Practicum IV: Image search and classification using Deep learning

已開始： 12月17日 11:45

測驗說明

Practicum IV: Image search and classification using Deep learning

(Not required in ML class, but encouraged to practice after the course finishes)

We have four objectives in this practicum:

1. According to the experiment in paper **ImageNet Classification with Deep Convolutional Neural Networks (https://papers.nips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf)**, we can **design a convolutional neural network to classify** an image into a specific class.
2. More importantly, as shown in the following figure from the paper, given any input image, the corresponding **feature vector learned from the last hidden layer of the CNN** can be used to find similar images in the training data set by calculating the Euclidean distance between any two feature vectors. In the last section, we will develop an image searching engine
3. Therefore, the feature vectors derived from the deep learning network can also be used for **image searching**.
4. We will extend our algorithms to develop an **image search web server**.

Note:

- This in-class hands-on activity reflects the key contents in Chapter 14 of the textbook and course slides in week 13
- Please follow the tutorial to practice the codes.
- We suggest everyone practice the codes and submit the results. You can copy texts/upload images from Google Colab and paste into the textbox.
- You can always leave this survey anytime without clicking submit button. Canvas will automatically save your work for latter use.

Practicum IV: Image search and classification using Deep learning

(Not required in ML class, but encouraged to practice after the course finishes)

We have four objectives in this practicum:

1. According to the experiment in paper [ImageNet Classification with Deep Convolutional Neural Networks (https://papers.nips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf)](https://papers.nips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf), we can **design a convolutional neural network to classify** an image into a specific class.
2. More importantly, as shown in the following figure from the paper, given any input image, the corresponding **feature vector learned from the last hidden layer of the CNN** can be used to find similar images in the training data set by calculating the Euclidean distance between any two feature vectors. In the last section, we will develop an image searching engine
3. Therefore, the feature vectors derived from the deep learning network can also be used for **image searching**.
4. We will extend our algorithms to develop an **image search web server**.

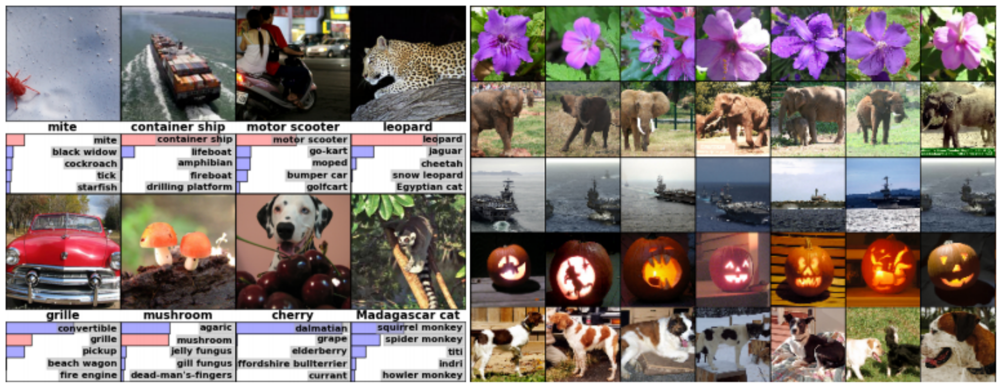


Figure 4: **(Left)** Eight ILSVRC-2010 test images and the five labels considered most probable by our model. The correct label is written under each image, and the probability assigned to the correct label is also shown with a red bar (if it happens to be in the top 5). **(Right)** Five ILSVRC-2010 test images in the first column. The remaining columns show the six training images that produce feature vectors in the last hidden layer with the smallest Euclidean distance from the feature vector for the test image.

In this practicum, we have in total of 10 tasks for everyone to complete. Please try your best to complete them.

