**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**
**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**
**KHOA CÔNG NGHỆ THÔNG TIN**
---------- 🕮🕮🕮 ----------


# BÀI TẬP VỀ NHÀ 2
## MÔN HỌC: THỊ GIÁC MÁY TÍNH

***Họ Và Tên Sinh Viên:*** *Trần Thuận Phát*
***MSSV:*** *21127666*

**GIÁO VIÊN MÔN HỌC:**
Thầy Phạm Minh Hoàng
Thầy Nguyễn Trọng Việt
Thầy Võ Hoài Việt

**TPHCM – 03/2024**

# MỤC LỤC

# NỘI DUNG

# I. Thông tin cá nhân:

| **Họ và tên** | **Trần Thuận Phát** |
|---|---|
| Mã số sinh viên | 21127666 |

# II. Bảng tự đánh giá:

| STT | Tiêu chí | Đánh giá |
|---|---|---|
| **1** | Có xây dựng và mô tả mô hình LeNet5 theo đúng yêu cầu và kèm hình ảnh minh họa. | **100%** |
| **2** | Có so sánh, đánh giá và trực quan hóa kết quả của mô hình với từng bộ dữ liệu. | **100%** |
| **3** | Có chạy và huấn luyện trên cả 2 bộ dữ liệu cùng một lúc và đánh giá kết quả. | **100%** |
| **4** | Có giải thích kết quả và nêu sự khác biệt. | **100%** |
| **5** | Cấu trúc thư mục bài nộp theo đúng yêu cầu. | **100%** |
| **6** | Báo cáo rõ ràng, đầy đủ, trực quan. | **100%** |
| **7** | Kết quả của thuật toán tốt. | **100%** |
| **8** | Thời gian chạy nhanh. (trong khoảng 400 giây) | **100%** |
| **9** | Dễ bảo trì và mã nguồn có chú thích. | **100%** |
| **10** | Có hướng dẫn cách chạy. | **100%** |
| **11** | Có kèm danh sách các chức năng (phân đoạn code) và giải thích. | **100%** |
| **12** | Nêu đủ tài liệu tham khảo. | **100%** |

# III. Các bước xây dựng một mô hình LeNet5:

**Bước 1:** Import các thư viện hỗ trợ cho quá trình xây dựng mô hình, lấy bộ dữ liệu, gán nhãn và trực quan hóa mô hình và kết quả như là numpy, pandas, matplotlib, random, keras.

**Bước 2:** Load bộ dữ liệu theo nhu cầu và lưu vào các mảng X_train, y_train, X_test và y_test với sự hỗ trợ của thư viện keras và lời gọi hàm load_data(). Như 2 câu lệnh đã dung trong bài tập dưới đây:

**(X_train, y_train), (X_test, y_test) = fashion_mnist.load_data()**

**hoặc**

**(X_train, y_train), (X_test, y_test) = mnist.load_data()**

**Bước 3:** Load tập test từ file csv được chuẩn bị trước và nằm trong thư mục data.

**Bước 4:** Hiển thị một số hình ảnh từ tập train đã lưu (biến X_train) để đảm bảo rằng dữ liệu đã được lưu đúng và chuẩn bị cho quá trình huấn luyện.

**Bước 5:** Tiếp theo ta tiền xử lý dữ liệu đầu vào. Đầu tiên dữ liệu ảnh sẽ được trải phẳng từ ma trận 2D (kích thước 28x28 pixels) thành một vectơ 1D có kích thước 784 (28x28). Việc này là cần thiết để có thể đưa dữ liệu vào mạng neural network được thiết kế để nhận dữ liệu đầu vào 1D. Tiếp theo ta chuẩn hóa dữ liệu bằng cách chia X_train cho 255 và X_test chia cho 255. Việc chuẩn hóa dữ liệu bằng cách chia mỗi giá trị pixel cho 255 làm cho các giá trị pixel về khoảng từ 0 đến 1, giúp tăng tốc độ hội tụ của mô hình trong quá trình huấn luyện và giảm khả năng bị ảnh hưởng bởi tỷ lệ giá trị lớn nhỏ khác nhau giữa các thuộc tính.

