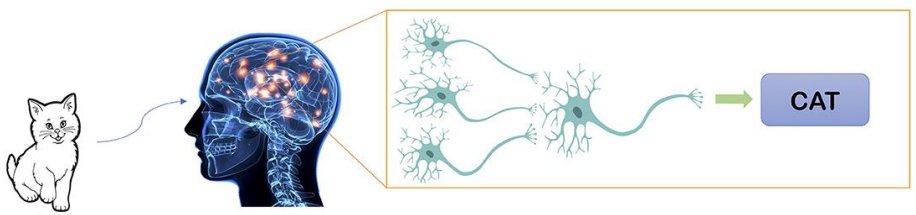
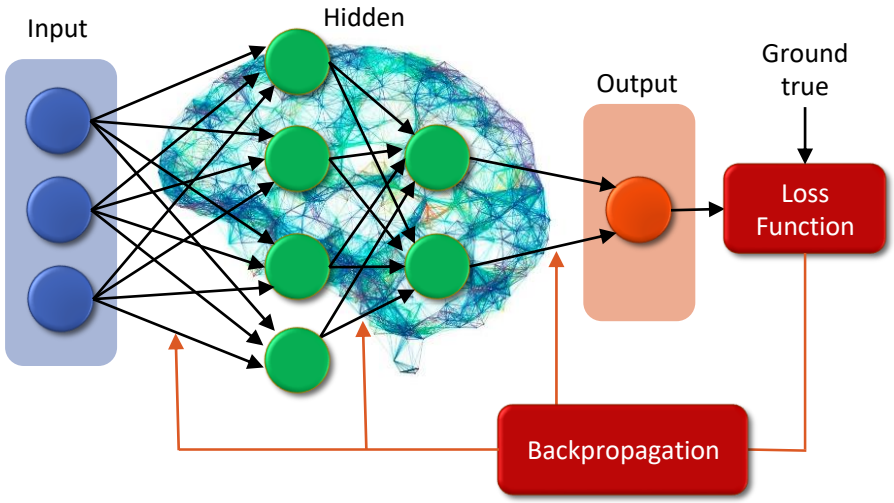
DEEP LEARNING

WORKFLOW VÀ VISUALIZATION CỦA MODEL





Tôn Quang Toại
Khoa Công nghệ thông tin
Trường đại học Ngoại ngữ - Tin học TP.HCM (HUFLIT)

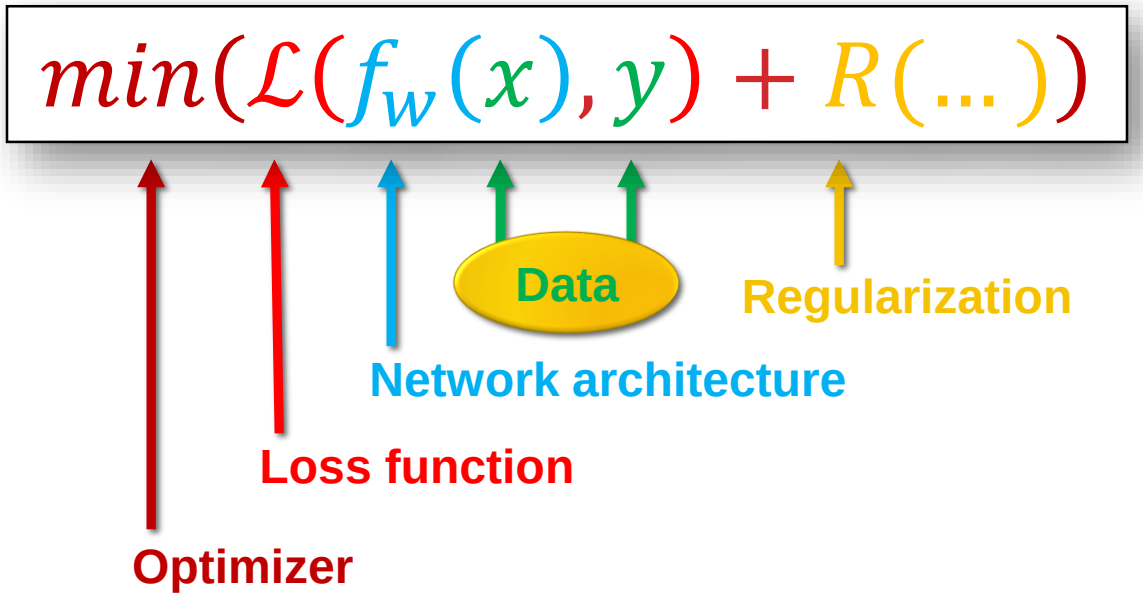
Nội dung

- Workflow của deep learning
- Vẽ kiến trúc của model (graphviz)
- Biểu đồ huấn luyện (matplotlib)
- Đánh giá mô hình và diễn giải mô hình cuối cùng (confusion matrix)

WORKFLOW CỦA DEEP LEARNING

Workflow

- **5 thành phần của mô hình deep learning**
  - Data
  - Network architecture
  - Loss function
  - Optimizer
  - Regularization



- **Mục tiêu của model**
  - Mô hình có tính tổng quá (generalization) trên dữ liệu chưa thấy

Workflow: Data

- **Bước 1. Hiểu rõ bài toán**
  - Mục tiêu/Yêu cầu của bài toán
  - Input
    - 1D tensor
    - 2D tensor (samples, features)
    - 3D tensor (samples, timeseries, features)
    - **4D tensor** (samples, width, height, channel): image
    - 5D tensor (samples, timeseries, width, height, channel): video
  - Output
    - Binary classification
    - Multiclass classification
    - …

