PS9

January 23, 2023

1 Simplifed ResNet implementation

1.1 Loading CIFAR10

[11]:
```
import tensorflow as tf
import tensorflow.keras as keras
```

[12]:
```
from tensorflow.keras.datasets import cifar10

(X_train, y_train), (X_test, y_test) = cifar10.load_data()
```

[13]:
```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(50000, 32, 32, 3)
(10000, 32, 32, 3)
(50000, 1)
(10000, 1)
```

[14]:
```
# normalization
X_train, X_test = X_train/255.0, X_test/255.0
```

1.2 Construct Resnet

[20]:
```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.layers import ReLU
from tensorflow.keras.layers import Conv2D, MaxPool2D
from tensorflow.keras.layers import Add, AveragePooling2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
```

[21]:
```
inputs = Input(shape=(32,32,3))
x = Conv2D(32, kernel_size=(3,3), strides=(1,1),
           padding='same', use_bias=False)(inputs)
```

```
x = BatchNormalization()(x)
x = ReLU()(x)

x = Conv2D(32, kernel_size=(3,3), strides=(1,1),
           padding='same', use_bias=False)(x)
x = BatchNormalization()(x)
x = ReLU()(x)

x = MaxPool2D(pool_size=(2,2), strides=(2,2),
              padding='valid')(x)
x = Conv2D(64, kernel_size=(3,3), strides=(1,1),
           padding='same', use_bias=False)(x)
x = BatchNormalization()(x)
x = ReLU()(x)

# 1st skip connection
skip = x
x = Conv2D(64, kernel_size=(3,3), strides=(1,1),
           padding='same', use_bias=False)(x)
x = BatchNormalization()(x)
x = ReLU()(x)
x = Conv2D(64, kernel_size=(3,3), strides=(1,1),
           padding='same', use_bias=False)(x)
x = BatchNormalization()(x)

x = Add()([x, skip])
x = ReLU()(x)

x = MaxPool2D(pool_size=(2,2), strides=(2,2),
              padding='valid')(x)
x = Conv2D(128, kernel_size=(3,3), strides=(1,1),
           padding='same', use_bias=False)(x)
x = BatchNormalization()(x)
x = ReLU()(x)

# 2nd skip connection
skip = x
x = Conv2D(128, kernel_size=(3,3), strides=(1,1),
           padding='same', use_bias=False)(x)
x = BatchNormalization()(x)
x = ReLU()(x)
x = Conv2D(128, kernel_size=(3,3), strides=(1,1),
           padding='same', use_bias=False)(x)
x = BatchNormalization()(x)

x = Add()([x, skip])
x = ReLU()(x)
```

```python
# Average Pooling
x = AveragePooling2D(pool_size=(8,8))(x)
x = Flatten()(x)
x = ReLU()(x)
outputs = Dense(10, activation='softmax')(x)

Model_resnet = Model(inputs = inputs, outputs = outputs)

Model_resnet.summary()
```

```
Model: "model_1"
________________________________________________________________________________
__________________
 Layer (type)                   Output Shape         Param #     Connected to
================================================================================
==================
 input_3 (InputLayer)           [(None, 32, 32, 3)]  0           []

 conv2d_21 (Conv2D)             (None, 32, 32, 32)   864
['input_3[0][0]']

 batch_normalization_16 (BatchN  (None, 32, 32, 32)  128
['conv2d_21[0][0]']
 ormalization)

 re_lu_17 (ReLU)                (None, 32, 32, 32)   0
['batch_normalization_16[0][0]']

 conv2d_22 (Conv2D)             (None, 32, 32, 32)   9216
['re_lu_17[0][0]']

 batch_normalization_17 (BatchN  (None, 32, 32, 32)  128
['conv2d_22[0][0]']
 ormalization)

 re_lu_18 (ReLU)                (None, 32, 32, 32)   0
['batch_normalization_17[0][0]']

 max_pooling2d_7 (MaxPooling2D)  (None, 16, 16, 32)  0
['re_lu_18[0][0]']

 conv2d_23 (Conv2D)             (None, 16, 16, 64)   18432
['max_pooling2d_7[0][0]']

 batch_normalization_18 (BatchN  (None, 16, 16, 64)  256
['conv2d_23[0][0]']
```

```
 ormalization)

 re_lu_19 (ReLU)                (None, 16, 16, 64)   0
['batch_normalization_18[0][0]']

 conv2d_24 (Conv2D)             (None, 16, 16, 64)   36864
['re_lu_19[0][0]']

 batch_normalization_19 (BatchN  (None, 16, 16, 64)  256
['conv2d_24[0][0]']
 ormalization)

 re_lu_20 (ReLU)                (None, 16, 16, 64)   0
['batch_normalization_19[0][0]']

 conv2d_25 (Conv2D)             (None, 16, 16, 64)   36864
['re_lu_20[0][0]']

 batch_normalization_20 (BatchN  (None, 16, 16, 64)  256
['conv2d_25[0][0]']
 ormalization)

 add_4 (Add)                    (None, 16, 16, 64)   0
['batch_normalization_20[0][0]',
're_lu_19[0][0]']

 re_lu_21 (ReLU)                (None, 16, 16, 64)   0           ['add_4[0][0]']

 max_pooling2d_8 (MaxPooling2D)  (None, 8, 8, 64)    0
['re_lu_21[0][0]']

 conv2d_26 (Conv2D)             (None, 8, 8, 128)    73728
['max_pooling2d_8[0][0]']

 batch_normalization_21 (BatchN  (None, 8, 8, 128)   512
['conv2d_26[0][0]']
 ormalization)

 re_lu_22 (ReLU)                (None, 8, 8, 128)    0
['batch_normalization_21[0][0]']

 conv2d_27 (Conv2D)             (None, 8, 8, 128)    147456
['re_lu_22[0][0]']

 batch_normalization_22 (BatchN  (None, 8, 8, 128)   512
['conv2d_27[0][0]']
 ormalization)
 re_lu_23 (ReLU)                (None, 8, 8, 128)    0
['batch_normalization_22[0][0]']

 conv2d_28 (Conv2D)             (None, 8, 8, 128)    147456
['re_lu_23[0][0]']

 batch_normalization_23 (BatchN  (None, 8, 8, 128)   512
['conv2d_28[0][0]']
 ormalization)

 add_5 (Add)                    (None, 8, 8, 128)    0
['batch_normalization_23[0][0]',
're_lu_22[0][0]']

 re_lu_24 (ReLU)                (None, 8, 8, 128)    0           ['add_5[0][0]']

 average_pooling2d_2 (AveragePo  (None, 1, 1, 128)   0
['re_lu_24[0][0]']
 oling2D)

 flatten_2 (Flatten)            (None, 128)          0
['average_pooling2d_2[0][0]']

 re_lu_25 (ReLU)                (None, 128)          0
['flatten_2[0][0]']

 dense_4 (Dense)                (None, 10)           1290
['re_lu_25[0][0]']

================================================================================
==================
Total params: 474,730
Trainable params: 473,450
Non-trainable params: 1,280
--------------------------------------------------------------------------------
------------------
```

