Specialist Programme on Artificial Intelligence for IT & ITES Industry

Convolutional neural networks, Part 2

By Dr. Tan Jen Hong
tanjenhong@nus.edu.sg

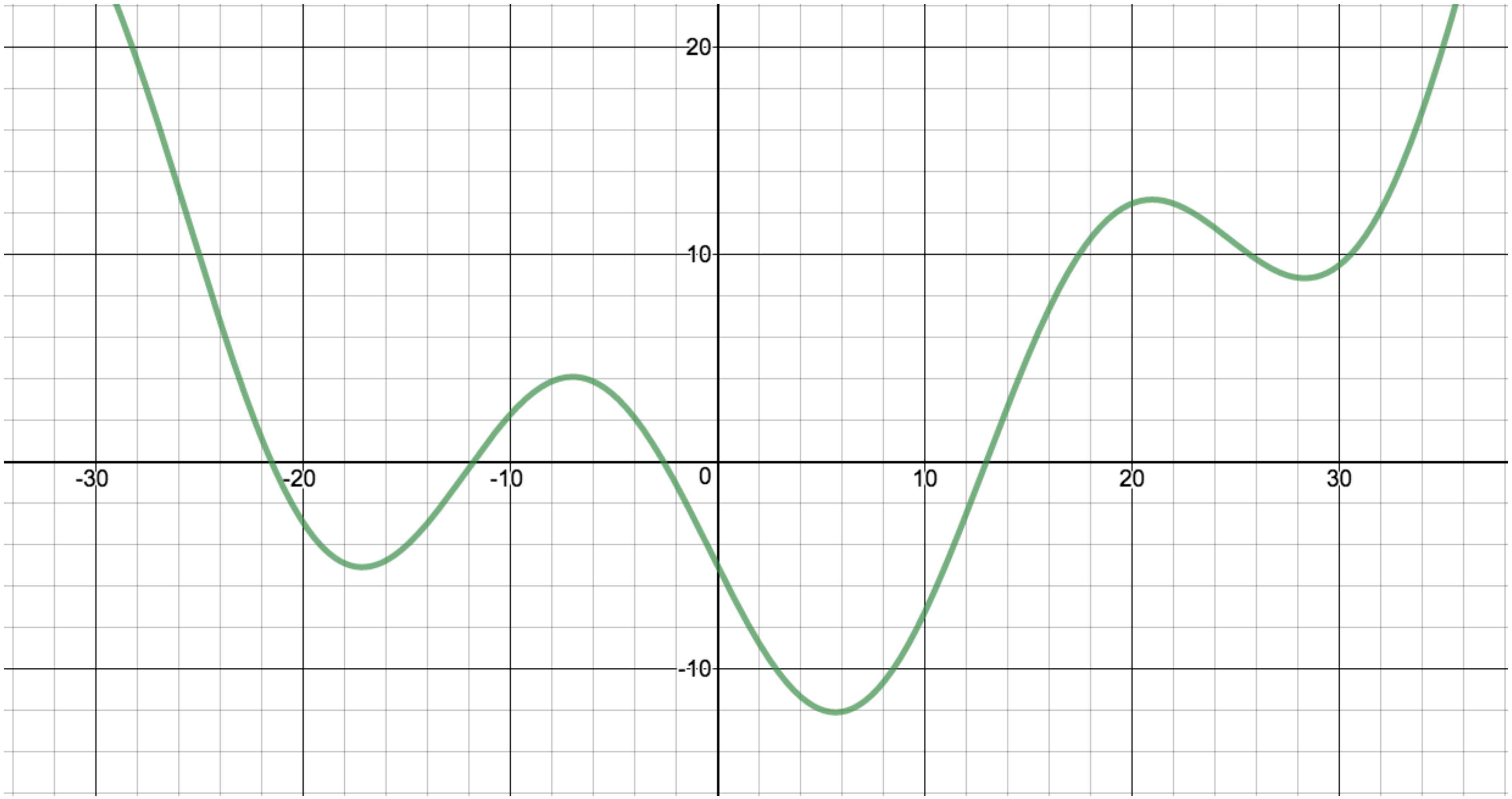
Singapore e-Government Leadership Centre
National University of Singapore

Inspire Lead Transform

The need of a deeper network and functional APIs

Representation

Do you think deep neural net can represent this function?

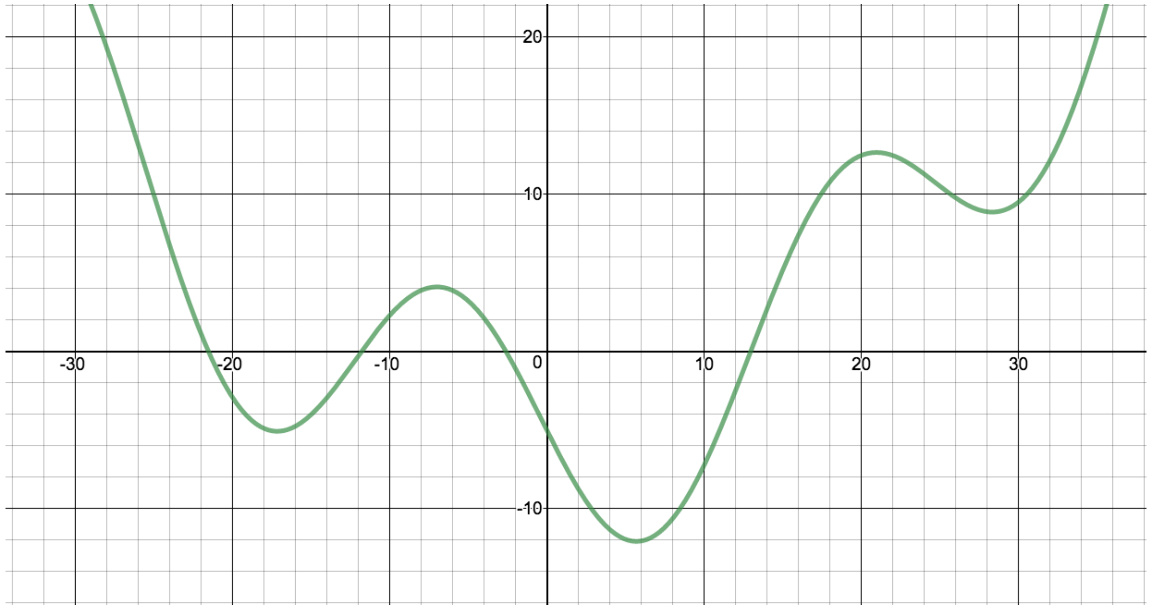


Source: By Brendan Fortuner

Universal approximation theorem

A theorem for all problem?

- A feedforward network with a single layer is enough to represent any function
- What is the implication?
- With the theorem, is deep neural network the panacea for all our challenges?



Source: By Brendan Fortuner

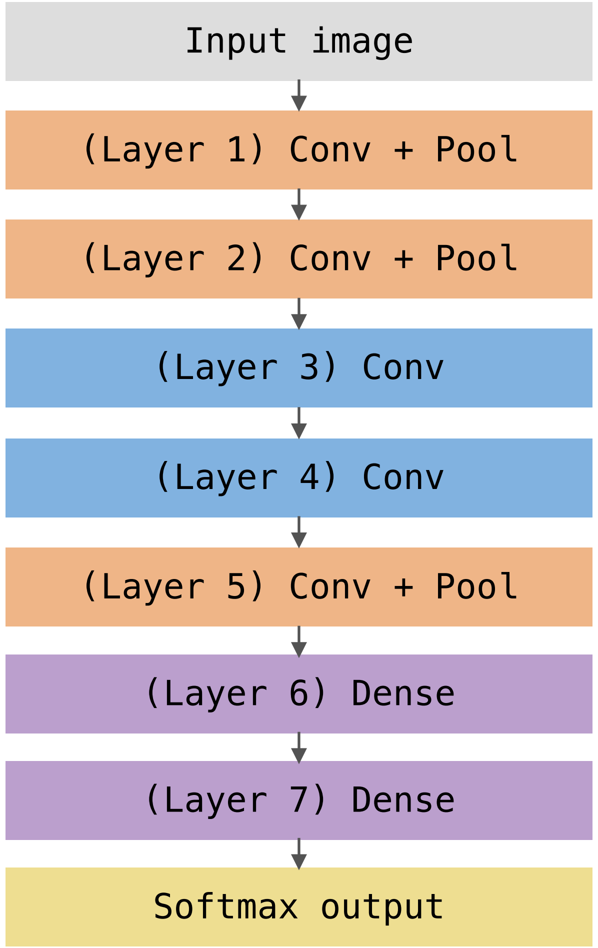
Back to the real world

The issue at hand

- Although by universal approximation theorem, possible to perform good classification on a single layer net, so far no learning algorithm can achieve that
- Key: a single layer net can approximate any function (proven), but the theorem does not provide any clue to achieve that
- On the contrary, the experiences from the past decade show that depth of a net is the key to great performance

The importance of depth

A study on Krizhevsky et al. model (2012)



- The model that won ILSVRC 2012
- 8 layers in total, 60 million parameters, 650,000 neurons
- Re-implementation gives 18.1% top-5 error

Source: Re-implementation by Rob Fergus

Top-5 error

How about top-1 error

Persian cat

Source: http://www.vetstreet.com/cats/persian#1_ugw20zmq

Top-1 error

```
Chihuahua (0.4)
Hyena (0.25)
Koala (0.15)
Persian cat (0.1)
Burmese cat (0.02)
```