**Bước 6:** Tiếp theo, ta thực hiện việc chuyển đổi nhãn y_train và y_test thành one-hot-encoding để phù hợp với mô hình mạng neural network. Vì các mạng neural network thường mong đợi đầu ra ở dạng phân phối xác suất (softmax function), với mỗi lớp đầu ra có một nút tương ứng. Và mỗi nhãn (class) được biểu diễn bằng một vectơ có kích thước bằng số lượng lớp, trong đó chỉ có một phần tử bằng 1 và các phần tử còn lại bằng 0. Bên cạnh đó nó giúp loại bỏ ảnh hưởng của thứ tự do các nhãn

không còn được sắp xếp theo thứ tự nhất định mà được biểu diễn dưới dạng một vectơ vị trí, giúp tránh bất kỳ ảnh hưởng nào của thứ tự trên việc huấn luyện mô hình. Cuối cùng nó giúp dễ dàng so sánh và đánh giá do mã hóa one-hot tạo ra một biểu diễn đồng nhất cho các nhãn, làm cho việc so sánh và đánh giá hiệu quả hơn.

**Bước 7:** Xây dựng và huấn luyện một mô hình Lenet5:

- **Bước 7.1:** Đầu tiên ta sử dụng phương thức Sequential() trong thư viện keras. Đây là một phương thức được sử dụng để tạo ra một mô hình neural network tuần tự. Cụ thể hơn nó là một loại mô hình neural network được sử dụng phổ biến nhất và nó cho phép chúng ta tạo ra một chuỗi tuần tự các lớp neural network và số node theo mong muốn bằng phương thức add() trong đó dữ liệu đi từ lớp này sang lớp khác theo một chiều nhất định, từ đầu vào đến đầu ra và các lớp này sẽ được xếp chồng lên nhau theo thứ tự mà chúng được thêm vào.

```
model = Sequential()
```

- **Bước 7.2:** Add vào mô hình với lớp Convolutional đầu tiên với 32 filters và kích thước kernel là (5, 5). Bên cạnh đó, ta sử dụng hàm kích hoạt ReLU và đầu vào của layer này là một ảnh kích thước 28x28x1.

```
model.add(Conv2D(filters=32, kernel_size=(5,5), padding='same',
activation='relu', input_shape=(28, 28, 1)))
```

- **Bước 7.3:** Tiếp theo ta add vào mô hình lớp MaxPooling đầu tiên với việc sử dụng MaxPool2D và stride có giá trị

là 2 (có nghĩa là sẽ lấy giá trị tối đa trong mỗi cửa sổ 2x2).

```
model.add(MaxPool2D(strides=2))
```

- **Bước 7.4:** Add vào mô hình với lớp Convolutional thứ hai với 48 filters và kích thước kernel là (5, 5). Bên cạnh đó, ta sử dụng hàm kích hoạt ReLU và ta đặt padding là valid nghĩa là ta sẽ không sử dụng zero-padding vì vậy kích thước của đầu ra sẽ giảm so với đầu vào.

```
model.add(Conv2D(filters=48, kernel_size=(5,5), padding='valid', activation='relu'))
```

- **Bước 7.5:** Tiếp theo ta add vào mô hình lớp MaxPooling thứ hai với việc sử dụng MaxPool2D và stride có giá trị là 2 (có nghĩa là sẽ lấy giá trị tối đa trong mỗi cửa sổ 2x2).

```
model.add(MaxPool2D(strides=2))
```

- **Bước 7.6:** Sau đó, ta sử dụng Flatten layer để chuyển từ đầu ra 2D của lớp MaxPooling cuối cùng thành một vector 1D.

```
model.add(Flatten())
```

- **Bước 7.7:** Ta sử dụng Dense để thêm Fully Connected layer đầu tiên với 256 nodes và hàm kích hoạt ReLU.

```
model.add(Dense(256, activation='relu'))
```

- **Bước 7.8:** Tiếp theo, ta sử dụng Dense để thêm Fully Connected layer thứ hai với 84 nodes và hàm kích hoạt ReLU.

```
model.add(Dense(256, activation='relu'))
```

- **Bước 7.8:** Cuối cùng, ta sử dụng Dense để thêm Fully Connected layer đầu ra với 10 nodes và hàm kích hoạt

softmax.

```
model.add(Dense(10, activation='softmax'))
```

**Bước 8:** Hiển thị biểu đồ trực quan với trục x là số epoch và trục y là độ chính xác và độ mất mát với sự hỗ trợ của thư viện matplotlib.

