Machine Learning

Deep Learning Assignment 2

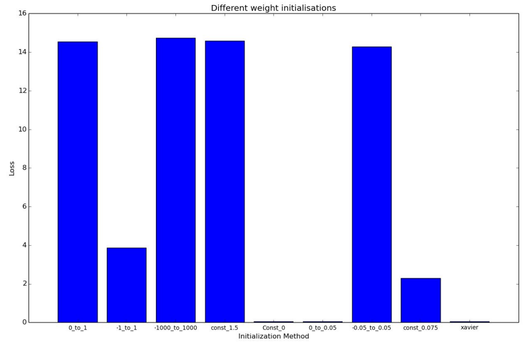
Mohit Sharma

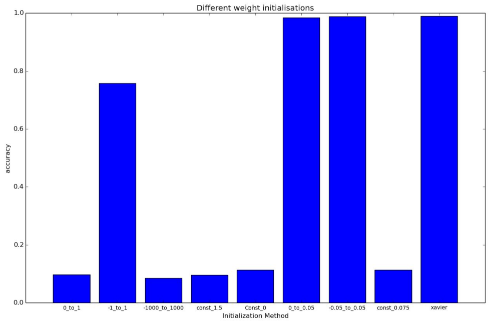
201505508

Q1. Train a neural network (CNN) on MNIST dataset with the following weight initializations:

1. All weights are either 0 or constant
2. All weights are random and random scale
3. Random with range [-1, 1]
4. Xavier weight initialization

Ans:





Observations:

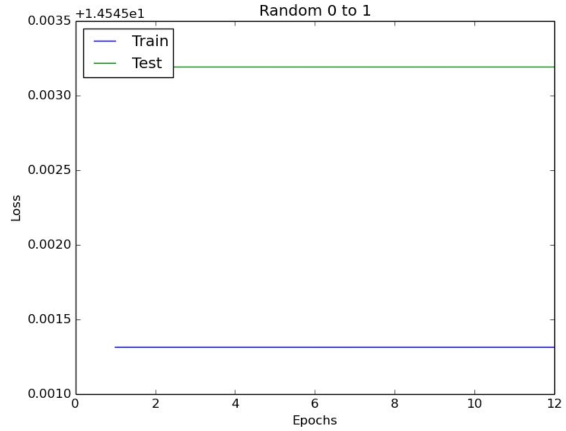
1. None of the initializations with constant weights (0, 1.5, 0.075) was moving towards convergence. There can be following reasons for this:
    a. If we start with the same weights, then all the connections/neurons will follow the same gradient, and will end up doing the same thing. So all will get updated with the same value resulting in no different learning by any of them.
    b. Neural networks can stuck in local minima, so it's a good idea to give them many different starting values. We can't do that if they all start with same value..
2. The initialization with weights between -1000 to 1000 didn't converge too. Possible reason can be:
    a. A small weights leads to very small gradient, so small inefficient steps. On the other hand, large weights saturates the activations, which again leads to small updates. Hence it is desirable to keep the weights in such a way that activation functions are in linear zone, so that gradients are optimal.
3. The initialization with weights between 0 to 1 didn't converge too. Possible reason can be:
    a. We didn't include any negative weight in initialization and the magnitude of weights isn't small enough to move to negative weights in few iterations On the other hand the weights between 0 to 0.05 converged and gave a good accuracy as magnitude was very less to achieve negative weights in few

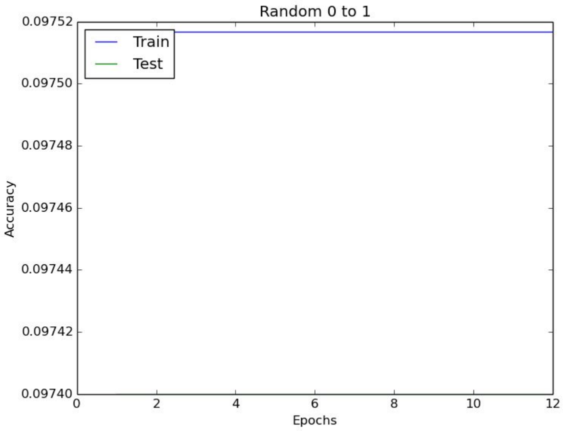
epochs. It is advisable to have both positive and negative values as initial weights.

4. The initialization with weights between -1 to 1 was converging but as magnitude was high so couldn't converge in 12 epochs. Check the learning curve below. It didn't reach to maximum but accuracy was continuously increasing with every epoch.

