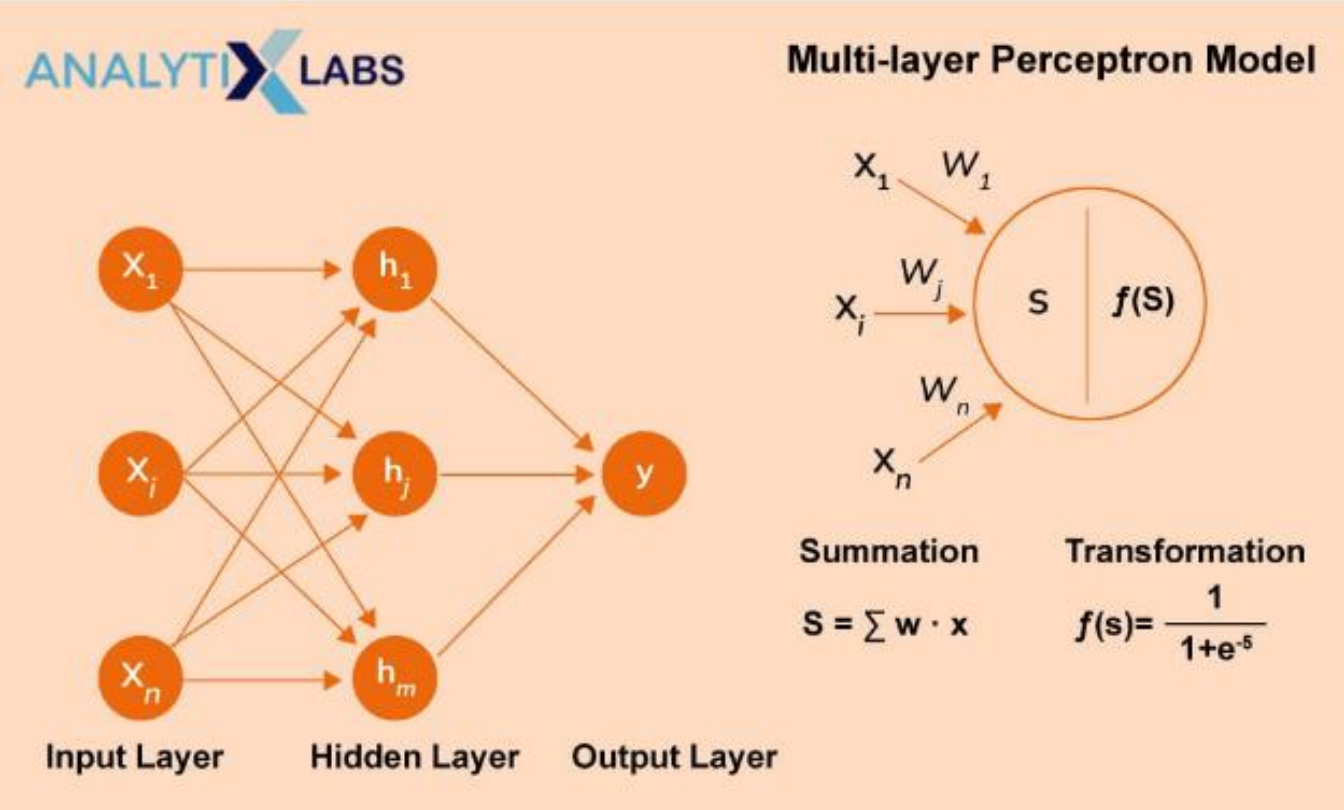
AI VIETNAM
All-in-One Course
(TA Session)

EXERCISE - MULTI-LAYER PERCEPTRON



Content

- **Review MLP**
- **MLP For Regression**
  - Dự đoán năng lượng tiêu thụ (MPG) của xe ô tô dựa trên 9 features (thông số kỹ thuật của xe)
  - Chuẩn bị data trước khi train
  - Train 2 model: linear regression và MLP regression model
  - So sánh kết quả
- **MLP For Classification (Non-Linearly Separable data)**
  - Phân loại 3 class thuộc dạng non-linearly separable dựa trên tọa độ x và y
  - Train 2 model: sofmax regresion và MLP classification model
  - So sánh kết quả

Content

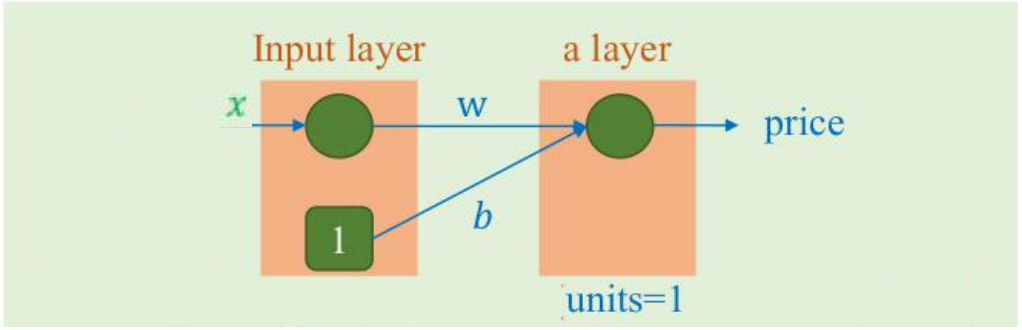
- **MLP For Classification (Image data)**
  - Phân loại loại cảm xúc của một người dựa trên ảnh đầu vào là gương mặt thể hiện cảm xúc của người đó
  - Chuẩn bị data trước khi train
  - Train 4 models:
    - Softmax regression (không normalize)
    - Softmax regression (normalize [-1, 1])
    - MLP - tanh (normalize [-1, 1])
    - MLP - relu (normalize [-1, 1])
  - So sánh kết quả
  - Giới thiệu cơ bản về Sigmoid, Tanh, Relu
- **Modern Uses of MLPs**

Content

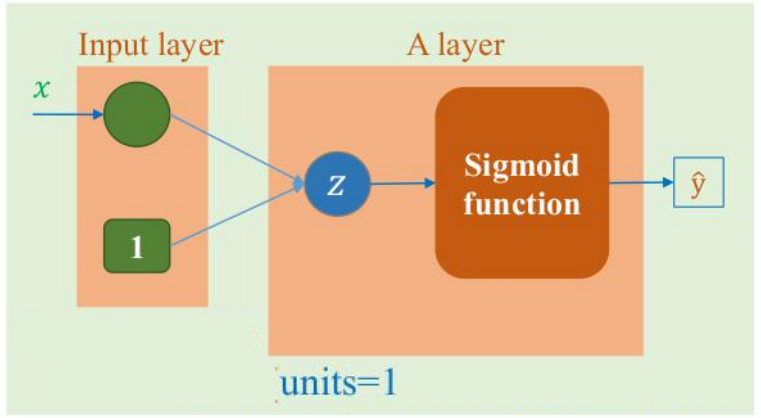
- **Review MLP**
- **MLP For Regression**
  - Dự đoán năng lượng tiêu thụ (MPG) của xe ô tô dựa trên 9 features (thông số kỹ thuật của xe)
  - Chuẩn bị data trước khi train
  - Train 2 model: linear regression và MLP regression model
  - So sánh kết quả
- **MLP For Classification (Non-Linearly Separable data)**
  - Phân loại 3 class thuộc dạng non-linearly separable dựa trên tọa độ x và y
  - Train 2 model: sofmax regresion và MLP classification model
  - So sánh kết quả

Review MLP

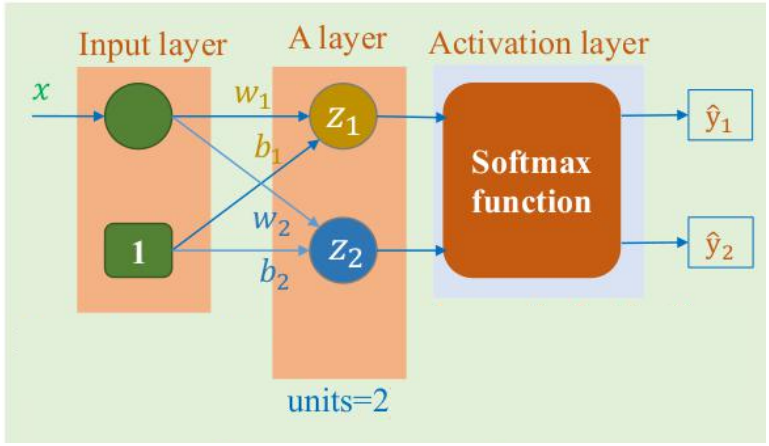
Linear Model



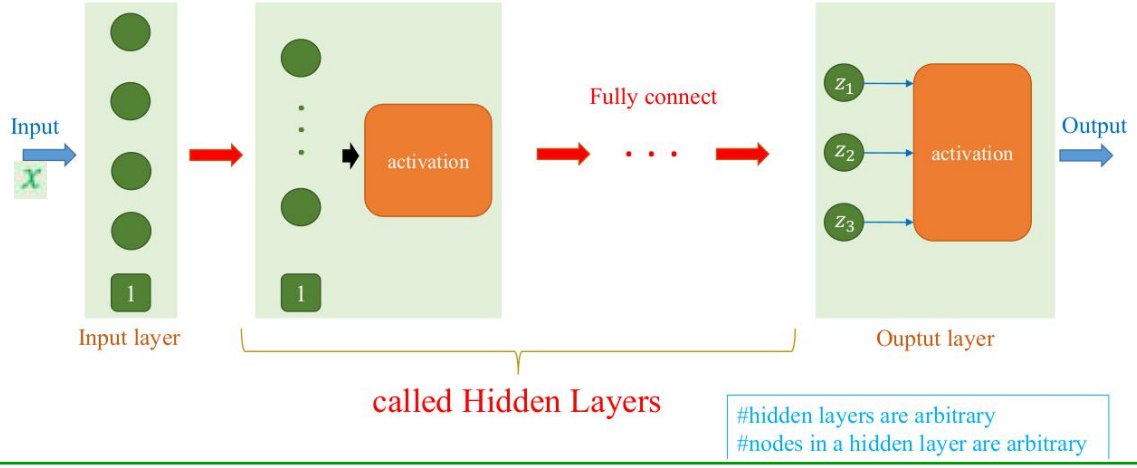
Logistic Regression Model



Softmax Regression Model



MLP Model



AI VIETNAM
All-in-One Course
(TA Session)

Review MLP

Linear Model

- Giải quyết bài toán về **regression**
- Sử dụng **linear function**
- Target có domain $y \in (-\infty, +\infty)$

Logistic Regression Model

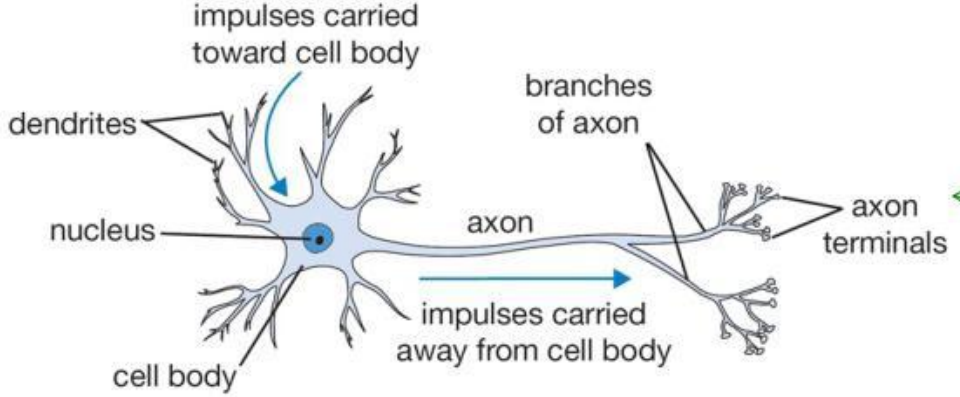
- Giải quyết bài toán về **binary classification**, hoặc **multi-label classification**
- Sử dụng **linear function + sigmoid function**
- Target có domain $y \in [0,1]$

Softmax Regression Model

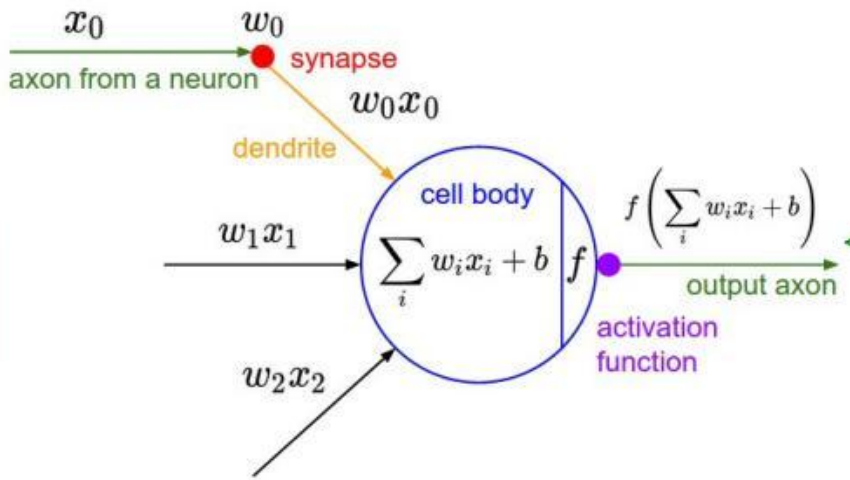
- Giải quyết bài toán về **multi-class classification**
- Sử dụng **linear function + softmax function**
- Target là discrete domain (thông thường được biểu diễn bằng one-hot encoding)

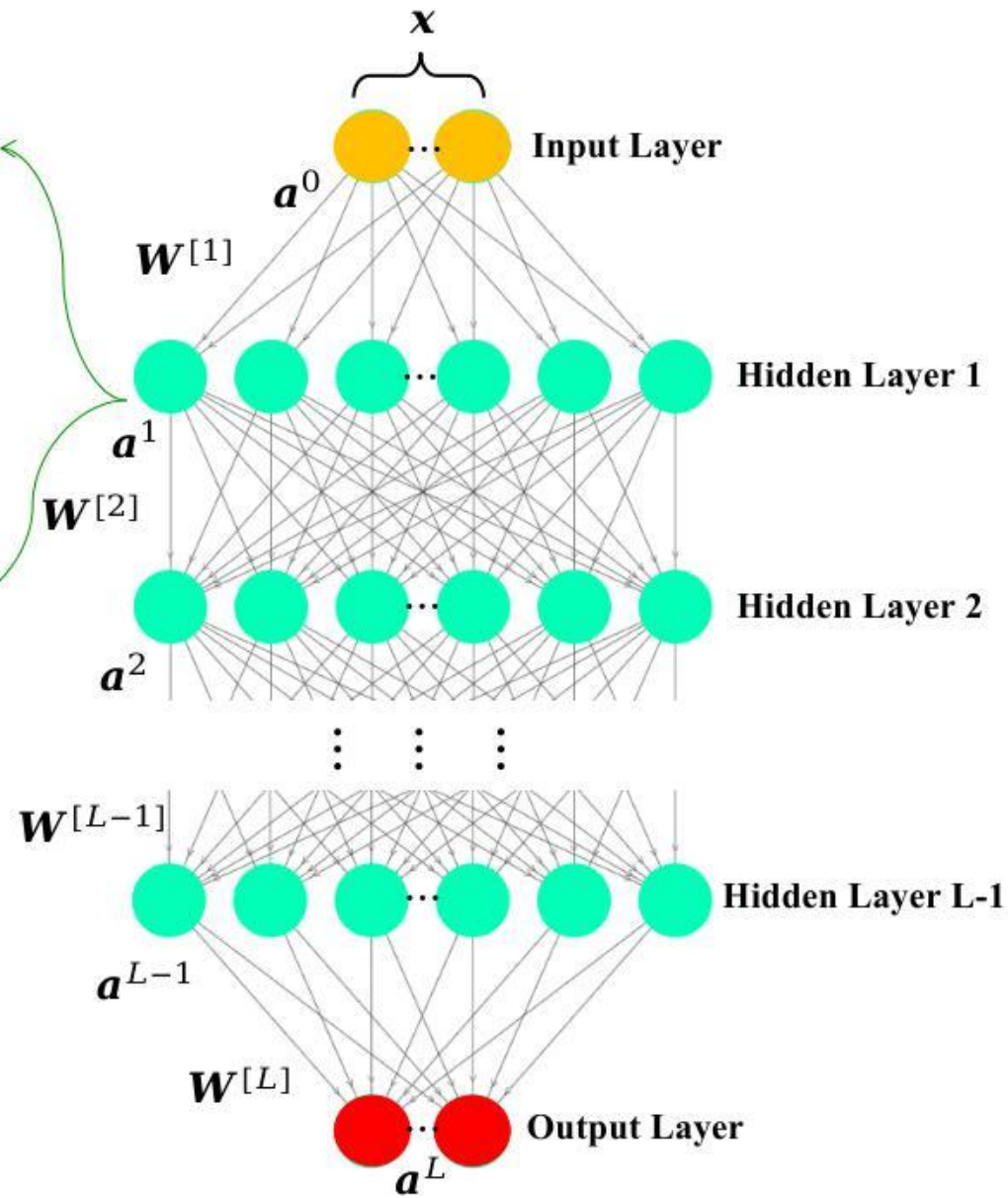
MLP Model

- Có thể giải quyết bài toán về **regression và classification**, ...
- Sử dụng các loại layers và các node (**linear function + activation functions**)
- Target có domain tùy vào bài toán



$$Loss = L(\hat{y}, y)$$





Update Weights

$$\boldsymbol{w}^{[l]} = \boldsymbol{w}^{[l]} - \eta \frac{\partial Loss}{\partial \boldsymbol{w}^{l}}$$

Forward

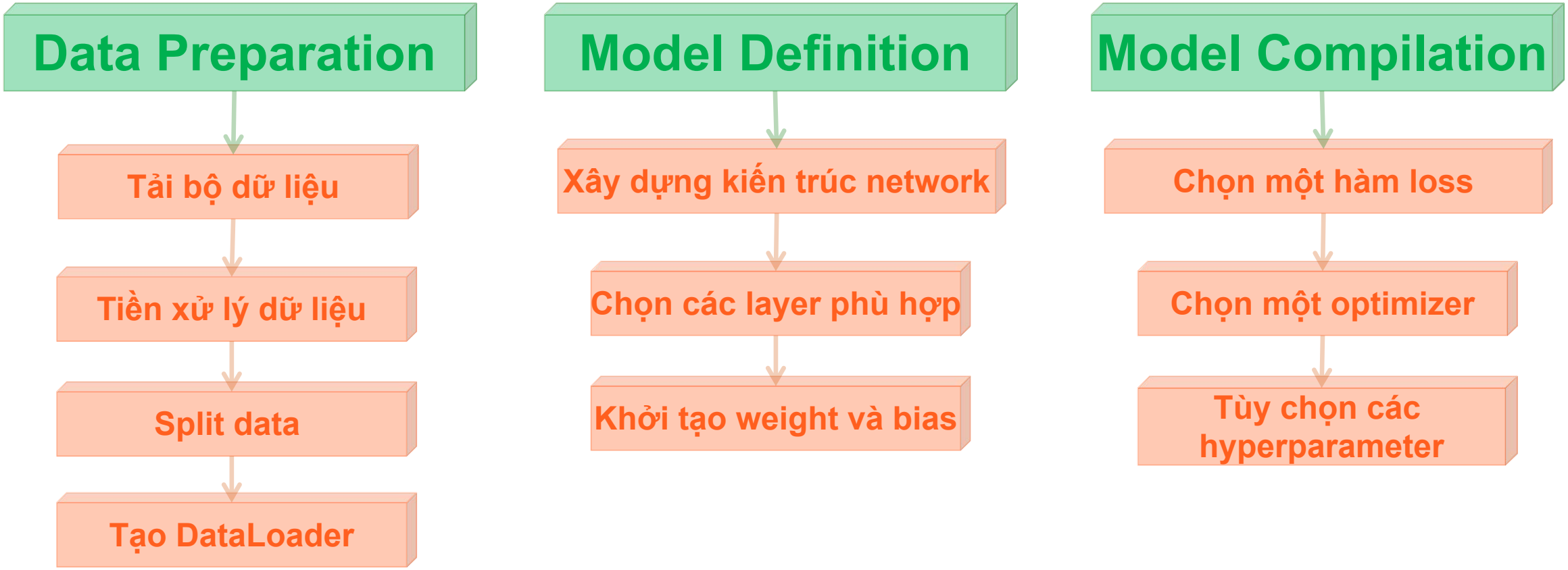
- $\boldsymbol{a}^0 = \boldsymbol{x}$
- $\boldsymbol{z}^{[l]} = \boldsymbol{W}^{[l]T} * \boldsymbol{a}^{[l-1]}$
- $\boldsymbol{a}^{[l]} = f(\boldsymbol{z}^{[l]})$
- $\hat{\boldsymbol{y}} = \boldsymbol{a}^{[L]} = f(\boldsymbol{z}^{[L]})$
  $= f(\boldsymbol{W}^{[L]T} * f(\boldsymbol{W}^{[L-1]T} *$
  $\ldots f(\boldsymbol{W}^{[2]T} * f(\boldsymbol{W}^{[1]T}\boldsymbol{x}))))$

