Hand-written Doodle recognition implementation

Problem Statement: Considering the character recognition world problem, which is referring to the MNIST image recognition dataset. The model is trained with the training part and tested in the doodle and demo parts. The demo part is tested and providing good accuracy, so we need to work on the hand-written doodle recognition to improve the accuracy.

Failed Experiments Conducted and Explanation:

1. ***Using Tanh activation function instead of Sigmoid activation function***: The non-linear activation function is changed to Tanh (hyperbolic tangent) activation function. The range of tanh function is from (-1 to 1) and tanh is also sigmoidal (s-shaped).
   The Tanh activation function is resulting in reducing the model accuracy to **15%** approx.

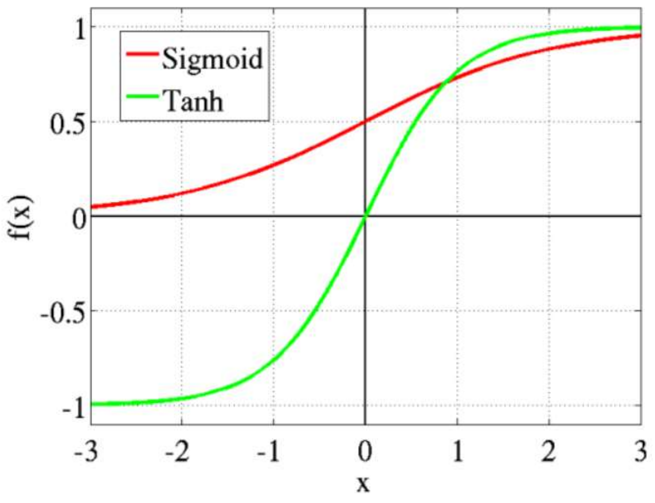


Fig: tanh v/s Logistic Sigmoid

2. ***Using ReLU activation function instead of Sigmoid activation function***: The non-linear activation function is changed to ReLU (Rectifier Linear Unit) activation function. The ReLU is half rectified (from bottom), f(z) is zero when z is less than zero, and f(z) is equal to z when z is above or equal to zero. Its range is [0 to infinity).
   The ReLU activation function is resulting in reducing the model accuracy to **10%** approx.

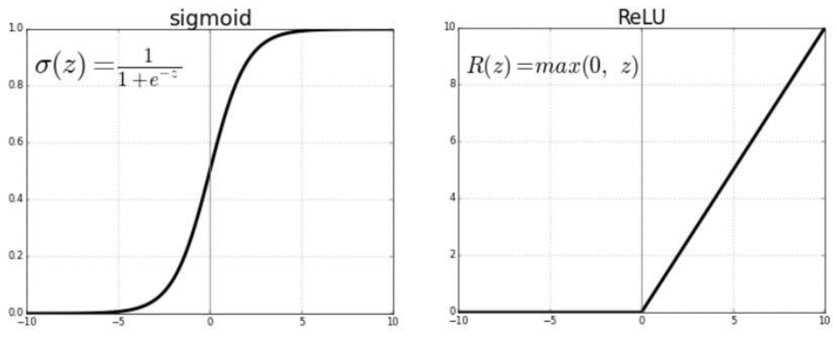


Fig: ReLU v/s Logistic Sigmoid

3. ***Changing to make Doodle more like the test set***: Making these aesthetic changes like blurring the doodle line, thickness of the doodle line, and restricting the user to draw smaller doodle was not enhancing the accuracy of the doodle recognition.
    a. The **DOODLE_BLUR** was resulting in blurring the edges making the image look like the same pixel representation as in the training and test set. Changing the blur factor from a range of [0,5] was not improving doodle recognition.
    b. The **DOODLE_THICK** was resulting in a thickness of doodle lines and changing it from the range of [12,20] was also not improving recognition. Mostly conducted to bring the doodle images close to training images.
    c. Same case with **ZOOMFACTOR** which refers to the size of canvas available for images, ranging from [5,9] was not improving doodle recognition. This is used to restrict the canvas available for the user to draw a doodle.

