

NUS-ISS

Pattern Recognition using Machine Learning System



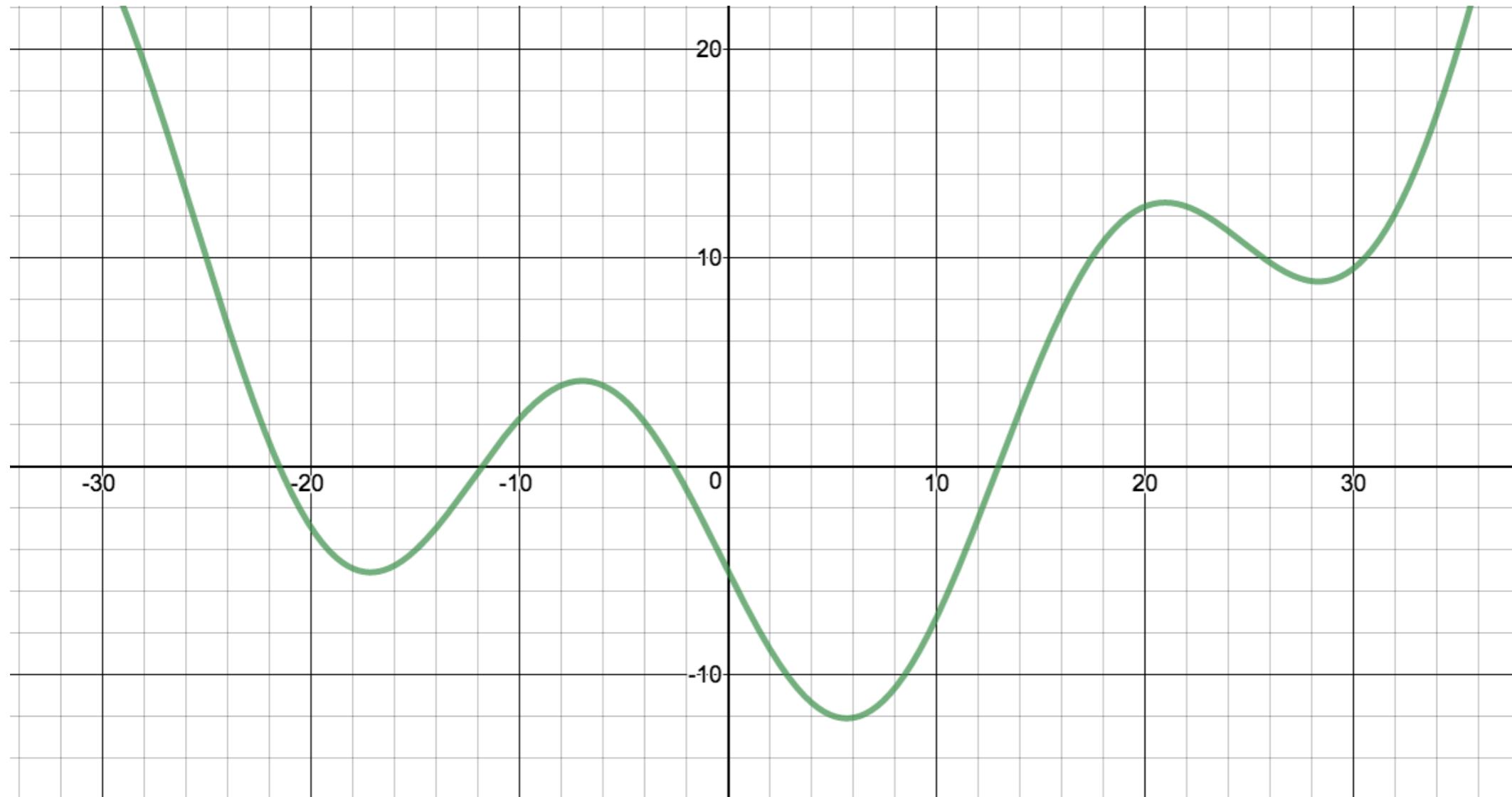
The need of a deeper network and functional APIs

by Dr. Tan Jen Hong

© 2019 National University of Singapore.
All Rights Reserved.

Representation

Do you think deep neural net can represent this function?



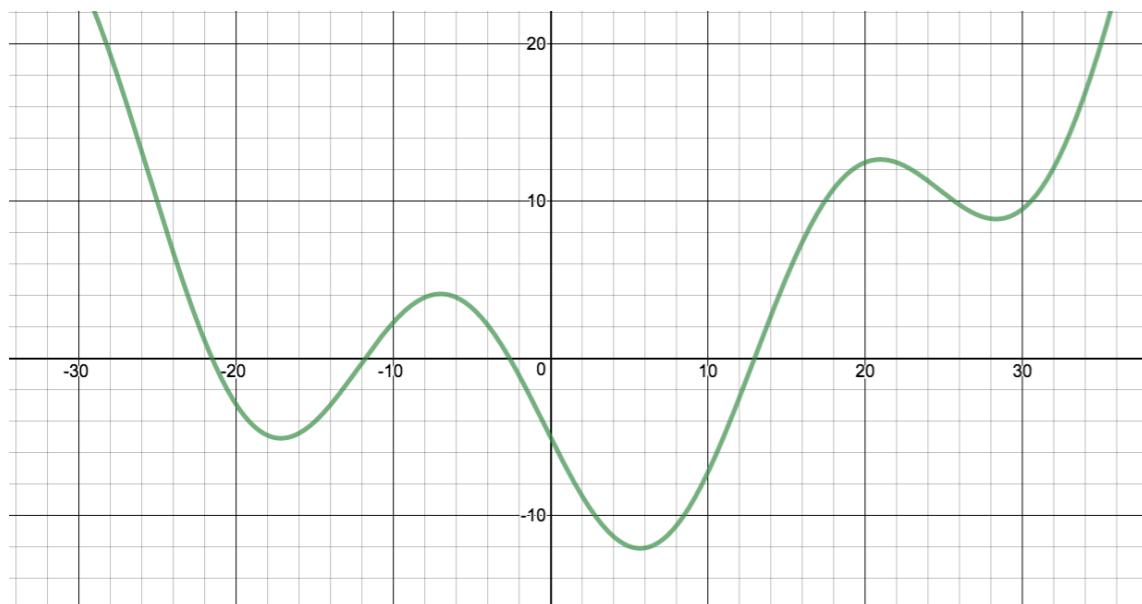
Source: By Brendan Fortuner

Group discussion

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?

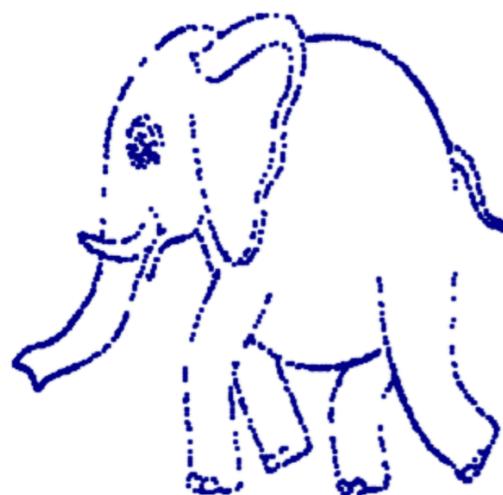


Source: By Brendan Fortuner

The extreme

An equation for all?

- Do you think it is possible to have one equation, with only a change in one parameter and generate the below four figure?



Source: By Laurent Boué

The extreme

An equation for all?

- In fact, all the samples of any arbitrary dataset can be reproduced by (where each sample has the form (x, y))

$$y = f_\alpha(x) = \sin^2 \left(2^{x\tau} \arcsin \sqrt{\alpha} \right)$$

- x must be a positive integer, τ is a constant controls the desired accuracy

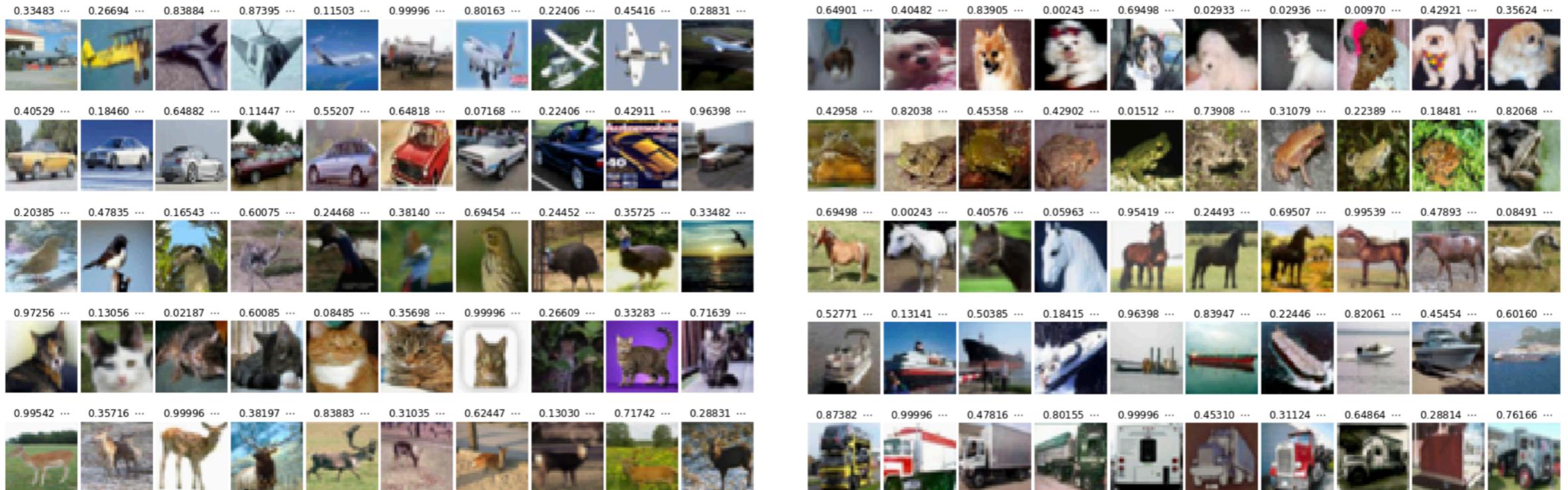


Source: <https://arxiv.org/pdf/1904.12320.pdf>

The extreme

An equation for all?