Workflow: Data

- **Bước 2. Chuẩn bị Data**
  - Load dữ liệu
  - Chia tập dữ liệu
    - Training data
    - Validation data
    - Test data
  - Tiền xử lý
    - Resize image
    - Chuẩn hóa dữ liệu ($x \in [0,1]$ hay $x \in [-1,-1]$ hay $x \in [0,1]$)
    - Sinh thêm dữ liệu (thường để sau)

Workflow: Network architecture

- **Bước 3. Chọn kiến trúc mô hình cơ bản**
  - Chọn 1 kiến trúc mô hình cơ bản: ban đầu nên chọn kiến trúc đơn giản nhất, ví dụ: MLP, LeNet, …)
- **Bước 4. Chọn các thành phần của mô hình**
  - Chọn đúng activation function ở tầng cuối: ReLU, Softmax
  - Chọn loss function: binary crossentropy, category crossentropy
  - Chọn phương pháp đo (metrics): Accuracy
  - Chọn optimizer: SGD
- **Bước 5. Kiểm tra khả năng học của mô hình**
  - Kiểm tra khả năng học của mô hình (loss có giảm không)
  - Nếu không quay lại Bước 4

Workflow: Network architecture

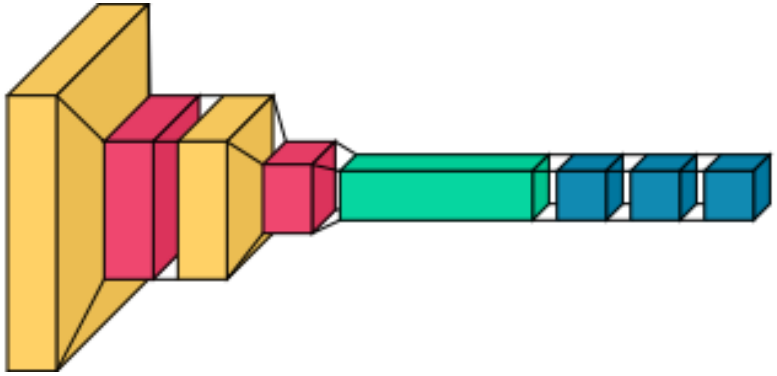
- Bảng thông tin kiến trúc của model

Layer type	Output size	Filter size / Stride	Params
Input image			
Conv			
Activation			
MaxPooling			
Flatten			
Dense/FC			
		Tổng:	

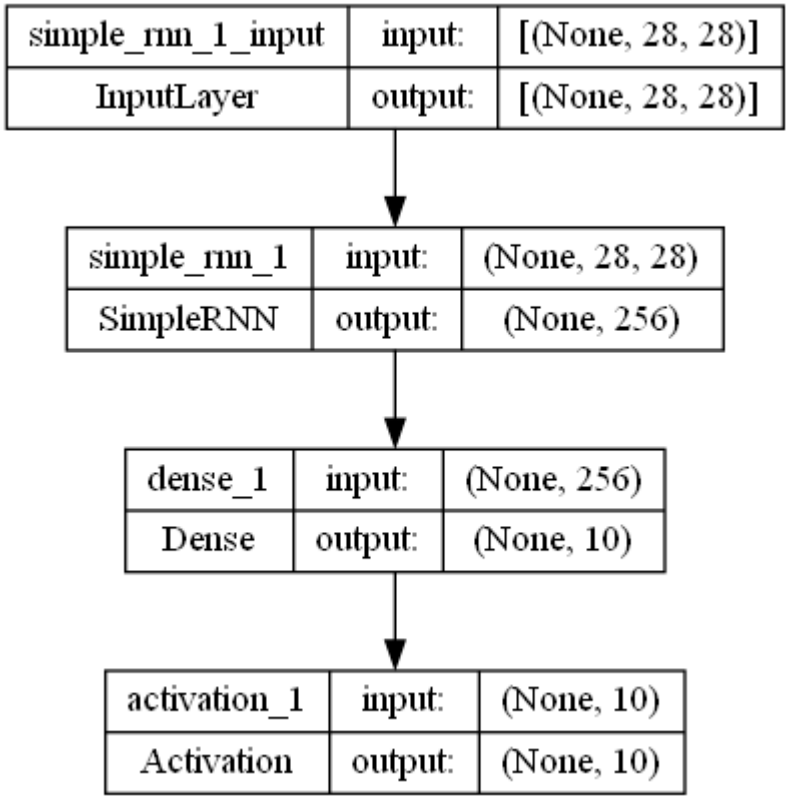
```
model = ...
print(model.summary())
```

Workflow: Network architecture

- Trực quan kiến trúc của model (graph)



visualkeras



Keras Visualization

Workflow: Network architecture

- Một graph của model gồm
  - Node: biểu diễn các layer
  - Connection giữa các nodes: biểu diễn luồng dữ liệu qua mạng
- Một node gồm có
  - Kích thước input
  - Kích thước output
  - Tên của layer
- Sử dụng graph
  - Debug kiến trúc của model
  - Xuất bản bài báo khoa học (viết tài liệu báo cáo)

Workflow: Network architecture

- Một số công cụ visualize kiến trúc model
  - Keras Visualization (dùng graphviz)
  - visualkeras
  - NN-SVG (http://alexlenail.me/NN-SVG/LeNet.html)
  - …

Workflow: Network architecture

- Keras Visualization (dùng graphviz)
  - Cài đặt gói: https://graphviz.org/download/
  - Tạo đường dẫn đến thư viện cài đặt vào biến mối trường
  - Cài các packages
  - Vẽ

```
pip install graphviz
pip install pydot
pip install pydotplus
```

```
from tensorflow.keras.utils import plot_model

model = ...
plot_model(model, to_file="file.png", show_shapes=True)
```

