Shiny Sherly Katuru

7007443414

Neural Networks and Deep Learning

Assignment-7

In class programming:

1. Follow the instruction below and then report how the performance changed.(apply all at once)

   • Convolutional input layer, 32 feature maps with a size of 3×3 and a rectifier activation function.

   • Dropout layer at 20%.

   • Convolutional layer, 32 feature maps with a size of 3×3 and a rectifier activation function.

   • Max Pool layer with size 2×2.

   • Convolutional layer, 64 feature maps with a size of 3×3 and a rectifier activation function.

   • Dropout layer at 20%.

   • Convolutional layer, 64 feature maps with a size of 3×3 and a rectifier activation function.

   • Max Pool layer with size 2×2.

   • Convolutional layer, 128 feature maps with a size of 3×3 and a rectifier activation function. • Dropout layer at 20%.

   • Convolutional layer,128 feature maps with a size of 3×3 and a rectifier activation function.

   • Max Pool layer with size 2×2.

   • Flatten layer.

   • Dropout layer at 20%.

   • Fully connected layer with 1024 units and a rectifier activation function.

   • Dropout layer at 20%.

   • Fully connected layer with 512 units and a rectifier activation function.

   • Dropout layer at 20%.

   • Fully connected output layer with 10 units and a Softmax activation function Did the performance change?

localhost:8888/notebooks/ICP7_NN.ipynb

jupyter ICP7_NN Last Checkpoint: 3 hours ago (autosaved)

In [13]:
```python
import numpy as np
from keras.datasets import cifar10
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.constraints import maxnorm
from keras.optimizers import SGD
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.utils import np_utils
```

In [3]:
```python
np.random.seed(7)
```

In [4]:
```python
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
```

In [5]:
```python
X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0
```

In [6]:
```python
y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)
num_classes = y_test.shape[1]
```