- Or even for this



Source: <https://arxiv.org/pdf/1904.12320.pdf>

$$y = f_\alpha(x) = \sin^2 \left(2^{x\tau} \arcsin \sqrt{\alpha} \right)$$

Back to the real world

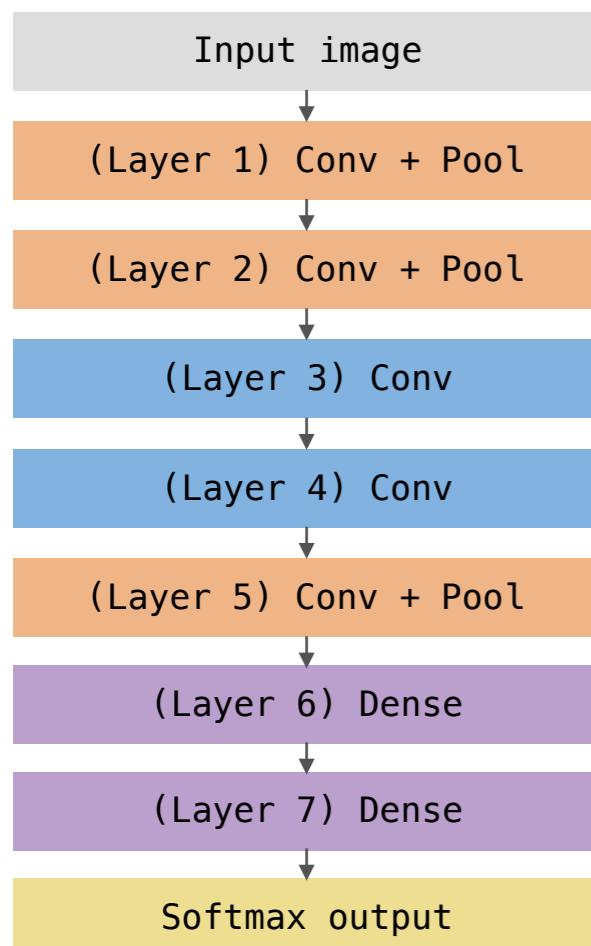
The issue at hand

- 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)

Top-5 error

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

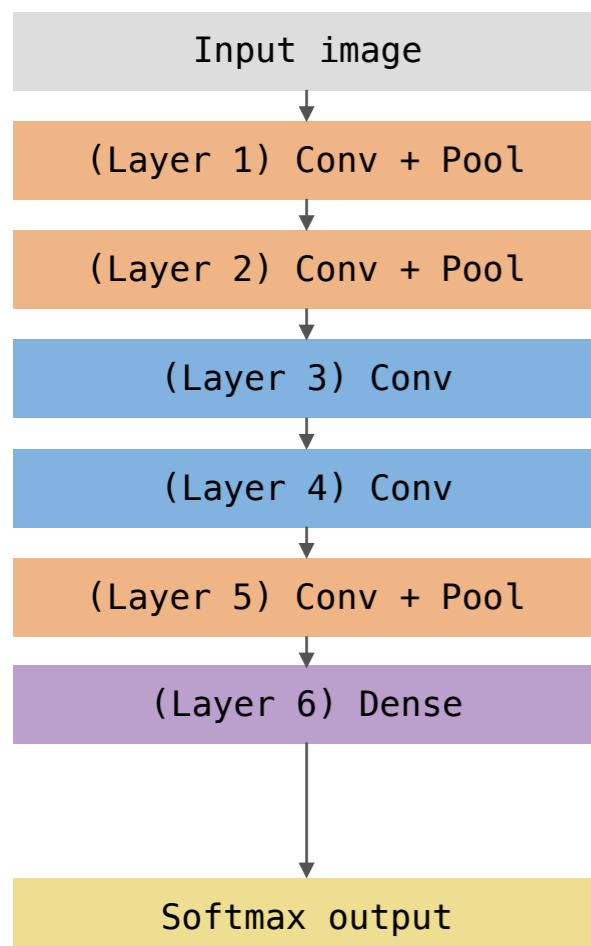
Considered **incorrect**

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!

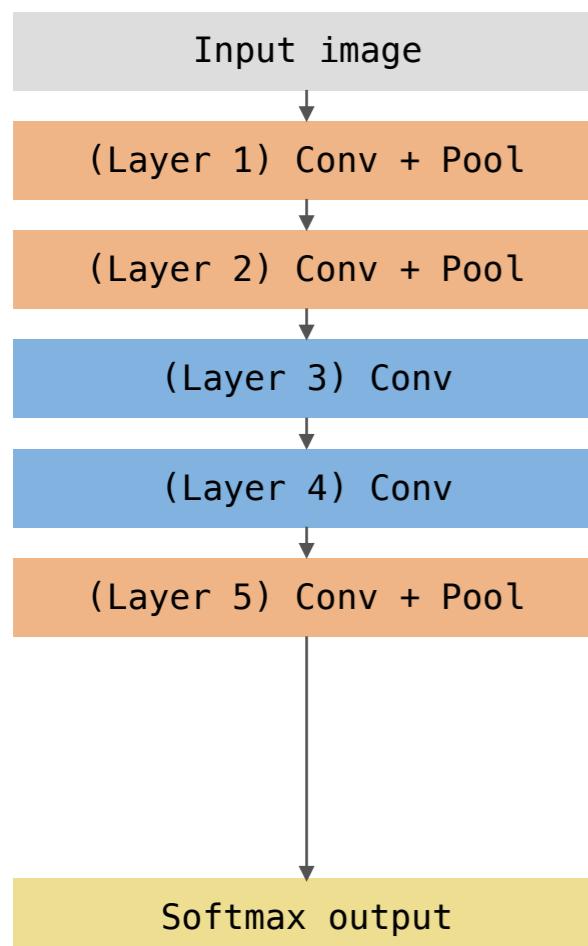


Source: Re-implementation by Rob Fergus

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

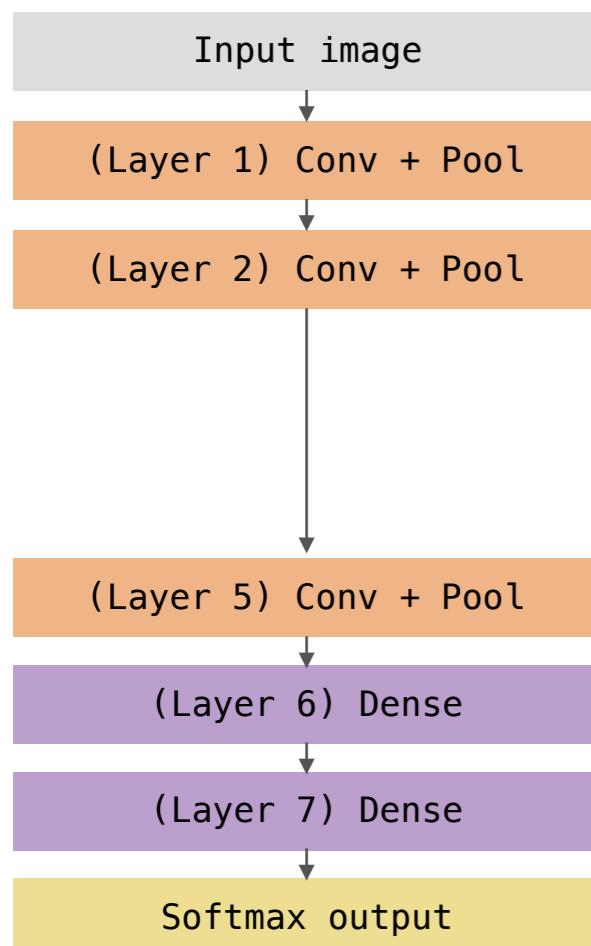
The importance of depth

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

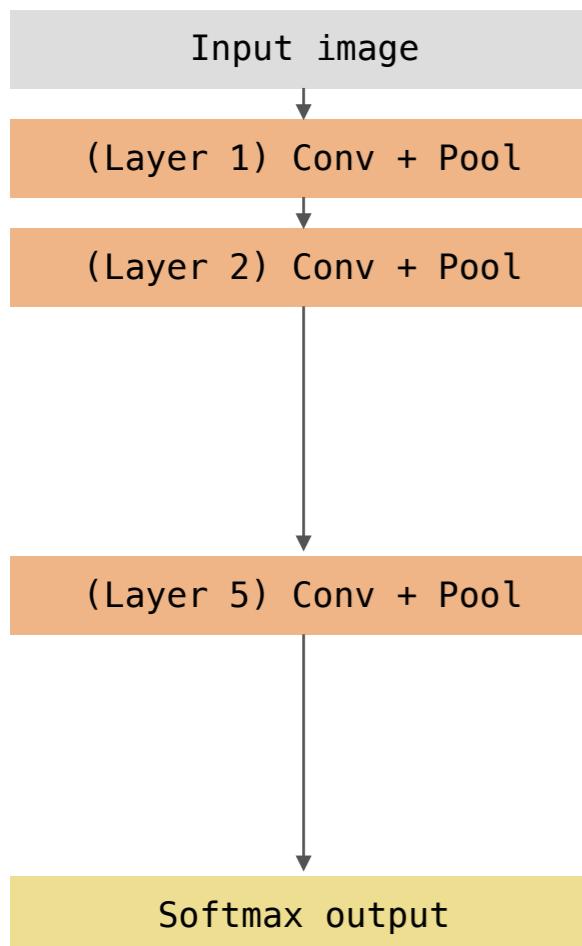
The importance of depth

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

Keras

What is Keras

The basic architecture

Keras API

TensorFlow / CNTK / MXNet / Theano / ...