問題 1 **5 分數**

Part 1. Design a convolutional neural network for image classification

(Please choose GPU on Google Colab for the following analysis)

Step 1: Data Loading

First of all, we can work on the popular dataset called 'CIFAR10', containing 32x32 color images of 10 categories of objects. Below are labels for the objects - airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

We can directly load the dataset into memory from module 'keras.datasets,' which contains:

Training Set: 50000 images, dimension shape: 50000 * 32 * 32 *3

Test Set: 10000 images, dimension shape: 10000 * 32 * 32 *3

```
from keras.datasets import cifar10
import matplotlib.pyplot as plt
import numpy as np

(X_train, y_train), (X_test, y_test) = cifar10.load_data()
y_train, y_test = y_train.reshape(-1, 1), y_test.reshape(-1, 1)

labels_map = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'ho
rse', 'ship', 'truck']
```

Task: Write codes to print out the data shape for the variables: X_train, y_train, X_test, y_test. Copy/Paste your output in this box

編輯 檢視 插入 格式 工具 表格

12pt 段落

p

0 個字

問題 2

5 分數

Step 2: Data Visualization

Using matplotlib, you can visualize the numpy data as image as follows:

```
fig = plt.figure(figsize=(20, 5))

for ii in range(10):
   jj = np.random.choice(range(len(X_train))) # randomly choose one image
   label = [labels_map[j] for j in y_train[jj]][-1]  # get its label
   imgplot = fig.add_subplot(2,5,ii+1)  ## set up grid of 2 rows and 5 columns fo
r visualization
   imgplot.imshow(X_train[jj]) # visualize the image
   imgplot.set_title(label, fontsize=20) # set the title
   imgplot.axis('off')
```

Task 2: Upload your visualization image in this box.

編輯 檢視 插入 格式 工具 表格

12pt 段落

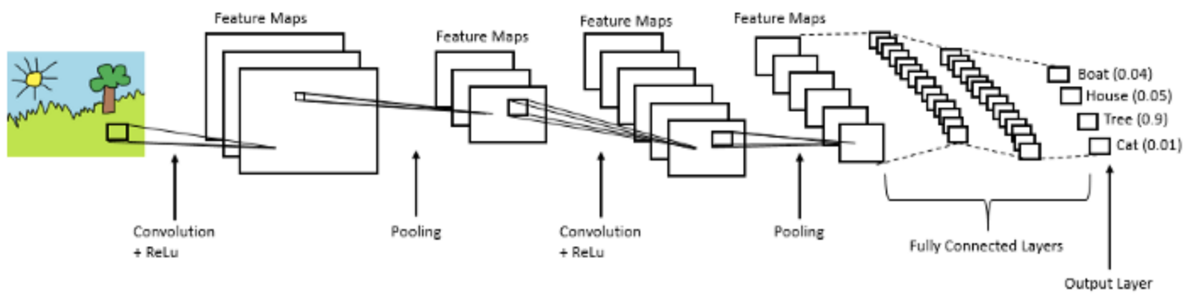
p

0 個字

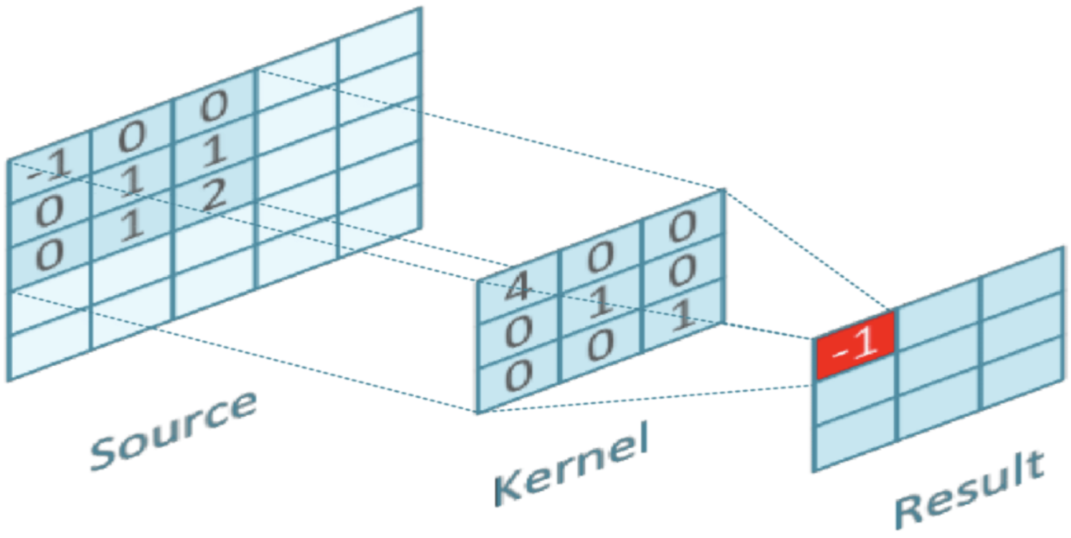
Step 3: Define model: Train the CNN network (use GPU on Google Colab)