**Bước 9:** Sau khi đã hoàn thành huấn luyện mô hình ta có thể đưa ra dự đoán bằng phương thức predict() với đầu vào tương ứng với đầu vào của việc huấn luyện.

# IV. Mô hình Lenet5 đã xây dựng:

```
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(5,5), padding='same', activation='relu', input_shape=(28, 28, 1)))
model.add(MaxPool2D(strides=2))
model.add(Conv2D(filters=48, kernel_size=(5,5), padding='valid', activation='relu'))
model.add(MaxPool2D(strides=2))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(84, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

*Mô hình Lenet5*

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 28, 28, 32)        832

 max_pooling2d_2 (MaxPoolin  (None, 14, 14, 32)        0
 g2D)

 conv2d_3 (Conv2D)           (None, 10, 10, 48)        38448

 max_pooling2d_3 (MaxPoolin  (None, 5, 5, 48)          0
 g2D)

 flatten_1 (Flatten)         (None, 1200)              0

 dense_3 (Dense)             (None, 256)               307456

 dense_4 (Dense)             (None, 84)                21588

 dense_5 (Dense)             (None, 10)                850

=================================================================
Total params: 369174 (1.41 MB)
Trainable params: 369174 (1.41 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

*Tóm tắt cấu trúc của mô hình Lenet5*

# V. Mô tả các phân đoạn code:

## 1. Import the required libraries

Ở phân đoạn này sẽ thực hiện import các thư viện cần thiết hỗ trợ cho quá trình xây dựng mô hình, lấy bộ dữ liệu, gán nhãn và trực quan hóa mô hình và kết quả như là numpy, pandas, matplotlib, random, keras.

## 2. Load the Dataset:

Ở phần đoạn này sẽ thực hiện chức năng load bộ dữ liệu theo nhu cầu và lưu vào các mảng X_train, y_train, X_test và y_test với sự hỗ trợ của thư viện keras và lời gọi hàm load_data() và hiển thị một số hình ảnh từ tập train đã lưu (biến X_train) để đảm bảo rằng dữ liệu đã được lưu đúng và chuẩn bị cho quá trình huấn luyện.

## 3. Data Preprocessing:

Ở phân đoạn này sẽ tiền xử lý dữ liệu.

## 4. Convert to one-hot encoding:

Ở phân đoạn này ta thực hiện việc chuyển đổi nhãn y_train và y_test thành one-hot-encoding để phù hợp với mô hình mạng neural network.

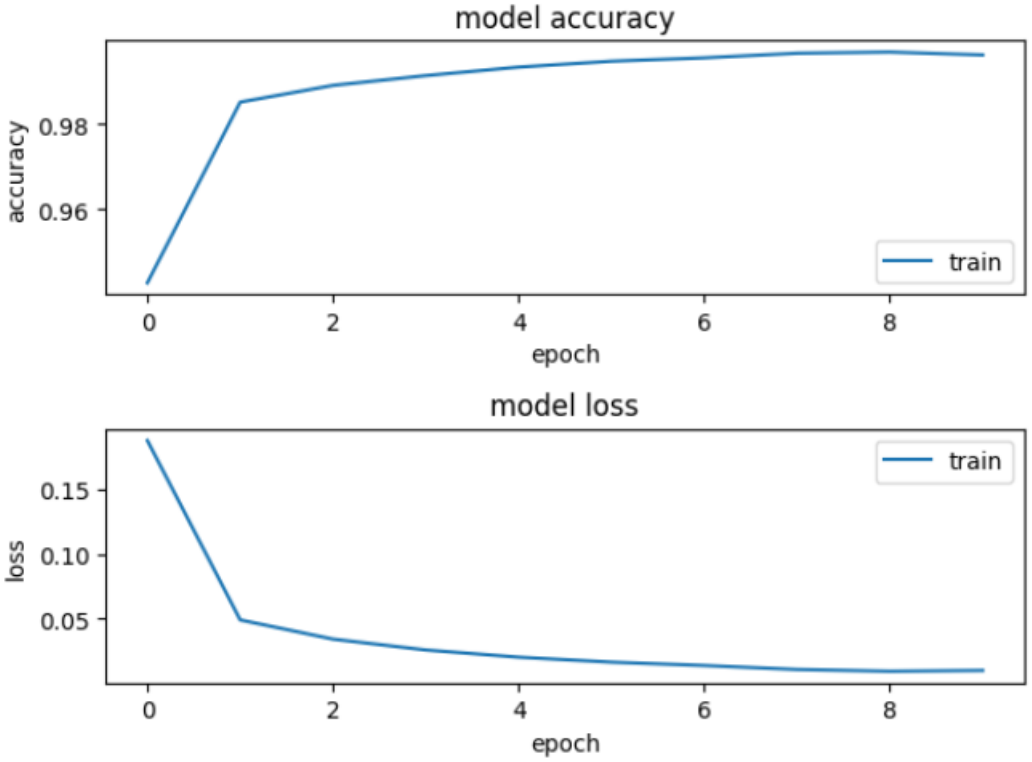
## 5. Lenet5 Model

Ở phân đoạn này ta xây dựng mô hình Lenet5.

