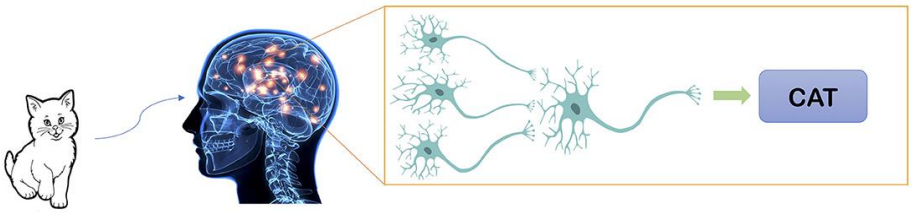
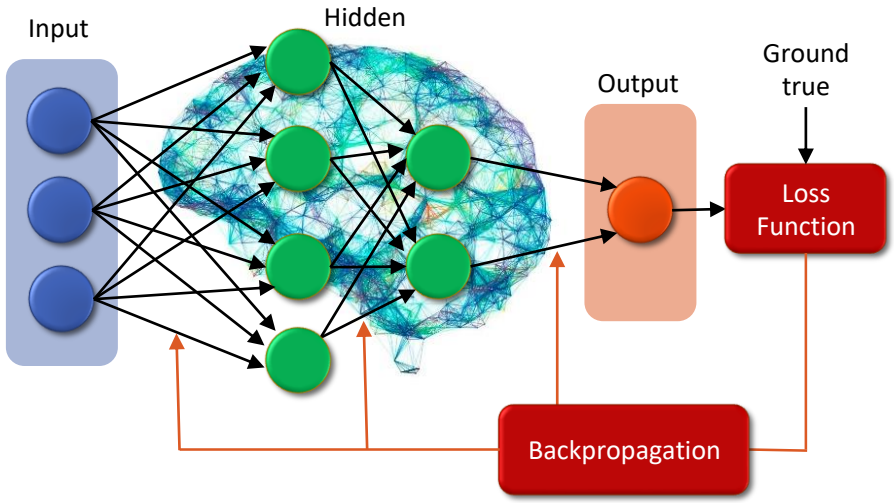
DEEP LEARNING

THƯ VIỆN KERAS





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

Nội dung

- Giới thiệu Keras
- Cài đặt Keras
- Import thư viện
- Sử dụng Keras trong 30 phút
- Tổng quan Keras
- Các module trong Keras

Giới thiệu Keras

- Keras: Keras là một **high-level API** về **neural network** được viết bằng Python và có thể chạy trên nhiều backend
  - TensorFlow
  - Theano
  - CNTK

- Mục tiêu
  - Cho phép người sử dụng thực nghiệm nhanh
  - Cho phép người dùng chuyển từ ý tưởng sang kết quả với nhanh nhất

- Tham khảo
  - https://keras.io/

Cài đặt Keras

- Python: Python ≥ 3.6
- Một số thư viên hữu dụng: numpy, scipy, scikit-learn, pillow, h5py, opencv-python, …
- Cuda
  - Cuda ToolKit 9.0 và
  - cuDNN (nếu chạy Keras trên GPU)
- Cài đặt TensorFlow/Keras

```
pip install tensorflow
```

Import thư viện

```
import numpy as np
import tensorflow as tf
from tensorflow import keras

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, Dense
from tensorflow.keras.layers import Activation, Flatten

from tensorflow.keras import backend as K
```

Chú ý: Để tiện cho trình bày, chúng ta dùng tf thay cho chữ tensorflow trong câu lệnh import

Ví dụ: thay vì viết

```
from tensorflow.keras.models import Sequential
```

Ta sẽ viết

```
from tf.keras.models import Sequential
```

Sử dụng Keras trong 30 phút

- Load dữ liệu

```
from tensorflow.keras import datasets

(x_train, y_train), (x_test, y_test) = datasets.mnist.load_data()
```

Sử dụng Keras trong 30 phút

- Tạo mô hình với lớp Sequential()

```
from tensorflow.keras import models
model = models.Sequential()
```

- Thêm các tầng với hàm .add()

```
model.add(Dense(units=64, activation='relu',
                input_dim=100))
model.add(Dense(units=10, activation='softmax'))
```

- Cấu hình quá trình học với hàm .compile()

```
model.compile(loss=''categorical_crossentropy,
              optimizer='sgd',
              metrics=['accuracy'])
```

Sử dụng Keras trong 30 phút

- Huấn luyện mô hình với hàm .fit()

```
EPOCHS = 20
h = model.fit(x_train, y_train,
              epochs=EPOCHS, batch_size=32)
```

- Vẽ biểu đồ huấn luyện

```
plt.plot(h.history['loss'])
plt.plot(h.history['val_loss'])
plt.title('…')
plt.xlabel('loss')
plt.ylabel('epoch')
plt.legend(['train','test'], loc='upper left')
plt.show()
```

- Đánh giá mô hình

```
lost, accuracy = model.evalute(x_test, y_test, batch_size=32)
```

