Sau bài thực hành này, sinh viên có khả năng:

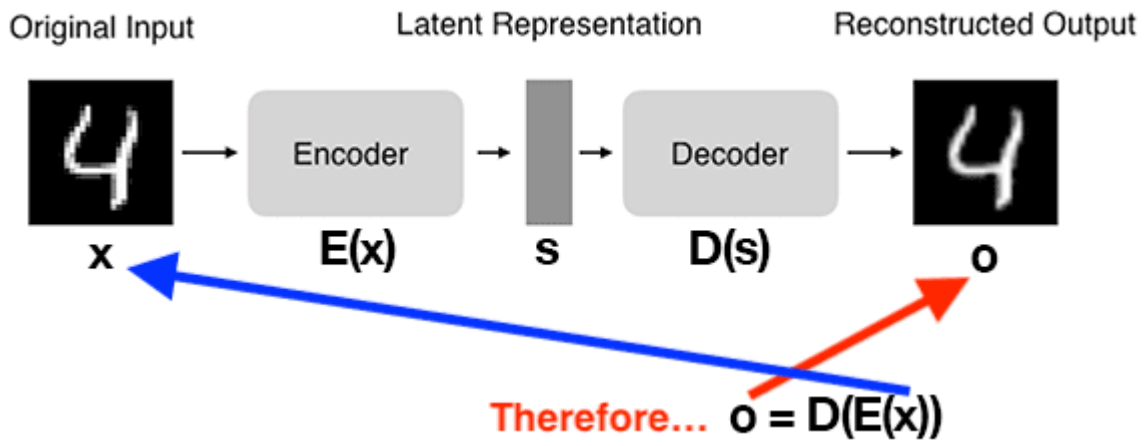
- Xây dựng được kiến trúc Convolutional Neural Network (CNN)
- Cài đặt mô hình Convolutional Neural Network bằng tensorflow
- Cài đặt ứng dụng thực tế sử dụng Convolutional Neural Network

1. GIỚI THIỆU

Autoencoder nhận một ảnh chưa gán nhãn và rút các đặc trưng quan trọng của một đối tượng (ảnh). Đây là phương pháp học không giám sát.

Thường Autoencoder có 2 thành phần:

- Encoder: Nhận dữ liệu đầu vào và nén trong không gian hidden. Gọi x là input data, E là hàm encoder thì biểu diễn đầu ra không gian ẩn là s = E(x)
- Decoder: nhận không gian ẩn s từ encoder và tái dựng lại dữ liệu ban đầu. Gọi D là hàm decoder, o là đầu ra thì o = D(s)
- Về mặt toán học, toàn bộ quá trình autoencoder sẽ được viết lại như sau: o = D(E(x))



2. CÀI ĐẶT MÔ HÌNH AUTOENCODER

Phần này sẽ trình bày cài đặt mô hình autoencoder cho bài toán phân lớp trên bộ dataset Fashion_mnist.

2.1. Nạp thư viện

```
[1]  from keras.datasets import mnist
     import numpy as np
     import keras
     from keras import layers
```

2.2. Nạp dataset

```
(x_train, _), (x_test, _) = mnist.load_data()
x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.
x_train = x_train.reshape((len(x_train), np.prod(x_train.shape[1:])))
x_test = x_test.reshape((len(x_test), np.prod(x_test.shape[1:])))
print(x_train.shape)
print(x_test.shape)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
(60000, 784)
(10000, 784)
```

2.3. Xây dựng mô hình Autoencoder

```
[3] # This is the size of our encoded representations
    encoding_dim = 32  # 32 floats -> compression of factor 24.5, assuming the input is 784 floats

    # This is our input image
    input_img = keras.Input(shape=(784,))
    # "encoded" is the encoded representation of the input
    encoded = layers.Dense(encoding_dim, activation='relu')(input_img)
    # "decoded" is the lossy reconstruction of the input
    decoded = layers.Dense(784, activation='sigmoid')(encoded)

    # This model maps an input to its reconstruction
    autoencoder = keras.Model(input_img, decoded)

    # This model maps an input to its encoded representation
    encoder = keras.Model(input_img, encoded)
    # This is our encoded (32-dimensional) input
    encoded_input = keras.Input(shape=(encoding_dim,))
    # Retrieve the last layer of the autoencoder model
    decoder_layer = autoencoder.layers[-1]
    # Create the decoder model
    decoder = keras.Model(encoded_input, decoder_layer(encoded_input))
```

2.4. Huấn luyện mô hình

```
[7] autoencoder.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    autoencoder.fit(x_train, x_train,
                    epochs=50,
                    batch_size=256, verbose=1,
                    shuffle=True,
                    validation_data=(x_test, x_test))
```