Workflow: Network architecture

- visualkeras
  - Cài package

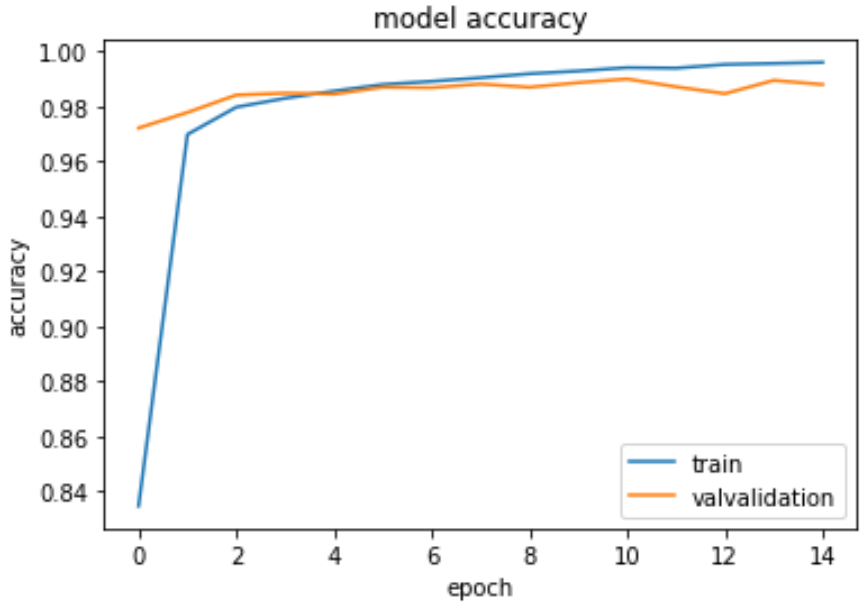
```
pip install visualkeras
```

  - Vẽ

```
model = ...

# display
visualkeras.layered_view(model).show()

# write to disk
visualkeras.layered_view(model, to_file='output.png')

# write and show
visualkeras.layered_view(model, to_file='output.png').show()
```

Workflow: Training

- **Bước 6. Huấn luyện mô hình**
  - Huấn luyện mô hình cho đến khi overfitting → epochs thứ mấy?
  - Ghi nhận kết quả của mô hình trên tập validation tại epochs trên
    - Accuracy
    - F1-Score
    - …
  - Vẽ biểu đồ huấn luyện



MATPLOTLIB

Matplotlib – Giới thiệu

- Matplotlib
  - Matplotlib là một thư viện vẽ, module quan trọng nhất trong matplotlib là matplotlib.pyplot
  - Năm tạo: 2003

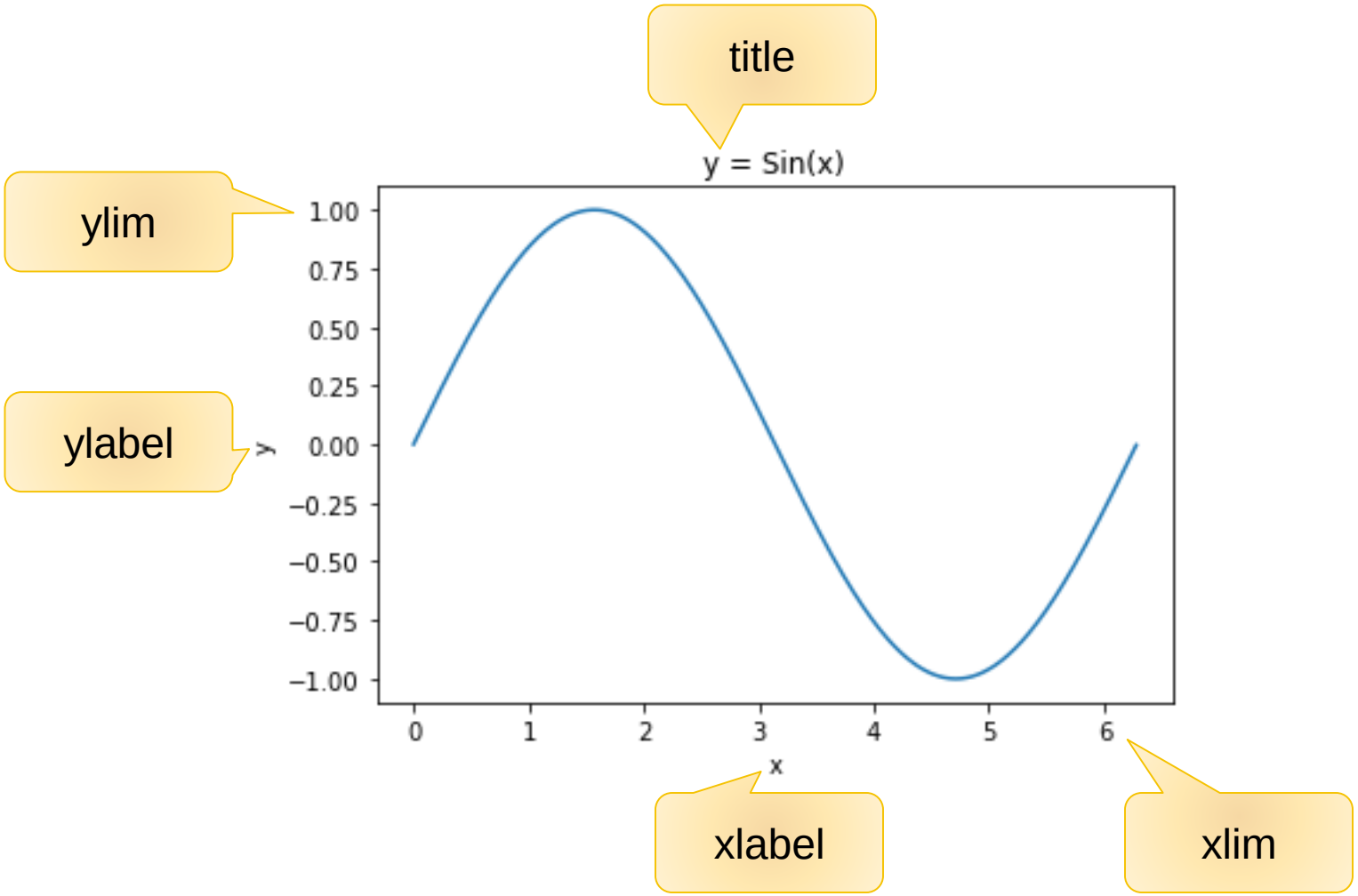
- Tác giả: John D. Hunter
  - 1968 – 2012
- Cài đặt

```
pip install matplotlib
```

