CENG 499 Homework 1

Ayse Ozdemir

March 22, 2019

1 Hyperparameter optimization

1.1 1-layer (0-hidden-layer) network

number of epochs = 50
batch size = default (32)

Layer Activations	Learning Rate 0.1	0.01	0.001	0.0001
-	acc: 0.3155 loss: 11.0325	acc: 0.3155 loss: 11.0325	acc: 0.6157 loss: 1.5666	acc: 0.6265 loss: 1.1309

Table 1: 1-layer network

We can easily determine that, for small learning rates model results have higher accuracy and lower loss values.
The python code for this kind of network model is as below(Also could be found in the hw1.py file):

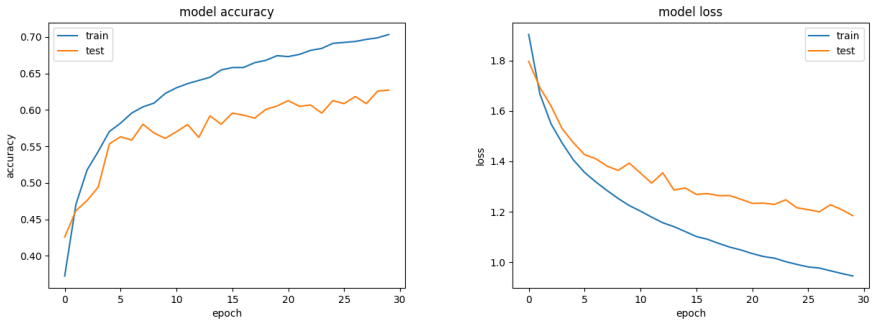


Figure 1: Accuracy and Loss Graphs

1.2 2-layer (1-hidden-layer) network

number of epochs = 50
batch size = default (32)

Layer Activations	Learning Rate 0.1	0.01	0.001	0.0001
S	acc: 0.3155 loss: 11.0325	acc: 0.3155 loss: 2.1877	acc: 0.5992 loss: 1.2909	acc: 0.6623 loss: 1.0298
T	acc: loss:	acc: loss:	acc: 0.3155 loss: 2.1130	acc: 0.6401 loss: 1.1353
R	acc: loss:	acc: loss:	acc: loss:	acc: 0.6589 loss: 1.0425

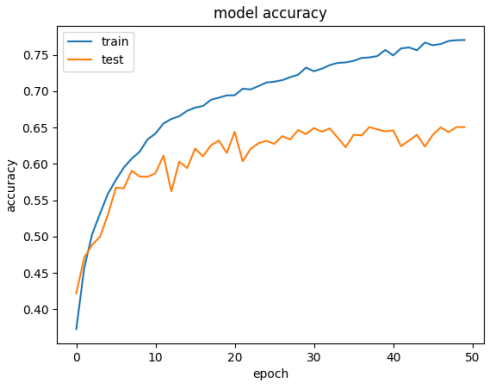
Table 2: 2-layer network

We can easily determine that, for small learning rates model results have higher accuracy and lower loss values.

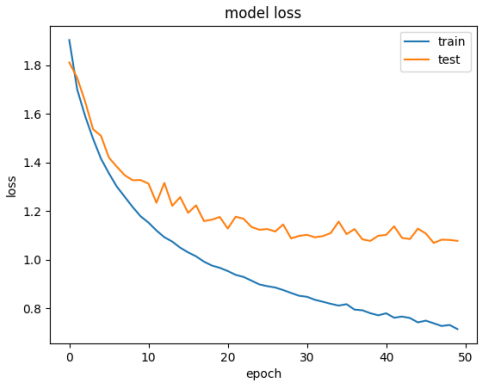
As learning rate is one of the most important hyper parameters for getting high accuracy results, test trials conducted only for the small learning rates(0.001, 0.0001) for each activation function.