Tổng quan Keras

- Module gốc Keras **có 3 phần**
  - Các **modules**
  - Các **lớp**
  - Các **Hàm**

- **Lớp**

```
from tensorflow.keras import Model
from tensorflow.keras import Sequential
```

- **Hàm**

```
Input(shape=None, batch_shape=None)
```

  - Dùng tạo keras **tensor**

Tổng quan Keras

- **Các modules**

```
from tf.keras import datasets

from tf.keras import models
from tf.keras import layers

from tf.keras import activations
from tf.keras import losses
from tf.keras import optimizers
from tf.keras import metrics

from tf.keras import regularizers
from tf.keras import applications

from tf.keras import utils
from tf.keras import preprocessing
```

```
from tf.keras import objectives
from tf.keras import initializers
```

Chú ý: những chổ tf được thay bằng tensorflow

Tổng quan Keras – Module datasets

- 7 datasets có sẵn

```
from tf.keras.datasets import boston_housing
from tf.keras.datasets import cifa
from tf.keras.datasets import cifa10
from tf.keras.datasets import cifa100
from tf.keras.datasets import mnist
from tf.keras.datasets import fashion_mnist
from tf.keras.datasets import imdb
from tf.keras.datasets import reuters
```

- **Load** dữ liệu

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

Module losses

```
from tf.keras.losses import BinaryCrossentropy
from tf.keras.losses import CategoricalCrossentropy
from tf.keras.losses import MeanAbsoluteError
from tf.keras.losses import MeanSquaredError
from tf.keras.losses import SparseCategoricalCrossentropy
```

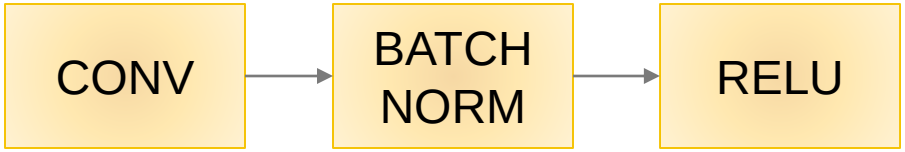
Module utils

- Một số hàm trong **utils**

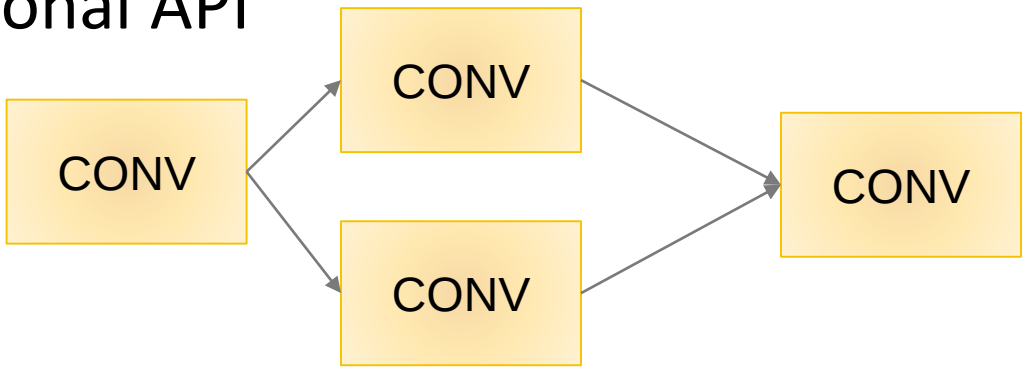
```
y = utils.to_categorical(y, num_classes=...)
x = utils.normalize(x, axis=-1, order=2) # L2 norm
utils.plot_model(model, to_file="model.png")
utils.multi_gpu_model(model, gpus=None, cpu_merge=True)
```

Module models

- Sequential API



- Functional API



- Model subclassing

```
class MyNetwork(Model):

...
```

Module models

- **Có 2 cách tạo model**
  - Sequential
  - Functional API: thông qua lớp **Model**
  - Thừa kết: **thông qua lớp Model**

```
model = models.Sequential()
model = models.Sequential(layers=[...])
```

```
model = models.Model(inputs=..., outputs=...)
```

```
class MyNetwork(Model):
```

Module models

- Một số hàm trong lớp Sequential, Model

```
model.add() # chỉ có trong Sequential
model.summary()

model.compile(optimizer, loss=None, metrics=[])

model.fit(x, y, batch_size)
model.fit_generator(generator)

loss, metric = model.evaluate(x, y, batch_size)
model.evaluate_generator(generator)

pred = model.predict(x, batch_size=None)
model.predict_generator(generator)

model.save(filepath)
```

Module models – lưu và nạp model

- **Lưu** toàn bộ model
  - Lưu architecture, weights, optimizer, state of the optimiz, learning rate, loss, …

```
model.save(filepath="mode_name.h5")
```

```
models.save(model, filepath="mode_name.h5")
```

