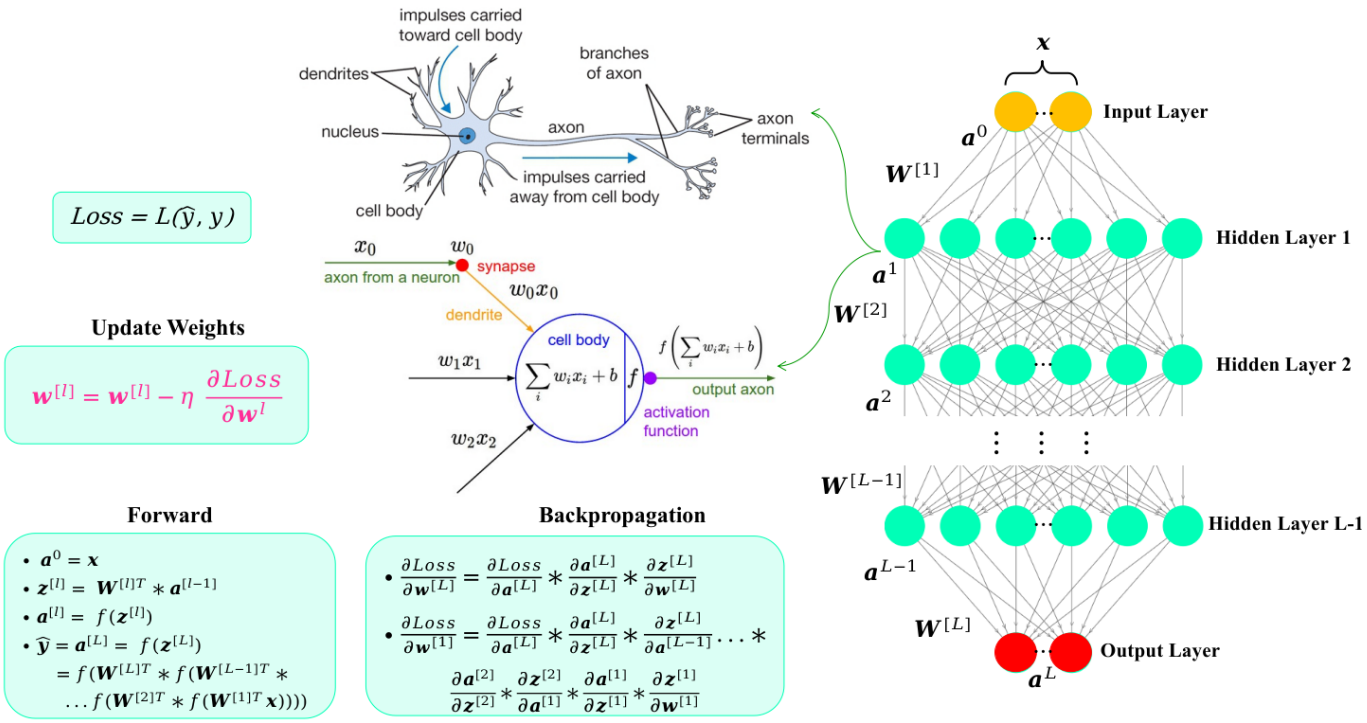
AI VIET NAM – COURSE 2023

MLP - Exercise

Ngày 14 tháng 10 năm 2023

Phần I: Mô Tả Lý Thuyết:

Bài tập tuần này sẽ xoay quanh việc sử dụng Multi-layer Perceptron (MLP) để thực hiện các task như: regression, classification với non-linear data, và classification với image và so sánh với các model đã được học như linear regression model và softmax regression model.



Hình 1: Kiến trúc, thành phần, giai đoạn Forward, Backpropagation của Multi-layer Perceptron (MLP)

1. **Linear Regression, Softmax Regression và MLP model:**

   (a) **Một Số Ký Hiệu**

   i. $\boldsymbol{x}$: Input vector (features)
   ii. $\boldsymbol{y}$: label, kết quả thực tế
   iii. $\hat{\boldsymbol{y}}$: Kết quả dự đoán
   iv. $\boldsymbol{W}$: weights của model (nhiệm vụ của huấn luyện là tìm bộ $\boldsymbol{W}$ để $\boldsymbol{y} \approx \hat{\boldsymbol{y}}$)
   v. Loss: loss function, mục tiêu huấn luyện là tìm loss nhỏ nhất
   vi. n: Số lượng feature của một sample

vii. $f(\cdot)$: Activation Function

viii. m: số lượng sample trong một batch

ix. C: số lượng class cần phân loại

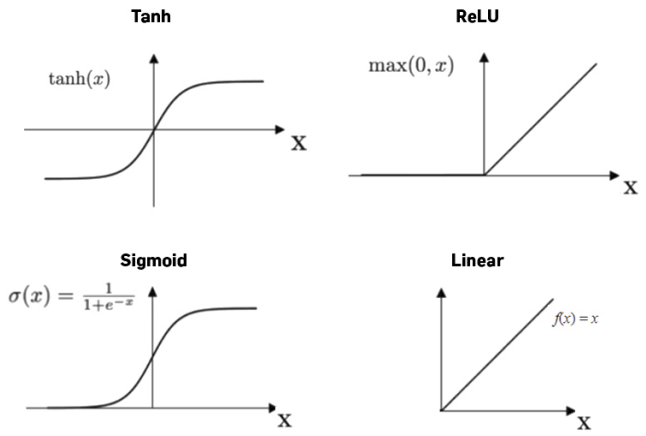
(b) **Linear Regression:** (Hình 3). Sử dụng cho regression task (ví dụ dự đoán giá nhà). $\hat{\boldsymbol{y}} = \boldsymbol{W}^T\boldsymbol{x} = w_0 + w_1x_1 + ... + w_nx_n$. Loss thường được dùng là SE: Loss $= L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{m}\sum_i^m (y_i - \hat{y}_i)^2$

(c) **Softmax Regression:** (Hình 3). Sử dụng cho bài toán phân loại. $\hat{\boldsymbol{y}} = f(\boldsymbol{W}^T\boldsymbol{x}) = f(w_0 + w_1x_1 + ... + w_nx_n)$. Đối với bài toán binary classification thì dùng Logistic Regression với $f(x) = \frac{1}{1+e^{-x}}$, nhiều hơn 2 classes thì dùng Softmax Regression với $f(\boldsymbol{x})_i = \frac{e^{x_i}}{\sum_j^C e^{x_j}}$. Loss là cross entropy $L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\frac{1}{m}\sum_i^m y_i \log \hat{y}_i$ với các class ở dạng one-hot-encoding

