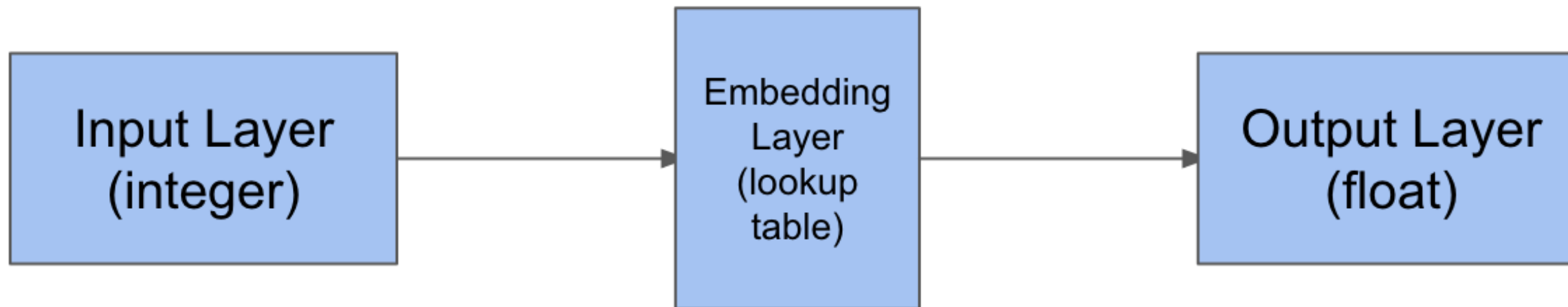ADVANCED DEEP LEARNING WITH KERAS IN PYTHON

Category embeddings

Zach Deane Mayer

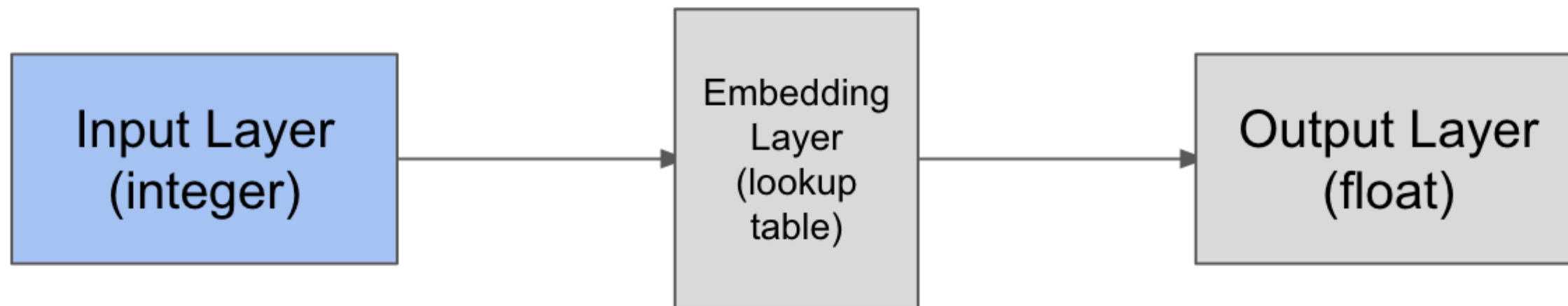Data Scientist

Category embeddings

- Input: integers

- Output: floats

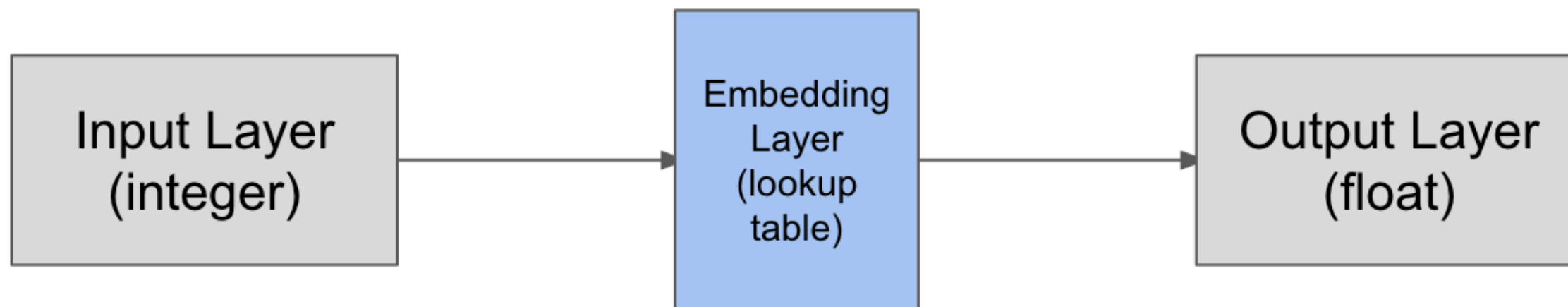- Note: Increased dimensionality: output layer flattens back to 2D