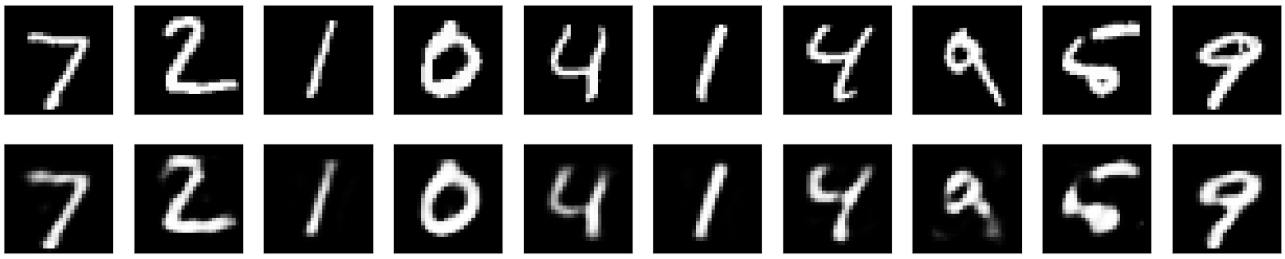
2.5. Dự báo loại ảnh sử dụng Autoencoder

```
[8]  encoded_imgs = encoder.predict(x_test)
     decoded_imgs = decoder.predict(encoded_imgs)
```

2.6. Hiển thị ảnh từ mô hình Autoencoder

```
# Use Matplotlib (don't ask)
import matplotlib.pyplot as plt

n = 10  # How many digits we will display
plt.figure(figsize=(20, 4))
for i in range(n):
    # Display original
    ax = plt.subplot(2, n, i + 1)
    plt.imshow(x_test[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

    # Display reconstruction
    ax = plt.subplot(2, n, i + 1 + n)
    plt.imshow(decoded_imgs[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
plt.show()
```

```
Epoch 16/20
235/235 [==============================] - 3s 13ms/step - loss: 0.0939 - accuracy: 0.0110 - val_loss: 0.0925 - val_accuracy: 0.0124
Epoch 17/20
235/235 [==============================] - 3s 12ms/step - loss: 0.0937 - accuracy: 0.0110 - val_loss: 0.0924 - val_accuracy: 0.0123
Epoch 18/20
235/235 [==============================] - 3s 12ms/step - loss: 0.0936 - accuracy: 0.0113 - val_loss: 0.0925 - val_accuracy: 0.0119
Epoch 19/20
235/235 [==============================] - 4s 15ms/step - loss: 0.0935 - accuracy: 0.0110 - val_loss: 0.0923 - val_accuracy: 0.0110
Epoch 20/20
235/235 [==============================] - 3s 14ms/step - loss: 0.0934 - accuracy: 0.0114 - val_loss: 0.0922 - val_accuracy: 0.0113
```



3. CẢI TIẾN AUTOENCODER

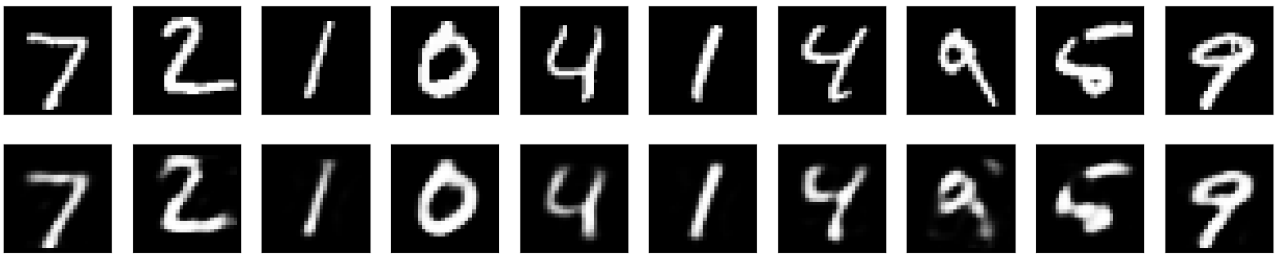
3.1. Thêm L1

Ta chỉ mới ràng buộc kích thước tầng ẩn là 32, tuy nhiên kết quả học của mô hình chưa thật sự rõ như input data. Ta có thể cải tiến bằng cách thêm L1 như sau

```
# This is the size of our encoded representations
encoding_dim = 32  # 32 floats -> compression of factor 24.5, assuming the input is 784 floats

# This is our input image
input_img = keras.Input(shape=(784,))
# "encoded" is the encoded representation of the input
encoded = layers.Dense(encoding_dim, activity_regularizer=regularizers.l1(10e-5), activation='relu')(input_img)
# "decoded" is the lossy reconstruction of the input
decoded = layers.Dense(784, activation='sigmoid')(encoded)

# This model maps an input to its reconstruction
autoencoder = keras.Model(input_img, decoded)

# This model maps an input to its encoded representation
encoder = keras.Model(input_img, encoded)
# This is our encoded (32-dimensional) input
encoded_input = keras.Input(shape=(encoding_dim,))
# Retrieve the last layer of the autoencoder model
decoder_layer = autoencoder.layers[-1]
# Create the decoder model
decoder = keras.Model(encoded_input, decoder_layer(encoded_input))
```

```
autoencoder.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
autoencoder.fit(x_train, x_train,
                epochs=100,
                batch_size=256, verbose=1,
                shuffle=True,
                validation_data=(x_test, x_test))
```

```
Epoch 16/20
235/235 [==============================] - 3s 12ms/step - loss: 0.1043 - accuracy: 0.0117 - val_loss: 0.1029 - val_accuracy: 0.0109
Epoch 17/20
235/235 [==============================] - 3s 12ms/step - loss: 0.1039 - accuracy: 0.0118 - val_loss: 0.1025 - val_accuracy: 0.0103
Epoch 18/20
235/235 [==============================] - 4s 17ms/step - loss: 0.1035 - accuracy: 0.0115 - val_loss: 0.1021 - val_accuracy: 0.0116
Epoch 19/20
235/235 [==============================] - 3s 12ms/step - loss: 0.1032 - accuracy: 0.0118 - val_loss: 0.1019 - val_accuracy: 0.0112
Epoch 20/20
235/235 [==============================] - 3s 11ms/step - loss: 0.1029 - accuracy: 0.0117 - val_loss: 0.1016 - val_accuracy: 0.0116
```

