dl4cvhw05

运行并且确实训练课程中的DL模型，将结果的ACC和LOSS结果截图发过来，如果能够有曲线图，一定会获得加分。

In [2]:

```
# %load 基础_cifar10_序贯.py
'''Train a simple deep CNN on the CIFAR10 small images dataset.

It gets to 75% validation accuracy in 25 epochs, and 79% after 50 epochs.
(it's still underfitting at that point, though).
'''

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

```
/home/ian/installed/anaconda3/lib/python3.6/site-packages/h5py/__init
__.py:36: FutureWarning: Conversion of the second argument of issubdt
ype from `float` to `np.floating` is deprecated. In future, it will b
e treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

In [3]:

```
batch_size = 32
num_classes = 10
epochs = 100
data_augmentation = True
num_predictions = 20
save_dir = os.path.join(os.getcwd(), 'saved_models')
model_name = 'keras_cifar10_trained_model.h5'

# 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 = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-pytho
n.tar.gz (https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz)
170500096/170498071 [==============================] - 41s 0us/step
x_train shape: (50000, 32, 32, 3)
50000 train samples
10000 test samples
```

In [4]:

```
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'))

# initiate RMSprop optimizer
opt = keras.optimizers.rmsprop(lr=0.0001, decay=1e-6)

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

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 32, 32, 32)        896
_________________________________________________________________
activation_1 (Activation)    (None, 32, 32, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 30, 30, 32)        9248
_________________________________________________________________
activation_2 (Activation)    (None, 30, 30, 32)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 15, 15, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 15, 15, 64)        18496
_________________________________________________________________
activation_3 (Activation)    (None, 15, 15, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 13, 13, 64)        36928
_________________________________________________________________
activation_4 (Activation)    (None, 13, 13, 64)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 6, 6, 64)          0
_________________________________________________________________
```

```
flatten_1 (Flatten)          (None, 2304)              0

_________________________________________________________________
dense_1 (Dense)              (None, 512)               1180160
_________________________________________________________________
activation_5 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                5130
_________________________________________________________________
activation_6 (Activation)    (None, 10)                0
=================================================================
Total params: 1,250,858
Trainable params: 1,250,858
Non-trainable params: 0
_________________________________________________________________
```

In [13]:

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

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 dat
        samplewise_std_normalization=False,  # divide each input by its std
        zca_whitening=False,  # apply ZCA whitening
        rotation_range=0,  # randomly rotate images in the range (degrees, 0 to
        width_shift_range=0.1,  # randomly shift images horizontally (fraction
        height_shift_range=0.1,  # randomly shift images vertically (fraction o
        horizontal_flip=True,  # randomly flip images
        vertical_flip=False)  # randomly flip images

    # 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),
                        steps_per_epoch=10,
                        epochs=epochs,
                        validation_data=(x_test, y_test),
                        workers=4)
```

```
Using real-time data augmentation.
Epoch 1/100
10/10 [==============================] - 3s 350ms/step - loss: 2.3029
- acc: 0.1062 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 2/100
10/10 [==============================] - 4s 350ms/step - loss: 2.3023
- acc: 0.1281 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 3/100
10/10 [==============================] - 3s 325ms/step - loss: 2.3022
- acc: 0.1187 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 4/100
10/10 [==============================] - 3s 349ms/step - loss: 2.3028
- acc: 0.0563 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 5/100
10/10 [==============================] - 3s 349ms/step - loss: 2.3023
- acc: 0.1219 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 6/100
10/10 [==============================] - 3s 347ms/step - loss: 2.3026
- acc: 0.0938 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 7/100
```

