BÀI THỰC HÀNH 1: XÂY DỰNG MẠNG NEURAL ĐƠN GIẢN

Hướng dẫn nộp bài: Nộp file jupyter notebook (đuôi là .jpynb), đặt tên là MSSV_BaiThucHanh1.jpynb + + file PDF báo cáo trả lời các câu hỏi: MSSV_BaiThucHanh1.pdf
Nộp qua course, giảng viên sẽ tạo submission sau mỗi buổi học.

1. Load bộ dữ liệu:

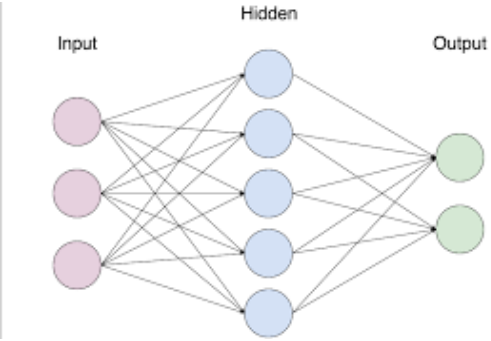
Bộ dữ liệu: MNIST (bộ dữ liệu nhận dạng chữ số viết tay)

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()
```

2. Kiến trúc mạng neural cơ bản:



3. Chuẩn bị dữ liệu

Xét input của bộ dữ liệu:

+ Tập train: gồm 60K bức ảnh, mỗi bức ảnh có kích thước là 28x28.

+ Tập test: gồm 10K bức ảnh, mỗi bức ảnh có kích thước là 28x28.

Xem kích thước của dữ liệu: sử dụng shape trong python

```
X_train.shape
```

Reshape kích thước của dữ liệu: Đưa về dạng (m, 784), với m là số lượng dữ liệu.

Sử dụng lệnh reshape trong numpy

```
X_train_reshaped = X_train.reshape(-1, 784)
```

Ghi chú: 28 x 28 = 784. Như vậy, mỗi điểm dữ liệu sẽ là một vector 784 chiều

=> Input sẽ là: X = $\begin{bmatrix} x_1 \\ \cdots \\ x_{784} \end{bmatrix}$

Đầu ra của dữ liệu sẽ có giá trị từ 0 -> 9 (ứng với 10 chữ số viết tay). Như vậy, đây là một bài toán phân lớp đa lớp (multi-class classification). Mỗi điểm dữ liệu sẽ được phân loại vào một trong 10 lớp tương ứng. Như vậy, ta cần xây dựng một binary class matrix ứng với số lượng nhãn bằng hàm to_catergorical như sau:

```
from tensorflow.keras.utils import to_categorical

y_train = to_categorical(y_train, num_classes = 10)
```

4. Xây dựng mạng neural bằng keras.

Keras là một thư việc cho phép xây dựng các kiến trúc deep learning một cách hiệu quả và tiết kiệm chi phí. Keras hỗ trợ sẵn các thư viện và các hàm dùng để xây dựng và huấn luyện một mạng neural học sâu.

Mạng neural học sâu sẽ gồm nhiều lớp, các lớp sẽ được kết nối với nhau để truyền thông tin. Để kết nối các lớp với nhau, keras hỗ trợ thư viện Sequential để "stack" các lớp (layer) lại với nhau.

Khởi tạo một sequential model:

```
from keras.models import Sequential

model = Sequential()
```

Thêm một lớp (layers) vào mô hình.

Với dữ liệu trước đó, một điểm dữ liệu sẽ có số lượng thuộc tính là 784. Do đó, ta sử dụng một lớp Fully connected gồm 784 unit như sau:

```
Dense(784, input_shape=(784, ))
```

Trong đó, *input_shape* là chiều (shape) của dữ liệu đầu vào.

Ghi chú: lớp *Densce* có tham số activation để xác định hàm kích hoạt nào cần dùng cho mỗi lớp. Mặc định của activation là None - tương ứng với hàm tuyến tính (linear).

Để thêm vào mô hình, ta dùng hàm add của Sequential:

