

## Keras - Software Analysis


### About The Software

Keras คือ Open Source Deep learning API สำหรับการพัฒนา deep learning model ที่เขียนขึ้นในภาษา Python โดยทำงานอยู่บน libraries ที่มีชื่อว่า Tensorflow และ Theano

ในการนำ Keras ไปประยุกต์ใช้งาน สามารถทำได้ด้วยการนำผลลัพธ์ของการ train deep learning model ไปใช้ในการต่อยอดทำแอปพลิเคชันต่างๆได้ เช่น

- สร้างเว็บแอปพลิเคชันสำหรับการจำแนกประเภทสิ่งต่างๆ เช่น คน สัตว์ สิ่งของ
- สร้าง feature ย่อยๆสำหรับแอปพลิเคชัน โดยแอปพลิเคชันชื่อดังที่มีการใช้งานสามารถยกตัวอย่างได้เช่น Netflix, Uber, Yelp, Instacart, Zocdoc, Square

# I am



## Golden Retriever

### Results

Golden Retriever	99%
Labrador Retriever	1%
Kuvasz	0%

[< Back](#)

### Golden Retriever

โกลเด้นรีทรีฟเวอร์ มีถิ่นกำเนิดในประเทศสกอตแลนด์ มีขนยาว มีสีขนเหลืองอ่อน เหลืองทอง และเหลืองเข้ม โดยทั่วไปแล้ว เพศผู้มีความสูง ประมาณ 23- 24 นิ้ว และเพศเมียมีความสูงประมาณ 21.5-22.5 นิ้ว แต่เดิมนั้นสายพันธุ์นี้ได้ปรับปรุงคุณลักษณะบางอย่างเพื่อในงานของนายพราน เช่น มีรูปร่างสมส่วน คล่องตัว เชื่อฟังและพินิจขึ้นเหนือโดยให้เข้าเพียงเล็กน้อย เป็นต้น ไม่มีปรากฏว่าสุนัขสายพันธุ์นี้มีความก้าวร้าวแต่อย่างใด โกลเด้นรีทรีฟเวอร์ เป็นสุนัขที่มีความเฉลียวฉลาดมากจนสามารถนำมาฝึกเป็นสุนัขตำรวจ เนื่องจากมีประสาทสัมผัสที่ดีทั้งในด้านของการฟังเสียง การดมกลิ่นสะกดรอย นอกจากนี้ยังมีสายตาสั้นเฉียบคมและแม่นยำ (ที่มา: Wikipedia)

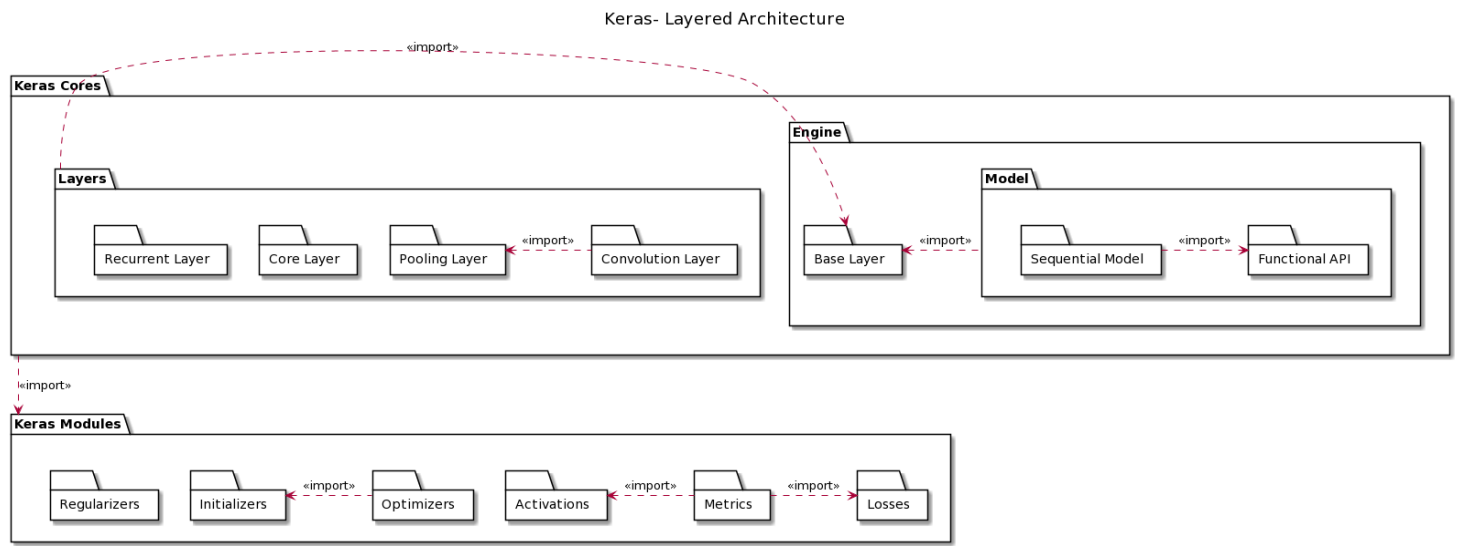
ตัวอย่างเว็บแอปฯ : เว็บแอปพลิเคชันทำนายสายพันธุ์สุนัข

