Classifying CIFAR-10 Images with LeNet

Amila Ferron
aferron@pdx.edu

1 LeNet Architecture with Varied Learning Rate, Activation Function, and Loss Function

The Colab notebook is linked here.

Initial trials with the LeNet architecture included 12 base parameter combinations, allowing all other parameters to be at default settings. The learning rates were either 0.1, 0.01, or 0.001; activation functions were Sigmoid or TanH; and the loss function was Mean Squared Error, or Cross-Entropy. Additionally, the data was normalized, training was run for 30 epochs, no batch size was specified, no bias was used in the convolution layers, default kernel initialization was used, and no regularizers were used for the kernel, bias, or activation. The results, seen in Tables 1 and 2, show learning taking place only for the TanH activation function with Mean Squared Error as the loss function, and for Sigmoid activation with Cross-Entropy as the loss function.

These initial results show that the success of the model depends on the combination of parameters, an activation/loss function combination, and that adjusting a single parameter will not necessarily show the full impact it can have on the model's performance. Also interesting is that TanH combined with Mean Squared Error brings quick initial improvements, while Sigmoid with Cross-Entropy has a flat start to improvement, followed by steep improvement and a gradual plateauing.

The most accurate model used TanH as the activation function with Mean Squared Error as the loss function, with an accuracy of 49.19%. At nearly that level, Sigmoid activation with Cross-Entropy produced an accuracy of 48.81%, but only after twice as many epochs.

In order to see the subtleties between the performance of the other parameter combinations and more fully see the potential of the LeNet architecture, additional parameter adjustments were made for further trials. Kernel initialization was adjusted from the default Glorot Uniform, to Random Normal and Glorot Normal. With this single parameter adjustment, Glorot Uniform was found to perform best. In order to allow the models to learn, normalization was removed from the data preprocessing. The theory was that by feeding larger numbers to the model, it would be less likely to get stuck in the flatter parts of the activation functions. This was confirmed by improved results after removing normalization. Adding a bias to the convolution layers was found to yield significant improvement in accuracy as well. The default for the bias is set to zero with Keras, and selecting an initializer for the bias improved performance, possibly for a similar reason as removing normalization showed improvement.

During the process of adjusting the parameters in the convolution layers, it was discovered that the 84-node fully connected layer had been omitted from the models previously tested. After adding the layer, performance dropped significantly, returning to the levels seen in initial testing. This is due to the number of layers causing the backpropagation to produce very small numbers, allowing the model to get stuck in the flat parts of the activation functions. Further adjustments to other parameters were made in order to improve performance again.

A series of experiments were made with regularizers for the kernel, bias, and activation in the convolution layers. They were added individually and in combination, with various values for regularization. The relationships between the regularizers seem complex and there is a lot that can be explored in this area. The configuration that was selected after these trials was an L2 regularizer on the kernel with a regularization factor of 0.00001. Changing the size of the standard deviation on the bias initializer in the second convolutional layer from 1/4704

to 1/3702 gives a 1% increase in accuracy of the model. Regularization seems like a powerful way to influence the direction training goes.

The bias initializer was set with a 0.5 mean Gaussian with a standard deviation of 1/the number of nodes in the input. Setting the mean to 0.5 showed improvement over setting it at 0 or 1. During experimentation, setting the final layer's activation to Sigmoid or TanH, as used by the rest of the model performed better than using softmax. After adjusting all other parameters, changing it back to softmax gave better results. A momentum term was added and removed from the model after it did not improve performance. The biggest improvement was seen after changing the batch size from 128 to 32.

The results with all adjustments to the parameters are shown in Tables 3 and 4. The adjustments made show a shift in the category of the best-performing models from TanH with Mean Squared Error to TanH with Cross-Entropy. TanH with Mean Squared Error shows strong results as well, closely followed by Sigmoid with Cross-Entropy. Sigmoid with Mean Squared Error lags in performance again, even after the improvements in the parameters.

