Bryce Smith
Homework 3
STAA 578

The goal of the dataset was to accurately classify images from the CIFAR-10 dataset. In order to classify the images I developed a Convolutional Neural Network. To begin, I normalized the pixel values by converting to the range of 0 to 1 by dividing them by 255. To pre-process the data I one-hot encoded the class labels, which originally were listed as 0 to 9. The neural network model I created was a sequential model with four convolutional layers, two with 32 filters and two with 64 filters. Following each set of convolutional layers I added a max-pooling layer, a dropout layer with 0.25 dropout rate and s batch normalization layer. The activation function used for the convolutional layers was “ReLu”. After creating the convolutional, max-pool, dropout layers and batch normilzation layers, I added a Flatten layer to the model to convert the inputs into a one-dimensional vector. Lastly, I added a fully connected layer with 512 units and ReLu activation function, with another batch normalization and dropout layer. The last layer is the output layer with 10 units for the 10 classes and a softmax activation function for multi-class classification. After compiling the model with the RMSprop optimizer and categorical cross-entropy loss, I created an early stopping callback to monitor the validation loss. I set the patience to 3 in an effort to increse the accuracy on the test data. When training the model I used a batch size of 64, max 20 epochs and 20% of the training data for validation. Overall, after predicting on the test data, I determined this to be my best model and submitted to Kaggle. My score on the Kaggle leaderboard is a 0.76171.

BryceSmith_HW3

April 27, 2023

[ ]:
```
#load necessary packages
import tensorflow
from tensorflow.keras.datasets import cifar10
from tensorflow.keras import layers
from tensorflow.keras import models
from tensorflow.keras.utils import to_categorical
import matplotlib
import matplotlib.pyplot as plt
from keras import optimizers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,␣
  ↪Dropout, BatchNormalization
import numpy as np
import pandas as pd

(train_x, train_y), (test_x, test_y) = cifar10.load_data()

# Normalize pixel values
train_x = train_x.astype('float32') / 255
test_x = test_x.astype('float32') / 255

# One-hot encode the class labels
train_y = tensorflow.keras.utils.to_categorical(train_y, num_classes=10)
test_y = tensorflow.keras.utils.to_categorical(test_y, num_classes=10)

#Create sequential convolution neural network model
model = Sequential()

#add layers
model.add(Conv2D(32, (3, 3), activation='relu', padding='same',␣
  ↪input_shape=train_x.shape[1:]))
model.add(BatchNormalization())
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
```

```python
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

#Flatten and make fully connected
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))

#compile the model
model.compile(optimizer = optimizers.RMSprop(lr=1e-4), loss =␣
 ↪"categorical_crossentropy", metrics=['acc'])

#load early stopping
from tensorflow.keras.callbacks import EarlyStopping
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=3)

#fit the model with 20% validation and early stopping
history_cnn_maxpool = model.fit(train_x,
                                      train_y,
                                      epochs=20,
                                      batch_size=64,
                                      validation_split = 0.2,
                                      callbacks=[es])
#run predictions on test data
predictions = model.predict(test_x)
#produve test accuracy
test_acc_cnn_maxpool, test_acc_cnn_maxpool = model.evaluate(test_x, test_y)
print('CNN 1 (max pooling) test accuracy ', test_acc_cnn_maxpool)

predicted_classes = np.argmax(predictions, axis=1)

#create dataframe to submit prediction to kaggle in required format
submission_df = pd.DataFrame({'id': np.arange(1, len(predicted_classes) + 1)})

#add class to id
submission_df['class'] = predicted_classes

#create .csv with predictions
submission_df.to_csv('CNN_Submission.csv', index=False)
```