Pratheek D'Souza Rebello 2017CS10361 Part (d)

DATA AUGMENTATION

The images are of cars, trucks, frog etc. These images are not left-right orientation specific. A frog facing left is also a frog facing right.
I have therefore doubled the train data size by flipping all images along the vertical axis (i.e. a left right flip)
This greatly reduces the overfitting and helps the network improve feature detection that is orientation independent.
I obtained a ~3% improvement in test accuracy from 48% to 51% with this flipping.
Thus I have a total of 40000 images now.

FILTERS

SOBEL: from skimage package, I used the sobel filter for edge detection. See below for wikipedia source of Sobel filter. Gx finds vertical edges, Gy finds horizontal edges. Sobel filter is giving a 50+% test accuracy. I have selected it

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A}$$

where $*$ here denotes the 2-dimensional signal processing convolution operation.

Since the Sobel kernels can be decomposed as the products of an averaging and a differentiation kernel, they compute the gradient with smoothing. For example, $\mathbf{G}_x$ can be written as

$$\mathbf{G}_x = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \left(\begin{bmatrix} -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \right) \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} -1 \\ 0 \\ +1 \end{bmatrix} * \left(\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \mathbf{A} \right)$$

The *x*-coordinate is defined here as increasing in the "right"-direction, and the *y*-coordinate is defined as increasing in the "down"-direction. At each point in the image, the resulting gradient approximations can be combined to give the gradient magnitude, using:

$$\mathbf{G} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2}$$

GABOR: from skimage package, I used the Gabor filter that returns real and imaginary parts. The filter is taking 0.04s per image, which translates to 800s for the dataset. This is not feasible for our current data.

CANNY: from skiimagu package, I used canny filter. This also detects edges and returns an array of zeros and ones. Ones wherever an edge is present.
This filter doesn't seem to be a good filter to use. Due to the binary nature of 0s and 1s, the train data is easily overfitting to the model.
I obtained a 100% accuracy on training data within 2000 iterations, while the test data is hovering at around 36%.

FINAL FEAUTURE CREATION

I took the (40000,1024) images, and reshaped to (40000,32,32). I then applied a sobel filter to each filter, which outputs a (32,32) image, reshaped to (1024).
Finally I concat the image and its sobel output to create a (40000,2048) vector as my training data. *Anything bigger than this is giving an out of memory error.*