The python code for this kind of network model is as below(Also could be found in the hw1.py file):

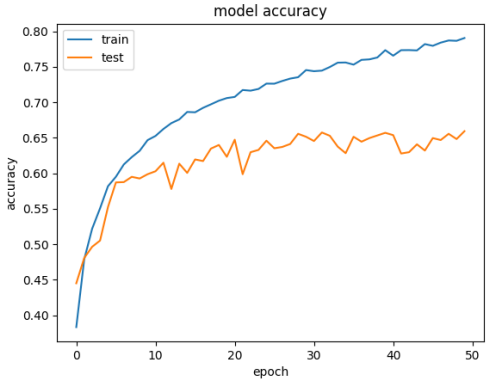
```
model=tf.keras.models.Sequential()
model.add(tf.keras.layers.Flatten(input shape=(128, 64, 3)))

sigmoid:
model.add(tf.keras.layers.Dense(128, activation=tf.nn.sigmoid))
tanh:
model.add(tf.keras.layers.Dense(128, activation=tf.nn.tanh))
relu:
model.add(tf.keras.layers.Dense(128, activation=tf.nn.relu))

model.add(tf.keras.layers.Dense(11, activation=tf.nn.softmax))
```



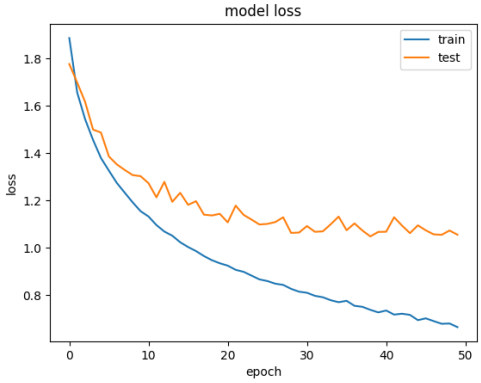
(a) Activation Relu



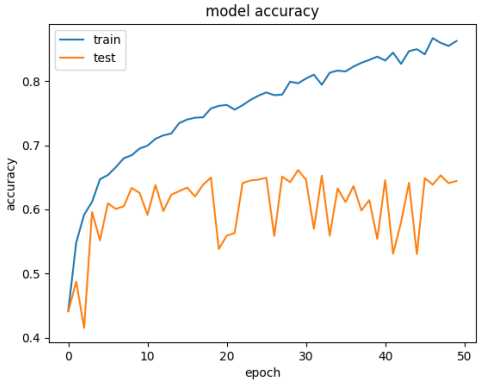
(b) Loss Relu



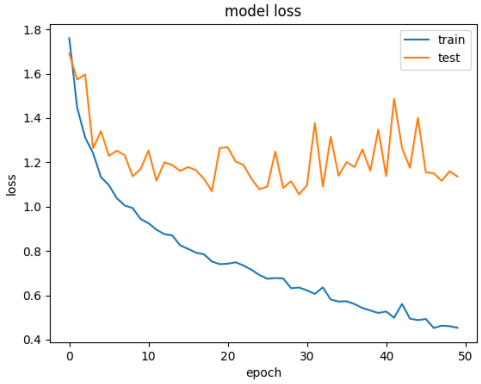
(c) Activation Sigmoid



(d) Loss Sigmoid



(e) Activation Tanh



(f) Loss Tanh

Figure 2: Accuracy and Loss Graphs

1.3 3-layer (2-hidden-layer) network

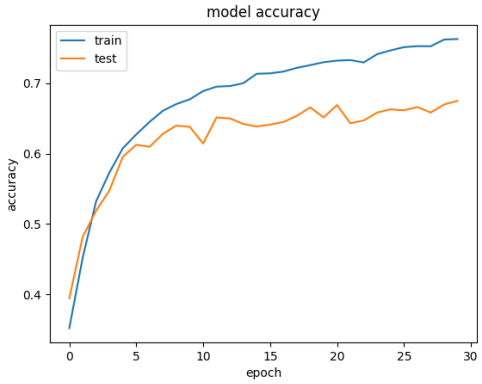
number of epochs = 30 (For time restrictions, and also from the graphs of previous trials, 30 is decided to be used as the number of epochs in this phase)
batch size = default (32)

Layer Activations	Learning Rate 0.1	0.01	0.001	0.0001
SS	acc: loss:	acc: loss:	acc: loss:	acc: 0.6754 loss: 0.9977
ST	acc: loss:	acc: loss:	acc: loss:	acc: 0.6634 loss: 1.0588
SR	acc: loss:	acc: loss:	acc: loss:	acc: 0.6674 loss: 1.0345
TS	acc: loss:	acc: loss:	acc: loss:	acc: 0.6674 loss: 1.0346
TT	acc: loss:	acc: loss:	acc: loss:	acc: 0.6464 loss: 1.0873
TR	acc: loss:	acc: loss:	acc: loss:	acc: 0.6650 loss: 1.0807
RS	acc: loss:	acc: loss:	acc: loss:	acc: 0.6771 loss: 0.9965
RT	acc: loss:	acc: loss:	acc: loss:	acc: 0.6617 loss: 1.0619
RR	acc: loss:	acc: loss:	acc: 0.4838 loss: 1.4932	acc: 0.6748 loss: 1.0404

