< Deep Learning - PART2 TF2 CNNs >

Ch 5. CNNs Workshop 3 - CIFAR10 : Image Classifier with Data Augmentation & Dropout

2021/10/01

[Reference] :

- Keras Documentation : **Train a simple deep CNN on the CIFAR10 small images dataset** https://keras.io/examples/cifar10_cnn/ (https://keras.io/examples/cifar10_cnn/)
- **CIFAR-10 and CIFAR-100 datasets** https://www.cs.toronto.edu/~kriz/cifar.html (https://www.cs.toronto.edu/~kriz/cifar.html)
- 李飛飛教授：Convolutional Neural Networks (CNNs / ConvNets) (https://cs231n.github.io/convolutional-networks/ (https://cs231n.github.io/convolutional-networks/))
- `tf.keras.layers.Conv2D` (https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D (https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D))

[NOTE] : Running the following code on Google Colab with GPU !!

Running TensorFlow 2.0 on Google Colab

In [2]:

```
from __future__ import print_function

import tensorflow as tf
print(tf.__version__)

from tensorflow.keras.datasets import cifar10
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
import os
```

```
2.1.0
```

Hyperparameters

In [0]:

```
batch_size = 32
num_classes = 10
epochs = 50
data_augmentation = True
num_predictions = 20

save_dir = os.path.join(os.getcwd(), 'saved_models')
model_name = 'keras_cifar10_trained_model.h5'
```

Loading data...

In [4]:

```
# The data, split between train and test sets:
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# Convert class vectors to binary class matrices.
y_train = tf.keras.utils.to_categorical(y_train, num_classes)
y_test = tf.keras.utils.to_categorical(y_test, num_classes)
```

```
x_train shape: (50000, 32, 32, 3)
50000 train samples
10000 test samples
```

Forward Propagation

- **Building a training model with `tf.keras`**

In [0]:

```
model = Sequential()
model.add(Conv2D(32, (3, 3), padding='same',
                 input_shape=x_train.shape[1:]))
model.add(Activation('relu'))
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes))
model.add(Activation('softmax'))
```

Backpropagation

In [0]:

```
# initiate RMSprop optimizer
opt = tf.keras.optimizers.RMSprop(learning_rate=0.0001, decay=1e-6)

# Let's train the model using RMSprop
model.compile(loss='categorical_crossentropy',
              optimizer=opt,
              metrics=['accuracy'])
```