Backpropagation

- $\frac{\partial Loss}{\partial \boldsymbol{w}^{[L]}} = \frac{\partial Loss}{\partial \boldsymbol{a}^{[L]}} * \frac{\partial \boldsymbol{a}^{[L]}}{\partial \boldsymbol{z}^{[L]}} * \frac{\partial \boldsymbol{z}^{[L]}}{\partial \boldsymbol{w}^{[L]}}$
- $\frac{\partial Loss}{\partial \boldsymbol{w}^{[1]}} = \frac{\partial Loss}{\partial \boldsymbol{a}^{[L]}} * \frac{\partial \boldsymbol{a}^{[L]}}{\partial \boldsymbol{z}^{[L]}} * \frac{\partial \boldsymbol{z}^{[L]}}{\partial \boldsymbol{a}^{[L-1]}} \cdots *$
  $\frac{\partial \boldsymbol{a}^{[2]}}{\partial \boldsymbol{z}^{[2]}} * \frac{\partial \boldsymbol{z}^{[2]}}{\partial \boldsymbol{a}^{[1]}} * \frac{\partial \boldsymbol{a}^{[1]}}{\partial \boldsymbol{z}^{[1]}} * \frac{\partial \boldsymbol{z}^{[1]}}{\partial \boldsymbol{w}^{[1]}}$

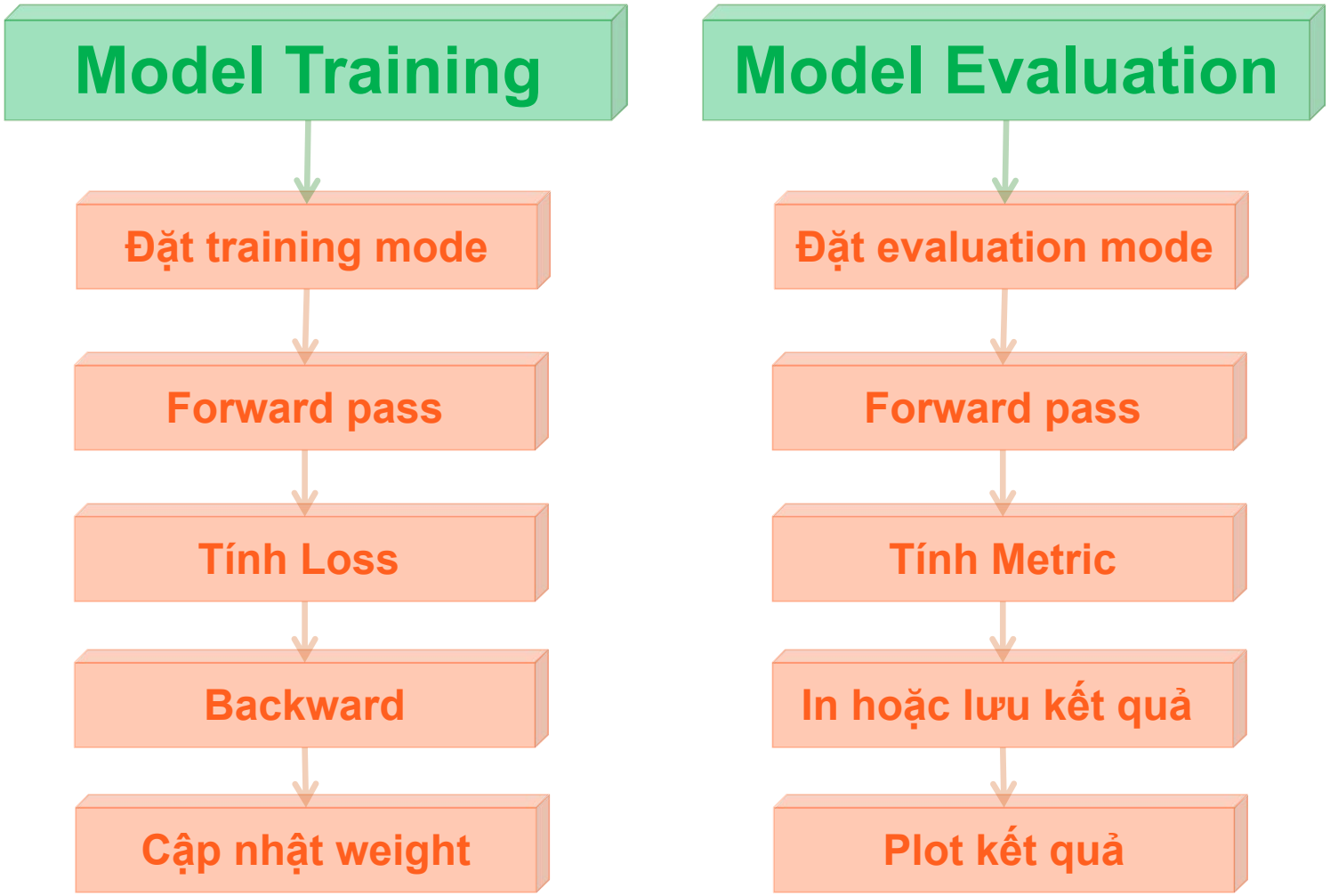
AI VIETNAM
All-in-One Course
(TA Session)

Review MLP



Các bước cơ bản trong việc build model và train khi dùng Pytorch

Review MLP



Các bước cơ bản trong việc build model và train khi dùng Pytorch

- Review MLP
- **MLP For Regression**
  - Dự đoán năng lượng tiêu thụ (MPG) của xe ô tô dựa trên 9 features (thông số kỹ thuật của xe)
  - Chuẩn bị data trước khi train
  - Train 2 model: linear regression và MLP regression model
  - So sánh kết quả
- **MLP For Classification (Non-Linearly Separable data)**
  - Phân loại 3 class thuộc dạng non-linearly separable dựa trên tọa độ x và y
  - Train 2 model: sofmax regresion và MLP classification model
  - So sánh kết quả

AI VIETNAM
All-in-One Course
(TA Session)

MLP For Regression

- **Yêu cầu đề bài**

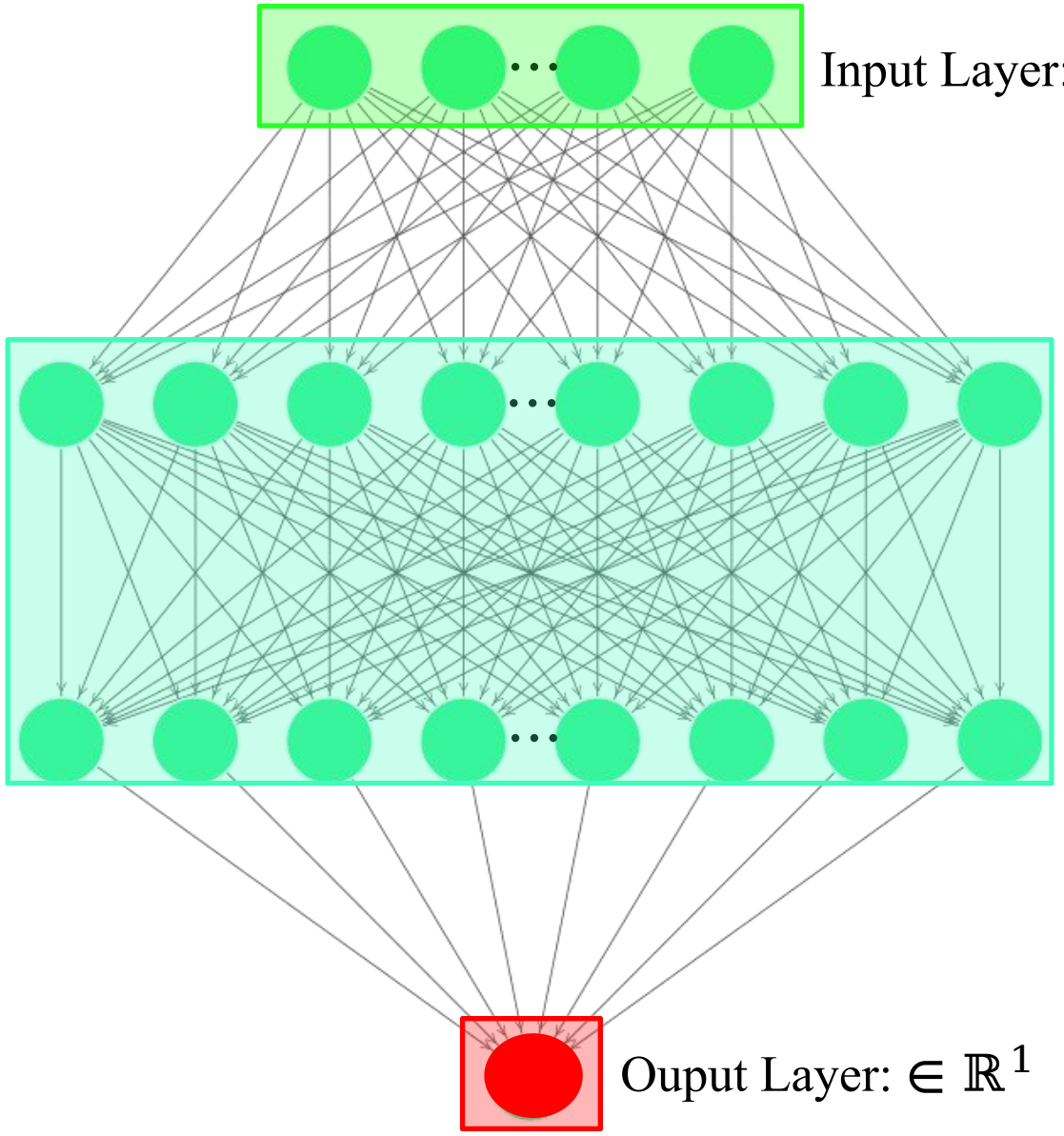
1. **MLP FOR REGRESSION:** Cho 1 tập data **Auto_MPG_data.csv** thống kê việc số lượng nhiên liệu được tiêu thụ (MPG) dựa trên 9 features của xe ô tô từ thập niên 70s đến thập niên 80s. Các bạn hãy xử lý data và xây dựng model sau để dự đoán nhiên liệu tiêu thụ của xe ô tô:

(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

- Model1: Linear regression
  Loss: MSE
  Epochs: 100
  Optimizer: SGD
  LR: 0.1
- Model2: MLP regression
  Hidden layers: 2
  Nodes: 64, 64
  Activation: relu
  Loss: MSE
  Epochs: 100
  Optimizer: SGD
  LR: 0.003



Input Layer: $\in \mathbb{R}^9$

Hidden Layer1: $\in \mathbb{R}^{64}$

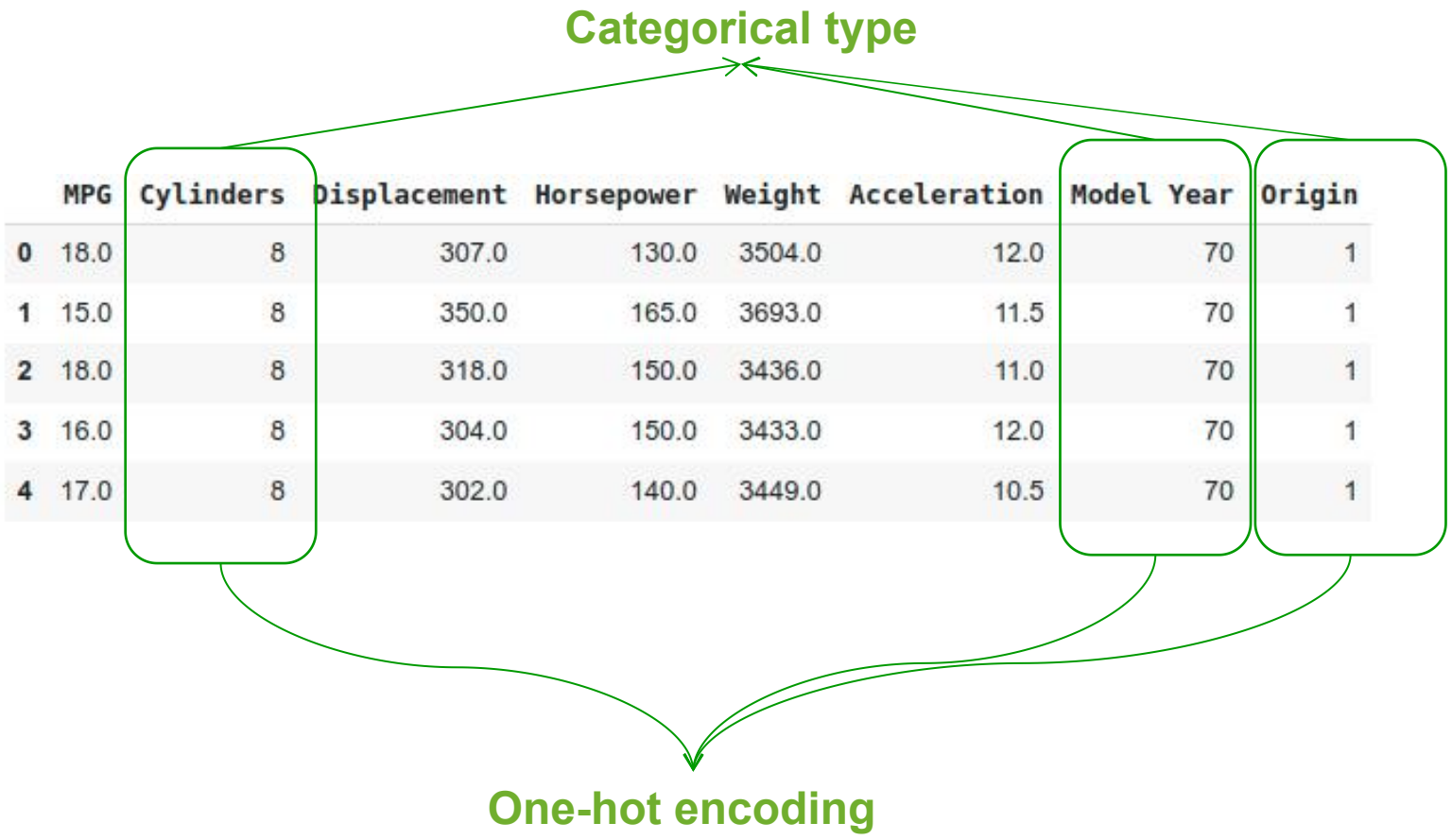
Hidden Layer2: $\in \mathbb{R}^{64}$

Ouput Layer: $\in \mathbb{R}^1$

```
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()
)
```

MLP For Regression

- **Chuẩn bị data trước khi train**



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

MLP For Regression

- **Chuẩn bị data trước khi train**

Categorical type

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

One-hot encoding

	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

MLP For Regression

- **Chuẩn bị data trước khi train**

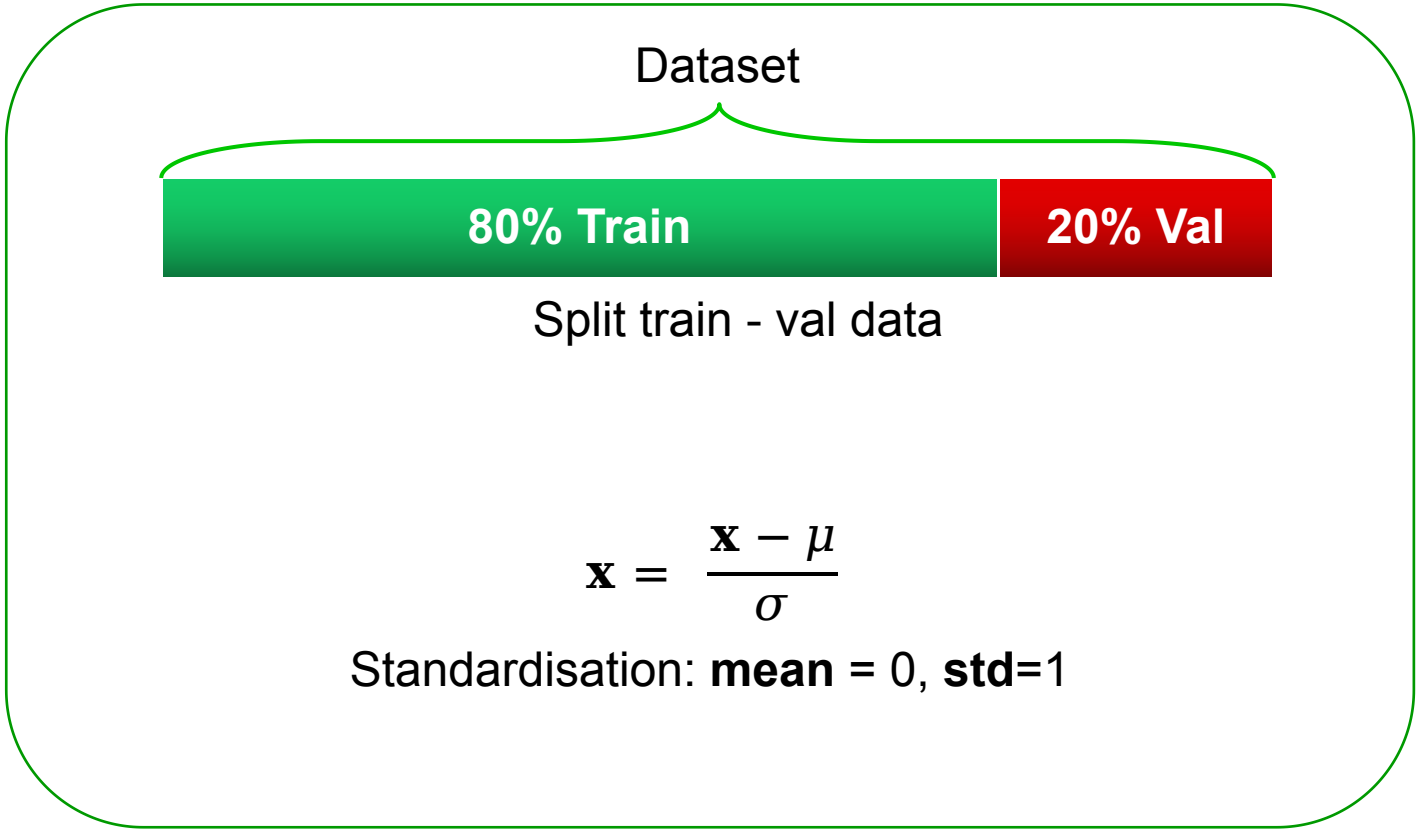
	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	Europe	Japan	USA
0	18.0	8	307.0	130.0	3504.0	12.0	70	0	0	1
1	15.0	8	350.0	165.0	3693.0	11.5	70	0	0	1
2	18.0	8	318.0	150.0	3436.0	11.0	70	0	0	1
3	16.0	8	304.0	150.0	3433.0	12.0	70	0	0	1
4	17.0	8	302.0	140.0	3449.0	10.5	70	0	0	1

Target

Features

MLP For Regression

- **Chuẩn bị data trước khi train**



MLP For Regression

- **Chuẩn bị data trước khi train**

```
1 dataset = pd.read_csv("/content/Auto_MPG_data.csv")
2 dataset.head()
```

	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

MLP For Regression

- **Chuẩn bị data trước khi train**

Syntax: pd.read_csv*(filepath_or_buffer, sep=' ,' , header='infer',*
index_col=None, usecols=None, engine=None, skiprows=None,
nrows=None)

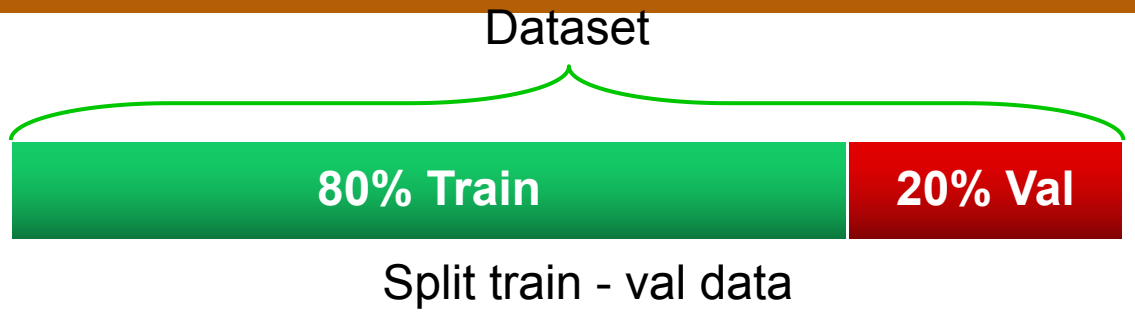
Parameters:

- ***filepath_or_buffer****: Location of the csv file. It accepts any string path or URL of the file.*
- ***sep****: It stands for separator, default is ', '.*
- ***header****: It accepts int, a list of int, row numbers to use as the column names, and the start of the data. If no names are passed, i.e., header=None, then, it will display the first column as 0, the second as 1, and so on.*
- ***usecols****: Retrieves only selected columns from the CSV file.*
- ***nrows****: Number of rows to be displayed from the dataset.*
- ***index_col****: If None, there are no index numbers displayed along with records.*
- ***skiprows****: Skips passed rows in the new data frame.*

https://www.geeksforgeeks.org/python-read-csv-using-pandas-read_csv/

MLP For Regression

- **Chuẩn bị data trước khi train**




```
[ ]   1 train_dataset = dataset.sample(frac=0.8, random_state=0)
      2 val_dataset = dataset.drop(train_dataset.index)
```

```
[ ]   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)
```

