# Helping Autonomous Cars Recognise Street Signs

**Team Members :**
Vikram Nitin, Abhay Koushik, M. Avadesh

**Overview :**
Our solution used Convolutional Neural Networks (CNNs) as a binary classifier to decide whether a given image was a sign or not. We trained two networks, one to approximately locate the position of a sign inside a bigger image, and one to fine-tune the bounding box around the sign. We used Keras with Theano.

**Preparing the Training Set :**
We used a 'sliding window' technique to go through the given images. Using these, we prepared two training sets.
For the first set, we moved a window of fixed size (60x60) over the image. Whenever it overlapped with the given bounding box, it was labelled as a 1, otherwise 0.
For the second set, we moved a window of variable size around the given bounding box and resized these images to 60x60. These were all counted as positive samples. To get negative samples, we moved a window in fixed steps around the entire image, excluding the places where it overlapped with the given bounding box.

**Architecture :**
The same architecture was used for both the networks trained.
We used 4 convolutional layers (with 32, 32, 64 and 64 filters respectively). After every layer, a ReLU activation was applied, and after every two convolutional layers, a Max Pooling layer was applied. Sizes of the convolutional filters were 3x3, and the pooling window size was 2x2.
This was followed by two fully-connected layers of sizes 4096 and 256 (with a dropout of 0.5 after each), finally connected to a single output unit (with Sigmoid activation).
More details of the architecture can be found in *cnn.py*.

**Pre-Processing :**
We normalized each pixel of the training images by subtracting the mean and dividing by the variance pixel-wise. The mean and variance were saved so that they could be used with test images. Batch Normalization was applied to the input images to the CNN. Random horizontal flips were applied to the input images.

**Training :**
The two networks were trained one after another. SGD with Nesterov momentum was used for

both. The weights were saved after each epoch and the weights with the best validation accuracy were chosen.

**Prediction :**

First, we moved a sliding window of fixed size around the whole image, and used the first trained network to predict the maximum output activation among these windows. Then this was used as a reference point.

Next, we moved a square window of variable size around this reference point, and used the second network to predict the maximum activation among all these positions and sizes.

Thus, the best position and size were found.