**Neural Network Design**

For our neural network we’ve made several design choices that have led to performance exceeding 90% accuracy for our networks.

When choosing the size of layers to accurately model the problem we use empirically-derived rules-of-thumb. In particular, it’s well known that in order to reduce the chance of overfitting and chose a neural network of adequate size to model the problem at hand, the number of neurons in the hidden layer should ideally with the range of the size of neurons in the input layer and output layer. In this case [10, 256].

Activation functions allow us to model non-linear properties for modeling non-linear problems. The choice of activation functions is important because during the learning the backpropagation algorithm calculates the gradient of the activation function, so we can pass the maximum amount of the error though the network during back-propagation if we use ReLU which always has a derivative value of 1 or 0. over Sigmoid and Tanh because the max derivative of Sigmoid is 0.25, Tanh on the other hand has a max derivative of 1. This implies that during our backpropagation algorithm that ReLU consistently learns faster. In an unconstrained problem, ReLU in every layer proves to be empirically superior for learning. In this problem we explore the option of considering Sigmoid and Tanh in the hidden layers.

For the final layer SoftMax is chosen because we’d like our classifier to calculate a probability distribution used to calculate the loss and that determines the error that our backpropagation will exploit to correct the weights. The goal is that as the certainty of a classification increases, the corresponding class probabilities should decrease.

***Fully Connected Neural Network***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Layer # | Type | Neurons | Kernel Size | Activation | Input Shape |
| 1 | Input | 256 |  |  | 256 |
| 2 | Fully Connected | 128 |  | ReLU |  |
| 3 | Fully Connected | 128 |  | Sigmoid |  |
| 4 | Fully Connected | 128 |  | Tanh |  |
| 5 | Output | 10 |  | SoftMax |  |

***Locally Connected Neural Network (No Weights Shared)***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Layer # | Type | Neurons | Kernel Size | Activation | Input Shape |
| 1 | Input | 256 |  |  | 16 x 16 x 1 |
| 2 | Locally Connected 2D | 32 | 3 x 3 | ReLU |  |
| 3 | Locally Connected 2D | 64 | 3 x 3 | Tanh |  |
| 4 | Fully Connected | 128 |  | Sigmoid |  |
| 5 | Output | 10 |  | SoftMax |  |

***Locally Connected Neural Network (Convolutional Neural Network)***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Layer # | Type | Neurons | Kernel Size | Activation | Input Shape |
| 1 | Input | 256 |  |  | 16 x 16 x 1 |
| 2 | Convolutional 2D | 32 | 3 x 3 | ReLU |  |
| 3 | Convolutional 2D | 64 | 3 x 3 | Tanh |  |
| 4 | Fully Connected | 128 |  | Sigmoid |  |
| 5 | Output | 10 |  | SoftMax |  |

**Task 2**

***Parameter initialization strategies.* For each of the networks, analyze how parameters should be initialized. Then demonstrate three cases based on your analysis:**

1. **learning is very slow;**

FNN: weights are initialized to 0.001. Bias defaulted initialized as 0.

LNN: weights are initialized to 0.001. Bias defaulted initialized as 0.

CNN: weights are initialized to 0.001. Bias defaulted initialized as 0.

1. **learning is effective (i.e., fast with accurate results);**

FNN: Weight Initialized using Glorot uniform distribution. Bias defaulted initialized as 0.

LNN: Weight Initialized using Glorot uniform distribution. Bias defaulted initialized as 0.

CNN: Weight Initialized using Glorot uniform distribution. Bias defaulted initialized as 0.

1. **the learning is too fast (i.e., the network does not give good performance).**

FNN: Weights in the first layer Initialized to 1.2. Bias defaulted initialized as 0.

LNN: Weights in the first layer Initialized to 1.2. Bias defaulted initialized as 0.

CNN: Weights in the first layer Initialized to 1.2. Bias defaulted initialized as 0.