```
10/10 [==============================] - 4s 363ms/step - loss: 2.3015
- acc: 0.1250 - val_loss: 2.3027 - val_acc: 0.1000
Epoch 8/100
10/10 [==============================] - 3s 343ms/step - loss: 2.3026
- acc: 0.0844 - val_loss: 2.3027 - val_acc: 0.1000
Epoch 9/100
10/10 [==============================] - 4s 362ms/step - loss: 2.3031
- acc: 0.0906 - val_loss: 2.3027 - val_acc: 0.1000
Epoch 10/100
10/10 [==============================] - 4s 357ms/step - loss: 2.3031
- acc: 0.0906 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 11/100
10/10 [==============================] - 3s 330ms/step - loss: 2.3035
- acc: 0.0875 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 12/100
10/10 [==============================] - 4s 354ms/step - loss: 2.3018
- acc: 0.1062 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 13/100
10/10 [==============================] - 3s 333ms/step - loss: 2.3022
- acc: 0.0938 - val_loss: 2.3027 - val_acc: 0.1000
Epoch 14/100
10/10 [==============================] - 3s 346ms/step - loss: 2.3033
- acc: 0.0844 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 15/100
10/10 [==============================] - 4s 350ms/step - loss: 2.3026
- acc: 0.1094 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 16/100
10/10 [==============================] - 4s 356ms/step - loss: 2.3012
- acc: 0.1187 - val_loss: 2.3027 - val_acc: 0.1000
Epoch 17/100
10/10 [==============================] - 4s 357ms/step - loss: 2.3022
- acc: 0.0844 - val_loss: 2.3027 - val_acc: 0.1000
Epoch 18/100
10/10 [==============================] - 4s 353ms/step - loss: 2.3020
- acc: 0.1313 - val_loss: 2.3027 - val_acc: 0.1000
Epoch 19/100
10/10 [==============================] - 3s 341ms/step - loss: 2.3034
- acc: 0.0781 - val_loss: 2.3027 - val_acc: 0.1000
Epoch 20/100
10/10 [==============================] - 3s 346ms/step - loss: 2.3017
- acc: 0.1031 - val_loss: 2.3027 - val_acc: 0.1000
Epoch 21/100
10/10 [==============================] - 4s 363ms/step - loss: 2.3027
- acc: 0.0813 - val_loss: 2.3027 - val_acc: 0.1000
Epoch 22/100
10/10 [==============================] - 3s 337ms/step - loss: 2.3034
- acc: 0.0687 - val_loss: 2.3027 - val_acc: 0.1000
Epoch 23/100
10/10 [==============================] - 3s 340ms/step - loss: 2.3039
- acc: 0.0625 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 24/100
10/10 [==============================] - 3s 329ms/step - loss: 2.3025
- acc: 0.1031 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 25/100
10/10 [==============================] - 3s 348ms/step - loss: 2.3034
- acc: 0.1187 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 26/100
10/10 [==============================] - 4s 356ms/step - loss: 2.3023
- acc: 0.0906 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 27/100
10/10 [==============================] - 4s 353ms/step - loss: 2.3029
```

- acc: 0.1062 - val_loss: 2.3026 - val_acc: 0.1000