The feature maps for the last convolutional layer for ten randomly selected images are shown in Table 5. Although the image size makes it difficult to understand how the feature maps relate to the images, it appears that there is some differentiation between the object and the background. Some of the feature maps seem to emphasize the boundaries in the images. It's interesting that many of the selected images were labeled wrong by the model, despite this model achieving 54% accuracy.

2 ReLU Activation with 3x3 Kernels

The model used for Section 1 was altered to use ReLU activation, 3x3 kernels instead of 5x5 kernels, and a learning rate of 0.001. The training plot and results are similar to using TanH and Cross-Entropy from the previous model. In order to understand the effect of each of the changes alone, the model used for Section 1 was used with a learning rate of 0.001, Cross-Entropy, and the ReLU activation function. The results are seen in Figure 2. The graph shows a profile similar to the ones produced by the Sigmoid activation function, with a sort of step following a slow start. The adjustment to smaller kernel sizes likely gives enough input to quickly get over the initial bump before learning begins.

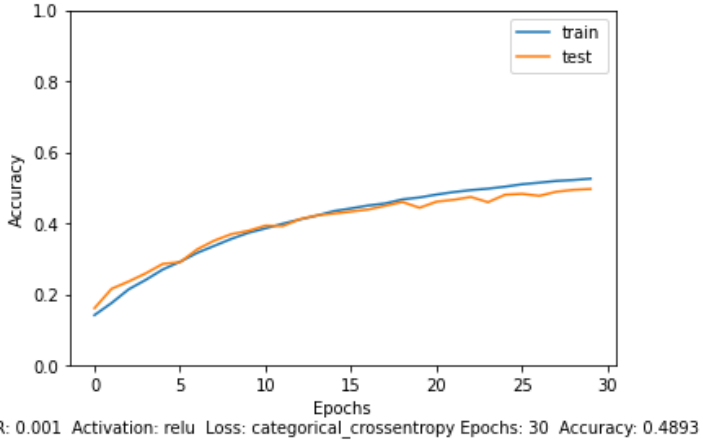


Figure 1: Accuracy per epoch for the model used for Section 2

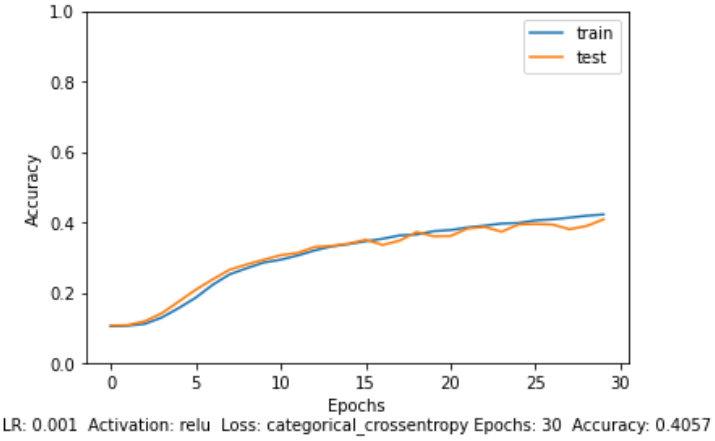


Figure 2: Accuracy per epoch for the model used for Section 1, used with the ReLU activation function