Normalization

In [0]:

```
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
```

Data Augmentation

In [8]:

```
if not data_augmentation:
    print('Not using data augmentation.')
    model.fit(x_train, y_train,
              batch_size=batch_size,
              epochs=epochs,
              validation_data=(x_test, y_test),
              shuffle=True)
else:
    print('Using real-time data augmentation.')
    # This will do preprocessing and realtime data augmentation:
    datagen = ImageDataGenerator(
        featurewise_center=False,  # set input mean to 0 over the dataset
        samplewise_center=False,  # set each sample mean to 0
        featurewise_std_normalization=False,  # divide inputs by std of the
        samplewise_std_normalization=False,  # divide each input by its std
        zca_whitening=False,  # apply ZCA whitening
        zca_epsilon=1e-06,  # epsilon for ZCA whitening
        rotation_range=0,  # randomly rotate images in the range (degrees,
        # randomly shift images horizontally (fraction of total width)
        width_shift_range=0.1,
        # randomly shift images vertically (fraction of total height)
        height_shift_range=0.1,
        shear_range=0.,  # set range for random shear
        zoom_range=0.,  # set range for random zoom
        channel_shift_range=0.,  # set range for random channel shifts
        # set mode for filling points outside the input boundaries
        fill_mode='nearest',
        cval=0.,  # value used for fill_mode = "constant"
        horizontal_flip=True,  # randomly flip images
        vertical_flip=False,  # randomly flip images
        # set rescaling factor (applied before any other transformation)
        rescale=None,
        # set function that will be applied on each input
        preprocessing_function=None,
        # image data format, either "channels_first" or "channels_last"
        data_format=None,
        # fraction of images reserved for validation (strictly between 0 an
        validation_split=0.0)

    # Compute quantities required for feature-wise normalization
    # (std, mean, and principal components if ZCA whitening is applied).
    datagen.fit(x_train)

    # Fit the model on the batches generated by datagen.flow().
    model.fit_generator(datagen.flow(x_train, y_train,
                                     batch_size=batch_size),
                        epochs=epochs,
                        validation_data=(x_test, y_test),
                        workers=4)
```

```
Using real-time data augmentation.
WARNING:tensorflow:From <ipython-input-8-58db1f5097ea>:49: Model.
fit_generator (from tensorflow.python.keras.engine.training) is d
eprecated and will be removed in a future version.
```

```
Instructions for updating:
Please use Model.fit, which supports generators.
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
Train for 1563 steps, validate on 10000 samples
Epoch 1/50
1563/1563 [==============================] - 34s 22ms/step - los
s: 1.8522 - accuracy: 0.3156 - val_loss: 1.5686 - val_accuracy:
0.4282
Epoch 2/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 1.5820 - accuracy: 0.4212 - val_loss: 1.3962 - val_accuracy:
0.4895
Epoch 3/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 1.4621 - accuracy: 0.4668 - val_loss: 1.2855 - val_accuracy:
0.5424
Epoch 4/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 1.3848 - accuracy: 0.5029 - val_loss: 1.3087 - val_accuracy:
0.5283
Epoch 5/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 1.3199 - accuracy: 0.5261 - val_loss: 1.1952 - val_accuracy:
0.5807
Epoch 6/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 1.2646 - accuracy: 0.5492 - val_loss: 1.0823 - val_accuracy:
0.6243
Epoch 7/50
1563/1563 [==============================] - 32s 20ms/step - los
s: 1.2151 - accuracy: 0.5705 - val_loss: 1.0979 - val_accuracy:
0.6160
Epoch 8/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 1.1746 - accuracy: 0.5853 - val_loss: 1.0508 - val_accuracy:
0.6367
Epoch 9/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 1.1351 - accuracy: 0.6001 - val_loss: 0.9939 - val_accuracy:
0.6555
Epoch 10/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 1.1063 - accuracy: 0.6082 - val_loss: 1.0137 - val_accuracy:
0.6450
Epoch 11/50
1563/1563 [==============================] - 32s 20ms/step - los
s: 1.0778 - accuracy: 0.6211 - val_loss: 0.9684 - val_accuracy:
0.6643
Epoch 12/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 1.0528 - accuracy: 0.6297 - val_loss: 0.9301 - val_accuracy:
0.6730
Epoch 13/50
1563/1563 [==============================] - 32s 20ms/step - los
```

```
s: 1.0243 - accuracy: 0.6415 - val_loss: 0.9945 - val_accuracy:
0.6649
Epoch 14/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 1.0060 - accuracy: 0.6457 - val_loss: 0.8892 - val_accuracy:
0.6916
Epoch 15/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 0.9801 - accuracy: 0.6570 - val_loss: 0.8424 - val_accuracy:
0.7071
Epoch 16/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 0.9681 - accuracy: 0.6589 - val_loss: 0.8317 - val_accuracy:
0.7079
Epoch 17/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 0.9542 - accuracy: 0.6675 - val_loss: 0.8605 - val_accuracy:
0.6934