## 6. Evaluate the model:

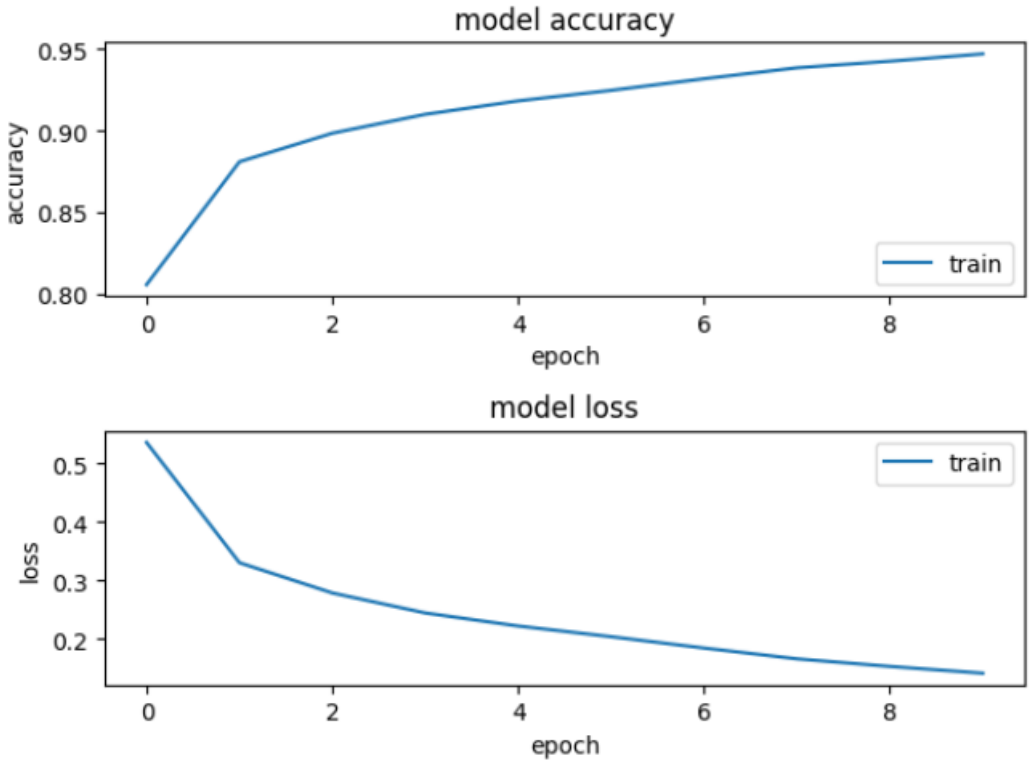
Ở phân đoạn này giúp đánh giá độ chính xác của mô hình và trực quan độ mất mát và độ chính xác của tập train và test qua từng epoch. Cuối cùng là hiển thị một số hình ảnh dự đoán đúng và sai từ mô hình đã đào tạo.

# VI. Phân tích đánh giá kết quả với tập MNIST được huấn luyện riêng biệt:




- **Độ chính xác:** 0.99
- **Đánh giá:** Đầu tiên xét về độ chính xác từ tập test, ta thấy ở đây rất cao và hoàn toàn như gần đúng với mọi trường hợp kiểm thử và mô hình không bị overfitting cho thấy mô hình được xây dựng đưa ra dự đoán tốt và phù hợp với tập dữ liệu. Thứ hai, thời gian huấn luyện khá nhanh trong tầm 400 giây và tổng bộ nhớ mô hình chỉ chiếm 1.41MB với 369174 tham số. Cuối cùng là đường đi của 2 đường acc và loss trong mỗi epoch ở đồ thị trên rất ổn định và tăng hoặc giảm rất đều đặn cho thấy mô hình phù hợp dữ liệu đang huấn luyện.

# VII. Phân tích đánh giá kết quả với tập MNIST-FASHION được huấn luyện riêng biệt:




- **Độ chính xác:** 0.92
- **Đánh giá:** Tương tự ta xét về độ chính xác từ tập test, ta thấy ở đây rất cao và mô hình không bị overfitting và cho thấy mô hình được xây dựng đưa ra dự đoán tốt và phù hợp với tập dữ liệu. Thứ hai, thời gian huấn luyện khá nhanh trong tầm 400 giây và tổng bộ nhớ mô hình chỉ chiếm 1.41MB với 369174 tham số. Cuối cùng là đường đi của 2 đường acc và loss trong mỗi epoch ở đồ thị trên rất ổn định và tăng hoặc giảm rất đều đặn cho thấy mô hình phù hợp dữ liệu đang huấn luyện.

# VIII. Phân tích đánh giá kết quả cả 2 tập được huấn luyện cùng lúc:

**1. Đối với tập dữ liệu MNIST:**

- **Độ chính xác:** 0.26
- **Đánh giá:** Sau khi huấn luyện mô hình với tập dữ liệu MNIST và tiếp tục huấn luyện cho mô hình tập dữ liệu MNIST-FASHION thì độ chính xác của mô hình đối với tập này là rất thấp và gần như quên đi cách phân lớp đối với tập dữ liệu này.

**2. Đối với tập dữ liệu MNIST-FASHION:**

- **Độ chính xác:** 0.91
- **Đánh giá:** Sau khi huấn luyện mô hình với tập dữ liệu MNIST và tiếp tục huấn luyện cho mô hình tập dữ liệu MNIST-FASHION thì độ chính xác của mô hình đối với tập này là vẫn cao và gần như độ chính xác tương đồng với độ chính xác của trường hợp tập dữ liệu MNIST-FASHION được huấn luyện riêng biệt.