Inputs

```
input_tensor = Input(shape=(1,))
```
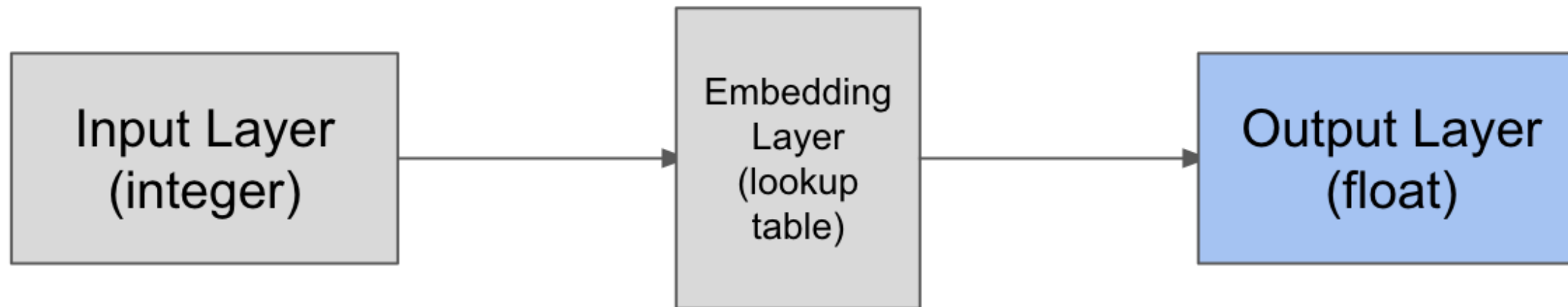
Embedding Layer

```python
from keras.layers import Embedding
input_tensor = Input(shape=(1,))
n_teams = 10887
embed_layer = Embedding(input_dim=n_teams,
                        input_length=1,
                        output_dim=1,
                        name='Team-Strength-Lookup')
embed_tensor = embed_layer(input_tensor)
```

Flattening

```python
from keras.layers import Flatten
flatten_tensor = Flatten()(embed_tensor)
```

Put it all together

```python
input_tensor = Input(shape=(1,))
n_teams =  10887
embed_layer = Embedding(input_dim=n_teams,
                        input_length=1,
                        output_dim=1,
                        name='Team-Strength-Lookup')
embed_tensor = embed_layer(input_tensor)
flatten_tensor = Flatten()(embed_tensor)
model = Model(input_tensor, flatten_tensor)
```
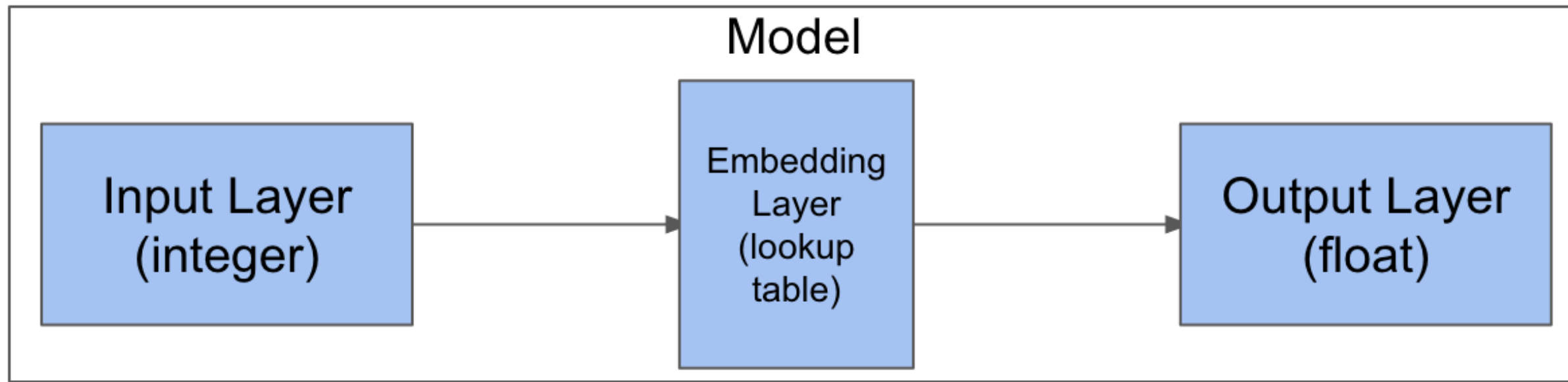
ADVANCED DEEP LEARNING WITH KERAS IN PYTHON

Let's practice!