GPU

CPU

TPU

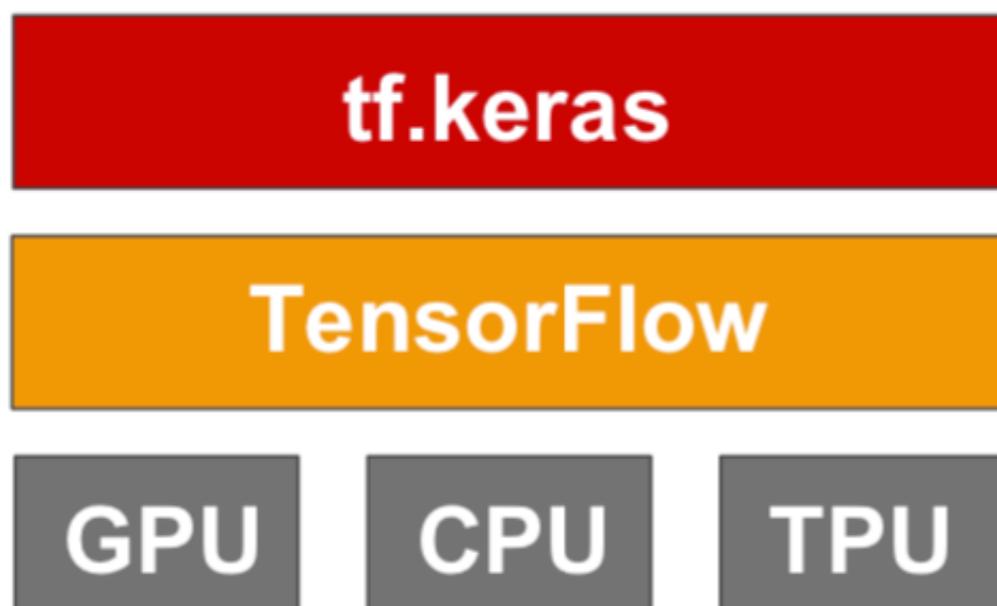
Source: Francois Chollet

Keras + Tensorflow

The official high-level API of Tensorflow

- 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

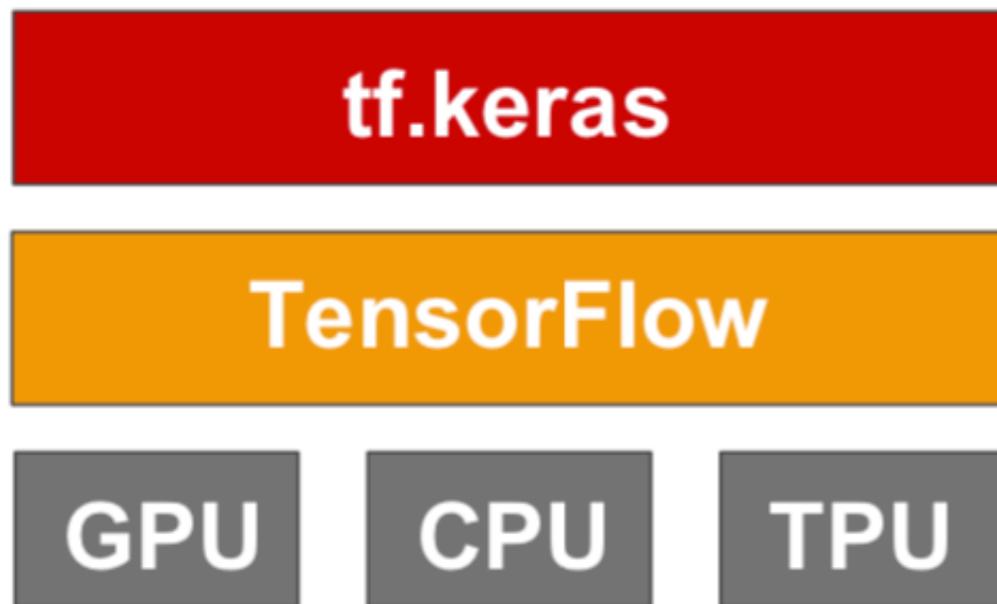


Source: Francois Chollet

Keras + Tensorflow

The official high-level API of
Tensorflow

- In tensorflow.keras, we have full Keras API
- Better optimized for TF
- Better integration with TF-specific features
- Estimator API, Eager execution, tf.Data



Source: Francois Chollet

Keras + Plaidml

For those who do not have NVIDIA GPU

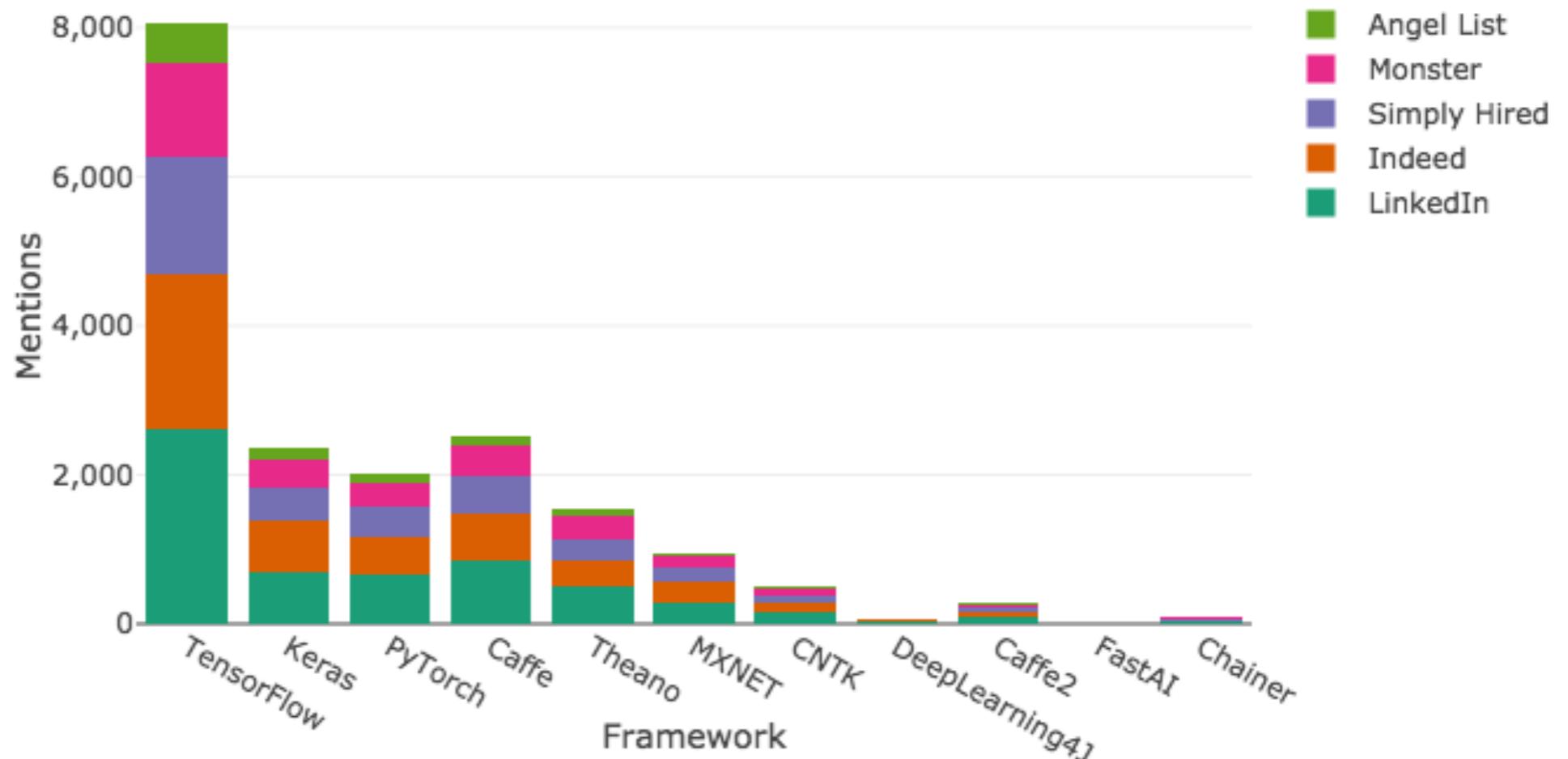
- Currently deep learning mostly runs on Nvidia GPU, as Nvidia has CUDA, performed better than OpenCL
- Still, there is a need / request to run deep learning on non-Nvidia GPU
- With Plaidml as the backend (instead of Tensorflow or Theano), now it is possible
- Vertex.AI, the startup that build Plaidml, was bought by Intel in 2018



Keras

The demand

Online Job Listings

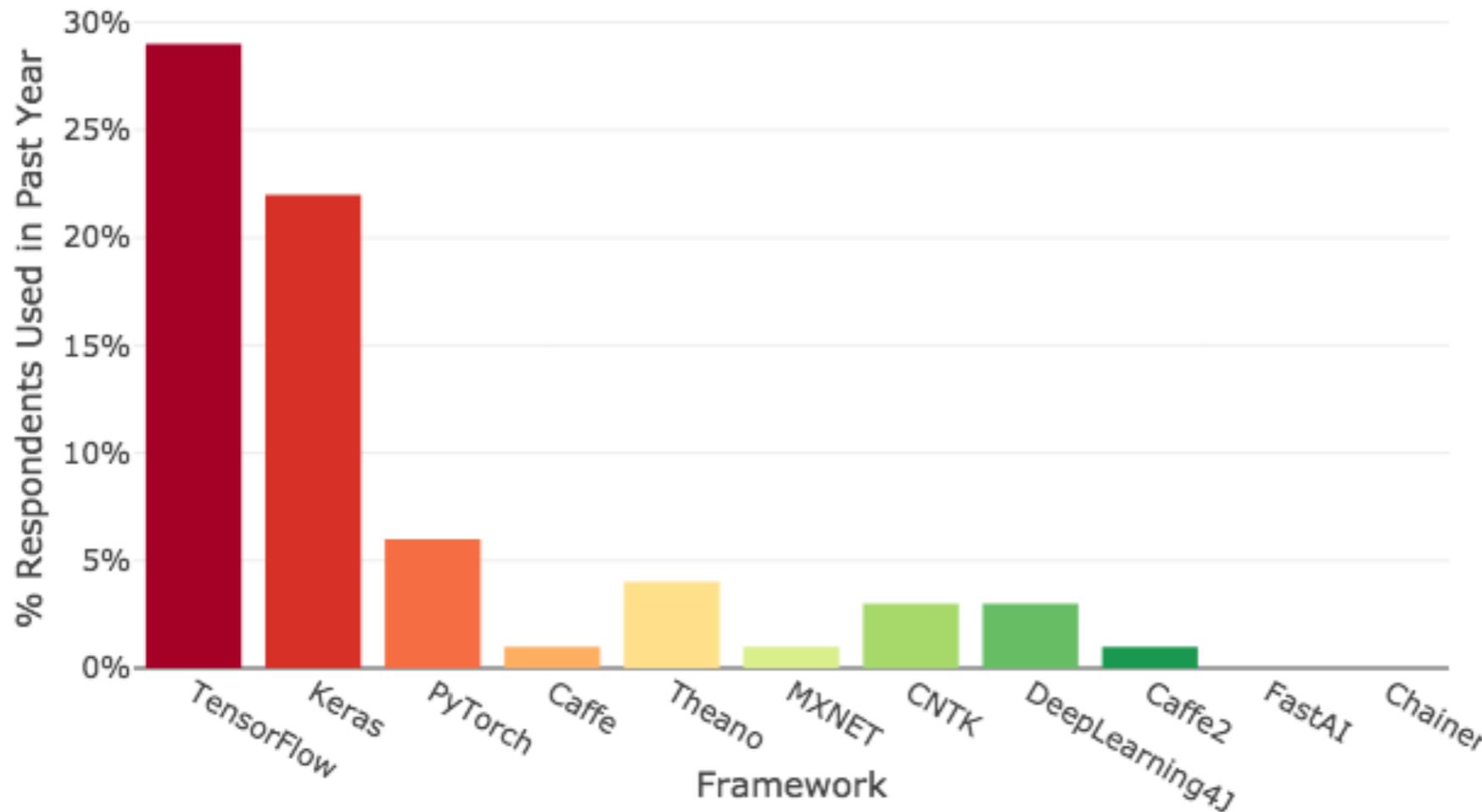


Source: <https://towardsdatascience.com/deep-learning-framework-power-scores-2018-23607ddf297a>