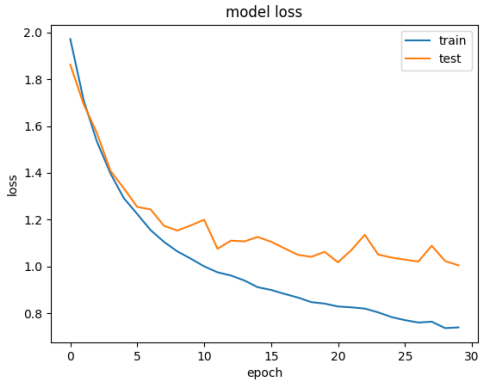
Table 3: 3-layer network

We can easily determine that, for small learning rates model results have higher accuracy and lower loss values.
The python code for this kind of network model is as below(Also could be found in the hw1.py file):

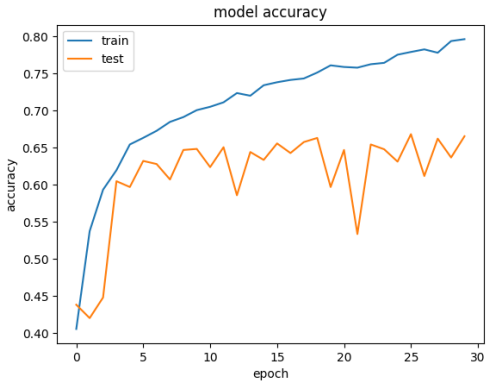
sigmoid-sigmoid:
model=tf.keras.models.Sequential()
model.add(tf.keras.layers.Flatten(input shape=(128, 64, 3)))
model.add(tf.keras.layers.Dense(512, activation=tf.nn.sigmoid))
model.add(tf.keras.layers.Dense(128, activation=tf.nn.sigmoid))
model.add(tf.keras.layers.Dense(11, activation=tf.nn.softmax))
–other combinations of activation functions could be found in hw1.py file.



