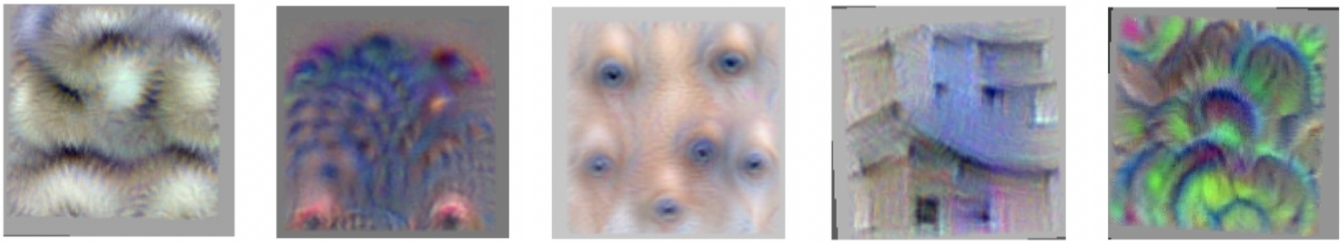

Fig. 9: A selection of conceptually evocative images from layer 5 of AlexNet

[2] Ponce, C. R., Xiao, W., Schade, P. F., Hartmann, T. S., Kreiman, G., Livingstone, M. S. *Evolving images for visual neurons using a deep generative network reveals coding principles and neuronal preferences.* Cell, 177(4), 999-1009, 2019.

[3] Walker, E. Y., Sinz, F. H., Cobos, E., Muhammad, T., Froudarakis, E., ... Tolias, A. S. *Inception loops discover what excites neurons most using deep predictive models.* Nature neuroscience, 1-6, 2019.

[4] Krizhevsky, A., Sutskever, I., Hinton, G. E. *Imagenet classification with deep convolutional neural networks.* In Advances in neural information processing systems (pp. 1097-1105), 2012

[5] Nayebi, A., Bear, D., Kubilius, J., Kar, K., Ganguli, S., Sussillo, D., ... Yamins, D. L. *Task-Driven convolutional recurrent models of the visual system.* In Advances in Neural Information Processing Systems (pp. 5290-5301), 2018

[6] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., Fei-Fei, L. *Imagenet: A large-scale hierarchical image database.* In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255). Ieee.

[7] Erhan, D., Bengio, Y., Courville, A., Vincent, P. *Visualizing higher-layer features of a deep network.* University of Montreal, 1341(3), 1, 2009.

[8] Mordvintsev, A., Olah, C., Tyka, M. *Deepdream-a code example for visualizing neural networks.* Google Research, 2(5), 2015.

[9] Olah, C., Mordvintsev, A., Schubert, L. *Feature visualization.* Distill, 2(11), e7, 2017

[10] Nguyen, A., Dosovitskiy, A., Yosinski, J., Brox, T., Clune, J. *Synthesizing the preferred inputs for neurons in neural networks via deep generator networks.* In Advances in Neural Information Processing Systems (pp. 3387-3395), 2016.

[11] Mahendran, A., Vedaldi, A. *Understanding deep image representations by inverting them.* In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 5188-5196), 2015.

[12] Kingma, D. P., Ba, J. *Adam: A method for stochastic optimization.* arXiv preprint arXiv:1412.6980, 2014.

[13] `https://distill.pub/2017/feature-visualization/appendix/`

[14] Odena, A., Dumoulin, V., Olah, C. *Deconvolution and checkerboard artifacts.* Distill, 1(10), e3, 2016.

[15] Bau, D., Zhou, B., Khosla, A., Oliva, A., Torralba, A. *Network dissection: Quantifying interpretability of deep visual representations.* In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 6541-6549), 2017.

[16] Schroff, F., Kalenichenko, D., Philbin, J. *Facenet: A unified embedding for face recognition and clustering.* In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 815-823), 2015.

[17] Python packages used: `tensorflow`, `tfutils`, `numpy`, `matplotlib`, `ipdb`, `pprint`, `argparse`, `os`, `sys`, `facenet` (https://github.com/davidsandberg/facenet)