MLP For Regression

- **Chuẩn bị data trước khi train**

Syntax:

DataFrame.sample(n=None, frac=None, replace=False, weights=None, random_state=None, axis=None)

Parameters:

n: *int value, Number of random rows to generate.*
frac: *Float value, Returns (float value * length of data frame values). frac cannot be used with n.*
replace: *Boolean value, return sample with replacement if True.*
random_state: *int value or numpy.random.RandomState, optional. if set to a particular integer, will return same rows as sample in every iteration.*
axis: *0 or 'row' for Rows and 1 or 'column' for Columns.*

https://www.geeksforgeeks.org/python-pandas-dataframe-sample/

MLP For Regression

- **Chuẩn bị data trước khi train**

$$\mathbf{x} = \frac{\mathbf{x} - \mu}{\sigma} = \frac{\mathbf{x} - \frac{1}{N}\sum_{i=1}^{N} x_i}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}}$$

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

Standardisation (Z-score):
- Là một kỹ thuật Feature Scaling đưa các giá trị của feature theo distribution có mean = 0 và standard deviation = 1.
- Giúp đưa variance của các feature về gần bằng nhau:
  + Giá trị < 0: giá trị dưới mean
  + Giá trị > 0: giá trị trên mean
  + Giá trị = 0: giá trị chính là mean
- Phù hợp với các feature có ít outliers hoặc outliers không quá lớn để clipping

- $\hat{y} = \boldsymbol{w}^T \boldsymbol{x}$
- $\mathrm{SE} = (\hat{y} - \mathrm{y})^2$
- $\boldsymbol{w} = \boldsymbol{w} - \eta * \frac{\partial \mathrm{SE}}{\partial \boldsymbol{w}}$

	Student	CGPA	Salary '000
0	1	3.0	60
1	2	3.0	40
2	3	4.0	40

	Student	CGPA	Salary '000
0	1	-1.184341	1.520013
1	2	-1.184341	-1.100699
2	3	0.416120	-1.100699

- $\hat{y} = 3w_1 + 60w_2 + w_0$
- $\mathrm{SE} = (3w_1 + 60w_2 + w_0 - \mathrm{y})^2$
- $\frac{\partial \mathrm{SE}}{\partial \boldsymbol{w}} = 2 * (3w_1 + 60w_2 + w_0 - \mathrm{y})\ldots$
- $\boldsymbol{w} = \boldsymbol{w} - \eta * \frac{\partial \mathrm{SE}}{\partial \boldsymbol{w}}$

- $\hat{y} = -1.18w_1 + 1.5w_2 + w_0$
- $\mathrm{SE} = (-1.18w_1 + 1.52w_2 + w_0 - \mathrm{y})^2$
- $\frac{\partial \mathrm{SE}}{\partial \boldsymbol{w}} = 2 * (-1.18w_1 + 1.52w_2 + w_0 - \mathrm{y})\ldots$
- $\boldsymbol{w} = \boldsymbol{w} - \eta * \frac{\partial \mathrm{SE}}{\partial \boldsymbol{w}}$

MLP For Regression

- **Chuẩn bị data trước khi train**

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

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
```

MLP For Regression

• Metric for regression

- Là việc tính toán error score để kết luận khả năng dự đoán của model trên một tập data
- Vì là bài toán regression nên không thể dùng accuracy như bài toán clasification mà thay vào đó là kết quả dự đoán gần với kết quả thật là bao nhiêu
- Có 3 metric phổ biến: MSE, MAE, và R-squared

$$\text{MAE} = \frac{1}{N}\sum|\hat{y} - y|$$

$$\text{MSE} = \frac{1}{N}\sum(\hat{y} - y)^2$$

$$\text{R-squared} = 1 - \frac{RSS}{TSS}$$

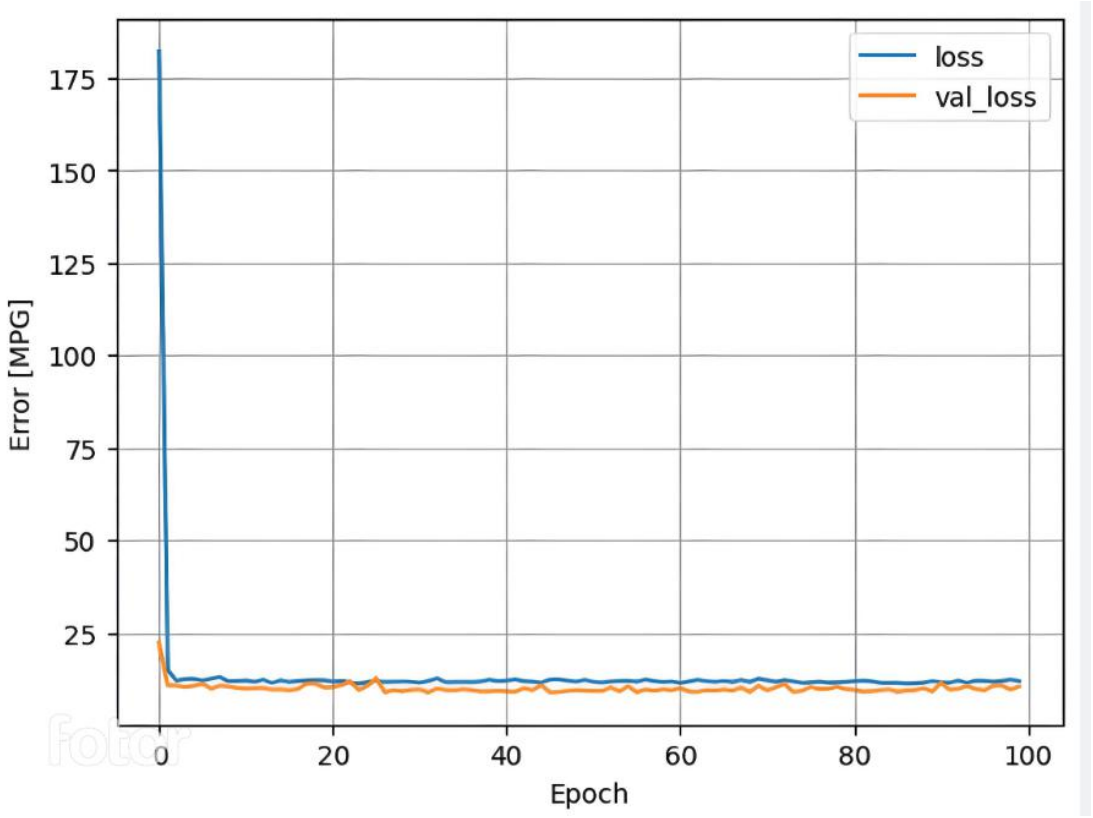
$$\text{RSS} = \sum(\hat{y} - y)^2$$

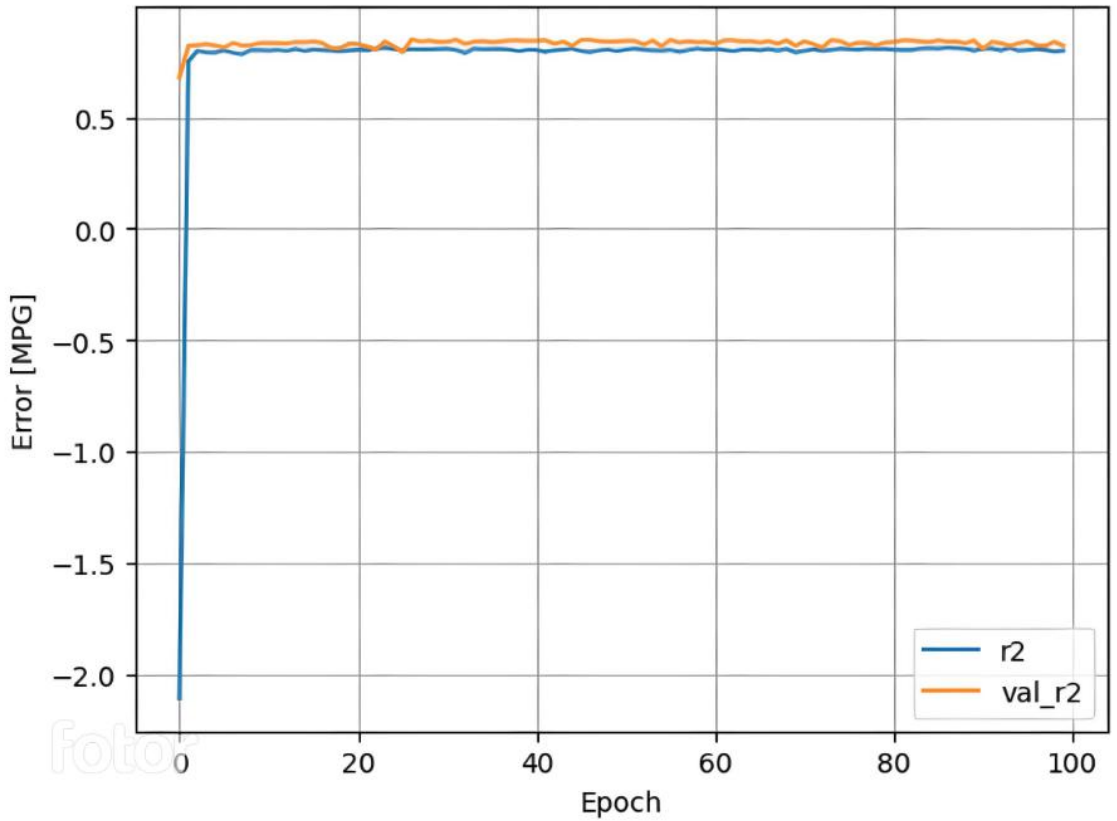
$$\text{TSS} = \sum(y - \overline{y})^2$$

- **MAE:** Đo trị tuyệt đối sai lệch giữa kết quả dự đoán và giá trị thật. Ít bị ảnh hưởng bởi outlier hơn MSE. Nhưng có thể không phản ánh được hết tác động của outliers
- **MSE:** Đo bình phương độ sai lệch giữa kết quả dự đoán và giá trị thật. Độ sai lệch càng lớn khi outliers càng lớn nhưng nếu quá lớn thì sẽ chỉ chịu sự tác động của outliers
- **R-squared:** Có range [0,1], giá trị càng lớn thì thể hiện performance của model càng tốt (ko giống với MAE và MSE càng bé càng tốt). Hoạt động theo giả thuyết là ban đầu dự đoán kết quả bằng trung bình của data (R-squared = 0, RSS=TSS) sau đó nhờ huấn luyện kết quả dự đoán tốt hơn (R-squared > 0, RSS<TSS)

MLP For Regression

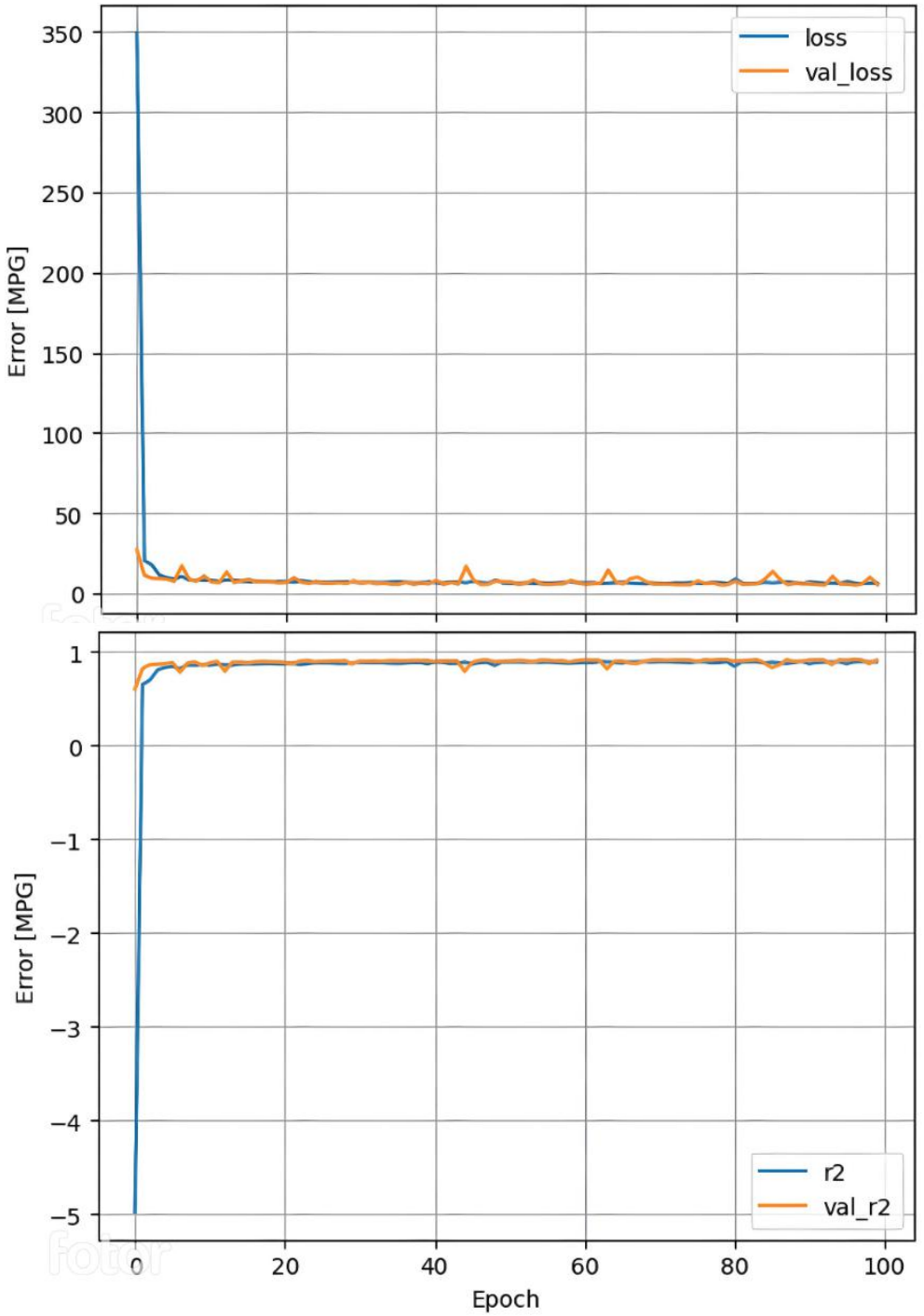
- **Model1: Linear regression**
  - **Loss**: MSE
  - **Epochs**: 100
  - **Optimizer**: SGD
  - **LR**: 0.1





MLP

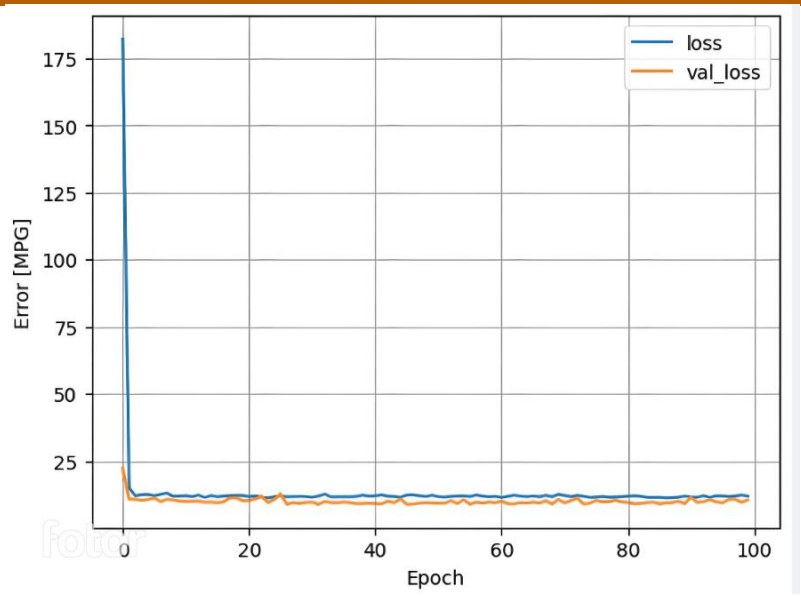
- **Model2: MLP regression**
  - **Hidden layers**: 2
  - **Nodes**: 64, 64
  - **Activation**: relu
  - **Loss**: MSE
  - **Epochs**: 100
  - **Optimizer**: SGD
  - **LR**: 0.003

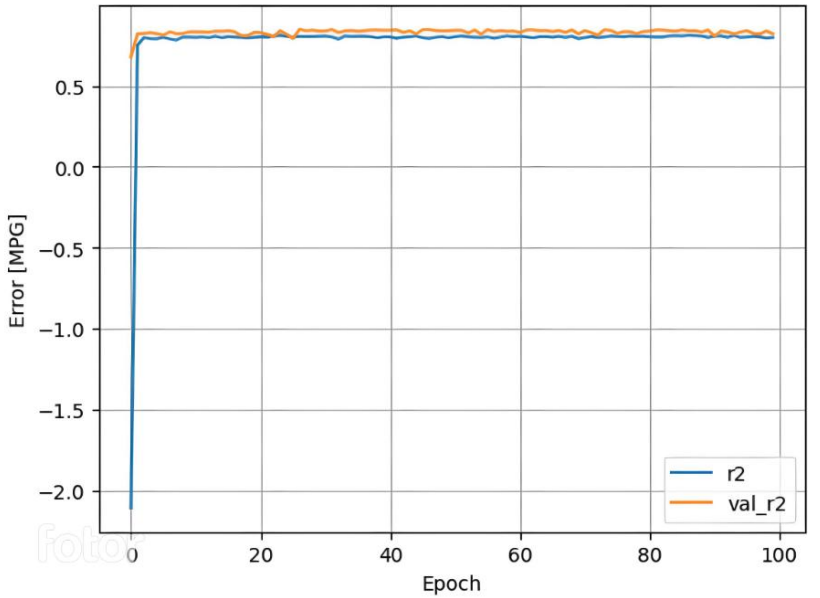


MLP For Regression

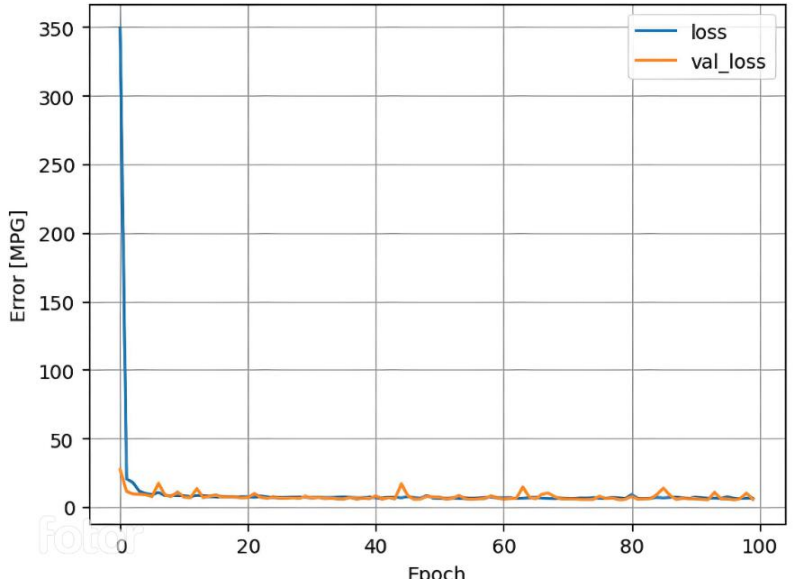
- **So sánh kết quả**

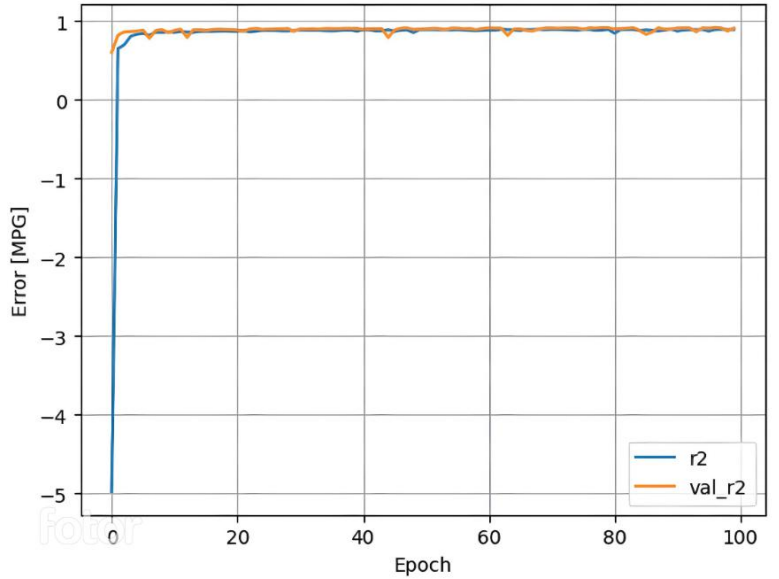
- **Model1: Linear regression**
  - **Loss**: MSE
  - **Epochs**: 100
  - **Optimizer**: SGD
  - **LR**: 0.1





- **Model2: MLP regression**
  - **Hidden layers**: 2
  - **Nodes**: 64, 64
  - **Activation**: relu
  - **Loss**: MSE
  - **Epochs**: 100
  - **Optimizer**: SGD
  - **LR**: 0.003



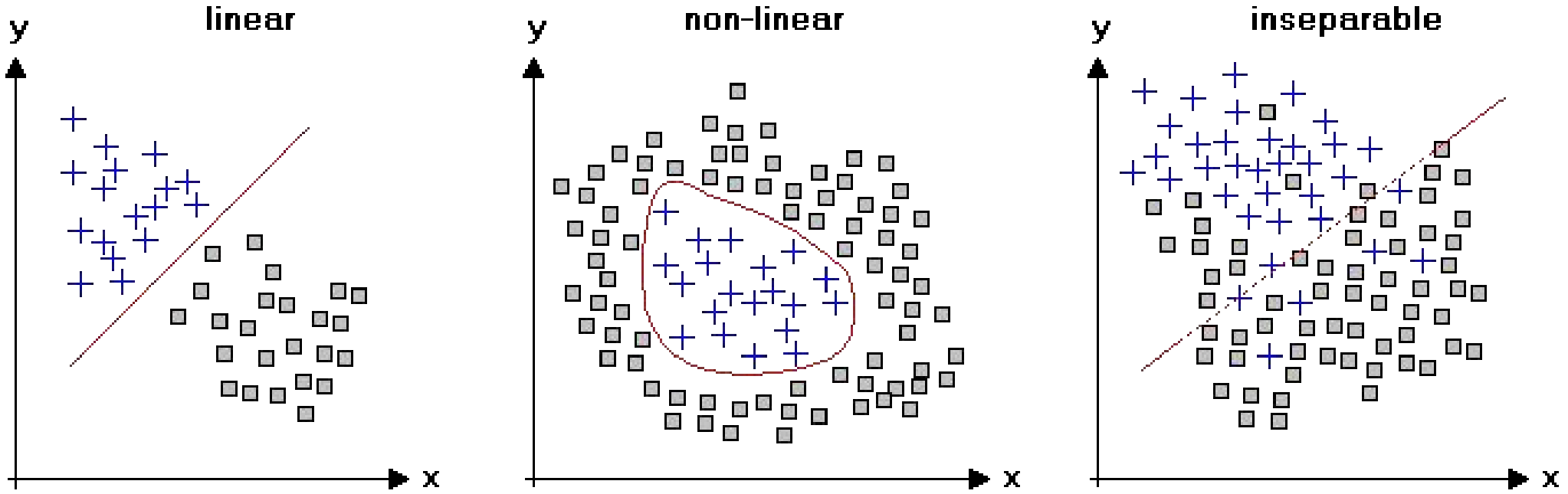


AI VIETNAM
All-in-One Course
(TA Session)

- Review MLP
- MLP For Regression
  - Dự đoán năng lượng tiêu thụ (MPG) của xe ô tô dựa trên 9 features (thông số kỹ thuật của xe)
  - Chuẩn bị data trước khi train
  - Train 2 model: linear regression và MLP regression model
  - So sánh kết quả
- **MLP For Classification (Non-Linearly Separable data)**
  - Phân loại 3 class thuộc dạng non-linearly separable dựa trên tọa độ x và y
  - Train 2 model: sofmax regresion và MLP classification model
  - So sánh kết quả

MLP For Classification (Non-Linearly Separable data)

- **Trong bài toán phân loại thường có 3 trường hợp về data**

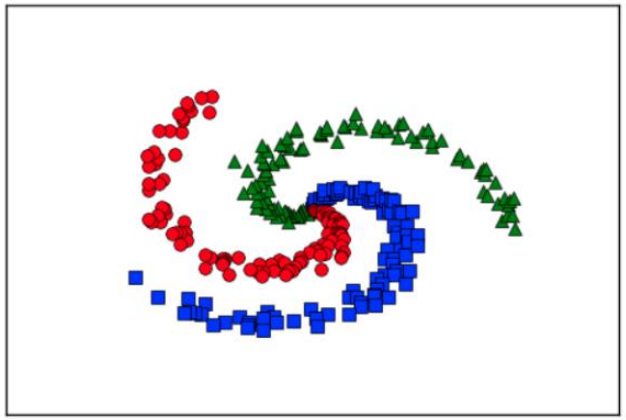


http://www.statistics4u.com/fundstat_eng/cc_data_structure.html

MLP For Classification (Non-Linearly Separable data)

- **Yêu cầu đề bài**

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:

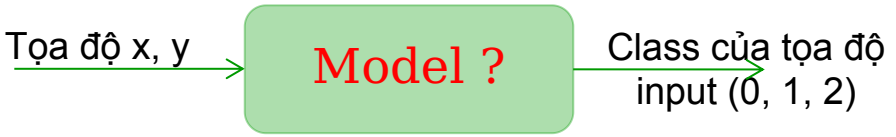


(a) Xây dựng softmax regression model: Loss= Cross Entropy, Metric = Accuracy, Epochs=200, Batch size=32, Optimizer=SGD, Learning rate = 0.1

(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

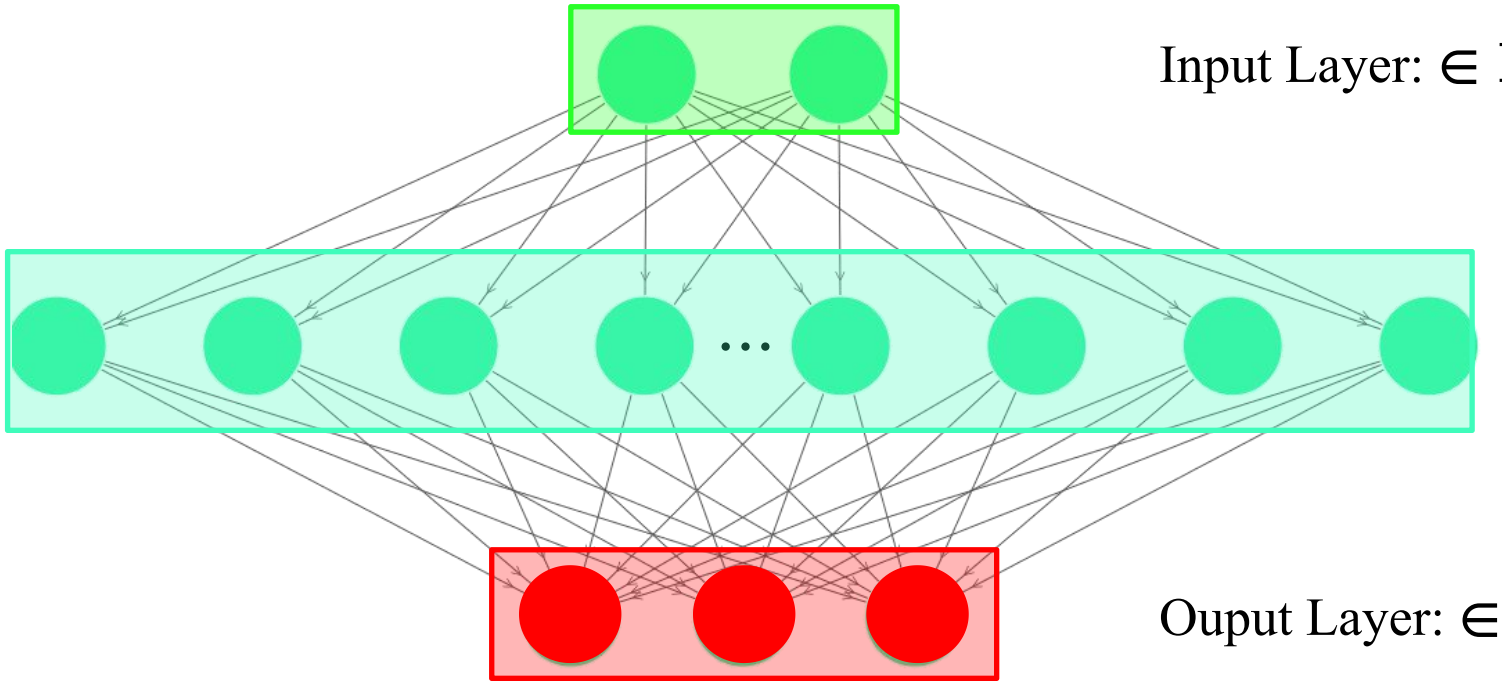
(c) So sánh và nhận xét kết quả giữa 2 model

https://machinelearningcoban.com/2017/02/24/mlp/



- Model1: Softmax regression
  Loss: CE
  Epochs: 500
  Optimizer: SGD
  Learning rate: 0.1

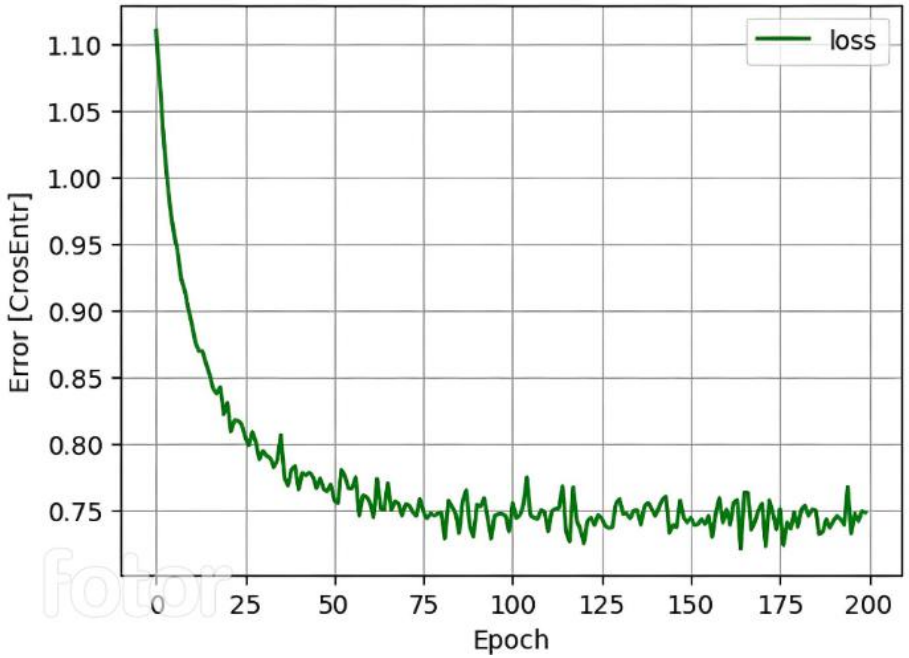
- Model2: MLP Classification
  Hidden layers: 1
  Nodes: 128
  Activation: relu
  Loss: CE
  Epochs: 500
  Optimizer: SGD
  LR: 0.1

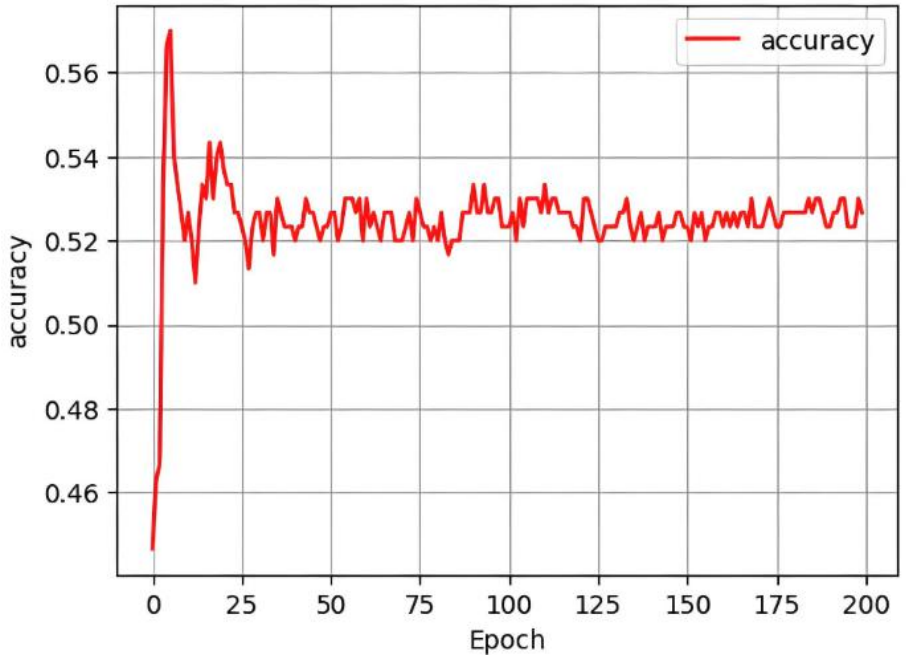