***Learning rate.* Estimate a good learning rate for each of the networks. Then demonstrate three cases based on your analysis:**

1) **learning is very slow;**

FNN: Learning rate set to 0.00005.

LNN: Learning rate set to 0.00005.

CNN: Learning rate set to 0.00005.

2) **learning is effective;**

FNN: Learning rate set to 0.001.

LNN: Learning rate set to 0.001.

CNN: Learning rate set to 0.001.

3) **learning is too fast.**

FNN: Learning rate set to 0.05.

LNN: Learning rate set to 0.05.

CNN: Learning rate set to 0.05.

**(3) *Batch size* and batch normalization for each of the networks. Then demonstrate an effective batch size and an ineffective batch size on each of the three networks you have.**

Batch normalization works best with a batch size adequately large enough to sample the real distribution of the entire set. A reduced sample size available for the calculations provides a less accurate representations of the mean and standard deviation of the batch in correspondence to the actual mean and standard deviation of the entire dataset.

An effective Batch size in this case is identified as 128. Our results notably degrade with a batch size of 2 and worsen by a large margin with a batch size of 1.

FNN: Effective Batch Size is 128. Ineffective Batch Size is

LNN: Effective Batch Size is 32. Ineffective Batch Size is

CNN: Effective Batch Size is 128. Ineffective Batch Size is

**(4*) Momentum.* Commonly used momentum coefficient values are 0.5, 0.9, and 0.99. Using the best parameter initialization strategy, the best learning rate, and the best batch size you have found so far, experiment with the three different momentum values on the three networks you have and document the results. Explain the differences you have observed on the three neural networks you have.**

Task 3

For this task, you need to do the required analysis and apply the following regularization techniques with the goal to improve the performance on the 2007 samples in zip\_test.txt.

1. **Use an ensemble to improve the generalization performance. Here you need to use bagging of at least six neural networks to improve the performance of the individual neural networks. You need to analyze your results.**

Our ensemble consists of 6 convolutions neural networks generated from 3 varying architectures. The first architecture is our convolutional neural network design presented in task 1. The Second CNN architecture sets a higher kernel size to possible capture different features / broad features of the images. The final architecture design includes more neurons in the first hidden layers, this is done in the hopes of modeling the data differently and applied to the first layer to reap the benefits of fast learning since is uses the ReLU Activation function.

The 6 neural networks are evenly distributed between the 3 network design, then trained separately on training data. For our ensemble algorithm each CNN predicts the class probabilities for each image in our validation set. The probabilities are summed and averaged then we select the class corresponding to the highest probability of the summed networks.  
  
The ensemble net receives an accuracy average that exceeds 96%. The average accuracy of our ensembled neural network not only exceeds the average results of each of the CNN’s individually but also exceeds the max value for any individual CNN for any given run.

1. **Dropout. Explain the effects of the dropout parameter (probability of keeping a neuron) on the three neural networks you have. Then demonstrate an effective case and an ineffective case on each of the three neural networks you have.**

With dropout we could not achieve superior validation accuracy. In fact, it seems that the accuracy of our model degrades as we increase the probability of dropout. We believe this is because our models generalize well because our model is “relatively small” to the dataset so regularization isn’t necessary because overfitting isn’t so prevalent.

1. **L1 regularization. Explain the effects of the L1 regularization on the three neural networks you have. Then demonstrate an effective L1 regularization case and an ineffective L1 regularization case on each of the three neural networks you have.**

Our previous conclusion regarding regularization for the individual networks seems to be supported because L1 regularization also could not achieve superior validation accuracy. In fact, it also seems that the accuracy of our model degrades as we increase the effects of L1 regularization penalty. Noting that the model has not reached convergence as quickly with regularization then the standard model we increase the epochs (training time) and found that the network indeed does not converge to yield a higher generalization performance. It does seem that the network has reduced its overfitting of the training set, but at the cost of lower overall validation accuracy.