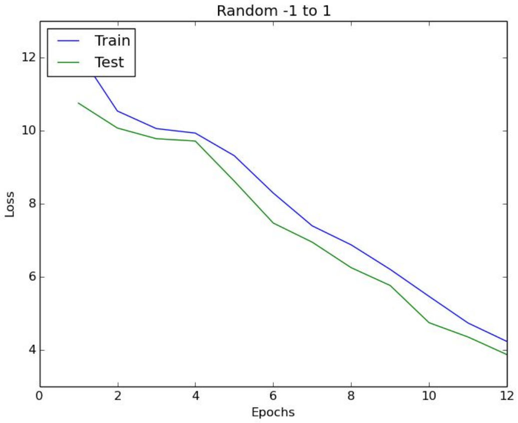
Individual learning plots:

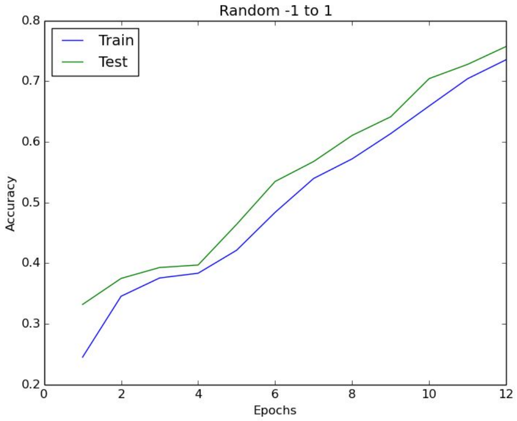
1. Random between 0 to 1



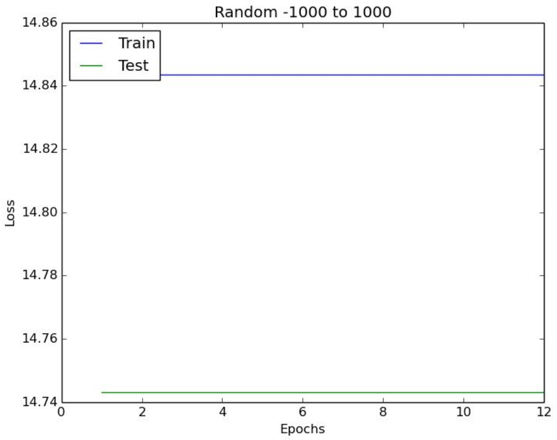


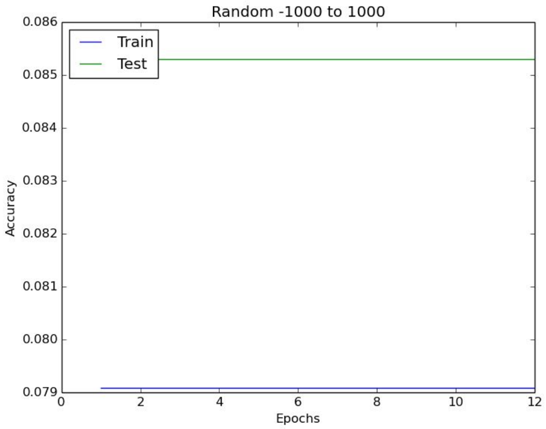
2. Random between -1 to 1



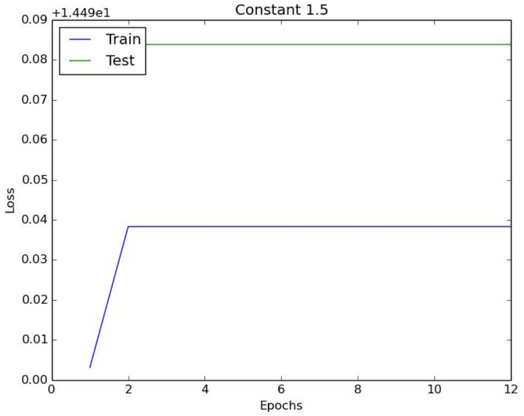


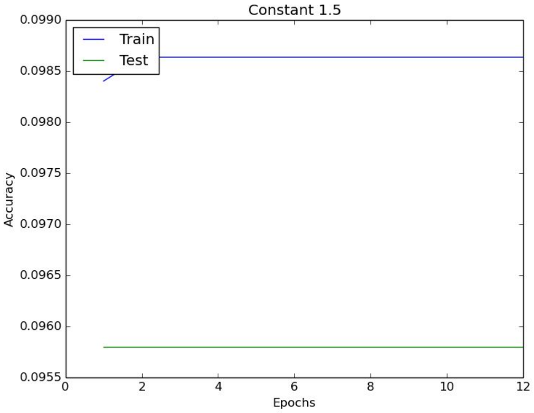