In [7]:
```python
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(32, 32, 3), padding='same', activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', kernel_constraint=maxnorm(3)))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))
model.add(Flatten())
model.add(Dense(512, activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```

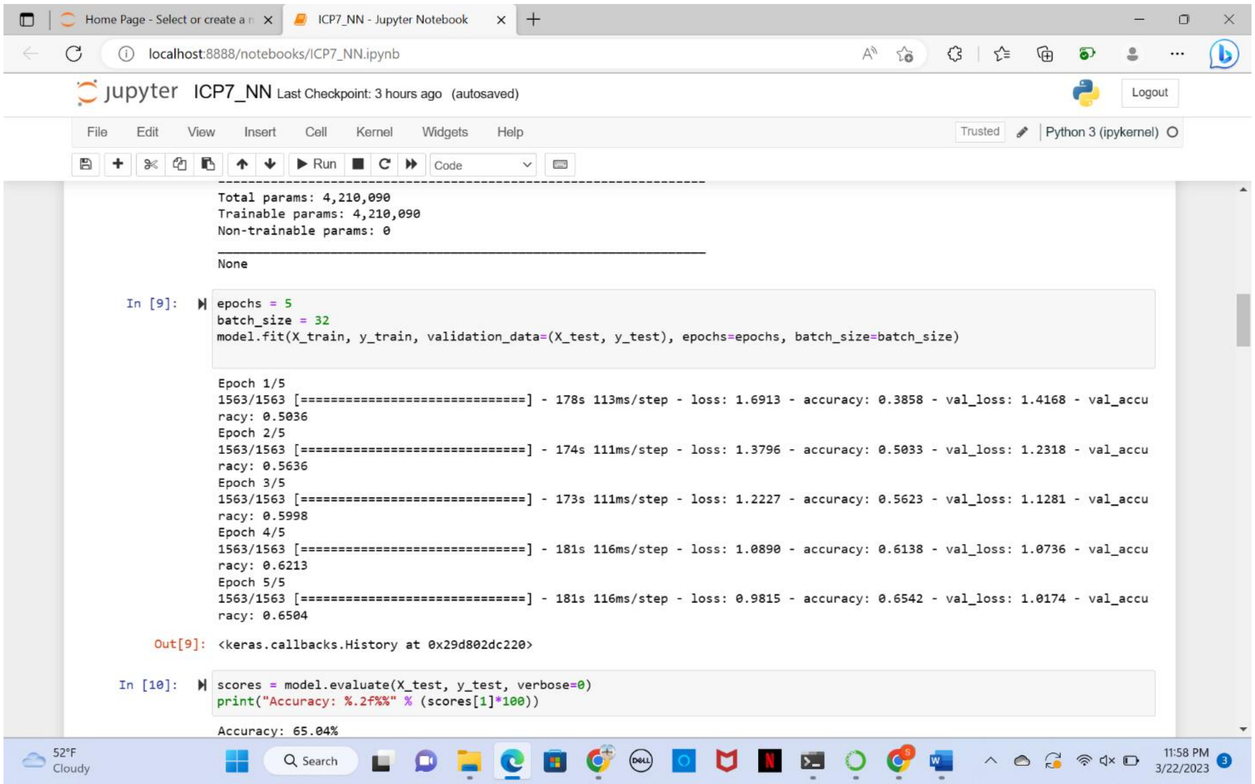

localhost:8888/notebooks/ICP7_NN.ipynb

jupyter ICP7_NN Last Checkpoint: 3 hours ago (autosaved)

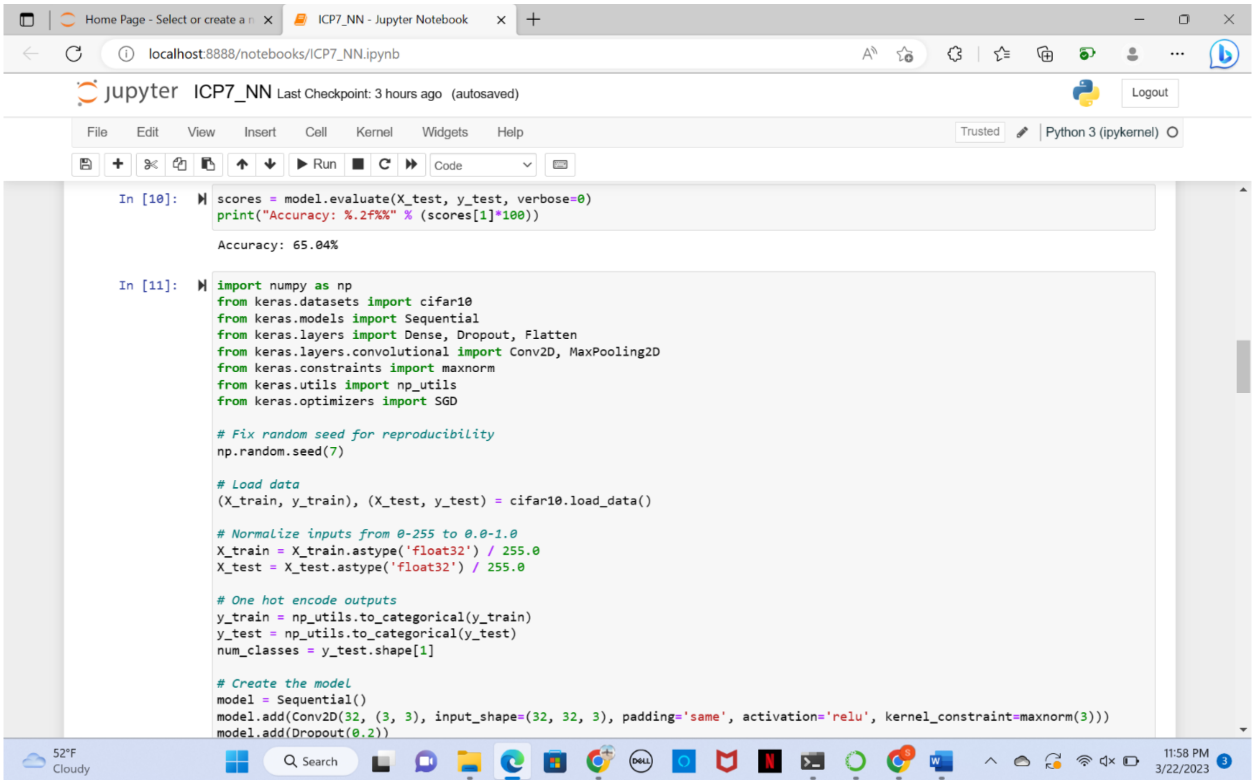
In [7]:
```python
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(32, 32, 3), padding='same', activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', kernel_constraint=maxnorm(3)))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))
model.add(Flatten())
model.add(Dense(512, activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```