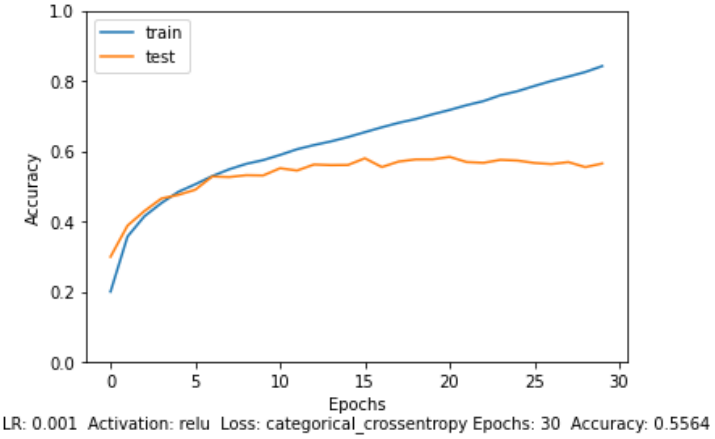
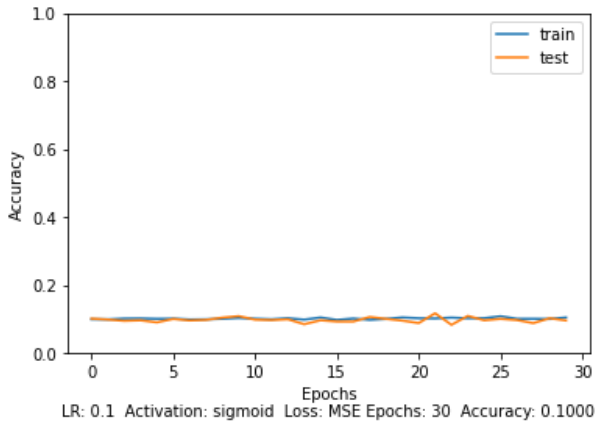


Figure 3: Accuracy per epoch for the model used for Section 3

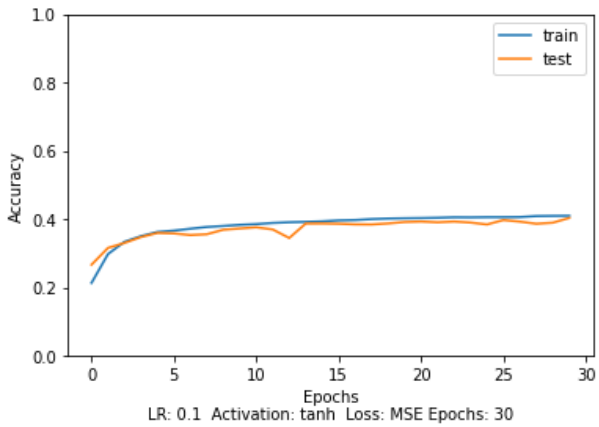
3 Increased Convolution Layers with 3x3 Kernels

The final series of trials modified the architecture to use five convolution layers using 3x3 kernels and padding to keep the input and output sizes the same through convolution and pooling. The new architecture increases the rate of learning so there is a very quick initial improvement phase. Overfitting begins to occur as early as epoch 5 or 6. As with the previous section, the increased information produced seems to effect an increase in the learning rate. The accuracy seen for this architecture – 0.5564 – is the best seen yet and the number of epochs to train is fewer, making it a better option for most uses. Using a learning rate of 0.01 produced an accuracy of 0.5133, slightly lower than with the slower learning rate. Adjusting regularization options for the kernel, bias, and activation gave poorer performance but it seems like there is potential that this could be used to produce gains in accuracy. It would be interesting to try adjusting this and other parameters to maximize performance for this architecture and see how high it could get.

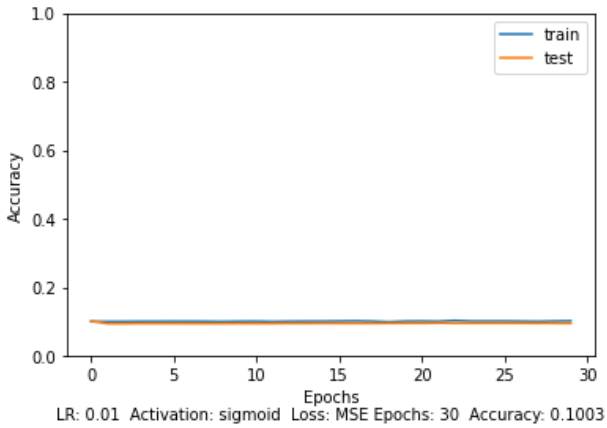
Table 1: Plots from the initial training using Mean Squared Error for Part 1