➔ **Giải thích:** Do sự khác biệt trong đặc trưng của hai tập dữ liệu. Với dữ liệu MNIST và MNIST-Fashion có những đặc trưng khác nhau một tập dữ liệu là chứa các hình ảnh chữ số và một tập dữ liệu là chứa các ảnh trang phục nhưng lại cùng số lớp nhãn và giá trị nhãn là giống nhau. Chính vì thế, gây nhiễu cho mô hình và do CNN là mô hình học tốt các đặc trưng cơ bản như cạnh và góc đến các đặc trưng phức tạp như các đặc điểm nổi bật của đối tượng trong hình ảnh nên có thể làm cho mô hình phải học lại các đặc trưng mới nên mô hình sẽ dự đoán tốt hơn đối với tập dữ liệu được huấn luyện sau cùng là tập MNIST-FASHION còn tập MNIST gần như mô hình đã quên sạch do bị thay đổi giá trị trọng số sau khi học tập MNIST-FASHION.

# IX. Giải thích kết quả và sự khác biệt của chúng:

- Với cùng một mô hình và huấn luyện trên 2 tập dữ liệu này thì độ chính xác của tập MNIST luôn cao hơn tập MNIST-FASHION với lý do là nhận diện chữ số sẽ ít chi tiết hơn trong việc phát hiện các chi tiết trong trang phục. Bên cạnh đó, các đường nét trong chữ số sẽ dễ dàng dự đoán hơn trang phục.
- Thời gian chạy và bộ nhớ lưu trữ của 2 tập dữ liệu trên cùng mô hình gần tương đồng nhau do có cùng kích thước đầu vào.
- Với việc huấn luyện hai mô hình cùng lúc sẽ rất khó do sự khác biệt trong đặc trưng của hai tập dữ liệu. Với dữ liệu MNIST và MNIST-Fashion có những đặc trưng khác nhau một tập dữ liệu là chứa các hình ảnh chữ số và một tập dữ liệu là chứa các ảnh trang phục nhưng lại cùng số lớp nhãn và giá trị nhãn là giống nhau. Chính vì thế, gây nhiễu cho mô hình và do CNN là mô hình học tốt các đặc trưng cơ bản như cạnh và góc đến các đặc trưng phức tạp như các đặc điểm nổi bật của đối tượng trong hình ảnh nên có thể làm cho mô hình phải học lại các đặc trưng mới nên mô hình sẽ dự đoán tốt hơn đối với tập dữ liệu được huấn luyện sau cùng là tập MNIST-FASHION còn tập MNIST gần như mô hình đã quên sạch do bị thay đổi giá trị trọng số sau khi học tập MNIST-FASHION.

# X. Hướng dẫn cách chạy:

### 1. Đối với huấn luyện riêng biệt tập dữ liệu MNIST:

**Các bước chạy**

**Bước 1:** Mở VSCode.

**Bước 2:** Chọn File → Open Folder → Chọn thư mục Source chứa source code.

**Bước 3:** Nhấp chọn file juypter notebook với tên MNIST.ipynb.

**Bước 4:** Ấn vào nút Run All để khởi chạy chương trình.