## Architectural Styles

Architectural Styles ของ Keras อยู่ในรูปแบบของ Layered โดยใน Keras Cores จะมี 2 package ที่สำคัญ ซึ่งก็คือ Layers และ Engine โดยที่ package Layers จะมีการใช้งานตัว package Base Layer ที่อยู่ใน package Engine และ package Model ใน Engine ก็มีการใช้งานตัว Base Layer เช่นเดียวกัน

ในส่วนของ Keras Cores ก็มีการใช้งาน Keras Modules อื่นๆด้วย เช่น Activations, Optimizers, Losses, Initializers, Regularizers, Metrics เป็นต้น

UML Diagram



## Quality Attributes

Quality Attributes ของ Keras มีดังนี้

จุดเด่น

### 1. Performance

เหตุผลที่เลือก:

เพราะ Keras เป็น API ที่ใช้ในงาน deep learning ดังนั้นจึงมีความต้องการให้การ train model ใช้ระยะเวลาสั้นลงและสามารถทำงานได้อย่างมีประสิทธิภาพ

วิธีการ:

“Mixed precision training is the use of lower-precision operations (float16 and bfloat16) in a model during training to make it run faster and use less memory. Using mixed precision can improve performance by more than 3 times on modern GPUs and 60% on TPUs.”

Keras มีเทคนิคที่ช่วยเพิ่มความเร็วในการ train model โดยใช้วิธีการ mixed precision training ซึ่งเป็นการใช้ floating-point ชนิด 16 bits และ 32 bits ผสมกัน โดยในระหว่างการ train จะมีการเปลี่ยนชนิดข้อมูล จาก float32 เป็น float16 หรือ bfloat16 ส่งผลให้มีการใช้ Memory น้อยลงทำให้ประสิทธิภาพสูงขึ้น 3 เท่าบน GPUs รุ่นใหม่ และสูงขึ้น 60% บน TPUs (หน่วยประมวลผลของ Google ที่ใช้สำหรับงาน AI โดยเฉพาะ)

Source: [https://keras.io/api/mixed\\_precision/](https://keras.io/api/mixed_precision/)

[https://www.tensorflow.org/guide/mixed\\_precision](https://www.tensorflow.org/guide/mixed_precision)

### 2. Scalability

เหตุผลที่เลือก:

เพราะ Keras เป็น API ที่มีการใช้งานในกลุ่มผู้ใช้งานที่หลากหลาย ตั้งแต่ระดับผู้ใช้งานทั่วไปจนถึงระดับอุตสาหกรรม หรืองานวิจัย ดังนั้นจึงต้องมีการออกแบบให้รองรับอุปกรณ์ประมวลผลตั้งแต่จำนวนน้อยไปจนถึงอุปกรณ์ประมวลผลจำนวนมากถึงหนึ่งพันตัว

วิธีการ:

“Keras is scalable. Using the TensorFlow DistributionStrategy API, which is supported natively by Keras, you easily can run your models on large GPU clusters (up to thousands of devices) or an entire TPU pod, representing over one exaFLOPs of computing power”

Keras มีการรองรับการเพิ่มหรือลดอุปกรณ์ประมวลผล เช่น GPU หรือ TPU ในการ train model ได้ด้วยวิธีการทำ Multi-GPU and distributed training โดยจะมี 2 รูปแบบคือ Data parallelism คือการที่อุปกรณ์ประมวลผลแต่ละตัวใช้ model เดียวกันแต่แบ่งข้อมูลกัน train เป็นส่วนๆแล้วนำมาผลลัพธ์มารวมกัน ส่วน Model parallelism คือการแบ่ง model เป็นส่วนๆไปในอุปกรณ์ประมวลผลแต่ละตัวแต่ใช้ข้อมูลชุดเดียวกันในการ train

Source:

[https://keras.io/why\\_keras/#keras-has-strong-multigpu-amp-distributed-training-support](https://keras.io/why_keras/#keras-has-strong-multigpu-amp-distributed-training-support)

แนวทาง Multi-GPU and distributed training :

[https://keras.io/guides/distributed\\_training](https://keras.io/guides/distributed_training) และ

[https://keras.io/getting\\_started/faq/#how-can-i-train-a-keras-model-on-multiple-gpus-on-a-single-machine](https://keras.io/getting_started/faq/#how-can-i-train-a-keras-model-on-multiple-gpus-on-a-single-machine)

### 3. Portability

เหตุผลที่เลือก:

เพราะในปัจจุบันมีผู้ใช้งานบริการที่เกี่ยวข้องกับ AI บนหลากหลาย platform เช่น Web Browser, Application บน Smartphone และ Embedded system

วิธีการ:

Keras ถูกออกแบบให้ model ที่เป็นผลลัพธ์สามารถนำไปใช้งานได้บนหลากหลาย platform เช่น Android, iOS, WebServer โดยสามารถใช้วิธี save model ที่ train แล้วนำไป deploy จากนั้นก็ให้ service ของแต่ละ platform นั้นเรียกใช้ API

