Họ tên : Nguyễn Quang Anh

Mã sinh viên : B21DCDT036

CHƯƠNG 10 : Học Sâu trong Xử Lý Ảnh - Phân Loại Ảnh

THUẬT TOÁN 1 : Phân loại với TensorFlow ( Classification with TF )

Bây giờ, chúng ta sẽ triển khai phân loại hình ảnh sử dụng deep learning với TensorFlow. Đầu tiên, cần tải tập dữ liệu **MNIST** và chia hình ảnh huấn luyện thành hai phần: phần lớn hơn (50k hình ảnh) dùng để huấn luyện, và phần còn lại (10k hình ảnh) dùng để xác thực. Nhãn sẽ được định dạng lại dưới dạng vector nhị phân mã hóa một-hot để biểu diễn các lớp ảnh. Sau đó, biểu đồ TensorFlow được khởi tạo cùng với các tensor biến, hằng số và placeholder. Một thuật toán tối ưu hóa **stochastic gradient descent (SGD)** sẽ được sử dụng với kích thước batch là 256, để giảm thiểu hàm mất mát **softmax cross-entropy logit** cùng với các bộ điều chuẩn L2 trên các lớp trọng số (với giá trị siêu tham số λ₁=λ₂=1). Cuối cùng, phiên TensorFlow sẽ chạy trong 6.000 bước (mini-batches), thực hiện truyền xuôi và ngược (forward/backpropagation) để cập nhật mô hình (trọng số) đã học, sau đó đánh giá mô hình trên tập dữ liệu xác thực. Sau khi hoàn thành batch cuối cùng, độ chính xác đạt được là **96.5%**.

1, Giới thiệu thuật toán :

Thuật toán sử dụng thư viện TensorFlow và Keras để xây dựng, huấn luyện, và đánh giá một mô hình mạng nơ-ron nhân tạo (ANN) trên tập dữ liệu MNIST. MNIST là tập dữ liệu hình ảnh chữ số viết tay, rất phổ biến trong các bài toán nhận dạng hình ảnh.

**Mục đích chính:**

1. Tiền xử lý dữ liệu hình ảnh MNIST.
2. Xây dựng một mạng nơ-ron với một lớp ẩn.
3. Huấn luyện mô hình và đánh giá độ chính xác.
4. Hiển thị trực quan hóa trọng số (weights) học được từ lớp ẩn.

 **Dữ liệu**: Sử dụng tập dữ liệu MNIST chứa các hình ảnh 28x28 pixel, mỗi hình ảnh là chữ số từ 0 đến 9.

 **Mạng nơ-ron**:

* Gồm một lớp ẩn với 1.024 nút và hàm kích hoạt ReLU.
* Lớp đầu ra với 10 nút, dùng hàm softmax để phân phối xác suất cho 10 lớp.

 **Tối ưu hóa**: Sử dụng SGD với batch size = 256 và điều chuẩn L2 để giảm thiểu hiện tượng overfitting.

 **Huấn luyện**: Chạy 6.000 bước mini-batches, tối ưu trọng số thông qua backpropagation.

 **Đánh giá**: Đo độ chính xác trên tập xác thực (10.000 hình ảnh).

2,Các bước thực hiện thuật toán :

B1: **Tải và chia dữ liệu**:

* Tải tập MNIST.
* Chia tập dữ liệu thành 50.000 hình ảnh để huấn luyện và 10.000 hình ảnh để xác thực.

B2: **Tiền xử lý dữ liệu**:

* Làm phẳng các hình ảnh 28x28 thành vector 784 chiều.
* Chuẩn hóa giá trị pixel về khoảng [0, 1].
* Biến đổi nhãn thành vector one-hot để tương thích với lớp đầu ra.

B3: **Xây dựng mô hình**:

* Khởi tạo mô hình với:
  + Lớp đầu vào nhận vector 784 chiều.
  + Lớp ẩn với 1.024 nút và hàm kích hoạt ReLU.
  + Lớp đầu ra với 10 nút và hàm softmax.
* Áp dụng điều chuẩn L2 lên trọng số các lớp.

B4: **Huấn luyện mô hình**:

* Chọn thuật toán SGD với learning rate = 0.008 và batch size = 256.
* Chạy qua nhiều epoch để tối ưu hàm mất mát cross-entropy.

B5: **Đánh giá mô hình**:

* Kiểm tra độ chính xác trên tập xác thực.