In [8]:
```python
sgd = SGD(learning_rate=0.01, momentum=0.9, decay=1e-6)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
print(model.summary())
```

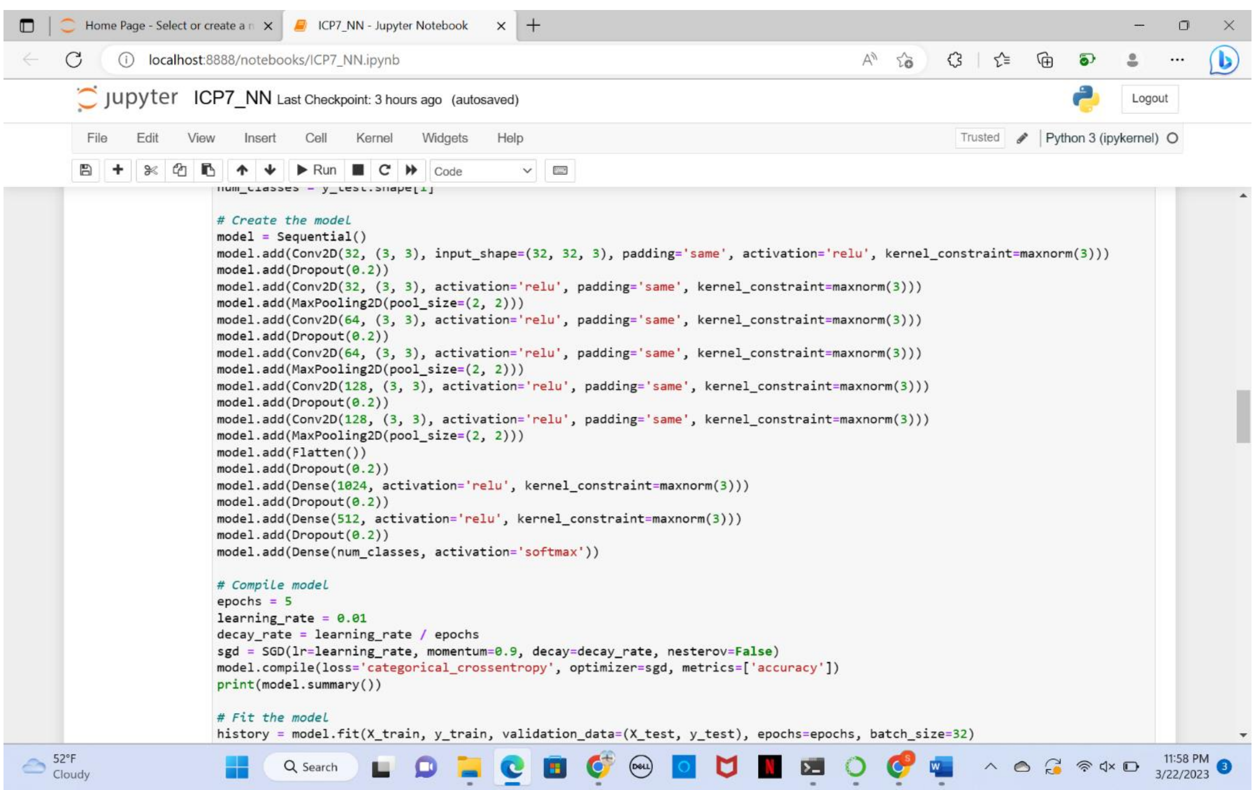
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 32)        896

 dropout (Dropout)           (None, 32, 32, 32)        0

 conv2d_1 (Conv2D)           (None, 32, 32, 32)        9248

 max_pooling2d (MaxPooling2D  (None, 16, 16, 32)       0
 )