ตัวอย่างที่สามารถนำ model ไป deploy ได้

- On server via a Python runtime or a Node.js runtime
- On server via TFX / TF Serving
- In the browser via TF.js
- On Android or iOS via TF Lite or Apple's CoreML
- On Raspberry Pi, on Edge TPU, or another embedded system

Source:

[https://keras.io/why\\_keras/#keras-makes-it-easy-to-turn-models-into-products](https://keras.io/why_keras/#keras-makes-it-easy-to-turn-models-into-products)

วิธี import Keras Model ใน TF.js:

[https://www.tensorflow.org/js/tutorials/conversion/import\\_keras](https://www.tensorflow.org/js/tutorials/conversion/import_keras)

จุดด้อย

### 1. Testability

ในมุมมองของผู้ใช้งานนั้น การทำ Debug และการ Test นั้นเป็นไปได้ด้วยความยากลำบากเมื่อเกิดข้อผิดพลาดขึ้น เนื่องจาก keras ทำงานในลักษณะของ High-Level API ซึ่งอาจทำให้เราอาจไม่ทราบสาเหตุที่แท้จริงของต้นตอปัญหาได้

ข้อเสนอแนะ:

- ควรมีการเพิ่ม assertions หรือ raising exceptions ใน source code ให้มากขึ้นและมีความละเอียดยิ่งขึ้นเพื่อใช้ในการแสดง error ให้แก่ผู้ใช้งานเพื่อทำให้รู้ถึงต้นตอของปัญหาได้ง่ายยิ่งขึ้น

Source: <https://www.edureka.co/blog/keras-vs-tensorflow-vs-pytorch/> และ

<https://www.geeksforgeeks.org/keras-vs-pytorch/> และ

<https://analyticsindiamag.com/pytorch-vs-keras-who-suits-you-the-best/>

## Design Pattern

### 1. Adapter

ผู้พัฒนาต้องการให้ชนิดข้อมูลที่หลากหลาย เช่น python list, numpy array, generator และอื่นๆ สามารถแปลงเป็น tf.data.Dataset และทำงานกับ Keras ได้อย่างถูกต้องและยังเป็นไปตามหลักการ SRP และ OCP

มีการ subclass DataAdapter เพื่อรองรับประเภทข้อมูลที่แตกต่างกันโดยแต่ละ class ที่ subclass มาเช่น TensorLikeDataAdapter, CompositeTensorLikeDataAdapter มีการใช้งาน tf.data.Dataset เพื่อแปลงข้อมูลชนิดต่างๆเป็น tf.data.Dataset

Source: -

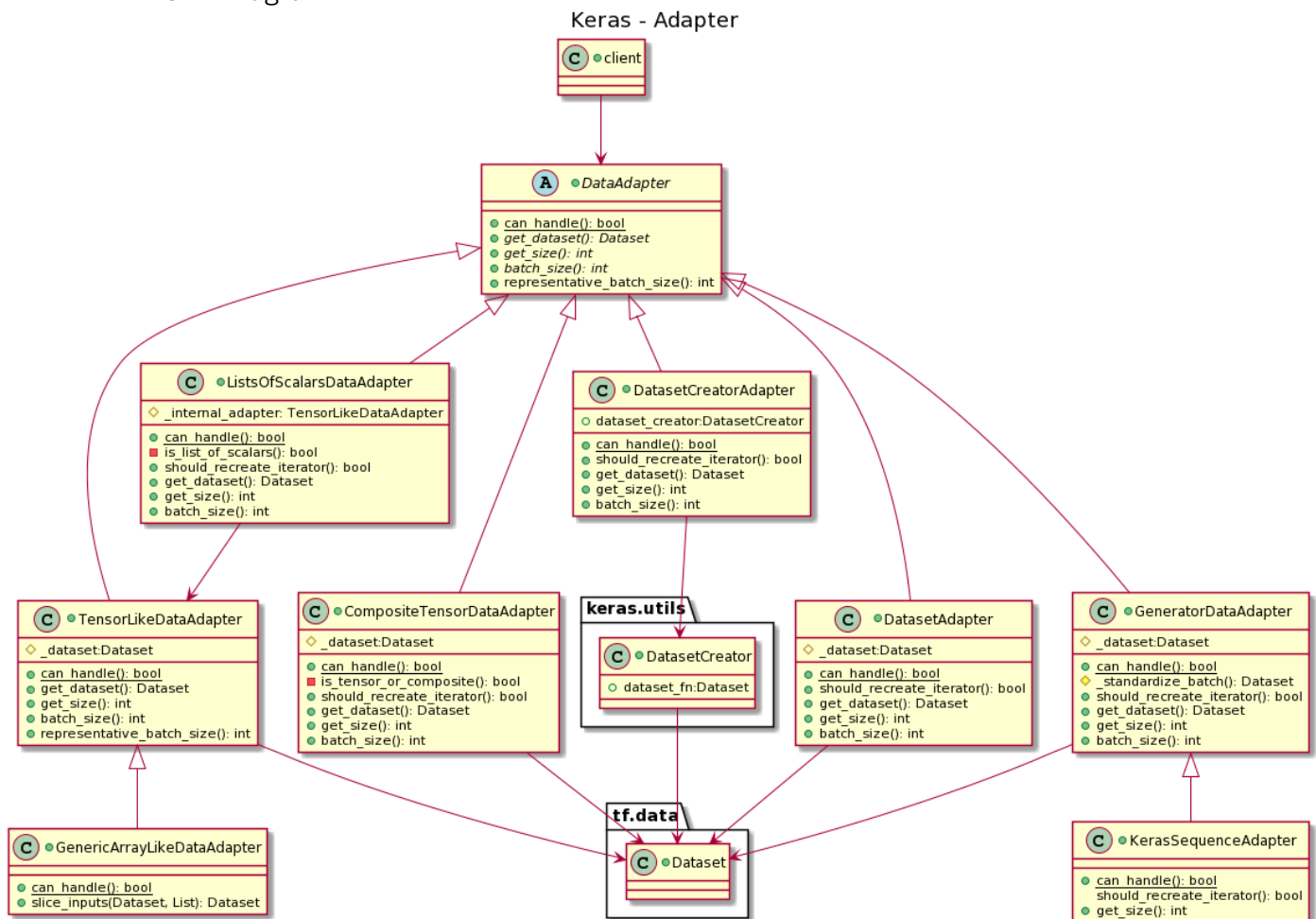
[https://github.com/keras-team/keras/blob/v2.6.0/keras/engine/data\\_adapter.py](https://github.com/keras-team/keras/blob/v2.6.0/keras/engine/data_adapter.py)