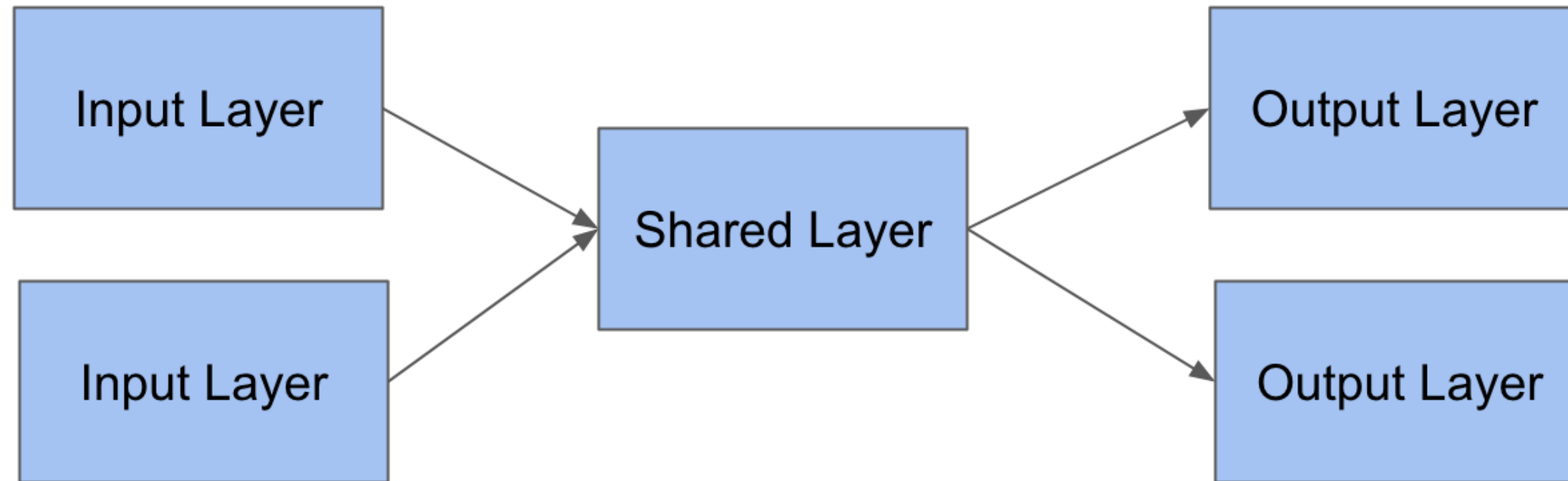ADVANCED DEEP LEARNING WITH KERAS IN PYTHON