 flatten (Flatten)           (None, 8192)              0

 dense (Dense)               (None, 512)               4194816
```


Home Page - Select or create a n
ICP7_NN - Jupyter Notebook
localhost:8888/notebooks/ICP7_NN.ipynb
jupyter ICP7_NN Last Checkpoint: 3 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Total params: 4,210,090
Trainable params: 4,210,090
Non-trainable params: 0
None
In [9]:
epochs = 5
batch_size = 32
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=epochs, batch_size=batch_size)
Epoch 1/5
1563/1563 [==============================] - 178s 113ms/step - loss: 1.6913 - accuracy: 0.3858 - val_loss: 1.4168 - val_accuracy: 0.5036
Epoch 2/5
1563/1563 [==============================] - 174s 111ms/step - loss: 1.3796 - accuracy: 0.5033 - val_loss: 1.2318 - val_accuracy: 0.5636
Epoch 3/5
1563/1563 [==============================] - 173s 111ms/step - loss: 1.2227 - accuracy: 0.5623 - val_loss: 1.1281 - val_accuracy: 0.5998
Epoch 4/5
1563/1563 [==============================] - 181s 116ms/step - loss: 1.0890 - accuracy: 0.6138 - val_loss: 1.0736 - val_accuracy: 0.6213
Epoch 5/5
1563/1563 [==============================] - 181s 116ms/step - loss: 0.9815 - accuracy: 0.6542 - val_loss: 1.0174 - val_accuracy: 0.6504
Out[9]: <keras.callbacks.History at 0x29d802dc220>
In [10]:
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
Accuracy: 65.04%
52°F
Cloudy
Search
11:58 PM
3/22/2023

2



```
In [10]:  scores = model.evaluate(X_test, y_test, verbose=0)
          print("Accuracy: %.2f%%" % (scores[1]*100))

          Accuracy: 65.04%

In [11]:  import numpy as np
          from keras.datasets import cifar10
          from keras.models import Sequential
          from keras.layers import Dense, Dropout, Flatten
          from keras.layers.convolutional import Conv2D, MaxPooling2D
          from keras.constraints import maxnorm
          from keras.utils import np_utils
          from keras.optimizers import SGD

          # Fix random seed for reproducibility
          np.random.seed(7)

          # Load data
          (X_train, y_train), (X_test, y_test) = cifar10.load_data()

          # Normalize inputs from 0-255 to 0.0-1.0
          X_train = X_train.astype('float32') / 255.0
          X_test = X_test.astype('float32') / 255.0

          # One hot encode outputs
          y_train = np_utils.to_categorical(y_train)
          y_test = np_utils.to_categorical(y_test)
          num_classes = y_test.shape[1]

          # Create the model
          model = Sequential()
          model.add(Conv2D(32, (3, 3), input_shape=(32, 32, 3), padding='same', activation='relu', kernel_constraint=maxnorm(3)))
          model.add(Dropout(0.2))
```



```
num_classes = y_test.shape[1]

# Create the model
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(32, 32, 3), padding='same', activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', kernel_constraint=maxnorm(3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', kernel_constraint=maxnorm(3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', kernel_constraint=maxnorm(3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dropout(0.2))
model.add(Dense(1024, activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))

# Compile model
epochs = 5
learning_rate = 0.01
decay_rate = learning_rate / epochs
sgd = SGD(lr=learning_rate, momentum=0.9, decay=decay_rate, nesterov=False)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
print(model.summary())

# Fit the model
history = model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=epochs, batch_size=32)
```

```python
# Fit the model
history = model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=epochs, batch_size=32)