B6: **Trực quan hóa trọng số**:

* Vẽ các trọng số đã học ở lớp ẩn để thấy cách mạng nơ-ron "học" các đặc trưng từ dữ liệu.

3, Giải thích đoạn mã nguồn :

import tensorflow as tf  
import numpy as np  
import matplotlib.pyplot as plt  
from tensorflow.keras.datasets import mnist  
  
# Load dữ liệu  
(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data()  
np.random.seed(0)  
train\_indices = np.random.choice(60000, 50000, replace=False)  
valid\_indices = [i for i in range(60000) if i not in train\_indices]  
X\_valid, y\_valid = X\_train[valid\_indices, :, :], y\_train[valid\_indices]  
X\_train, y\_train = X\_train[train\_indices, :, :], y\_train[train\_indices]  
  
image\_size = 28  
num\_labels = 10  
  
# Hàm tiền xử lý dữ liệu  
def reformat(dataset, labels):  
 dataset = dataset.reshape((-1, image\_size \* image\_size)).astype(np.float32) / 255.0  
 labels = tf.keras.utils.to\_categorical(labels, num\_labels)  
 return dataset, labels  
  
X\_train, y\_train = reformat(X\_train, y\_train)  
X\_valid, y\_valid = reformat(X\_valid, y\_valid)  
X\_test, y\_test = reformat(X\_test, y\_test)  
  
print('Training set', X\_train.shape)  
print('Validation set', X\_valid.shape)  
print('Test set', X\_test.shape)  
  
# Định nghĩa mô hình  
batch\_size = 256  
num\_hidden\_units = 1024  
lambda1 = 0.1  
lambda2 = 0.1  
  
model = tf.keras.Sequential([  
 tf.keras.layers.InputLayer(input\_shape=(image\_size \* image\_size,)),  
 tf.keras.layers.Dense(num\_hidden\_units, activation='relu', kernel\_regularizer=tf.keras.regularizers.l2(lambda1)),  
 tf.keras.layers.Dense(num\_labels, activation='softmax', kernel\_regularizer=tf.keras.regularizers.l2(lambda2))  
])  
  
model.compile(  
 optimizer=tf.keras.optimizers.SGD(learning\_rate=0.008),  
 loss='categorical\_crossentropy',  
 metrics=['accuracy']  
)  
  
# Huấn luyện mô hình  
history = model.fit(X\_train, y\_train, epochs=10, batch\_size=batch\_size, validation\_data=(X\_valid, y\_valid))  
  
# Hiển thị trực quan trọng số  
weights = model.layers[0].get\_weights()[0]  
plt.figure(figsize=(18, 18))  
indices = np.random.choice(num\_hidden\_units, 225)  
for j in range(225):  
 plt.subplot(15, 15, j + 1)  
 plt.imshow(weights[:, indices[j]].reshape((image\_size, image\_size)), cmap='gray')  
 plt.xticks([], [])  
 plt.yticks([], [])  
plt.title('Weights Visualization')  
plt.show()

 **Tải dữ liệu MNIST**:

(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data()

* Tải tập dữ liệu huấn luyện và kiểm tra từ thư viện Keras.
* X\_train, y\_train: Hình ảnh và nhãn dùng để huấn luyện.
* X\_test, y\_test: Hình ảnh và nhãn dùng để kiểm tra.

 **Chia dữ liệu**:

train\_indices = np.random.choice(60000, 50000, replace=False)

valid\_indices = [i for i in range(60000) if i not in train\_indices]

* Tạo hai tập con từ dữ liệu huấn luyện:
  + 50.000 hình ảnh để huấn luyện.
  + 10.000 hình ảnh để xác thực.

 **Hàm tiền xử lý dữ liệu**:

def reformat(dataset, labels):

dataset = dataset.reshape((-1, image\_size \* image\_size)).astype(np.float32) / 255.0

labels = tf.keras.utils.to\_categorical(labels, num\_labels)

return dataset, labels

* **Reshape**: Chuyển đổi hình ảnh từ ma trận 28x28 thành vector 784 chiều.
* **Chuẩn hóa**: Đưa các giá trị pixel về [0, 1] để tăng hiệu quả huấn luyện.
* **One-hot**: Mã hóa nhãn thành vector nhị phân, ví dụ 3 → [0, 0, 0, 1, 0, 0, 0, 0, 0, 0].