```
Epoch 28/100
10/10 [==============================] - 4s 369ms/step - loss: 2.3029
- acc: 0.1156 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 29/100
10/10 [==============================] - 4s 362ms/step - loss: 2.3028
- acc: 0.0969 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 30/100
10/10 [==============================] - 4s 362ms/step - loss: 2.3027
- acc: 0.1062 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 31/100
10/10 [==============================] - 4s 355ms/step - loss: 2.3030
- acc: 0.1062 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 32/100
10/10 [==============================] - 4s 357ms/step - loss: 2.3025
- acc: 0.0938 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 33/100
10/10 [==============================] - 4s 357ms/step - loss: 2.3021
- acc: 0.1062 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 34/100
10/10 [==============================] - 4s 359ms/step - loss: 2.3019
- acc: 0.1031 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 35/100
10/10 [==============================] - 3s 348ms/step - loss: 2.3027
- acc: 0.0969 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 36/100
10/10 [==============================] - 4s 359ms/step - loss: 2.3038
- acc: 0.0844 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 37/100
10/10 [==============================] - 4s 351ms/step - loss: 2.3006
- acc: 0.1156 - val_loss: 2.3027 - val_acc: 0.1000
Epoch 38/100
10/10 [==============================] - 3s 333ms/step - loss: 2.3032
- acc: 0.0719 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 39/100
10/10 [==============================] - 4s 362ms/step - loss: 2.3032
- acc: 0.1031 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 40/100
10/10 [==============================] - 3s 339ms/step - loss: 2.3031
- acc: 0.0844 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 41/100
10/10 [==============================] - 3s 350ms/step - loss: 2.3033
- acc: 0.0656 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 42/100
10/10 [==============================] - 4s 357ms/step - loss: 2.3039
- acc: 0.0625 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 43/100
10/10 [==============================] - 4s 353ms/step - loss: 2.3036
- acc: 0.0656 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 44/100
10/10 [==============================] - 3s 346ms/step - loss: 2.3030
- acc: 0.0813 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 45/100
10/10 [==============================] - 3s 334ms/step - loss: 2.3028
- acc: 0.0875 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 46/100
10/10 [==============================] - 4s 350ms/step - loss: 2.3023
- acc: 0.0906 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 47/100
10/10 [==============================] - 4s 362ms/step - loss: 2.3031
- acc: 0.0969 - val_loss: 2.3026 - val_acc: 0.1000
```

```
Epoch 48/100
10/10 [==============================] - 4s 352ms/step - loss: 2.3028
- acc: 0.0969 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 49/100
10/10 [==============================] - 4s 360ms/step - loss: 2.3027
- acc: 0.0906 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 50/100
10/10 [==============================] - 4s 364ms/step - loss: 2.3026
- acc: 0.0750 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 51/100
10/10 [==============================] - 4s 368ms/step - loss: 2.3028
- acc: 0.0875 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 52/100
10/10 [==============================] - 4s 352ms/step - loss: 2.3021
- acc: 0.1156 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 53/100
10/10 [==============================] - 3s 342ms/step - loss: 2.3030
- acc: 0.0750 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 54/100
10/10 [==============================] - 3s 328ms/step - loss: 2.3028
- acc: 0.0844 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 55/100
10/10 [==============================] - 3s 343ms/step - loss: 2.3026
- acc: 0.1000 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 56/100
10/10 [==============================] - 4s 358ms/step - loss: 2.3026
- acc: 0.0875 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 57/100
10/10 [==============================] - 3s 339ms/step - loss: 2.3029
- acc: 0.0969 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 58/100
10/10 [==============================] - 3s 342ms/step - loss: 2.3030
- acc: 0.1094 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 59/100
10/10 [==============================] - 3s 339ms/step - loss: 2.3023
- acc: 0.1094 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 60/100
10/10 [==============================] - 3s 347ms/step - loss: 2.3031
- acc: 0.0813 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 61/100
10/10 [==============================] - 3s 323ms/step - loss: 2.3026
- acc: 0.0813 - val_loss: 2.3026 - val_acc: 0.1000

Epoch 62/100
10/10 [==============================] - 3s 342ms/step - loss: 2.3017
- acc: 0.1719 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 63/100
10/10 [==============================] - 3s 332ms/step - loss: 2.3025
- acc: 0.0906 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 64/100
10/10 [==============================] - 3s 341ms/step - loss: 2.3026
- acc: 0.1031 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 65/100
10/10 [==============================] - 4s 352ms/step - loss: 2.3029
- acc: 0.1094 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 66/100
10/10 [==============================] - 4s 356ms/step - loss: 2.3029
- acc: 0.1000 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 67/100
10/10 [==============================] - 4s 357ms/step - loss: 2.3024
- acc: 0.1219 - val_loss: 2.3026 - val_acc: 0.1000
```