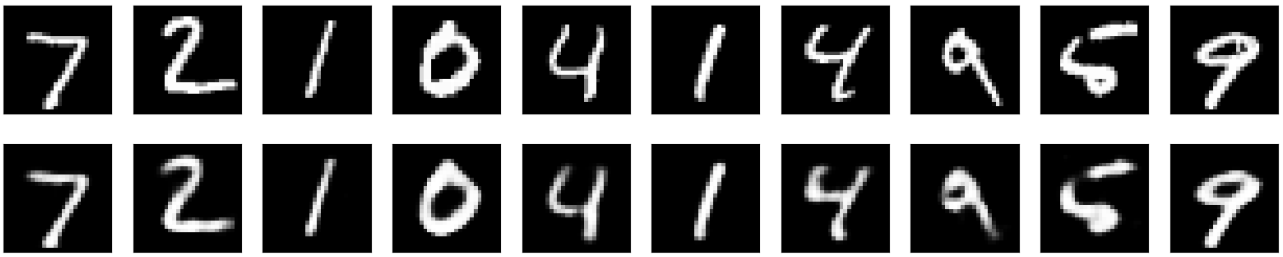


3.2. Thêm layers

Ta cũng có thể thêm layer để cải tiến như sau

```
# This is the size of our encoded representations
encoding_dim = 32  # 32 floats -> compression of factor 24.5, assuming the input is 784 floats

#method 2: increasing multi-layers

# This is our input image
input_img = keras.Input(shape=(784,))
# "encoded" is the encoded representation of the input
encoded = layers.Dense(128, activation='relu')(input_img)
encoded = layers.Dense(64, activation='relu')(encoded)
encoded = layers.Dense(encoding_dim, activation='relu')(encoded)
# "decoded" is the lossy reconstruction of the input

decoded = layers.Dense(784, activation='sigmoid')(encoded)


# This model maps an input to its reconstruction
autoencoder = keras.Model(input_img, decoded)

# This model maps an input to its encoded representation
encoder = keras.Model(input_img, encoded)
encoded_input = keras.Input(shape=(32,))
# Retrieve the last layer of the autoencoder model
decoder_layer = autoencoder.layers[-1]
# Create the decoder model
decoder = keras.Model(encoded_input, decoder_layer(encoded_input))
```

```
Epoch 16/20
235/235 [==============================] - 4s 15ms/step - loss: 0.0942 - accuracy: 0.0136 - val_loss: 0.0927 - val_accuracy: 0.0134
Epoch 17/20
235/235 [==============================] - 5s 21ms/step - loss: 0.0937 - accuracy: 0.0138 - val_loss: 0.0925 - val_accuracy: 0.0160
Epoch 18/20
235/235 [==============================] - 3s 15ms/step - loss: 0.0933 - accuracy: 0.0136 - val_loss: 0.0922 - val_accuracy: 0.0161
Epoch 19/20
235/235 [==============================] - 4s 17ms/step - loss: 0.0929 - accuracy: 0.0137 - val_loss: 0.0915 - val_accuracy: 0.0150
Epoch 20/20
235/235 [==============================] - 4s 19ms/step - loss: 0.0926 - accuracy: 0.0137 - val_loss: 0.0914 - val_accuracy: 0.0149
```



Yêu cầu: sinh viên huấn luyện lại với số epochs: 50, 100, 200 để xem sự khác biệt.

4. BÀI TẬP

Viết chương trình cài đặt Autoencoder để nhận dạng ảnh trên bộ dataset CIFAR10 có sẵn trong tensorflow với các nhãn sau

Label	Description
0	airplane
1	automobile
2	bird
3	cat
4	deer
5	dog
6	frog
7	horse
8	ship
9	truck

2. Hãy viết chương trình cài đặt Autoencoder để nhận dạng ảnh Cat hoặc Dog. Dữ liệu do giảng viên cung cấp

3. Hãy viết chương trình cài đặt Autoencoder để nhận dạng ảnh Fashion-MNIST. Dữ liệu do giảng viên cung cấp.

4. Hãy viết chương trình cài đặt Autoencoder để nhận dạng ảnh khuôn mặt {Nam, Nữ}. Dữ liệu do giảng viên cung cấp.

5. Triển khai câu 1 - 4 trên nền tảng WEB sử dụng Flask