 **Xây dựng mô hình**:

model = tf.keras.Sequential([

tf.keras.layers.InputLayer(input\_shape=(image\_size \* image\_size,)),

tf.keras.layers.Dense(num\_hidden\_units, activation='relu', kernel\_regularizer=tf.keras.regularizers.l2(lambda1)),

tf.keras.layers.Dense(num\_labels, activation='softmax', kernel\_regularizer=tf.keras.regularizers.l2(lambda2))

])

* **Lớp đầu vào**: Nhận vector 784 chiều.
* **Lớp ẩn**: 1.024 nút, kích hoạt ReLU, áp dụng điều chuẩn L2.
* **Lớp đầu ra**: 10 nút, kích hoạt softmax để tính xác suất.

 **Huấn luyện mô hình**:

history = model.fit(X\_train, y\_train, epochs=10, batch\_size=batch\_size, validation\_data=(X\_valid, y\_valid))

* Huấn luyện qua 10 epoch với kích thước batch 256.
* Tự động đánh giá trên tập xác thực sau mỗi epoch.

 **Trực quan hóa trọng số**:

python

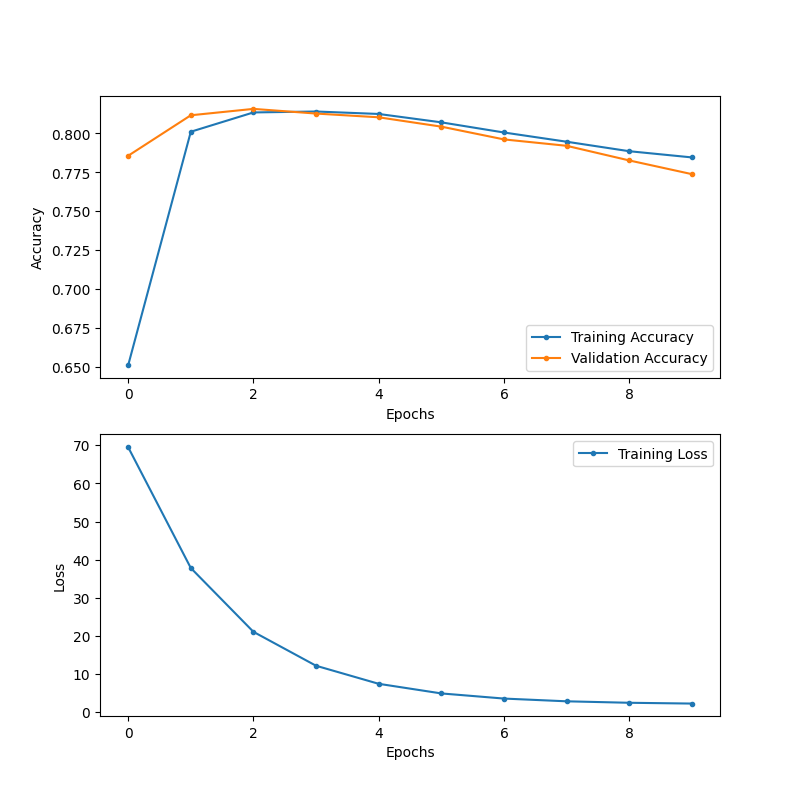
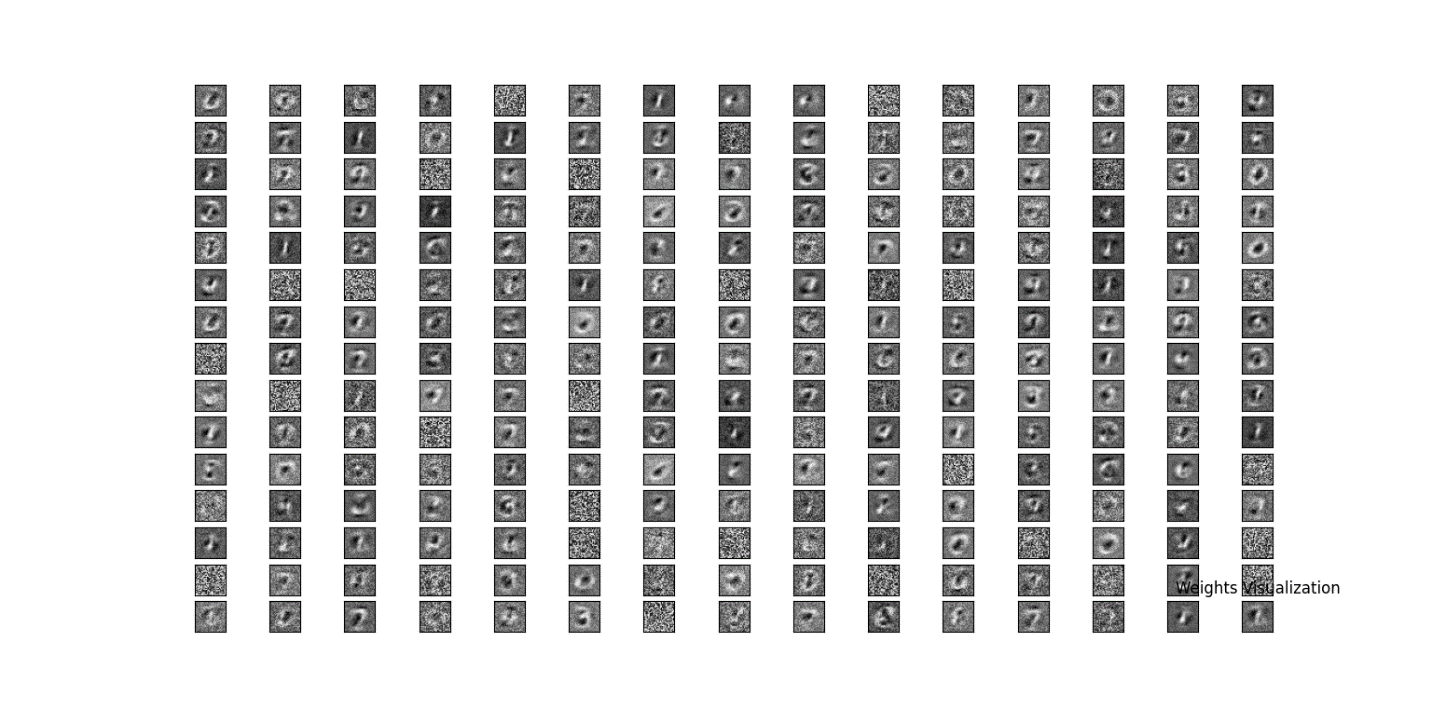
Sao chép mã