**Bước 5:** Chương trình sẽ yêu cầu chọn ngôn ngữ lập trình thì ở đây chúng ta sẽ chọn ngôn ngữ Python 3.6 trở lên.

**Bước 6:** Trong quá trình chạy đảm bảo rằng các thư viện được sử dụng trong chương trình đã được tải về hết với lệnh pip install [tên thư viện].

**Bước 7:** Đợi chương trình chạy hoàn thành và lướt xuống để xem kết quả.

### 2. Đối với huấn luyện riêng biệt MNIST-FASHION:

**Các bước chạy**

**Bước 1:** Mở VSCode.

**Bước 2:** Chọn File → Open Folder → Chọn thư mục Source chứa source code.

**Bước 3:** Nhấp chọn file juypter notebook với tên MNIST-FASHION.ipynb.

**Bước 4:** Ấn vào nút Run All để khởi chạy chương trình.

**Bước 5:** Chương trình sẽ yêu cầu chọn ngôn ngữ lập trình thì ở

đây chúng ta sẽ chọn ngôn ngữ Python 3.6 trở lên.

**Bước 6:** Trong quá trình chạy đảm bảo rằng các thư viện được sử dụng trong chương trình đã được tải về hết với lệnh pip install [tên thư viện].

**Bước 7:** Đợi chương trình chạy hoàn thành và lướt xuống để xem kết quả.

**3. Đối với huấn luyện cả 2 tập cùng lúc:**

**Các bước chạy**

**Bước 1:** Mở VSCode.

**Bước 2:** Chọn File → Open Folder → Chọn thư mục Source chứa source code.

**Bước 3:** Nhấp chọn file juypter notebook với tên MNIST-AND-MNIST-FASHION.ipynb.

**Bước 4:** Ấn vào nút Run All để khởi chạy chương trình.

**Bước 5:** Chương trình sẽ yêu cầu chọn ngôn ngữ lập trình thì ở đây chúng ta sẽ chọn ngôn ngữ Python 3.6 trở lên.

**Bước 6:** Trong quá trình chạy đảm bảo rằng các thư viện được sử dụng trong chương trình đã được tải về hết với lệnh pip install [tên thư viện].

**Bước 7:** Đợi chương trình chạy hoàn thành và lướt xuống để xem kết quả.

# XI. Tài liệu tham khảo:

**[1]** **Slides – Google Drive. (n.d.-b).**

**https://drive.google.com/drive/folders/1DIE9e6TLL8Iz8BacyBnmzZu-8O2MN70I**

**[2]** **Curiousprogrammer. (2019, February 19). *LeNet-5 CNN with Keras - 99.48%*. Kaggle.**

**https://www.kaggle.com/code/curiousprogrammer/lenet-5-cnn-with-keras-99-48**

**[3]** **Blurredmachine. (2020, July 13). *LeNet Architecture: A complete guide*. Kaggle.**

**https://www.kaggle.com/code/blurredmachine/lenet-architecture-a-complete-guide**

**[4]** **Heeraldedhia. (2020, September 19). MNIST Classifier - first Deep Learning project. Kaggle.**

**https://www.kaggle.com/code/heeraldedhia/mnist-classifier-first-deep-learning-project**