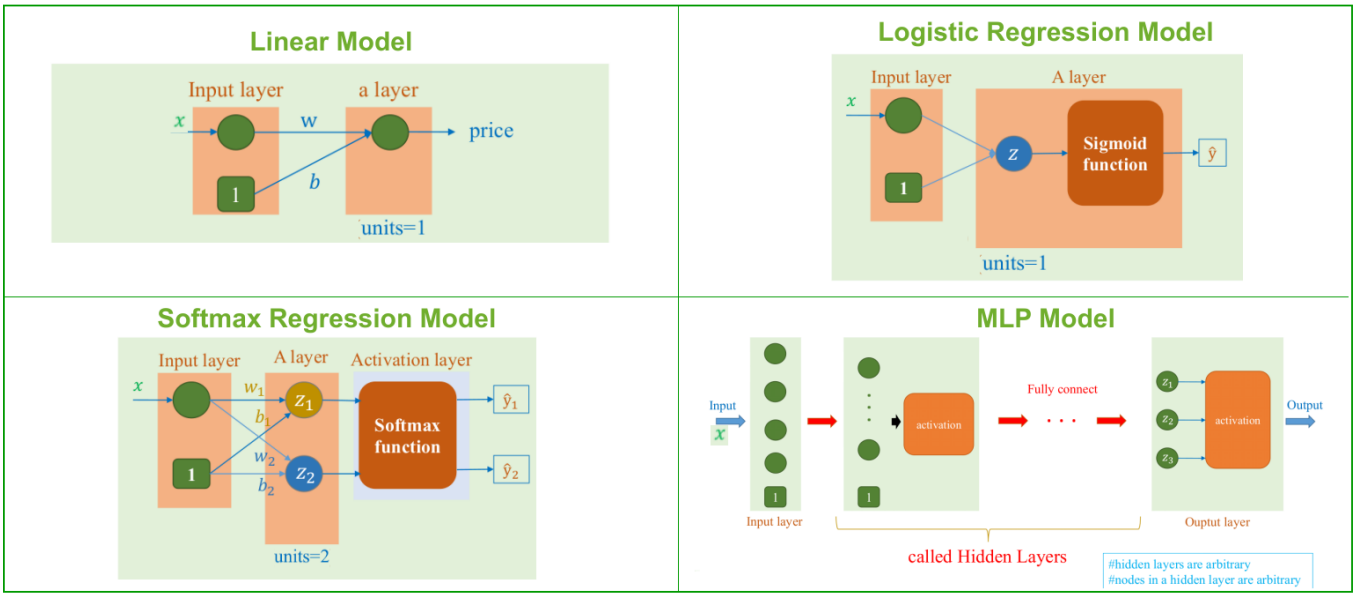
(d) **MLP:** (Hình 1 và hình 3). Là một Neural Network gồm các layer xếp chồng nhau lên nhau hình thành nên một network. Thông thường có 3 loại layer, Input layer, Output layer và Hidden layer. Có thể có nhiều Hidden layer và thường được bắt đầu từ 1 (số lượng hidden layer càng nhiều thì model càng deep). Mỗi layer sẽ có nhiều node (hình tròn), với Input layer là số lượng features, Output layer là số lượng node dùng cho output (ví dụ bài toán classification phân loại 3 classes thì cần 3 nodes), và mỗi Hidden layer là số lượng neuron tham gia vào trích xuất đặc trưng của layer đó. Loss sẽ tùy thuộc vào từng bài toán mà sẽ có function cụ thể. Activation thường được dùng cho MLP (hình 2). Khi train

i. **Forward:** Thông tin sẽ đi theo chiều từ Input layer đến Output layer. Tại mỗi layer 'l' của các Hidden layer thông tin đầu vào sẽ là output từ layer 'l-1' goi là $\boldsymbol{a}^{[l-1]}$. Tại đây ta tính được $\boldsymbol{z}^{[l]} = \boldsymbol{W}^{[l]T}\boldsymbol{a}^{[l-1]}$. Tiếp theo $\boldsymbol{z}^{[l]}$ sẽ đi qua một activation và thu được $\boldsymbol{a}^{[l]} = f(\boldsymbol{z}^{[l]})$. Việc này lặp lại cho đến khi đi qua hết các layer và ra output cuối cùng (hình 1)

ii. **Backpropagation:** Lúc này hàm Loss sẽ được dùng để tính toán sự sai lệch giữa $\boldsymbol{y}$ và $\hat{\boldsymbol{y}}$. Dựa vào thông tin này để gửi phản hồi theo chiều từ Output layer đến Input layer để điều chỉnh các tham số trong mỗi layer (quá trình này là backpropagation). Thông tin phản hồi về chính là gradient và được tính toán dựa vào đạo hàm và chain rule ($\frac{\partial Loss}{\boldsymbol{W}} = \frac{\partial Loss}{\boldsymbol{a}} * \frac{\boldsymbol{a}}{\boldsymbol{z}} * \frac{\boldsymbol{z}}{\boldsymbol{w}}$). Sau đó sử dụng thuật toán gradient descent để update weights. (hình 1)



Hình 2: Một số activation thông dụng



Hình 3: Linear Regression, Logistic Regression, Softmax Regression và MLP model

2. **Normalization**:

   (a) **Standardisation (Z-Score Normalization)**: là kỹ thuật scale các feature (Feature Scaling) quan trọng trong tiền xử lý data. Kỹ thuật này giúp cho việc học được hội tụ nhanh hơn, giúp cho các feature có range độ lớn cách biệt nhau quá lớn về scale gần nhau hơn, không ảnh hưởng bởi outlier (outlier trong ngưỡng chấp nhận được). Standardisation chuẩn hóa giá trị của các feature dựa trên mean và standard deviation của feature đó, mean được đưa về 0 và standard deviation là 1 (công thứ như hình bên dưới).

$$\mathbf{x} = \frac{\mathbf{x} - \mu}{\sigma}$$

Standardisation: **mean** = 0, **std**=1

3. **Metric cho regression task**: Bài tập tuần này sẽ giới thiệu thêm một metric cho linear regression task được gọi là R Squared (coefficient of determination). R Squared thể hiện mức độ phù hợp của model với data. R Square có range [0,1], giá trị càng lớn thì model càng phù hợp với data.