Let's first review our previous demo to train a deep learning network**Practicum: class practice for neural network training (https://canvas.slu.edu/courses/28036/assignments/227318?wrap=1) (https://canvas.slu.edu/courses/28036/quizzes/46064)**

Introduction to Convolutional Neural Network



We will talk about the convolution algorithm in the next topics. Here, we need to know that 2D-CNN accepts a matrix as an input instead of a single-dimensional vector as the standard neural network does.



CNN will use a small-size matrix called 'kernel' (i.e., 3x3) to slide over all the image locations by following the directions of left to right and up to down. The kernel matrix will multiply the window within the input feature matrix in each location and then sum up values. The sum value will be treated as the convolution results and saved in the corresponding location in the new feature map'.

問題 3 **20 分數**

Step 4: Feature scaling before feeding into the CNN

```
num_classes = len(labels_map)
from keras.utils.np_utils import to_categorical


## Step 4.1: convert to one-hot encoding for classification
y_train_categorical = to_categorical(y_train, num_classes)
y_test_categorical = to_categorical(y_test, num_classes)

## Step 4.2: Feature Min-Max normalization
```

```
X_train_s = X_train.astype('float32')/255
X_test_s = X_test.astype('float32')/255
```

Task 1: write codes to check the dimension of the following variables:

1. X_train
2. X_test
3. X_train_s
4. X_test_s
5. y_train
6. y_test
7. y_train_categorical
8. y_test_categorical

Task 2: write codes to visualize the pixel distributions using histogram for variables X_train, X_train_s. Upload your images in this box. You can refer to codes in Step 7 of [Week 10: class practice for neural network training (https://canvas.slu.edu/courses/28036/quizzes/46064)](https://canvas.slu.edu/courses/28036/quizzes/46064)

編輯 檢視 插入 格式 工具 表格

12pt 段落

p 0 個字

問題 4

10 分數

Step 5: Define the CNN architecture

Task 1: Read the codes below step-by-step, assign appropriate number of neurons in the following highlighted layer.

```
### Step 5.1: Load the Keras class
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split
from keras.layers import Conv2D,MaxPooling2D
from keras import regularizers
from keras.layers import Flatten,BatchNormalization,Dropout,Activation
from keras.callbacks import EarlyStopping, ModelCheckpoint

### Step 5.2: Define early stop
keras_callbacks = [
 EarlyStopping(monitor='val_loss', patience=5, mode='min', min_delta=0.0001),
 ModelCheckpoint('./checkmodel.h5', monitor='val_loss', save_best_only=True, mode
='min')
]


### Step 5.3: Define the CNN architecture

model = Sequential()

### Step 5.3.1: define First convolution block
model.add(Conv2D( Enter Your Neurons, (3, 3), padding='same', input_shape=(32,32,
3)))
model.add(Activation('relu'))

### Step 5.3.2: define Second convolution block
model.add(Conv2D(Enter Your Neurons, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

### Step 5.3.3: define Third convolution block
model.add(Conv2D(96, (3, 3), padding='same'))
model.add(Activation('relu'))


### Step 5.3.4: define Fourth convolution block
model.add(Conv2D(96, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

### Step 5.3.5: define flatten layer, this will convert 2D dimension matrix into
1D array for each image
model.add(Flatten())

### Step 5.3.6: define fully-connected layer
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))

### Step 5.3.7:Your task, set appropriate number of hiddhen neurons. This may aff
ect your dimension of feature vectors for image searching
model.add(Dense(Enter Your Neurons (try 50-100)))
model.add(Activation('relu'))

### Step 5.3.8: Define the last classification layer with softmax function
model.add(Dense(num_classes, activation='softmax'))

### Step 5.3.9: compile the model
model.compile(loss = "categorical_crossentropy", optimizer = "adam", metrics =
['accuracy'])
```

Task 2: print out the model summary, upload the image of architecture. And report how many total trainable parameters you have in your model.