3. Random between -1000 to 1000



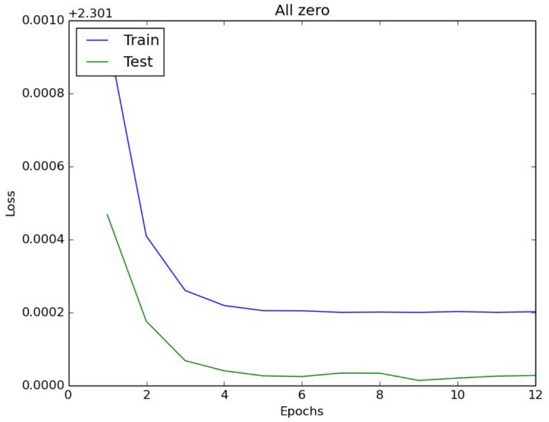


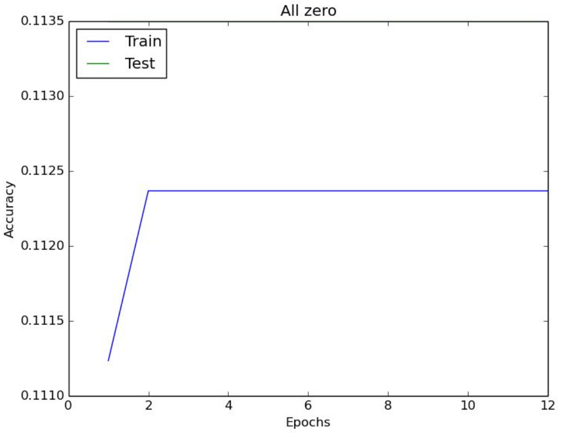
4. All weights initialized to 1.5



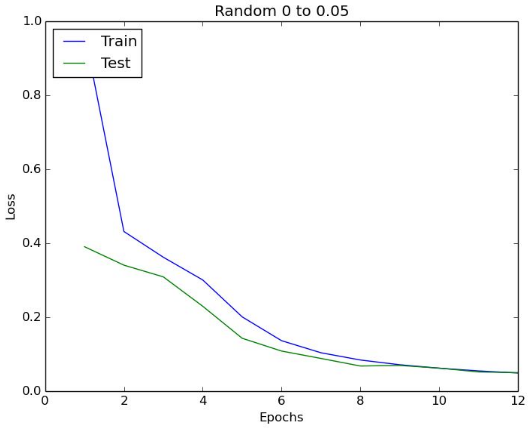


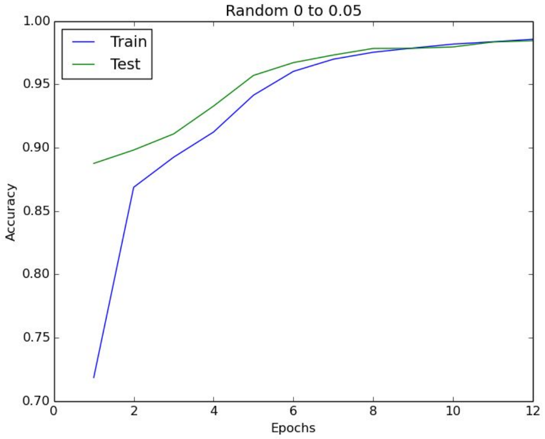
5. All weights initialized to 0



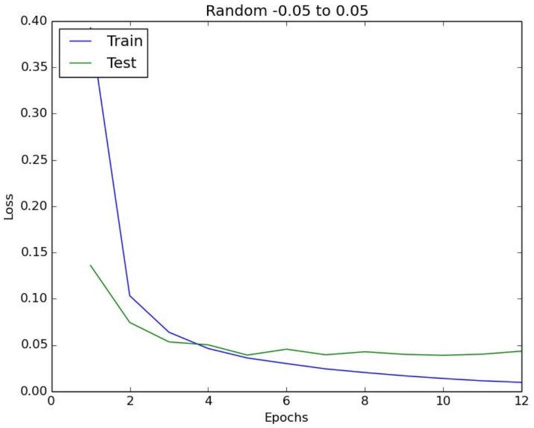


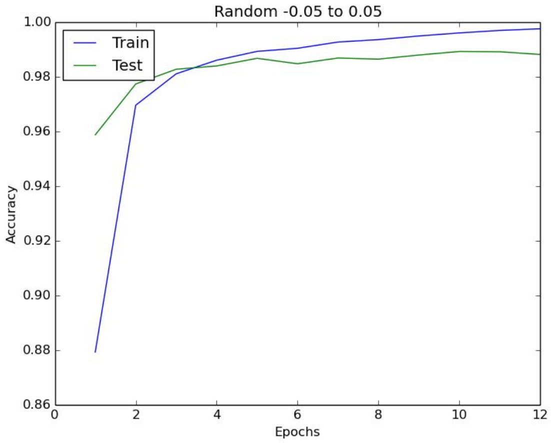