Keras

The demand

- KDnuggets, a popular data science website, polled data scientists around the world on the framework they used

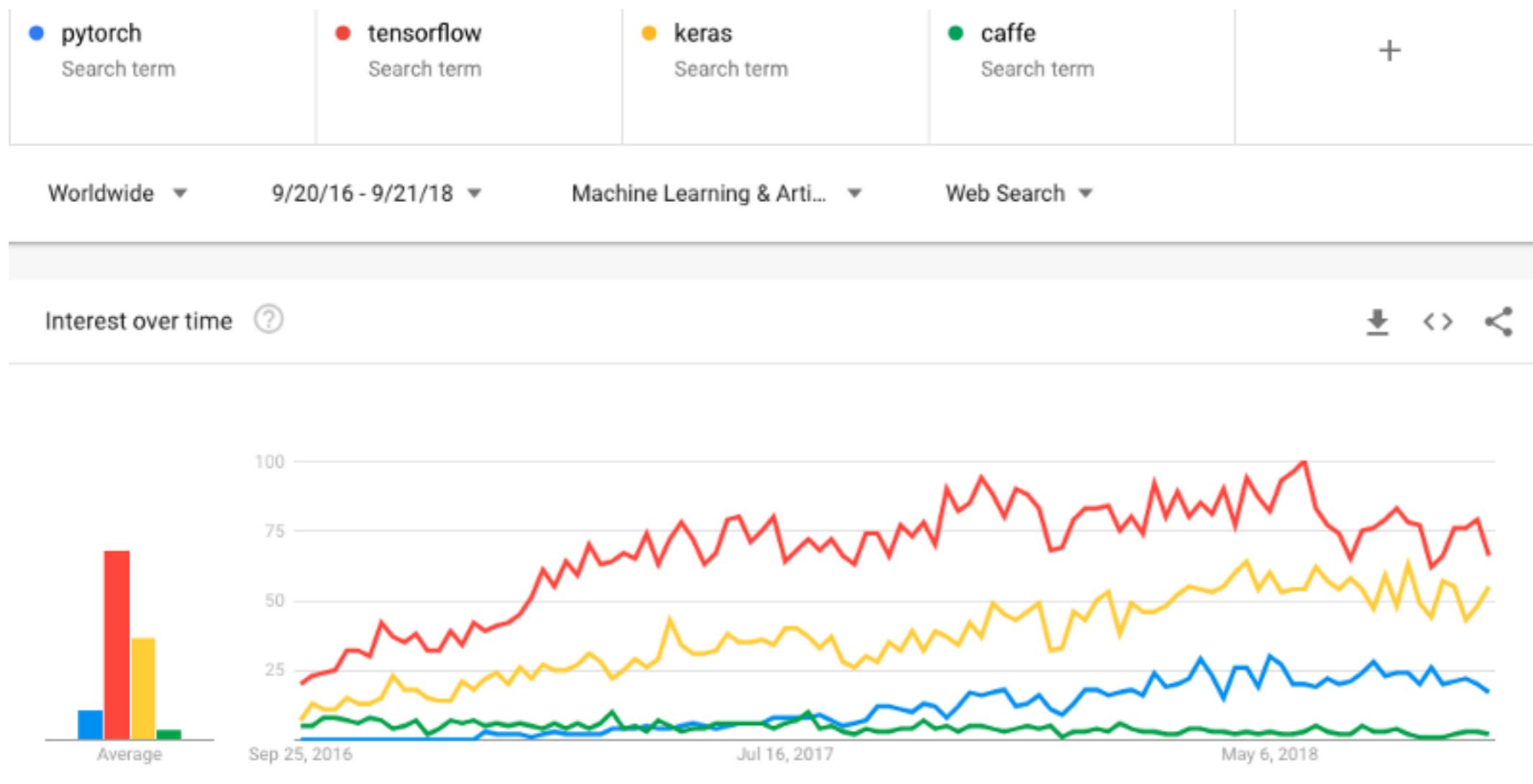


Source: <https://towardsdatascience.com/deep-learning-framework-power-scores-2018-23607ddf297a>

Keras

The demand

- Search in Google

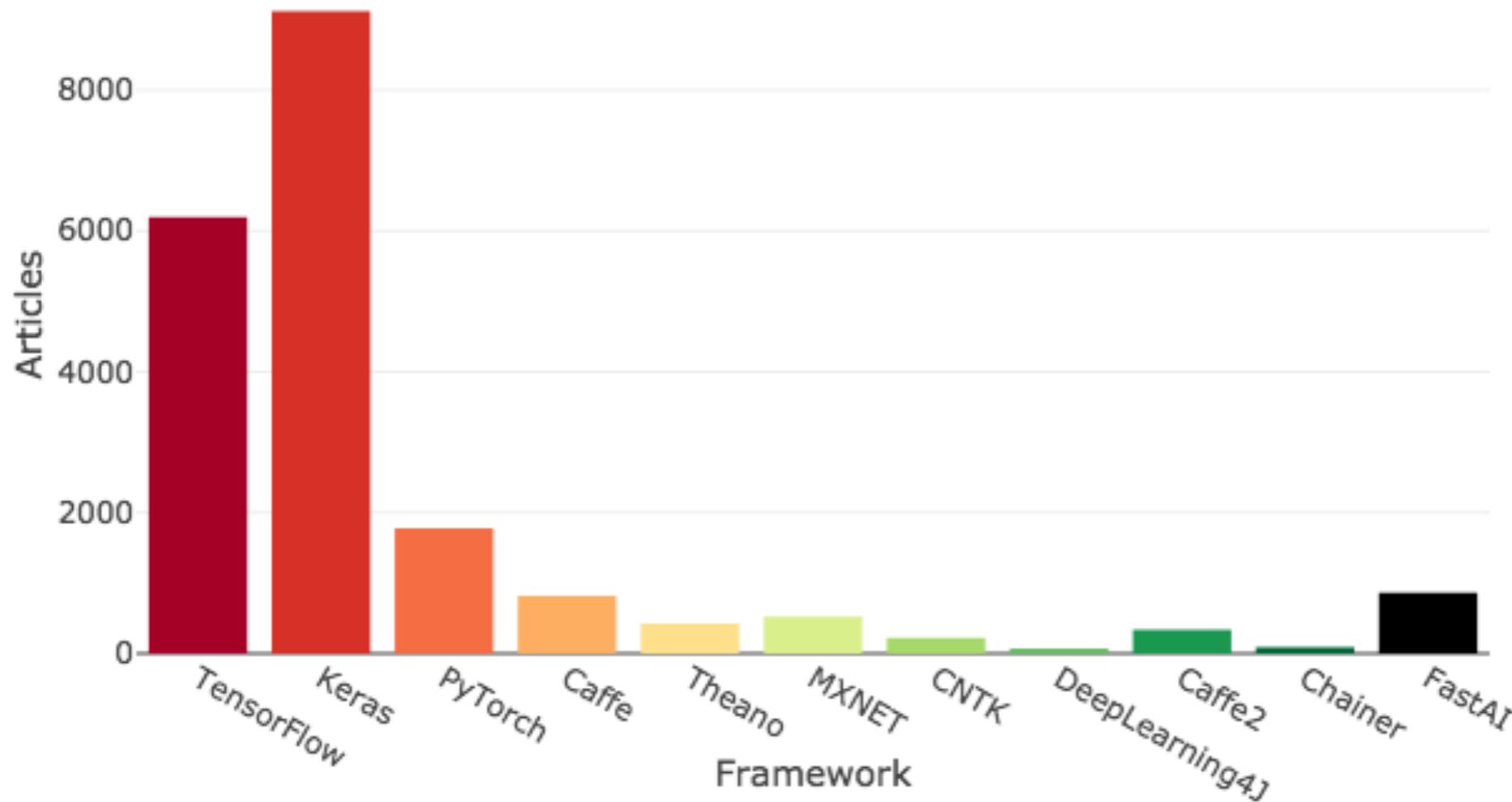


Source: <https://towardsdatascience.com/deep-learning-framework-power-scores-2018-23607ddf297a>