บรรทัดที่ 40 - 988 class DataAdapter, TensorLikeDataAdapter, ...

- [https://github.com/keras-team/keras/blob/v2.6.0/keras/utils/dataset\\_creator.py](https://github.com/keras-team/keras/blob/v2.6.0/keras/utils/dataset_creator.py)

บรรทัดที่ 23 - 100 class DatasetCreator

UML Diagram



Source:

[https://raw.githubusercontent.com/pholpaphankorn/keras\\_softarch\\_assignment/4f07c88949ce1932bb09c835296df4cafc61a28e/uml\\_diagram/image/svg/keras\\_data\\_adapter.svg](https://raw.githubusercontent.com/pholpaphankorn/keras_softarch_assignment/4f07c88949ce1932bb09c835296df4cafc61a28e/uml_diagram/image/svg/keras_data_adapter.svg)

### ตัวอย่างการใช้งาน



```
1 numpy_input = np.zeros((5, 5))
2 print(type(numpy_input))
3 numpy_input
```



```
<class 'numpy.ndarray'>
array([[0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.]])
```

```
[14] 1 adapter = data_adapter.TensorLikeDataAdapter(numpy_input, batch_size=1)
      2 for a in adapter.get_dataset():
      3     print(type(a))
      4     print(a)
```

```
<class 'tensorflow.python.framework.ops.EagerTensor'>
tf.Tensor([[0. 0. 0. 0. 0.]], shape=(1, 5), dtype=float32)
<class 'tensorflow.python.framework.ops.EagerTensor'>
tf.Tensor([[0. 0. 0. 0. 0.]], shape=(1, 5), dtype=float32)
<class 'tensorflow.python.framework.ops.EagerTensor'>
tf.Tensor([[0. 0. 0. 0. 0.]], shape=(1, 5), dtype=float32)
<class 'tensorflow.python.framework.ops.EagerTensor'>
tf.Tensor([[0. 0. 0. 0. 0.]], shape=(1, 5), dtype=float32)
<class 'tensorflow.python.framework.ops.EagerTensor'>
tf.Tensor([[0. 0. 0. 0. 0.]], shape=(1, 5), dtype=float32)
```

จะได้ว่าชนิดของข้อมูลเปลี่ยนจาก numpy array เป็น EagerTensor

Source:

[https://colab.research.google.com/drive/1Kk44xpyfD8xl49gWOkhhly\\_FIWZGt2dc?usp=sharing](https://colab.research.google.com/drive/1Kk44xpyfD8xl49gWOkhhly_FIWZGt2dc?usp=sharing)

