Calculator

Group Members: Aadeesh Jain, Riya Bhalla, Sujal Singh

Abstract

In this project we proposed a convolutional neural network approach towards handwritten digit and basic operator recognition by extracting features and converting patterns into categories. Further we worked on implementing a basic calculator model, that can evaluate an expression in a given image by first separating out numbers and operators using contour plotting and then predicting their class using the model. Experiments are performed on the 333,895 images belonging to 16 different classes. Major libraries used for implementation are opencv, numpy and tensorflow.

Working

1. Image Pre-Processing

The 333,895 images were divided in a 1:4 ratio between validation and training set. These images were loaded at a target size of (40,40) and in grayscale mode. These images were also rescaled by a factor of 1./255 so that pixel values are not too high for the model to process.

2. The Architecture

The architecture of our neural network contains four learned layers, two convolutional and two fully connected. The activation function we used for the first three layers (two convolutional and one fully connected) is, rectified linear unit (ReLu), as it trains much faster and helps us to deal with the problem of vanishing gradients. We also used max pooling layers with kernel size two for training our model in order to reduce overfitting. Our network maximizes the multinomial logistic regression objective, which is

equivalent to maximizing the average across training cases of the log-probability of the correct label under the prediction distribution.
The first convolutional layer is fed with 40×40×1 input images and consists of 32 kernels of size 3x3 using a stride of 1 pixel (this is the distance between the receptive field centers of neighboring neurons in a kernel map). The second convolutional layer is fed with the output of the previous layer and filters it with 64 kernels of size 3x3. Then we flatten the output channels and pass it to the fully connected layer that has 128 neurons and then finally to our output layer that uses a 16-way softmax which produces a distribution over the 16 class labels.

3. Optimizer

We trained our model using Adam Optimizer with a learning rate of 0.0005. The update rule for weight w was :

$$w_t = w_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Where w is model weights, eta is the step size.

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \qquad m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \qquad v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

Where m and v are moving averages, g is gradient on current mini-batch, and betas — newly introduced hyper-parameters of the algorithm.

4. Detection

Firstly the image was converted into grayscale using cvtColor function and then to sharpen the image gaussian filter was applied on the image.This was then followed by canny edge detection method for identifying the edges of the numbers and symbols and then finally plotting contours.
All these were applied using python openCV library functions.

Limitations

As our main focus was on recognizing digits and the operation symbols we didn’t code for the bodmas rule and so multiplication and division can’t be carried out simultaneously with others.

Dataset

The dataset can be found here .