```
MLP(
  (linear1): Linear(in_features=2, out_features=128, bias=True)
  (output): Linear(in_features=128, out_features=3, bias=True)
  (relu): ReLU()
)
```

MLP For Classification (Non-Linearly Separable data)

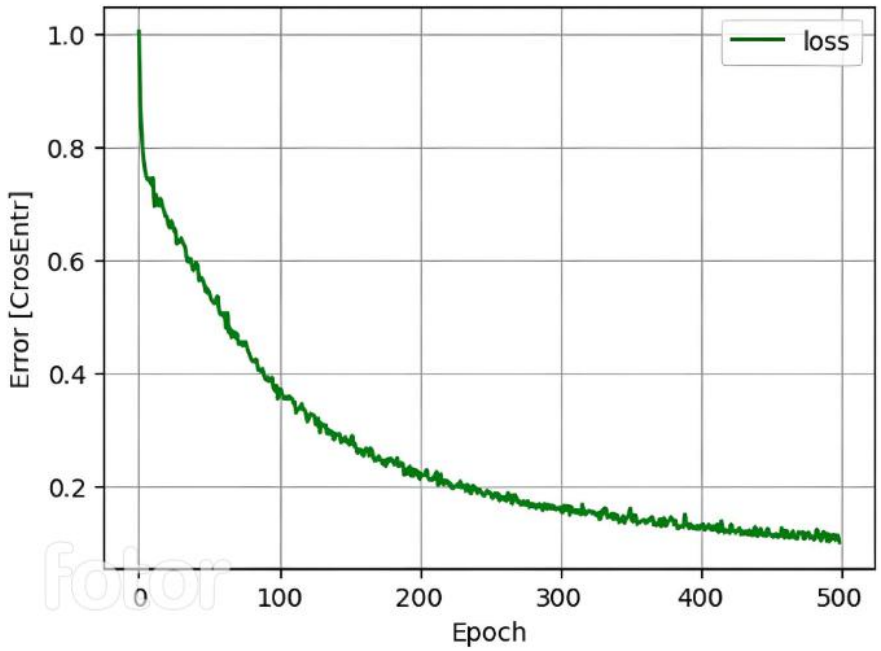
- **Model1: Softmax regression**
  Loss: CE
  Epochs: 500
  Optimizer: SGD
  Learning rate: 0.1

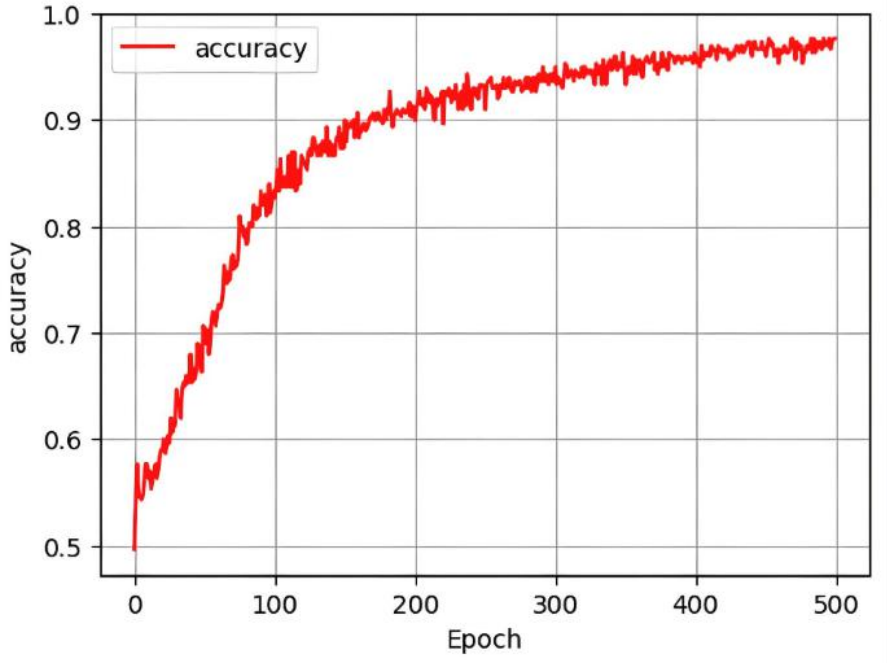




MLP For Classification (Non-Linearly Separable data)

- **Model2: MLP Classification**
  - **Hidden layers**: 1
  - **Nodes**: 128
  - **Activation**: relu
  - **Loss**: CE
  - **Epochs**: 500
  - **Optimizer**: SGD
  - **LR**: 0.1

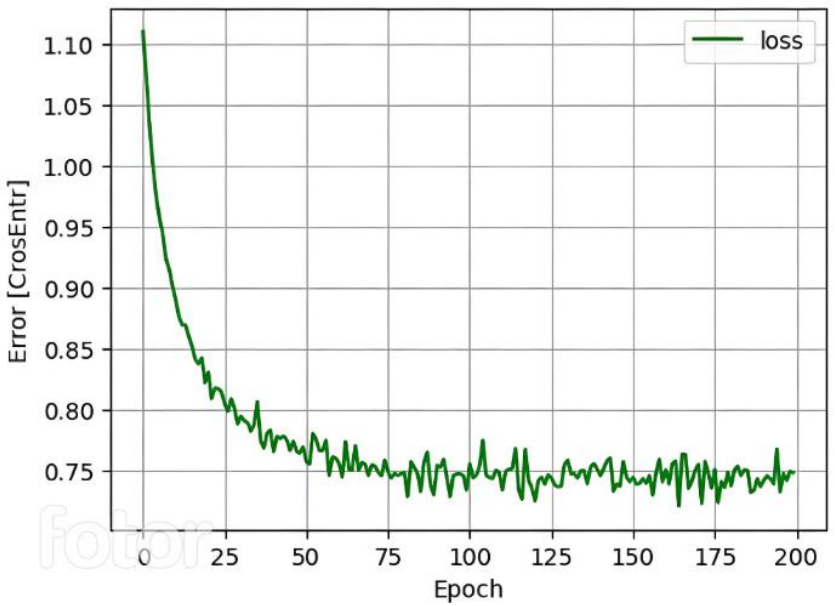


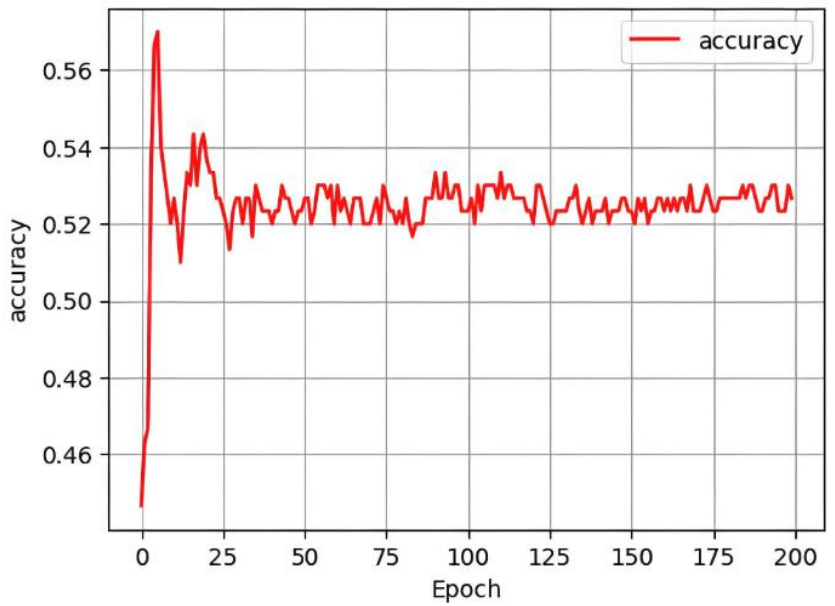


MLP For Classification (Non-Linearly Separable data)

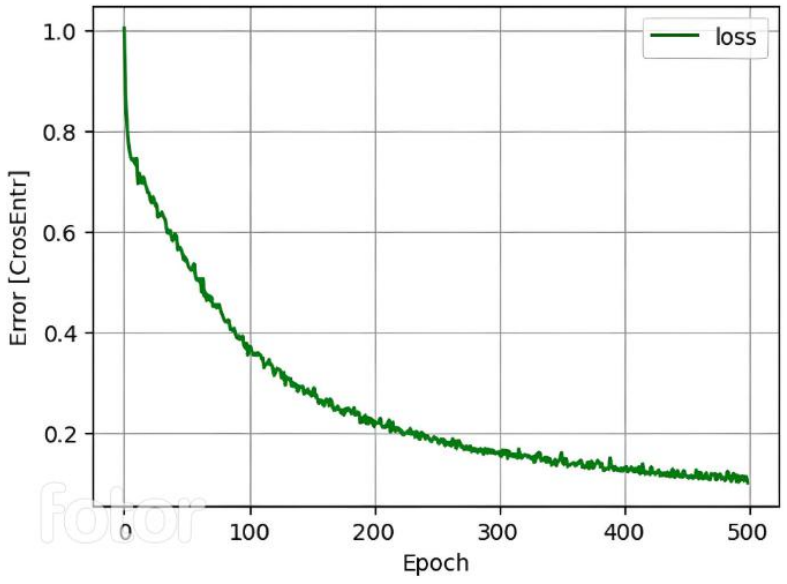
- **So sánh kết quả**

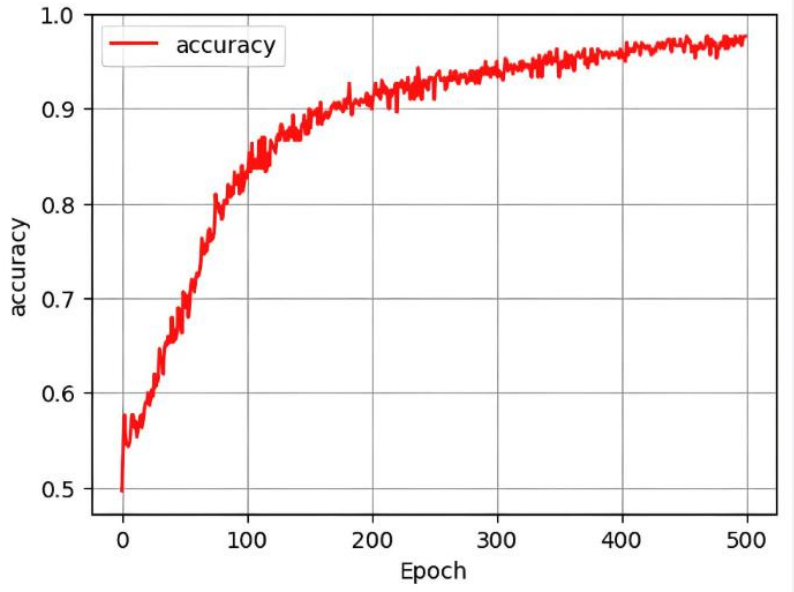
- **Model1: Softmax regression**
  Loss: CE
  Epochs: 500
  Optimizer: SGD
  Learning rate: 0.1





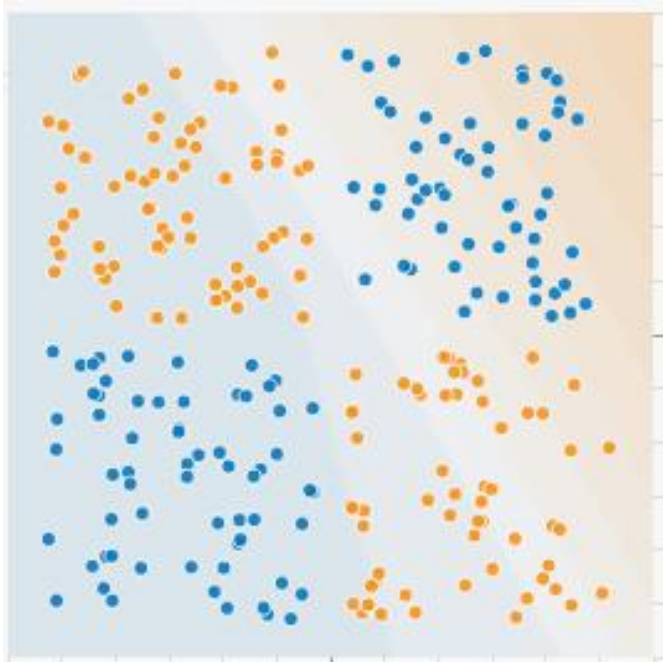
- **Model2: MLP Classification**
  Hidden layers: 1
  Nodes: 128
  Activation: relu
  Loss: CE
  Epochs: 500
  Optimizer: SGD
  LR: 0.1

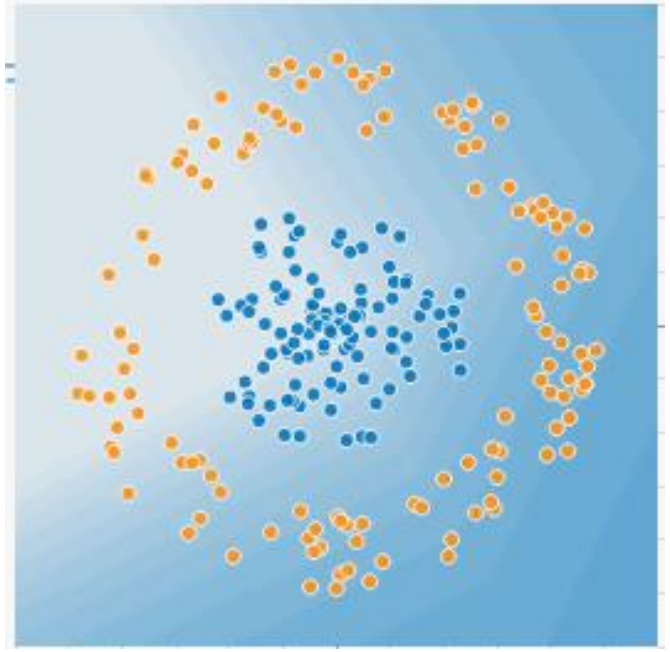




MLP For Classification (Non-Linearly Separable data)

- **Other Non-linearly Separable data**
  - **Test capacity of MLP**





https://playground.tensorflow.org/

Content

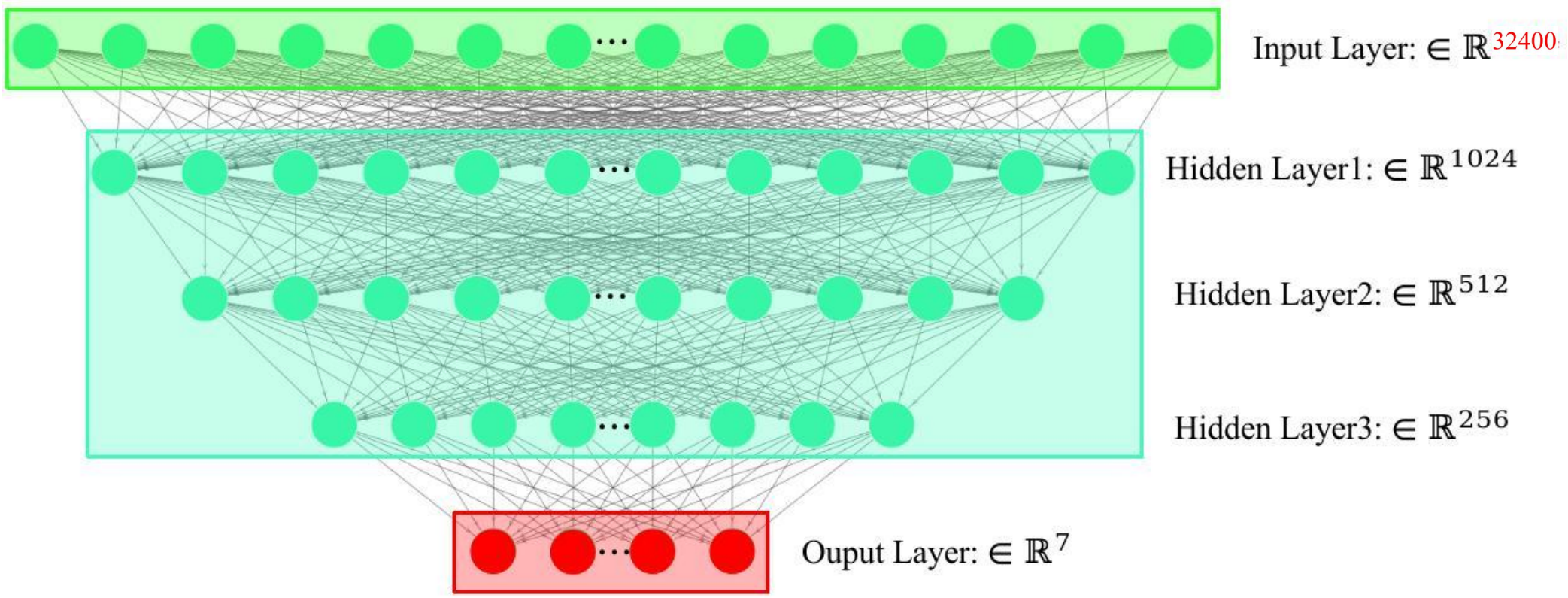
- **MLP For Classification (Image data)**
  - Phân loại loại cảm xúc của một người dựa trên ảnh đầu vào là gương mặt thể hiện cảm xúc của người đó
  - Chuẩn bị data trước khi train
    - High-level Keras preprocessing utilities
    - Input pipeline from scratch dùng tf.data
  - Train 4 models:
    - Softmax regression (không normalize)
    - Softmax regression (normalize [-1, 1])
    - MLP - tanh (normalize [-1, 1])
    - MLP - relu (normalize [-1, 1])
  - So sánh kết quả
  - Giới thiệu cơ bản về Sigmoid, Tanh, Relu

AI VIETNAM
All-in-One Course
(TA Session)

MLP For Classification (Image data)

- **Yêu cầu đề bài**

• Model1: Softmax regression **Epochs**: 100 **Learning rate**: 0.006 **Normalize data**: None	• Model2: Softmax regression **Epochs**: 100 **Learning rate**: 0.006 **Normalize data**: $x = \frac{x}{127.5} - 1$
• Model3: MLP Classification **Hidden layers**: 3 **Nodes**: 1024, 512, 256 **Activation**: tanh **Epochs**: 100 **Learning rate**: 0.006 **Normalize data**: $x = \frac{x}{127.5} - 1$	• Model4: MLP Classification **Hidden layers**: 3 **Nodes**: 1024, 512, 256 **Activation**: relu **Epochs**: 100 **Learning rate**: 0.006 **Normalize data**: $x = \frac{x}{127.5} - 1$