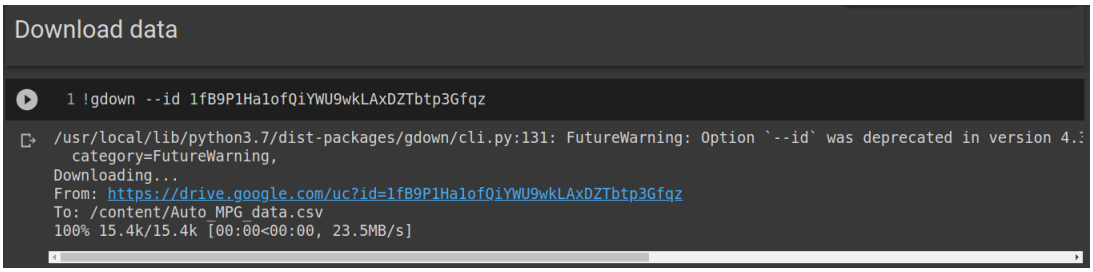
Phần II: Bài Tập và Gợi Ý:

1) Gợi Ý Các Bước Làm Bài:

1. **Giới thiệu sơ lược về bài tập**: Bài tập tuần này sẽ có gồm 3 bài tập (Chi tiết xem phần Yêu cầu bài tập)

   - **Bài 1**: Task về regression cho một tập data **Auto_MPG_data.csv** dựa vào 9 features của xe ô tô để dự đoán năng lượng tiêu thụ (MPG)
   - **Bài 2**: Task về classification (nonlinear data). Cho 1 tập data **NonLinear_data.npy** dựa vào 2 features để phân loại 3 classes
   - **Bài 3**: Task về classification ảnh. Cho 1 tập data **FER-2013.zip** gồm các ảnh thể hiện 7 (classes) cảm xúc khác nhau trên khuôn mặt con người
   - Các bạn sẽ thực hiện theo hướng dẫn bên dưới để làm các bài tập

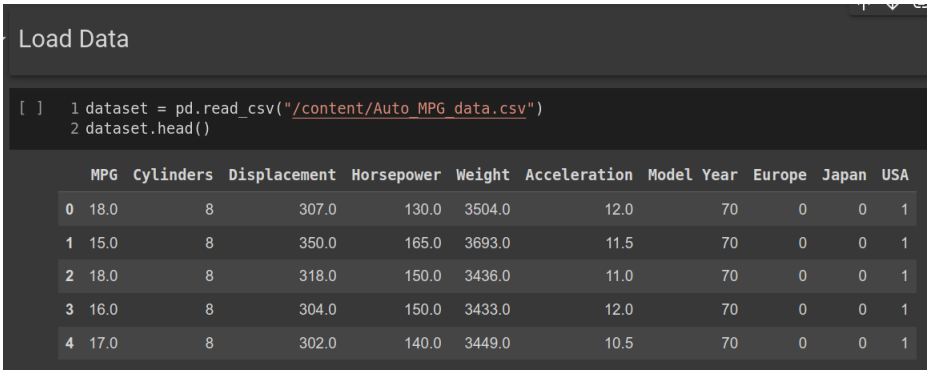
2. **Download Data**: Bước đầu tiên là download data cho các bài tập bằng cách thực hiện chạy lệnh "!gdown" như ảnh bên dưới. Mỗi bài tập đều đã được code sẵn các bạn chỉ cần chạy lại cell đầu tiên của mục Download Data trong mỗi file hint (Các file này sẽ được gửi kèm file bài tập)



3. **Giải Nén Data (Chỉ dành cho bài tập 3)**: Bài tập 3 sử dụng data dạng ảnh. Khi download data về sẽ là một file nén dạng .zip, do đó các bạn cần giải nén ra bằng lệnh như hình bên dưới

4. **Load Data**: Vì mỗi bài tập data được lưu trữ theo từng dạng khác nhau nên các bước load data cũng khác nhau.

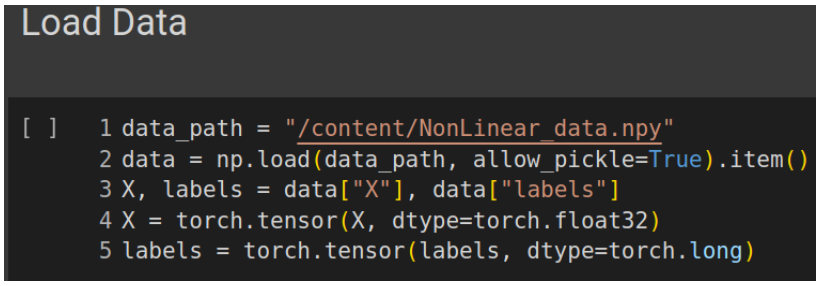
   (a) **Bài 1**: Data của bài tập 1 là dạng table dành cho task regression và được lưu dưới dạng file .csv. Do đó, để thuận tiện thư viện Pandas được sử dụng để load data từ file csv. Mục đích sử dụng Pandas vì thư viện này hỗ trợ khá nhiều method để xử lý và làm việc trên data đặc biệt là dạng table. Sau khi load data như line 1 (method read_csv nhận đường dẫn đến file .csv) của hình bên dưới ta sẽ thu được một dataframe có shape là (392,10) (số lượng samples, số lượng features 9 + 1 lable (MPG)). Line 2 hiển thị 5 samples đầu tiên của data.



	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	Europe	Japan	USA
0	18.0	8	307.0	130.0	3504.0	12.0	70	0	0	1
1	15.0	8	350.0	165.0	3693.0	11.5	70	0	0	1
2	18.0	8	318.0	150.0	3436.0	11.0	70	0	0	1
3	16.0	8	304.0	150.0	3433.0	12.0	70	0	0	1
4	17.0	8	302.0	140.0	3449.0	10.5	70	0	0	1

Hình 4: Cách load data bài 1

(b) **Bài 2**: Data của bài tập 2 cũng được xem là dạng bảng và dành cho classification task. Tuy nhiên, data lại được lưu dưới dạng .npy (Python NumPy Array File). Do đó, ta sẽ sử dụng thư viện Numpy để load data từ file .npy. Line 2 thể hiện cách load data từ file .npy (method load nhận đường dẫn đến file .npy) và sẽ thu được data là dictionary. Line 3, thể hiện cách lấy data để đưa vào model huấn luyện, với X có shape (300, 2) (số lượng sample, số lượng features) và labels là một vector có 300 element thể hiện class tương ứng với từng sample ở X (có tổng cộng 3 classes). Line 4, 5 là ép kiểu data

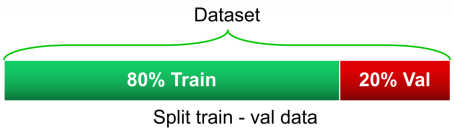


Hình 5: Cách load data bài 2

(c) **Bài 3**: Do cách load data dạng ảnh phức tạp hơn nên sẽ được miêu tả chi tiết ở mục Load và Xử Lý Data Dạng Ảnh

5. **Tiền Xử Lý Data Dạng Table**:

(a) **Lấy label và chia bộ dữ liệu train/val**: Ở bài tập 1 (bài 2 không cần chia) ta sẽ chia data thành 2 bộ data là train và validation (val). Hình 7 thể hiện cách chia bộ dữ liệu và tách lấy label từ table. Tại line 1 của cell 1, dataset sẽ được lấy 80% và dùng làm cho việc huấn luyện (train_dataset). Tại line 2 của cell 1, 20% còn lại của dataset sẽ được dùng cho việc đánh giá model (val_dataset). Tiếp theo, ở cell 2, line 4-5 thể hiện việc tách feature MPG ra khỏi table và dùng làm target mà chúng ta sẽ xây dựng model dự đoán dựa trên 9 features còn lại. Line 7-10 thể hiện việc ép kiểu các data sang dạng float32