# Evaluate the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1] * 100))
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 32, 32, 32)        896

 dropout_2 (Dropout)         (None, 32, 32, 32)        0

 conv2d_3 (Conv2D)           (None, 32, 32, 32)        9248

 max_pooling2d_1 (MaxPooling  (None, 16, 16, 32)       0
 2D)

 conv2d_4 (Conv2D)           (None, 16, 16, 64)        18496

 dropout_3 (Dropout)         (None, 16, 16, 64)        0

 conv2d_5 (Conv2D)           (None, 16, 16, 64)        36928

 max_pooling2d_2 (MaxPooling  (None, 8, 8, 64)         0
 2D)

 conv2d_6 (Conv2D)           (None, 8, 8, 128)         73856

 dropout_4 (Dropout)         (None, 8, 8, 128)         0
```

```
 dense_4 (Dense)             (None, 10)                5130

=================================================================
Total params: 2,915,114
Trainable params: 2,915,114
Non-trainable params: 0
_________________________________________________________________
```

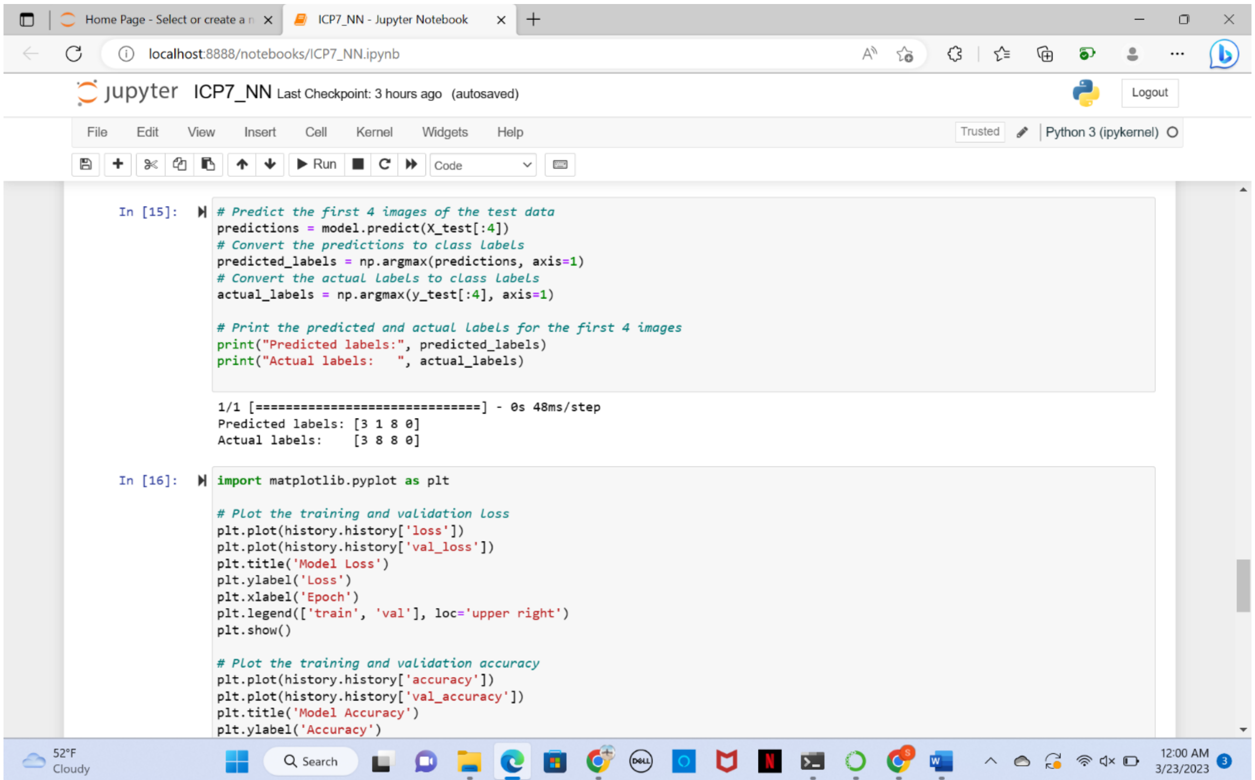
```
C:\Users\shiny\anaconda3\lib\site-packages\keras\optimizers\optimizer_v2\gradient_descent.py:114: UserWarning: The `lr` argu
ment is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)
```

