Task 5 : Custom Activation and Loss Functions

Sanjeev Narayanan

10th March 2018

Activation Function

The activation function [1] of a node defines the output of that node given an input or set of inputs. In biologically inspired neural networks, the activation function is usually an abstraction representing the rate of action potential firing in the cell. In its simplest form, this function is binary—that is, either the neuron is firing or not. Activation functions have the following properties:

- Nonlinear
- Continuously differentiable
- Finite Range
- Monotonic and have a smooth monotonic derivative
- Approximates Identity near origin

In this experiment, the following activations were tested (not available in Keras)

- $Gaussian(X)$
- $Softplus$ - $f(x) = \ln(1 + e^x)$ [2]
- $Swish$ - $f(x) = x(\frac{1}{1+e^(\beta x)})$

Loss Functions

Sometimes referred to as the cost function or error function , the loss function is a function that maps values of one or more variables onto a real number intuitively representing some "cost" associated with those values. In this experiment, the following losses were used.

- Stock Loss [3] - Very similar to Mean squared error, but has a tweaking in the parameters
- Mean Square Error Log Loss [3] : $L(Y_t, Y_p) = (\ln(Y_t) - ln(Y_p))^2$
- Cubed Hinge Loss [4]: $\Sigma_j max(0, 0.5 - Y_t^j Y_p^j)^3$

Experiment

In this experiment, activation functions and loss functions are constructed and are required to be added to the keras library. There are two ways to do this. One, register the operation in a C++ code and build a python wrapper which is the public API that's used to create the op in Python. The gradients of the op are then computed and then the op is tested. The second and more commonly used method is adding a new layer to the model as a separate activation and loss [5]. Since the loss function can't be accessed from a local directory as it is a part of *model.compile* attribute, we need to add the custom loss python script to the keras library.

The MNIST dataset [6] was used to test the custom functions. The convolution neural network used in the test has a

- Convolution layer of 96 filters of size 11×11 with activation to be specified.
- A 2×2 Max pool layer
- Convolution layer of 48 filters of size 5×5 with activation to be specified.
- A 2×2 Max pool layer
- 20% dropout and flatten layer
- 3 Dense layers with sizes 256,128 and 10 with activation to be specified
- Loss layer to be specified and Adam Optimizer

The results obtained are displayed below. What was observed was that when the custom activations are applied to custom losses, the accuracy is very poor. However, when the custom losses are applied to Rectified Linear Unit activation, the results are very close to the standard values and that which was obtained in Task 2. Probably, deeper understanding regarding the layers of activation is required in order to achieve better accuracy. Nonetheless, the loss functions used int he experiment are effective in minimizing losses and yield highly accurate results.

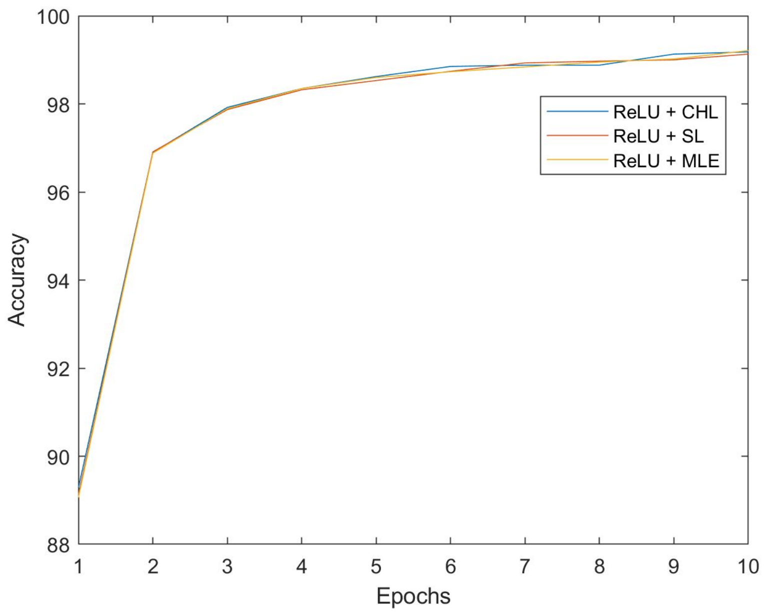

Figure 1: Plot of Accuracy vs Epoch for the loss functions with ReLU Activation

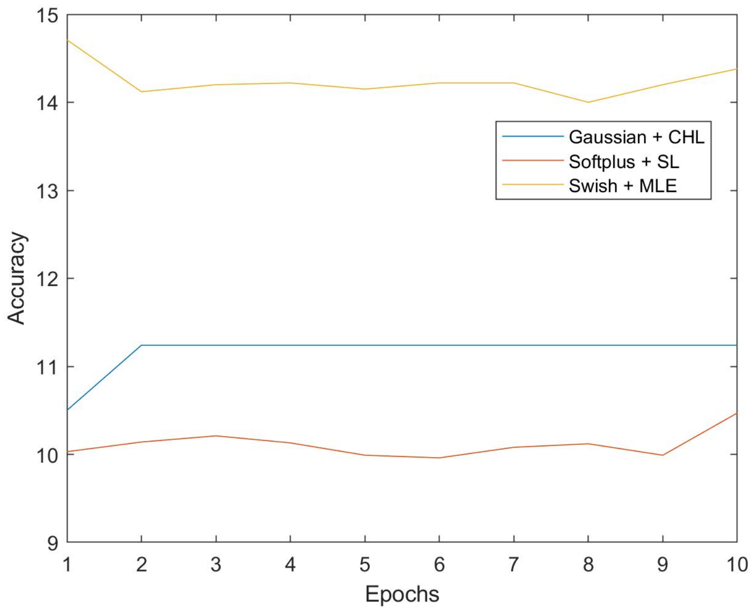

Figure 2: Plot of Accuracy vs Epoch for the loss functions with activation functions

References

[1] Antonio Gulli and Sujit Pal. *Deep Learning with Keras*. Packt Publishing Ltd, 2017.

[2] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.

[3] CD Pilon. Probabilistic programming and bayesian methods for hackers, 2015.

[4] Katarzyna Janocha and Wojciech Marian Czarnecki. On loss functions for deep neural networks in classification. *arXiv preprint arXiv:1702.05659*, 2017.

[5] François Chollet et al. Keras, 2015.

[6] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.