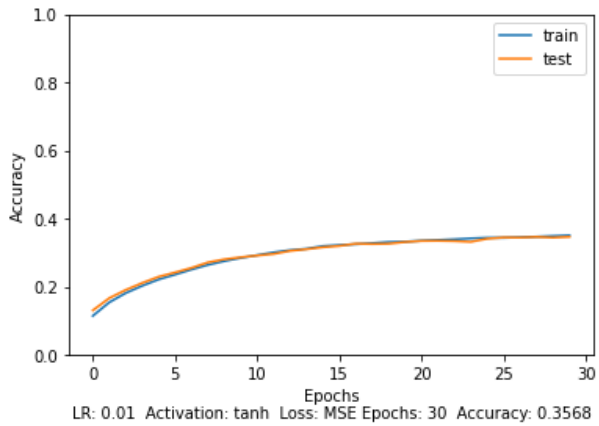
LR: 0.1 Activation: sigmoid Loss: MSE Epochs: 30 Accuracy: 0.1000



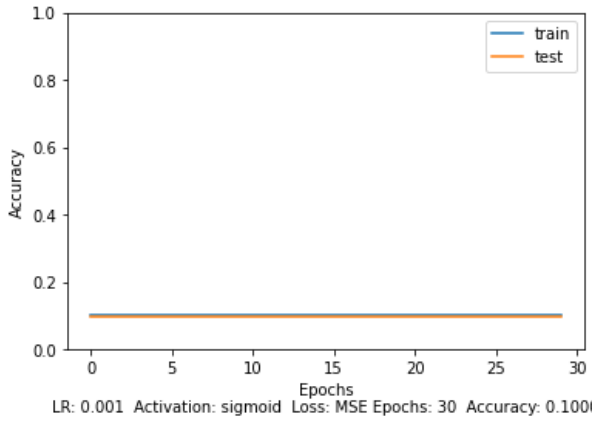
LR: 0.1 Activation: tanh Loss: MSE Epochs: 30



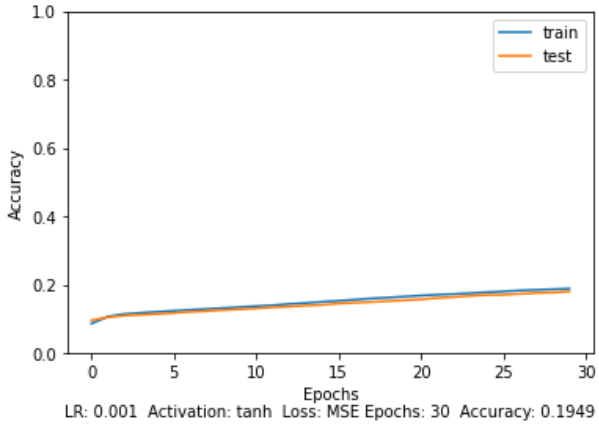
LR: 0.01 Activation: sigmoid Loss: MSE Epochs: 30 Accuracy: 0.1003