編輯 檢視 插入 格式 工具 表格

12pt 段落

p 0 個字

問題 5

10 分數

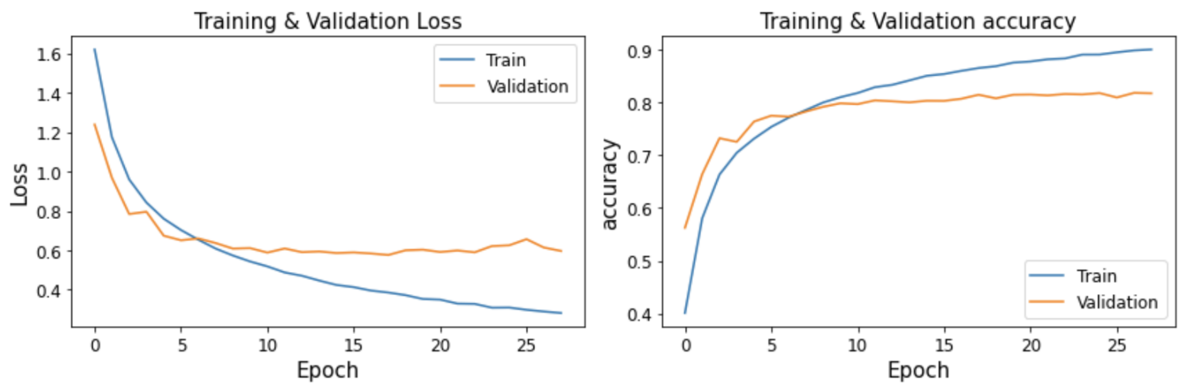
Step 6: Start training the CNN model

```
# Step 6: training the architecture, input is normalized data
history_log = model.fit(X_train_s, y_train_categorical, validation_split=0.1, bat
ch_size=64, epochs = 30, callbacks=keras_callbacks)
```

Task 1: Upload your training process.

Task 2: Visualize the learning curves for your model. Refer to step 11 in Practicum: class practice for neural network training (https://canvas.slu.edu/courses/28036/assignments/227318?wrap=1)

Sample image is attached:



編輯 檢視 插入 格式 工具 表格

12pt 段落

p

0 個字

問題 6

10 分數

Step 7: Evaluate the CNN model

Task: write codes to evaluate the classification accuracy in training set and testing set. Copy/paste your codes here, and report the accuracy results.

Note: Make sure to use normalized data as input to the pre-trained model (as the model is trained on the normalized data)

編輯 檢視 插入 格式 工具 表格

12pt 段落

p

0 個字

問題 7

5 分數

Part II: Image search using deep learning

Given any input image, use the corresponding **feature vector derived from the last hidden layer of the CNN** to find similar images in the training data set by calculating the Euclidean distance between any two feature vectors.

Let's look at the model's architecture again:

```
## print out the model summary
model.summary()
```

Task: print out the model architecture, and describe which layer will be used for feature extraction.