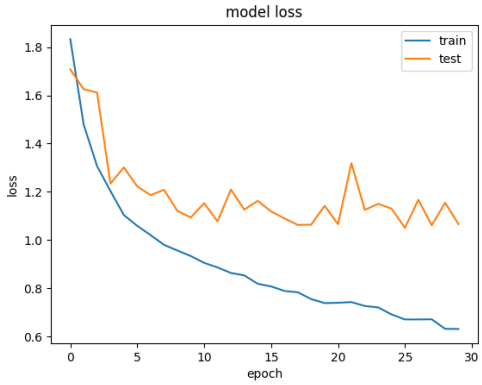
(a) SS Accuracy



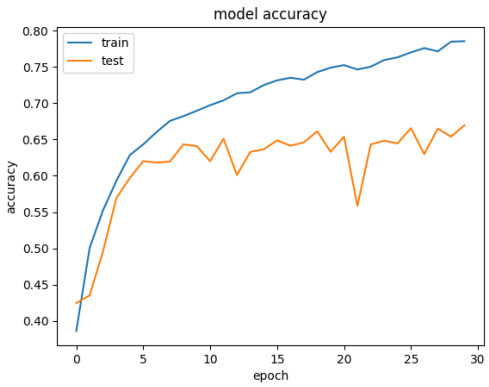
(b) SS Loss



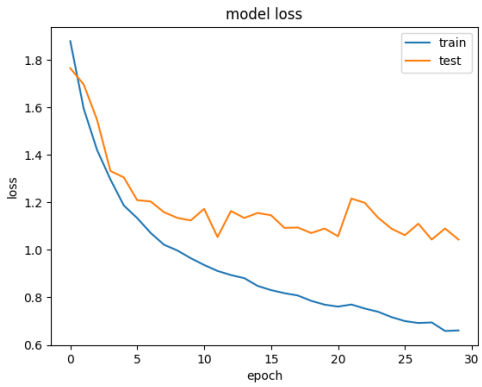
(c) ST Accuracy



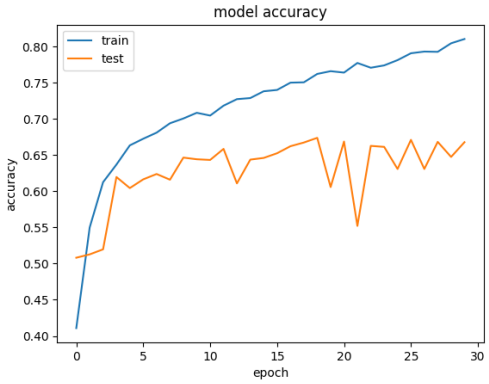
(d) ST Loss



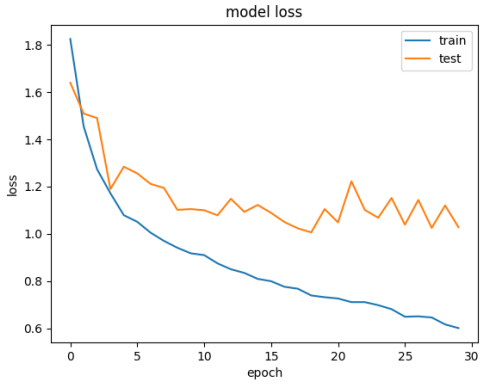
(e) SR Accuracy



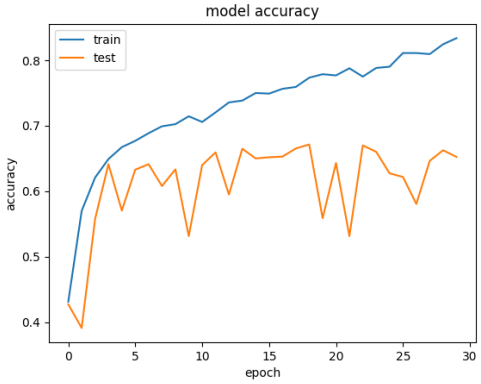
(f) SR Loss



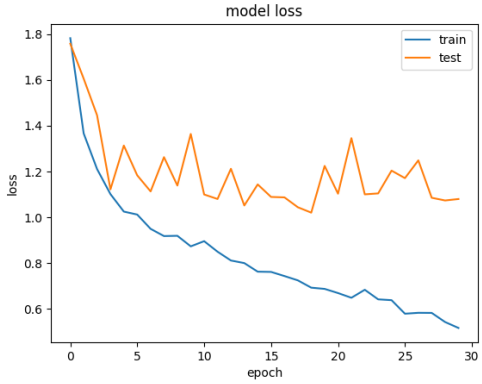
(a) TS Activation



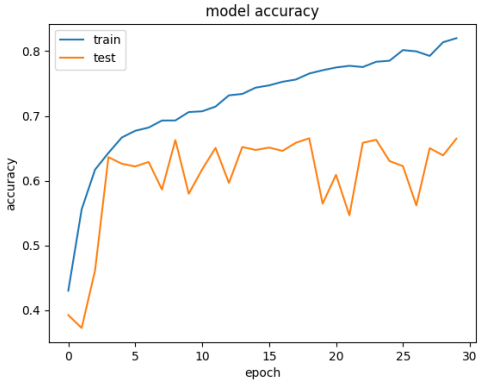
(b) TS Loss



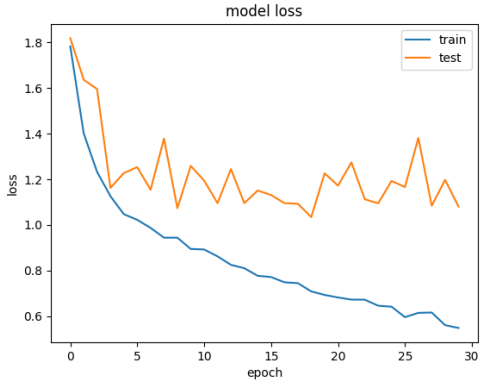
(c) TT Activation



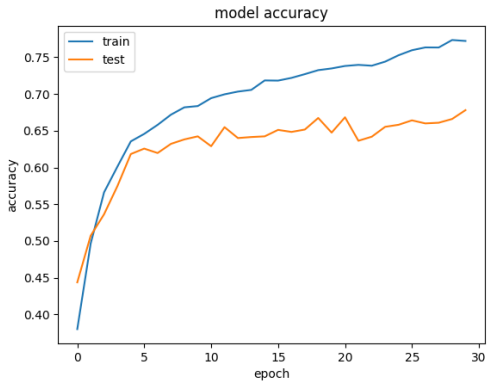
(d) TT Loss



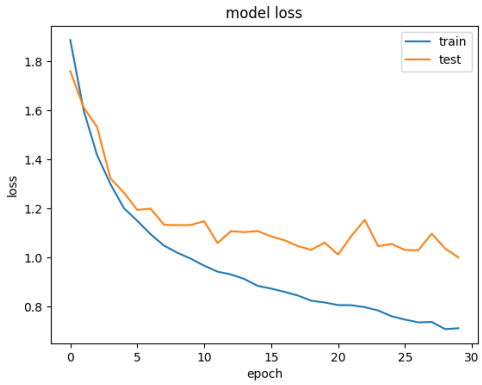
(e) TR Activation



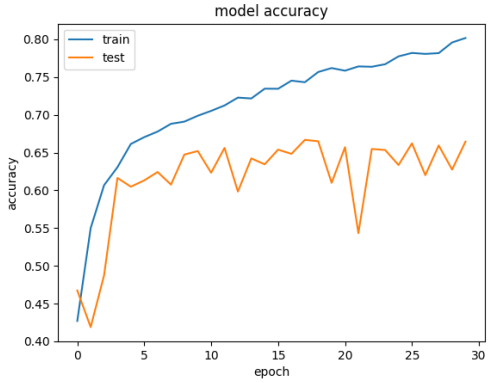
(f) TR Loss



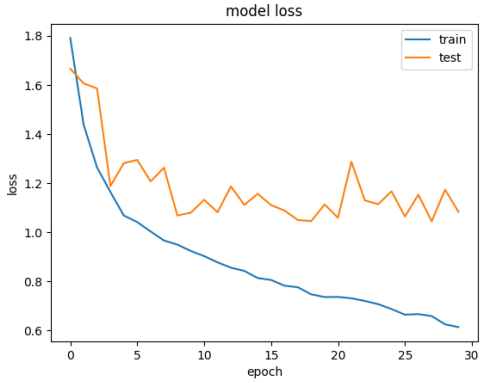
(a) RS Activation



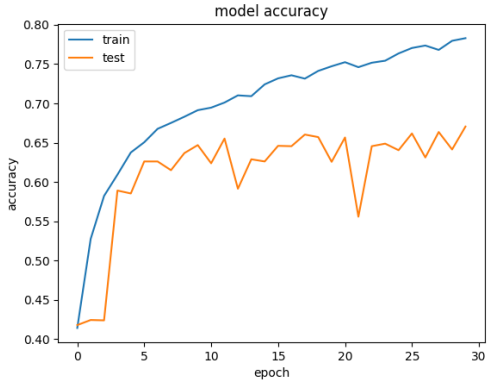