```
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()
)
```

MLP For Classification (Image data)

- **Chuẩn bị data trước khi train**

Directory Tree Structure

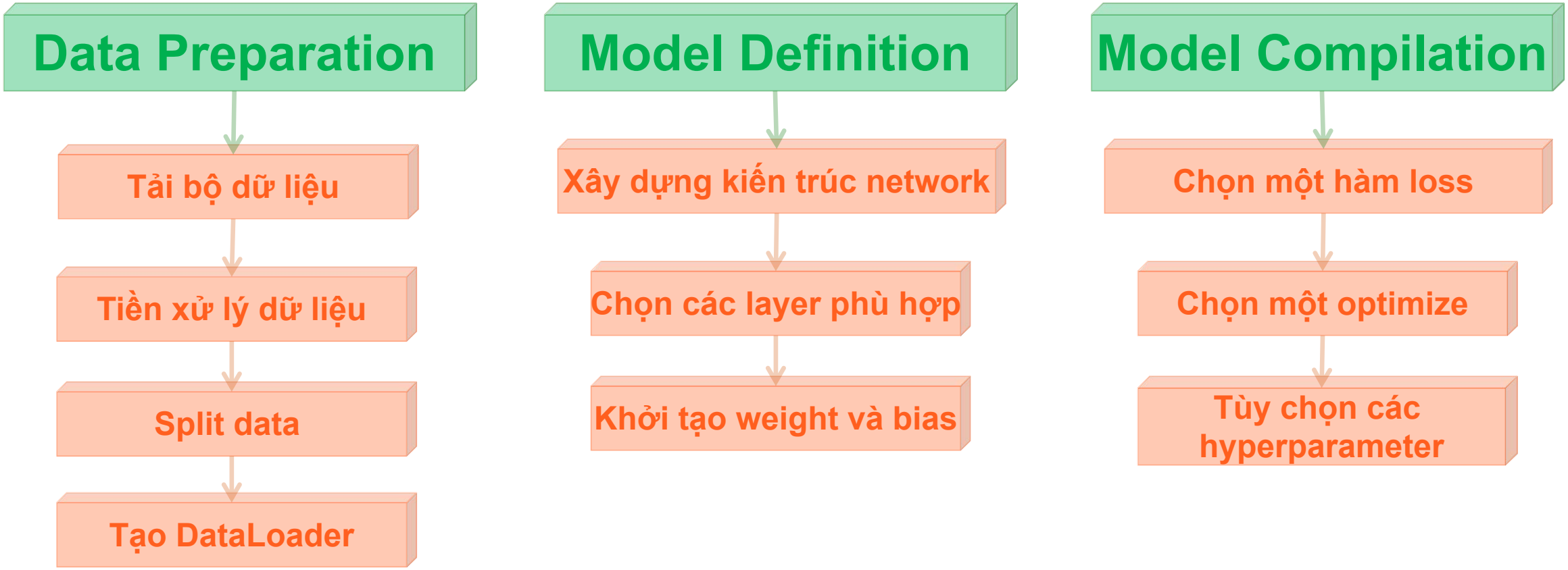

