[redacted]

Question 2: Lost in the closet (Classification)

Problem: (a) Given the problem, what is the most appropriate loss function to use?

Error function aka loss function is used to estimate the loss of the model so that the weights can be updated to reduce the loss on the next evaluation. Choice of loss function must match the framing of the specific predictive modelling problems, such as classification or regression.

Neural networks use stochastic gradient descent to train data and typically seek to minimize the error. Therefore, the objective function is often referred to as a cost function or a loss function and the value calculated by the loss function is referred to as simply "loss".

Cross Entropy Loss Function

- **What?**
  Cross-entropy (also log loss) is a measure from the field of information theory, building upon entropy and generally calculating the difference between two probability distributions (between 0 and 1). Cross-entropy can be thought to calculate the total entropy between the distributions.
- **Why?**
  Cross-entropy is widely used as a loss function when optimizing classification models because it minimizes the distance between two probability distributions - predicted and actual.
  - **Concept/Math**
    In binary classification, where the number of classes M equals 2, cross-entropy can be calculated as:
    $-y(log(p) + (1-y)\,log(1-P))$

    If M>2 (i.e., multiclass classification), we calculate a separate loss for each class label per observation and sum the result. (refer the below written formula):

    $-\sum_{c=1}^{M} y(0,c)\,log(p_{0,c})$

Problem: (b) Create and train a Convolutional Neural Network?

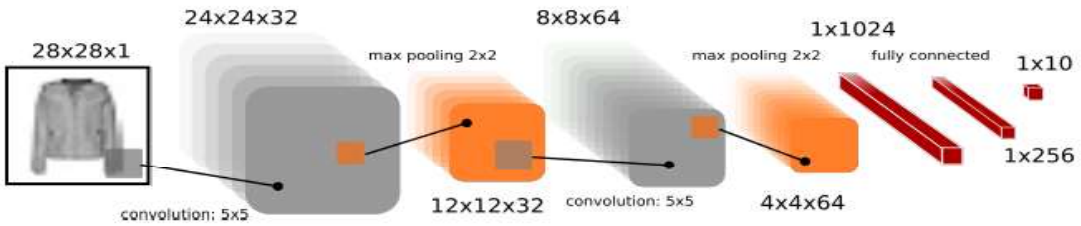
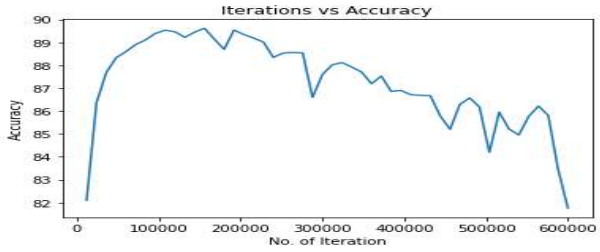
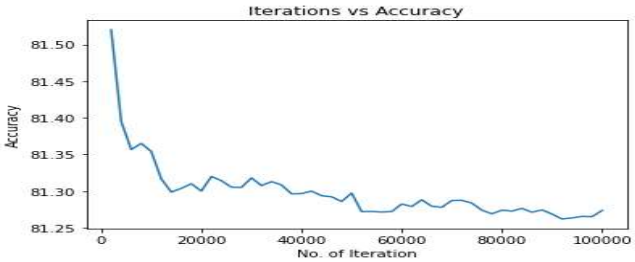


Figure: 9.CNN Model Architecture

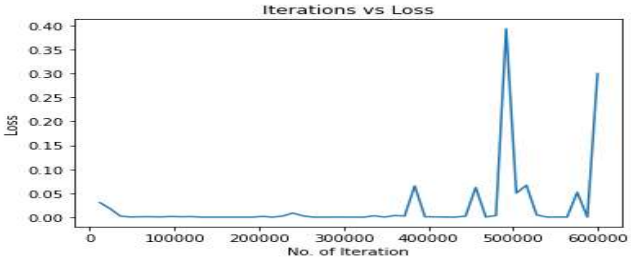
- **Final Train Accuracy = 82.755 %**



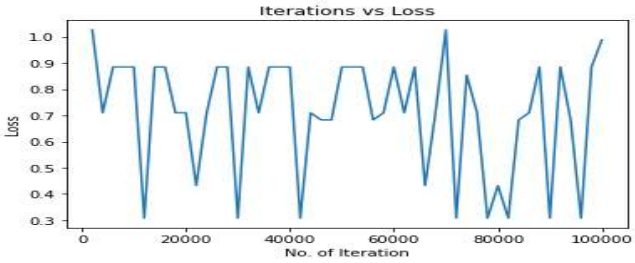
- **Final Test Accuracy = 81.27%**



- **Final Train Loss = 0.2988**



- **Final Test Loss = 0.9**



Discussion over Losses during the epochs

Loss value varies inversely to the Accuracy value of the Training dataset. As the accuracy of the training data increases the loss value decrease and vice versa.

Losses obtained over the simulation.

`[tensor(0.0303), tensor(0.0176), tensor(0.0024), tensor(0.0003), tensor(0.0008), tensor(0.0008) ... . . .. tensor(3.5331e-05), tensor(0.2998)]`

Problem: (c) Now, change the activation function to Tanh, Sigmoid and ELU. Provide only the final classification accuracy. Keeping ReLU, use 5 different learning rates: 0.001, 0.1, 0.5, 1, 10. What do you observe? Explain.

- **For ReLU (0.001, 0.1, 0.5, 1, 10)**

Learning Rate Value	Training Accuracy	Test Accuracy	Overall Accuracy
0.001	94.7%	90.11%	90%
0.1	81.75%	80%	81%
0.5	10.97%	9.8%	10%
1	10.5%	9.7%	10%
10	10.01%	9.45%	10%

Observation: Increasing the learning rate values varies the training accuracy negatively (hence, reduces them) resulting in giving the overall accuracy of around 10% almost every time when the learning rate value was taken greater than 0.1

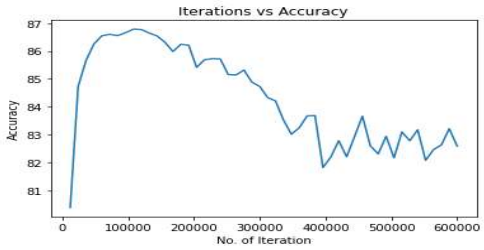
- **For Tanh**
  Final Classification Accuracy = 78%

- **For Sigmoid.**
  Final Classification Accuracy = 8.9%

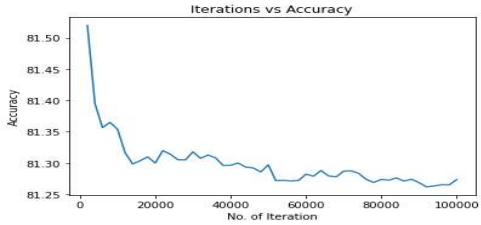
- **For ELU.**
  Final Classification Accuracy = 9.6%

Problem: (d) Now, add a dropout of 0.3 rate on the second fully connected layer. What is the impact of dropout on the performance? Provide the plot for training and test after each epoch. What happens if you decrease or increase the dropout rate?

- Training accuracy = 82.5850 %



- Test Accuracy = 81.20 %



- Training Loss = 0.0043
- **Impact of dropout on the System's Performance**.
  With the addition of dropout, the overall accuracy of the system increased and the training loss of the system decreases significantly.
- **Discussion (What happens if you decrease or increase the dropout rate?)**.
  Increasing the dropout value will going to decrease the system's accuracy over the period of time whereas the deceasing the dropout value reduces any chances of overfitting and increases the accuracy.