```
model.add(Dense(784, input_shape=(784, )))
```

Lớp đầu ra sẽ là xác suất ứng với mỗi nhãn (xem thêm về bài toán phân loại đối với MNIST). Do đó, lớp này sẽ có 10 units

```
model.add(Dense(10, input_shape=(10, )))
```

Như vậy, mô hình chỉ gồm 2 layer: một lớp hidden layer và một lớp output layer. Để xem mô hình đã xây dựng, ta dùng lệnh sau:

```
model.summary()
```

Kết quả:

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_8 (Dense)              (None, 784)               615440
_________________________________________________________________
dense_9 (Dense)              (None, 10)                7850
=================================================================
Total params: 623,290
Trainable params: 623,290
Non-trainable params: 0
_________________________________________________________________
```

5. Huấn luyện model

Sử dụng hàm optimizer: **Adam**

```
optimizer = Adam(learning_rate=0.01)
```

Hàm loss: **BinaryCrossEntropy**

```
loss = BinaryCrossentropy()
```

Metric đánh giá: **accuracy**

Compile model:

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

Huấn luyện mô hình với các thông số: batch_size = 128, epoch = 10

```
model.fit(X_train_reshaped, y_train, batch_size=128, epochs=10)
```

6. Đánh giá mô hình

Lưu ý: đánh giá trên tập dữ liệu kiểm tra (test).

Dự đoán cho tập test:

```
y_pred = model.predict(X_test_reshaped)
```

Kết quả dự đoán của mô hình sẽ là một ma trận có kích thước **(10000, 10).**

Ý nghĩa:

+ 10000: số lượng dữ liệu test.

+ 10: mỗi kết quả dự đoán cho một điểm giá trị sẽ là một vector thể hiện cho xác suất một điểm dữ liệu đó thuộc về mỗi lớp trong tổng cộng 10 lớp.

=> Lớp được chọn là lớp có **xác suất cao nhất**.

Để tìm xác suất cao nhất: dùng hàm **argmax** trong thư viện numpy

```
y_pred = np.argmax(y_pred, axis = -1)
```

Tính độ chính xác của mô hình:

```
from sklearn.metrics import accuracy_score


accuracy_score(y_test, y_pred)*100
```

Kết quả sẽ hơi thấp nhé !!!

LƯU Ý: Các bạn hãy bật GPU của Colab lên để thực hiện huấn luyện nhanh hơn.

7. Lưu mô hình

Keras sẽ lưu lại mô hình dưới dạng file h5.

```
model.save('my_model.h5')
```

Load lại model

```
from keras.models import load_model
model = load_model('my_model.h5')
```

BÀI TẬP THỰC HIỆN:

Bài 1: Hãy sử dụng activation sigmoid và xem kết quả.

Bài 2: Hãy sử dụng **activation relu cho lớp thứ 1 và activation sigmoid cho lớp đầu ra** và xem kết quả.

Bài 3: Normalizing giá trị cho data như sau:

```
X_train_new = X_train/255.0
X_test_new = X_test/255.0
```

Hãy thực hiện lại model ở bài 2 trên dữ liệu X_train và X_test đã chuẩn hoá.

Bài 4: In ra ma trận nhầm lẫn của mô hình ở bài 3 và nhận xét.

Gợi ý: code in ma trận nhầm lẫn

```
df_cm = pd.DataFrame(cm, index = [i for i in
range(0,10)], columns = [i for i in range(0,10)])

plt.figure(figsize = (10,7))
sn.heatmap(df_cm, annot=True, fmt='.5g')
```

Bài 5: *Hãy xây dựng model đơn giản gồm 2 lớp cho bộ dữ liệu **small CIFAR10**.

Gợi ý: load dữ liệu như sau

```
from keras.datasets.cifar10 import load_data
(X_train, y_train), (X_test, y_test) = load_data()
```