- Thêm thư viện

```
import matplotlib.pyplot as plt
```

Matplotlib – Giới thiệu



Matplotlib – Các kiểu vẽ

- Một số hàm vẽ

STT	Hàm	Ý nghĩa
1	plt.plot(x,y,…)	Vẽ đường nối các điểm
2	plt.scatter(x,y,…)	Các điểm dữ liệu
3	plt.hist(x,…)	Vẽ biểu đồ Histogram
4	plt.boxplot(x,y,…)	Hình hộp

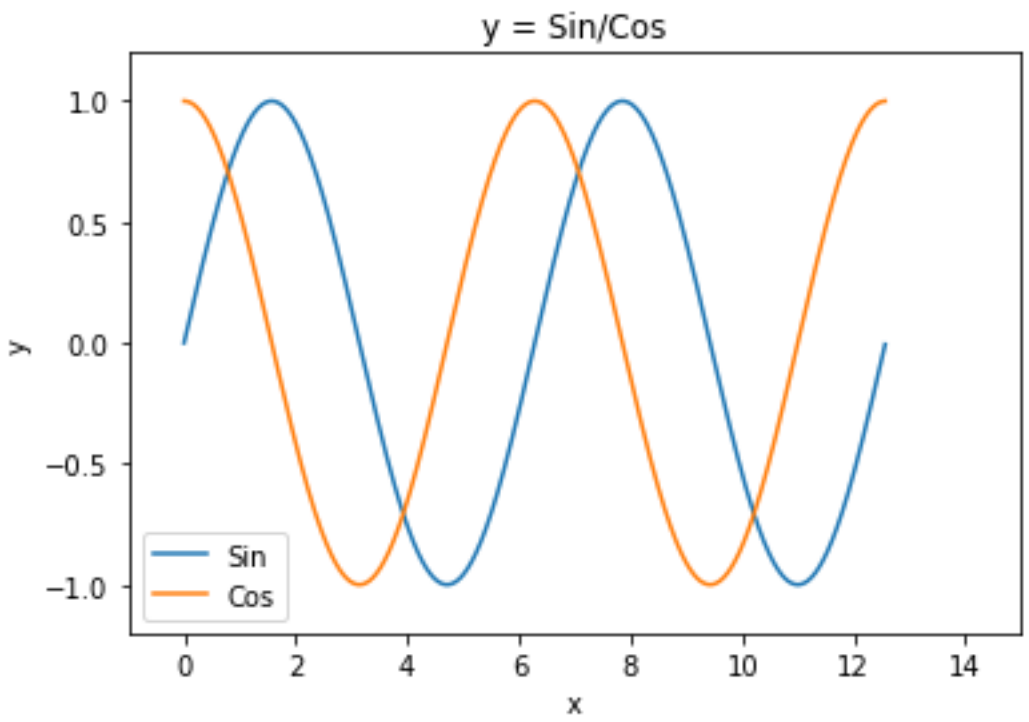
Matplotlib – Hàm về trục

- Hàm thao tác trên trục

STT	Hàm	Ý nghĩa
1	plt.xlabel("") plt.ylabel("")	Nhãn trục x, y
2	plt.title("")	Tiêu đề của hình
3	plt.xlim(min,max) plt.ylim(min,max)	Miền giá trị trục x, y
4	plt.legend(["…", "…", …])	Các ghi chú

Matplotlib – vẽ trên 1 hình

```
import numpy as np
import math
import matplotlib.pyplot as plt

x = np.arange(0, 4*math.pi, 0.01)
y_sin = np.sin(x)
y_cos = np.cos(x)

plt.plot(x, y_sin)
plt.plot(x, y_cos)

plt.title("y = Sin/Cos")

plt.xlabel("x")
plt.ylabel("y")


plt.xlim(-1, 15)
plt.ylim(-1.2, 1.2)

plt.legend(["Sin", "Cos"], )

plt.show()
```