weights = model.layers[0].get\_weights()[0]

plt.imshow(weights[:, indices[j]].reshape((image\_size, image\_size)), cmap='gray')

* Lấy trọng số từ lớp ẩn đầu tiên.
* Chuyển trọng số thành hình ảnh 28x28 để xem đặc trưng mà mạng đã học.

4, Kết quả thực hiện thuật toán :

Ví dụ : 

THUẬT TOÁN 2 : Thuật toán Phân loại Sử dụng Mạng Nơ-ron Dày đặc (Fully Connected - FC) trong Keras

1,Giới thiệu thuật toán

Trong bài toán phân loại, mục tiêu của thuật toán là phân loại các dữ liệu đầu vào (ở đây là hình ảnh chữ số viết tay trong bộ dữ liệu MNIST) vào các lớp tương ứng. Chúng ta sử dụng mạng nơ-ron nhân tạo với các lớp dày đặc (fully connected) để thực hiện công việc phân loại này.

Thuật toán phân loại sử dụng Mạng Nơ-ron Dày đặc (Fully Connected - FC) có thể được mô tả theo các bước sau:

1. **Tiền xử lý dữ liệu**: Bao gồm việc chuẩn hóa dữ liệu (chuyển đổi giá trị ảnh từ 0 đến 255 thành dải [0,1]).
2. **Xây dựng mô hình mạng nơ-ron**: Mô hình này có thể bao gồm các lớp Dense (dày đặc), lớp Flatten (chuyển đổi ảnh 2D thành vector 1D), và Dropout (giảm overfitting).
3. **Huấn luyện mô hình**: Sử dụng một thuật toán tối ưu như Adam, và huấn luyện mô hình trên dữ liệu huấn luyện.
4. **Đánh giá mô hình**: Kiểm tra hiệu quả của mô hình trên bộ dữ liệu kiểm tra.
5. **Trực quan hóa trọng số**: Trực quan hóa các trọng số mà mô hình học được trong các lớp ẩn.

Mô hình này có thể đạt được độ chính xác cao, như trong ví dụ này là 98.04% khi phân loại các hình ảnh trong bộ dữ liệu MNIST.

