You are provided with Jupyter notebook: 5.2-using-convnets-with-small-datasets.ipynb, 5.4-visualizing-what-convnets-learn00.ipynb and chapter09\_visualizing \_filters.ipynb and visualizing\_what\_convnets\_learn.ipynb created by Francois Chollet, author of “Deep Learning with Python”. The first two files are from the 1st Edition and the 3rd file is from the 2nd Edition of the book. The last file is from Keras.io examples: <https://keras.io/examples/vision/visualizing_what_convnets_learn.ipynb>. Please save the unmodified copies of those Jupyter notebooks so that you could compare Chollet’s (original) results with your own. These notebooks are also described in the class notes. Please create HTML images of the original notebooks for easy browsing.

Before you start running your Jupyter notebook, please install Python packages: opencv-python. If you are running your experiments on an AWS instance make sure that you stop the instance the moment you are done with your calculations. You can transfer your Jupyter notebooks from AWS to your local machine using scp command and continue examining results by running a local notebook.

Programs in this assignment are not exceedingly long and you might be able to run them on your own machine if you have a GPU card. Other way to run these programs is in Google Colab. Once you are in the Colab, change your runtime environment to GPU.

**Problem 1.** From Google/Images fetch an image of a sparrow and an image of a slightly cloud sky. Use OpenCV Python API to blend those two images. Experiment with several weights assigned to one and the other image and find the most natural looking image. Save your images. Do not sweat it out. Five tries for different weights is sufficient. Try subtracting the sparrow from the blended image using one of OpenCV routines.Present you working code and generated images. It might be convenient if you add the code and the results to your Jupyter notebook. If you insist you can submit a separate script.

**(25%)**

**Problem 2**. Create a CNN with an architecture similar or identical to the one constructed at the beginning of attached Jupyter notebook 5.2-using-convnets-with-small-datasets.ipynb. Train your network on a small portion of Kaggle dogs-vs-cats dataset. A zip file with that dataset is provided in the folder for week 5 on the class site. Use the same counts (1000, 500, 500) of images for the train, validation as test portion of the dataset as used in the notebook. Select images for all three subsets: train, validate and test, randomly from the big train set. Prove that your model work by displaying training and validation accuracy vs epoch number. Next use EarclyStoping callback ModelCheckpoint callback to save the best model, i.e. the model with the smallest overfitting. Save your “best” model. We will use it in the next problem/

**(25%)**

## Problem 3. Load the model saved in the previous problem and start new analysis similar to the one presented in the section “Visualizing intermediate activations” of the attached Jupyter notebook 5.4-visualizing-what-convnets-learn00.ipynb. Fetch a jpg of a Chinese New Year Dragon’s head from Google images, trim it to size 150x150 pixels using OpenCV and repeat the analysis in the notebook. Dragon is not a cat but the results should be close to what we have seen with cats. Visualize intermediate activations in the first convnet layer, for the channels with indexes 2 and 28. You are demonstrating that the code works. We want to see activations for some other layers. Please select two other convolutional layers and one of MaxPooling layer and generate activation images for those layers as well.

**(25%)**

**Problem 4**. This time we want to find images that excite various filters the most. Start with the code in the attached Jupyter notebook chapter09\_visualizing\_filter.ipynb

Challet’scode in visualizing\_what\_convnets\_learn.ipynb works for ResNet50V2 network. Import model for that network from kears.application package. Display the list of layers and an image of the network architecture (One could certainly be found on the Internet. You are not asked to draw the architecture yourself. Modify that code to display images which maximize outputs of several filters in the imported NN. Display 8 or 16 such images for filters in each convolutional layer of the NN. This time we have different names and number of layers but the process should be the same. Please note that the layer name is a “global variable” in the supplied code. Save your notebook with all generated images.You can generate images as files on the operating system as well.

**(25%)**

Preferably, your main submission should be one Jupyter notebooks and its HTML images. Your notebook(s) should include all results, images, and comments. If you insist, you may submit more than one Jupyter notebook. For every notebook, please submit an HTML image.

Please, place your name on every document you submit. Please, describe every step of your work and present all intermediate and final results in your Jupyter notebook(s).

Please provide clear and full comments for all of important steps or changes you are making.

If your notebook(s) contain(s) excessively long outputs please copy a meaningful and illustrative number of initial and/or final lines and paste those in a markdown (comment) cell. Then, delete the long output(s).