4. ***Changing the network itself***: These involve making the changes to the neural network architecture itself like changing or turning off training, changing the number of hidden nodes, or changing the randomWeight function return values.
    a. Updating the **do_training** to **false** leads to random recognition of the images and as a result, the test accuracy is also drastically reduced in the demo part. This is because the network is seeing that test set of images the first time without getting trained to see similar data. Updating the **TRAINPERSTEP = 1**, or **TRAINPERSTEP = 50** leads to improve the recognition but not that effective. This is because the training is carried out at a slower rate.
    b. Updating the number of hidden nodes, **nohidden = 50** leads training accuracy of approx. **30%** in the initial exemplars and then increase at a fast rate seeing some 100 exemplars. Updating **nohidden = 1** leads training accuracy of approx. **10%** in the initial exemplars and then gradually increase at a very slow rate seeing some 100 exemplars. Updating **nohidden = 200** leads training accuracy of approx. **40%** in the initial exemplars and then gradually increase at a fast rate seeing some 100 exemplars.
    c. Updating the **randomWeight** to return **0.5 (constant)** leads to a drastic bad score of approx. **10%** resulting in always classifying both the doodle and demo as 0 irrespective of the input provided. Updating the **randomWeight** to return **0** leads to a bad score of approx. **13%** resulting in mostly classifying both the doodle and demo wrong.

5. ***AB save and restore to store weights***: The weights computed in the training of the neural network were unable to save. Facing issues in saving (**AB.saveData**) the training weights and then restoring (**AB.restoreData**) back when guessing the hand-written doodle recognition. Using the save and restore method leads to some memory issues unable to traceback in it. Even referred AB.runloggedin and AB.myuserid to debug the save and restore feature.

Implemented Convolutional Neural Network (CNN) and Explanation:

- CNN’s are regularized versions of multilayer perceptrons.
- Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer.
- Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function.
- CNN’s take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns.

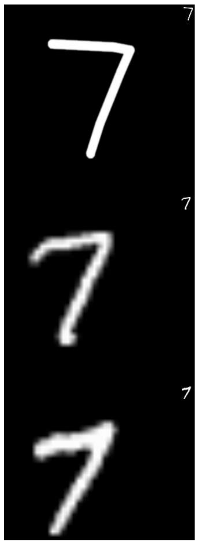
The CNN is implemented to classify the hand-written doddle images. This network architecture is implemented from the WebCNN [1] framework by Adam Smith.

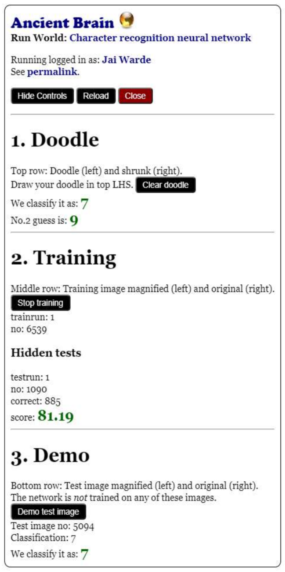
- There are certain layers with max-pooling with variable size and stride.
- The convolutional layers with square kernels, the stride of 1, no zero paddings, and ReLU activation function
- The neural network is having multilayer perceptrons, and having tanh and linear activation functions applied at some layers.
- The final layer is applied with the soft-max classifier support.

Making this implementation in Ancient-Brain is typical on changing the canvas function availability in the P5 library. The canvas is cropped and formatted to get the object specified and masked as per the convolutional neural network requirement.

- The **DOODLE_BLUR** is set to 1, as the image with heavily blurred edges will lead to inaccurate classification. So, having this value we are having a better classification of the doodle images.
- The **DOODLE_THICK** is set to 16, as initially defaulted to bring the doodle images look-alike to the training set of images. But now we have the doodle recognition carried through CNN.
- The other existing network architecture parameters are not altered.
- The weights are no longer required to be stored as referring to the pre-defined weights for the CNN doodle image classification.
- There are a couple of functions written to set up the canvas as per the existing doodle canvas structure.
- The CNN pre-defined JSON weight file is loaded and new model implementation is carried in response.
- Alongside, training the previous model too for having the model accuracy maintained in the test or demo dataset.
- The doodle image pixels are even masked and formatted to feed the CNN model, as can be checked in the guessDoodle() function.
- Additional efforts are made to add the AI cool GIF in the background and setting up a Robotic voice customize the message as an initial sound effect.

Refer to the below Screenshot for showing doodle recognition of number 7





Ancient Brain
Run World: Character recognition neural network

Running logged in as: **Jai Warde**
See **permalink**.

1. Doodle

Top row: Doodle (left) and shrunk (right).
Draw your doodle in top LHS.
We classify it as: 7
No.2 guess is: 9

2. Training

Middle row: Training image magnified (left) and original (right).

trainrun: 1
no: 6539

Hidden tests

testrun: 1
no: 1090
correct: 885
score: 81.19

3. Demo

Bottom row: Test image magnified (left) and original (right).
The network is *not* trained on any of these images.

Test image no: 5094
Classification: 7
We classify it as: 7

The above implementation is classifying the hand-written doodle images with decent accuracy and the demo is maintained to work on the same accuracy as before the problem.

References:

[1] Smith, Adam. "WebCNN". GitHub, 1 Feb. 2017, https://github.com/DenseInL2/webcnn