N2Best

December 23, 2021

[ ]:
```
import tensorflow as tf
import tensorflow.keras as keras
import numpy as np
import matplotlib.pyplot as plt
```

[ ]:
```
(x_train, y_train), (x_test, y_test) = keras.datasets.cifar10.load_data()
x_train = x_train
x_test = x_test
y_train = keras.utils.to_categorical(y_train)
y_test = keras.utils.to_categorical(y_test)
```

[ ]:
```
import random

rand_selected = random.sample([x for x in range(0, len(x_train))], 30000)
x_train = x_train[rand_selected]
y_train = y_train[rand_selected]

print(f"shape of x_train is {tf.shape(x_train)}")
print(f"shape of y_train is {tf.shape(y_train)}")
print(f"shape of x_test is {tf.shape(x_test)}")
print(f"shape of y_test is {tf.shape(y_test)}")
```

```
shape of x_train is [30000    32    32     3]
shape of y_train is [30000    10]
shape of x_test is [10000    32    32     3]
shape of y_test is [10000    10]
```

[ ]:
```
fig = plt.figure()

for i in range(1, 5):
    ax = fig.add_subplot(5, 1, i)
    ax.imshow(x_train[i])
```

[ ]:
```
mean = np.mean(x_train, axis=(0, 1, 2, 3))
std = np.std(x_train, axis=(0, 1, 2, 3))
x_train = (x_train-mean)/(std+1e-7)
x_test = (x_test-mean)/(std+1e-7)
```

[ ]:
```
datagen = keras.preprocessing.image.ImageDataGenerator(
    featurewise_center=False,
    samplewise_center=False,
    featurewise_std_normalization=False,
    samplewise_std_normalization=False,
    zca_whitening=False,
    rotation_range=15,
    width_shift_range=0.1,
    horizontal_flip=True,
    vertical_flip=False
)
datagen.fit(x_train)
```

[ ]:
```
weight_decay = 0.0005
model = keras.models.Sequential([
    keras.layers.Conv2D(64, (3, 3), padding='same', kernel_regularizer=keras.
 ↪regularizers.l2(weight_decay), input_shape=x_train.shape[1:]),
    keras.layers.BatchNormalization(),
    keras.layers.Activation('relu'),
    keras.layers.Dropout(0.4),

    keras.layers.Conv2D(64, (3, 3), padding='same', kernel_regularizer=keras.
 ↪regularizers.l2(weight_decay)),
```

```
    keras.layers.BatchNormalization(),
    keras.layers.Activation('relu'),
    keras.layers.MaxPooling2D(pool_size=(2, 2)),

    keras.layers.Conv2D(128, (3, 3), padding='same', kernel_regularizer=keras.
 ↪regularizers.l2(weight_decay)),
    keras.layers.BatchNormalization(),
    keras.layers.Activation('relu'),
    keras.layers.Dropout(0.4),

    keras.layers.Conv2D(128, (3, 3), padding='same', kernel_regularizer=keras.
 ↪regularizers.l2(weight_decay)),
    keras.layers.BatchNormalization(),
    keras.layers.Activation('relu'),
    keras.layers.MaxPooling2D(pool_size=(2, 2)),

    keras.layers.Conv2D(256, (3, 3), padding='same', kernel_regularizer=keras.
 ↪regularizers.l2(weight_decay)),
    keras.layers.BatchNormalization(),
    keras.layers.Activation('relu'),

    keras.layers.Conv2D(256, (3, 3), padding='same', kernel_regularizer=keras.
 ↪regularizers.l2(weight_decay)),
    keras.layers.BatchNormalization(),
    keras.layers.Activation('relu'),
    keras.layers.Dropout(0.4),

    keras.layers.Flatten(),
    keras.layers.Dense(256, activation='relu', kernel_regularizer=keras.
 ↪regularizers.l2(weight_decay)),
    keras.layers.Dense(10, activation='softmax')
])

model.summary()

opt = keras.optimizers.Adam()
loss = keras.losses.CategoricalCrossentropy()
model.compile(loss=loss, optimizer=opt, metrics=['accuracy'])
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 64)        1792
_________________________________________________________________
batch_normalization (BatchNo (None, 32, 32, 64)        256
_________________________________________________________________
```

```
activation (Activation)      (None, 32, 32, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 64)        36928
_________________________________________________________________
batch_normalization_1 (Batch (None, 32, 32, 64)        256
_________________________________________________________________
activation_1 (Activation)    (None, 32, 32, 64)        0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 128)       73856
_________________________________________________________________
batch_normalization_2 (Batch (None, 16, 16, 128)       512
_________________________________________________________________
activation_2 (Activation)    (None, 16, 16, 128)       0
_________________________________________________________________
dropout_1 (Dropout)          (None, 16, 16, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 128)       147584
_________________________________________________________________
batch_normalization_3 (Batch (None, 16, 16, 128)       512
_________________________________________________________________
activation_3 (Activation)    (None, 16, 16, 128)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 128)         0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 256)         295168
_________________________________________________________________
batch_normalization_4 (Batch (None, 8, 8, 256)         1024
_________________________________________________________________
activation_4 (Activation)    (None, 8, 8, 256)         0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 8, 8, 256)         590080
_________________________________________________________________
batch_normalization_5 (Batch (None, 8, 8, 256)         1024
_________________________________________________________________
activation_5 (Activation)    (None, 8, 8, 256)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 8, 8, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 16384)             0
_________________________________________________________________
dense (Dense)                (None, 256)               4194560
_________________________________________________________________
dense_1 (Dense)              (None, 10)                2570
=================================================================
```

```
Total params: 5,346,122
Trainable params: 5,344,330
Non-trainable params: 1,792
_________________________________________________________________
```