- **Nạp** toàn bộ model

```
model = models.load_model(filepath="mode_name.h5")
```

- Nhận xét
  - Save model có thể dùng biến model hay module models
  - Load model dùng module models

Module models – lưu và nạp model

- **Lưu** kiến trúc model

```
s = model.to_json()
```

```
s = model.to_yaml()
```

- **Nạp** kiến trúc model

```
model = models.model_from_json(s)
```

```
model = models.model_from_yaml(s)
```

Module models – lưu và nạp model

- **Lưu** trọng số của model

```
model.save_weights(filepath="weights_name.h5")
```

- **Nạp** trọng số của model

```
model.load_weights(filepath="weights_name.h5")
```

- Nhận xét: dùng hàm trong biến model

Module layers

- Các lớp trong layers

```
from tf.keras.layers import Conv2D, Convolution2D

from tf.keras.layers import MaxPool2D, MaxPooling2D
from tf.keras.layers import AvgPool2D, AveragePooling2D
from tf.keras.layers import ZeroPadding2D

from tf.keras.layers import Flatten, Dense

from tf.keras.layers import Activation

from tf.keras.layers import Dropout, BatchNormalization
```

Module layers

- Convolutional layer

```
layers.Conv2D(filters, kernel_size, strides=(1,1),
              padding='valid', activation='...',
              input_shape, ...)
```

  - filters: số filters (32, 64, 128, 256, 512, …)
  - kernel_size: số lượng kernel
  - strides: mặc định (1,1)
  - padding: valid, same
  - activation: relu, sigmoid, …
  - input shape: (x, y)
  - …

Module layers

- Tạo Convolutional layer
  - Tầng đầu tiên

```
# Image 320x320x3
input_shape=(224, 224, 3)

model.add(layers.Conv2D(48, (3, 3), activation='relu',
          input_shape= input_shape))
```

  - Tầng phía sau

```
model.add(layers.Conv2D(48, (3, 3), activation='relu'))
```

Module layers

- Pooling
  - Tham số: pooling size

```
model.add(layers.MaxPooling2D(pool_size=(2, 2)))
```

- Dense layer
  - Tham số: output size

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

- Dropout layer
  - Tham số: Xác suất drop out

```
model.add(layers.Dropout(0.5))
```

Module layers

- Compiling, Training, and Evaluate
  - Compiling: loss function, optimizer, metrics

```
model.compile(loss='mean_squared_error', optimizer='adam',
              metrics=[])
```

  - Training: training data, batch size, epochs, validation data

```
model.fit(X_train, X_train, batch_size=32, epochs=10,
          validation_data=(x_val, y_val))
```

  - Evaluate

```
loss, metric = model.evaluate(x_test, y_test, batch_size=32)
```

Module preprocessing

- Có 3 module tiền xử lý

```
from tf.keras.preprocessing.sequence import image
from tf.keras.preprocessing.sequence import sequence
from tf.keras.preprocessing.sequence import text
```

- Tiền xử lý sequence và text

```
from tf.keras.preprocessing.sequence import make_sampling_table
from tf.keras.preprocessing.sequence import pad_sequences
from tf.keras.preprocessing.sequence import skipgrams
from tf.keras.preprocessing.sequence import TimeseriesGenerator

from tf.keras.preprocessing.text import one_host
from tf.keras.preprocessing.text import hashing_trick
from tf.keras.preprocessing.text import text_to_word_sequence
```

Module preprocessing

- Tiền xử lý ảnh (image)
  - Một số hàm

```
from tf.keras.preprocessing.image import img_to_array
from tf.keras.preprocessing.image import array_to_img

from tf.keras.preprocessing.image import load_img
from tf.keras.preprocessing.image import save_img
```

  - Lớp

```
from tf.keras.preprocessing.image import ImageDataGenerator
```

Đọc ảnh từ file và tiền xử ảnh

- Dùng lớp `ImageDataGenerator` để tiền xử và đọc ảnh từ file

```
dataGen = ImageDataGenerator(rescale = 1./255)

generator = dataGen.flow_from_directory(
    directory,
    target_size=(128, 128),
    batch_size = 32,
    class_mode = 'binary'
)
```

  - rescale = 1./255, chuyển ảnh về [0,1]
  - dictionary: thư mục chứa ảnh, mỗi thư mục con là 1 loại ảnh
  - target_size: kích thước ảnh được chuyển về
  - class_mode: binary, category

Augmentation

- Dùng lớp `ImageDataGenerator` để tiền xử và đọc ảnh từ file

```
dataGen = ImageDataGenerator(rescale = 1./255)

generator = dataGen.flow_from_directory(
    directory,
    target_size=(128, 128),
    batch_size = 32,
    class_mode = 'binary'
)
```

- rescale = 1./255, chuyển ảnh về [0,1]
- dictionary: thư mục chứa ảnh, mỗi thư mục con là 1 loại ảnh
- target_size: kích thước ảnh được chuyển về
- class_mode: binary, category