Hình 6: Dataset được chia thành 80% cho train và 20% cho validation

```
[ ]  1 X_train = train_dataset.copy()
     2 X_val = val_dataset.copy()
     3
     4 y_train = X_train.pop('MPG')
     5 y_val = X_val.pop('MPG')
     6
     7 X_train = torch.tensor(X_train.values, dtype=torch.float32)
     8 y_train = torch.tensor(y_train.values, dtype=torch.float32)
     9 X_val = torch.tensor(X_val.values, dtype=torch.float32)
    10 y_val = torch.tensor(y_val.values, dtype=torch.float32)
```

Hình 7: Lấy label và chia bộ dữ liệu train/val

(b) **Chuẩn hóa dữ liệu (Data Standardisation)**: Ta sẽ Standardise 9 features để các features này có mean = 0, và standard deviation = 1, điều này giúp cho model học tốt hơn vì model sẽ học thông tìn từ cả 9 features chứ không bị bias vào một feature có giá trị vượt trội hơn so với các feature khác dẫn đến việc model học không như mong muốn. Đầu tiên ta sẽ tính mean và standard deviation của từng feature trên tập train. Sau đó ta dùng hai giá trị này và áp dụng công thức stadardisation cho cả tập train và val. Cuối cùng ta thu được các samples của tập train và val đã được chuẩn hóa

$$\mathbf{x} = \frac{\mathbf{x} - \mu}{\sigma}$$

Standardisation: **mean** = 0, **std**=1

Chuẩn hóa dữ liệu (Data Standardisation)

```
1 _MEAN = X_train.mean(axis=0)
2 _STD = X_train.std(axis=0)
3
4 X_train = ( X_train-_MEAN)/_STD
5 X_val = ( X_val-_MEAN)/_STD
```

Hình 8: Chuẩn hóa dữ liệu (Data Standardisation)

6. **Xây dựng data pipeline với Dataset và DataLoader cho dữ liệu dạng bảng (bài 1 và 2)**: Đầu tiên, chúng ta định nghĩa một tập dữ liệu tùy chỉnh CustomDataset dựa trên lớp Dataset của PyTorch. Tập dữ liệu này cho phép chúng ta truy cập vào dữ liệu theo chỉ số và lấy ra số lượng sample. Sau khi định nghĩa xong, chúng ta tạo ra hai tập dữ liệu riêng biệt cho quá trình train và val. Cuối cùng, chúng ta sử dụng DataLoader để đóng gói các tập dữ liệu này, giúp chúng ta dễ dàng lấy ra dữ liệu theo batch và xáo trộn dữ liệu trong quá trình huấn luyện. (Hình 9)

- **Cell 1**: Định nghĩa một tập dữ liệu tùy chỉnh có tên là CustomDataset, kế thừa từ lớp Dataset của PyTorch.
  - 2-4: Phương thức __init__ khởi tạo tập dữ liệu với các features X và label y.
  - 6-7: Phương thức __len__ trả về số lượng sample trong data
  - 8-11: Phương thức __getitem__ cho phép truy cập vào data theo chỉ số idx, trả về các features và label cho một chỉ số idx cụ thể.
- **Cell 2**: Tạo train và val loader
  - 1: Tạo một train data sử dụng lớp CustomDataset, khởi tạo với X_train và y_train.
  - 2: Tạo một val data tương tự sử dụng X_val và y_val.

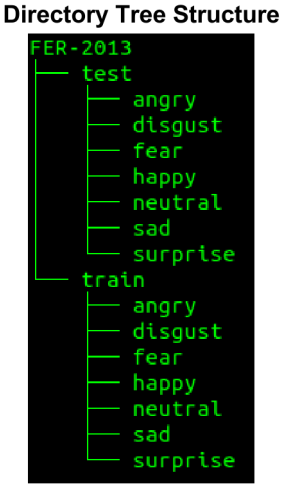
- 3: Tạo một DataLoader cho tập train data với kích thước batch là 32 và có kích hoạt chức năng xáo trộn. Việc xáo trộn dữ liệu huấn luyện là một phương pháp phổ biến để đảm bảo rằng mô hình nhận được một thứ tự dữ liệu khác nhau trong mỗi epoch, giúp mô hình tổng quát hóa tốt hơn.
- 4: Tạo một DataLoader cho val data với kích thước batch là 32.

```python
class CustomDataset(Dataset):
    def __init__(self, X, y):
        self.X = X
        self.y = y

    def __len__(self):
        return len(self.y)

    def __getitem__(self, idx):
        # print(idx)
        return self.X[idx], self.y[idx]
```

```python
train_dataset = CustomDataset(X_train, y_train)
val_dataset = CustomDataset(X_val, y_val)
train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=32)
```

Hình 9: Xây dựng data pipeline với Dataset và DataLoader

7. **Load và Xử Lý Data Dạng Ảnh (Chỉ dành cho bài tập 3)**: Vì data của bài 3 là dạng ảnh nên việc load và xử lý sẽ phức tạp hơn. Bài tập này sẽ cũng cấp cho các bạn 1 cách để load và xử lý data. Cách load data này trong PyTorch cung cấp sự linh hoạt cho người dùng để xác định cách data được load và tiền xử lý, đồng thời tận dụng sức mạnh của DataLoader để tải dữ liệu một cách hiệu quả. Lưu ý rằng ta cần thiết kế để đọc image từ các thư mục, với mỗi thư mục đại diện cho một class của ảnh. Điều này rất phổ biến trong các bài toán phân loại hình ảnh, nơi dữ liệu được tổ chức theo cấu trúc thư mục dựa trên các class.



Hình 10: Cấu trúc thư mục của tập data FER-2013 sau khi được giải nén ra. Các tên của thư mục con cũng chính là class tương ứng của ảnh

Trong PyTorch, việc tải dữ liệu thường được thực hiện thông qua hai khái niệm chính: Dataset và DataLoader.

- Dataset: Đây là một lớp trừu tượng mà mọi tập dữ liệu tùy chỉnh cần kế thừa. Lớp này yêu cầu hai phương thức cần được ghi đè:
  - $\_\_len\_\_$: Trả về tổng số mẫu trong tập dữ liệu.
  - $\_\_getitem\_\_$: Trả về một mẫu dữ liệu cụ thể dựa trên chỉ số.

  Khi tạo một Dataset tùy chỉnh, người dùng có thể xác định cách dữ liệu được đọc từ nguồn gốc (ví dụ: từ đĩa cứng), cũng như cách nó được tiền xử lý trước khi được trả về.
- DataLoader: Đây là một trình tải dữ liệu có thể lặp lại, cho phép truy cập vào dữ liệu trong tập dữ liệu theo cách hiệu quả, thường là theo batch. Nó cung cấp các tính năng như tự động chia batch, xáo trộn dữ liệu và tải dữ liệu song song sử dụng nhiều tiến trình.

