Assignment 4 – Screenshots

1. CNN Model
2. No normalization Chart, line chart, histogram

   Description automatically generated
3. With pre-processing of images i.e normalization (values between 0-1)

Chart, line chart

Description automatically generated

1. High learning rate – 0.01

Chart

Description automatically generated

Learning rate = 0.0001 and num of epocs = 50

Epoch 50/50

1563/1563 [==============================] - 6s 4ms/step - loss: 0.1680 - accuracy: 0.9477 - val\_loss: 1.1220 - val\_accuracy: 0.7334

Test loss: 1.1219960451126099

Test accuracy: 0.7333999872207642

Chart, line chart, histogram

Description automatically generated

Learning rate = 0.01 epochs=50

Epoch 50/50

1563/1563 [==============================] - 6s 4ms/step - loss: 1.4907 - accuracy: 0.4679 - val\_loss: 1.5411 - val\_accuracy: 0.4557

Test loss: 1.5410703420639038

Test accuracy: 0.45570001006126404

Graphical user interface, chart

Description automatically generated

LR = 0.1 epoch=20

Epoch 20/20

1563/1563 [==============================] - 6s 4ms/step - loss: 2.3156 - accuracy: 0.0991 - val\_loss: 2.3153 - val\_accuracy: 0.1000

Test loss: 2.315288782119751

Test accuracy: 0.10000000149011612

Chart, line chart

Description automatically generated

Kernel (4,4), epochs-20,lr=0.001

Epoch 20/20

1563/1563 [==============================] - 8s 5ms/step - loss: 0.1290 - accuracy: 0.9568 - val\_loss: 1.7931 - val\_accuracy: 0.7077

Test loss: 1.7930519580841064

Test accuracy: 0.7077000141143799

Chart, line chart

Description automatically generated

Lr=0.001,epochs=30,kernel=3,3,strides=1,padding = same

Epoch 30/30

1563/1563 [==============================] - 6s 4ms/step - loss: 0.0916 - accuracy: 0.9816 - val\_loss: 1.6135 - val\_accuracy: 0.6272

Test loss: 1.6135224103927612

Test accuracy: 0.6272000074386597

Chart, line chart

Description automatically generated

Activation function – LeakyReLU

epochs = 30

num\_classes = 10

learning\_rate = 0.001

#Normalize pixel values to be between 0 and 1

# convert from integers to floats

norm\_x\_train = x\_train.astype('float32')

norm\_x\_test = x\_test.astype('float32')

# normalize to range 0-1

x\_train = (norm\_x\_train / 255.0 )

x\_test = (norm\_x\_test / 255.0)

#One hot encode label values

y\_train\_one\_hot = to\_categorical(y\_train)

y\_test\_one\_hot = to\_categorical(y\_test)

#Optimizer

opt = tf.keras.optimizers.Adam(learning\_rate=learning\_rate)

#Create CNN model - covolutional layers, pooling layer, FC layer

model = Sequential()

model.add(Conv2D(32, kernel\_size=(3, 3),activation='linear',input\_shape=(32, 32, 3),padding='same'))

model.add(LeakyReLU(alpha=0.1))

model.add(MaxPooling2D((2, 2),padding='same'))

#model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), activation='linear',padding='same'))

model.add(LeakyReLU(alpha=0.1))

model.add(MaxPooling2D(pool\_size=(2, 2),padding='same'))

#model.add(Dropout(0.25))

model.add(Conv2D(128, (3, 3), activation='linear',padding='same'))

model.add(LeakyReLU(alpha=0.1))

model.add(MaxPooling2D(pool\_size=(2, 2),padding='same'))

#model.add(Dropout(0.4))

model.add(Flatten())

model.add(Dense(128, activation='linear'))

model.add(LeakyReLU(alpha=0.1))

#model.add(Dropout(0.3))

model.add(Dense(num\_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical\_crossentropy, optimizer=opt,metrics=['accuracy'])

model.summary()

#history = model.fit(x\_train, y\_train\_one\_hot,epochs=epochs,batch\_size=batch\_size,verbose=1,validation\_data=(x\_test, y\_test\_one\_hot))

history = model.fit(x\_train, y\_train\_one\_hot,epochs=epochs,verbose=1,validation\_data=(x\_test, y\_test\_one\_hot))