[ ]:
```
model.fit(datagen.flow(x_train, y_train), epochs=40, verbose=2, batch_size=64,␣
↪validation_data=(x_test, y_test), workers=4)
```

```
Epoch 1/40
938/938 - 39s - loss: 2.4121 - accuracy: 0.2938 - val_loss: 1.9914 -
val_accuracy: 0.3903
Epoch 2/40
938/938 - 38s - loss: 1.6742 - accuracy: 0.4714 - val_loss: 1.5063 -
val_accuracy: 0.5260
Epoch 3/40
938/938 - 33s - loss: 1.4796 - accuracy: 0.5439 - val_loss: 1.4448 -
val_accuracy: 0.5721
Epoch 4/40
938/938 - 32s - loss: 1.3955 - accuracy: 0.5862 - val_loss: 1.5007 -
val_accuracy: 0.5701
Epoch 5/40
938/938 - 32s - loss: 1.3610 - accuracy: 0.6190 - val_loss: 1.8723 -
val_accuracy: 0.5083
Epoch 6/40
938/938 - 32s - loss: 1.3252 - accuracy: 0.6375 - val_loss: 1.5927 -
val_accuracy: 0.5828
Epoch 7/40
938/938 - 32s - loss: 1.2873 - accuracy: 0.6588 - val_loss: 1.2137 -
val_accuracy: 0.6919
Epoch 8/40
938/938 - 32s - loss: 1.2535 - accuracy: 0.6737 - val_loss: 1.3527 -
val_accuracy: 0.6572
Epoch 9/40
938/938 - 32s - loss: 1.2224 - accuracy: 0.6880 - val_loss: 1.2558 -
val_accuracy: 0.6967
Epoch 10/40
938/938 - 32s - loss: 1.2030 - accuracy: 0.7000 - val_loss: 1.3785 -
val_accuracy: 0.6545
Epoch 11/40
938/938 - 32s - loss: 1.1747 - accuracy: 0.7115 - val_loss: 1.2554 -
val_accuracy: 0.6910
Epoch 12/40
938/938 - 34s - loss: 1.1509 - accuracy: 0.7207 - val_loss: 1.0767 -
val_accuracy: 0.7512
Epoch 13/40
938/938 - 34s - loss: 1.1309 - accuracy: 0.7285 - val_loss: 1.2313 -
val_accuracy: 0.6964
Epoch 14/40
```

```
938/938 - 37s - loss: 1.1054 - accuracy: 0.7378 - val_loss: 1.0916 -
val_accuracy: 0.7470
Epoch 15/40
938/938 - 34s - loss: 1.0946 - accuracy: 0.7425 - val_loss: 1.1496 -
val_accuracy: 0.7286
Epoch 16/40
938/938 - 39s - loss: 1.0876 - accuracy: 0.7453 - val_loss: 1.0215 -
val_accuracy: 0.7685
Epoch 17/40
938/938 - 34s - loss: 1.0730 - accuracy: 0.7540 - val_loss: 1.0657 -
val_accuracy: 0.7614
Epoch 18/40
938/938 - 32s - loss: 1.0598 - accuracy: 0.7553 - val_loss: 0.9827 -
val_accuracy: 0.7838
Epoch 19/40
938/938 - 32s - loss: 1.0501 - accuracy: 0.7591 - val_loss: 1.0224 -
val_accuracy: 0.7670
Epoch 20/40
938/938 - 32s - loss: 1.0363 - accuracy: 0.7657 - val_loss: 1.0949 -
val_accuracy: 0.7499
Epoch 21/40