(b) RS Loss



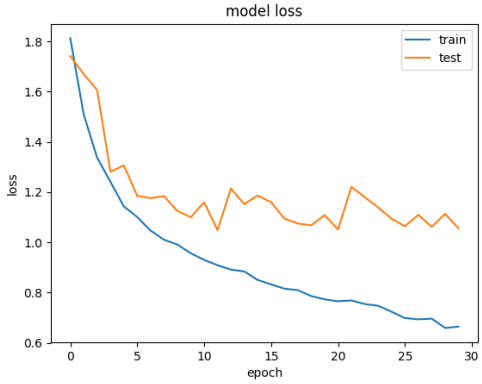
(c) RT Activation



(d) RT Loss



(e) RR Activation



(f) RR Loss

Figure 5: Accuracy and Loss Graphs

2 Comments

As number of hidden layers increased, accuracy is also increased. It could be concluded that; neural networks learn better with the appropriate layer counts.

Learning rate is also a very important hyper parameter in terms of neural network performance. During this study, it is observed that; for small learning rates model results have higher accuracy and lower loss values.

Activation function is also a very important hyper parameter in neural networks. In the 2 layer network model; with a slight difference from other 2 activation functions, Sigmoid gave the best accurate results in the learning rate of 0.0001, and batch size 50.