1. **Introduction**

The code can be used to load pre-trained computer vision models to make prediction, feature extraction and fine-tuning. The inputs for the function should be raw images, or you could modify the code and let it take in RGB arrays. The output for each image would be one long vector (size depends on pre-trained models or self-defined) describing the features of the image.

Most of the codes are based on <https://keras.io/applications/>, refer to the website for more detailed information.

To run the code, you should have keras and tensorflow (or theano, CNTK) for python installed.

1. **Coding explanation**

Take ResNet50 for example. Firstly, the input image will be converted to a 3-dimensional vector, with shape 224x224x3. 224 is the default input width and height for ResNet50, images of larger shape can be resized to 224x224. The last dimension is the color channel, as colorful images are represented in RGB 3 channels. Each value in the vector represents the RGB-value of the image at specific location point and channel.

Next, this 3-dimensional vector will be pre-processed, and then sent into the ResNet model. The output will be a 1-dimensional vector with 4096 values, representing the features of this image.

Since the CNN models are deep and have many layers, you could also extract the features in the intermediate layers, by simply specifying the layer\_name variable in the code. If not specified, by default, the last layer before softmax classification will be used, which is the layer closest to the output layer. All layer\_name can be found via “model.summary()” command in python.

1. **Example demo**

16 residual heat maps are provided in plots folder for you to try out. Running the feature\_extraction.py will generate the feature outputs for these 16 plots. There are 4 pre-trained models defined in the code. More are available in <https://keras.io/applications/>.