Considered **incorrect**

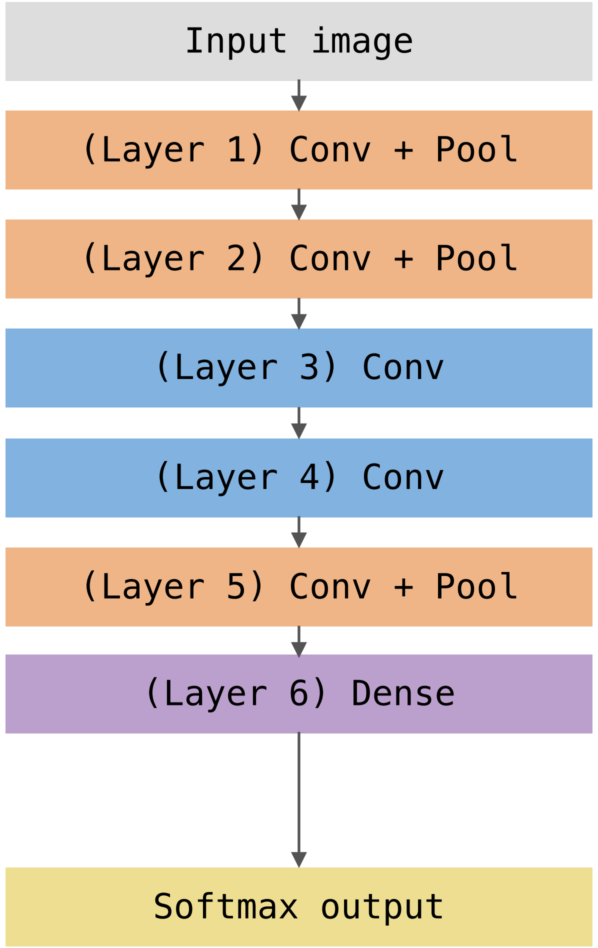
Top-5 error

```
Chihuahua (0.4)
Hyena (0.25)
Koala (0.15)
Persian cat (0.1)
Burmese cat (0.02)
```

Considered **correct**

The importance of depth

A study on Krizhevsky et al. model (2012)



- Remove layer 7, the fully connected layer

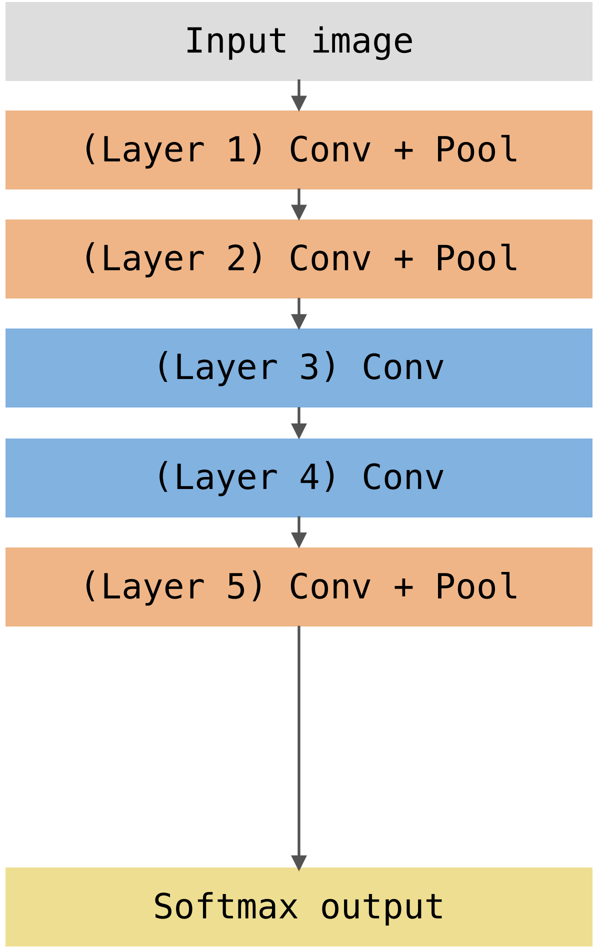
- 16 million less parameters compared to the original model

- Only 1.1% drop in performance!

Source: Re-implementation by Rob Fergus

The importance of depth

A study on Krizhevsky et al. model (2012)



- Remove both layer 6 and layer 7, the two fully connected layers

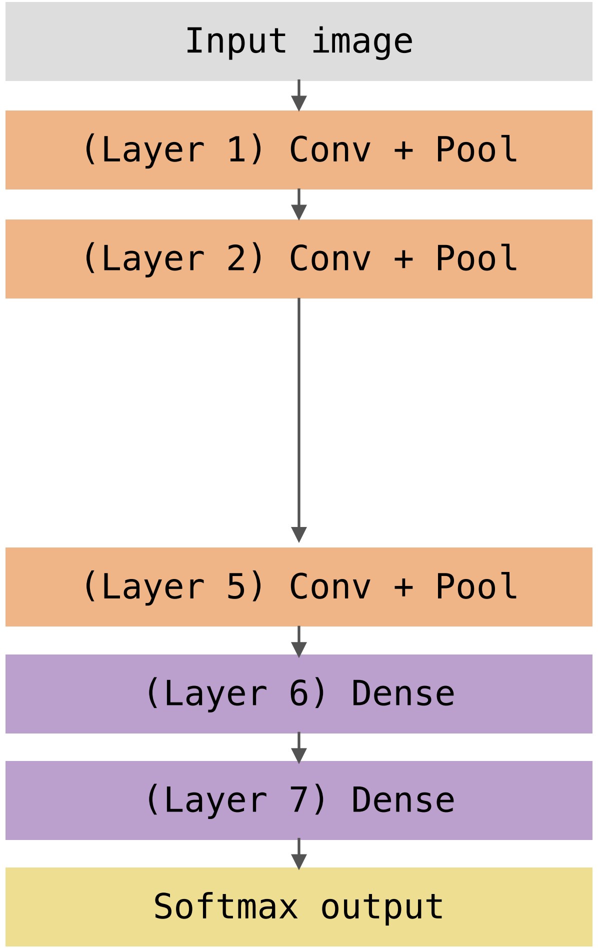
- 50 million less parameters compared to the original model

- 5.7% drop in performance

Source: Re-implementation by Rob Fergus

The importance of depth

A study on Krizhevsky et al. model (2012)



- Now try remove upper layers, the feature extractor. Remove layer 3 and 4

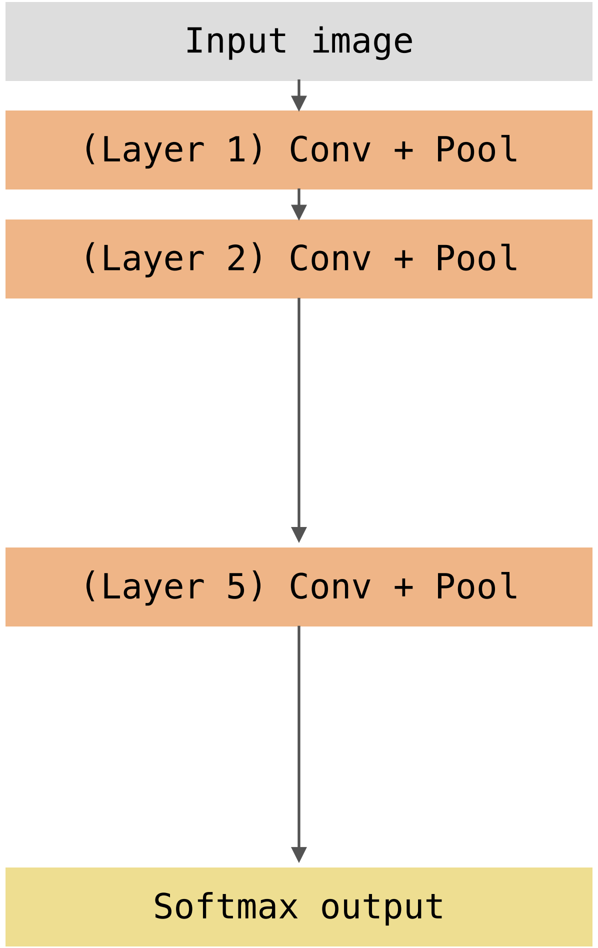
- 1 million less parameters compared to the original model

- 3.0% drop in performance

Source: Re-implementation by Rob Fergus

The importance of depth

A study on Krizhevsky et al. model (2012)



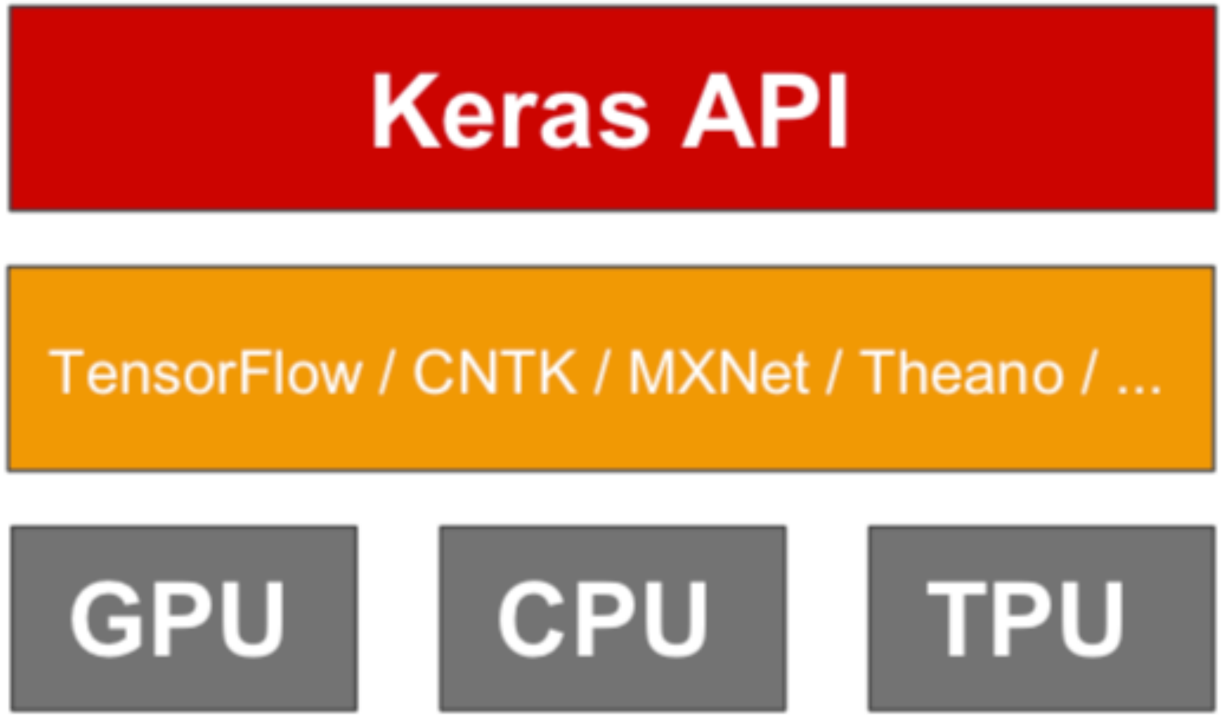
- Now try remove upper layers, the feature extractor. Remove layer 3, 4, 6, and 7
- Only 4 layers left
- 33.5% drop in performance!
- Depth is the key

