**Problem Statement:**

* The 2019 GRiD challenge for students is to leverage a predefined data-set from Flipkart to enable ‘Vertical Classification’ using images.
* A dataset was provided which has images and a metadata file containing name of image and bounding box coordinates around the object of image.
* **Libraries used:-**

1. Keras
2. Tensor Flow
3. Pandas
4. Numpy
5. Open CV

* The model has been trained locally on a Lenovo Legion Y530, which has a Nvidia Graphics card belonging to the GeForce family, a 6GB 1060X.
* We sorted the data before training into the respective train / test directories by referring to the respective csv files
* The submitted model takes images as input and produces the coordinates of the bounding box.
* We used the simplest algorithm available for object detection;

a convolution neural network as base network to extract feature maps of images and over this we attached a fully connected layer which acts as a regression head to predict the bounding box values.

* Both CNN and the regression head are trained together with the mean square error loss function and adam optimizer.
* The loss values tend to converge as the epoch goes on.
* In the convolutional layer, we use dropout and batch normalisation for regularisation.
* We preprocessed the input, the size of every image is 128\*128\*3 .
* We load all the images according to the training.csv file from the training data set. We scale the pixel value of the images between 0 and 1 before feeding it to the net. We also scaled the bounding box values between 0 to 1 for regression.
* 10% of the training dataset was set apart for validation.
* We used Adam as our optimizer as:-

1. Parameters update are invariant to re-scaling of gradient
2. The step-size is approximatelybounded by the step-size hyper-parameter.
3. It doesn’t require stationary objective.
4. Naturally performs step size annealing.

* The convolutional neural network was built using the Functional API of Keras. The function ‘create\_cnn’ loops overs the filters which are provided as the arguments to create a set of convolutional, Relu activation, batch normalization and max pooling layers. Then we flatten the next layer, and add a fully-connected layer with batch normalization and dropout.
* The directory ‘module’ has all the helper functions as modules.
* The file ‘mixed\_training.py’ is used to train the network and store the model and its weights.
* We store our trained model in the form of a ‘Keras.json’ file.
* The Keras.json configuration uses the following parameters to load/store image format:

1. Image data format
2. Epsilon
3. Floatx
4. Backend module

* The weights are stored with a ‘.h5’ extension.
* The models are then stored in the “models” folder
* ‘predict.py’ loads the model using the ‘.h5’ and the ‘.json’ files. And finally predicts the bounding box values and writes it to a csv file present in ‘res’ folder.

**NOTE**: All the paths to files mentioned in ‘mixed training.py’ and ‘predict.py’ are hardcoded and need to be changed accordingly by the evaluator.