## 2. Composite

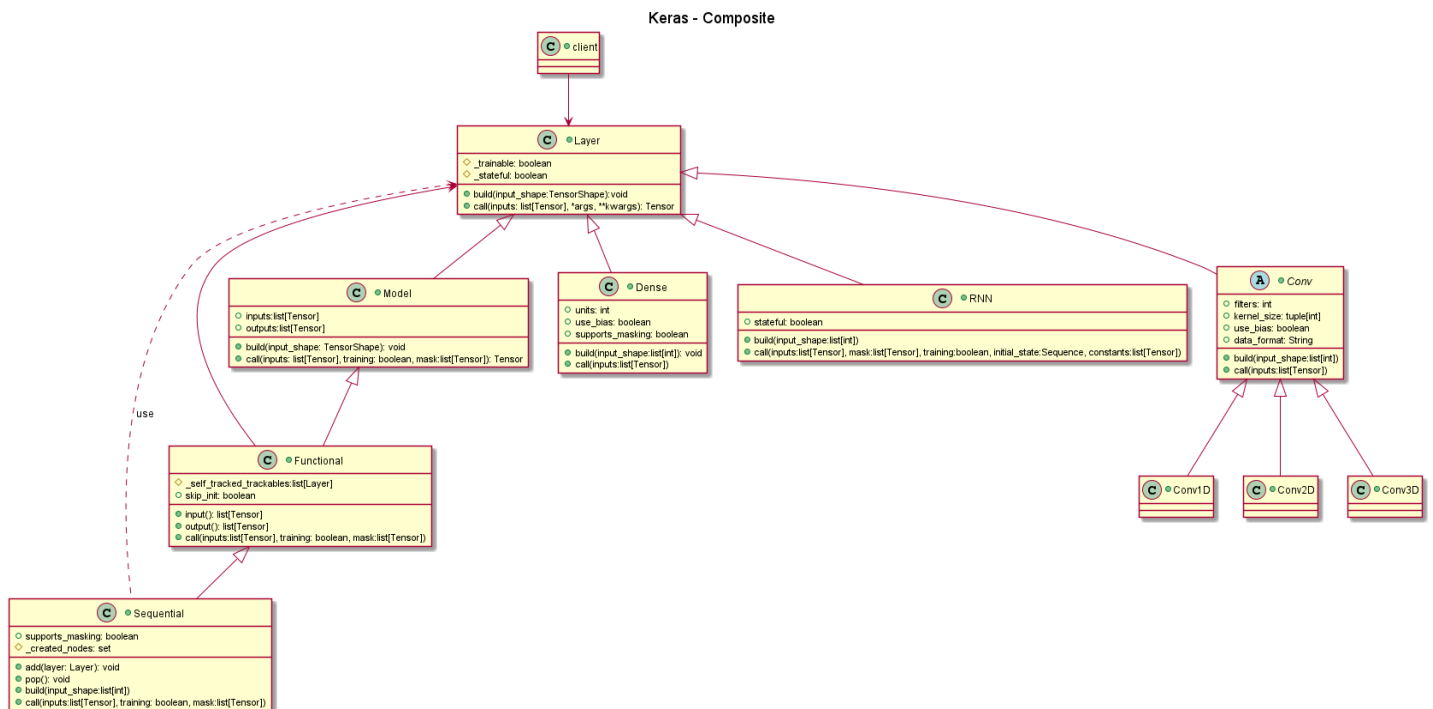
เพราะผู้พัฒนาต้องการให้ instance ของ Sequential สามารถเก็บ instance ของ Sequential ด้วยตัวเอง หรือ เก็บ instance ของ Layer ได้ ทำให้ง่ายต่อ client code ในการเรียกใช้งาน เมื่อมีการ train model โดยเมื่อ client code เรียกใช้งานเพียงแค่ method เดียว ตัว Sequential ก็จะ delegate หน้าที่ไปให้ sub-element ของตัวมันเองเพื่อทำงานต่อไปโดยใช้หลักการของ Polymorphism และ Recursion

นอกจากนั้นสามารถเพิ่ม Layer ใหม่ๆที่มีการทำงานใกล้เคียงกับ Layer อื่นๆ โดยไม่ทำให้ code ที่มีมาก่อนหน้านี้ทำงานผิดพลาดซึ่งเป็นไปตามหลักการ OCP(Open/Closed Principle)

Source:

- <https://github.com/keras-team/keras/blob/v2.6.0/keras/engine/sequential.py#L41-L467>  
บรรทัด 41-467 class Sequential
- <https://github.com/keras-team/keras/blob/v2.6.0/keras/engine/training.py>  
บรรทัด 103-3068 class Model
- [https://github.com/keras-team/keras/blob/v2.6.0/keras/engine/base\\_layer.py](https://github.com/keras-team/keras/blob/v2.6.0/keras/engine/base_layer.py)  
บรรทัด 84-3113 class Layer
- <https://github.com/keras-team/keras/tree/v2.6.0/keras/layers>  
Folder ที่เก็บ class Dense, Recurrent, Conv

UML Diagram



Source:

[https://raw.githubusercontent.com/pholpaphankorn/keras\\_softarch\\_assignment/main/uml\\_diagram/image/svg/keras\\_Composite.svg](https://raw.githubusercontent.com/pholpaphankorn/keras_softarch_assignment/main/uml_diagram/image/svg/keras_Composite.svg)