  Cách này cho phép người dùng xác định cách dữ liệu của họ được lưu trữ, đọc và tiền xử lý. Điều này rất hữu ích khi dữ liệu không tuân theo một định dạng tiêu chuẩn hoặc khi người dùng muốn áp dụng các bước tiền xử lý dữ liệu cụ thể.

```python
class ImageDataset(Dataset):
    def __init__(self, img_dir, norm):
        self.resize = Resize((img_height, img_width))
        self.norm = norm
        self.img_dir = img_dir
        self.classes = os.listdir(img_dir)
        self.image_files = [(os.path.join(cls, img), cls)
                            for cls in self.classes
                            for img in os.listdir(os.path.join(img_dir, cls))]
        self.class_to_idx = {cls: idx for idx, cls in enumerate(self.classes)}
        self.idx_to_class = {idx:cls  for cls, idx in self.class_to_idx.items()}

    def __len__(self):
        return len(self.image_files)

    def __getitem__(self, idx):
        img_path, cls = self.image_files[idx]
        image = self.resize(read_image(os.path.join(self.img_dir, img_path)))
        image = image.type(torch.float32)
        label = self.class_to_idx[cls]
        if self.norm:
            image = (image/127.5) - 1
        return image, label
```

Hình 11: ImageDataset Class

ImageDataset kế thừa từ lớp Dataset. Lớp này được thiết kế để xử lý dữ liệu hình ảnh được tổ chức vào các thư mục, nơi mỗi thư mục đại diện cho một lớp hoặc danh mục hình ảnh riêng biệt.

- Step1: Initialization ($\_\_init\_\_$ method):
  - Tham số img_dir chỉ định thư mục nơi chứa các thư mục hình ảnh (class).
  - Tham số norm là một giá trị boolean quyết định liệu dữ liệu hình ảnh có nên được chuẩn hóa hay không.
  - Thuộc tính resize được sử dụng để thay đổi kích thước hình ảnh theo chiều cao và chiều rộng đã chỉ định (img_height và img_width đã được định nghĩa).

  - Thuộc tính classes liệt kê tất cả các thư mục con (class) có trong img_dir.
  - Thuộc tính image_files tạo một list các elemnt về ảnh. Mỗi element chứa đường dẫn đến một image và class tương ứng của nó. Điều này được thực hiện bằng cách lặp lại qua mỗi class và image tương ứng của nó.
  - Thuộc tính class_to_idx tạo một dictionary ánh xạ mỗi class thành một index số nguyên duy nhất.
  - Thuộc tính idx_to_class tạo một ánh xạ ngược từ index số nguyên trở lại class tương ứng của nó.
- Step2: Length (__len__ method)
  - Phương thức này trả về tổng số hình ảnh có trong tập dữ liệu.
- Step3: Get Item (__getitem__ method):
  - Với một chỉ số idx, phương thức này lấy đường dẫn hình ảnh và class của nó từ danh sách image_files.
  - Hình ảnh sau đó được đọc từ disk, thay đổi kích thước và chuyển đổi thành một tensor dạng float.
  - Class của hình ảnh được ánh xạ thành index số nguyên tương ứng của nó bằng cách sử dụng dictionary class_to_idx.
  - Nếu thuộc tính norm được đặt thành True, dữ liệu hình ảnh được chuẩn hóa trong khoảng [-1, 1].
  - Cuối cùng, phương thức trả về hình ảnh đã xử lý và label của nó (index số nguyên).

Chúng ta cần chuẩn bị hai bộ dữ liệu: một cho việc training và một cho validation.Những bộ dữ liệu này có thể được sử dụng trong vòng lặp training để lấy các batch hình ảnh và class cho việc train và validate model. DataLoader của PyTorch, giúp load data một cách hiệu quả và song song. (Bên dưới là ví dụ cho trường hợp không dùng chuẩn hoá)

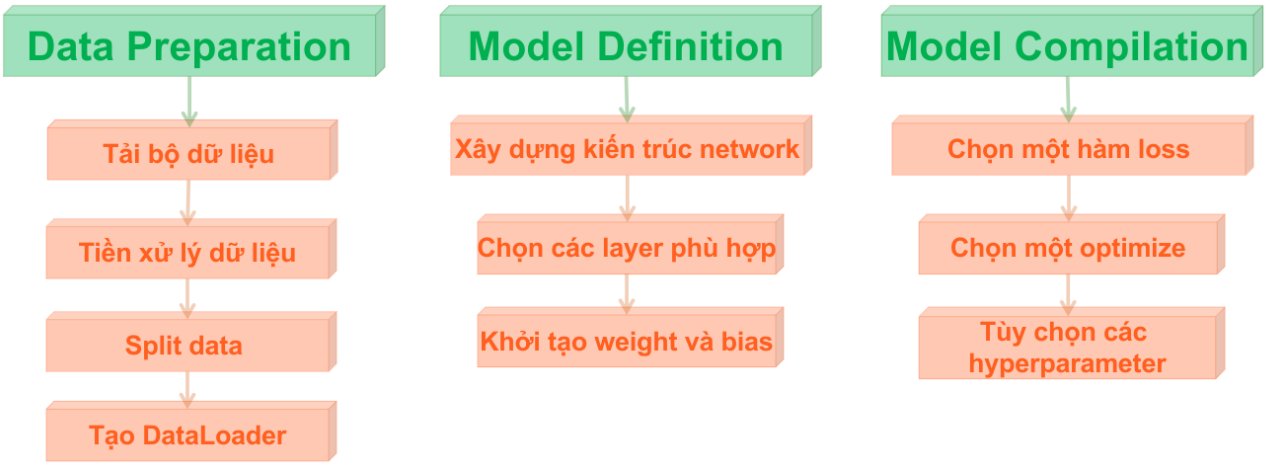
```
train_dataset = ImageDataset(train_dir, False)
train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)

val_dataset = ImageDataset(val_dir, False)
val_loader = DataLoader(val_dataset, batch_size=batch_size, shuffle=False)
```