```
None
Epoch 1/5
1563/1563 [==============================] - 349s 222ms/step - loss: 1.8774 - accuracy: 0.3035 - val_loss: 1.5661 - val_accu
racy: 0.4307
Epoch 2/5
1563/1563 [==============================] - 1512s 968ms/step - loss: 1.5070 - accuracy: 0.4497 - val_loss: 1.4609 - val_acc
uracy: 0.4680
Epoch 3/5
1563/1563 [==============================] - 342s 219ms/step - loss: 1.3795 - accuracy: 0.4988 - val_loss: 1.3096 - val_accu
racy: 0.5230
Epoch 4/5
1563/1563 [==============================] - 343s 219ms/step - loss: 1.2962 - accuracy: 0.5332 - val_loss: 1.2196 - val_accu
racy: 0.5589
Epoch 5/5
1563/1563 [==============================] - 340s 217ms/step - loss: 1.2410 - accuracy: 0.5519 - val_loss: 1.2445 - val_accu
racy: 0.5544
Accuracy: 55.44%
```

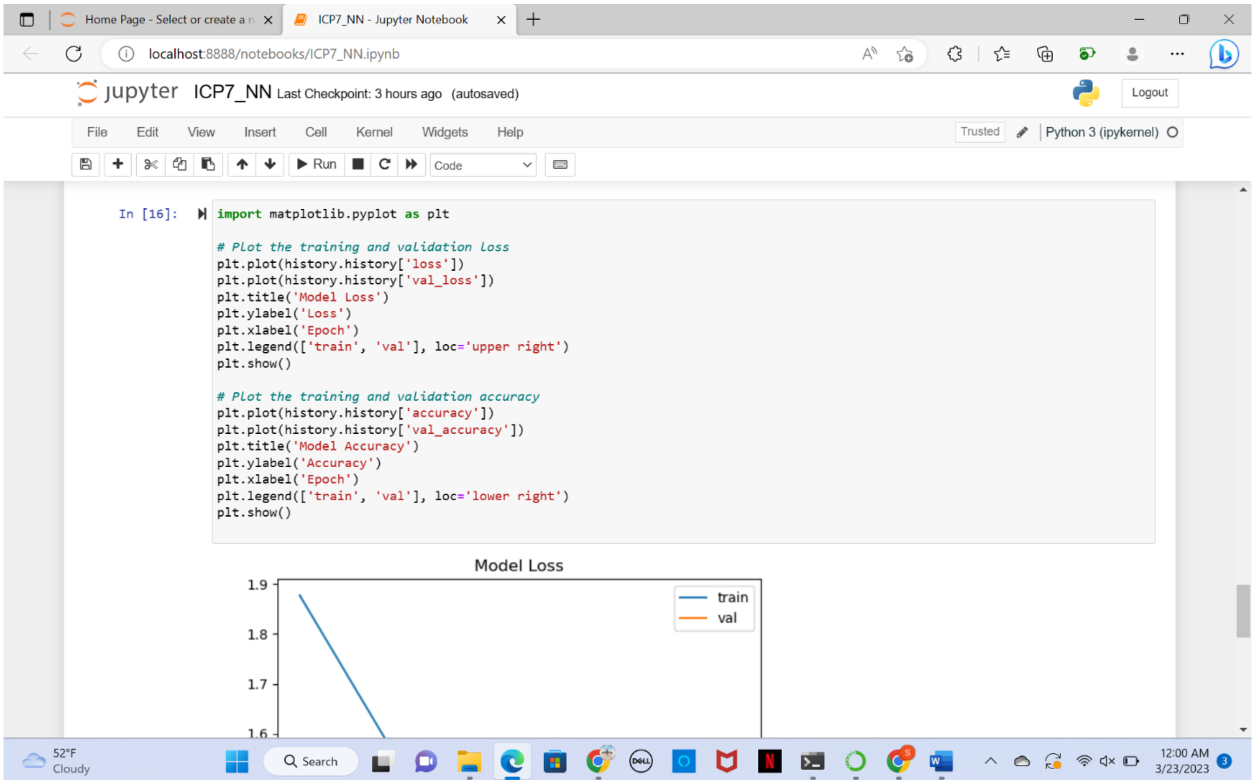
In [15]:

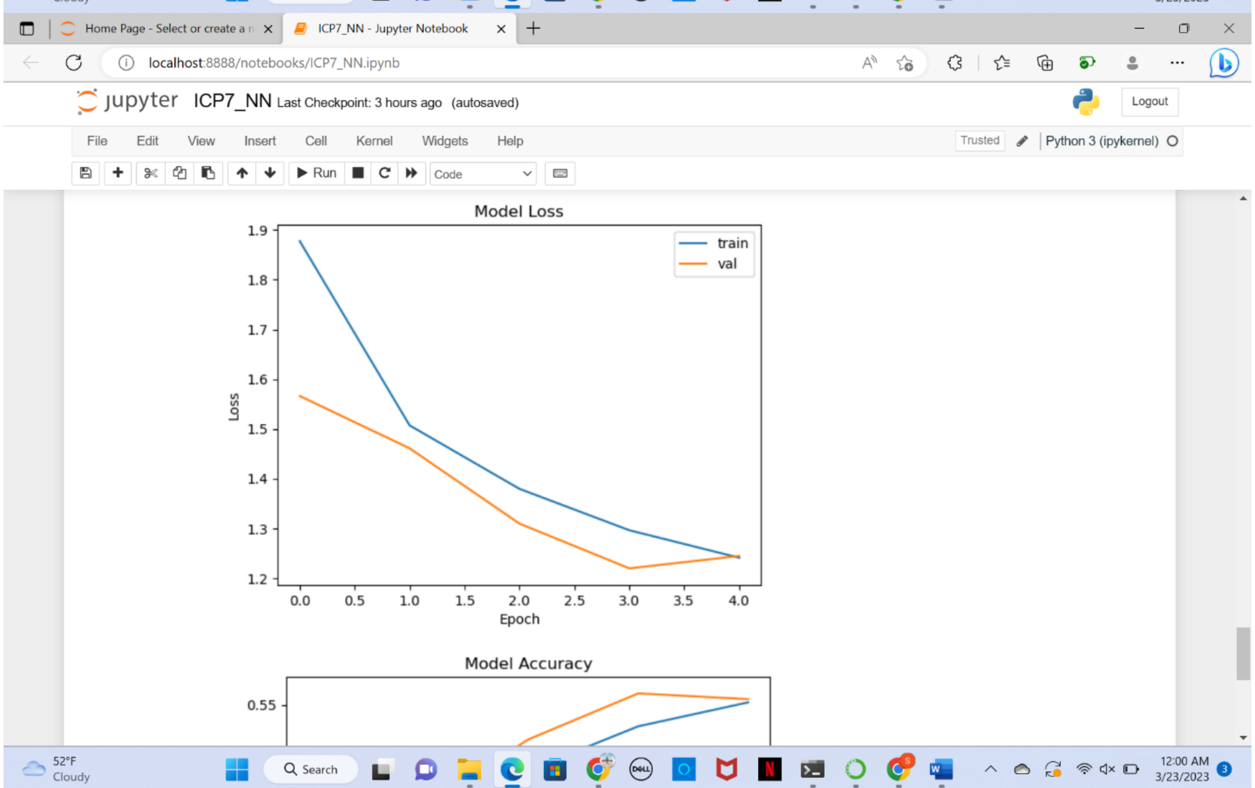