## ตัวอย่างการใช้งาน

```
img_width, img_height = 224, 224
batch_size = 256
n_class = 3
conv_base = InceptionV3(weights='imagenet',
                        include_top=False,
                        input_shape=(img_width, img_height, 3),
                        pooling="avg"
                        ) # 3 = number of channels in RGB pictures
conv_base.trainable = False

model_1 = tf.keras.Sequential()
model_1.add(conv_base)
model_1.add(layers.Dense(4))

model_2 = tf.keras.Sequential()
model_2.add(model_1)
model_2.add(layers.Dense(16))

model_3 = tf.keras.Sequential()
model_3.add(layers.Dense(16))
model_3.add(layers.Dense(8))

model_4 = tf.keras.Sequential()
model_4.add(model_2)
model_4.add(model_3)
model_4.add(layers.Dense(32))
model_4.add(layers.Dense(32))
```



```
model_2.summary()

Model: "sequential_5"
Layer (type)                Output Shape                Param #
=====
sequential_4 (Sequential)    (None, 4)                   21810980
dense_7 (Dense)              (None, 16)                  80
=====
Total params: 21,811,060
Trainable params: 8,276
Non-trainable params: 21,802,784

[18] model_4.summary()

Model: "sequential_7"
Layer (type)                Output Shape                Param #
=====
sequential_5 (Sequential)    (None, 16)                  21811060
sequential_6 (Sequential)    (None, 8)                   408
dense_10 (Dense)             (None, 32)                  288
dense_11 (Dense)             (None, 32)                  1056
=====
Total params: 21,812,812
Trainable params: 10,028
Non-trainable params: 21,802,784
```

จะได้ว่า model\_4 มี model\_2 และ model\_3 อยู่ข้างใน และ model\_2 มี model\_1 อยู่ข้างใน

Source:

<https://colab.research.google.com/drive/1FWjmyaWCcf7fJTob9o8g8YpWlFsvoG4q?usp=sharing>

### 3. Decorator

เพราะผู้พัฒนาต้องการให้ Layer สามารถเพิ่มความสามารถได้ไปในทิศทางใดทิศทางหนึ่ง เช่น ความสามารถในการทำให้ input ที่รับมาสามารถแบ่งเป็นช่วงของเวลาได้ ซึ่งถูก implement ใน TimeDistributed wrapper หรือ ความสามารถในการทำให้ Layer สามารถที่จะส่งค่าได้ทั้งสอง ทิศทางซึ่งจะใช้ใน RNNs (Recurrent Neural Network) ซึ่งโดยปกติแล้ว Layer จะสามารถทำได้แค่ ส่งค่าแบบ Forward propagation และปรับ weight ผ่าน Back propagation ความสามารถที่จะส่ง ค่าได้ทั้งสองทิศทางก็จะถูก implement อยู่ใน Bidirectional wrapper ซึ่งสามารถ wrap layer ที่ เป็น RNN หรือ LSTM(Long Short Term Memory) หรือ GRU(Gated Recurrent Unit) หรือ อาจ จะเป็น Layer ปกติก็ได้แต่ต้องเป็นไปตามข้อกำหนดก็คือต้องรับ 3D input ได้และมีความสามารถ เหมือน RNN

นอกจากนี้ ผู้พัฒนายังสามารถเพิ่มความสามารถของ Layer ไปได้เรื่อยๆโดยที่ไม่ต้อง subclass ที่ concrete object ของ Layer โดยตรง และ สามารถแบ่ง class ซึ่งมีหน้าที่ความ รับผิดชอบหรือความสามารถหลายๆอย่างออกไปเป็น class ที่มีความรับผิดชอบเพียงอย่างเดียวซึ่ง เป็นไปตามหลักการ SRP(Single-Responsibility Principle)

Source: - <https://github.com/keras-team/keras/blob/v2.6.0/keras/layers/wrappers.py>

บรรทัด 34-800

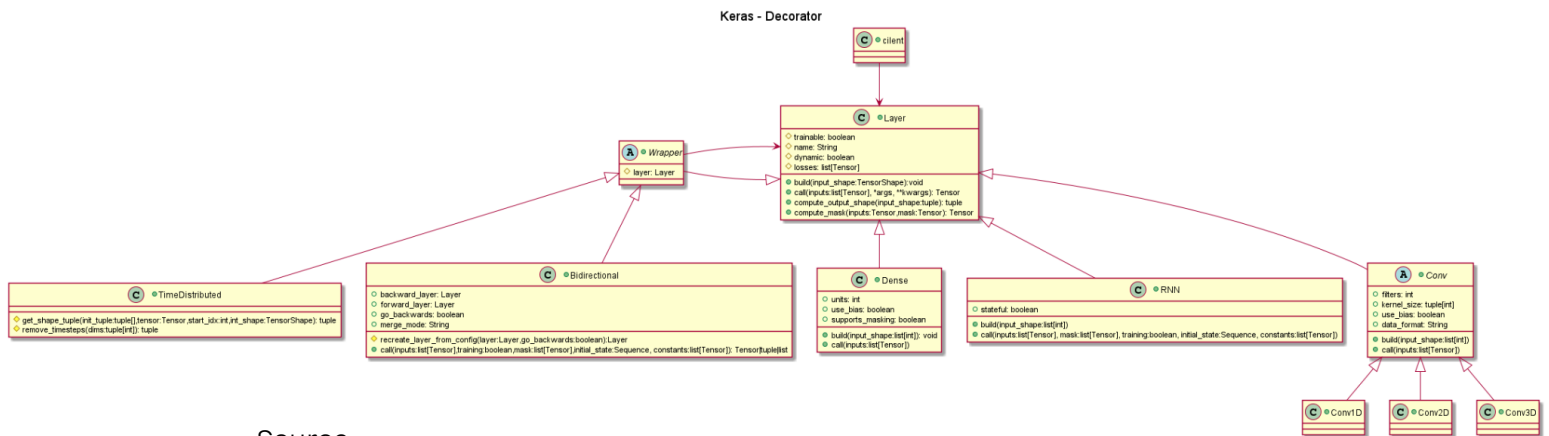
- [https://github.com/keras-team/keras/blob/v2.6.0/keras/engine/base\\_layer.py](https://github.com/keras-team/keras/blob/v2.6.0/keras/engine/base_layer.py)