LR: 0.01 Activation: tanh Loss: MSE Epochs: 30 Accuracy: 0.3568

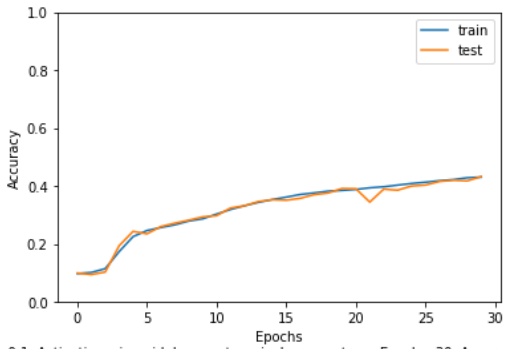


LR: 0.001 Activation: sigmoid Loss: MSE Epochs: 30 Accuracy: 0.1000

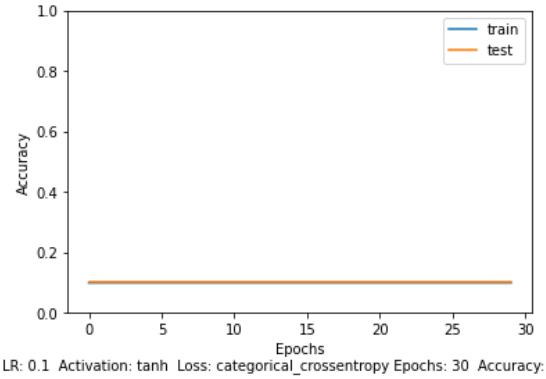


LR: 0.001 Activation: tanh Loss: MSE Epochs: 30 Accuracy: 0.1949

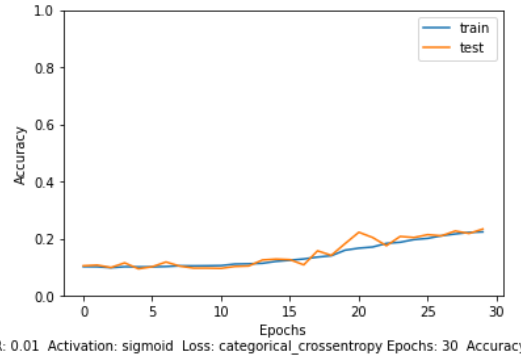
Table 2: Plots from the initial training using Cross-Entropy for Part 1



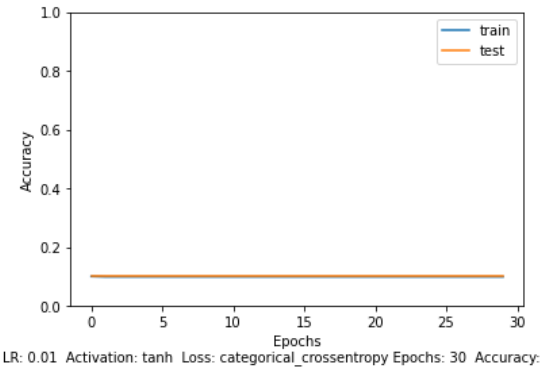
LR: 0.1 Activation: sigmoid Loss: categorical_crossentropy Epochs: 30 Accuracy: 0.4322



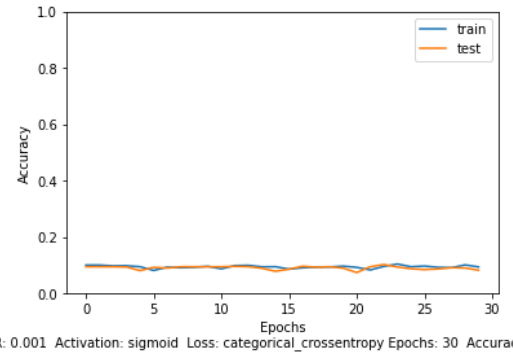
LR: 0.1 Activation: tanh Loss: categorical_crossentropy Epochs: 30 Accuracy: 0.1000



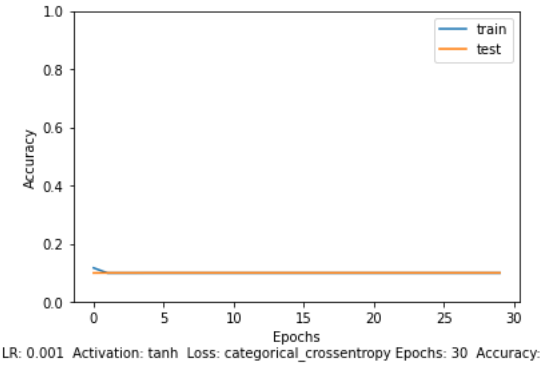
LR: 0.01 Activation: sigmoid Loss: categorical_crossentropy Epochs: 30 Accuracy: 0.2424