Hình 12: Load Train and Validation Dataset

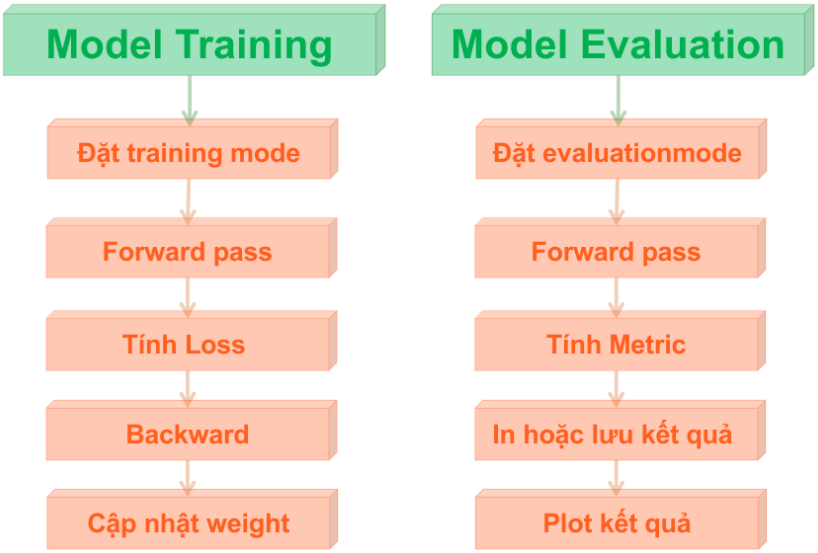
- Training Dataset and DataLoader:
  - train_dataset: Một instance của lớp ImageDataset được tạo ra sử dụng thư mục train_dir. Tham số False chỉ ra rằng hình ảnh sẽ không được chuẩn hóa.
  - train_loader: Một instance của DataLoader được tạo ra sử dụng train_dataset. Tham số batch_size xác định số lượng mẫu trong mỗi batch. Tham số shuffle=True đảm bảo rằng dữ liệu huấn luyện sẽ được xáo trộn ở đầu mỗi epoch, điều để đảm bảo rằng model không ghi nhớ thứ tự của các training sample và cải thiện sự hội tụ trong quá trình train.
- Validation Dataset and DataLoader:
  - val_dataset: Tương tự, một instance của lớp ImageDataset được tạo ra cho tập validation data sử dụng thư mục val_dir, mà không áp dụng chuẩn hóa.
  - val_loader: Một instance của DataLoader được tạo ra cho validation data. Khác với train data, tham số shuffle được đặt thành False, bởi vì trong quá trình validation, thứ tự của các mẫu không quan trọng và việc xáo trộn không cần thiết.

8. **Tóm tắt các bước cơ bản để thực hiện bài tập tuần này với Pytorch** Các bước sau đây sẽ hướng dẫn bạn qua quá trình xây dựng, huấn luyện và đánh giá kết quả cho bài tập tuần này. Tuy nhiên, mỗi bài tập sẽ có những yêu cầu riêng biệt, và chúng sẽ được chi tiết hơn trong file gợi ý.

- Data Preparation:
  - Tải bộ dữ liệu (ví dụ: ER-2013).
  - Tiền xử lý dữ liệu: Normalize, Augmentation, ...
  - Split data thành train, val, test set.
  - Tạo DataLoader cho việc tạo data và xáo trộn.
- Model Definition
  - Xây dựng kiến trúc network.
  - Chọn các layer phù hợp (ví dụ: Linear, ReLU).
  - Khởi tạo weight và bias
- Model Compilation
  - Chọn một hàm loss (ví dụ: CrossEntropyLoss cho phân loại).
  - Chọn một optimizer (ví dụ: Adam, SGD)
  - Tùy chọn các hyperparameter khác
- Model Training
  - Đặt mô hình ở chế độ training.
  - Đối với mỗi epoch:
    - Đối với mỗi batch trong dữ liệu huấn luyện:
      - Forward pass: Tính kết quả dự đoán.
      - Tính Loss.
      - Backward: Tính gradient.
      - sử dụng optimizer để cập nhật weight.
    - Tùy chọn, kiểm tra trên một tập con của dữ liệu và in kết quả.
- Model Evaluation
  - Đặt mô hình ở evaluation mode.
  - Đối với mỗi epoch:
    - Đối với mỗi batch trong validation data:
      - Forward pass: Tính kết quả dự đoán.
      - Tính Metric
      - In hoặc lưu kết quả đánh giá.
  - Thể hiện bằng hình ảnh bằng cách sử dụng matplotlib: Đã có sẵn mỗi đoạn code plot loss và metric (nếu có dùng) ở mỗi phần training theo yêu cầu đề bài các bạn chỉ cần chạy các cell có matplotlib sau khi đã train xong model.



Hình 13: Các bước yêu cầu trong việc build model cơ bản trong bài tập tuần này



Hình 14: Các bước yêu cầu trong việc build train cơ bản trong bài tập tuần này

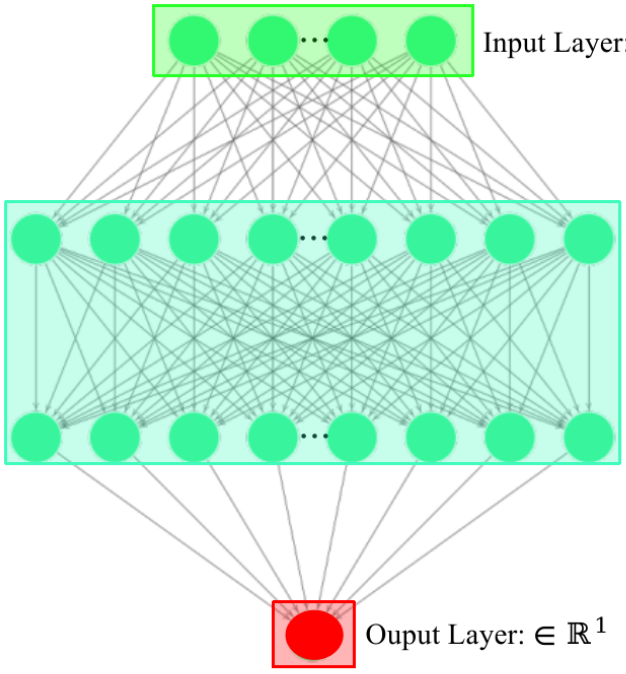
2) Yêu Cầu Bài Tập: CÁC BẠN DÙNG CÁC FILE HINT ĐỂ LÀM BÀI TẬP (LINK)

1. **MLP FOR REGRESSION:** Cho 1 tập data **Auto_MPG_data.csv** thống kê việc số lượng nhiên liệu được tiêu thụ (MPG) dựa trên 9 features của xe ô tô từ thập niên 70s đến thập niên 80s. Các bạn hãy xử lý data và xây dựng model sau để dự đoán nhiên liệu tiêu thụ của xe ô tô:

   (a) Xây dựng linear regression model: Loss = Mean Square Error, Epochs=100, Batch size=32, Optimizer = SGD, Learning rate = 0.1

   (b) Xây dựng MLP regression model: Hidden layers=2, Nodes cho mỗi layer=64, Activation cho hidden layer = ReLu, Loss = Mean Square Error, Epochs=100, Batch size=32, Optimizer = SGD. Learning rate = 0.003. Các bạn thực hiện mạng MLP như hình 15