Keras

The demand

- Number of times mentioned in Medium articles

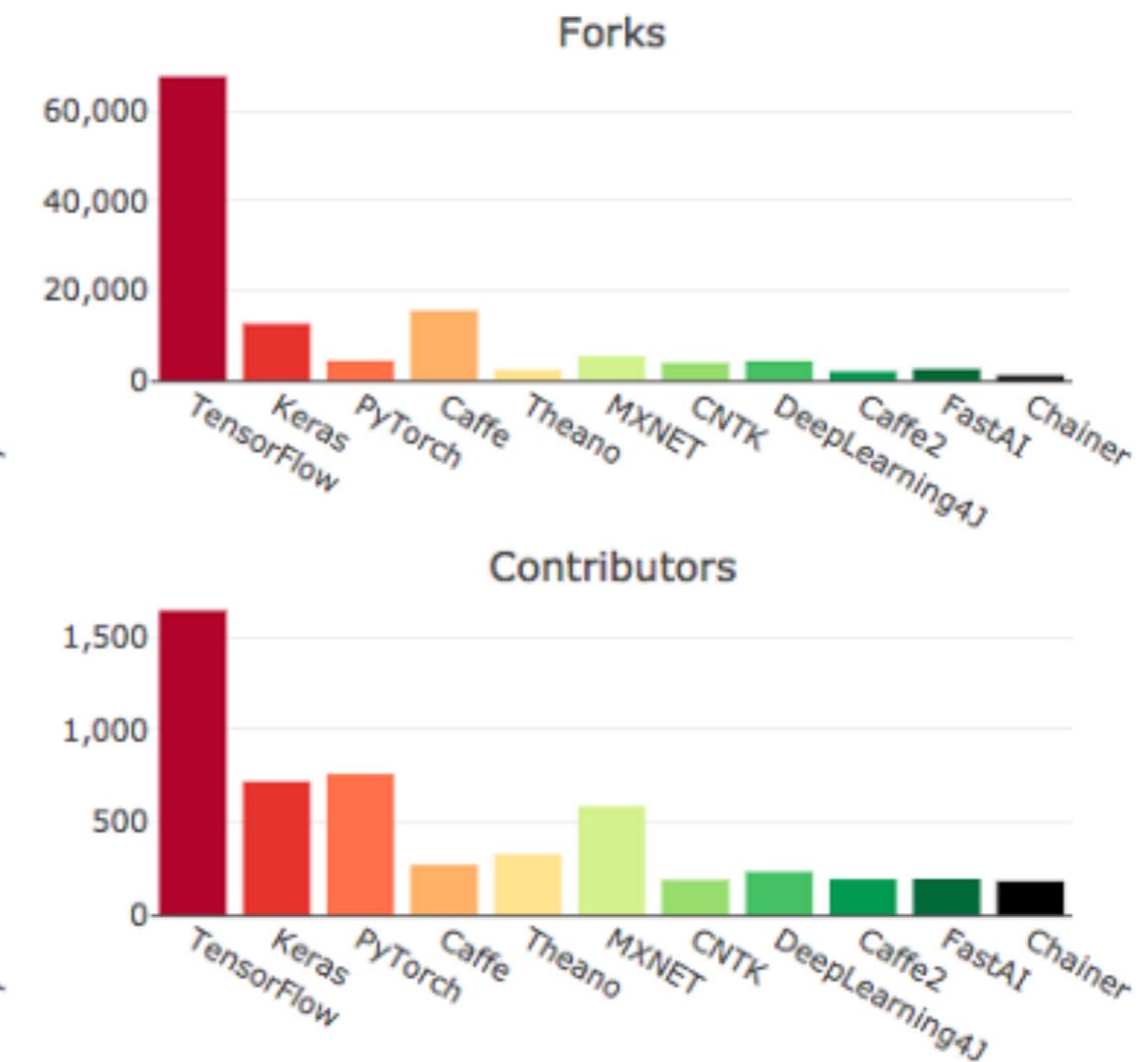
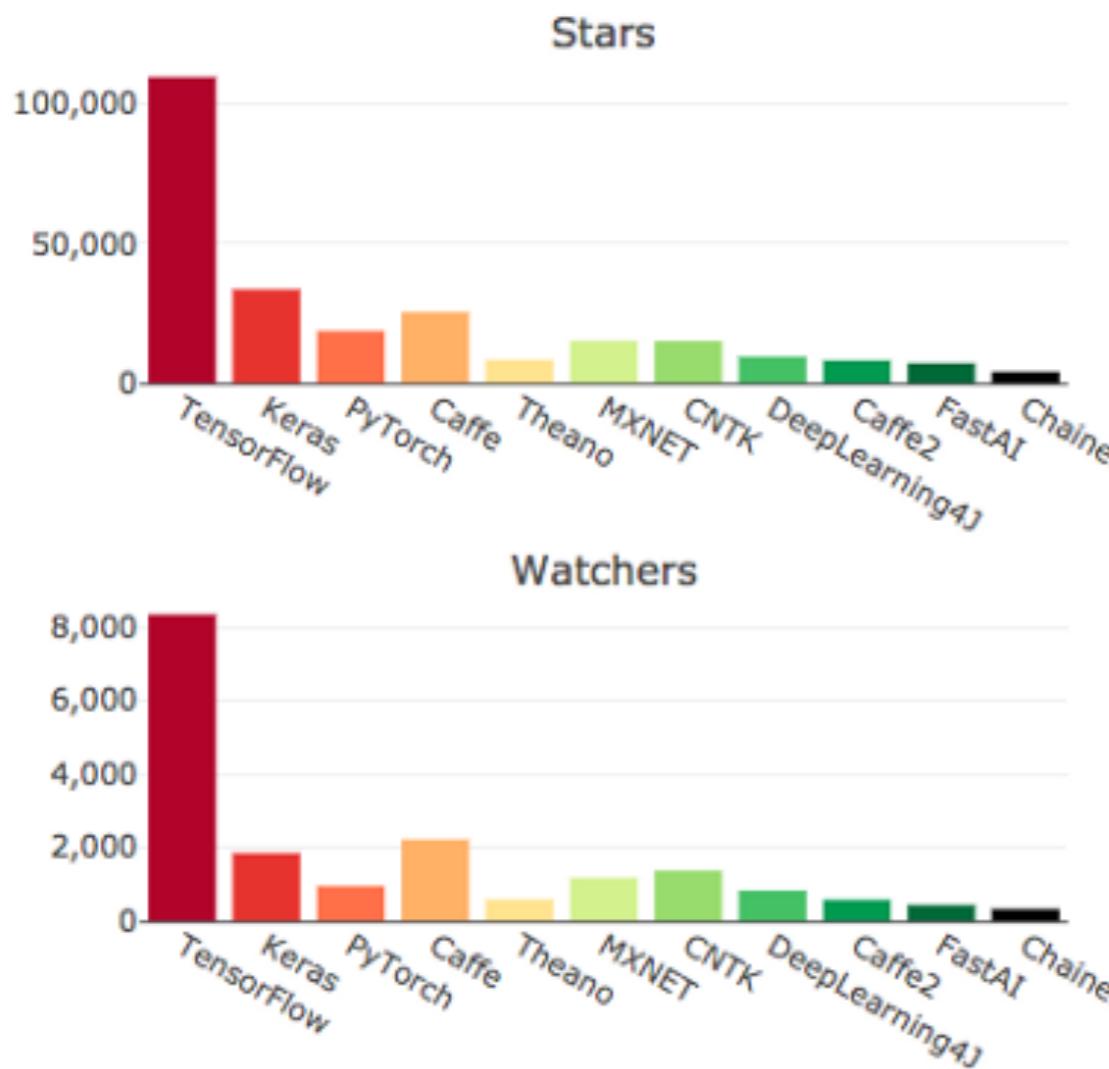


Source: <https://towardsdatascience.com/deep-learning-framework-power-scores-2018-23607ddf297a>

Keras

The demand

- Activities in Github



Source: <https://towardsdatascience.com/deep-learning-framework-power-scores-2018-23607ddf297a>

Keras

Future

- As of Q1 2019, 320,000 developers worldwide are using Keras (about 1 in 60 developers globally)
- Meanwhile, about 1,000,000 developers using machine learning in some form (about 1 developer in 20 globally)
- Projection: 500,000 by Q1 2020; 1,000,000 by Q1 2022



Source: <https://github.com/keras-team/governance>
Image: <https://www.kaggle.com>

Keras

The user experience

- Follow best practices for reducing cognitive load
- Offer consistent and simple APIs
- Minimize number of user actions required for common use cases
- Provide clear and actional feedback on user error

Source: Francois Chollet

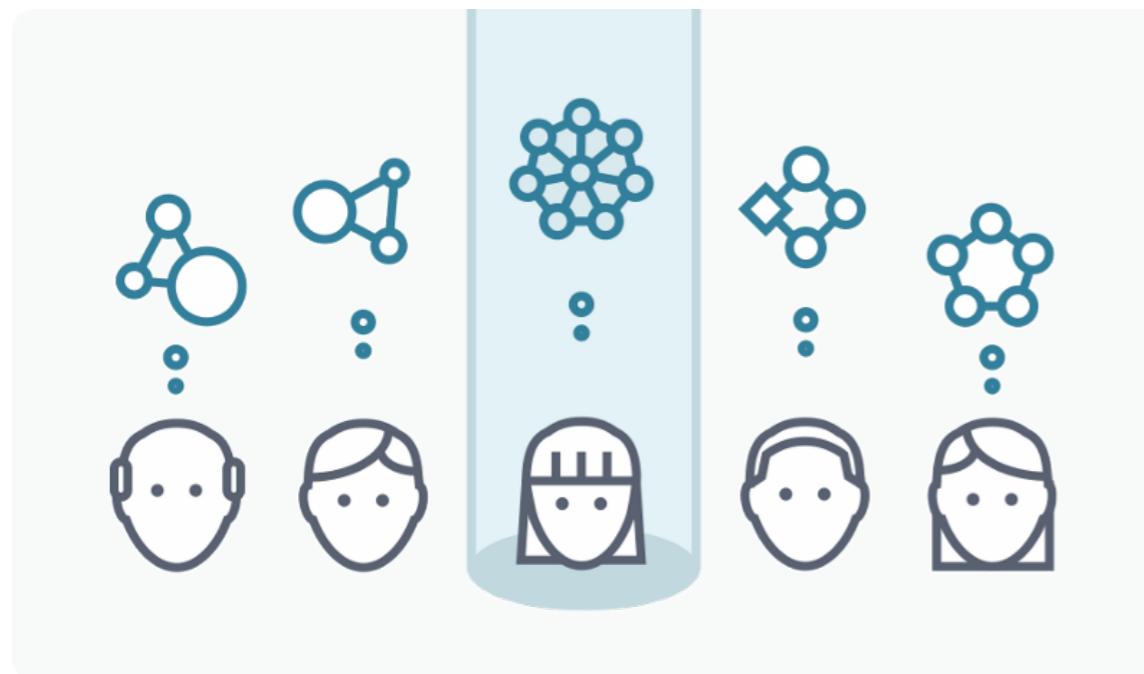
Keras

A higher chance of success

- Since the APIs are easy to use, developers are more productive, have more time to try more ideas

- They key to win competitions and further explorations!