```python
class ImageDataset(Dataset):
    def __init__(self, img_dir, norm):
        self.resize = Resize((img_height, img_width))
        self.norm = norm
        self.img_dir = img_dir
        self.classes = os.listdir(img_dir)
        self.image_files = [(os.path.join(cls, img), cls)
                            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
```

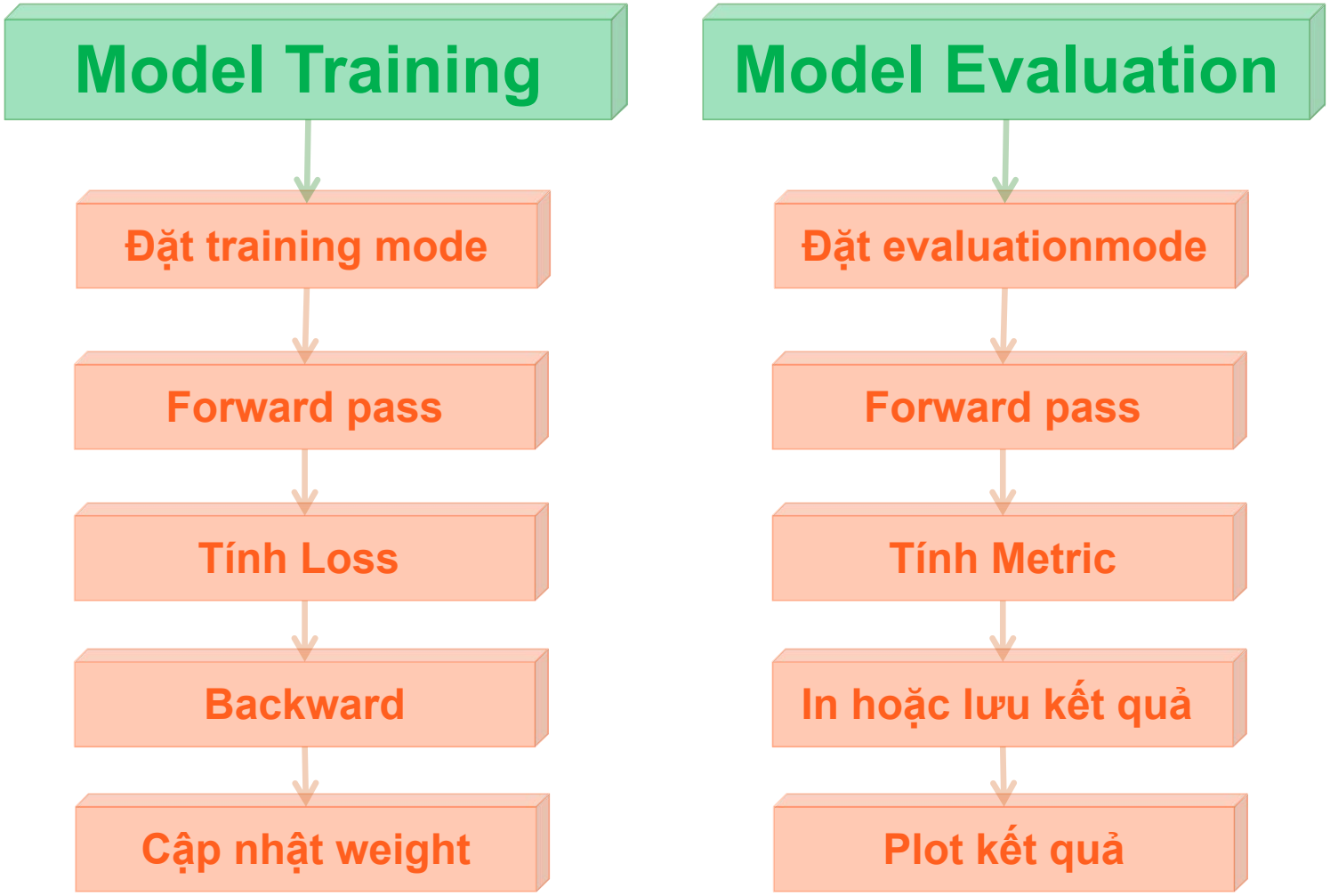
AI VIETNAM
All-in-One Course
(TA Session)

MLP For Classification (Image data)



Các bước cơ bản trong việc build model và train khi dùng Pytroch

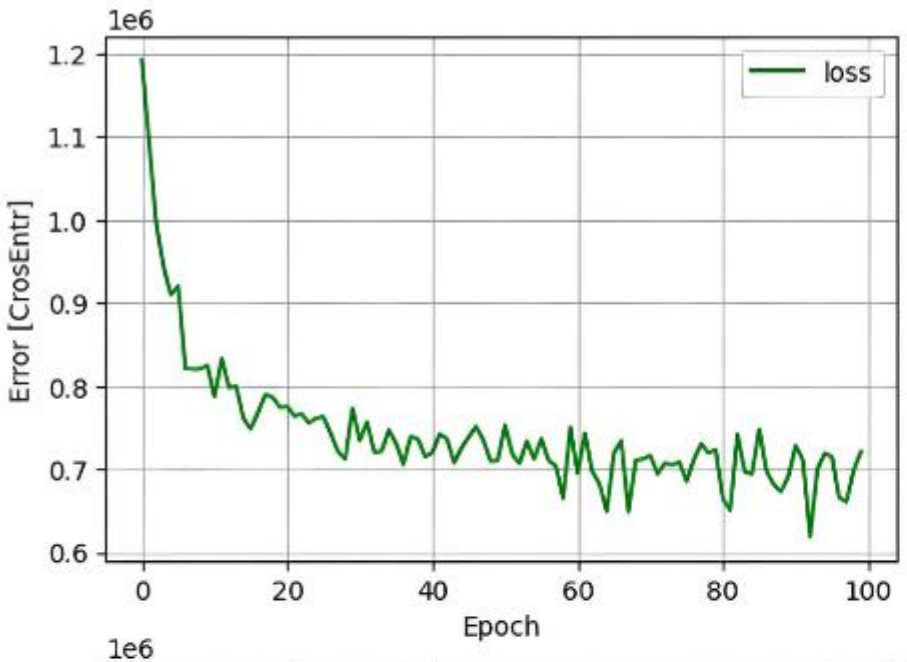
MLP For Classification (Image data)

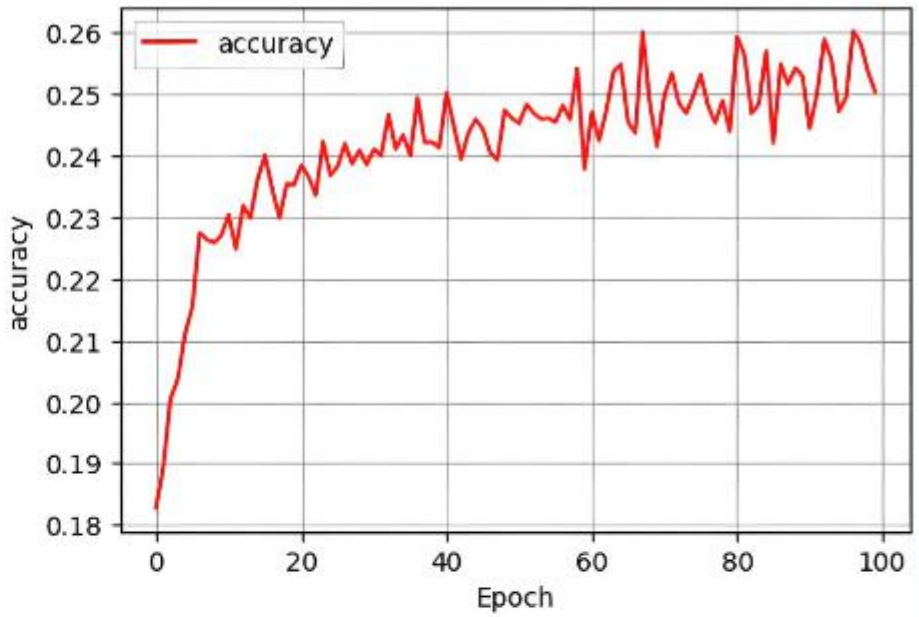


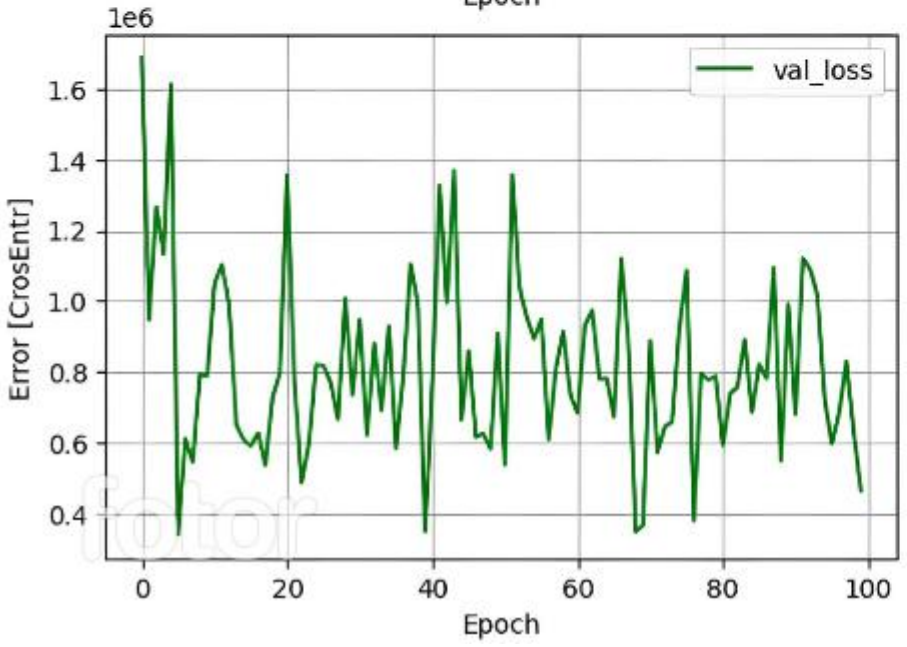
Các bước cơ bản trong việc build model và train khi dùng Pytroch

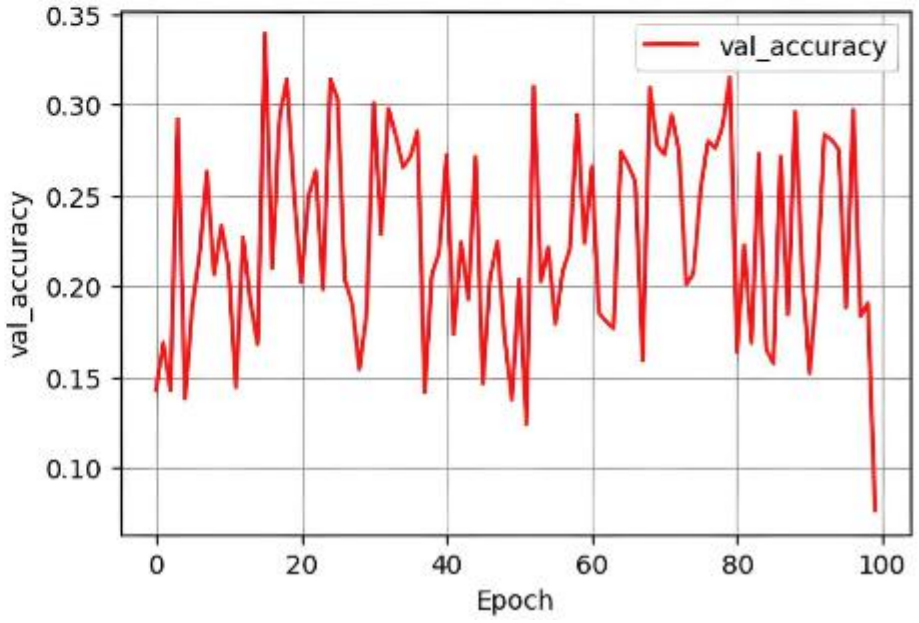
MLP For Classification (Image data)

Model1: Softmax regression



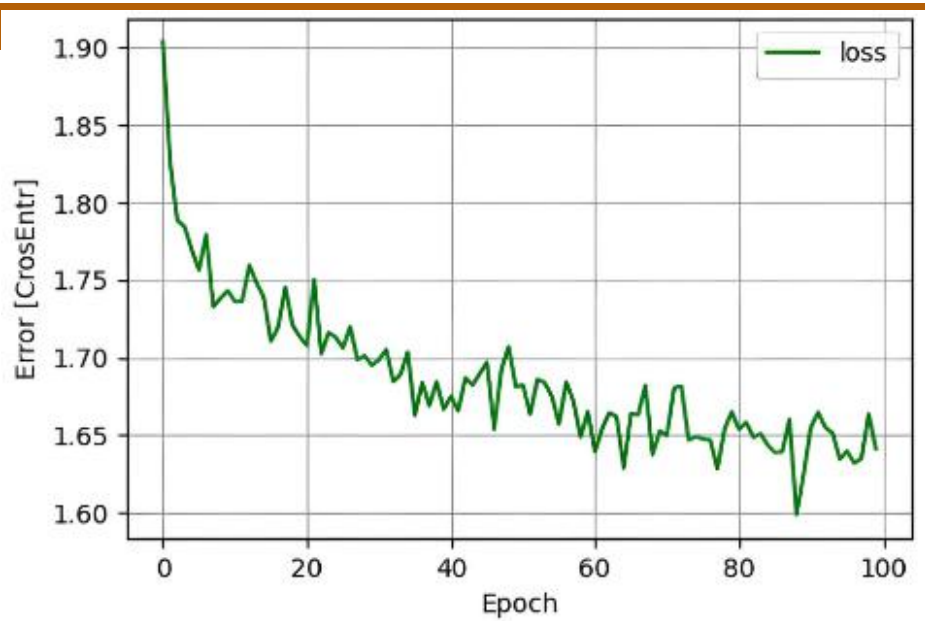


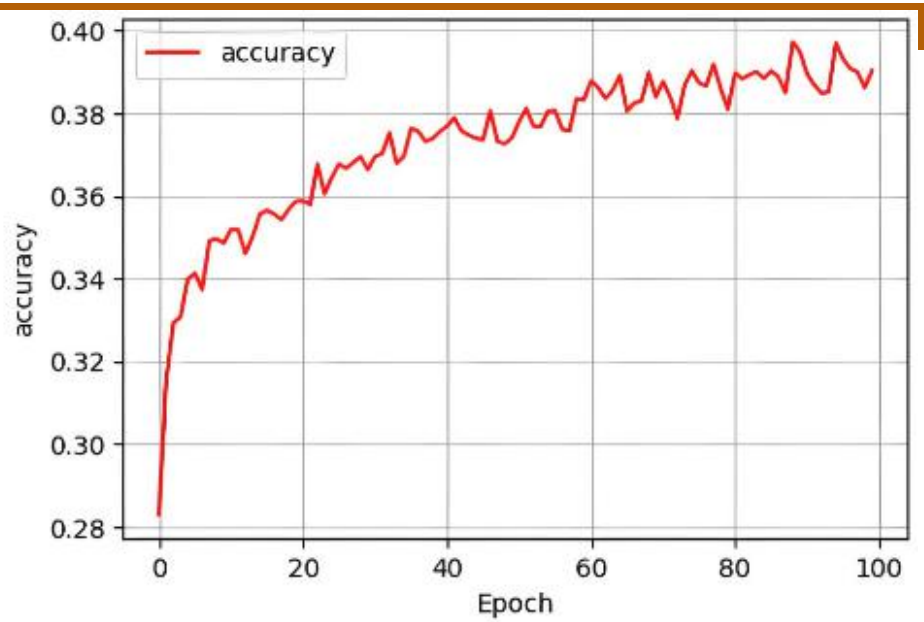


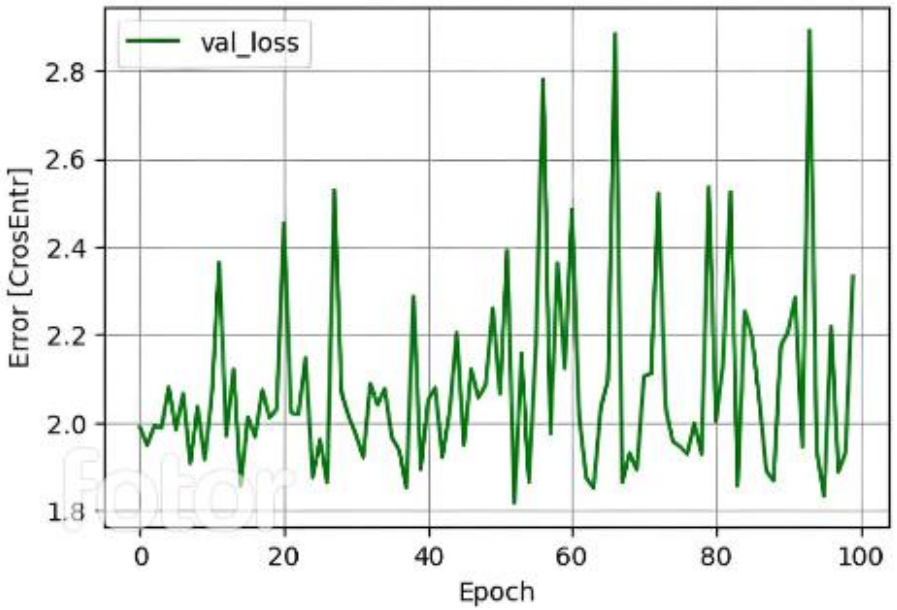


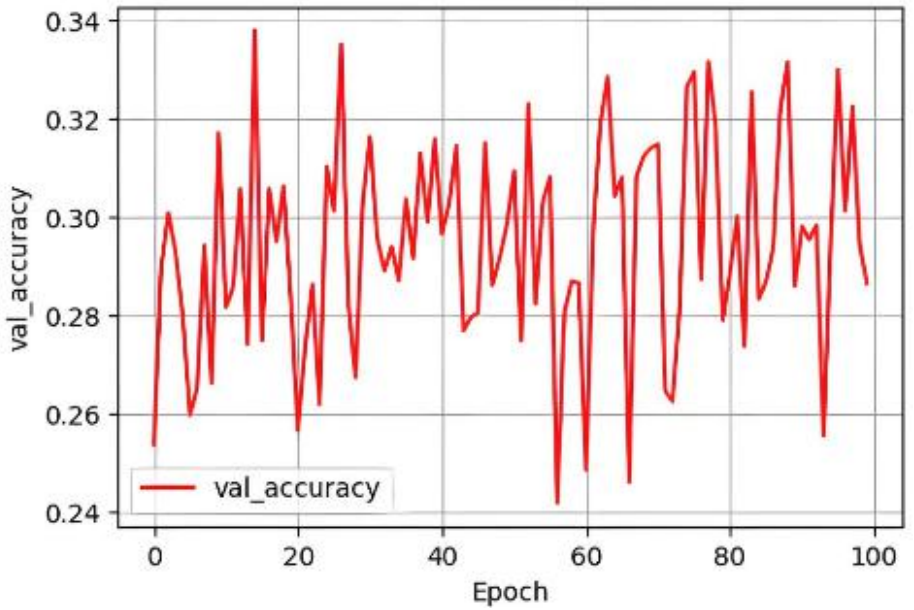
MLP For Classification (Image data)

Model2: Softmax regression - Normalized



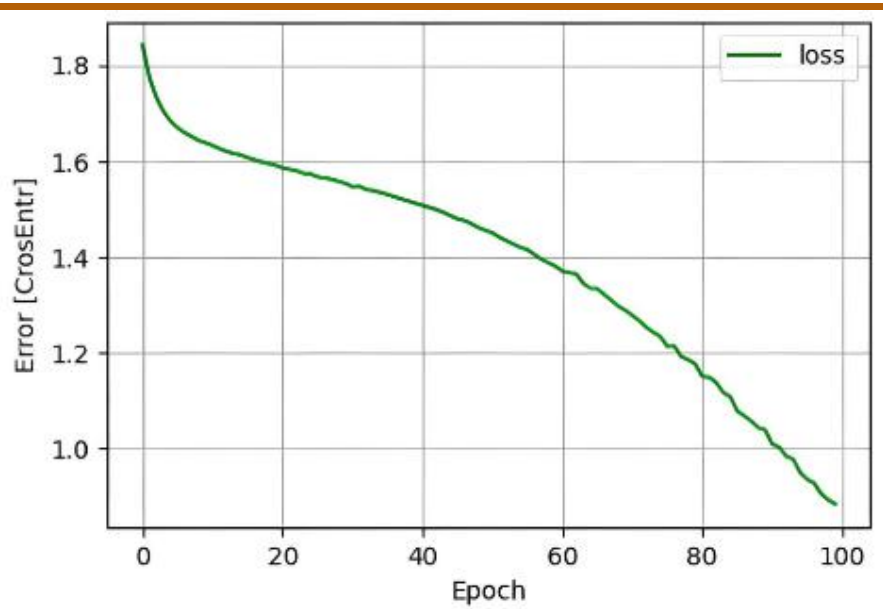


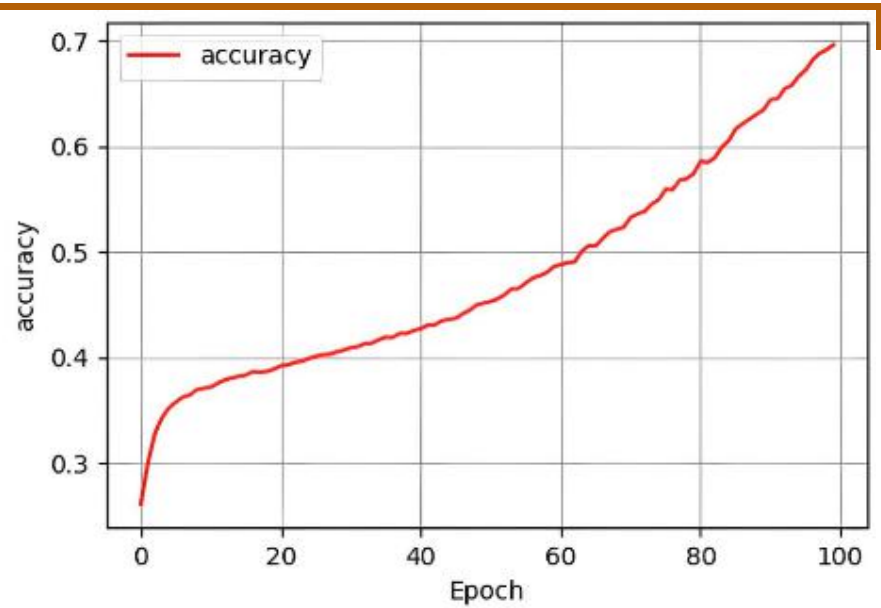


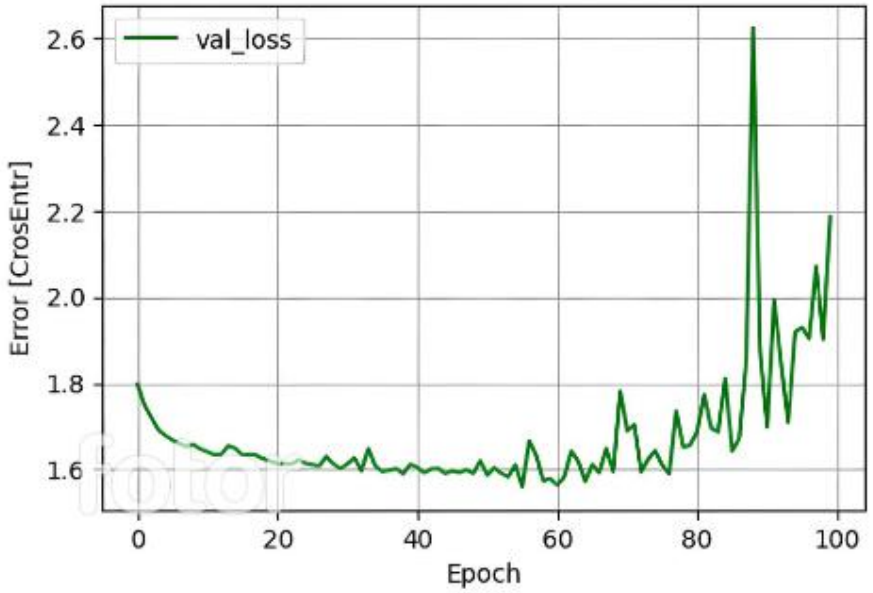


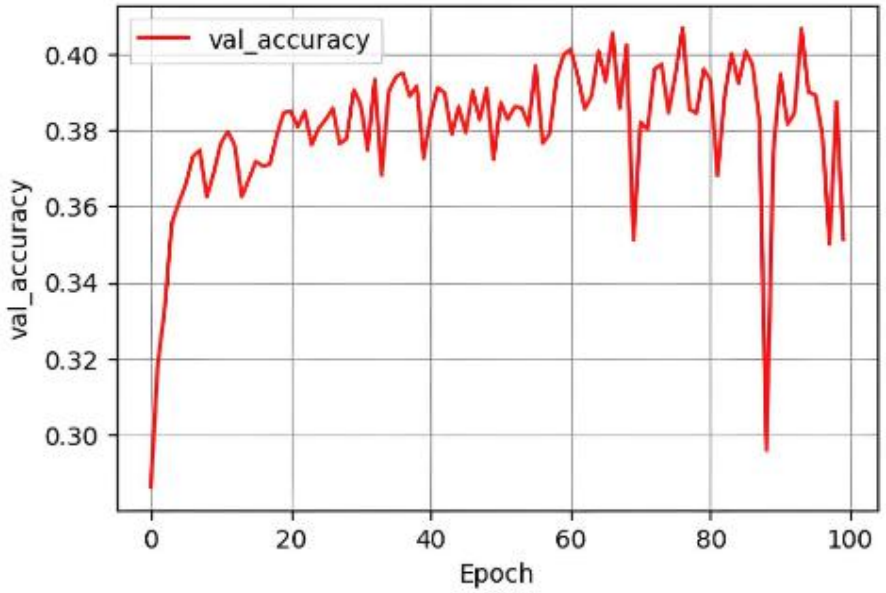
MLP For Classification (Image data)

Model3: MLP - Tanh - Normalized



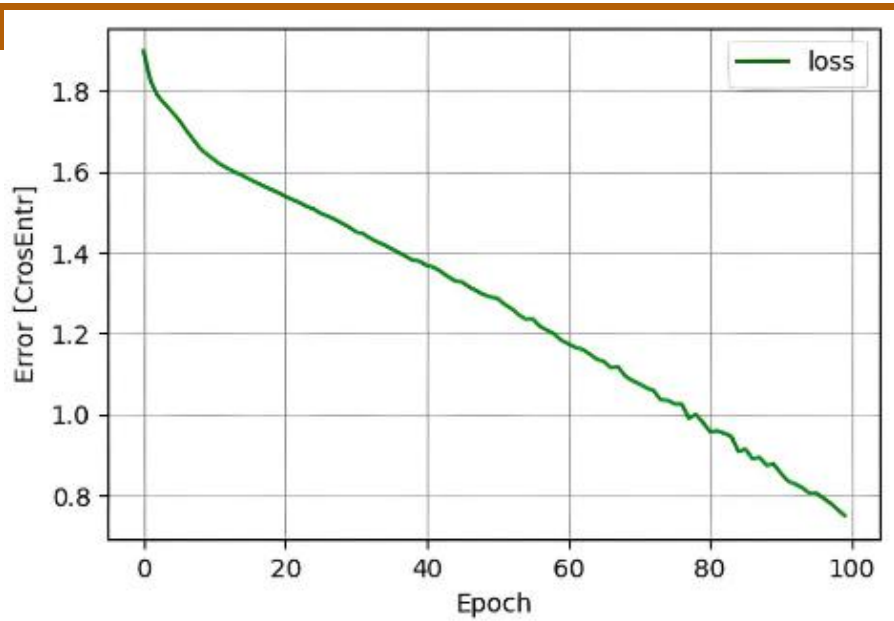


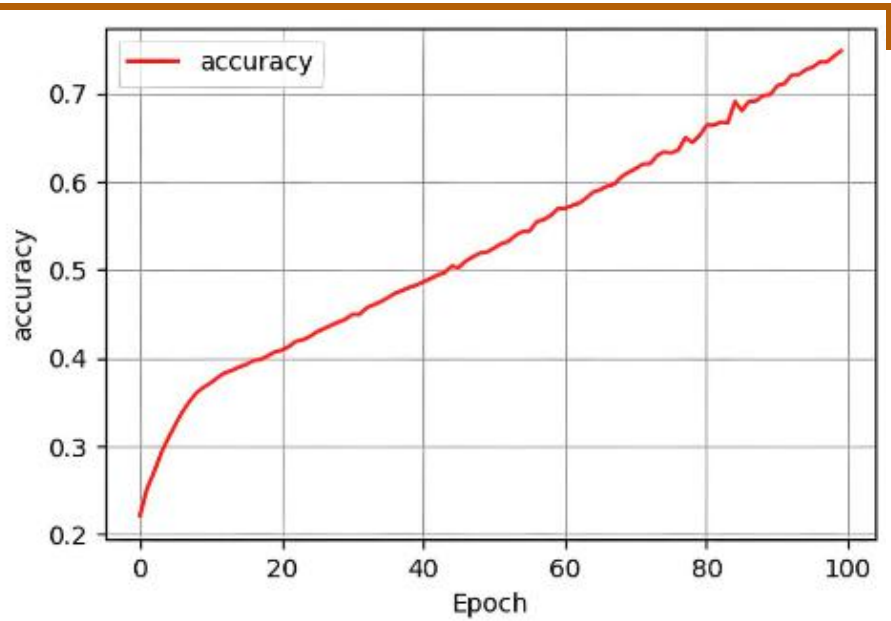


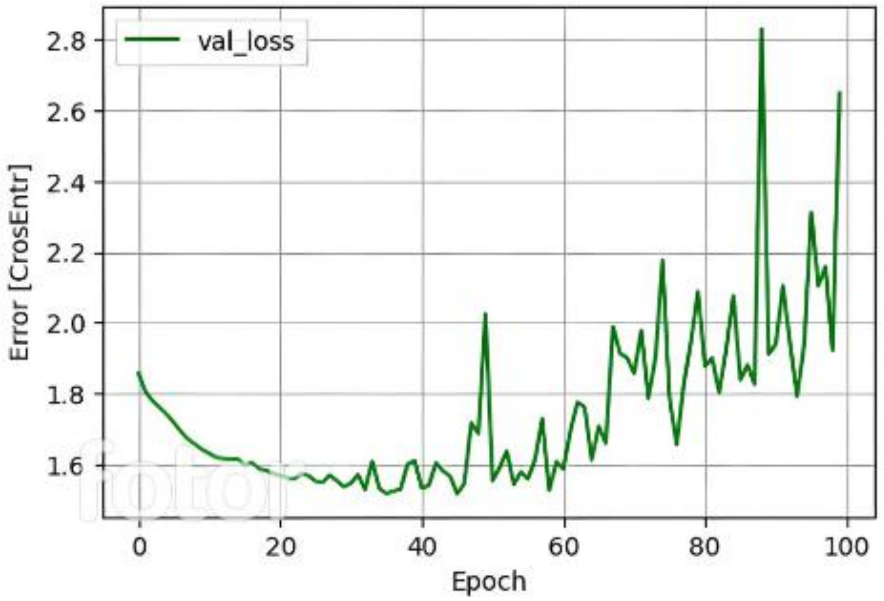


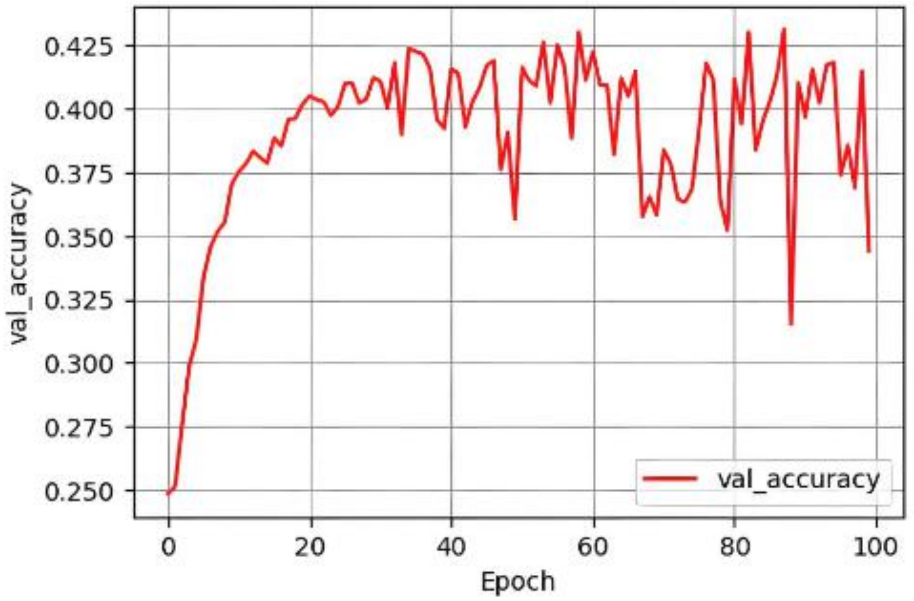
MLP For Classification (Image data)

Model4: MLP - Relu - Normalized





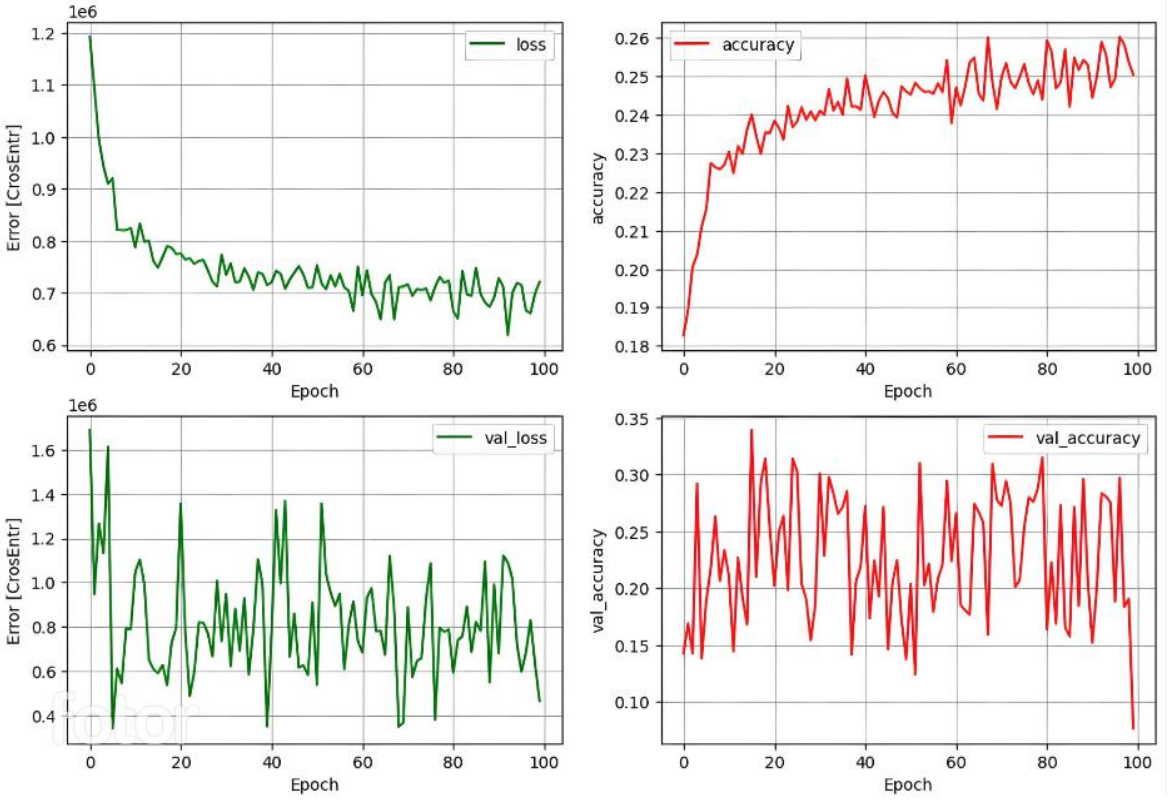




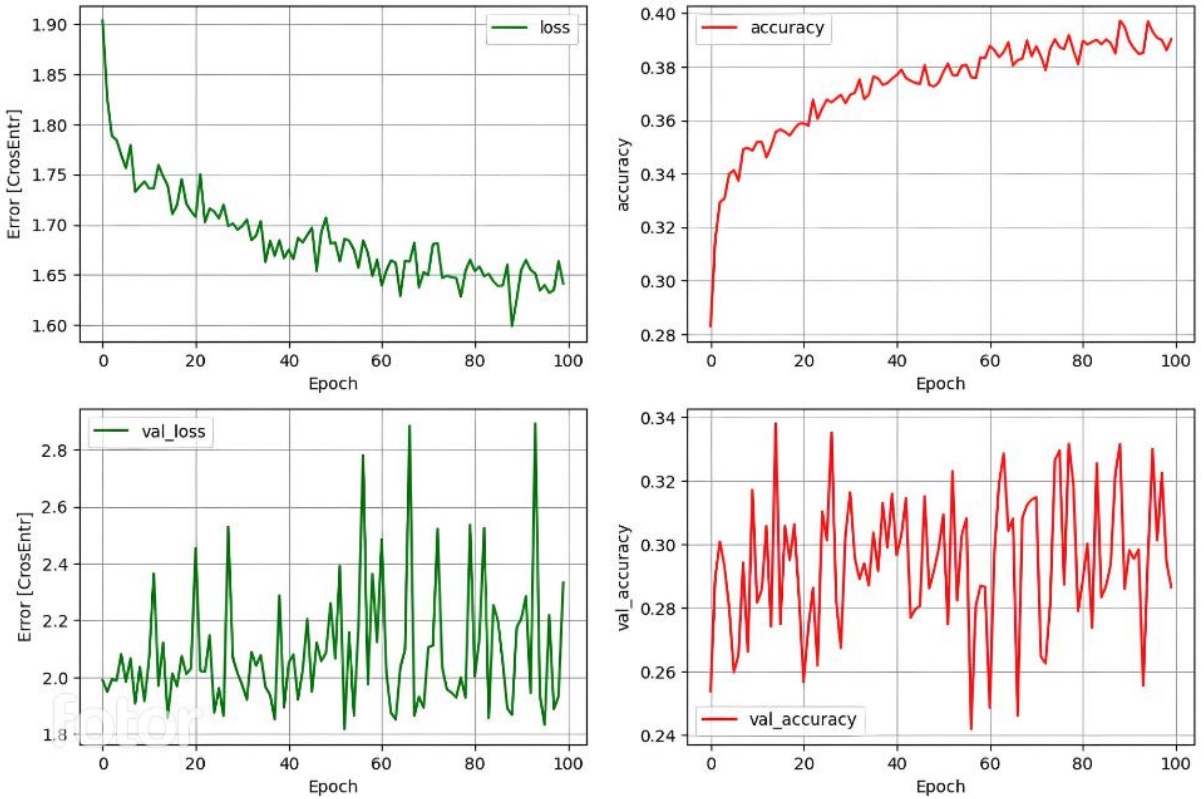
AI VIETNAM
All-in-One Course
(TA Session)

MLP For Classification (Image data)

Model1: Softmax regression

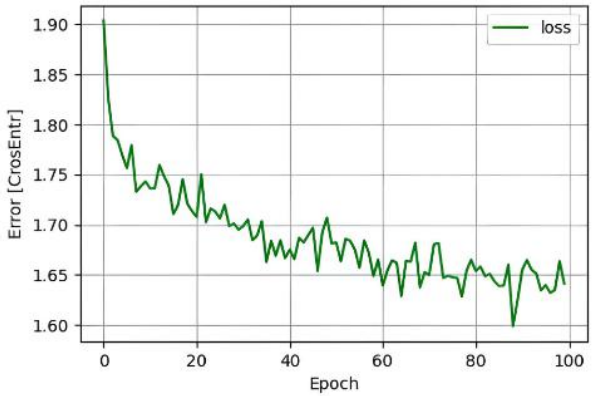


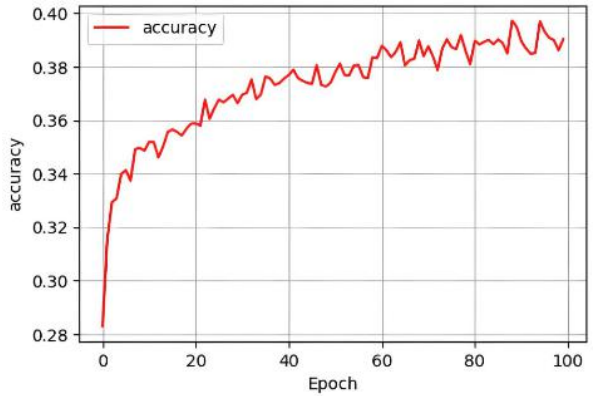
Model2: Softmax regression - Normalized

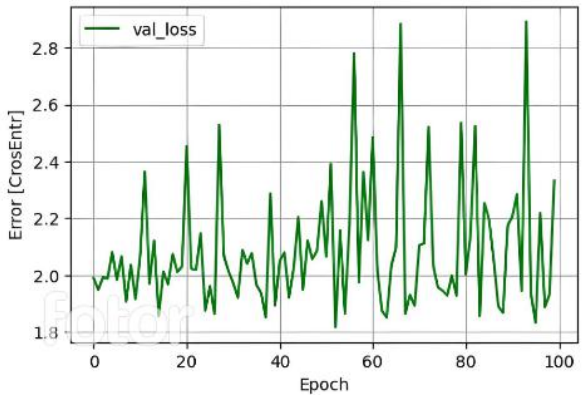


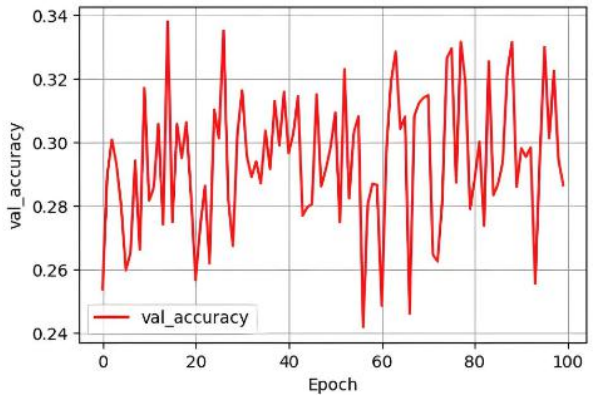
MLP For Classification (Image data)

Model2: Softmax regression - Normalized

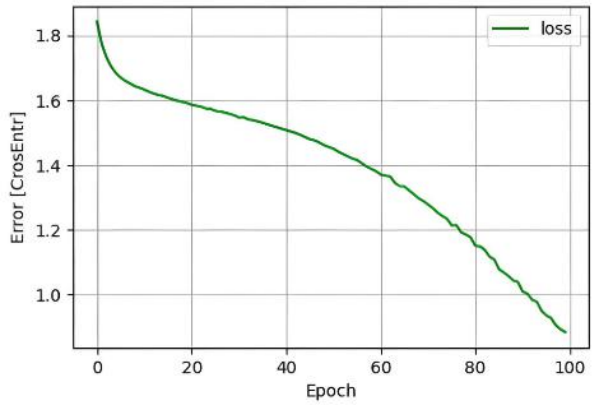


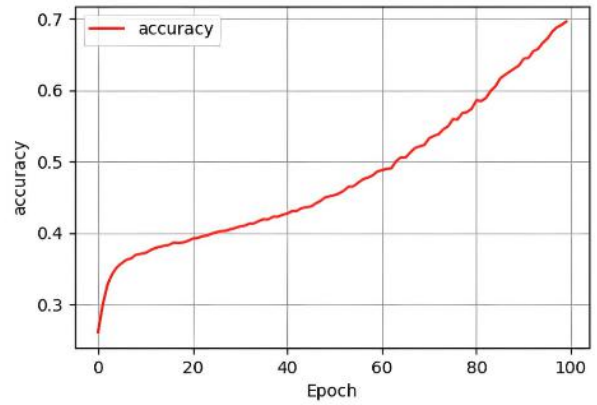