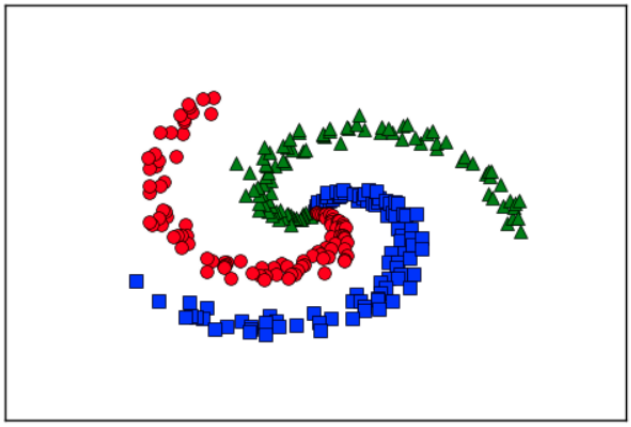
   (c) So sánh và nhận xét kết quả giữa 2 model




```
MLP(
  (linear1): Linear(in_features=9, out_features=64, bias=True)
  (linear2): Linear(in_features=64, out_features=64, bias=True)
  (output): Linear(in_features=64, out_features=1, bias=True)
  (relu): ReLU()
)
```

Hình 15: Kiến trúc mạng MLP như yêu cầu bài tập 1 (b), phần summary model (đen) không bao gồm input layer

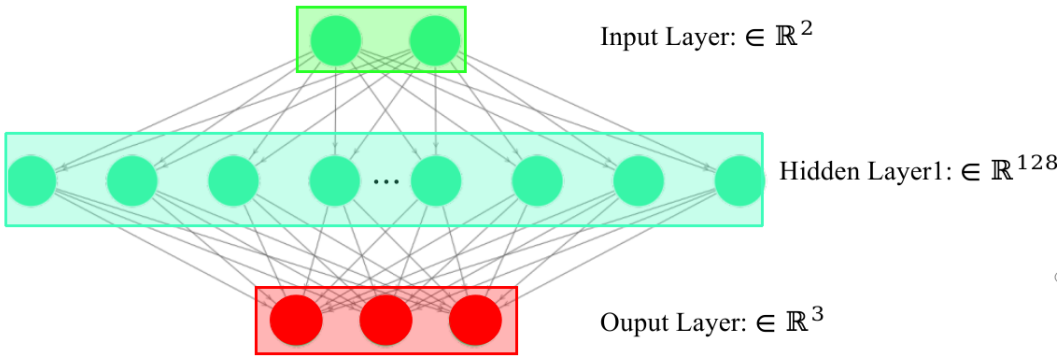
2. **MLP FOR CLASSIFICATION (nonlinear data)** Cho 1 tập data **NonLinear_data.npy** (data này không cần xử lý) là nonlinear data gồm label (có 3 class khác nhau) và 2 features (tọa độ x và y). Các bạn hãy xây dựng model sau để phân loại đúng được 3 class này:



   (a) Xây dựng softmax regression model: Loss= Cross Entropy, Metric = Accuracy, Epochs=200, Batch size=32, Optimizer=SGD, Learning rate = 0.1

   (b) Xây dựng MLP classification model: Hidden layers=1, Nodes cho mỗi layer=128, Activation cho hidden layer = ReLu, Loss = Cross Entropy, Metric = Accuracy, Epochs=500, Learning rate=0.1, Batch size = 32, Optimizer = SGD. Các bạn thực hiện mạng MLP như hình 16

   (c) So sánh và nhận xét kết quả giữa 2 model

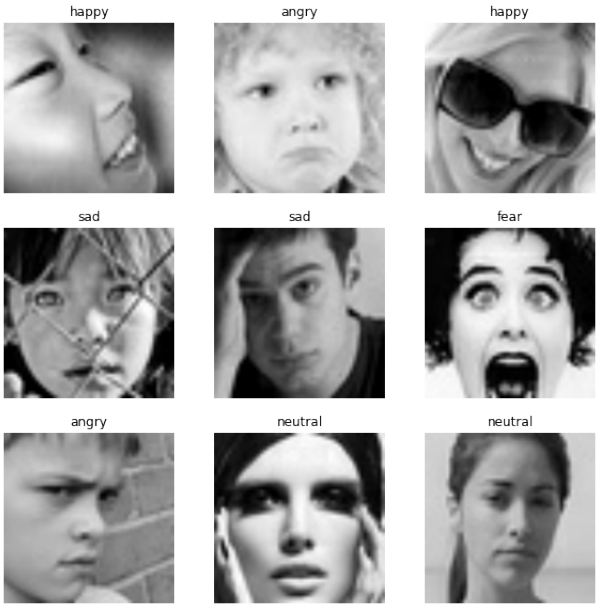



```
MLP(
  (linear1): Linear(in_features=2, out_features=128, bias=True)
  (output): Linear(in_features=128, out_features=3, bias=True)
  (relu): ReLU()
)
```

Hình 16: Kiến trúc mạng MLP như yêu cầu bài tập 2 (b), phần summary model (đen) không bao gồm input layer

3. **MLP FOR CLASSIFICATION (image data)** Cho 1 tập data **FER-2013.zip** gồm các ảnh thể hiện 7 cảm xúc khác nhau trên khuôn mặt con người. Các bạn hãy xử lý data và xây dựng

model sau để dự phân loại được cảm xúc gương mặt được thể hiện trong ảnh input (Dựa vào file hint các bạn có thể chọn một trong hai các load và xử lý data để làm bài):



(a) Xây dựng sofmax regression model: Loss= Cross Entropy, Metric = Accuracy, Epochs=100, Learning rate=0.006, Batch size=256, Optimizer = SGD, (Không Normalize ảnh).

(b) Xây dựng sofmax regression model: Loss= Cross Entropy, Metric = Accuracy, Epochs=100, Learning rate=0.006, Batch size=256, Optimizer = SGD, (Ảnh được Normalize theo công thức $x = \frac{x}{127.5} - 1$).

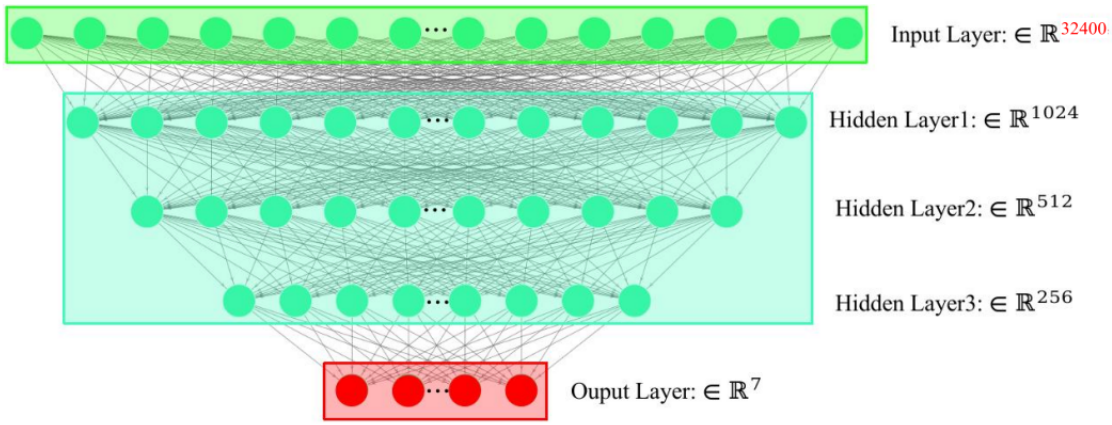
(c) Xây dựng MLP classification model có 3 Hidden layers với số lương Node lần lượt là 1024, 512, 256, Activation cho hidden layer là **Tanh**, Loss= Cross Entropy, Metric = Accuracy, Epoch=100, Learning rate=0.006, Batch size=256, Optimizer = SGD, (Ảnh được Normalize theo công thức $x = \frac{x}{127.5} - 1$). (Layer cuối không có activation function). Các bạn thực hiện mạng MLP như hình 17

