1. Feature engineering

Feature engineering is encorporated in the function *preprocess_data* which takes 2 parameters:

1. number of rows for the grid
2. number of columns for the grid

It uses two helper functions for CSV preprocessing: *add_labels* and *calculate_group_center*. *add_labels* has the responsibility of dividing the map of Croatia into zones(groups), therefore creating a new feature - label, which holds the information in which group a certain entry belongs.
Furthermore, *calculate_group_center* is responsible for calculating coordinates of the group center and adding them into the dataset as two new features - *center_longitude* and *center_latitude*.
This leaves us with the preprocessed CSV ready for the classification task at hand.

2. Data cleaning

Data cleaning is based on two main procedures: one used for detecting images we want to remove, and the other for removing said images.
First procedure is implemented in the function *list_files* and it preprocesses the images for the OCR algorithm. Most important transformation used is cropping to restrict the visual area on the part where the image source label is located (bottom right corner). After retrieving images with *list_files* second procecdure starts and removes all images that are not from Google.

3. Data transformation

Given that certain groups are highly underrepresented we implemented functions *find_nearest_group* and *replace_groups*.
find_nearest_group takes a certain group, in our case - the underrepresented group, and finds groups that are nearest to it. With information of underrepresented groups' neighbours, function *replace_groups* takes care of merging the underrepresented groups' images with their neighbours group.

Still the dataset at this point contains entries of every location and its UUID but for our model input we have to "unpack" them so that dataset contains entries of UUIDs for every image in the set. The task described earlier is implemented in the method *expand_dataset* which takes dataset as an argument and returns it "unpacked".

Last dataset transformation that is required is dataset balancing. Function *balance_data* accomplishes balancing with duplicating all of the groups' images until it reaches target group size which is passed as an argument to the function.

4. Data loading

Loading data in the model is handled by ImageDataGenerator class taken from Keras library. There are three ImageDataGenerator objects *train_generator* ,*validation_generator* and *test_generator*.
each created for each model phases training, validation and testing ,respectively. *train_generator* variable has additional responsibility of taking care of image augmentation during training.
Main reason for using image generators is segmentational loading of the dataset into the memory and applying image transformations.

5. Train-validation-test split

To split the data in three parts, Scikit-learn function *train_test_split* has been used two times.
First use splits the dataset into train and test, and the other use is spliting the train set from the first step into train and validation. That gives us the original data split into three parts used in the model training and inference process.

6. Pretrained model selection

Model selection is achieved via *get_model* method which accepts an input shape and model type as parameters.
Depending on which model type has been input as a parameter, the method will load and return the pretrained model from Keras with corresponding weights.

7. Model creation

To create the model that will be used, method *init_model* creates the input layer, retrieves the pretrained model via *get_model* and concatenates a custom top layer. The final output layer has 35 classes which is a lot less than what would be usually on a 11x11 grid, but the simple reason behind it is that some of the classes have been merged into one, and some classes that contained 0 data points have been removed.

8. Regularization

Method *add_regularization* accepts two parameters, model and regularizer. Regularization is then applied to each layer in the model that can be regularized.