Source: Re-implementation by Rob Fergus

Keras

What is Keras

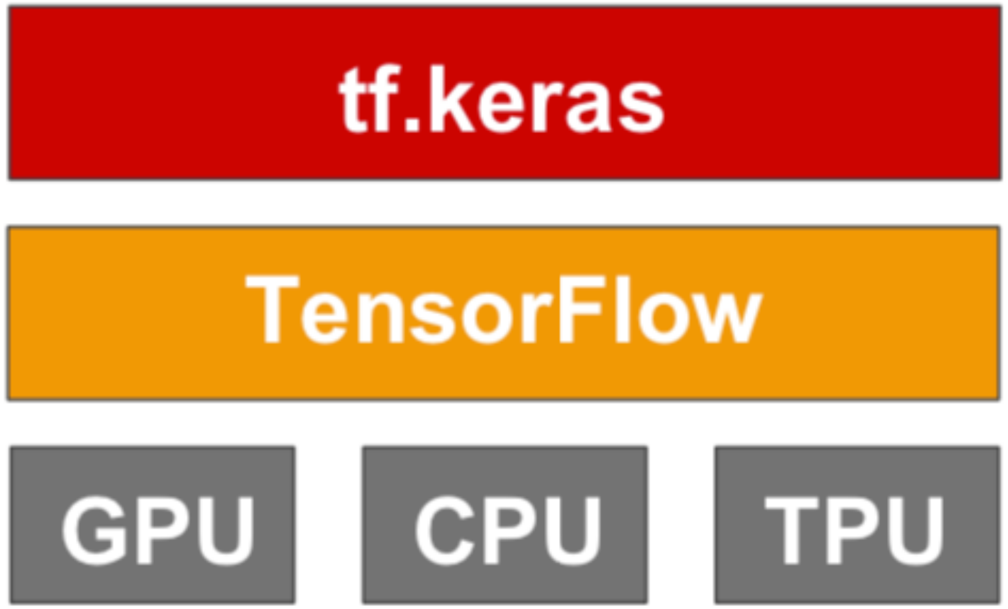
The basic architecture



Source: Francois Chollet

Keras + Tensorflow

The official high-level API of Tensorflow



Source: Francois Chollet

- Possible to build deep net solely using Tensorflow, but toooo many repetition
- Need to define every detail (weight, bias, initialization and etc.)
- Since Tensorflow v1.4, Keras is part of the core
- In Tensorflow v1.13, we have tensorflow.keras module
- From Tensorflow v2.0, Keras API is **THE preferred way** of building neural network

Keras

Three API styles

- Sequential model

  Super simple

  Only for single-input, single-output sequential layer stacks

  Good for 70+% of use cases

- Functional API
- Works like playing Lego bricks
- Multi-input, multi-output, arbitrary static graph topologies
- Good for 95% of use cases

- Model subclassing

  Maximum flexibility

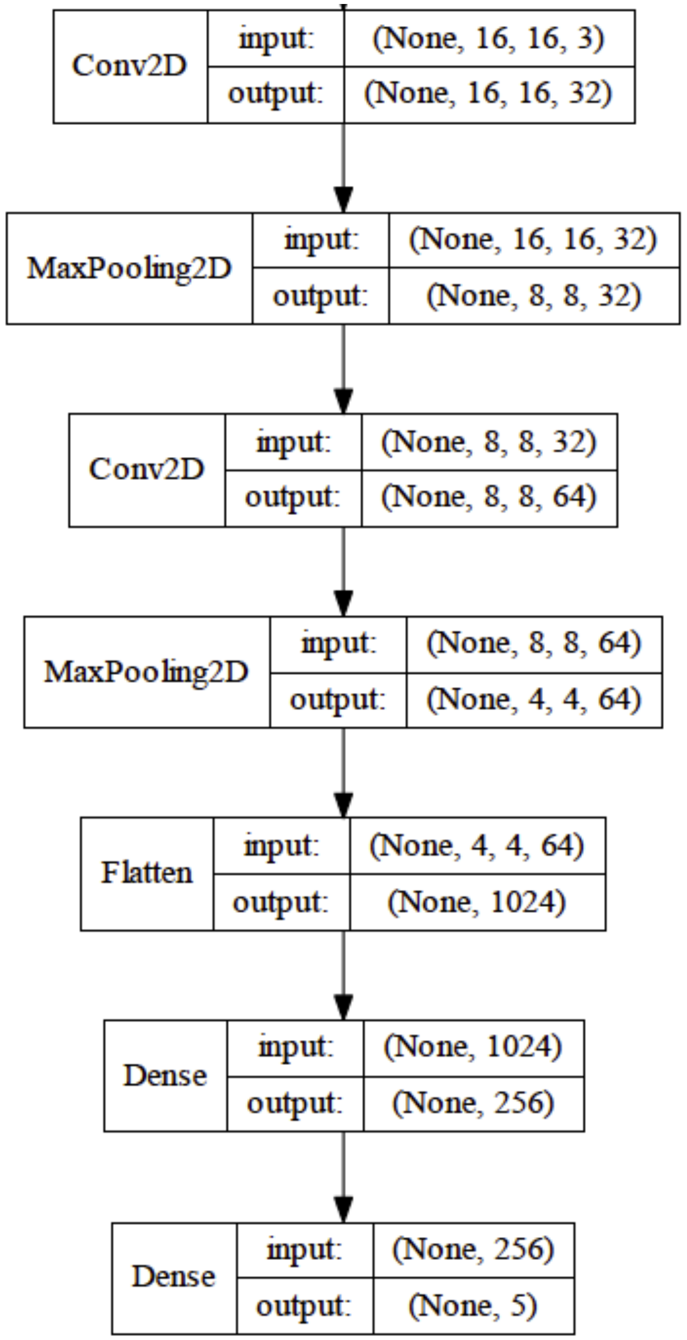
  Larger potential error surface

Source: Francois Chollet

Keras

Sequential model

- Adding layers by sequence (layer by layer)
- `input_shape` must be present in the first layer



Conv2D | input: (None, 16, 16, 3) | output: (None, 16, 16, 32)

MaxPooling2D | input: (None, 16, 16, 32) | output: (None, 8, 8, 32)

Conv2D | input: (None, 8, 8, 32) | output: (None, 8, 8, 64)

MaxPooling2D | input: (None, 8, 8, 64) | output: (None, 4, 4, 64)

Flatten | input: (None, 4, 4, 64) | output: (None, 1024)

Dense | input: (None, 1024) | output: (None, 256)

Dense | input: (None, 256) | output: (None, 5)

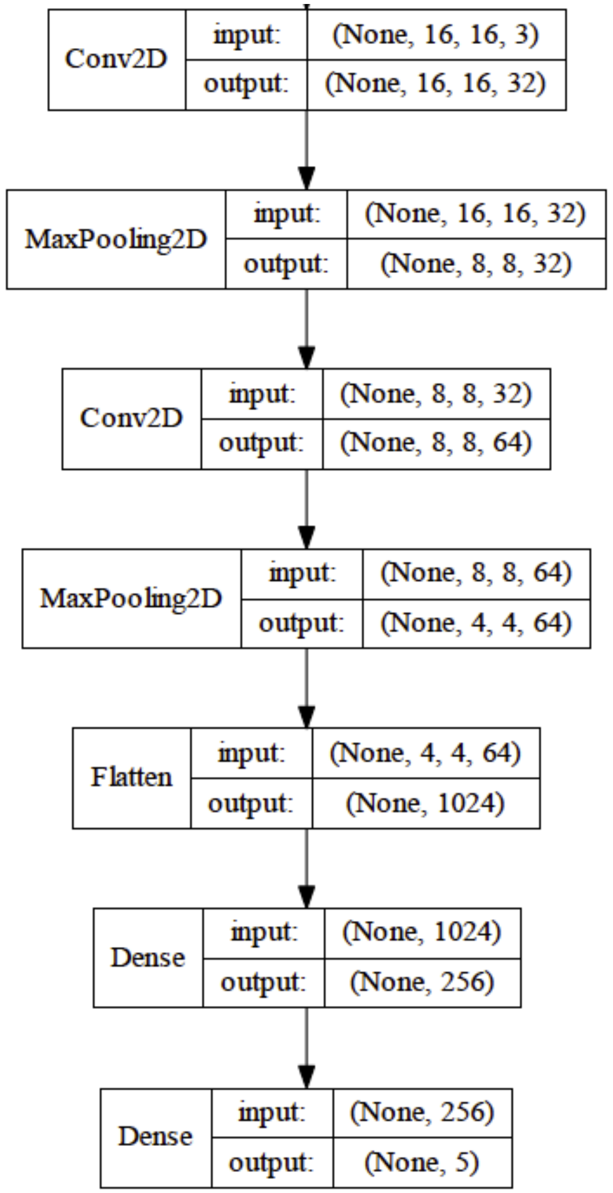
Keras

Sequential model

```
> def createSeqModel():
      model   = Sequential()
      model.add(Conv2D(32,(3,3),
                input_shape=(16,16,3),
                padding='same',
                activation='relu'))
      model.add(MaxPooling2D(pool_size=(2,2)))
      model.add(Conv2D(64,(3,3),
                padding='same',
                activation='relu'))
      model.add(MaxPooling2D(pool_size=(2,2)))

      model.add(Flatten())
      model.add(Dense(256,activation='relu'))
      model.add(Dense(5,activation='softmax'))

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



Conv2D input: (None, 16, 16, 3) output: (None, 16, 16, 32)

MaxPooling2D input: (None, 16, 16, 32) output: (None, 8, 8, 32)

Conv2D input: (None, 8, 8, 32) output: (None, 8, 8, 64)

MaxPooling2D input: (None, 8, 8, 64) output: (None, 4, 4, 64)

Flatten input: (None, 4, 4, 64) output: (None, 1024)

Dense input: (None, 1024) output: (None, 256)

Dense input: (None, 256) output: (None, 5)

Keras

Sequential model

- Create the model and show the model summary