1.3 Compile

[22]:
```python
from tensorflow.keras.optimizers import Adam

opt = Adam(learning_rate = 0.001,
          beta_1 = 0.9,
          beta_2 = 0.999)

Model_resnet.compile(optimizer = opt,
                    loss='sparse_categorical_crossentropy',
                    metrics=['acc'])
```

[23]:
```
# training
Model_resnet.fit(X_train,y_train,epochs=20)
```

```
Epoch 1/20
1563/1563 [==============================] - 177s 113ms/step - loss: 1.1540 -
acc: 0.5860
Epoch 2/20
1563/1563 [==============================] - 182s 116ms/step - loss: 0.7484 -
acc: 0.7386
Epoch 3/20
1563/1563 [==============================] - 117s 75ms/step - loss: 0.5961 -
acc: 0.7937
Epoch 4/20
1563/1563 [==============================] - 108s 69ms/step - loss: 0.5001 -
acc: 0.8280
Epoch 5/20
1563/1563 [==============================] - 103s 66ms/step - loss: 0.4288 -
acc: 0.8515
Epoch 6/20
1563/1563 [==============================] - 105s 67ms/step - loss: 0.3645 -
acc: 0.8735
Epoch 7/20
1563/1563 [==============================] - 106s 68ms/step - loss: 0.3150 -
acc: 0.8908
Epoch 8/20
1563/1563 [==============================] - 102s 65ms/step - loss: 0.2649 -
acc: 0.9075
Epoch 9/20
1563/1563 [==============================] - 102s 65ms/step - loss: 0.2226 -
acc: 0.9228
Epoch 10/20
1563/1563 [==============================] - 103s 66ms/step - loss: 0.1861 -
acc: 0.9351
Epoch 11/20
1563/1563 [==============================] - 103s 66ms/step - loss: 0.1603 -
acc: 0.9428
Epoch 12/20
1563/1563 [==============================] - 108s 69ms/step - loss: 0.1362 -
acc: 0.9530
Epoch 13/20
1563/1563 [==============================] - 101s 65ms/step - loss: 0.1172 -
acc: 0.9588
Epoch 14/20
1563/1563 [==============================] - 107s 69ms/step - loss: 0.1071 -
acc: 0.9611
```

```
Epoch 15/20
1563/1563 [==============================] - 106s 68ms/step - loss: 0.0943 -
acc: 0.9666
Epoch 16/20
1563/1563 [==============================] - 101s 65ms/step - loss: 0.0892 -
acc: 0.9682
Epoch 17/20
1563/1563 [==============================] - 101s 65ms/step - loss: 0.0777 -
acc: 0.9726
Epoch 18/20
1563/1563 [==============================] - 108s 69ms/step - loss: 0.0726 -
acc: 0.9745
Epoch 19/20
1563/1563 [==============================] - 122s 78ms/step - loss: 0.0695 -
acc: 0.9761
Epoch 20/20
1563/1563 [==============================] - 119s 76ms/step - loss: 0.0653 -
acc: 0.9774
```

```
[23]: <keras.callbacks.History at 0x1db9920a820>
```

```
[24]: # Evaluation
      test_performance = Model_resnet.evaluate(X_test,y_test)
      print(test_performance)
```

```
313/313 [==============================] - 5s 16ms/step - loss: 0.8323 - acc:
0.8139
[0.8322671055793762, 0.8138999938964844]
```