Matplotlib – chia làm 2 hình

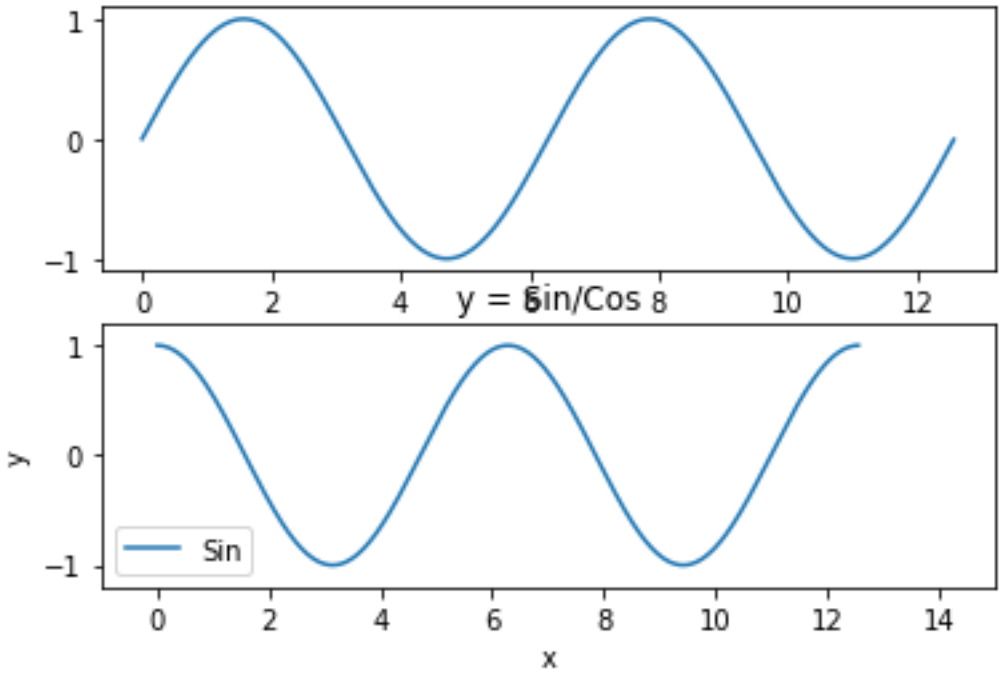
- Chia hình thành các vùng vẽ

```
plt.subplot(row, col, index)
```

  - Chia vùng vẽ thành row dòng và col cột
  - Active vùng index làm vùng vẽ

Matplotlib – chia làm 2 hình

```
x = np.arange(0, 4*math.pi, 0.01)
y_sin = np.sin(x)
y_cos = np.cos(x)

plt.subplot(2,1,1)
plt.plot(x, y_sin)

plt.subplot(2,1,2)
plt.plot(x, y_cos)

plt.title("y = Sin/Cos")

plt.xlabel("x")
plt.ylabel("y")


plt.xlim(-1, 15)
plt.ylim(-1.2, 1.2)

plt.legend(["Sin", "Cos"], )

plt.show()
```



Matplotlib – kiểu dáng

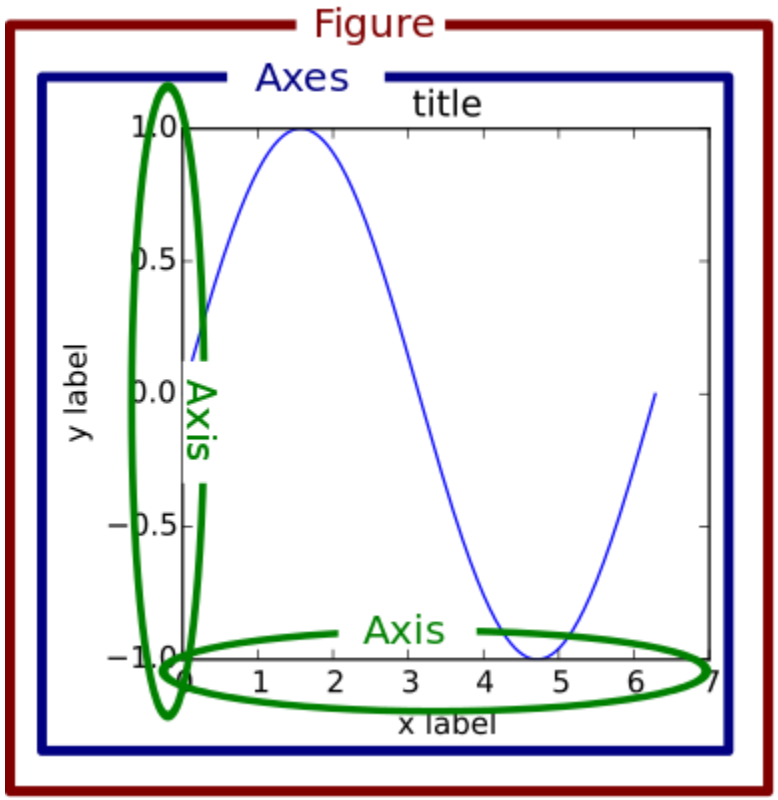
- Kiểu vẽ

```
plt.style.use('ggplot')
```

Matplotlib – hàm plot

- Cú pháp

```
plt.subplot(nrows=1, ncols=1, index)
```



https://realpython.com/python-matplotlib-guide/

Matplotlib – hàm legend

- Cú pháp

```
plt.legend(["", "", …], loc="upper right")
```

  - loc
    - "upper …"
    - "lower …"
    - "… left"
    - "… right"