6. Random between 0 to 0.05



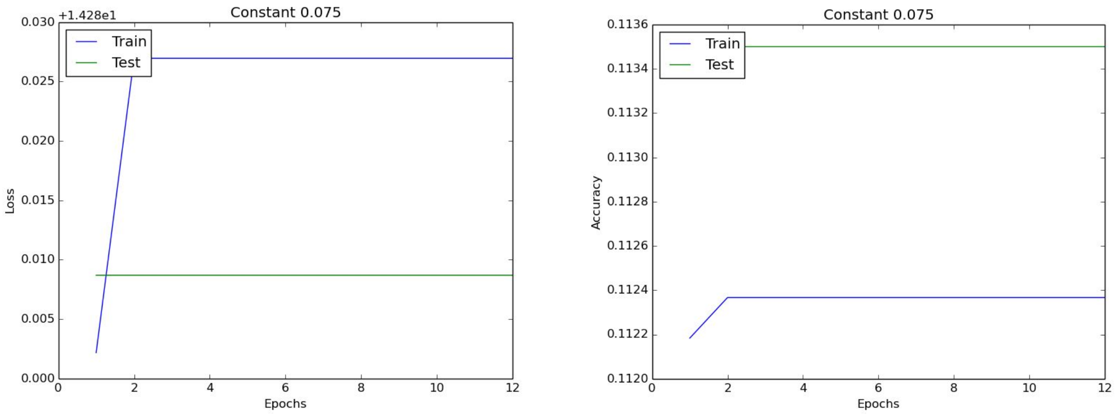


7. Random between -0.05 to 0.05

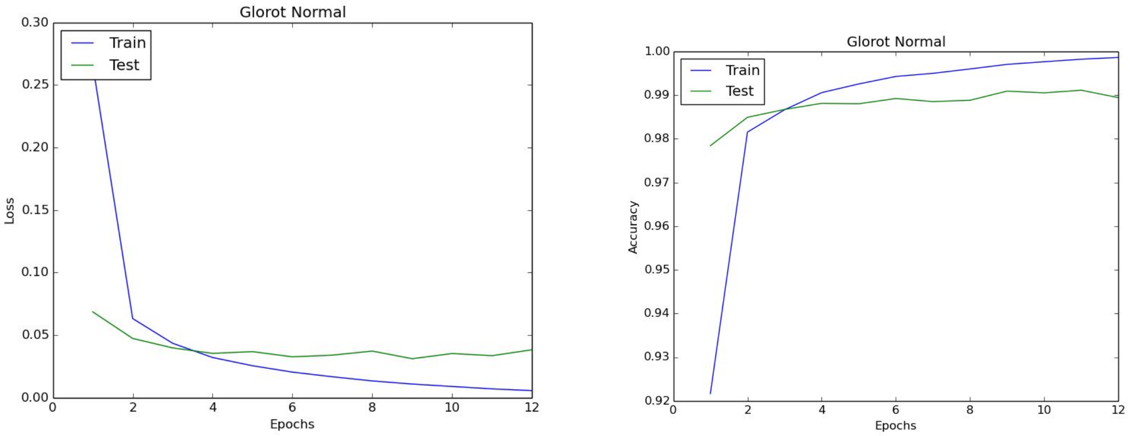




8. All weights initialized to 0.075



9. Xavier’s initialization



Q2. Train the Neural Network with a custom loss function (for ex: Hinge loss, variation of hinge loss etc.).