(d) Xây dựng MLP classification model có 3 Hidden layers với số lương Node lần lượt là 1024, 512, 256, Activation cho hidden layer là **ReLu**, Loss= Cross Entropy (SparseCategoricalCrossentropy), Metric = Accuracy, Epoch=100, Learning rate=0.006, Batch size=256, Optimizer = SGD, (Ảnh được Normalize theo công thức $x = \frac{x}{127.5} - 1$).(Layer cuối không có activation function). Các bạn thực hiện mạng MLP như hình 17

(e) So sánh và nhận xét kết quả giữa các models.

NOTE: Các bạn lưu ảnh ảnh sau khi được xử lý sẽ có shape (180x180), để có thể train được với Sofmax Regression model và MLP thì cần flatten thành vector có 180x180x1=32400 elements

NOTE: Ở mức độ bài học hiện tại chỉ cần model MLP có thể học được từ tập train dataset (ví dụ loss giảm sau mỗi epoch, ...) là đạt yêu cầu. Không cần phải giải quyết các vấn đề khác như overfitting cho tập data này vì đây là tập data tương đối khó cho MLP model cần thực hiện thêm nhiều kỹ thuật khác để tăng accuracy cho validation dataset (ví dụ accuracy của train dataset có thể cao hơn nhiều so với accuracy của validation $<$ 0.4 vẫn đạt yêu cầu của bài tập này)




```
MLP(
  (linear1): Linear(in_features=32400, out_features=1024, bias=True)
  (linear2): Linear(in_features=1024, out_features=512, bias=True)
  (linear3): Linear(in_features=512, out_features=256, bias=True)
  (output): Linear(in_features=256, out_features=7, bias=True)
  (relu): ReLU()
)
```

Hình 17: Kiến trúc mạng MLP như yêu cầu bài tập 3 (c và d), phần summary model (đen) có bao gồm input layer (ảnh 180x180 được flatten thành vector 180x180=32400 elments)

4. (Optional): Dựa trên các tập data trong bài giảng và và bài tập, các bạn xây dựng lựa chọn các hyperparameter phù hợp để MLP model đạt được accuarcy tốt nhất.
5. Link notebook (file hint) dùng để download và tiền xử lý data:
   (a) **1a:** LINK
   (b) **1b:** LINK
   (c) **2a:** LINK
   (d) **2b:** LINK
   (e) **3a:** LINK
   (f) **3b:** LINK
   (g) **3c:** LINK
   (h) **3d:** LINK

Phần III: Trắc Nghiệm:

1. Linear regression là một thuật toán phù hợp cho:
   (A). Regression
   (B). Binary Classification
   (C). Multi-class classification
   (D). Regression và Classification

2. Logistic regression là một thuật toán phù hợp cho:
   (A). Regression
   (B). Binary Classification
   (C). Multi-class classification
   (D). Regression và Classification

3. Softmax regression là một thuật toán phù hợp cho:

   (A). Regression
   (B). Binary Classification
   (C). Multi-class classification
   (D). Regression và Classification

4. Multilayer perceptron là một thuật toán phù hợp cho:

   (A). Regression
   (B). Binary Classification
   (C). Multi-class classification
   (D). Regression và Classification

5. Phân loại hoa dựa trên chiều dài cánh bằng cách sử dụng MLP model có 1 hidden layer chứa 2 neuron (trọng số $w_{hidden} = [0.5, -0.5]$, và bias $b_{hidden} = [1, -1]$) đi kèm với hàm ReLU và output layer chứa 1 neuron (trọng số $w_{output} = [0.3, -0.2]$, và bias $b_{output} = 0.5$) đi kèm với hàm Sigmoid. Nếu cho input x = 5.1 thì model phân loại thuộc loại nào (> 0.5 được xem như loại 1):

Chiều dài cánh	Loại cánh
5.1	0
6.0	1
5.7	0

Bảng 1: Bảng dữ liệu phân loại cánh hoa theo chiều dài cánh

   (A). Loại 0
   (B). Loại 1

6. Tính kết quả dự đoán giá nhà dựa vào diện tích bằng cách sử dụng MLP model có 1 hidden layer chứa 2 neuron (trọng số $w_{hidden} = [0.5, -0.5]$, và bias $b_{hidden} = [1, -1]$)đi kèm với hàm ReLU và output layer chứa 1 neuron (trọng số $w_{output} = [1.5, 2]$, và bias $b_{output} = 0$). Nếu cho input x = 50 thì model dự đoán giá nhà là bao nhiêu:

Diện tích	Giá nhà
50	100
60	120
70	140

Bảng 2: Bảng dữ liệu mô phỏng giá nhà theo diện tích

   (A). 39
   (B). 390
   (C). 93
   (D). 930

7. Multilayer perceptron model có các loại layer cơ bản nào:

   (A). Input layer
   (B). Hidden layer
   (C). Output layer
   (D). Tất cả đều đúng

8. Khi một neuron trong MLP có đầu vào **x** là [2, 3] và trọng số **w** là [0.5, -0.5], bias **b** bằng 1 và hàm activation là hàm ReLU, đầu ra là:

(A). 0.5 (B). 1.0
(C). 1.5 (D). 0.0

9. Trong context của neural networks, "backpropagation" được sử dụng để:

(A). Tính toán đầu vào cho mạng (B). Tính toán loss output layer
(C). Cập nhật trọng số dựa trên loss (D). Tăng số lượng hidden layers

10. Nếu bạn có một MLP model với một hidden layer chứa 5 neurons và output layer có 2 neurons, số trọng số tối thiểu bạn cần là:

(A). 5 (B). 7
(C). 10 (D). 12

11. Trong quá trình backward, đạo hàm của hàm kích hoạt sigmoid $\sigma(x)$ là:

(A). $\sigma(x)(1-\sigma(x))$ (B). $1-\sigma(x)$
(C). $\sigma(x)+1$ (D). $\sigma(x)-1$

12. Cho dữ liệu điểm số của 3 sinh viên là 85,90,78. Sử dụng Z-score normalization $z = \frac{x - mean}{StandardDeviation}$ để chuẩn hoá dữ liệu điểm của sinh viên trước khi đưa vào training. Kết quả sau chuẩn hoá là (chọn kết quả gần đúng nhất):

(A). [0.14, 1.15, -1.29] (B). [1.14, 1.15, -1.29]
(C). [2.14, 2.15, -2.29] (D). [3.14, 3.15, -3.29]