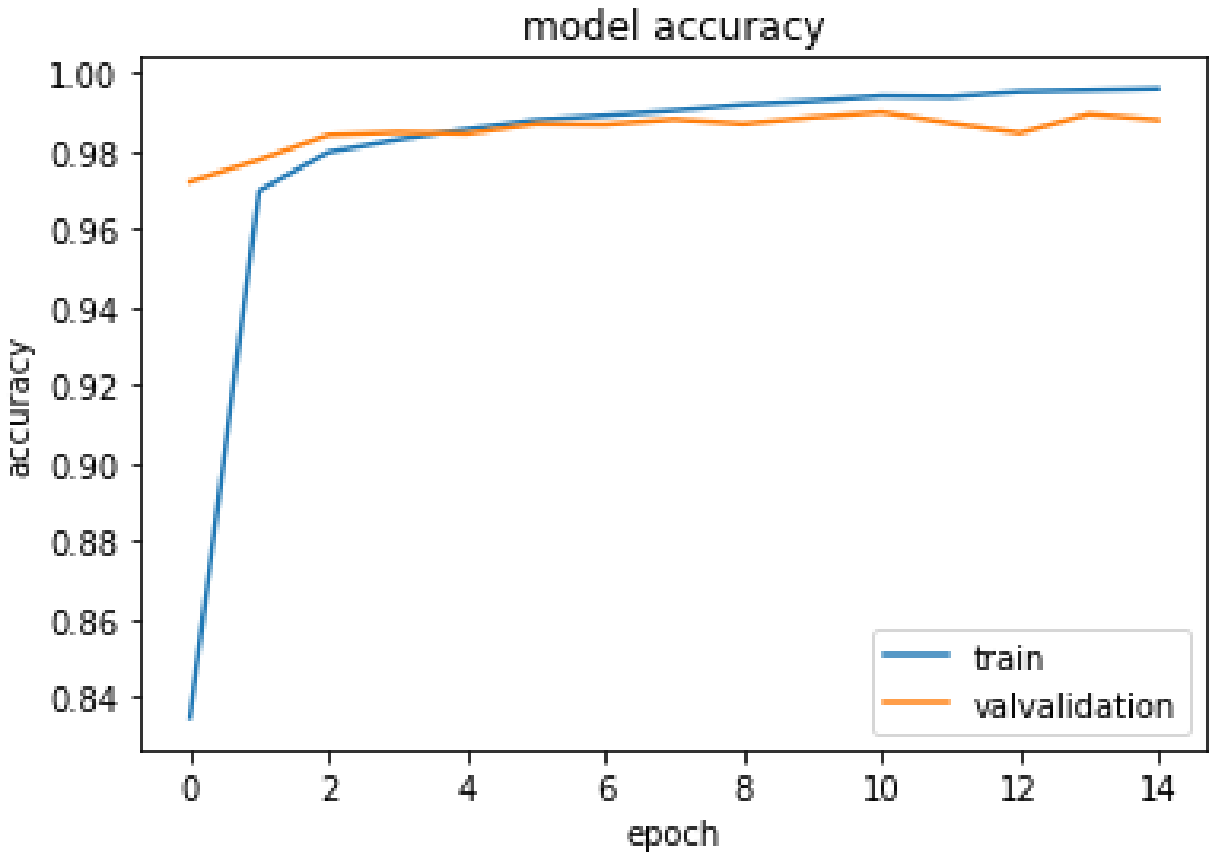
Workflow: Training

- Vẽ độ accuracy của model

```
h = model.fit(x_train, y_train, batch_size=batch_size,
epochs=epochs, validation_split=0.1)

from matplotlib import pyplot as plt

# Plot history
print(h.history.keys())

# summarize history for accuracy
plt.plot(h.history['accuracy'])
plt.plot(h.history['val_accuracy'])

plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')

plt.legend(['train', 'validation'], loc='lower right')
plt.show()
```

Workflow: Training

- Vẽ độ accuracy của model



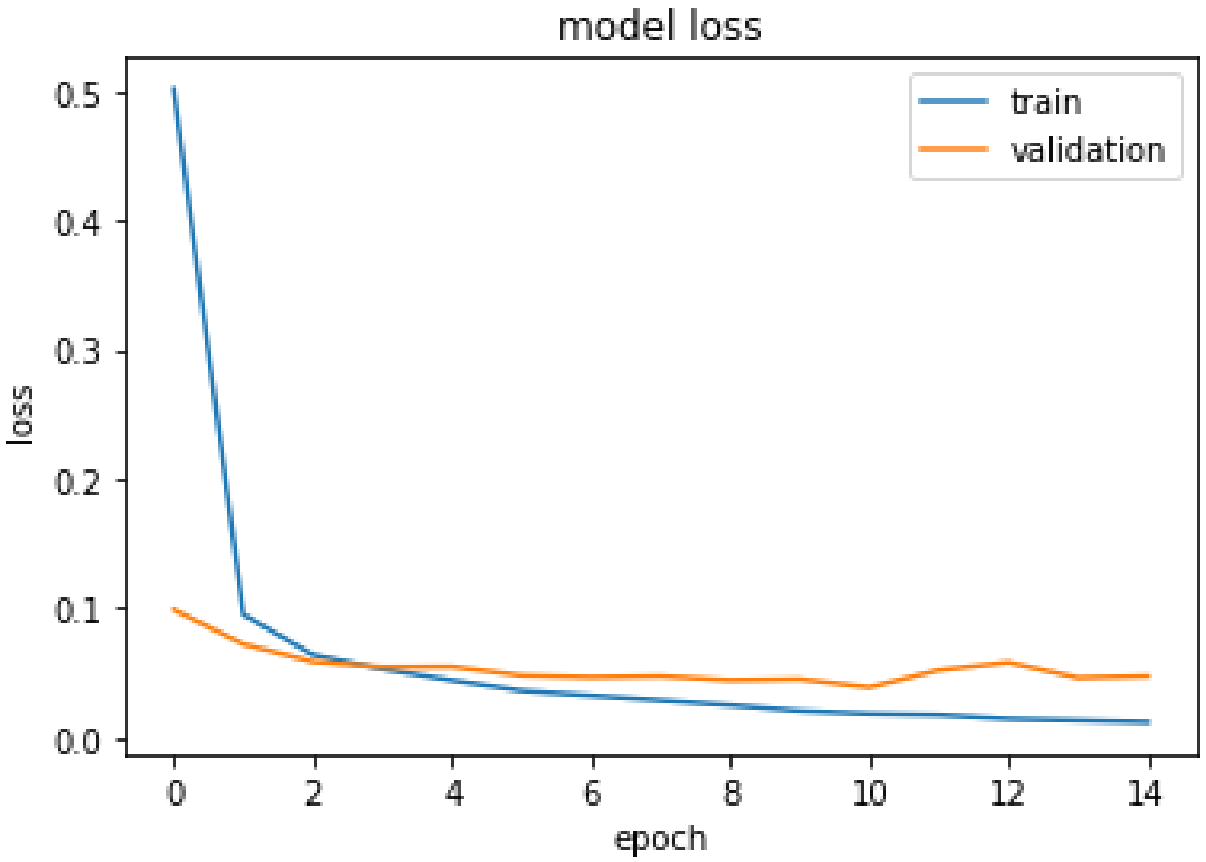
Workflow: Training

- Vẽ độ loss của model

```
# Plot history
print(h.history.keys())

# summarize history for loss
plt.plot(h.history['loss'])
plt.plot(h.history['val_loss'])

plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')

plt.legend(['train', 'validation'], loc='upper right')
plt.show()
```