บรรทัด 84-3113 class Layer

- <https://github.com/keras-team/keras/tree/v2.6.0/keras/layers>

Folder ที่เก็บ class Dense, Recurrent, Conv

UML Diagram



Source:

[https://raw.githubusercontent.com/pholpaphankorn/keras\\_softarch\\_assignment/4f07c88949ce1932bb09c835296df4cafc61a28e/uml\\_diagram/image/svg/keras\\_wrapper.svg](https://raw.githubusercontent.com/pholpaphankorn/keras_softarch_assignment/4f07c88949ce1932bb09c835296df4cafc61a28e/uml_diagram/image/svg/keras_wrapper.svg)

## ตัวอย่างการใช้งาน

```
inputs = tf.keras.Input(shape=(10, 128, 128, 3))
conv_2d_layer = tf.keras.layers.Conv2D(64, (3, 3))
outputs = tf.keras.layers.TimeDistributed(conv_2d_layer)(inputs)
outputs.shape
TensorShape([None, 10, 126, 126, 64])
```

มีการหุ้ม Conv2D ด้วย TimeDistributed เพื่อเพิ่มความสามารถ

```
tf.keras.layers.Bidirectional(
    layer, merge_mode="concat", weights=None, backward_layer=None, **kwargs
)
model = Sequential()
model.add(Bidirectional(LSTM(10, return_sequences=True), input_shape=(5, 10)))
model.add(Bidirectional(LSTM(10)))
model.add(Dense(5))
model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy', optimizer='rmsprop')
```

```
# With custom backward layer
model = Sequential()
forward_layer = LSTM(10, return_sequences=True)
backward_layer = LSTM(10, activation='relu', return_sequences=True,
                        go_backwards=True)
model.add(Bidirectional(forward_layer, backward_layer=backward_layer,
                        input_shape=(5, 10)))
model.add(Dense(5))
model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy', optimizer='rmsprop')
```

มีการหุ้ม LSTM ด้วย Bidirectional เพื่อเพิ่มความสามารถ

Source: [https://keras.io/api/layers/recurrent\\_layers/time\\_distributed/](https://keras.io/api/layers/recurrent_layers/time_distributed/)  
[https://keras.io/api/layers/recurrent\\_layers/bidirectional/](https://keras.io/api/layers/recurrent_layers/bidirectional/)

#### 4. Iterator

เพราะผู้พัฒนาต้องการให้ client code ภายใน keras สามารถเข้าถึงข้อมูลที่ถูกเก็บไว้ใน collections ต่างๆได้โดยไม่ต้องการซ่อนความซับซ้อนต่างๆภายใน collection นั้นและรวมถึง implementation ด้วย เช่น

- การ iterate ใน directory เพื่อดึงข้อมูลรูปภาพจาก class ต่างๆซึ่งถูกเก็บอยู่ภายใน folder ที่แตกต่างกัน ซึ่งถูก implement ใน DirectoryIterator
- การ iterate ภายใน numpy.array เพื่อดึงข้อมูลรูปภาพใน format ของ numpy.array ซึ่งถูก implement ใน NumpyArrayIterator
- การ iterate ใน pandas.DataFrame เพื่อดึงข้อมูลรูปภาพที่ถูกเก็บเป็น path ไว้ซึ่งถูก implement ใน DataFrameIterator

การที่ผู้พัฒนานำ Iterator pattern มาใช้ก็ทำให้ keras สามารถปรับเปลี่ยน Iterator ได้ตรงตามกับข้อมูลที่ user ป้อนเข้ามาและถ้าจะเพิ่ม collection และ iterator ใหม่เข้ามาใน source code ก็เพิ่มได้โดยง่ายโดยที่ไม่ทำให้ code ในปัจจุบันมีปัญหาและยังเป็นการแยกความรับผิดชอบระหว่าง iterator กับ collection ซึ่งเป็นไปตามหลักการ SRP(Single-Responsibility Principle) และ OCP(Open/Closed Principle)

Source:

