BÀI THỰC HÀNH 3: MẠNG NEURAL TÍCH CHẬP

Hướng dẫn nộp bài: Nộp file jupyter notebook (đuôi là .jpynb), đặt tên là MSSV_BaiThucHanh3.jpynb + file PDF báo cáo trả lời các câu hỏi: MSSV_BaiThucHanh3.pdf
Nộp qua course, giảng viên sẽ tạo submission sau mỗi buổi học.

1. Các lớp trong mạng neural tích chập.

+ Lớp Convolutional (kernel).
+ Lớp Pooling.
+ Lớp Dense (Fully connected).

a) Sử dụng CONV layer trong Keras: Conv2D gồm các thông số quan trọng sau:

filters: số lượng filter dùng trong lớp tích chập.

kernel_size: kích thước filter.

strides: bước trượt.

padding: thêm padding hay không. Nếu là valid: thì không dùng padding, nếu là same thì sẽ thêm padding sao cho kích thước đầu vào bằng với kích thước đầu ra (xem thêm về padding trong bài giảng).

activation: hàm kích hoạt.

b) Sử dụng lớp POOLING trong Keras: gồm có 3 dạng pooling là MaxPooling và AveragePoooling.

Ghi chú: Nếu sử dụng CONV2D thì cũng phải sử dụng MaxPooling2D

Các thông số quan trọng:

pool_size: Kích thước của pooling.

stride: bước trượt.

padding: thêm padding hay không. Nếu là valid: thì không dùng padding, nếu là same thì sẽ thêm padding sao cho kích thước đầu vào bằng với kích thước đầu ra (xem thêm về padding trong bài giảng).

c) Sử dụng lớp FullyConnected trong Keras: lớp Dense. Ý nghĩa các tham số tương tự các bài thực hành trước.

d) ***Dropout***: sử dụng lớp Dropout trong keras với tham số là p.

2. Chuẩn bị dữ liệu.

Bộ dữ liệu: MNIST (bộ dữ liệu nhận dạng chữ số viết tay)

PS: Bộ dữ liệu này được hỗ trợ mặc định trong thư viện keras

```
from keras.datasets.mnist import load_data
(X_train, y_train), (X_test, y_test) = load_data()
```

Chia tập train ra làm 2 phần: 90% cho huấn luyện và 10% cho validation (development). Sử dụng hàm train_test_split trong thư viện sklearn.

```
from sklearn.model_selection import train_test_split

X_train, X_dev, y_train, y_dev = train_test_split(X_train,
y_train, test_size=0.1)
```

Khi sử dụng tích chập, shape của lớp tích chập sẽ là (N, N, f), với N là số lượng filter và f là channel của ảnh. Như vậy, 1 bức ảnh trong bộ dữ liệu MNIST ban đầu sẽ phải có dạng là (28, 28, 1). Như vậy, ta phải "reshape" kích thuớc ảnh về dạng (28, 28,1).

Kích thước ban đầu: **(28, 28)** ==> **(28, 28, 1)**

Dùng **expand_dims** trong numpy để reshape bức ảnh (thêm vào 1 chiều).

Với tập huấn luyện ban đầu X_train sẽ có kích thước là: **(60000, 28, 28)**, ta thêm vào 1 chiều nữa ở **cột số 3** bằng cách dùng lệnh:

```
X_train_expanded = np.expand_dims(X_train, axis=3)
```

Thực hiện tương tự cho **X_dev** và **X_test**.

Chuẩn bị y_train và y_dev:

```
from tensorflow.keras.utils import to_categorical

y_train_new = to_categorical(y_train, num_classes=10)
y_dev_new = to_categorical(y_dev, num_classes = 10)
```

3. Xây dựng mô hình mạng tích chập.

Kiến trúc mô hình:

Input (28, 28, 1)

||

CONV2D: 32 filter kích thước 3x3, padding là valid, activation là relu

||

MaxPooling2D: kích thước là 2x2

||

CONV2D: 64 filter kích thước là 3x3, activation là relu

||

MaxPooling2D: kích thước là 2x2

||

Flatten: làm phẳng

||

Fully Conncected (FC): activation là softmax

Các kỹ thuật tối ưu:

+ Hàm los: CategoricalCrossentropy

+ Optimizer: Adam

```
model = Sequential()
model.add(Input(shape=(28, 28, 1)))
model.add(Conv2D(32, padding="valid", kernel_size=(3, 3),
activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, kernel_size=(3, 3),
activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(10, activation="softmax"))

optimizer = Adam(learning_rate=0.01)
loss = CategoricalCrossentropy()

model.compile(optimizer=optimizer, loss=loss,
metrics=['accuracy'])
```

Kết quả:

```
Model: "sequential_1"
_________________________________________________________________
```