LR: 0.01 Activation: tanh Loss: categorical_crossentropy Epochs: 30 Accuracy: 0.1000

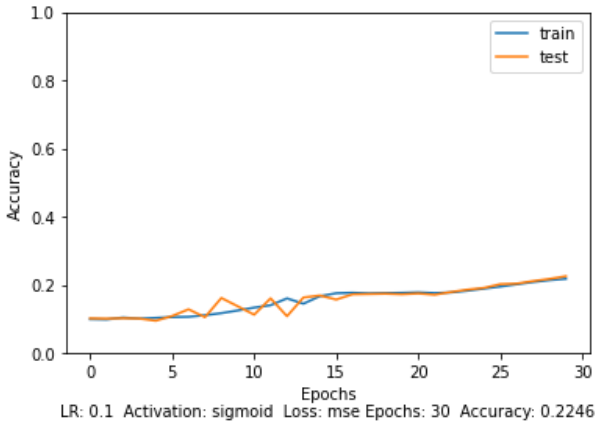


LR: 0.001 Activation: sigmoid Loss: categorical_crossentropy Epochs: 30 Accuracy: 0.0867

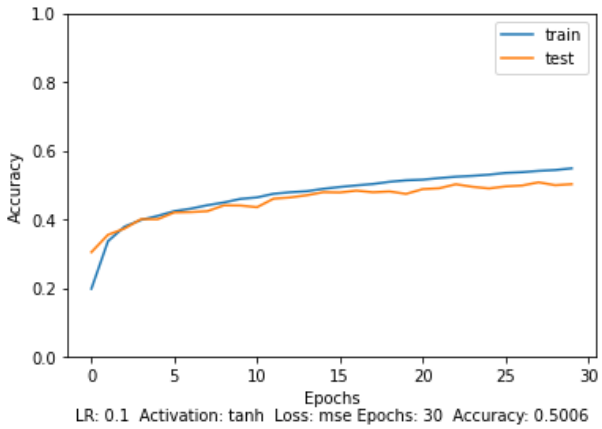


LR: 0.001 Activation: tanh Loss: categorical_crossentropy Epochs: 30 Accuracy: 0.1002

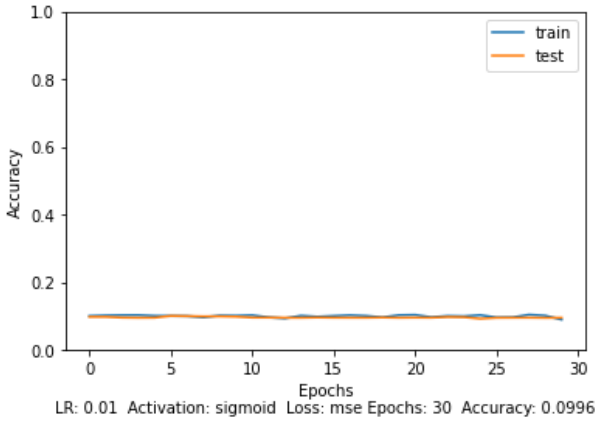
Table 3: Plots from training for Part 1, with Mean Squared Error after all parameter adjustments