- Ease of use does not come at the cost of reduced flexibility; Keras easily integrates with Tensorflow with tensorflow.keras



Source: Francois Chollet

Image: <https://medium.com/implodinggradients/should-you-kaggle-5b8dbdef442f>

Keras

Largest array of options for productizing models

- TF-Serving
- Within web browser, with GPU acceleration (WebKeras, Keras.js, WebDNN ...)
- Android (TF, TF Lite), iPhone (native CoreML support)
- Raspberry Pi
- JVM

Source: Francois Chollet

Keras

Keras + CoreML + ARKit in iOS

- Using the force



Source: <https://medium.com/s23nyc-tech/using-machine-learning-and-coreml-to-control-arkit-24241c894e3b>

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

```
> 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
```

- Adding layers by sequence (layer by layer)

- `input_shape` must be present in the first layer

Keras

Sequential model

- Create the model and show the model summary

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

Layer (type)	Output Shape	Param #
<hr/>		
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
<hr/>		
Total params: 283,077		
Trainable params: 283,077		
Non-trainable params: 0		
<hr/>		

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

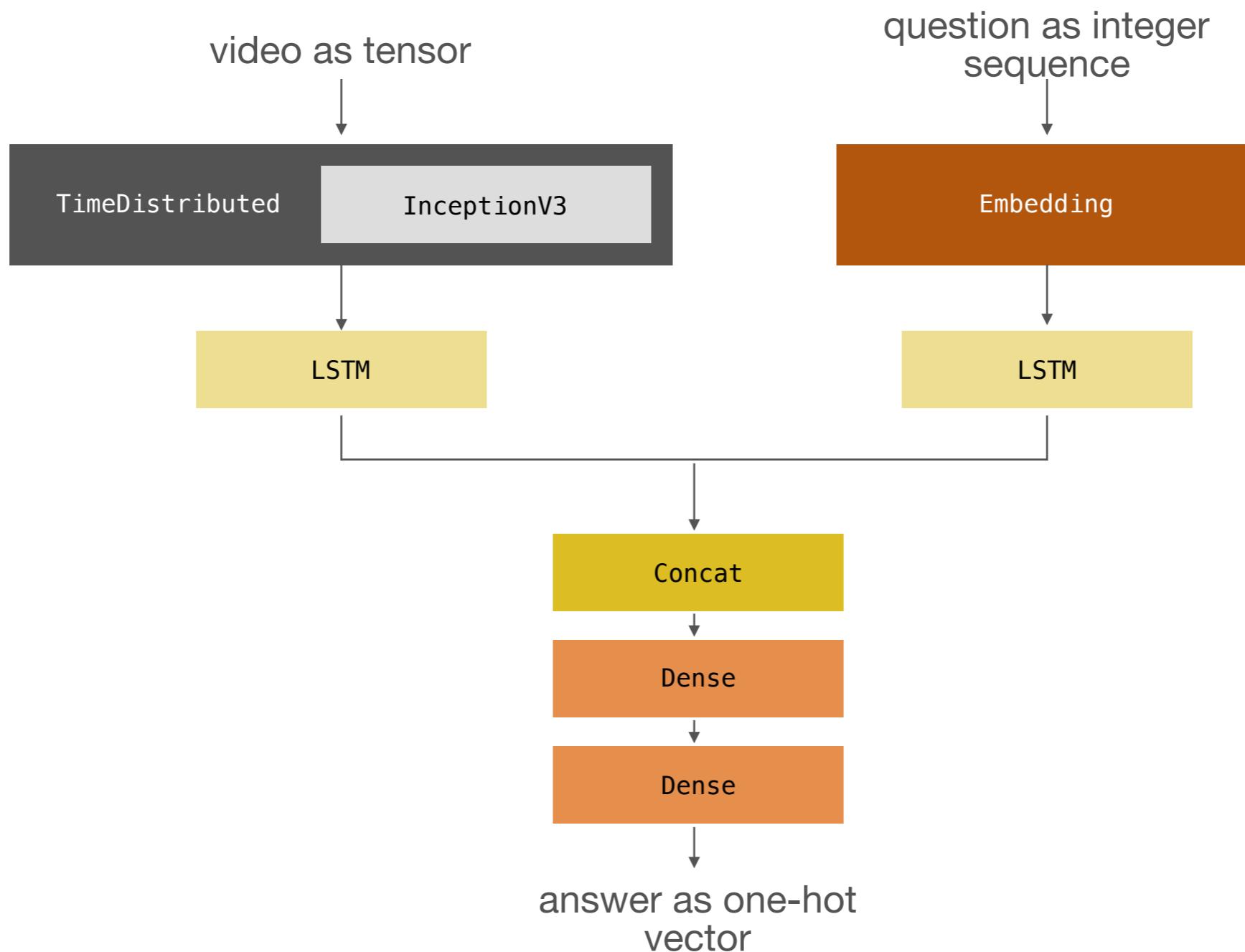
- "What is the person doing?"
"Tidying"
- "Who is the person?"
"Kondo"



Source: people.com

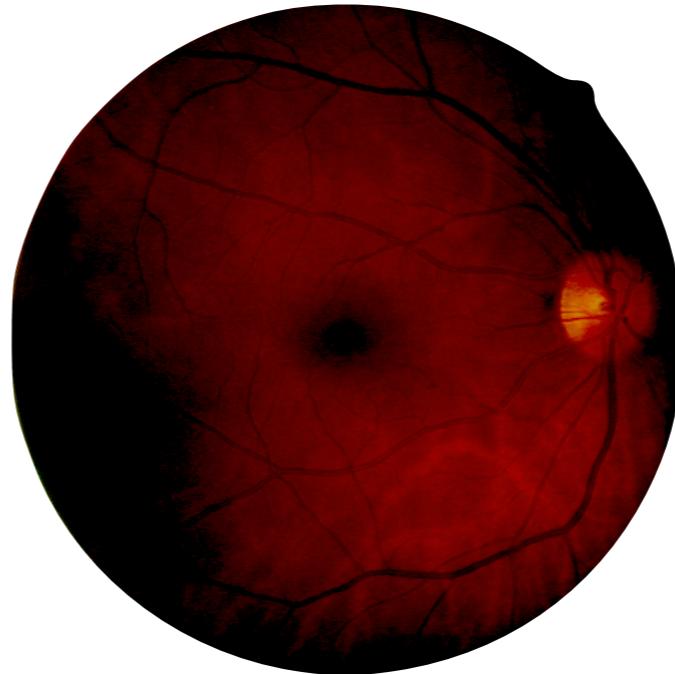
Multiple inputs

An video and a question



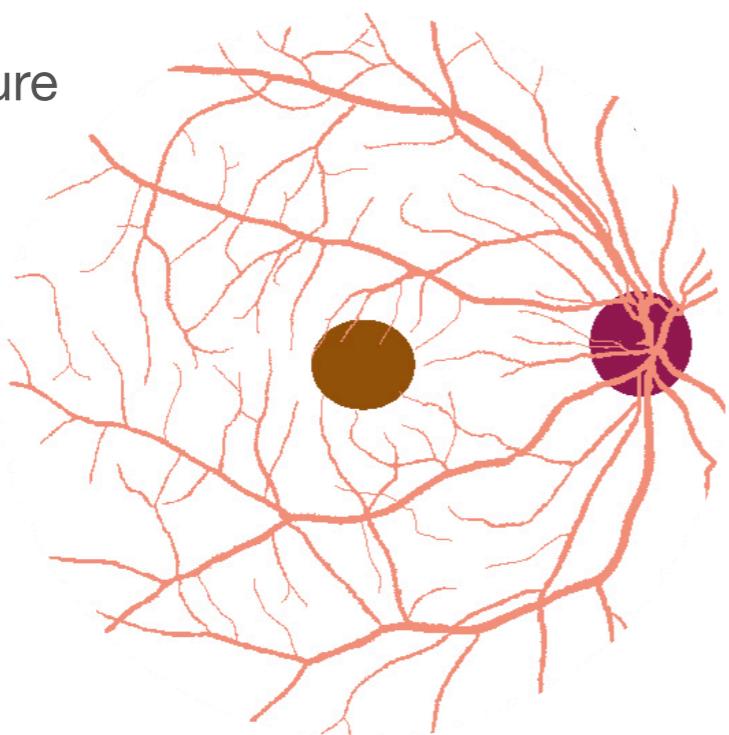
Multiple outputs?