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1600)              0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                16010
=================================================================
Total params: 34,826
Trainable params: 34,826
Non-trainable params: 0
```

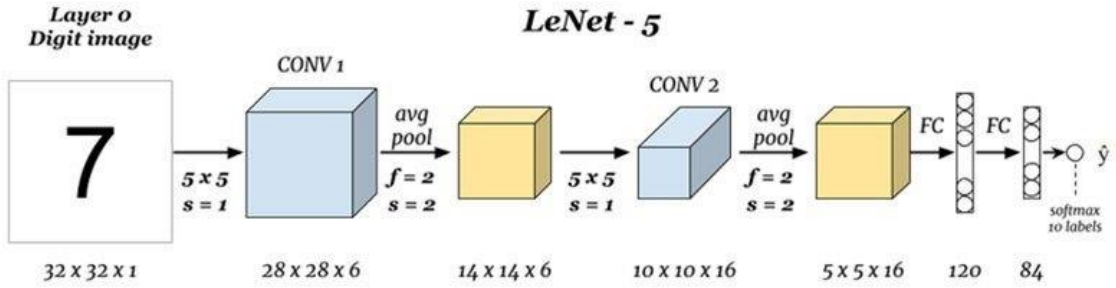
Huấn luyện mô hình với batch_size là 128 và số lượng epoch là 20.

```
history = model.fit(X_train_expanded, y_train_new,
              batch_size=128, epochs=20,
              validation_data=(X_dev_expanded,y_dev_new))
```

Kết quả thu được:

Accuracy: 96.88.

4. **Kiến trúc LeNET-5**:



STT	Tên layer	Shape	Layer Size	#Param
1	INPUT	32x32x1	1,024	0
2	CONV1 (f=5, s=1)	28x28x6	4,704	156
3	POOL	14x14x6	1,176	0
4	CONV2 (f=2, s=2)	10x10x16	1,600	2,416
5	POOL	5x5x16	400	0
6	FC	120x1	120	48,120
7	FC	84x1	84	10,164
8	Softmax	10x1	10	850
Tổng tham số				61,706

(các bạn xem lại slides để hiểu rõ hơn về mô hình LeNET-5 nhé)

Xây dựng mô hình:

```
model = Sequential()
model.add(Conv2D(filters=6, kernel_size=(5,5),
padding='same', activation='relu',input_shape=(28,28,1)))
model.add(AveragePooling2D(strides=2, pool_size=(2, 2)))
model.add(Conv2D(filters=16, kernel_size=(5,5),
padding='valid', activation='relu'))
model.add(AveragePooling2D(strides=2, pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(120, activation='relu'))
model.add(Dense(84, activation='relu'))
model.add(Dense(10, activation='softmax'))

optimizer = Adam(learning_rate=0.01)
loss = CategoricalCrossentropy()

model.compile(optimizer=optimizer, loss=loss,
metrics=['accuracy'])
```

Huấn luyện:

+ Optimizer: Adam.

+ Loss: CategoricalCrossentropy.

+ Batch_size: 128

+ Epoch: 30

BÀI TẬP:

Bài 1: Hiện thực mô hình ở phần 3, sử dụng hàm activation relu cho 2 lớp CONV. Thực hiện huấn luyện lại mô hình và xem kết quả? Cho biết độ chính xác và đồ thị học của mô hình.

Bài 2: Hiện thực mô hình ở phần 3, sử dụng hàm activation tanh cho 2 lớp CONV. Thực hiện huấn luyện lại mô hình và xem kết quả? Cho biết độ chính xác và đồ thị học của mô hình.

Bài 3: Thực hiện như bài 1, nhưng thêm vào lớp Dropout với giá trị p = 0.5 trước lớp Dense. Huấn luyện lại mô hình và cho biết kết quả (vẽ đồ thị học của mô hình).

Bài 4: Hiện thực lại mô hình **LeNET-5** theo hướng dẫn ở mục 4. Cho biết kết quả độ chính xác.

Bài 5: Lưu lại mô hình ở bài 3 thành file h5.

Bài 6: *Hiện thực mạng **AlexNET** với bộ dữ liệu MNIST

Thông tin mạng nerual như sau:

STT	**Tên**	**Thông số**	**Hàm activation**
1	CONV2D	num filter: 96 size: 12x12 input shape: (28,28,1) padding: same strides: 4	relu
2	MaxPooling2D	size: 3x3 pading: same strides: 2	
3	CONV2D	num filter: 256 size: 5x5 strides: 1	relu
4	MaxPooling2D	size: 3x3 pading: same strides: 2	
5	CONV2D	num filter: 384	relu

		size: 3x3 padding: same strides: 1	
6	CONV2D	num filter: 384 size: 3x3 padding: same strides: 1	relu
7	CONV2D	num filter: 256 size: 3x3 padding: same strides: 1	relu
8	MaxPooling2D	size: 3x3 pading: same strides: 2	
9	Dropout	p = 0.5	
10	Dense	num unit = 4096	relu
11	Dense	num unit = 4096	relu
12	Dense	num unit = 10	softmax

Loss: CategoricalCrossentropy

Optimizer: Adam

Bài 7: *Thực hiện các yêu cầu trên đối với bộ dữ liệu CIFAIR10