Ans: Below are the histograms showing losses and accuracies for 4 different objectives:

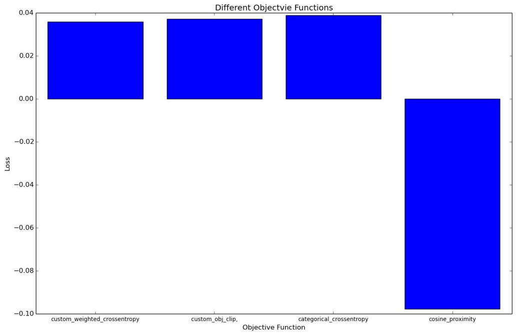
1. **Custom Objective with Weighted Cross Entropy:** We might have cases where digit 7 is being confused with 1, 8 with 3 or 0 and vice versa. To force a weight change corresponding to these special cases we can have a customized higher loss instead of normal cross entropy loss. For these special cases I multiplied the normal loss with a weight 1.3.
   Accuracy Achieved: *0.98999999999999999.*

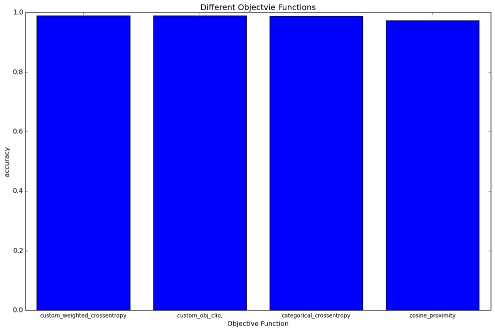
2. **Custom Objective with clipping:** Clipped the loss between 0 and 1. The technique is generally followed to handle exploding gradients. Though there was not a big improvement found in this case as normal cross entropy loss is already giving very high accuracy.

Accuracy Achieved: *0.98950000000000005*.

3. **Categorical_crossentropy:** Normal cross entropy loss provided by Keras.
   Accuracy Achieved: *0.98839999999999995*

4. **Cosine_proximity:** It is another interesting loss function based on cosine similarity between two vectors. It tries to maximize this similarity by minimizing the loss 1 - cosine_similarity. Hence the loss given by it goes negative. Accuracy achieved by this is a bit lesser as it ignores the magnitude of two vectors being compared and only focuses on minimizing the angle between them.
   Accuracy Achieved: *0.97289999999999999*

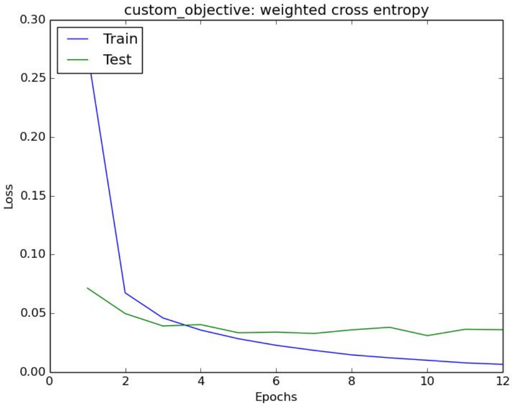


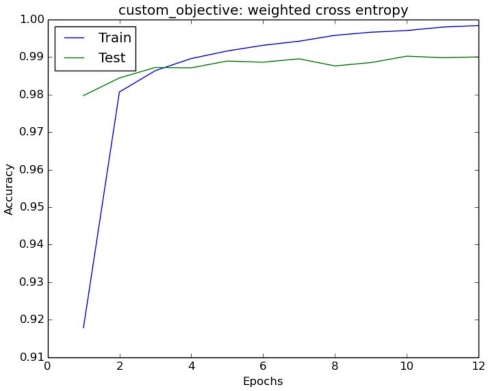


Custom objectives give very little improvement over cross entropy as cross entropy is already performing very good.