An adjusted image and a segmentation



input image

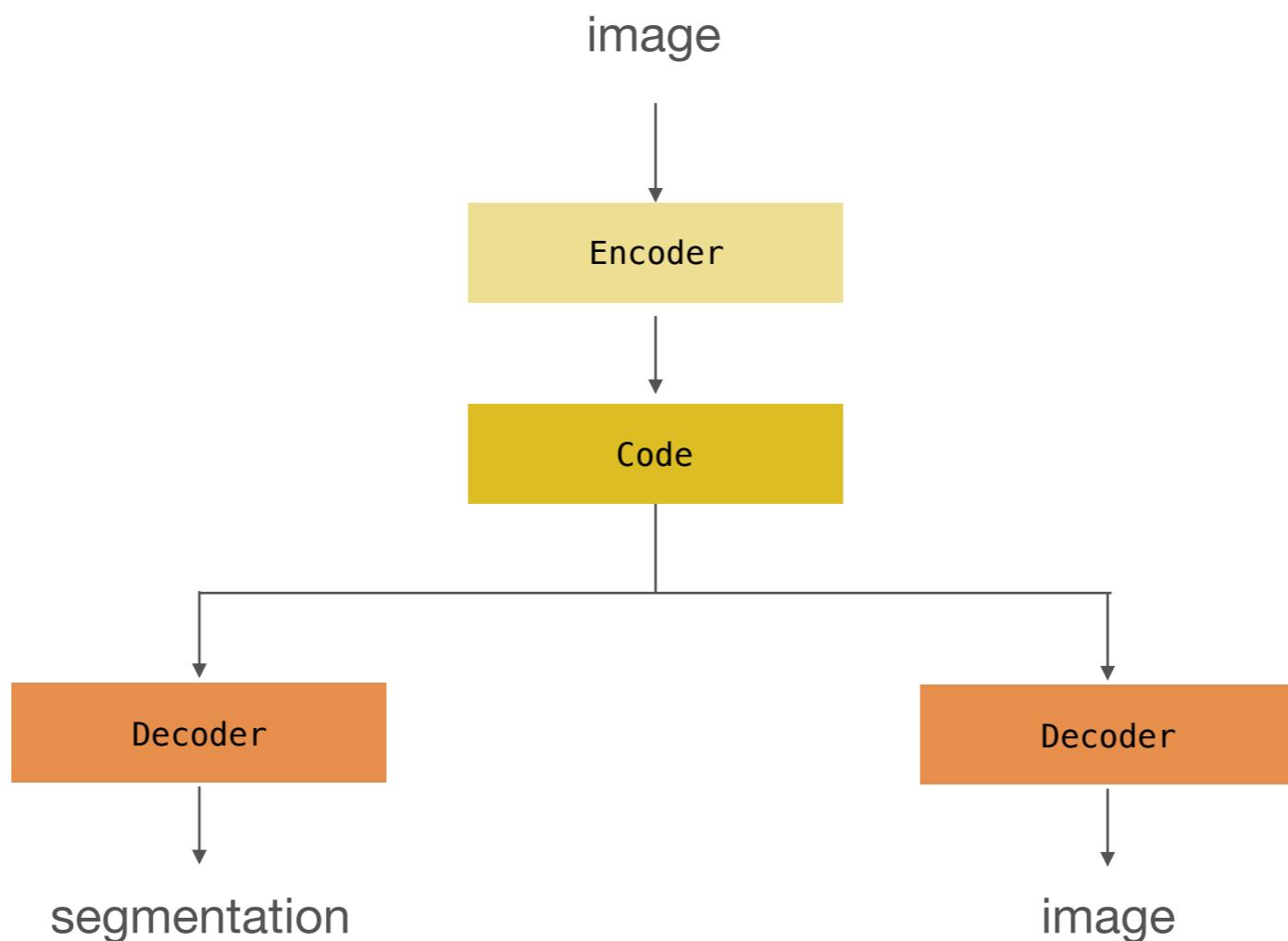
anatomy feature
segmented



adjusted output
image

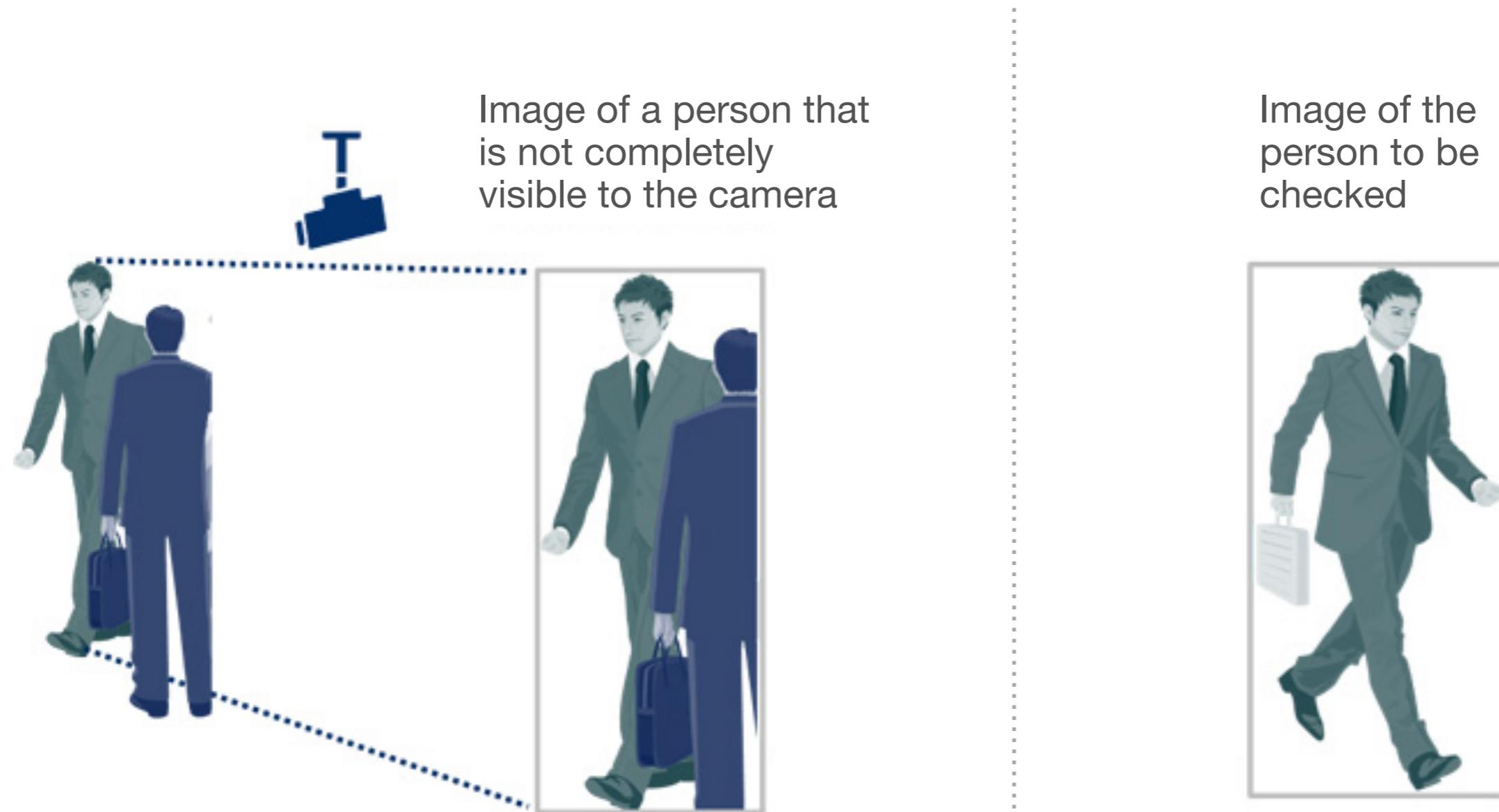
Multiple outputs

An adjusted image and a segmentation



Shared layers?

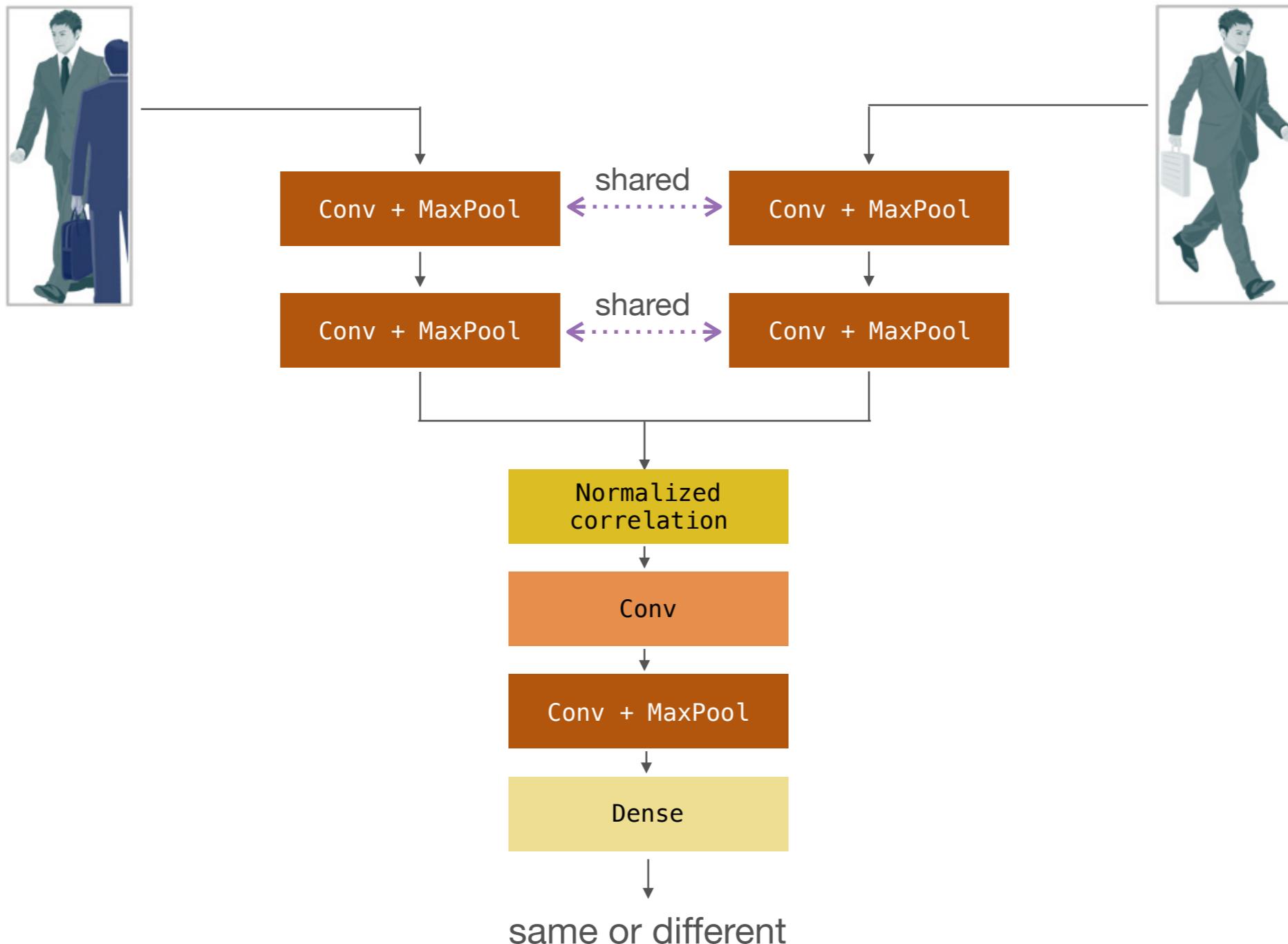
Person re-identification



Source: <https://yellrobot.com/nec-person-reidentification-technology-can-identify-body-shape/>

Shared layers

Person re-identification



Group discussion

Suggest a use case / scenario / application
where multiple inputs or multiple outputs are
required for a deep net