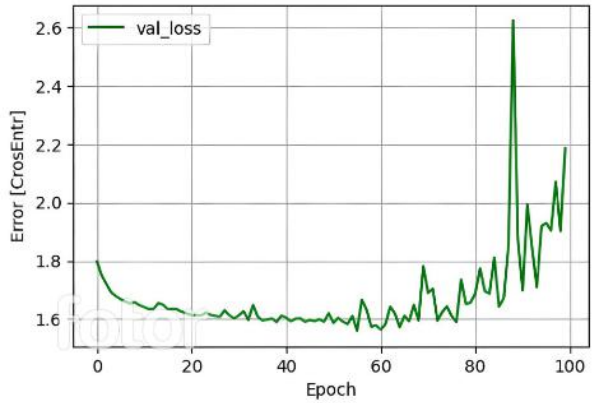


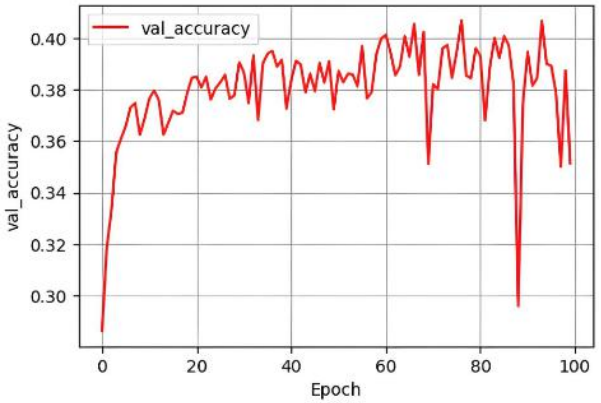


Model3: MLP - Tanh - Normalized





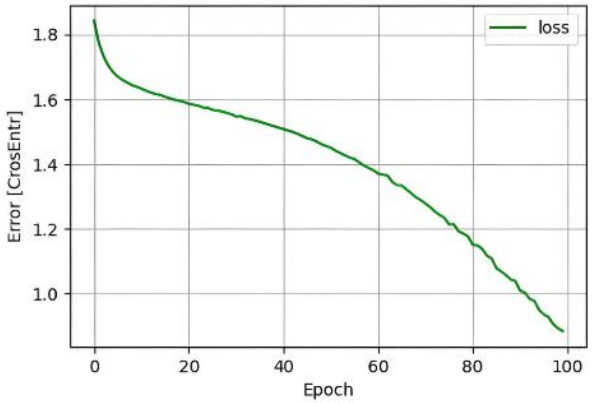


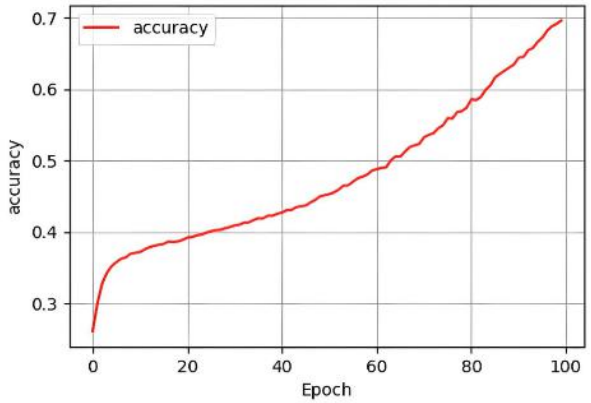


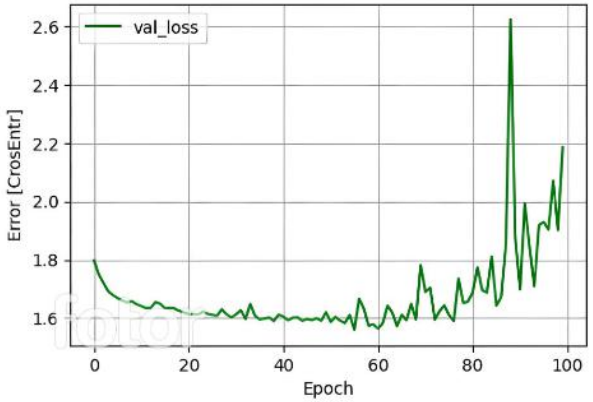
AI VIETNAM
All-in-One Course
(TA Session)

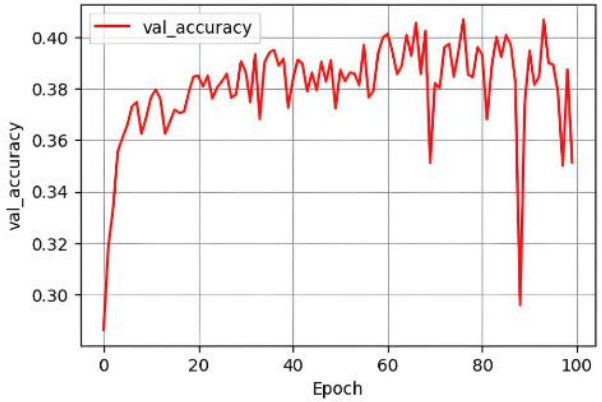
MLP For Classification (Image data)

Model3: MLP - Tanh - Normalized

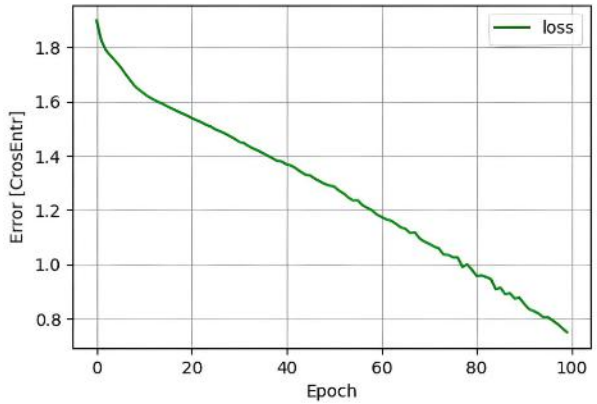


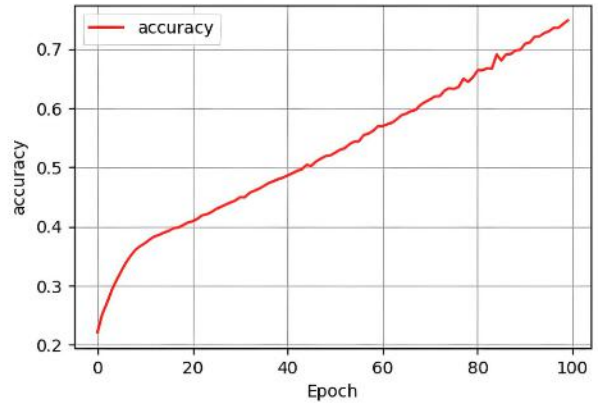


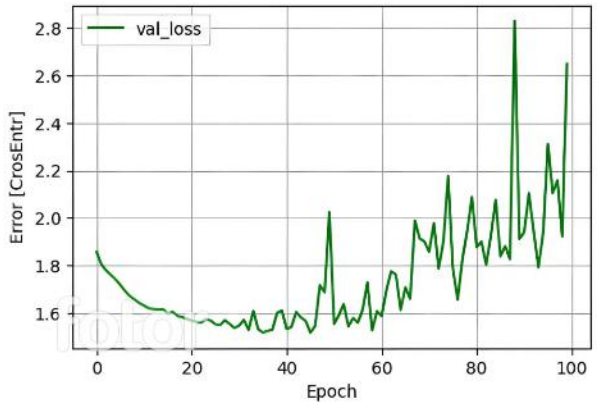


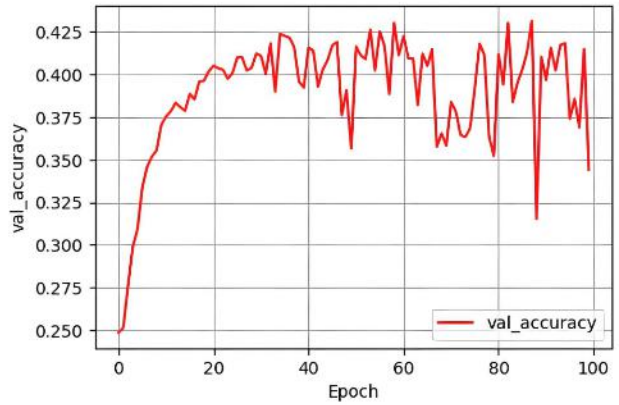


Model4: MLP - Relu - Normalized





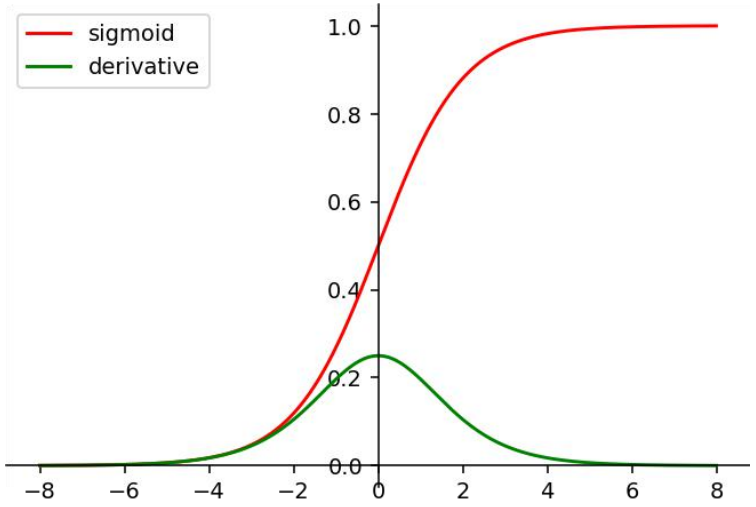




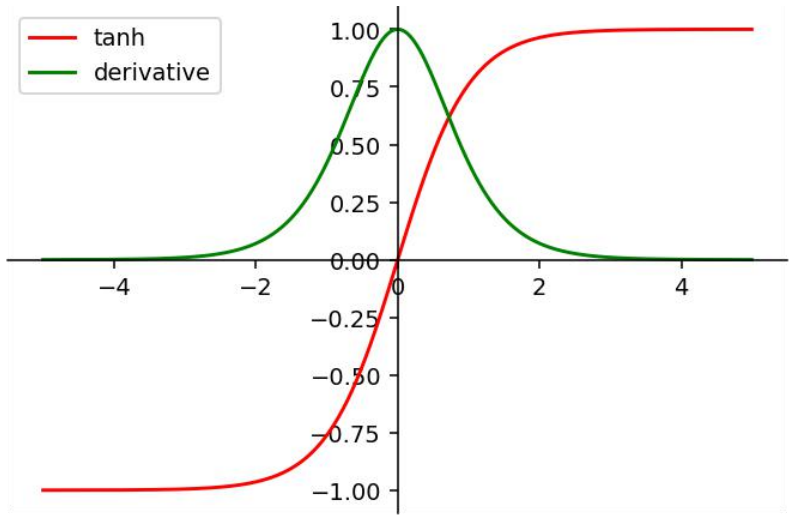
MLP For Classification (Image data)

- **Activation Function: Sigmoid, Tanh, Relu**

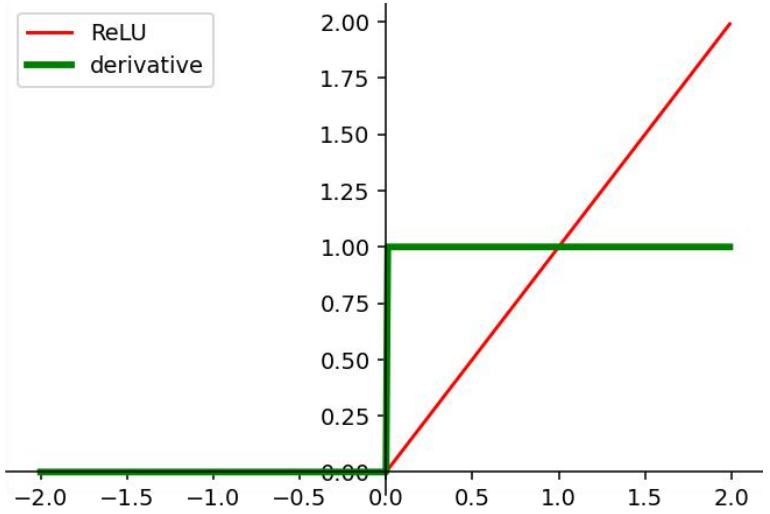
$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$



$$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$



$$\text{ReLU}(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$



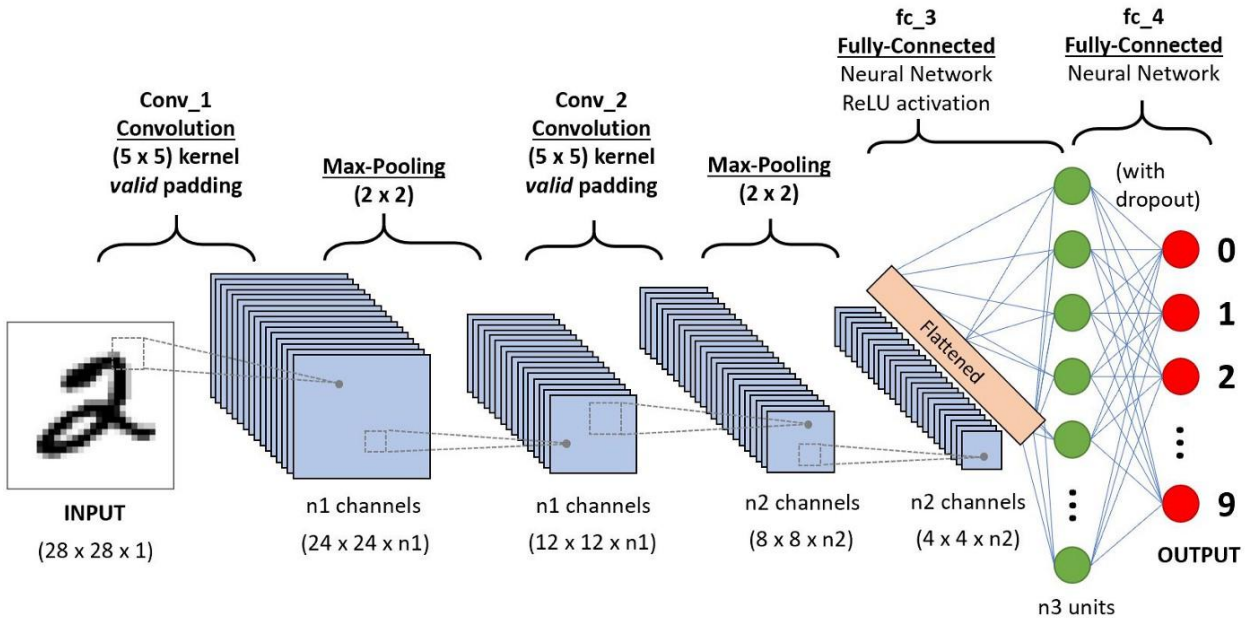
AI VIETNAM
All-in-One Course
(TA Session)

Modern Uses of MLPs

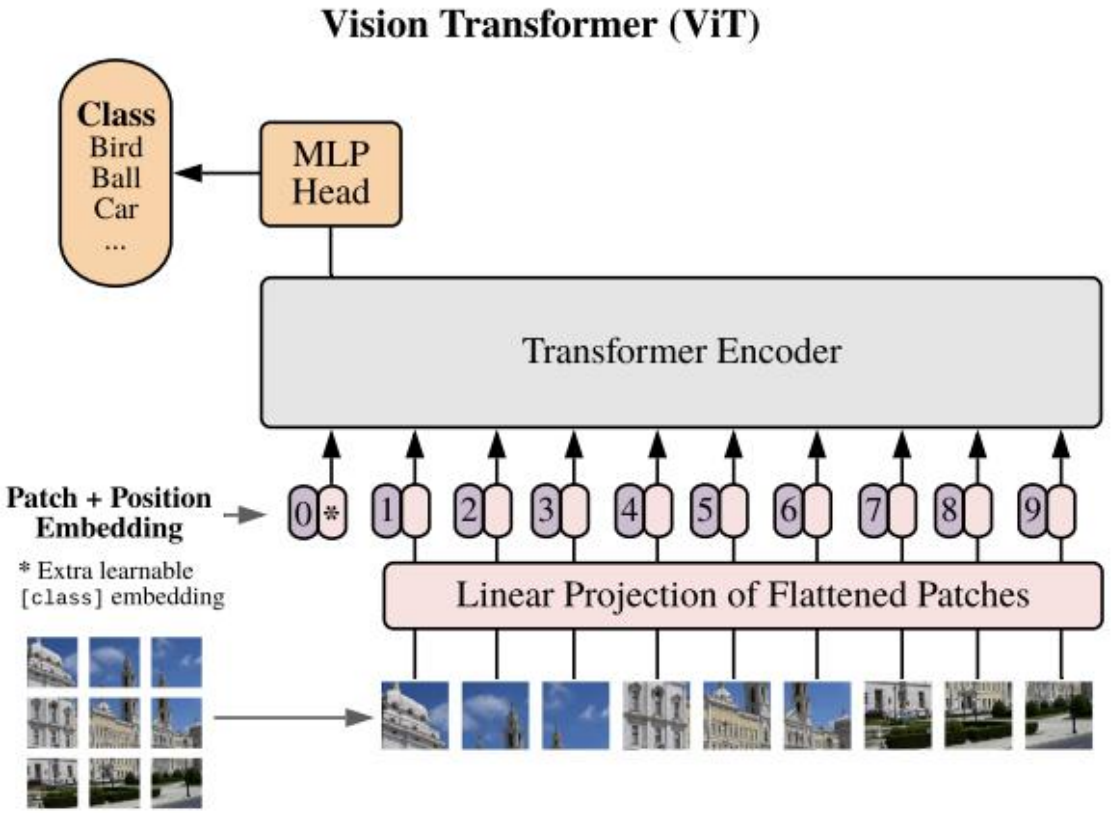
Traditional Use:
Definition: Multi-Layer Perceptrons (MLPs) are a class of feedforward neural networks that consist of at least three layers of nodes: an input layer, hidden layer(s), and an output layer. Each node is a neuron that uses a nonlinear activation function, typically a sigmoid or a rectified linear unit (ReLU).
Applications: Due to their straightforward architecture, MLPs have historically been used for simpler tasks such as basic binary or multiclass classification problems.

Modern Use:
Evolving Complexity: As the field of artificial intelligence progressed, the usage of MLPs has evolved. They are now being integrated into more complex, cutting-edge vision models.
Vision Transformers (ViTs): In Vision Transformers, MLPs are utilized within the transformer blocks, specifically in the feed-forward neural networks, aiding in learning position-wise non-linearities over the channels which is crucial for processing image data.
MLP-Mixer: The MLP-Mixer architecture takes this a step further by relying solely on MLPs for both mixing spatial and channel information, offering a novel approach to image classification tasks.