Epoch 30/30

1563/1563 [==============================] - 6s 4ms/step - loss: 0.0870 - accuracy: 0.9733 - val\_loss: 2.3046 - val\_accuracy: 0.7358

Test loss: 2.30458927154541

Test accuracy: 0.73580002784729

Graphical user interface, chart, line chart

Description automatically generated

RELU

batch\_size = 64

epochs = 30

num\_classes = 10

learning\_rate = 0.001

#Normalize pixel values to be between 0 and 1

# convert from integers to floats

norm\_x\_train = x\_train.astype('float32')

norm\_x\_test = x\_test.astype('float32')

# normalize to range 0-1

x\_train = (norm\_x\_train / 255.0 )

x\_test = (norm\_x\_test / 255.0)

#One hot encode label values

y\_train\_one\_hot = to\_categorical(y\_train)

y\_test\_one\_hot = to\_categorical(y\_test)

#Optimizer

opt = tf.keras.optimizers.Adam(learning\_rate=learning\_rate)

#Create CNN model - covolutional layers, pooling layer, FC layer

model = Sequential()

model.add(Conv2D(32, kernel\_size=(3, 3),activation='relu',input\_shape=(32, 32, 3),padding='same'))

#model.add(LeakyReLU(alpha=0.1))

model.add(MaxPooling2D((2, 2),padding='same'))

#model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), activation='relu',padding='same'))

#model.add(LeakyReLU(alpha=0.1))

model.add(MaxPooling2D(pool\_size=(2, 2),padding='same'))

#model.add(Dropout(0.25))

model.add(Conv2D(128, (3, 3), activation='relu',padding='same'))

#model.add(LeakyReLU(alpha=0.1))

model.add(MaxPooling2D(pool\_size=(2, 2),padding='same'))

#model.add(Dropout(0.4))

model.add(Flatten())

model.add(Dense(128, activation='relu'))

#model.add(LeakyReLU(alpha=0.1))

#model.add(Dropout(0.3))

model.add(Dense(num\_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical\_crossentropy, optimizer=opt,metrics=['accuracy'])

model.summary()

#history = model.fit(x\_train, y\_train\_one\_hot,epochs=epochs,batch\_size=batch\_size,verbose=1,validation\_data=(x\_test, y\_test\_one\_hot))

history = model.fit(x\_train, y\_train\_one\_hot,epochs=epochs,verbose=1,validation\_data=(x\_test, y\_test\_one\_hot))

Epoch 30/30

1563/1563 [==============================] - 6s 4ms/step - loss: 0.0862 - accuracy: 0.9714 - val\_loss: 2.1089 - val\_accuracy: 0.7321

Test loss: 2.1089119911193848

Test accuracy: 0.7321000099182129

Graphical user interface, chart

Description automatically generated

Sigmoid

epochs = 30

num\_classes = 10

learning\_rate = 0.001

#Normalize pixel values to be between 0 and 1

# convert from integers to floats

norm\_x\_train = x\_train.astype('float32')

norm\_x\_test = x\_test.astype('float32')

# normalize to range 0-1

x\_train = (norm\_x\_train / 255.0 )

x\_test = (norm\_x\_test / 255.0)

#One hot encode label values

y\_train\_one\_hot = to\_categorical(y\_train)

y\_test\_one\_hot = to\_categorical(y\_test)

#Optimizer

opt = tf.keras.optimizers.Adam(learning\_rate=learning\_rate)

#Create CNN model - covolutional layers, pooling layer, FC layer

model = Sequential()

model.add(Conv2D(32, kernel\_size=(3, 3),activation='sigmoid',input\_shape=(32, 32, 3),padding='same'))

#model.add(LeakyReLU(alpha=0.1))

model.add(MaxPooling2D((2, 2),padding='same'))

#model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), activation='sigmoid',padding='same'))

#model.add(LeakyReLU(alpha=0.1))

model.add(MaxPooling2D(pool\_size=(2, 2),padding='same'))

#model.add(Dropout(0.25))

model.add(Conv2D(128, (3, 3), activation='sigmoid',padding='same'))

#model.add(LeakyReLU(alpha=0.1))

model.add(MaxPooling2D(pool\_size=(2, 2),padding='same'))