2, Các Bước Để Thực Hiện Thuật Toán

 B1: **Tiền xử lý dữ liệu**:

* **Tải và chuẩn bị dữ liệu**: Dữ liệu MNIST được tải từ thư viện Keras và được chia thành 2 phần: tập huấn luyện và tập kiểm tra.
* **Chuyển đổi hình ảnh**: Chúng ta reshape dữ liệu đầu vào từ dạng (28, 28) thành (28, 28, 1) để phù hợp với các mô hình CNN hoặc FC.
* **Chuẩn hóa dữ liệu**: Chia giá trị pixel trong hình ảnh cho 255 để chuyển đổi chúng từ dải [0, 255] thành [0, 1] giúp quá trình huấn luyện ổn định hơn.
* **Mã hóa nhãn**: Các nhãn mục tiêu của bài toán được chuyển đổi thành mã hóa one-hot để sử dụng cho phân loại đa lớp.

B2: **Xây dựng mô hình mạng nơ-ron dày đặc (FC)**:

* **Flatten**: Chuyển đổi ảnh 28x28 thành một vector 1D dài 784 (28 \* 28).
* **Lớp Dense**: Sử dụng hai lớp Dense, mỗi lớp có 200 nút và kích hoạt ReLU để học các đặc trưng phức tạp của dữ liệu.
* **Lớp Dropout**: Được chèn vào giữa các lớp Dense để giảm overfitting bằng cách ngẫu nhiên bỏ đi một phần các kết nối giữa các lớp trong mỗi lần huấn luyện (15% dropout trong trường hợp này).
* **Lớp Output**: Lớp cuối cùng có 10 nút tương ứng với 10 lớp (từ 0 đến 9) và sử dụng hàm kích hoạt softmax để xác suất hóa các giá trị đầu ra.

B3: **Huấn luyện mô hình**:

* **Bộ tối ưu hóa Adam**: Được sử dụng vì khả năng tối ưu hiệu quả trong các bài toán học sâu, giúp mô hình học nhanh và ổn định hơn.
* **Hàm mất mát categorical\_crossentropy**: Được chọn vì đây là bài toán phân loại đa lớp, và hàm mất mát này đặc biệt phù hợp với các vấn đề phân loại.

B4: **Đánh giá mô hình**:

* Sau khi huấn luyện, mô hình được kiểm tra trên bộ dữ liệu kiểm tra MNIST để đánh giá độ chính xác và lỗi.

B5: **Trực quan hóa trọng số**:

* Trực quan hóa các trọng số của lớp Dense đầu tiên bằng cách chuyển đổi mỗi trọng số thành một hình ảnh 28x28 và hiển thị chúng.
* Điều này giúp chúng ta thấy được các đặc trưng mà mô hình học được từ dữ liệu, ví dụ như các đặc trưng hình học của chữ số.

3, Giải Thích Các Câu Lệnh Mã Nguồn :

import keras  
from keras.models import Sequential  
from keras.layers import Dense, Flatten, Dropout  
from keras.utils import to\_categorical  
from keras.datasets import mnist  
import matplotlib.pylab as pylab  
import numpy as np  
  
# Load data  
(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data()  
print(X\_train.shape, X\_test.shape) # (60000, 28, 28) (10000, 28, 28)  
  
# Reshape to [samples][pixels][width][height]  
X\_train = X\_train.reshape(X\_train.shape[0], 28, 28, 1).astype('float32')  
X\_test = X\_test.reshape(X\_test.shape[0], 28, 28, 1).astype('float32')  
  
# Normalize training and test data  
X\_train = X\_train / 255.0  
X\_test = X\_test / 255.0  
  
# Convert labels to one-hot encoding  
y\_train = to\_categorical(y\_train)  
y\_test = to\_categorical(y\_test)  
num\_classes = y\_test.shape[1] # Number of categories  
  
  
def FC\_model():  
 *"""Create and return a fully connected model."""* # Create model  
 model = Sequential()  
 model.add(Flatten(input\_shape=(28, 28, 1))) # Flatten input image to 1D vector  
 model.add(Dense(200, activation='relu')) # First hidden layer with 200 units  
 model.add(Dropout(0.15)) # Add dropout with 15% rate  
 model.add(Dense(200, activation='relu')) # Second hidden layer with 200 units  
 model.add(Dense(num\_classes, activation='softmax')) # Output layer for classification  
 # Compile model  
 model.compile(optimizer='adam',  
 loss='categorical\_crossentropy',  
 metrics=['accuracy'])  
 return model  
  
  
# Build the model  
model = FC\_model()  
model.summary()  
  
# Fit the model  
model.fit(X\_train, y\_train,  
 validation\_data=(X\_test, y\_test),  
 epochs=10,  
 batch\_size=200,  
 verbose=2)  
  
# Evaluate the model  
scores = model.evaluate(X\_test, y\_test, verbose=0)  
print("Accuracy: {:.4f} \nError: {:.4f}".format(scores[1], 100 - scores[1] \* 100))  
  
