STAA Homework 3

Cody Minns

May 21, 2023

This project deals with classifying the CIFAR-10 dataset using convolutional neural networks (CNNs).

I explored using CNNs with both max pooling and different strides, but I decided to focus on max pooling in my final model. I used several different kernel sizes in my convolutional layers with the intention of picking up image features of varying scale. The data was then flattened and run through two dense layers before the output layer. I also employed batch normalization and dropout layers after every convolutional and dense layer. I explored different dropout rates and found the rate that gave the best model predictions. Finally, I explored varying the number of convolutional layers and decided that two layers for each kernel size gave optimal predictions. Models using dropout more sparingly and using L2 normalization were also created, but the model that gave the best predictions is shown below.

[1]:
```
import numpy as np
import pandas as pd
from tensorflow.keras import layers
from tensorflow.keras import models
from tensorflow.keras.utils import to_categorical as to_cat
from tensorflow.keras.callbacks import EarlyStopping
from keras.datasets import cifar10
```

[2]:
```
#load data
(train, train_labels), (test, test_labels) = cifar10.load_data()

#format data
train = train.astype('float32') / 255
test = test.astype('float32') / 255
train_labels = to_cat(train_labels)
test_labels = to_cat(test_labels)
```

[3]:
```
#define early stopping criteria
es = EarlyStopping(monitor='val_loss', mode='min',\
                   patience=4)
```

[4]:
```
#define cnn model
cnn_maxpool = models.Sequential()

cnn_maxpool.add(layers.Conv2D(32, (3, 3), activation = 'relu',\
                              padding = 'same', input_shape = (32, 32, 3)))
```

```
cnn_maxpool.add(layers.BatchNormalization())
cnn_maxpool.add(layers.Dropout(0.3))

cnn_maxpool.add(layers.Conv2D(32, (3, 3), activation = 'relu',\
                              padding = 'same'))
cnn_maxpool.add(layers.MaxPooling2D((2, 2)))
cnn_maxpool.add(layers.BatchNormalization())
cnn_maxpool.add(layers.Dropout(0.3))

cnn_maxpool.add(layers.Conv2D(64, (5, 5), padding = 'same',\
                              activation = 'relu'))
cnn_maxpool.add(layers.BatchNormalization())
cnn_maxpool.add(layers.Dropout(0.3))

cnn_maxpool.add(layers.Conv2D(64, (5, 5), padding = 'same',\
                              activation = 'relu'))
cnn_maxpool.add(layers.MaxPooling2D((2, 2)))
cnn_maxpool.add(layers.BatchNormalization())
cnn_maxpool.add(layers.Dropout(0.3))

cnn_maxpool.add(layers.Conv2D(128, (7, 7), padding = 'same',\
                              activation = 'relu'))
cnn_maxpool.add(layers.BatchNormalization())
cnn_maxpool.add(layers.Dropout(0.3))

cnn_maxpool.add(layers.Conv2D(128, (7, 7), padding = 'same',\
                              activation = 'relu'))
cnn_maxpool.add(layers.MaxPooling2D((2, 2)))
cnn_maxpool.add(layers.BatchNormalization())
cnn_maxpool.add(layers.Dropout(0.3))

cnn_maxpool.add(layers.Flatten())

cnn_maxpool.add(layers.Dense(128, activation = 'relu'))
cnn_maxpool.add(layers.BatchNormalization())
cnn_maxpool.add(layers.Dropout(0.3))

cnn_maxpool.add(layers.Dense(64, activation = 'relu'))
cnn_maxpool.add(layers.BatchNormalization())
cnn_maxpool.add(layers.Dropout(0.3))

cnn_maxpool.add(layers.Dense(10, activation = 'softmax'))
```

[5]:
```
#compile model
cnn_maxpool.compile(
    optimizer = 'rmsprop',
    loss = 'categorical_crossentropy',
```

```
    metrics = ['accuracy'])
```

[ ]:
```
#fit model
cnn_maxpool.fit(train,
                train_labels,
                epochs = 50,
                batch_size = 64,
                validation_split = 0.2,
                callbacks = [es])
```

[ ]:
```
#make predictions and write to file
preds = cnn_maxpool.predict(test)
preds = np.array([np.argmax(x) for x in preds])
ids = np.array(range(10000)) + 1
out = pd.DataFrame(np.transpose([ids, preds]),\
                   columns = ['id', 'class'])
out.to_csv('predictions_maxpool.csv', index = False)
```