```
> modelSeq     = createSeqModel()
> modelSeq.summary()
```

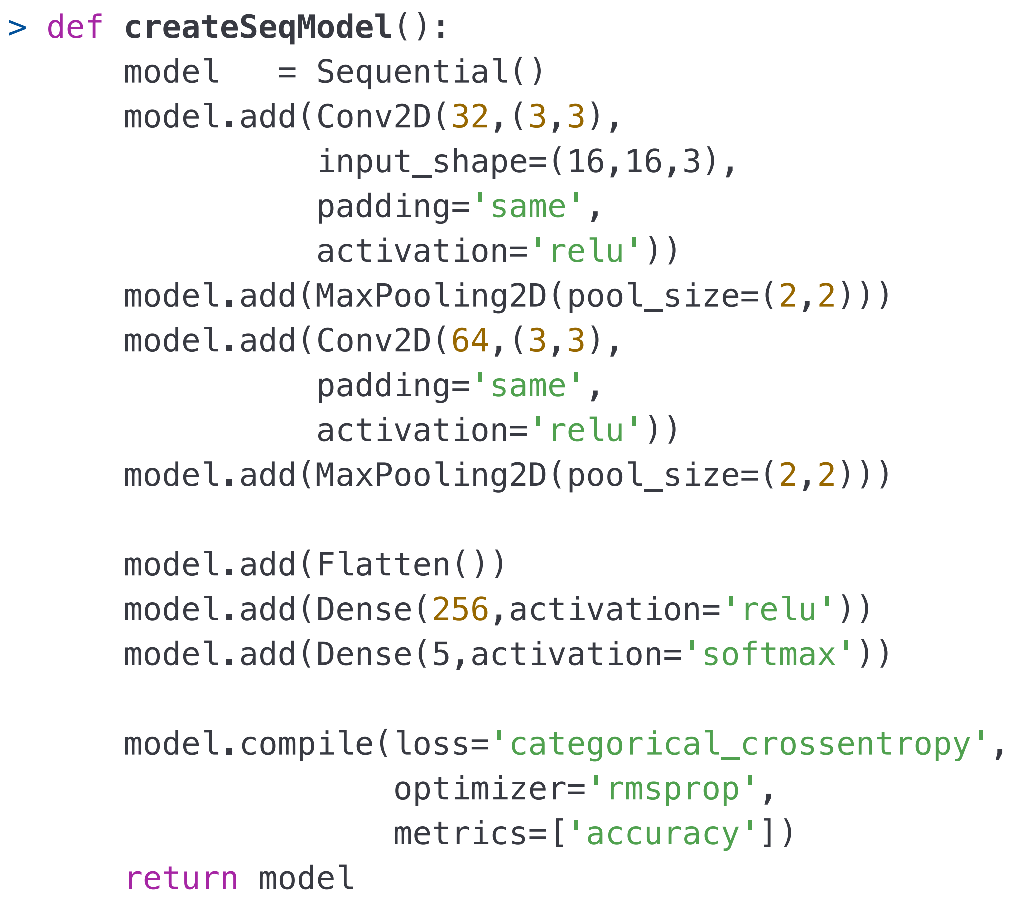
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 16, 16, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 8, 8, 32)          0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 8, 8, 64)          18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 256)               262400
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 1285
=================================================================
Total params: 283,077
Trainable params: 283,077
Non-trainable params: 0
_________________________________________________________________
```

Keras

Sequential model

- The problem

  Not possible for multiple input
  Not possible for multiple output
  A single direction of flow of tensors, no branching, no merging
  Not possible to reuse layer; shared layer is not possible