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
```

- Re-build the previous mode using functional APIs

- For single input and single output, the code for `model.fit` remains the same as before

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

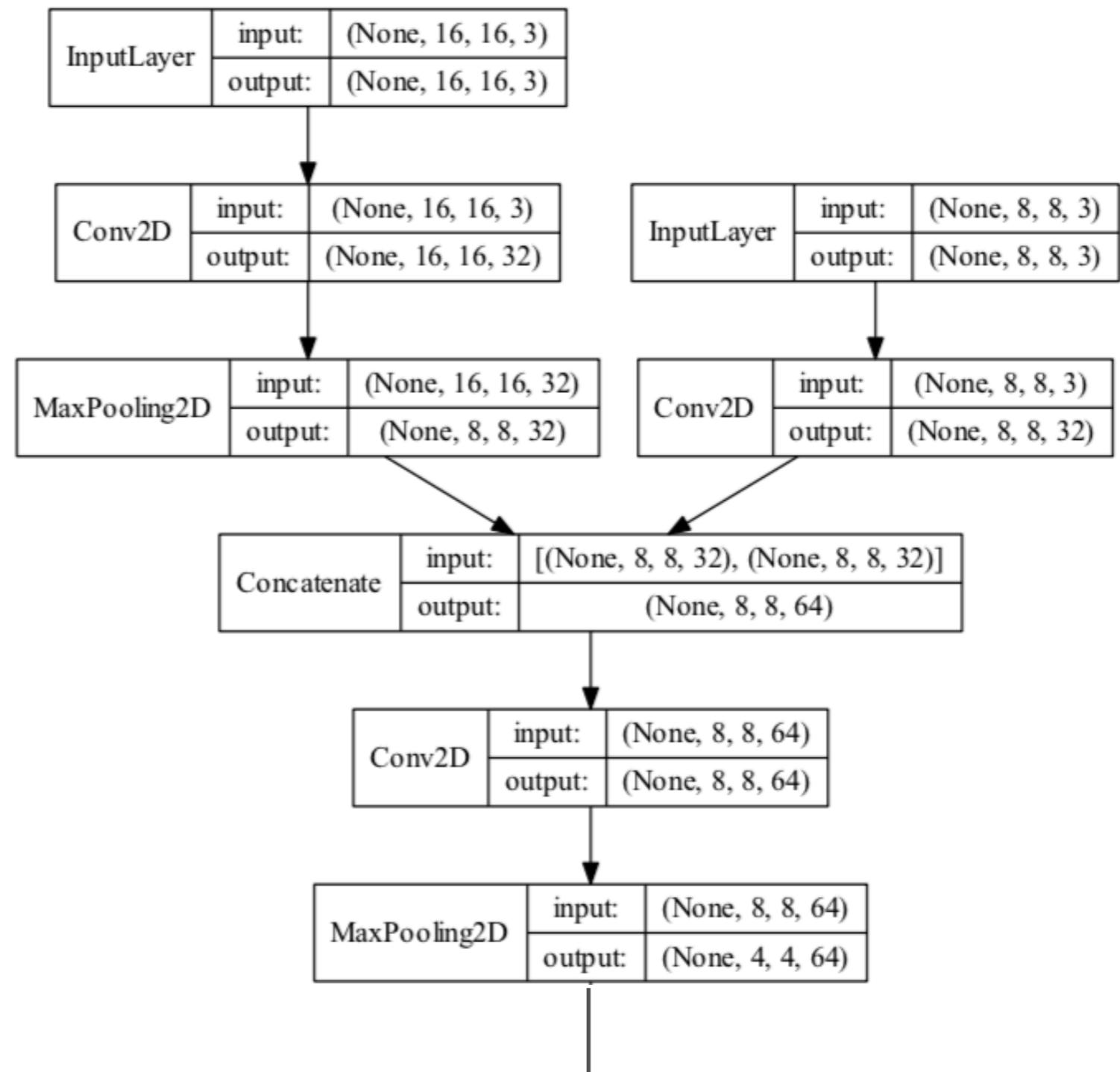
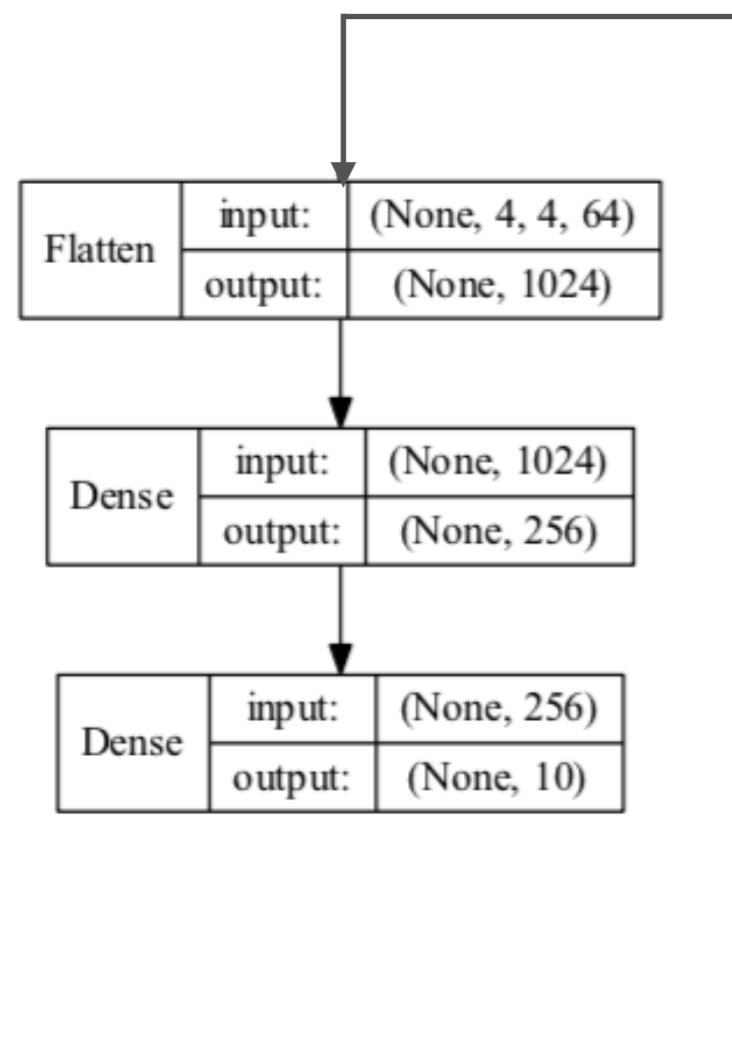
```
> def createSeqModel():
    model  = Sequential()
    model.add(Conv2D(32,(3,3),
                    input_shape=(16,16,3),
                    padding='same',
                    activation='relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Conv2D(64,(3,3),
                    padding='same',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



'Y' shape architecture

the code

- Need to include the below

```
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
<hr/>			
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]
<hr/>			
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( [LDat,RDat],  
             TLbl,  
             validation_data=( [vLDat,vRDat], vLbl),  
             epochs=100,  
             batch_size=128,  
             shuffle=True,  
             callbacks=callbacks_list)
```

Multiple inputs

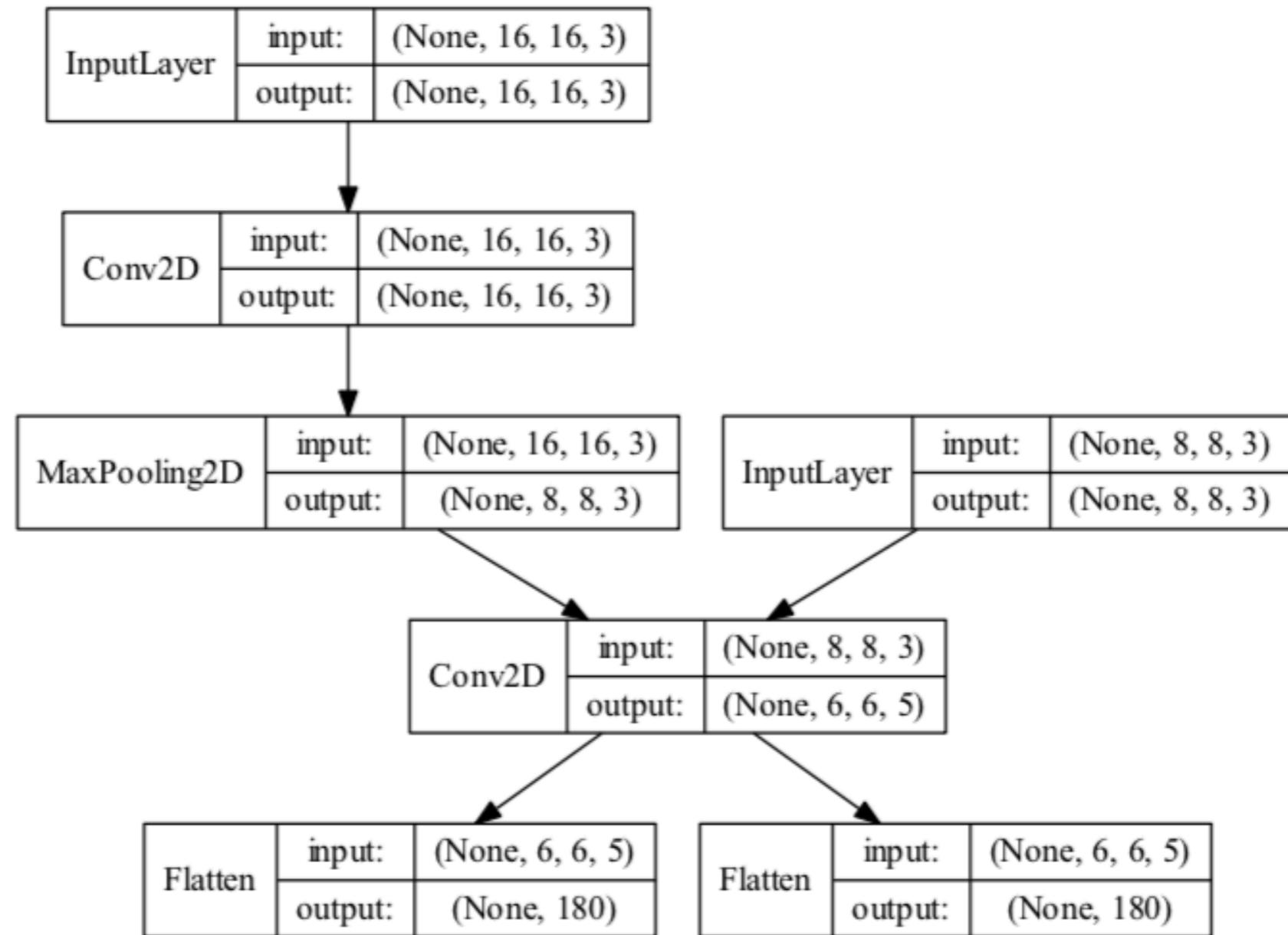
Getting the model plot

- 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

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

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

    Rin = Input(shape=(8,8,3))
    Rx = shared(Rin)
    Rx = Flatten()(Rx)

    model = Model(inputs=[Lin,Rin],outputs=[Lx,Rx])
    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)
```

Short exercise

Build the model using functional APIs