```
Epoch 68/100

10/10 [==============================] - 4s 355ms/step - loss: 2.3028
- acc: 0.1000 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 69/100
10/10 [==============================] - 4s 351ms/step - loss: 2.3024
- acc: 0.1062 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 70/100
10/10 [==============================] - 3s 334ms/step - loss: 2.3026
- acc: 0.1281 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 71/100
10/10 [==============================] - 3s 334ms/step - loss: 2.3030
- acc: 0.0969 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 72/100
10/10 [==============================] - 4s 352ms/step - loss: 2.3023
- acc: 0.1125 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 73/100
10/10 [==============================] - 3s 347ms/step - loss: 2.3028
- acc: 0.0813 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 74/100
10/10 [==============================] - 3s 345ms/step - loss: 2.3030
- acc: 0.0969 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 75/100
10/10 [==============================] - 3s 327ms/step - loss: 2.3028
- acc: 0.0656 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 76/100
10/10 [==============================] - 4s 353ms/step - loss: 2.3028
- acc: 0.1000 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 77/100
10/10 [==============================] - 3s 340ms/step - loss: 2.3028
- acc: 0.1125 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 78/100
10/10 [==============================] - 3s 339ms/step - loss: 2.3027
- acc: 0.0969 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 79/100
10/10 [==============================] - 3s 348ms/step - loss: 2.3024
- acc: 0.1125 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 80/100
10/10 [==============================] - 4s 355ms/step - loss: 2.3027
- acc: 0.1062 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 81/100
10/10 [==============================] - 4s 358ms/step - loss: 2.3027
- acc: 0.0781 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 82/100
10/10 [==============================] - 4s 364ms/step - loss: 2.3023
- acc: 0.1125 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 83/100
10/10 [==============================] - 4s 356ms/step - loss: 2.3020
- acc: 0.1187 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 84/100
10/10 [==============================] - 4s 353ms/step - loss: 2.3030
- acc: 0.0750 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 85/100
10/10 [==============================] - 4s 354ms/step - loss: 2.3026
- acc: 0.1125 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 86/100
10/10 [==============================] - 4s 357ms/step - loss: 2.3023
- acc: 0.1062 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 87/100
10/10 [==============================] - 4s 357ms/step - loss: 2.3031
- acc: 0.0844 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 88/100
```

```
10/10 [==============================] - 4s 353ms/step - loss: 2.3032
- acc: 0.0875 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 89/100
10/10 [==============================] - 4s 357ms/step - loss: 2.3026
- acc: 0.1219 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 90/100
10/10 [==============================] - 4s 352ms/step - loss: 2.3031
- acc: 0.0656 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 91/100
10/10 [==============================] - 4s 357ms/step - loss: 2.3026
- acc: 0.1156 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 92/100
10/10 [==============================] - 4s 353ms/step - loss: 2.3023
- acc: 0.1219 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 93/100
10/10 [==============================] - 4s 359ms/step - loss: 2.3026
- acc: 0.1156 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 94/100
10/10 [==============================] - 4s 353ms/step - loss: 2.3023
- acc: 0.1094 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 95/100
10/10 [==============================] - 3s 349ms/step - loss: 2.3022
- acc: 0.1156 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 96/100
10/10 [==============================] - 4s 353ms/step - loss: 2.3025
- acc: 0.1000 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 97/100
10/10 [==============================] - 4s 353ms/step - loss: 2.3026
- acc: 0.1031 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 98/100
10/10 [==============================] - 4s 351ms/step - loss: 2.3026
- acc: 0.1062 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 99/100
10/10 [==============================] - 4s 350ms/step - loss: 2.3024
- acc: 0.1125 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 100/100
10/10 [==============================] - 4s 358ms/step - loss: 2.3023
- acc: 0.1281 - val_loss: 2.3026 - val_acc: 0.1000
```

In [14]:

```
# 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)

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

```
Saved trained model at /home/ian/code/github/LSCJcourses/dl4cv/第5课/
saved_models/keras_cifar10_trained_model.h5
10000/10000 [==============================] - 3s 329us/step
Test loss: 2.302606481933594
Test accuracy: 0.1
```