Workflow: Training

- Vẽ độ loss của model



Workflow: Training

- Nhận xét về biểu đồ huấn luyện
  - Độ chính xác đạt được bao nhiêu
  - Độ lỗi đạt được bao nhiêu
  - Đường cong train và validation có giống nhau không?
    - Nếu không giống thì overfitting ở epoch số mấy
    - Nếu giống thì model không bị overfitting

Workflow: Training

- **Bước 7. Giải quyết vấn đề overfitting**
  - Thêm dữ liệu (thường không có)
  - Sinh thêm dữ liệu
  - Chuẩn hóa dữ liệu (batch norm)
  - Dropout
  - L1, L2

Workflow: Tinh chỉnh kiến trúc

- **Bước 8. Tinh chỉnh kiến trúc**
  - Tinh chỉnh kiến trúc
    - Thêm số tầng
    - Thêm số filter
  - Tinh chỉnh các siêu tham số khác
    - Hệ số học thích nghi
    - Optimizer khác: RMPprop, Adam, …
    - Loss function
  - Quay lại Bước 6

Workflow: Đánh giá mô hình

- **Bước 10. Đánh giá mô hình**
  - Ghi nhận kết quả của mô hình tốt nhất trên tập test
    - Accuracy
    - F1-Score
    - …
- **Bước 11. Chọn mô hình khác**
  - Chọn mô hình khác
  - Quay lại Bước 4

ĐÁNH GIÁ VÀ DIỄN GIẢI MODEL CUỐI CÙNG

Confusion matrix

		Model	
		Predict 0	Predict 1
Ground truth data	Actual 0	True Negative (Âm tính thật) Specificity	False Positive (Dương tính giả)
	Actual 1	False Negative (Âm tính giả)	True Positive (Dương tính thật) Sensitivity

- **Actual**
  - Nhãn thực của examples
  - Có 2 loại dữ liệu: 0 (N) và 1 (P)

- **Predict**
  - Nhãn dự đoán của examples
  - Có 2 loại dự đoán cho một example: 0 (N) và 1 (P)

Confusion matrix

		Predict		
		N	**P**	
Actual	**N**	$a\ (TN)$	$b\ (FP)$	n_1
	P	$c\ (FN)$	$d\ (TP)$	n_2
		m_1	m_2	

Nhãn của dự đoán: P hay N

XY

Dự đoán đúng (T) hay sai (F)

- **Ý nghĩa**
  - a: số lượng example được dự đoán thuộc loại N và example thực sự là N
  - d: số lượng example được dự đoán thuộc loại P và example thực sự là P

  - b: Số lượng example được dự đoán thuộc loại P, nhưng example không là loại P
  - c: Số lượng example được dự đoán thuộc loại N, nhưng example không là loại N

Confusion matrix

- $n_1 = a + b$: số lượng mẫu thuộc loại N
  (Giá trị n_1 chạy qua lại giữa a và b)
- $n_2 = c + d$: số lượng mẫu thuộc loại P
  (Giá trị n_2 chạy qua lại giữa c và d)

- $m_1 = a + c$: số lượng dự đoán là loại P
- $m_2 = b + d$: số lượng dự đoán là loại N

Các thức đo đánh giá mô hình

- Dữ liệu cân bằng
  - Accuracy
  - Error rate
- Dữ liệu không cân bằng
  - Recall
  - Precision
  - F-measure

Accuracy vs Error rate

		Predict		
		N	**P**	
Actual	**N**	$a\ (TN)$	$b\ (FP)$	n_1
	P	$c\ (FN)$	$d\ (TP)$	n_2
		m_1	m_2	

- Accuracy: đo độ chính xác của mô hình trên cả tập dữ liệu (không phân biệt từng lớp)

$$accuracy = \frac{a+d}{a+b+c+d} = \frac{TN+TP}{n_1+n_2}$$

- Misclassification rate (error rate): đo độ lỗi của mô hình trên cả tập dữ liệu (không phân biệt từng lớp)

$$error = \frac{b+c}{a+b+c+d} = \frac{FN+FP}{n_1+n_2}$$

Accuracy

- Nhận xét
  - Accuracy được sử dụng khi examples của các lớp cân bằng
  - Khi các examples trong các lớp không cân đối thì không nên dùng Accuracy làm chỉ số duy nhất để đánh giá mô hình

Recall

		Predict		
		N	**P**	
Actual	**N**	$a\ (TN)$	$b\ (FP)$	n_1
	P	$c\ (FN)$	$d\ (TP)$	n_2
		m_1	m_2	

- Recall: Đo khả năng phát hiện mẫu trong mỗi lớp (trong mỗi lớp, có bao nhiêu mẫu được phát hiện)

$$recall_N = \frac{a}{a+b} = \frac{TN}{n_1} = \frac{TN}{TN+FP}$$

$$recall_P = \frac{d}{c+d} = \frac{TP}{n_2} = \frac{TP}{FN+TP}$$

Precision

		Predict		
		N	**P**	
Actual	**N**	$a\ (TN)$	$b\ (FP)$	n_1
	P	$c\ (FN)$	$d\ (TP)$	n_2
		m_1	m_2	