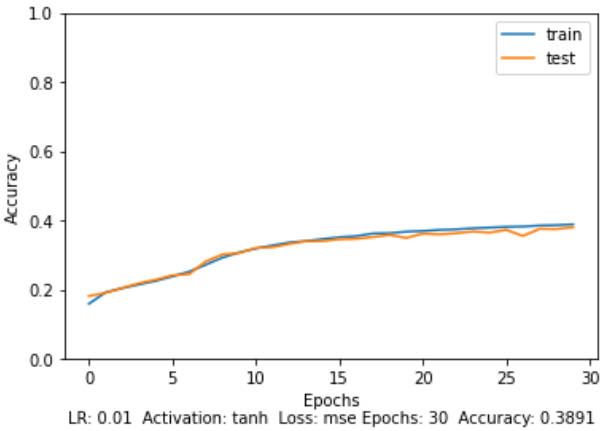
LR: 0.1 Activation: sigmoid Loss: mse Epochs: 30 Accuracy: 0.2246



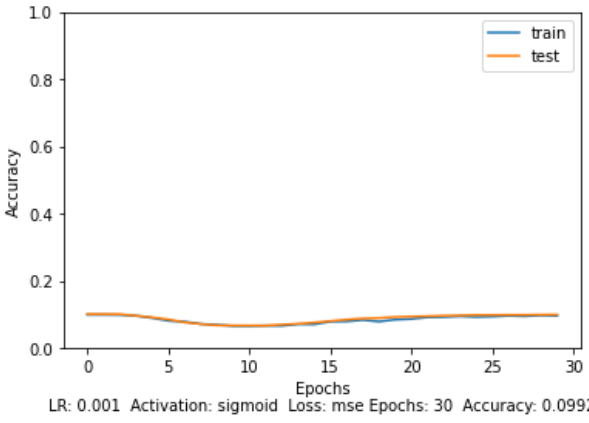
LR: 0.1 Activation: tanh Loss: mse Epochs: 30 Accuracy: 0.5006



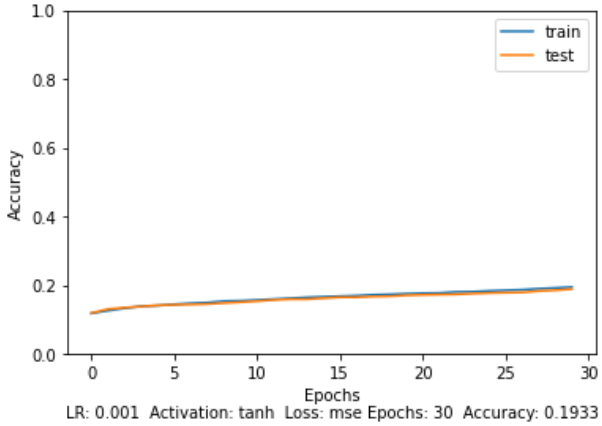
LR: 0.01 Activation: sigmoid Loss: mse Epochs: 30 Accuracy: 0.0996