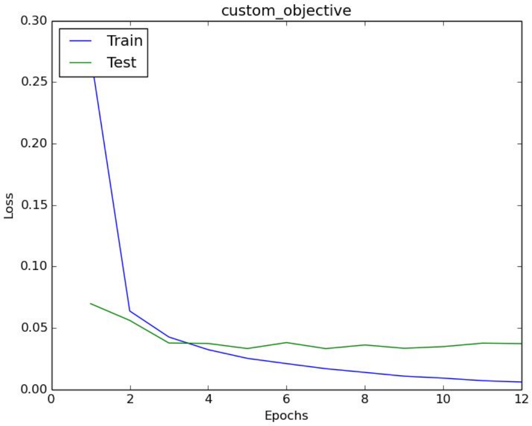
Individual learning plots:

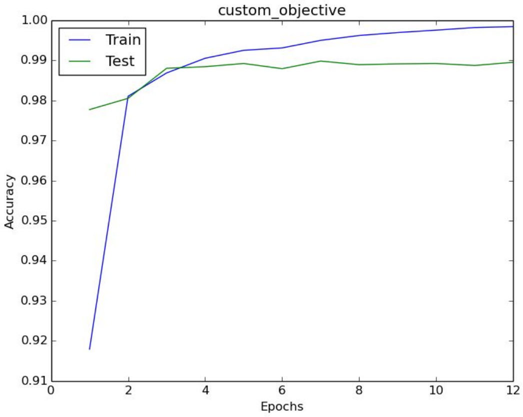
1. Custom Objective with Weighted Cross Entropy



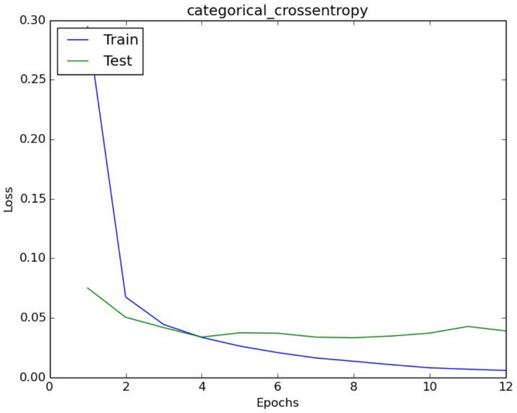


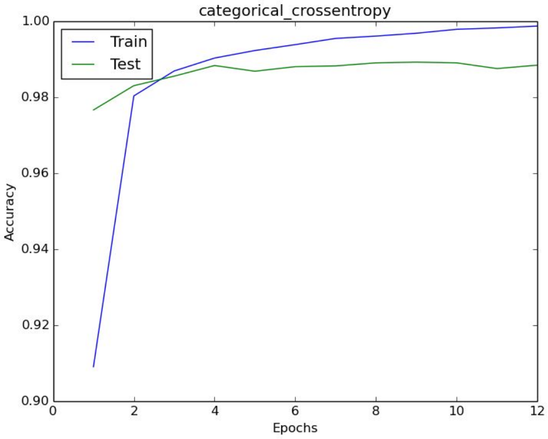
2. Custom Objective with clipping:



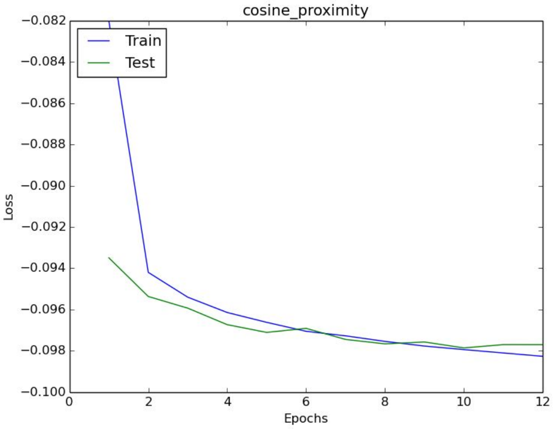


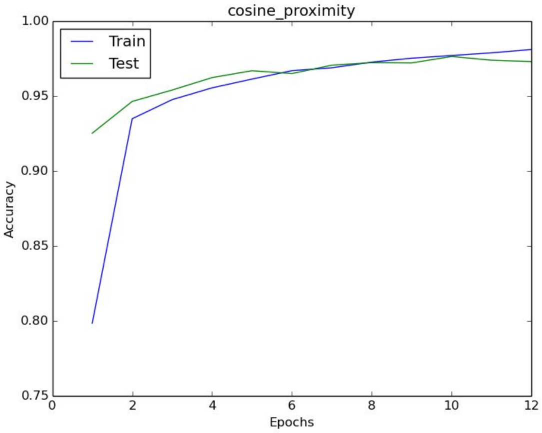
3. Categorical_crossentropy:





4. Cosine_proximity:


cosine_proximity
Train
Test
Loss
Epochs


cosine_proximity
Train
Test
Accuracy
Epochs