- Precision: Đo khả năng nhận dạng đúng trong mỗi lớp (example được xác định thuộc lớp P thì, khả năng đúng là bao nhiêu)

$$precision_N = \frac{a}{a+c} = \frac{TP}{m_1} = \frac{TN}{TN+FN}$$

$$precision_P = \frac{b}{b+d} = \frac{TP}{m_2} = \frac{TP}{FP+TP}$$

F-Measure

- F-Measure: Kết hợp giữa Recal và Precision

$$F - measure = 2 \times \frac{Recall \times Precision}{Recall + Precision}$$

- Hàm trong sk-learn

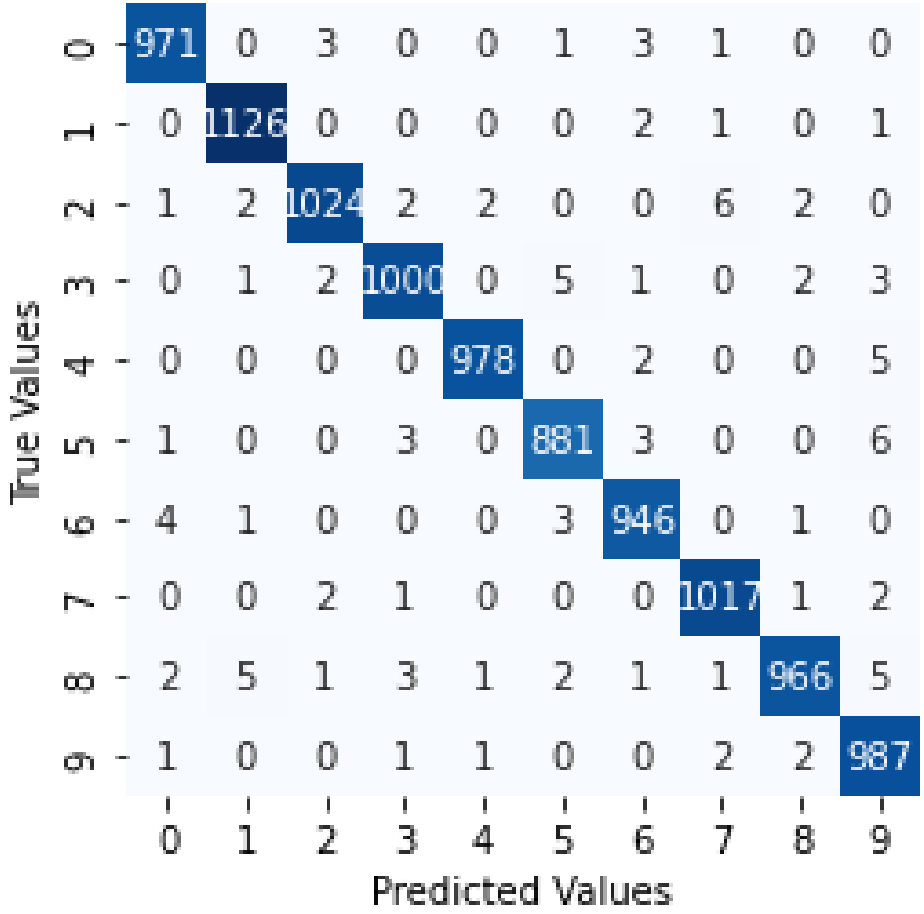
```
print(classification_report(y_true, y_pred, target_names=target_names))
```

Code

- Confusion matrix

```
from sklearn.metrics import confusion_matrix
import seaborn as sns

# Confusion matrix
mat = confusion_matrix(Y_test, Y_pred)
print(mat)

# Draw confusion matrix
sns.heatmap(mat.T, square=True, annot=True, cbar=False, cmap=plt.cm.Blues, fmt='.0f')
plt.xlabel('Predicted Values')
plt.ylabel('True Values');

plt.show();
```

Code


True Values
Predicted Values
0 971 0 3 0 0 1 3 1 0 0
1 0 1126 0 0 0 0 2 1 0 1
2 1 2 1024 2 2 0 0 6 2 0
3 0 1 2 1000 0 5 1 0 2 3
4 0 0 0 0 978 0 2 0 0 5
5 1 0 0 3 0 881 3 0 0 6
6 4 1 0 0 0 3 946 0 1 0
7 0 0 2 1 0 0 0 1017 1 2
8 2 5 1 3 1 2 1 1 966 5
9 1 0 0 1 1 0 0 2 2 987
0 1 2 3 4 5 6 7 8 9

Code

- Báo cáo kết quả của mô hình

```
from sklearn.metrics import classification_report
target_names = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
print(classification_report(Y_test, Y_pred, target_names=target_names))
```

	precision	recall	f1-score	support
0	0.99	0.99	0.99	980
1	1.00	0.99	0.99	1135
2	0.99	0.99	0.99	1032
3	0.99	0.99	0.99	1010
4	0.99	1.00	0.99	982
5	0.99	0.99	0.99	892
6	0.99	0.99	0.99	958
7	0.99	0.99	0.99	1028
8	0.98	0.99	0.99	974
9	0.99	0.98	0.99	1009
accuracy			0.99	10000
macro avg	0.99	0.99	0.99	10000
weighted avg	0.99	0.99	0.99	10000

Tóm tắt

- Workflow của deep learning
- Vẽ kiến trúc của model (graphviz)
- Biểu đồ huấn luyện (matplotlib)
- Đánh giá mô hình và diễn giải mô hình cuối cùng (confusion matrix)