# Visualize the weights of the first dense layer  
W = model.get\_layer('dense').get\_weights() # Get the weights of the first dense layer  
print(W[0].shape) # Shape of weight matrix  
print(W[1].shape) # Shape of bias vector  
  
# Visualizing weights of the first 200 hidden units  
fig = pylab.figure(figsize=(20, 20))  
fig.subplots\_adjust(left=0, right=1, bottom=0, top=0.95, hspace=0.05, wspace=0.05)  
  
pylab.gray() # Set color to gray  
  
# Loop through the first 200 neurons and visualize their weights  
for i in range(200):  
 # Reshape the weights to be 28x28 (reshape each column of W[0] into a 28x28 image)  
 pylab.subplot(15, 14, i + 1)  
 pylab.imshow(np.reshape(W[0][:, i], (28, 28))) # Reshape each column to 28x28 for visualization  
 pylab.axis('off') # Hide axis  
  
# Title for the plot  
pylab.suptitle('Dense Layer Weights (200 hidden units)', size=20)  
pylab.show()

import keras

from keras.models import Sequential

from keras.layers import Dense, Flatten, Dropout

from keras.utils import to\_categorical

from keras.datasets import mnist

import matplotlib.pylab as pylab

import numpy as np

* Các thư viện cần thiết cho việc xây dựng và huấn luyện mô hình mạng nơ-ron.
  + keras là thư viện chính để xây dựng mô hình.
  + matplotlib.pylab được dùng để trực quan hóa dữ liệu.
  + numpy dùng cho các phép toán số học.

(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data()

* Tải bộ dữ liệu MNIST, bao gồm hình ảnh chữ số viết tay và nhãn tương ứng.

X\_train = X\_train.reshape(X\_train.shape[0], 28, 28, 1).astype('float32')

X\_test = X\_test.reshape(X\_test.shape[0], 28, 28, 1).astype('float32')

* Chuyển đổi dữ liệu hình ảnh từ dạng (28, 28) thành dạng (28, 28, 1) để mô hình có thể nhận diện được.

X\_train = X\_train / 255.0

X\_test = X\_test / 255.0

* Chuẩn hóa giá trị pixel của ảnh từ 0-255 thành 0-1.

y\_train = to\_categorical(y\_train)

y\_test = to\_categorical(y\_test)

* Chuyển nhãn của bài toán thành dạng mã hóa one-hot.

def FC\_model():

model = Sequential()

model.add(Flatten(input\_shape=(28, 28, 1)))

model.add(Dense(200, activation='relu'))

model.add(Dropout(0.15))

model.add(Dense(200, activation='relu'))

model.add(Dense(num\_classes, activation='softmax'))

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

return model

* Định nghĩa một mô hình mạng nơ-ron với các lớp Dense, Dropout và một lớp Flatten.
* Mô hình được biên dịch với bộ tối ưu hóa adam và hàm mất mát categorical\_crossentropy.

model.fit(X\_train, y\_train, validation\_data=(X\_test, y\_test), epochs=10, batch\_size=200, verbose=2)

* Huấn luyện mô hình trên dữ liệu huấn luyện và kiểm tra độ chính xác trên dữ liệu kiểm tra sau mỗi epoch.

W = model.get\_layer('dense').get\_weights()

* Lấy trọng số từ lớp Dense đầu tiên (lớp ẩn).

for i in range(200):

pylab.subplot(15, 14, i + 1)

pylab.imshow(np.reshape(W[0][:, i], (28, 28)))

pylab.axis('off')

* Trực quan hóa các trọng số của lớp đầu tiên dưới dạng hình ảnh 28x28.

4, Kết quả



Nhận xét :

Kết quả đầu ra là các hình ảnh của trọng số từ lớp ẩn đầu tiên (lớp Dense). Mỗi hình ảnh này đại diện cho trọng số của một đơn vị nơ-ron trong lớp đó. Mỗi trọng số là một ma trận 28x28, biểu diễn một phần của hình ảnh chữ số mà mô hình đã học được.

* **Các hình ảnh trực quan** giúp bạn hiểu rõ hơn về các đặc trưng mà mô hình đang học, chẳng hạn như các hình dạng đặc trưng của các chữ số.
* **Trọng số** thể hiện cách mô hình học và phân biệt các chữ số dựa trên các đặc điểm như hình dạng, góc cạnh, hoặc cấu trúc đặc biệt của các chữ số viết tay.