938/938 - 32s - loss: 1.0348 - accuracy: 0.7655 - val_loss: 1.0376 -
val_accuracy: 0.7679
Epoch 22/40
938/938 - 35s - loss: 1.0266 - accuracy: 0.7674 - val_loss: 1.2916 -
val_accuracy: 0.7093
Epoch 23/40
938/938 - 33s - loss: 1.0294 - accuracy: 0.7692 - val_loss: 1.0691 -
val_accuracy: 0.7610
Epoch 24/40
938/938 - 35s - loss: 1.0179 - accuracy: 0.7729 - val_loss: 1.1101 -
val_accuracy: 0.7509
Epoch 25/40
938/938 - 33s - loss: 1.0180 - accuracy: 0.7712 - val_loss: 1.0744 -
val_accuracy: 0.7618
Epoch 26/40
938/938 - 33s - loss: 1.0060 - accuracy: 0.7804 - val_loss: 1.2151 -
val_accuracy: 0.7315
Epoch 27/40
938/938 - 33s - loss: 1.0032 - accuracy: 0.7777 - val_loss: 0.9757 -
val_accuracy: 0.7904
Epoch 28/40
938/938 - 33s - loss: 1.0072 - accuracy: 0.7780 - val_loss: 0.9939 -
val_accuracy: 0.7885
Epoch 29/40
938/938 - 36s - loss: 0.9995 - accuracy: 0.7800 - val_loss: 1.0463 -
val_accuracy: 0.7741
Epoch 30/40
```

```
938/938 - 34s - loss: 0.9924 - accuracy: 0.7886 - val_loss: 1.0755 -
val_accuracy: 0.7621
Epoch 31/40
938/938 - 33s - loss: 0.9983 - accuracy: 0.7820 - val_loss: 0.9967 -
val_accuracy: 0.7861
Epoch 32/40
938/938 - 33s - loss: 0.9880 - accuracy: 0.7845 - val_loss: 1.1101 -
val_accuracy: 0.7472
Epoch 33/40
938/938 - 36s - loss: 0.9807 - accuracy: 0.7854 - val_loss: 1.0075 -
val_accuracy: 0.7874
Epoch 34/40
938/938 - 35s - loss: 0.9877 - accuracy: 0.7862 - val_loss: 0.9648 -
val_accuracy: 0.7959
Epoch 35/40
938/938 - 34s - loss: 0.9808 - accuracy: 0.7866 - val_loss: 1.0299 -
val_accuracy: 0.7715
Epoch 36/40
938/938 - 33s - loss: 0.9871 - accuracy: 0.7865 - val_loss: 0.9427 -
val_accuracy: 0.8026
Epoch 37/40
938/938 - 35s - loss: 0.9790 - accuracy: 0.7860 - val_loss: 1.1330 -
val_accuracy: 0.7492
Epoch 38/40
938/938 - 35s - loss: 0.9702 - accuracy: 0.7901 - val_loss: 0.9384 -
val_accuracy: 0.8055
Epoch 39/40
938/938 - 34s - loss: 0.9702 - accuracy: 0.7901 - val_loss: 0.9134 -
val_accuracy: 0.8072
Epoch 40/40
938/938 - 37s - loss: 0.9702 - accuracy: 0.7901 - val_loss: 0.8937 -
val_accuracy: 0.7931
```

[ ]:
```
model.evaluate(x_test, y_test)
```

```
313/313 [==============================] - 4s 12ms/step - loss: 0.9820 -
accuracy: 0.7931
```

[ ]:
```
[0.9820287227630615, 0.7930999994277954]
```