Epoch 18/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 0.9347 - accuracy: 0.6730 - val_loss: 0.8303 - val_accuracy:
0.7125
Epoch 19/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 0.9208 - accuracy: 0.6768 - val_loss: 0.7733 - val_accuracy:
0.7314
Epoch 20/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 0.9115 - accuracy: 0.6806 - val_loss: 0.8550 - val_accuracy:
0.7100
Epoch 21/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 0.9024 - accuracy: 0.6855 - val_loss: 0.7837 - val_accuracy:
0.7230
Epoch 22/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 0.8937 - accuracy: 0.6900 - val_loss: 0.7865 - val_accuracy:
0.7286
Epoch 23/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 0.8833 - accuracy: 0.6945 - val_loss: 0.7812 - val_accuracy:
0.7257
Epoch 24/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 0.8740 - accuracy: 0.6971 - val_loss: 0.7955 - val_accuracy:
0.7253
Epoch 25/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 0.8716 - accuracy: 0.6961 - val_loss: 0.7891 - val_accuracy:
0.7319
Epoch 26/50
1563/1563 [==============================] - 32s 20ms/step - los
s: 0.8597 - accuracy: 0.7020 - val_loss: 0.7382 - val_accuracy:
0.7421
Epoch 27/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 0.8568 - accuracy: 0.7059 - val_loss: 0.7289 - val_accuracy:
```

```
0.7510
Epoch 28/50
1563/1563 [==============================] - 32s 20ms/step - los
s: 0.8541 - accuracy: 0.7052 - val_loss: 0.7725 - val_accuracy:
0.7366
Epoch 29/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 0.8476 - accuracy: 0.7087 - val_loss: 0.7659 - val_accuracy:
0.7441
Epoch 30/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 0.8410 - accuracy: 0.7123 - val_loss: 0.7346 - val_accuracy:
0.7480
Epoch 31/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 0.8377 - accuracy: 0.7139 - val_loss: 0.7613 - val_accuracy:
0.7409
Epoch 32/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 0.8360 - accuracy: 0.7144 - val_loss: 0.7264 - val_accuracy:
0.7492
Epoch 33/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 0.8283 - accuracy: 0.7155 - val_loss: 0.7821 - val_accuracy:
0.7365
Epoch 34/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 0.8243 - accuracy: 0.7176 - val_loss: 0.7258 - val_accuracy:
0.7615
Epoch 35/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 0.8299 - accuracy: 0.7177 - val_loss: 0.7439 - val_accuracy:
0.7461
Epoch 36/50
1563/1563 [==============================] - 33s 21ms/step - los
s: 0.8227 - accuracy: 0.7201 - val_loss: 0.7150 - val_accuracy:
0.7599
Epoch 37/50
1563/1563 [==============================] - 32s 20ms/step - los
s: 0.8220 - accuracy: 0.7201 - val_loss: 0.7034 - val_accuracy:
0.7635
Epoch 38/50
1563/1563 [==============================] - 32s 20ms/step - los
s: 0.8213 - accuracy: 0.7209 - val_loss: 0.7193 - val_accuracy:
0.7582
Epoch 39/50
1563/1563 [==============================] - 32s 20ms/step - los
s: 0.8152 - accuracy: 0.7207 - val_loss: 0.7304 - val_accuracy:
0.7671
Epoch 40/50
1563/1563 [==============================] - 32s 20ms/step - los
s: 0.8172 - accuracy: 0.7227 - val_loss: 0.7036 - val_accuracy:
0.7684
Epoch 41/50
1563/1563 [==============================] - 32s 20ms/step - los
s: 0.8120 - accuracy: 0.7248 - val_loss: 0.6881 - val_accuracy:
0.7684
```

```
Epoch 42/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 0.8116 - accuracy: 0.7257 - val_loss: 0.7248 - val_accuracy:
0.7570
Epoch 43/50
1563/1563 [==============================] - 32s 20ms/step - los
s: 0.8085 - accuracy: 0.7258 - val_loss: 0.7283 - val_accuracy:
0.7587
Epoch 44/50
1563/1563 [==============================] - 31s 20ms/step - los
s: 0.8088 - accuracy: 0.7257 - val_loss: 0.7152 - val_accuracy:
0.7633
Epoch 45/50
1563/1563 [==============================] - 32s 20ms/step - los
s: 0.8068 - accuracy: 0.7241 - val_loss: 0.7186 - val_accuracy:
0.7553
Epoch 46/50
1563/1563 [==============================] - 32s 20ms/step - los
s: 0.8049 - accuracy: 0.7264 - val_loss: 0.7038 - val_accuracy:
0.7662
Epoch 47/50
1563/1563 [==============================] - 32s 20ms/step - los
s: 0.8063 - accuracy: 0.7248 - val_loss: 0.7500 - val_accuracy:
0.7517
Epoch 48/50
1563/1563 [==============================] - 32s 20ms/step - los
s: 0.7995 - accuracy: 0.7315 - val_loss: 0.6898 - val_accuracy:
0.7786
Epoch 49/50
1563/1563 [==============================] - 33s 21ms/step - los
s: 0.8005 - accuracy: 0.7287 - val_loss: 0.7452 - val_accuracy:
0.7496
Epoch 50/50
1563/1563 [==============================] - 32s 21ms/step - los
s: 0.8020 - accuracy: 0.7296 - val_loss: 0.7083 - val_accuracy:
0.7645
```

Saving the model...

In [9]:

```
# Save model and weights
if not os.path.isdir(save_dir):
    os.makedirs(save_dir)
model_path = os.path.join(save_dir, model_name)
model.save(model_path)
print('Saved trained model at %s ' % model_path)
```

```
Saved trained model at /content/saved_models/keras_cifar10_traine
d_model.h5
```

Evaluation

In [10]:

```
# Score trained model.
scores = model.evaluate(x_test, y_test, verbose=2)
print('Test loss:', scores[0])
print('Test accuracy:', scores[1])
```

```
10000/10000 - 1s - loss: 0.7083 - accuracy: 0.7645
Test loss: 0.708253707408905
Test accuracy: 0.7645
```