#model.add(Dropout(0.4))

model.add(Flatten())

model.add(Dense(128, activation='sigmoid'))

#model.add(LeakyReLU(alpha=0.1))

#model.add(Dropout(0.3))

model.add(Dense(num\_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical\_crossentropy, optimizer=opt,metrics=['accuracy'])

model.summary()

#history = model.fit(x\_train, y\_train\_one\_hot,epochs=epochs,batch\_size=batch\_size,verbose=1,validation\_data=(x\_test, y\_test\_one\_hot))

history = model.fit(x\_train, y\_train\_one\_hot,epochs=epochs,verbose=1,validation\_data=(x\_test, y\_test\_one\_hot))

Epoch 30/30

1563/1563 [==============================] - 6s 4ms/step - loss: 0.1811 - accuracy: 0.9521 - val\_loss: 1.5493 - val\_accuracy: 0.6098

Test loss: 1.5492618083953857

Test accuracy: 0.6097999811172485

Chart, line chart

Description automatically generated

With dropout

epochs = 50

num\_classes = 10

learning\_rate = 0.001

#Normalize pixel values to be between 0 and 1

# convert from integers to floats

norm\_x\_train = x\_train.astype('float32')

norm\_x\_test = x\_test.astype('float32')

# normalize to range 0-1

x\_train = (norm\_x\_train / 255.0 )

x\_test = (norm\_x\_test / 255.0)

#One hot encode label values

y\_train\_one\_hot = to\_categorical(y\_train)

y\_test\_one\_hot = to\_categorical(y\_test)

#Optimizer

opt = tf.keras.optimizers.Adam(learning\_rate=learning\_rate)

#Create CNN model - covolutional layers, pooling layer, FC layer

model = Sequential()

model.add(Conv2D(32, kernel\_size=(3, 3),activation='relu',input\_shape=(32, 32, 3),padding='same'))

#model.add(LeakyReLU(alpha=0.1))

model.add(MaxPooling2D((2, 2),padding='same'))

model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), activation='relu',padding='same'))

#model.add(LeakyReLU(alpha=0.1))

model.add(MaxPooling2D(pool\_size=(2, 2),padding='same'))

model.add(Dropout(0.25))

model.add(Conv2D(128, (3, 3), activation='relu',padding='same'))

#model.add(LeakyReLU(alpha=0.1))

model.add(MaxPooling2D(pool\_size=(2, 2),padding='same'))

model.add(Dropout(0.4))

model.add(Flatten())

model.add(Dense(128, activation='relu'))

#model.add(LeakyReLU(alpha=0.1))

model.add(Dropout(0.3))

model.add(Dense(num\_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical\_crossentropy, optimizer=opt,metrics=['accuracy'])

model.summary()

#history = model.fit(x\_train, y\_train\_one\_hot,epochs=epochs,batch\_size=batch\_size,verbose=1,validation\_data=(x\_test, y\_test\_one\_hot))

history = model.fit(x\_train, y\_train\_one\_hot,epochs=epochs,verbose=1,validation\_data=(x\_test, y\_test\_one\_hot))

Epoch 50/50

1563/1563 [==============================] - 7s 4ms/step - loss: 0.6218 - accuracy: 0.7831 - val\_loss: 0.6327 - val\_accuracy: 0.7862

Test loss: 0.6326842904090881

Test accuracy: 0.7861999869346619

Chart

Description automatically generated

Dropout-0.2

epochs = 50

num\_classes = 10

learning\_rate = 0.001

#Normalize pixel values to be between 0 and 1

# convert from integers to floats

norm\_x\_train = x\_train.astype('float32')

norm\_x\_test = x\_test.astype('float32')

# normalize to range 0-1

x\_train = (norm\_x\_train / 255.0 )

x\_test = (norm\_x\_test / 255.0)

#One hot encode label values

y\_train\_one\_hot = to\_categorical(y\_train)

y\_test\_one\_hot = to\_categorical(y\_test)

#Optimizer

opt = tf.keras.optimizers.Adam(learning\_rate=learning\_rate)

#Create CNN model - covolutional layers, pooling layer, FC layer

model = Sequential()

model.add(Conv2D(32, kernel\_size=(3, 3),activation='relu',input\_shape=(32, 32, 3),padding='same'))