Modern Uses of MLPs



CNN

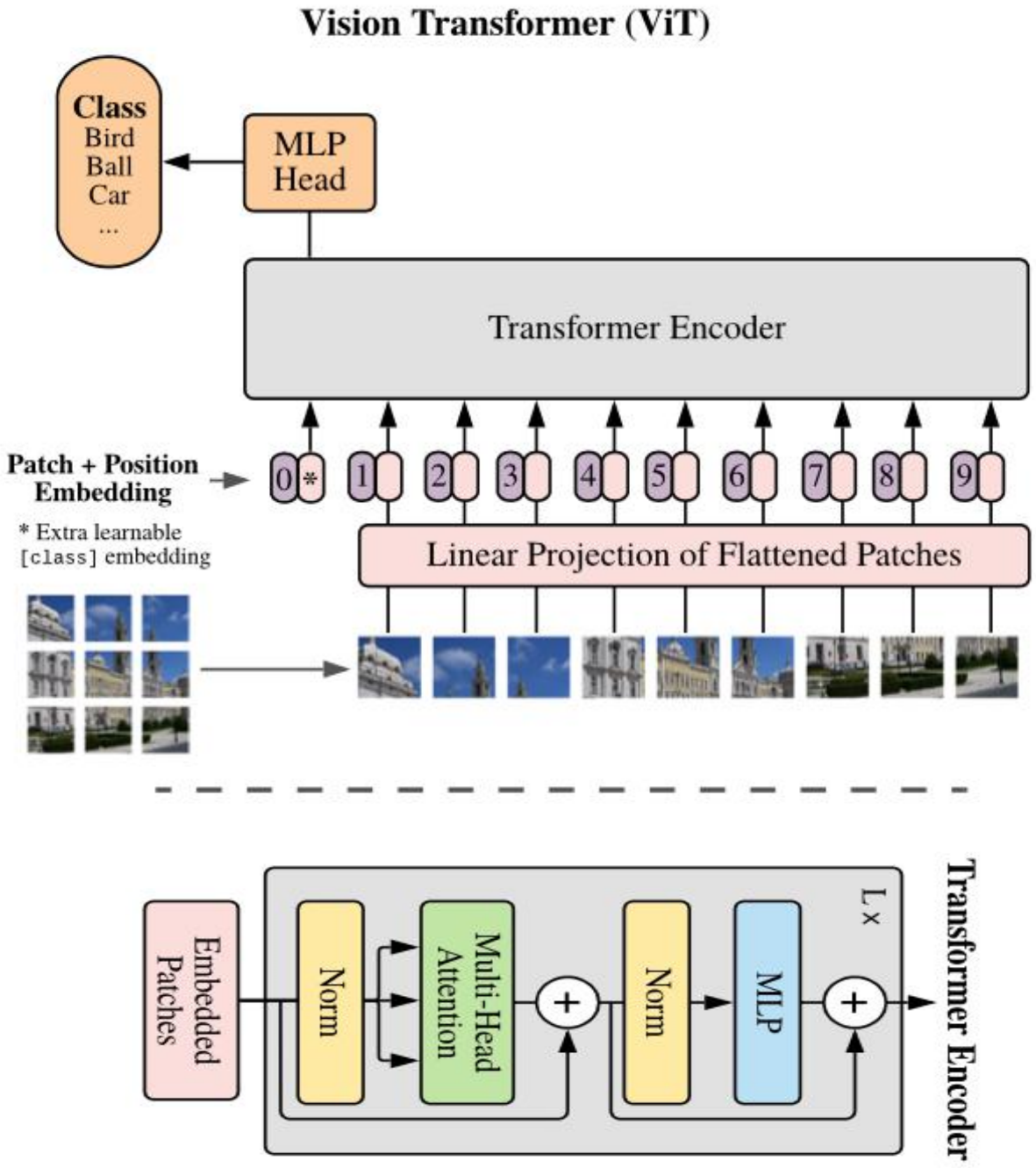


Modern Us

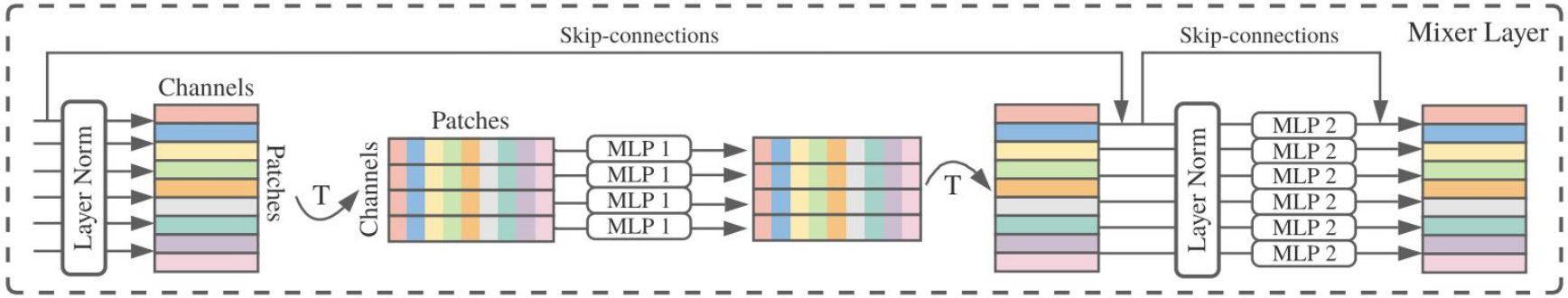
MLPs in FFNs:
Location in Architecture: Within each transformer block, there exists a feed-forward neural network (FFN) which comprises one or more layers of Multi-Layer Perceptrons (MLPs).

Functionality: The MLPs in the FFNs are responsible for learning position-wise non-linearities over the channels. In other words, they operate independently on each position, applying a series of linear transformations interspersed with non-linear activation functions.

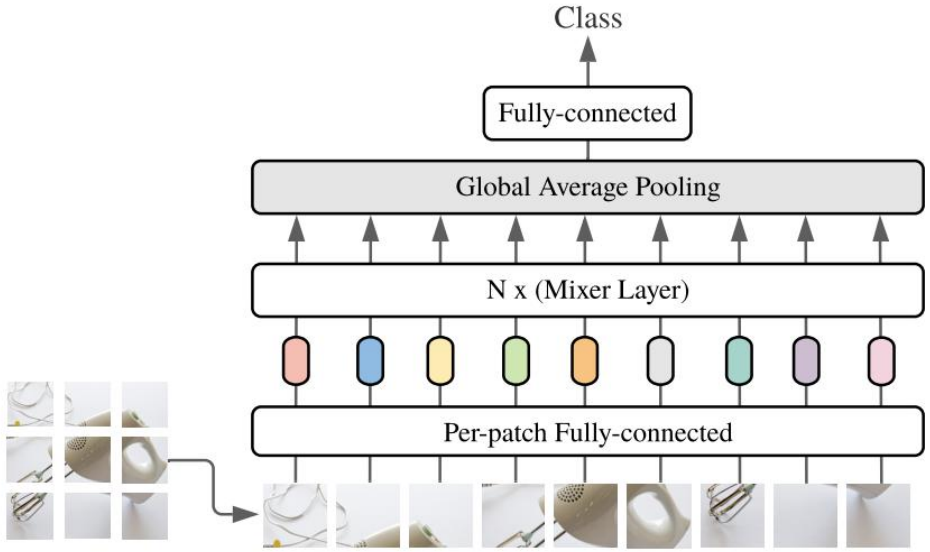
Importance: This mechanism enables the model to learn and represent complex patterns in the data, significantly contributing to the Vision Transformer's ability to understand and interpret image content.



Modern Uses of MLPs



Vision Transformers process the data through transformer blocks with self-attention and MLP components, MLP-Mixer dose not use the self-attention component, showcasing a different approach to handling image data.



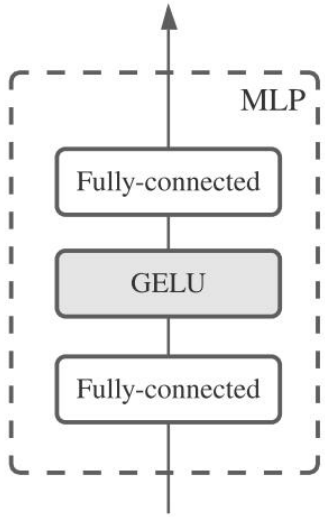


Figure 1: MLP-Mixer consists of per-patch linear embeddings, Mixer layers, and a classifier head. Mixer layers contain one token-mixing MLP and one channel-mixing MLP, each consisting of two fully-connected layers and a GELU nonlinearity. Other components include: skip-connections, dropout, and layer norm on the channels.

MLP Recently Research

Traditional to Modern Transition: MLPs, initially employed for simpler tasks due to their straightforward architecture, have transitioned into being integral components in sophisticated vision models like Vision Transformers and MLP-Mixer.

Architectural Innovation: Modern vision models have leveraged MLPs in novel ways, such as in the mixing of spatial and channel information in MLP-Mixer, and within the feed-forward networks of Vision Transformers, expanding the capabilities of MLPs beyond their traditional use-cases.

Papers:
- **Vision Transformer**: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale
- **MLP-Mixer**: An all-MLP Architecture for Vision