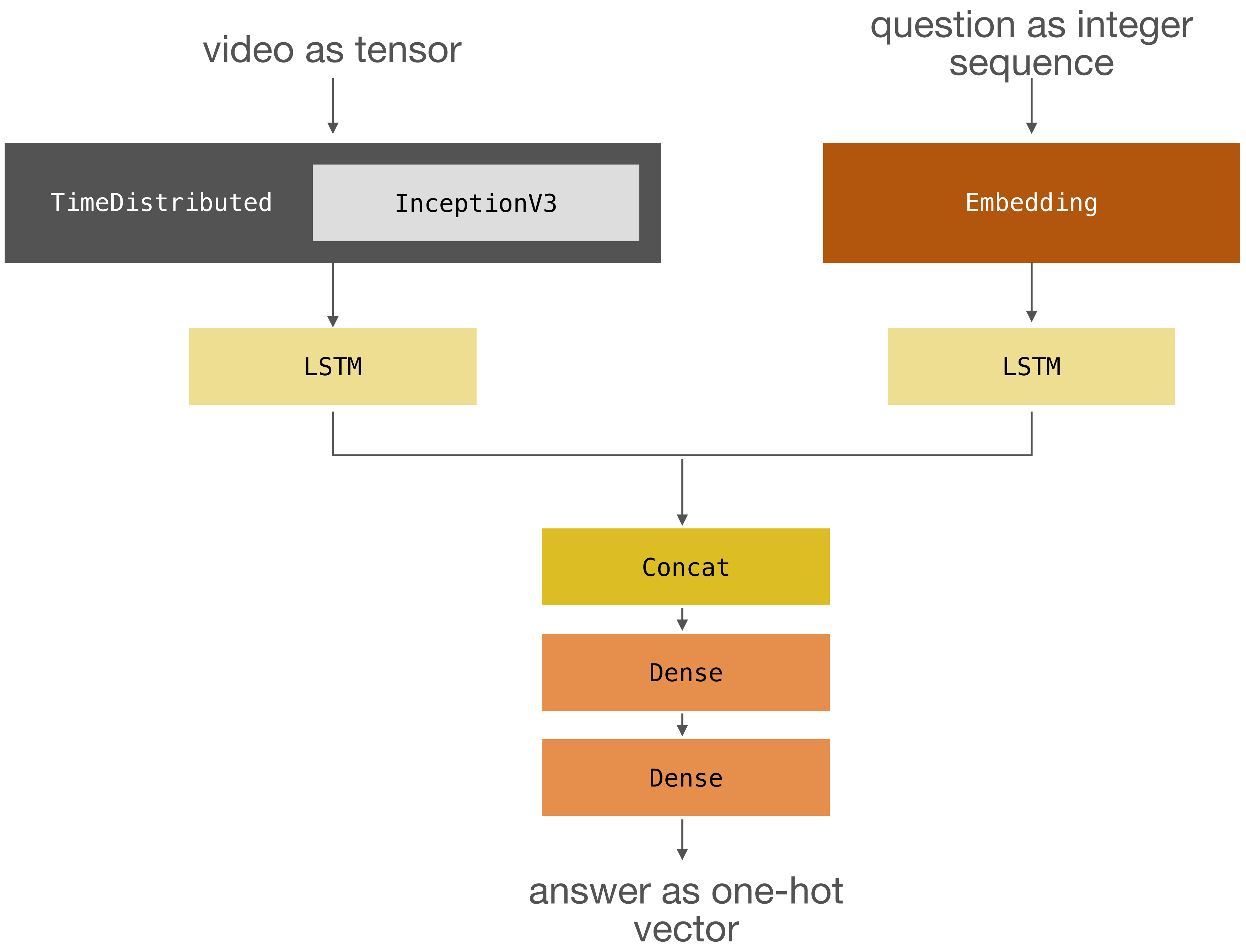
```
> def createSeqModel():
      model   = Sequential()
      model.add(Conv2D(32,(3,3),
                input_shape=(16,16,3),
                padding='same',
                activation='relu'))
      model.add(MaxPooling2D(pool_size=(2,2)))
      model.add(Conv2D(64,(3,3),
                padding='same',
                activation='relu'))
      model.add(MaxPooling2D(pool_size=(2,2)))

      model.add(Flatten())
      model.add(Dense(256,activation='relu'))
      model.add(Dense(5,activation='softmax'))

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

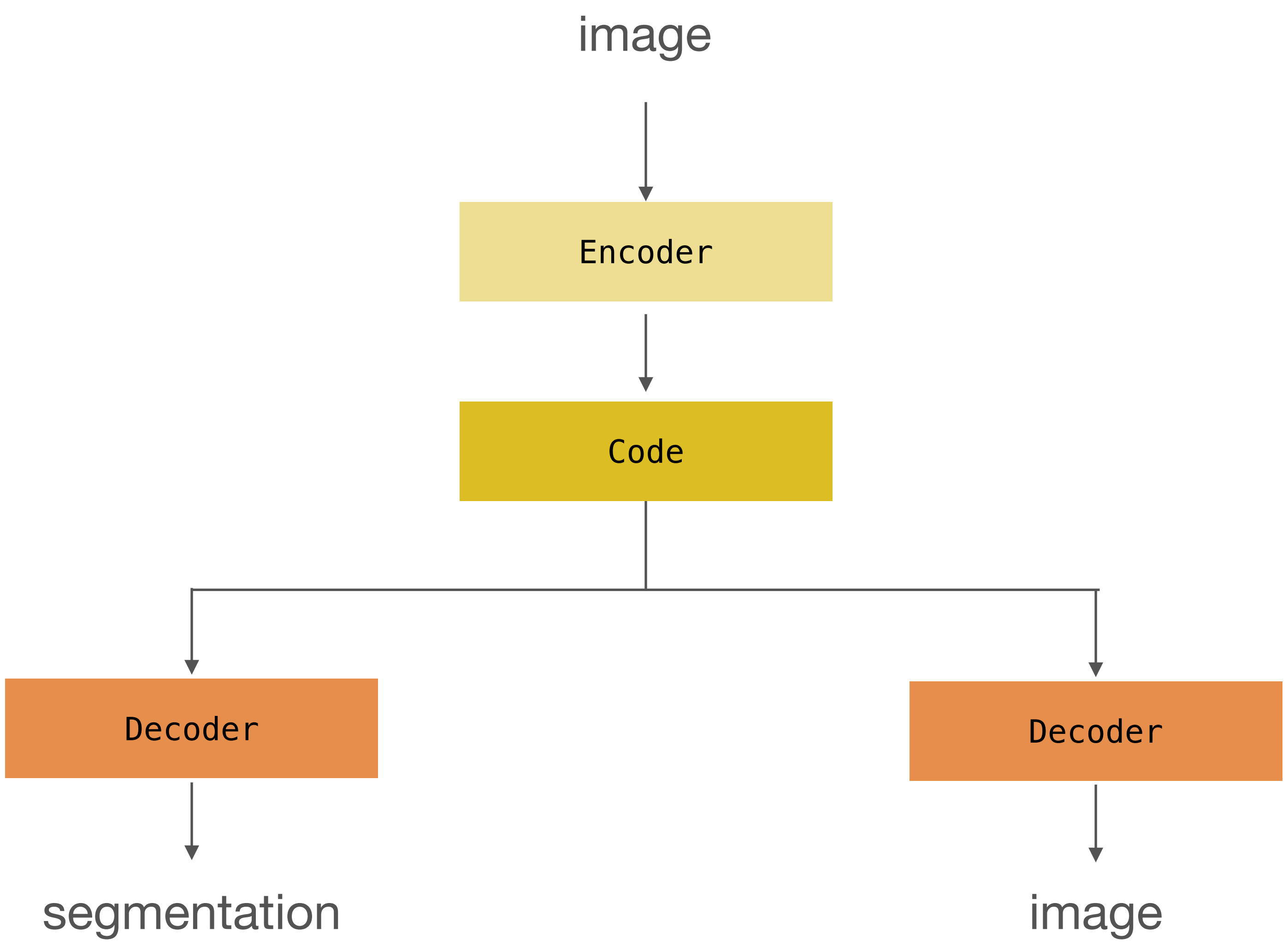
Multiple inputs

An video and a question



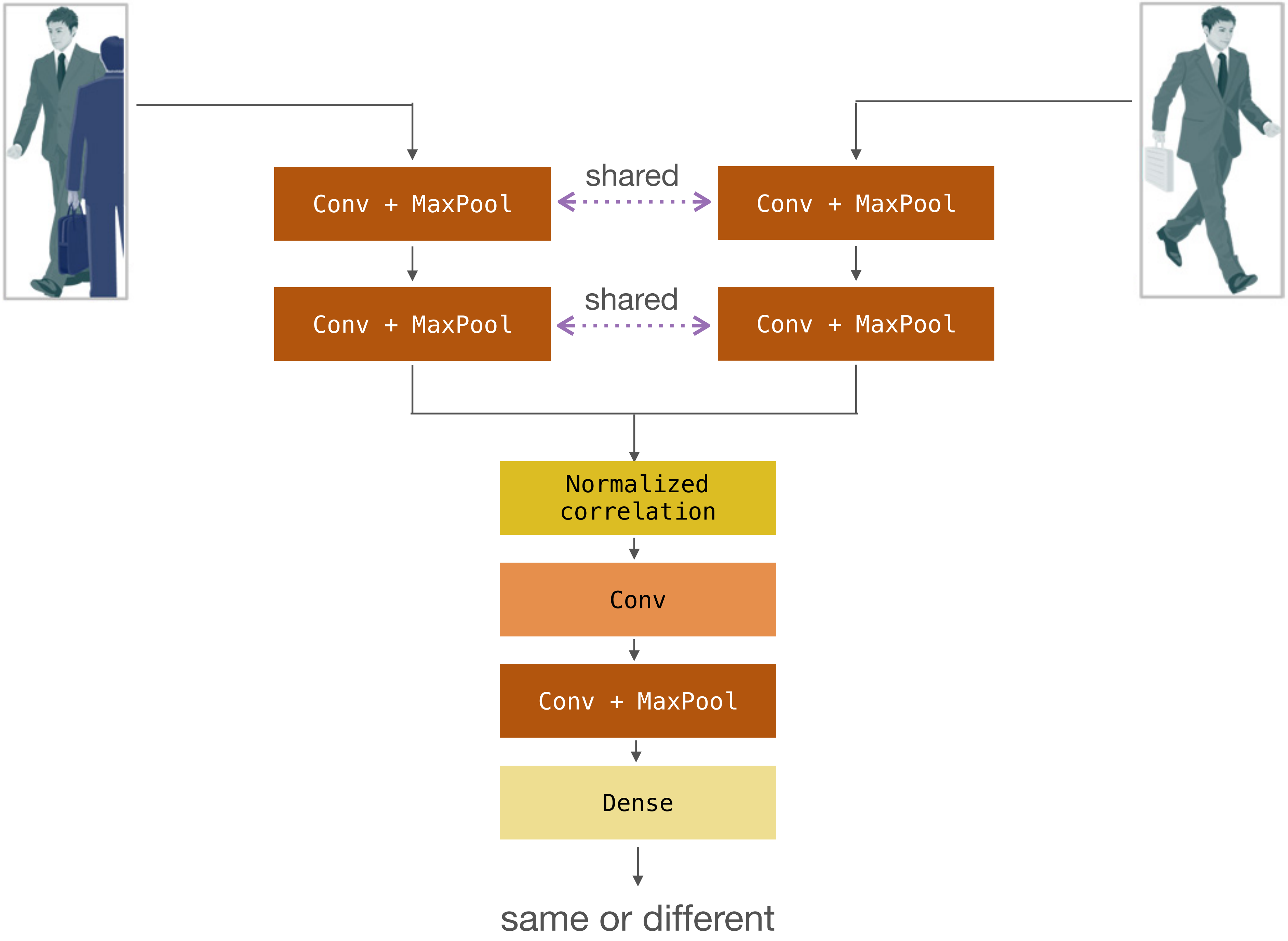
Multiple outputs

An adjusted image and a segmentation



Shared layers

Person re-identification



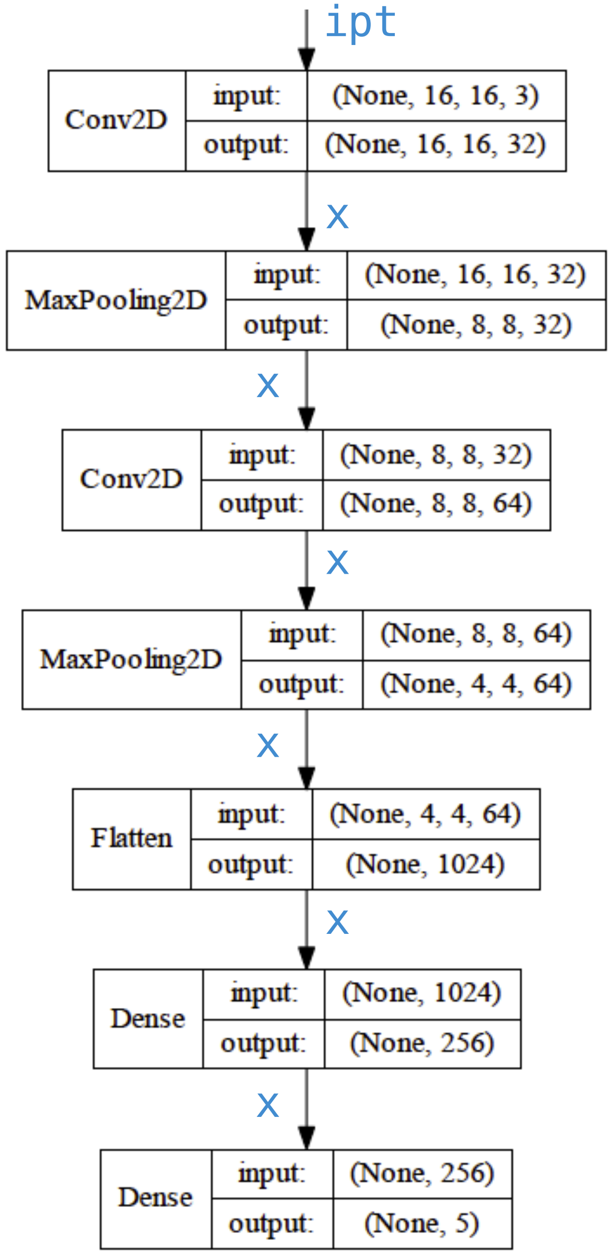
Keras

functional APIs

```
> def createFuncModel():
      ipt      = Input(shape=(16,16,3))
      x        = Conv2D(32,(3,3),
                       padding='same',
                       activation='relu')(ipt)
      x        = MaxPooling2D(pool_size=(2,2))(x)
      x        = Conv2D(64,(3,3),
                     padding='same',
                     activation='relu')(x)
      x        = MaxPooling2D(pool_size=(2,2))(x)
      x        = Flatten()(x)
      x        = Dense(256,activation='relu')(x)
      x        = Dense(5,activation='relu')(x)

      model    = Model(inputs=ipt,outputs=x)