編輯 檢視 插入 格式 工具 表格

12pt 段落

p

0 個字

問題 8

5 分數

Step 8: Feature extraction from CNN model

Let's create a function to get the last hidden layers from the model for feature extraction.

```
### Step 8.1: get all layers of the model and save into a list
model_layers = model.layers
print(model_layers)
```

```
[<keras.layers.convolutional.Conv2D at 0x7f5e359c4a50>,
 <keras.layers.core.activation.Activation at 0x7f5e359c4ad0>,
 <keras.layers.convolutional.Conv2D at 0x7f5e351a9c10>,
 <keras.layers.core.activation.Activation at 0x7f5e351a2910>,
 <keras.layers.pooling.MaxPooling2D at 0x7f5e359d56d0>,
 <keras.layers.core.dropout.Dropout at 0x7f5e359c99d0>,
 <keras.layers.convolutional.Conv2D at 0x7f5e373f31d0>,
 <keras.layers.core.activation.Activation at 0x7f5e359c5b90>,
 <keras.layers.convolutional.Conv2D at 0x7f5e359ccf50>,
 <keras.layers.core.activation.Activation at 0x7f5e359cc6d0>,
 <keras.layers.pooling.MaxPooling2D at 0x7f5e369ed350>,
 <keras.layers.core.dropout.Dropout at 0x7f5e369eddd0>,
 <keras.layers.core.flatten.Flatten at 0x7f5e359cc7d0>,
 <keras.layers.core.dense.Dense at 0x7f5e369dced0>,
 <keras.layers.core.activation.Activation at 0x7f5e369e7d10>,
 <keras.layers.core.dropout.Dropout at 0x7f5e359d53d0>,
 <keras.layers.core.dense.Dense at 0x7f5eae63aa10>,
 <keras.layers.core.activation.Activation at 0x7f5e369dc1d0>,
 <keras.layers.core.dense.Dense at 0x7f5e373b5f90>]
```

```
###Step 8.2: Let's select last second layer as the output for feature extractor
# model.layers[-2].output corresponds to the output of last second layer
print(model.layers[-2].output)


###Step 8.3: Let's select the first layer as the input of feature extractor
# model.layers[0].input corresponds to the input of the first layer
print(model.layers[0].input)
```

```
###Step 8.3: Then we can build a feature extractor function to map the in
put to feature vectors
from keras import backend as K
Get_Hidden_Layered_Output = K.function([model.layers[0].input], [model.la
yers[-2].output])
```

```
### Step 8.4: Let's apply the feature extractor on one image, and check t
he feature vector
image_sample = X_train_s[0].reshape(1,32,32,3)
image_sample_outfeatures = Get_Hidden_Layered_Output(image_sample)[0]

print("extracted_features:", image_sample_outfeatures)
print("shape of original features:", image_sample.shape)
print("shape of extracted features:", image_sample_outfeatures.shape)
```

Task: Describe the activities in this step. What's the data dimension of the extracted features from your CNN model.

編輯 檢視 插入 格式 工具 表格

12pt 段落

p 0 個字

問題 9

15 分數

Step 9: Apply the feature extractor on all training and test images

```
# Step 9.1: Apply the feature extractor on all training images
X_train_featureVector = []
batch_size = 500 # set batch size to avoid memory issues
for i in range(int(X_train_s.shape[0]/batch_size)):
    X_train_featureVector.append(Get_Hidden_Layered_Output([X_train_s[i*batch_siz
e:(i+1)*batch_size,:,:,:]])[0])

X_train_featureVector = np.concatenate(X_train_featureVector,axis=0)
```

```
# Step 9.2: Apply the feature extractor on all testing images
X_test_featureVector = []
batch_size = 500
for i in range(int(X_test_s.shape[0]/batch_size)):
   X_test_featureVector.append(Get_Hidden_Layered_Output([X_test_s[i*batch_size:
(i+1)*batch_size,:,:,:]])[0])
X_test_featureVector = np.concatenate(X_test_featureVector,axis=0)
```

Task 1: Report the data dimensions of variables: X_train_featureVector, and X_test_featureVector.

Task 2: Describe your solutions using these transformed features for test images to find similar samples in the training data.

Task 3: Could we apply these transformed features to other ML approaches, such as KNN, softmax, decision trees, random forests or other classification techniques? Compared to using the original flattened features (32*32*3), would you expect better performance? Feel free to pick any of the above ML approaches to see the classification accuracy.

編輯　檢視　插入　格式　工具　表格

12pt　段落

p　　0 個字

問題 10

15 分數

Step 10: Design algorithms to find the similar images using euclidean distance.

You can follow the instructions below:

1. Randomly select one image from the test set
2. Predict the class for the image
3. Get the hidden feature vector for the image
4. Calculate the euclidean distance between the hidden feature vector of the image and hidden feature vectors of all images in the training set
5. Select the top 10 images have smallest Euclidean distance
6. Visualization of the searching results

A similar application is what we have done in KNN algorithm, as described in **Homework04: Build ML workflow for image classification** **(https://canvas.slu.edu/courses/28036/assignments/209584?wrap=1)**
Sample demo: **https://huggingface.co/spaces/SLU-CSCI4750/demo4_imageclassification** **(https://huggingface.co/spaces/SLU-CSCI4750/demo4_imageclassification)**
Task: create a similar image search tool based on distance calculation. Upload your notebook
Requirement:
Input: test image
Output: a list of images in training set that are most similar to the input image

編輯 檢視 插入 格式 工具 表格

12pt 段落

p

0 個字 </>

未保存 提交測驗