We have built a **deep learning CNN - Regression model** to solve the task of localizing objects in an image. Convolutional Neural Networks have been harnessed to extract appropriate features from the training and test images. Due to limited computing capacity we had to restrain ourselves to work with images of size 128 x 128. Moreover, we couldn’t go further in our quest to use deeper convolutional batches of size greater than 256 because of the same reason.

**Approach**

**Preprocessing :**

1. We begin by converting the RGB images into grayscale images and then resizing them from their original size (480 x 640) to dimensions 128 x 128. These two steps were performed so as to suit our computing capacity.
2. The images are then normalized which further helped us out with faster processing of data.

**Data Augmentation:**

To make our model more robust we performed data augmentation as follows:

1. preparing duplicate files of the training images by randomly **flipping** the given fourteen thousand training images in 5 different ways (rotating by 90,180, 270 degrees and flipping them left to right and top to bottom).
2. preparing duplicate files of the training images by **cropping** them with random slicing options.

(**NOTE:** We also tried augmentation by randomly changing brightness of various images but the experiments didn’t brought out good results and hence it’s use was dropped.)

This way, we were able to increase our training images count from fourteen thousand to forty two thousand. After preprocessing these images we got images of dimensions 128 x 128 x 1 ( original size being 480 x 640 x 3 ).

Next, we tried various deep learning architectures to find the best model for our task. The following architecture is finally proposed.

**Model Architecture**

|  |  |
| --- | --- |
| Input Images | **128 x 128 x 1 (grayscale)** |
| Convolutional Layer  Activation  Batch Normalization  Max Pooling | **32, Kernel 3 x 3,**  **Relu**  **2 x 2** |
| Convolutional Layer  Activation Layer  Batch Normalization  Max Pooling | **64, Kernel 3 x 3**  **Relu**  **2 x 2** |
| Convolutional Layer  Activation Layer  Max Pooling | **128**  **Relu**  **2 x 2** |
| Convolutional Layer  Activation Layer  Max Pooling | **256**  **Relu**  **2 x 2** |
| Dense Layer  Activation Layer  Dropout | **2048**  **Relu**  **0.5** |
| Dense Layer  Activation Layer  Dropout | **2048**  **Relu**  **0.5** |
| Dense Layer (Output) | **4 (x1,x2,y1,y2)** |

**Adam optimizer** and **Mean Squared Error** as the loss function was used while training the model. The model was run for **110 epochs**. We were able to achieve an IoU score of 87.3133 on the public leaderboard.

**Tools and Libraries Used:**

1. **OpenCV (cv2)** for preprocessing images
2. **Keras** and it’s dependencies to build the Deep Learning Model
3. **Google Colaboratory,** for better computing power for free.