LR: 0.01 Activation: tanh Loss: mse Epochs: 30 Accuracy: 0.3891

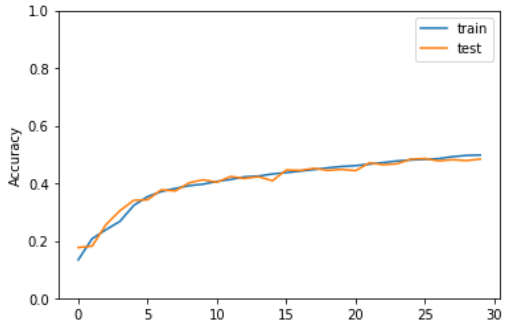


LR: 0.001 Activation: sigmoid Loss: mse Epochs: 30 Accuracy: 0.0992

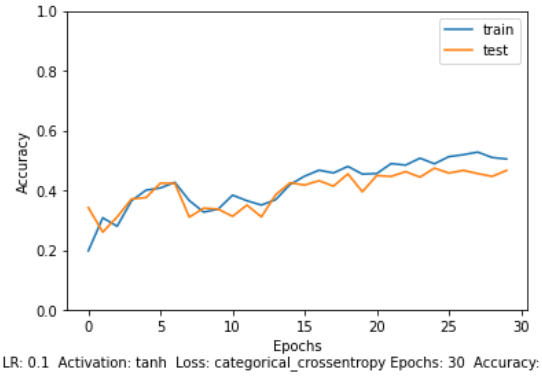


LR: 0.001 Activation: tanh Loss: mse Epochs: 30 Accuracy: 0.1933

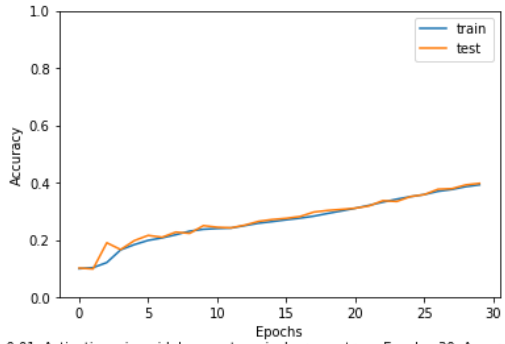
Table 4: Plots from training for Part 1, with cross entropy after all parameter adjustments



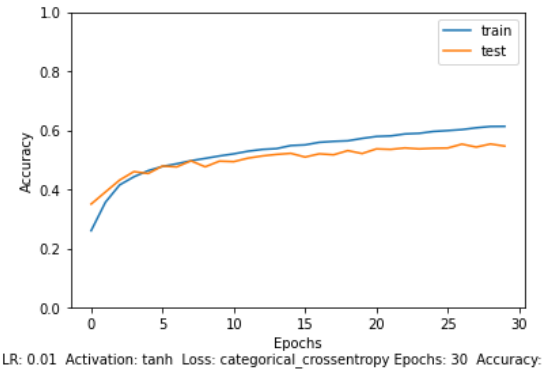
LR: 0.1 Activation: sigmoid Loss: categorical_crossentropy Epochs: 30 Accuracy: 0.4825



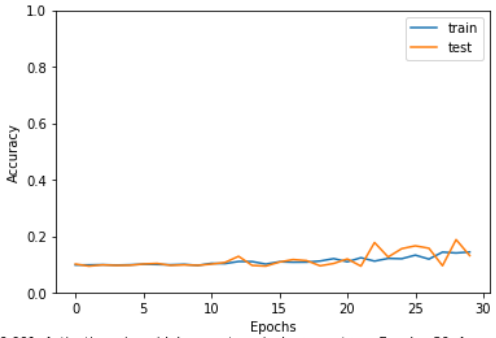
LR: 0.1 Activation: tanh Loss: categorical_crossentropy Epochs: 30 Accuracy: 0.4595



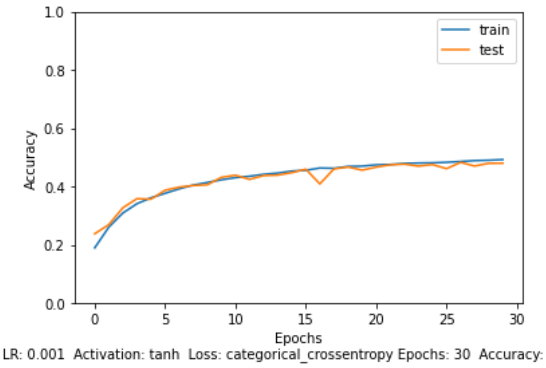
LR: 0.01 Activation: sigmoid Loss: categorical_crossentropy Epochs: 30 Accuracy: 0.3988



LR: 0.01 Activation: tanh Loss: categorical_crossentropy Epochs: 30 Accuracy: 0.5475



LR: 0.001 Activation: sigmoid Loss: categorical_crossentropy Epochs: 30 Accuracy: 0.1409



LR: 0.001 Activation: tanh Loss: categorical_crossentropy Epochs: 30 Accuracy: 0.4797

Table 5: Feature maps from Section 1 model using learning rate of 0.01, TanH activation function, and Cross Entropy as the loss function