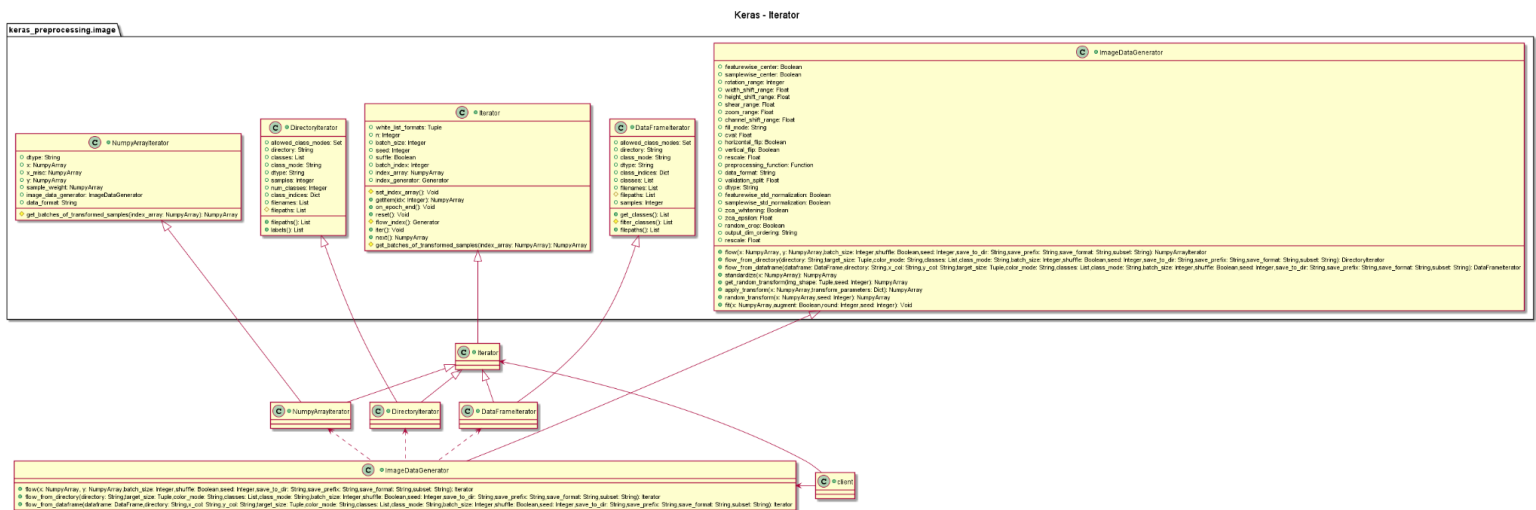
- <https://github.com/keras-team/keras/blob/v2.6.0/keras/preprocessing/image.py#L318-L319>

บรรทัด 318-1139 class Iterator,DirectoryIterator,NumpyArrayIterator,DataFrameIterator

- [https://github.com/keras-team/keras-preprocessing/tree/4538765fd369def80f81ad977bcf8e40e58c2f82/keras\\_preprocessing/image](https://github.com/keras-team/keras-preprocessing/tree/4538765fd369def80f81ad977bcf8e40e58c2f82/keras_preprocessing/image)

Folder เก็บ keras\_preprocessing.numpy\_array\_iterator, image\_data\_generator, directory\_iterator, dataframe\_iterator, Iterator

#### UML Diagram



Source:

[https://raw.githubusercontent.com/pholpaphankorn/keras\\_softarch\\_assignment/main/uml\\_diagram/image/svg/keras\\_iteratorv2.svg](https://raw.githubusercontent.com/pholpaphankorn/keras_softarch_assignment/main/uml_diagram/image/svg/keras_iteratorv2.svg)

### ตัวอย่างการใช้งาน

```
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=input_shape))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation('sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])

# this is the augmentation configuration we will use for training
train_datagen = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

# this is the augmentation configuration we will use for testing:
# only rescaling
test_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='binary')
```

```
validation_generator = test_datagen.flow_from_directory(  
    validation_data_dir,  
    target_size=(img_width, img_height),  
    batch_size=batch_size,  
    class_mode='binary')  
  
model.fit_generator(  
    train_generator,  
    steps_per_epoch=nb_train_samples // batch_size,  
    epochs=epochs,  
    validation_data=validation_generator,  
    validation_steps=nb_validation_samples // batch_size)
```

มีการใช้ flow\_from\_directory ในการ Iterate ภาพที่เก็บอยู่ใน directory ของเครื่อง

Source:

[https://gist.github.com/fchollet/0830affa1f7f19fd47b06d4cf89ed44d#file-classifier\\_from\\_little\\_data\\_script\\_1-py-L86](https://gist.github.com/fchollet/0830affa1f7f19fd47b06d4cf89ed44d#file-classifier_from_little_data_script_1-py-L86)

จัดทำโดย

1. 62010525 นายปภักร ธนโรจน์
2. 62010602 นายพลพัฒน์ สงวนสิริกุล
3. 62010604 นายพลภัทร จงวัฒนศิริ
4. 62010615 นายพัฒน์ภูมิ หาแก้ว
5. 62010619 นายพัทพล จันทร์ชู
6. 62010785 นายลิขิตภูมิ ลิขิตงาม