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



ipt
Conv2D | input: | (None, 16, 16, 3)
output: | (None, 16, 16, 32)
x
MaxPooling2D | input: | (None, 16, 16, 32)
output: | (None, 8, 8, 32)
x
Conv2D | input: | (None, 8, 8, 32)
output: | (None, 8, 8, 64)
x
MaxPooling2D | input: | (None, 8, 8, 64)
output: | (None, 4, 4, 64)
x
Flatten | input: | (None, 4, 4, 64)
output: | (None, 1024)
x
Dense | input: | (None, 1024)
output: | (None, 256)
x
Dense | input: | (None, 256)
output: | (None, 5)

Keras

Comparison

functional APIs

```
> def createFuncModel():
      ipt     = Input(shape=(16,16,3))
      x       = Conv2D(32,(3,3),
                       padding='same',
                       activation='relu')(ipt)
      x       = MaxPooling2D(pool_size=(2,2))(x)
      x       = Conv2D(64,(3,3),
                     padding='same',
                     activation='relu')(x)
      x       = MaxPooling2D(pool_size=(2,2))(x)
      x       = Flatten()(x)
      x       = Dense(256,activation='relu')(x)
      x       = Dense(5,activation='relu')(x)

      model   = Model(inputs=ipt,outputs=x)

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

Sequential model

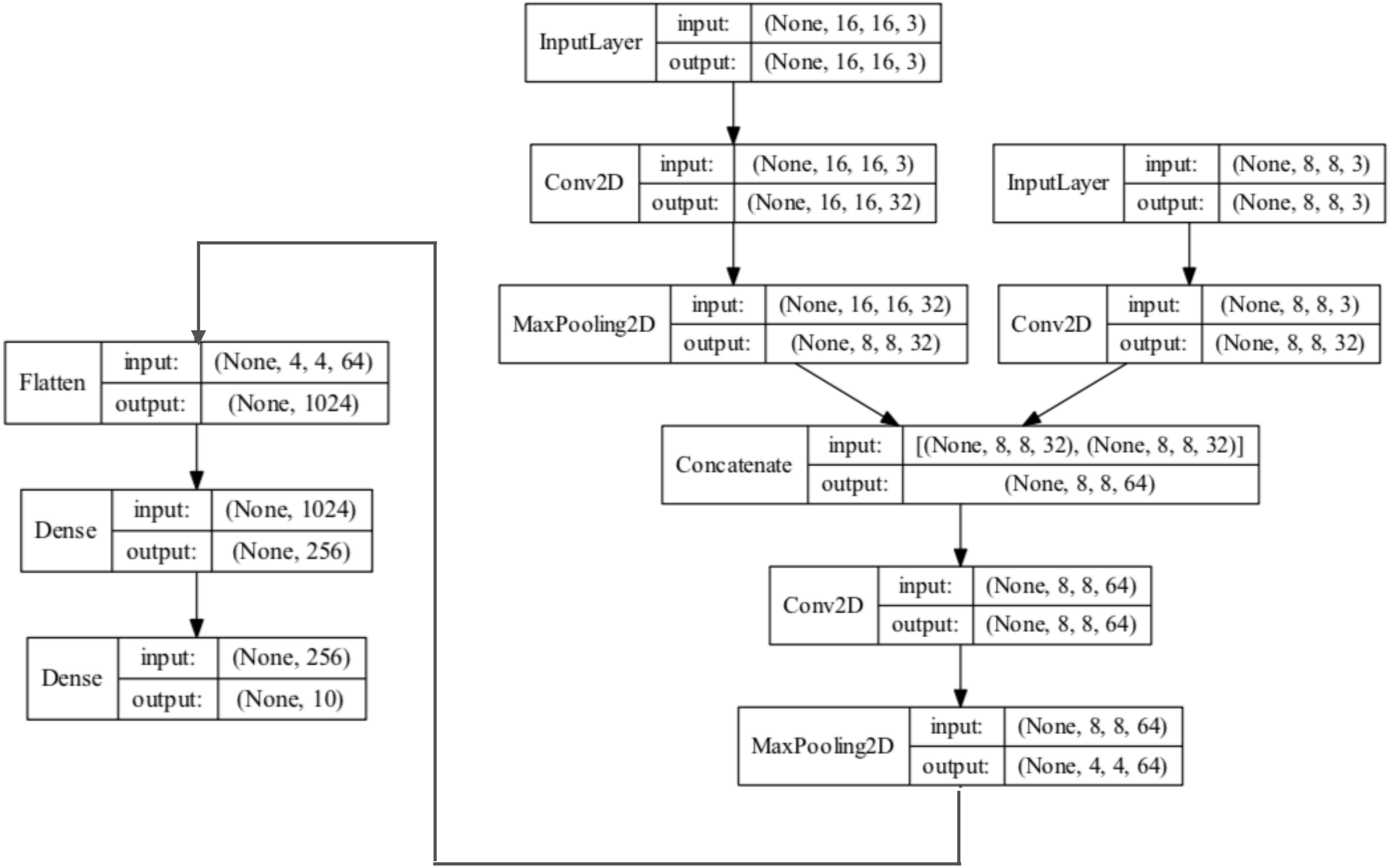
```
> def createSeqModel():
      model   = Sequential()
      model.add(Conv2D(32,(3,3),
                input_shape=(16,16,3),
                padding='same',
                activation='relu'))
      model.add(MaxPooling2D(pool_size=(2,2)))
      model.add(Conv2D(64,(3,3),
                padding='same',
                activation='relu'))
      model.add(MaxPooling2D(pool_size=(2,2)))

      model.add(Flatten())
      model.add(Dense(256,activation='relu'))
      model.add(Dense(5,activation='softmax'))

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

'Y' shape architecture

how to create?



InputLayer | input: (None, 16, 16, 3) | output: (None, 16, 16, 3)

Conv2D | input: (None, 16, 16, 3) | output: (None, 16, 16, 32)

MaxPooling2D | input: (None, 16, 16, 32) | output: (None, 8, 8, 32)

InputLayer | input: (None, 8, 8, 3) | output: (None, 8, 8, 3)

Conv2D | input: (None, 8, 8, 3) | output: (None, 8, 8, 32)

Concatenate | input: [(None, 8, 8, 32), (None, 8, 8, 32)] | output: (None, 8, 8, 64)

Conv2D | input: (None, 8, 8, 64) | output: (None, 8, 8, 64)

MaxPooling2D | input: (None, 8, 8, 64) | output: (None, 4, 4, 64)

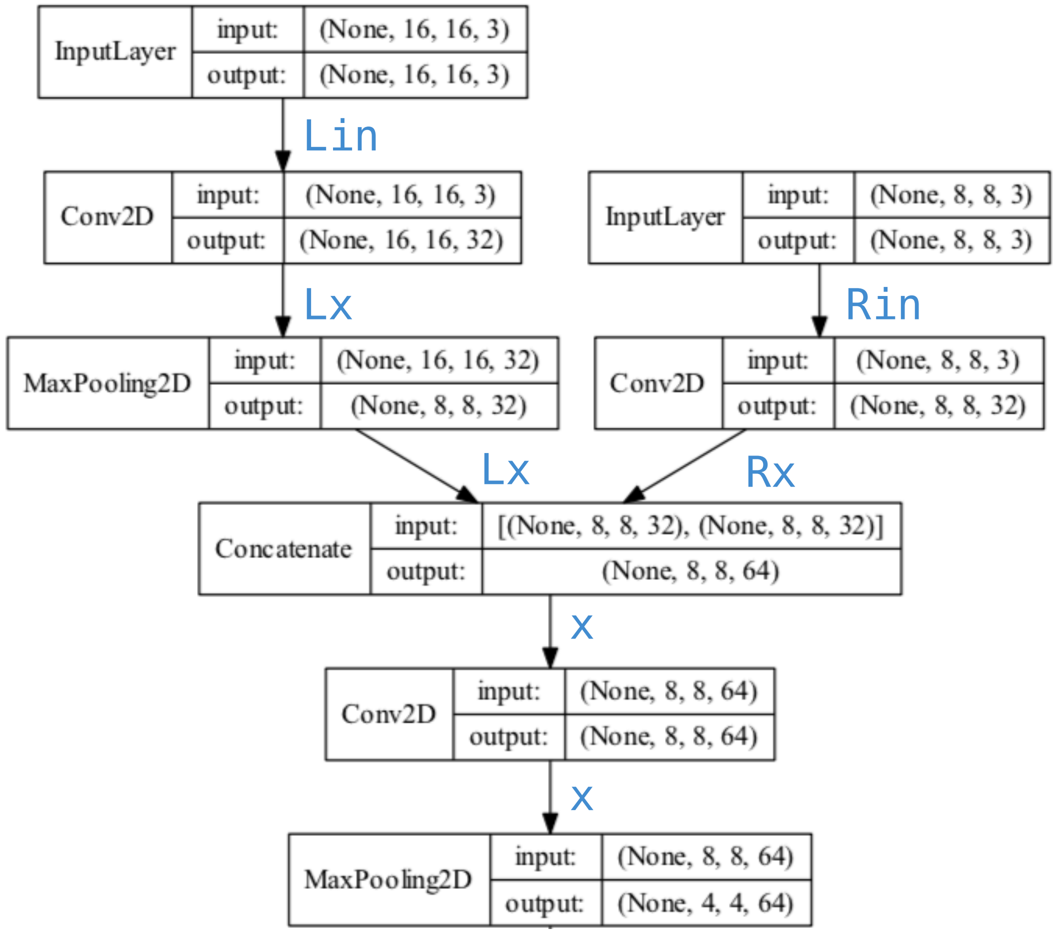
Flatten | input: (None, 4, 4, 64) | output: (None, 1024)

Dense | input: (None, 1024) | output: (None, 256)

Dense | input: (None, 256) | output: (None, 10)

'Y' shape architecture

the code


InputLayer | input: (None, 16, 16, 3) | output: (None, 16, 16, 3)

Conv2D | input: (None, 16, 16, 3) | output: (None, 16, 16, 32)

InputLayer | input: (None, 8, 8, 3) | output: (None, 8, 8, 3)

