Question 2 Hw3

April 18, 2022

[1]:
```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
```

[2]:
```
# Change to Markdown if GPU is not supported.


import os

os.environ["TF_CPP_MIN_LOG_LEVEL"] = "2"

physical_devices = tf.config.list_physical_devices("GPU")
tf.config.experimental.set_memory_growth(physical_devices[0], True)
```

[3]:
```
# You don't need to change this session
um_classes = 10
input_shape = (32, 32, 3)

(X_train, y_train), (X_test, y_test) = keras.datasets.cifar10.load_data()

print("x_train shape: {} - y_train shape: {}".format(X_train.shape,y_train.
 ↪shape))
print("x_test shape: {} - y_test shape: {}".format(X_test.shape,y_test.shape))

# Scale images to the [0, 1] range
X_train = X_train.astype("float32") / 255
X_test = X_test.astype("float32") / 255
# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, um_classes)
y_test = keras.utils.to_categorical(y_test, um_classes)
```

```
x_train shape: (50000, 32, 32, 3) - y_train shape: (50000, 1)
x_test shape: (10000, 32, 32, 3) - y_test shape: (10000, 1)
```

[46]:
```
# Designing the Custom Model


inputs = keras.Input(shape=(32, 32, 3))

x=layers.Conv2D(64, kernel_size=(3, 3))(inputs)
x=layers.Activation("relu")(x)
x=layers.BatchNormalization()(x)

x=layers.MaxPooling2D(pool_size=(2, 2))(x)
x=layers.Conv2D(128, kernel_size=(3, 3))(x)
x=layers.Activation("relu")(x)
x=layers.BatchNormalization()(x)

x=layers.Conv2D(256, kernel_size=(3, 3))(x)
x=layers.Activation("relu")(x)
x=layers.BatchNormalization()(x)

x=layers.Conv2D(512, kernel_size=(3, 3)(x)
x=layers.Activation("relu")(x)
x=layers.BatchNormalization()(x)

x=layers.MaxPooling2D(pool_size=(2, 2))(x)
x=layers.Flatten()(x)

x=layers.Dense(512, activation='relu')(x)
x=layers.Dense(256, activation='relu')(x)
x=layers.Dense(128, activation='relu')(x)
x=layers.Dense(64, activation='relu')(x)
x=layers.Dense(32, activation='relu')(x)

outputs=layers.Dense(um_classes, activation="softmax")(x)
```

[50]:
```
# Compiling the Model


model=keras.Model(inputs,outputs)
model.compile(loss="categorical_crossentropy", optimizer="adam",␣
 ↪metrics=["accuracy"])
model.summary()
```

```
Model: "model_11"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_12 (InputLayer)       [(None, 32, 32, 3)]       0
```

```
conv2d_43 (Conv2D)           (None, 30, 30, 64)        1792

activation_46 (Activation)   (None, 30, 30, 64)        0

batch_normalization_40 (Bat  (None, 30, 30, 64)        256
chNormalization)

max_pooling2d_25 (MaxPoolin  (None, 15, 15, 64)        0
g2D)

conv2d_44 (Conv2D)           (None, 13, 13, 128)       73856

activation_47 (Activation)   (None, 13, 13, 128)       0

batch_normalization_41 (Bat  (None, 13, 13, 128)       512
chNormalization)

conv2d_45 (Conv2D)           (None, 11, 11, 256)       295168

activation_48 (Activation)   (None, 11, 11, 256)       0

batch_normalization_42 (Bat  (None, 11, 11, 256)       1024
chNormalization)

conv2d_46 (Conv2D)           (None, 9, 9, 512)         1180160

activation_49 (Activation)   (None, 9, 9, 512)         0

batch_normalization_43 (Bat  (None, 9, 9, 512)         2048
chNormalization)

max_pooling2d_26 (MaxPoolin  (None, 4, 4, 512)         0
g2D)

flatten_11 (Flatten)         (None, 8192)              0

dense_62 (Dense)             (None, 512)               4194816

dense_63 (Dense)             (None, 256)               131328

dense_64 (Dense)             (None, 128)               32896

dense_65 (Dense)             (None, 64)                8256

dense_66 (Dense)             (None, 32)                2080

dense_67 (Dense)             (None, 10)                330
```

```
=================================================================
Total params: 5,924,522
Trainable params: 5,922,602
Non-trainable params: 1,920
_________________________________________________________________
```

[51]:
```
# Fitting the Model


history = model.fit(X_train, y_train, epochs=10, batch_size=32,␣
 ↪validation_split=0.2, verbose=1)
```

```
Epoch 1/10
1250/1250 [==============================] - 41s 32ms/step - loss: 0.1290 -
accuracy: 0.9605 - val_loss: 1.0593 - val_accuracy: 0.7571
Epoch 2/10
1250/1250 [==============================] - 30s 24ms/step - loss: 0.1041 -
accuracy: 0.9686 - val_loss: 1.2016 - val_accuracy: 0.7411
Epoch 3/10
1250/1250 [==============================] - 33s 26ms/step - loss: 0.0948 -
accuracy: 0.9719 - val_loss: 1.4311 - val_accuracy: 0.7143
Epoch 4/10
1250/1250 [==============================] - 30s 24ms/step - loss: 0.0928 -
accuracy: 0.9721 - val_loss: 1.1589 - val_accuracy: 0.7588
Epoch 5/10
1250/1250 [==============================] - 30s 24ms/step - loss: 0.0858 -
accuracy: 0.9746 - val_loss: 1.1024 - val_accuracy: 0.7541
Epoch 6/10
1250/1250 [==============================] - 29s 23ms/step - loss: 0.0832 -
accuracy: 0.9756 - val_loss: 1.1593 - val_accuracy: 0.7472
Epoch 7/10
1250/1250 [==============================] - 33s 26ms/step - loss: 0.0674 -
accuracy: 0.9810 - val_loss: 1.0693 - val_accuracy: 0.7667
Epoch 8/10
1250/1250 [==============================] - 23s 18ms/step - loss: 0.0775 -
accuracy: 0.9781 - val_loss: 1.2517 - val_accuracy: 0.7567
Epoch 9/10
1250/1250 [==============================] - 23s 18ms/step - loss: 0.0720 -
accuracy: 0.9793 - val_loss: 1.1480 - val_accuracy: 0.7464
Epoch 10/10
1250/1250 [==============================] - 23s 18ms/step - loss: 0.0667 -
accuracy: 0.9808 - val_loss: 1.2986 - val_accuracy: 0.7397
```

[52]:
```
# Final Scores


score = model.evaluate(X_test, y_test, verbose=0)
```

```
print("Test loss:", score[0])
print("Test error:", 1-score[1])
```

```
Test loss: 1.3165239095687866
Test error: 0.25950002670288086
```