Mạng Neural Convolutions (CNN) là một dạng cơ bản của mạng neural, được sử dụng rộng rãi trong lĩnh vực thị giác máy tính và xử lý hình ảnh. CNN đã đạt được những thành tựu ấn tượng trong nhiều ứng dụng, bao gồm nhận dạng chữ viết tay, nhận diện vật thể trong hình ảnh, phân loại ảnh y tế, và nhiều lĩnh vực khác.

**Cấu trúc của CNN:**

* **Lớp Convolutional (Convolutional Layer)**:

Là lớp cơ bản của CNN, áp dụng các bộ lọc (kernels) để trích xuất các đặc trưng từ ảnh đầu vào. Bằng cách trượt các bộ lọc qua ảnh, chúng ta có thể nhận ra các đặc trưng như cạnh, góc, và các hình dạng phức tạp hơn.

* **Lớp Pooling (Pooling Layer)**:

Là lớp giảm bớt kích thước của đầu ra từ lớp convolutional bằng cách chọn giá trị đại diện từ các vùng nhỏ của ảnh. Pooling giúp giảm thiểu overfitting và giảm độ phức tạp của mô hình.

* **Lớp Fully Connected (Fully Connected Layer)**:

Lớp này kết hợp các đặc trưng đã được trích xuất từ lớp convolutional và pooling, và thực hiện phân loại dữ liệu đầu vào vào các lớp phân loại khác nhau. Thông thường, lớp fully connected được kết nối đến các lớp softmax để tính toán xác suất của từng lớp.

* **Lớp Dropout**:

Đây là một kỹ thuật chính trong CNN để ngăn chặn hiện tượng overfitting. Lớp dropout tắt ngẫu nhiên một số lượng đơn vị trong quá trình huấn luyện, giúp mô hình trở nên tổng quát hơn.

**Ưu điểm của CNN:**

* **Trích xuất đặc trưng tự động**: CNN có khả năng tự động học và trích xuất các đặc trưng phức tạp từ dữ liệu hình ảnh mà không cần sự can thiệp của con người.
* **Độ chính xác cao**: CNN thường có khả năng phân loại và nhận dạng ảnh với độ chính xác cao, vượt qua được nhiều phương pháp truyền thống.
* **Tính linh hoạt**: Có thể áp dụng CNN cho nhiều ứng dụng khác nhau, từ nhận dạng chữ viết tay đến nhận diện vật thể trong thời gian thực.

CNN đã trở thành công cụ quan trọng trong lĩnh vực thị giác máy tính và xử lý hình ảnh, và tiếp tục được nghiên cứu và phát triển để đáp ứng nhu cầu ngày càng tăng trong xã hội số hiện nay.

**Trong source code 2 file.py: trong đó CNN.py là chứa code để huấn luyện mô hình CNN và FORM.py dùng để chứa code tạo ra form vẻ chữ số viết tay để nhận dạng. File mnist\_cnn\_model\_improved.keras dùng để chứa dữ liệu huấn luyện trong quá trình chạy mô hình có tác dụng để chạy file FORM.py bạn có thể chạy file CNN.py để huấn luyện lại mô hình, thay đổi các thông số như số lần huấn luyện, tỉ lệ học để kiểm tra tốc độ học và độ chính xác của mô hình.**

**Giải thích code:**

**Import các thư viện cần thiết**:

import tensorflow as tf

from tensorflow.keras import datasets, layers, models, callbacks

import matplotlib.pyplot as plt

import numpy as np

tensorflow và tensorflow.keras: Thư viện TensorFlow và một API cấp cao được tích hợp trong TensorFlow để xây dựng và huấn luyện mạng neural.

datasets, layers, models: Các module trong keras cung cấp chức năng để tải dữ liệu, xây dựng mô hình và các lớp của mạng neural.

callbacks: Module trong keras chứa các callback hữu ích như EarlyStopping và ReduceLROnPlateau.

matplotlib.pyplot: Thư viện để vẽ đồ thị.

numpy: Thư viện cho tính toán khoa học và tính toán số học trong Python.

**Tải và chuẩn bị dữ liệu MNIST**:

(train\_images, train\_labels), (test\_images, test\_labels) = datasets.mnist.load\_data()

Tải dữ liệu MNIST và chia thành dữ liệu huấn luyện (train\_images, train\_labels) và dữ liệu kiểm tra (test\_images, test\_labels).

**Chuẩn hóa dữ liệu**:

train\_images = train\_images.reshape((60000, 28, 28, 1)).astype('float32') / 255

test\_images = test\_images.reshape((10000, 28, 28, 1)).astype('float32') / 255

Chuẩn hóa dữ liệu bằng cách chia giá trị pixel cho 255 để đưa chúng về khoảng [0, 1].

**Xây dựng mô hình CNN**:

model = models.Sequential([

layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(28, 28, 1)),

layers.BatchNormalization(),

layers.MaxPooling2D((2, 2)),

layers.Dropout(0.25),

# Các lớp khác...

])

Xây dựng một mô hình Sequential với các lớp Convolutional, BatchNormalization, MaxPooling và Dropout.

**Hiển thị kiến trúc mô hình**:

model.summary()

Hiển thị tổng quan về kiến trúc của mô hình, bao gồm số lượng tham số và kích thước đầu ra của mỗi lớp.

**Compile mô hình**:

model.compile(optimizer='adam',

loss='sparse\_categorical\_crossentropy',

metrics=['accuracy'])

Compile mô hình với hàm loss là 'sparse\_categorical\_crossentropy', optimizer là 'adam' và metrics là 'accuracy'.

**Thiết lập callbacks**:

early\_stopping = callbacks.EarlyStopping(monitor='val\_loss', patience=3, restore\_best\_weights=True)

reduce\_lr = callbacks.ReduceLROnPlateau(monitor='val\_loss', factor=0.2, patience=2, min\_lr=0.001)

Thiết lập callbacks EarlyStopping và ReduceLROnPlateau để kiểm soát quá trình huấn luyện.

**Huấn luyện mô hình**:

history = model.fit(train\_images, train\_labels, epochs=20,

validation\_data=(test\_images, test\_labels),

callbacks=[early\_stopping, reduce\_lr])

Huấn luyện mô hình trên dữ liệu huấn luyện và kiểm tra, sử dụng callbacks để kiểm soát quá trình huấn luyện.

**Đánh giá mô hình**:

test\_loss, test\_acc = model.evaluate(test\_images, test\_labels)

print(f'Test accuracy: {test\_acc}')

Đánh giá hiệu suất của mô hình trên dữ liệu kiểm tra và in ra tỉ lệ đúng (accuracy) trên dữ liệu kiểm tra.

**Lưu mô hình đã huấn luyện**:

model.save('mnist\_cnn\_model\_improved.h5')

- Lưu mô hình đã huấn luyện vào một tệp tin.

**Vẽ đồ thị huấn luyện và validation accuracy**:

plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)

plt.plot(history.history['accuracy'], label='Training Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.legend(loc='lower right')

plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)

plt.plot(history.history['loss'], label='Training Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.xlabel('Epoch')

plt.ylabel('Loss')

plt.legend(loc='upper right')

plt.title('Training and Validation Loss')

plt.show()

Vẽ đồ thị biểu diễn sự biến thiên của độ chính xác và hàm mất mát trên cả tập huấn luyện và tập validation qua các epoch.