MaxPooling2D | input: (None, 16, 16, 32) | output: (None, 8, 8, 32)

Conv2D | input: (None, 8, 8, 3) | output: (None, 8, 8, 32)

Concatenate | input: [(None, 8, 8, 32), (None, 8, 8, 32)] | output: (None, 8, 8, 64)

Conv2D | input: (None, 8, 8, 64) | output: (None, 8, 8, 64)

MaxPooling2D | input: (None, 8, 8, 64) | output: (None, 4, 4, 64)

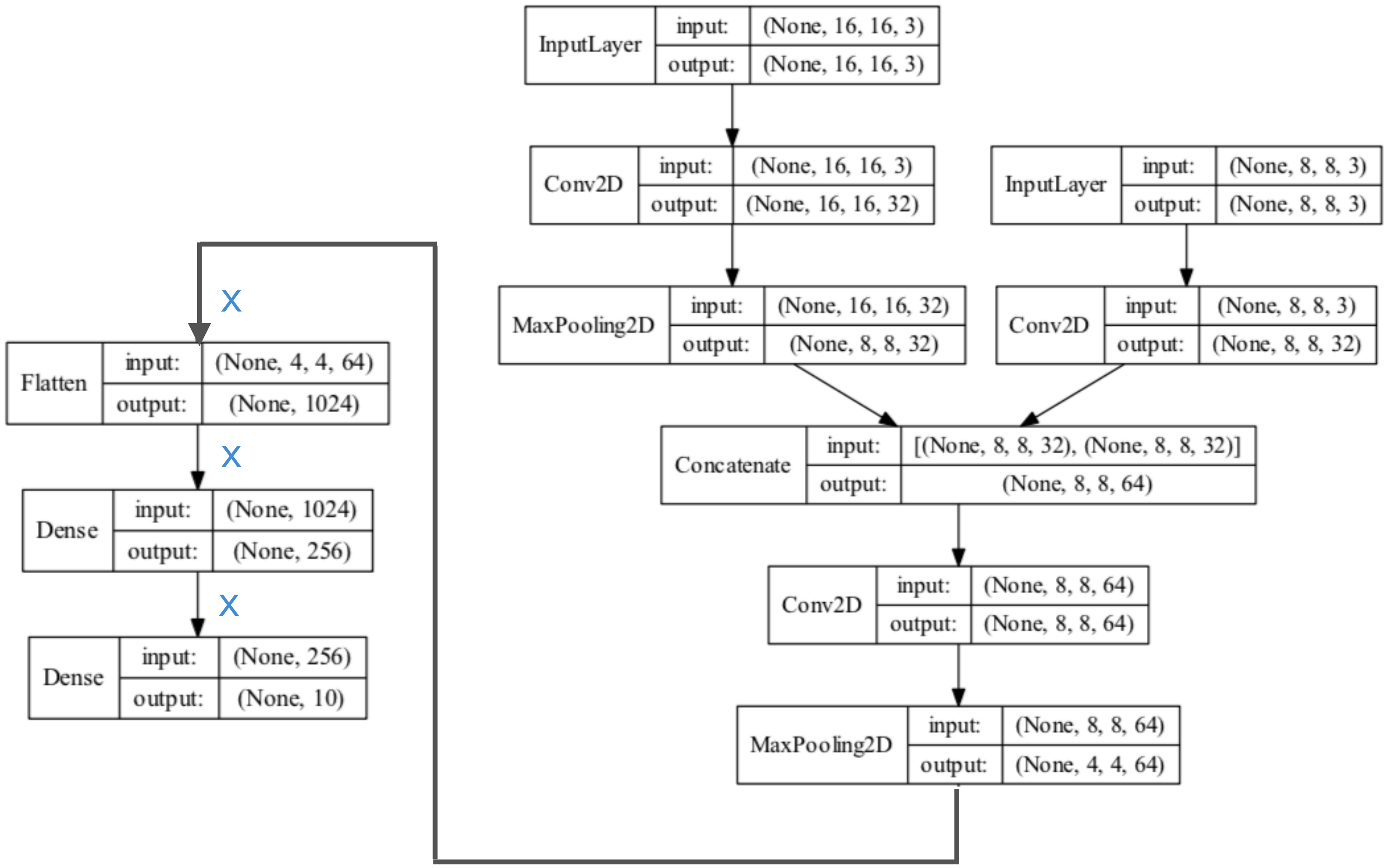
```
> def createDualInputModel():
      Lin      = Input(shape=(16,16,3))
      Lx       = Conv2D(32,(3,3),padding='same',activation='relu')(Lin)
      Lx       = MaxPooling2D(pool_size=(2,2))(Lx)

      Rin      = Input(shape=(8,8,3))
      Rx       = Conv2D(32,(3,3),padding='same',activation='relu')(Rin)

      x        = concatenate([Lx,Rx],axis=-1)
      x        = Conv2D(64,(3,3),padding='same',activation='relu')(x)
      x        = MaxPooling2D(pool_size=(2,2))(x)
      ...
```

'Y' shape architecture

the code



```
> def createDualInputModel():
      ....
      x       = Flatten()(x)
      x       = Dense(256,activation='relu')(x)
      x       = Dense(10,activation='softmax')(x)

      model   = Model(inputs=[Lin,Rin],outputs=x)
      model.compile(loss='categorical_crossentropy',
                    optimizer='rmsprop',
                    metrics=['accuracy'])
      return model
```

'Y' shape architecture

Putting all together ....

- Remember to include the below before the function

```
from tensorflow.keras.layers import concatenate
```

```
> def createDualInputModel():
      Lin      = Input(shape=(16,16,3))
      Lx       = Conv2D(32,(3,3),padding='same',activation='relu')(Lin)
      Lx       = MaxPooling2D(pool_size=(2,2))(Lx)

      Rin      = Input(shape=(8,8,3))
      Rx       = Conv2D(32,(3,3),padding='same',activation='relu')(Rin)

      x        = concatenate([Lx,Rx],axis=-1)
      x        = Conv2D(64,(3,3),padding='same',activation='relu')(x)
      x        = MaxPooling2D(pool_size=(2,2))(x)

      x        = Flatten()(x)
      x        = Dense(256,activation='relu')(x)
      x        = Dense(10,activation='softmax')(x)

      model    = Model(inputs=[Lin,Rin],outputs=x)
      model.compile(loss='categorical_crossentropy',
                    optimizer='rmsprop',
                    metrics=['accuracy'])
      return model
```

Multiple inputs

the code

- Create the model and show the model summary

```
> modelDual     = createDualInputModel()
> modelDual.summary()
```

```
Layer (type)                    Output Shape          Param #     Connected to
==================================================================================================
input_1 (InputLayer)            (None, 16, 16, 3)     0
__________________________________________________________________________________________________
conv2d (Conv2D)                 (None, 16, 16, 32)    896         input_1[0][0]
__________________________________________________________________________________________________
input_2 (InputLayer)            (None, 8, 8, 3)       0
__________________________________________________________________________________________________
max_pooling2d (MaxPooling2D)    (None, 8, 8, 32)      0           conv2d[0][0]
__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 8, 8, 32)      896         input_2[0][0]
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 8, 8, 64)      0           max_pooling2d[0][0]
                                                                  conv2d_1[0][0]
__________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 8, 8, 64)      36928       concatenate[0][0]
__________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 4, 4, 64)      0           conv2d_2[0][0]
__________________________________________________________________________________________________
flatten (Flatten)               (None, 1024)          0           max_pooling2d_1[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 128)           131200      flatten[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 3)             387         dense[0][0]
==================================================================================================
Total params: 170,307
Trainable params: 170,307
Non-trainable params: 0
__________________________________________________________________________________________________
```

Multiple inputs

training

- Assume RDat and LDat are the training input, TLbl is the label for the training
- Furthermore, vRDat and vLDat is the validation input, and vLbl is the label
- The training is done by

```
> model.fit([RDat,LDat],
            TLbl,
            validation_data=([vRDat,vLDat], vLbl),
            epochs=100,
            batch_size=128,
            shuffle=True,
            callbacks=callbacks_list)
```