#model.add(LeakyReLU(alpha=0.1))

model.add(MaxPooling2D((2, 2),padding='same'))

model.add(Dropout(0.2))

model.add(Conv2D(64, (3, 3), activation='relu',padding='same'))

#model.add(LeakyReLU(alpha=0.1))

model.add(MaxPooling2D(pool\_size=(2, 2),padding='same'))

model.add(Dropout(0.2))

model.add(Conv2D(128, (3, 3), activation='relu',padding='same'))

#model.add(LeakyReLU(alpha=0.1))

model.add(MaxPooling2D(pool\_size=(2, 2),padding='same'))

model.add(Dropout(0.2))

model.add(Flatten())

model.add(Dense(128, activation='relu'))

#model.add(LeakyReLU(alpha=0.1))

model.add(Dropout(0.2))

model.add(Dense(num\_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical\_crossentropy, optimizer=opt,metrics=['accuracy'])

model.summary()

#history = model.fit(x\_train, y\_train\_one\_hot,epochs=epochs,batch\_size=batch\_size,verbose=1,validation\_data=(x\_test, y\_test\_one\_hot))

history = model.fit(x\_train, y\_train\_one\_hot,epochs=epochs,verbose=1,validation\_data=(x\_test, y\_test\_one\_hot))

Epoch 50/50

1563/1563 [==============================] - 6s 4ms/step - loss: 0.4421 - accuracy: 0.8453 - val\_loss: 0.6681 - val\_accuracy: 0.7880

Test loss: 0.6680977940559387

Test accuracy: 0.7879999876022339

Graphical user interface

Description automatically generated

Kernel initializer = glorot\_normal

epochs = 100

num\_classes = 10

learning\_rate = 0.001

#Normalize pixel values to be between 0 and 1

# convert from integers to floats

norm\_x\_train = x\_train.astype('float32')

norm\_x\_test = x\_test.astype('float32')

# normalize to range 0-1

x\_train = (norm\_x\_train / 255.0 )

x\_test = (norm\_x\_test / 255.0)

#One hot encode label values

y\_train\_one\_hot = to\_categorical(y\_train)

y\_test\_one\_hot = to\_categorical(y\_test)

#Optimizer

opt = tf.keras.optimizers.Adam(learning\_rate=learning\_rate)

#Create CNN model - covolutional layers, pooling layer, FC layer

model = Sequential()

model.add(Conv2D(32, kernel\_size=(3, 3),kernel\_initializer="glorot\_normal",input\_shape=(32, 32, 3),padding='same'))

model.add(LeakyReLU(alpha=0.1))

model.add(MaxPooling2D((2, 2),padding='same'))

model.add(Dropout(0.2))

model.add(Conv2D(64, (3, 3),kernel\_initializer="glorot\_normal",padding='same'))

model.add(LeakyReLU(alpha=0.1))

model.add(MaxPooling2D(pool\_size=(2, 2),padding='same'))

model.add(Dropout(0.2))

model.add(Conv2D(128, (3, 3),kernel\_initializer="glorot\_normal",padding='same'))

model.add(LeakyReLU(alpha=0.1))

model.add(MaxPooling2D(pool\_size=(2, 2),padding='same'))

model.add(Dropout(0.2))

model.add(Flatten())

model.add(Dense(128,kernel\_initializer="glorot\_normal"))

model.add(LeakyReLU(alpha=0.1))

model.add(Dropout(0.2))

model.add(Dense(num\_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical\_crossentropy, optimizer=opt,metrics=['accuracy'])

model.summary()

#history = model.fit(x\_train, y\_train\_one\_hot,epochs=epochs,batch\_size=batch\_size,verbose=1,validation\_data=(x\_test, y\_test\_one\_hot))

history = model.fit(x\_train, y\_train\_one\_hot,epochs=epochs,verbose=1,validation\_data=(x\_test, y\_test\_one\_hot))

Epoch 100/100

1563/1563 [==============================] - 7s 4ms/step - loss: 0.3078 - accuracy: 0.8918 - val\_loss: 0.8006 - val\_accuracy: 0.7970

Test loss: 0.8006476759910583

Test accuracy: 0.796999990940094

Graphical user interface, application

Description automatically generated