```python
# Predict the first 4 images of the test data
predictions = model.predict(X_test[:4])
# Convert the predictions to class labels
```

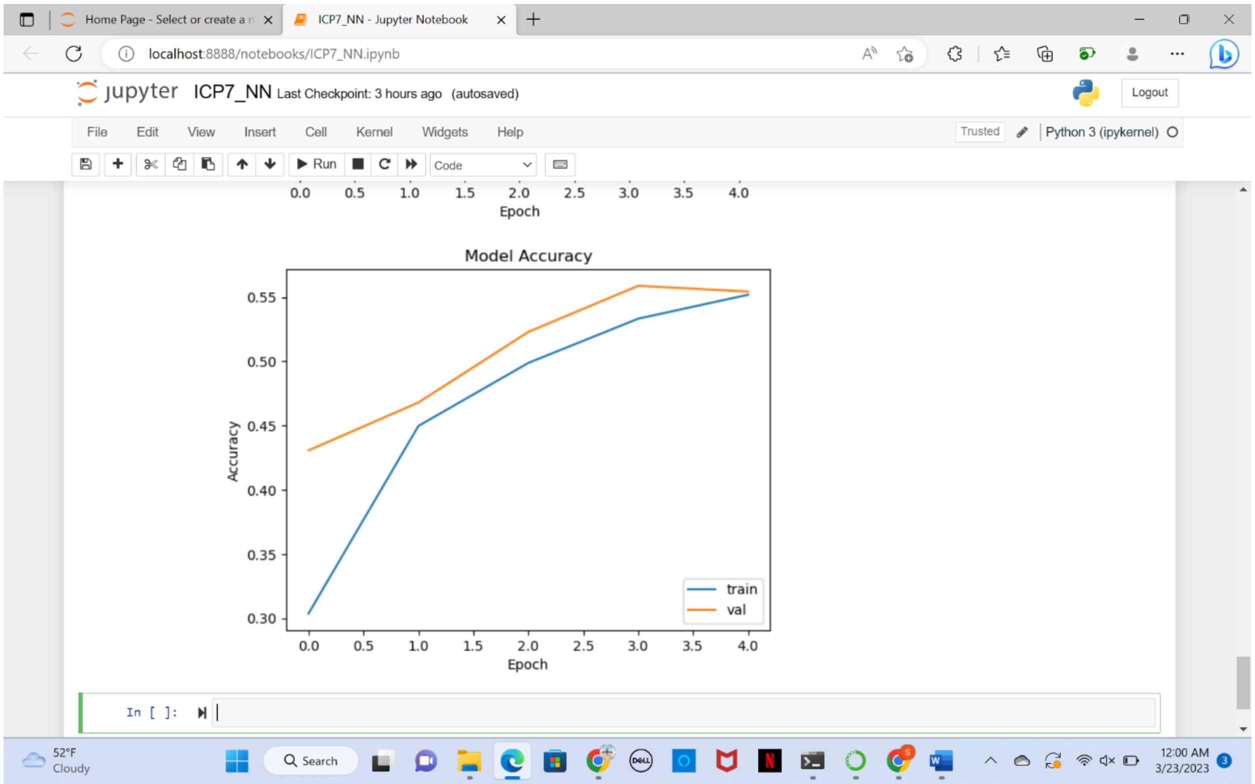
2. Predict the first 4 images of the test data using the above model. Then, compare with the actual label for those 4 images to check whether or not the model has predicted correctly.



3. Visualize Loss and Accuracy using the history object


Home Page - Select or create a n
ICP7_NN - Jupyter Notebook
localhost:8888/notebooks/ICP7_NN.ipynb
jupyter ICP7_NN Last Checkpoint: 3 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [16]:
import matplotlib.pyplot as plt
# Plot the training and validation loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['train', 'val'], loc='upper right')
plt.show()
# Plot the training and validation accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['train', 'val'], loc='lower right')
plt.show()
Model Loss
1.9
1.8
1.7
1.6
train
val
52°F
Cloudy
Search
12:00 AM
3/23/2023


Home Page - Select or create a n
ICP7_NN - Jupyter Notebook
localhost:8888/notebooks/ICP7_NN.ipynb
jupyter ICP7_NN Last Checkpoint: 3 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Model Loss
train
val
Loss
1.9
1.8
1.7
1.6
1.5
1.4
1.3
1.2
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0
Epoch
Model Accuracy
0.55
52°F
Cloudy
Search
12:00 AM
3/23/2023


Home Page - Select or create a n
ICP7_NN - Jupyter Notebook
localhost:8888/notebooks/ICP7_NN.ipynb
jupyter ICP7_NN Last Checkpoint: 3 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0
Epoch
Model Accuracy
Accuracy
0.55
0.50
0.45
0.40
0.35
0.30
train
val
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0
Epoch
In [ ]:
52°F
Cloudy
Search
12:00 AM
3/23/2023