Multiple inputs

Getting the model plot

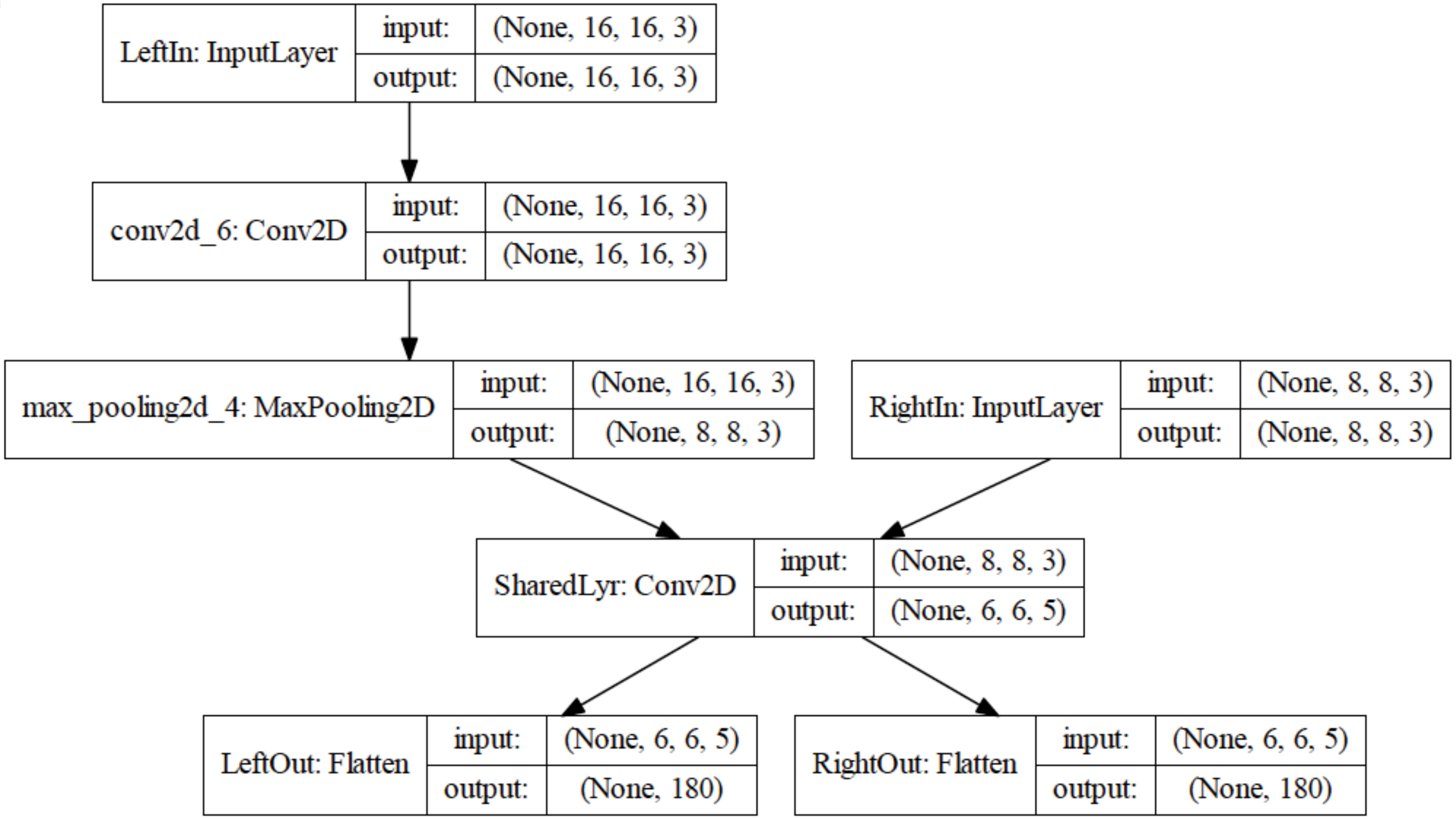
- To plot model, it requires pydot and graphviz
- On Mac, there is a need for additional installation for graphviz

```
> from tensorflow.keras.utils import plot_model

> plot_model(modelDual,
             to_file='Dual_model.pdf',
             show_shapes=True,
             show_layer_names=False,
             rankdir='TB')
```

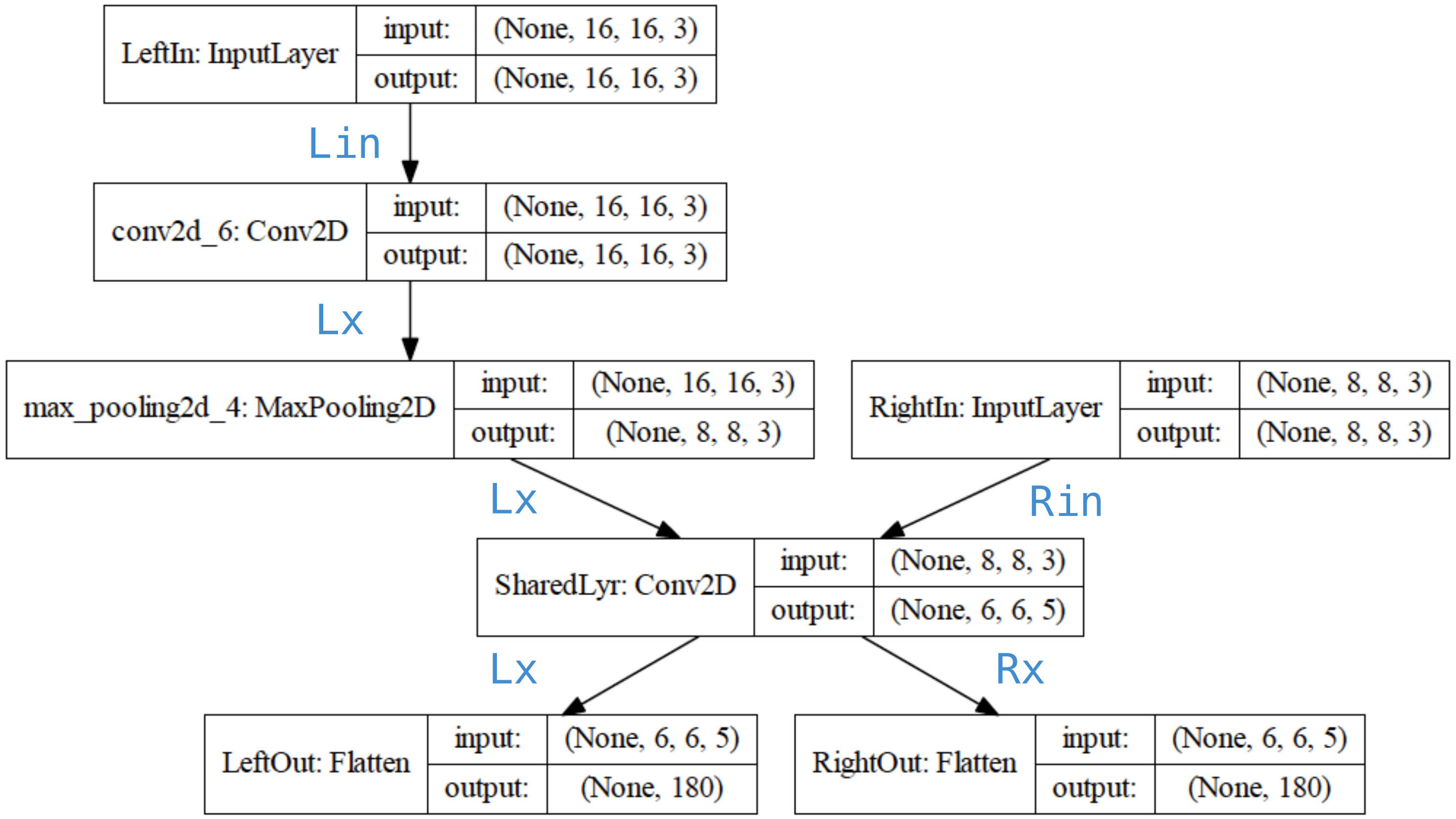
Shared layer

how to create?



Shared layer

the code



LeftIn: InputLayer | input: (None, 16, 16, 3) | output: (None, 16, 16, 3)

Lin

conv2d_6: Conv2D | input: (None, 16, 16, 3) | output: (None, 16, 16, 3)

Lx

max_pooling2d_4: MaxPooling2D | input: (None, 16, 16, 3) | output: (None, 8, 8, 3)

RightIn: InputLayer | input: (None, 8, 8, 3) | output: (None, 8, 8, 3)

Lx

Rin

SharedLyr: Conv2D | input: (None, 8, 8, 3) | output: (None, 6, 6, 5)

Lx

Rx

LeftOut: Flatten | input: (None, 6, 6, 5) | output: (None, 180)

RightOut: Flatten | input: (None, 6, 6, 5) | output: (None, 180)

```
> def createSharedModel():
      shared  = Conv2D(5,(3,3),activation=‘relu',name='SharedLyr')

      Lin     = Input(shape=(16,16,3),name='LeftIn')
      Lx      = Conv2D(3,(3,3),padding='same',activation='relu')(Lin)
      Lx      = MaxPooling2D(pool_size=(2,2))(Lx)
      Lx      = shared(Lx)
      Lx      = Flatten(name='LeftOut')(Lx)

      Rin     = Input(shape=(8,8,3),name='RightIn')
      Rx      = shared(Rx)
      Rx      = Flatten(name='RightOut')(Rx)
      model   = Model(inputs=[Lin,Rin],outputs=[Lx,Rx])
      ...
```

Shared layer

the code

- For model with more than one output, pass in a dictionary that specifies the loss function for each output

```
> def createSharedModel():
      shared  = Conv2D(5,(3,3),activation=‘relu',name='SharedLyr')

      Lin     = Input(shape=(16,16,3),name='LeftIn')
      Lx      = Conv2D(3,(3,3),padding='same',activation='relu')(Lin)
      Lx      = MaxPooling2D(pool_size=(2,2))(Lx)
      Lx      = shared(Lx)
      Lx      = Flatten(name='LeftOut')(Lx)

      Rin     = Input(shape=(8,8,3),name='RightIn')
      Rx      = shared(Rx)
      Rx      = Flatten(name='RightOut')(Rx)

      model   = Model(inputs=[Lin,Rin],outputs=[Lx,Rx])
      model.compile(loss={'LeftOut':'categorical_crossentropy',
                          'RightOut':'mean_squared_error'},
                    optimizer='rmsprop',
                    metrics=['accuracy'])
      return model
```

Keras

Merge layer

- Other possible layers to fuse multiple flows of tensors in Keras

```
tensorflow.keras.layers.add
tensorflow.keras.layers.subtract
tensorflow.keras.layers.multiply
tensorflow.keras.layers.average
tensorflow.keras.layers.maximum
tensorflow.keras.layers.minimum
```

- For the above, the tensor inputs to the layer should have the same size

- Good enough for most use cases, otherwise write your own lambda layer to perform the necessary task

Keras

Multiple GPU

- Training using multiple GPUs is easy in Keras
- In the example, each GPU will process 64 samples in a single batch
- Why a need for multiple GPUs?

```
> from tensorflow.keras.utils import multi_gpu_model

> parallel    = multi_gpu_model(model, gpus=4)
> parallel.compile(loss='categorical_crossentropy',
                   optimizer='rmsprop')

> parallel.fit(x, y, epochs=50, batch_size=256)
```

Contact eGL

Singapore e-Government Leadership Centre
National University of Singapore
29 Heng Mui Keng Terrace
Block D & E
Singapore 119620

Tel	:	(65) 6516 1156
Fax	:	(65) 6778 2571
URL	:	www.egl.sg
Email	:	egl-enquiries@nus.edu.sg

Inspire *Lead* *Transform*