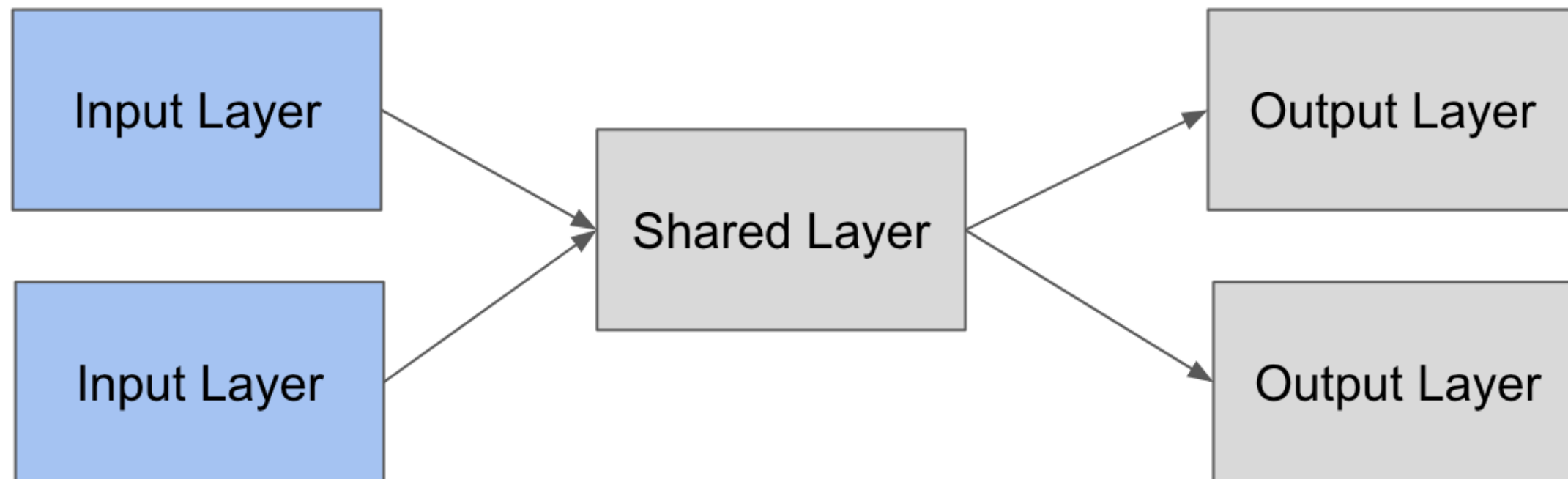# Shared layers

Zach Deane Mayer

Data Scientist

Shared layers

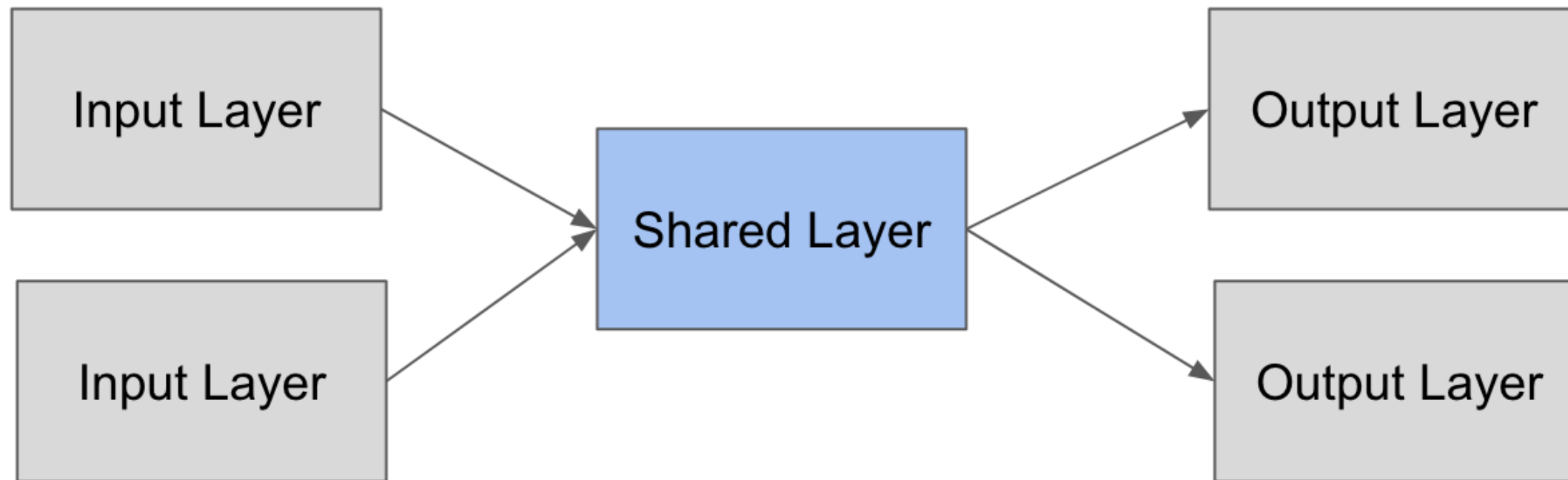- Require the functional API

- Very flexible

Shared layers

```python
input_tensor_1 = Input((1,))
input_tensor_2 = Input((1,))
```

Shared layers

```python
shared_layer = Dense(1)
output_tensor_1 = shared_layer(input_tensor_1)
output_tensor_2 = shared_layer(input_tensor_2)
```

Sharing multiple layers as a model

```python
input_tensor = Input(shape=(1,))
n_teams = 10887
embed_layer = Embedding(input_dim=n_teams,
                        input_length=1,
                        output_dim=1,
                        name='Team-Strength-Lookup')
embed_tensor = embed_layer(input_tensor)
flatten_tensor = Flatten()(embed_tensor)
model = Model(input_tensor, flatten_tensor)
```
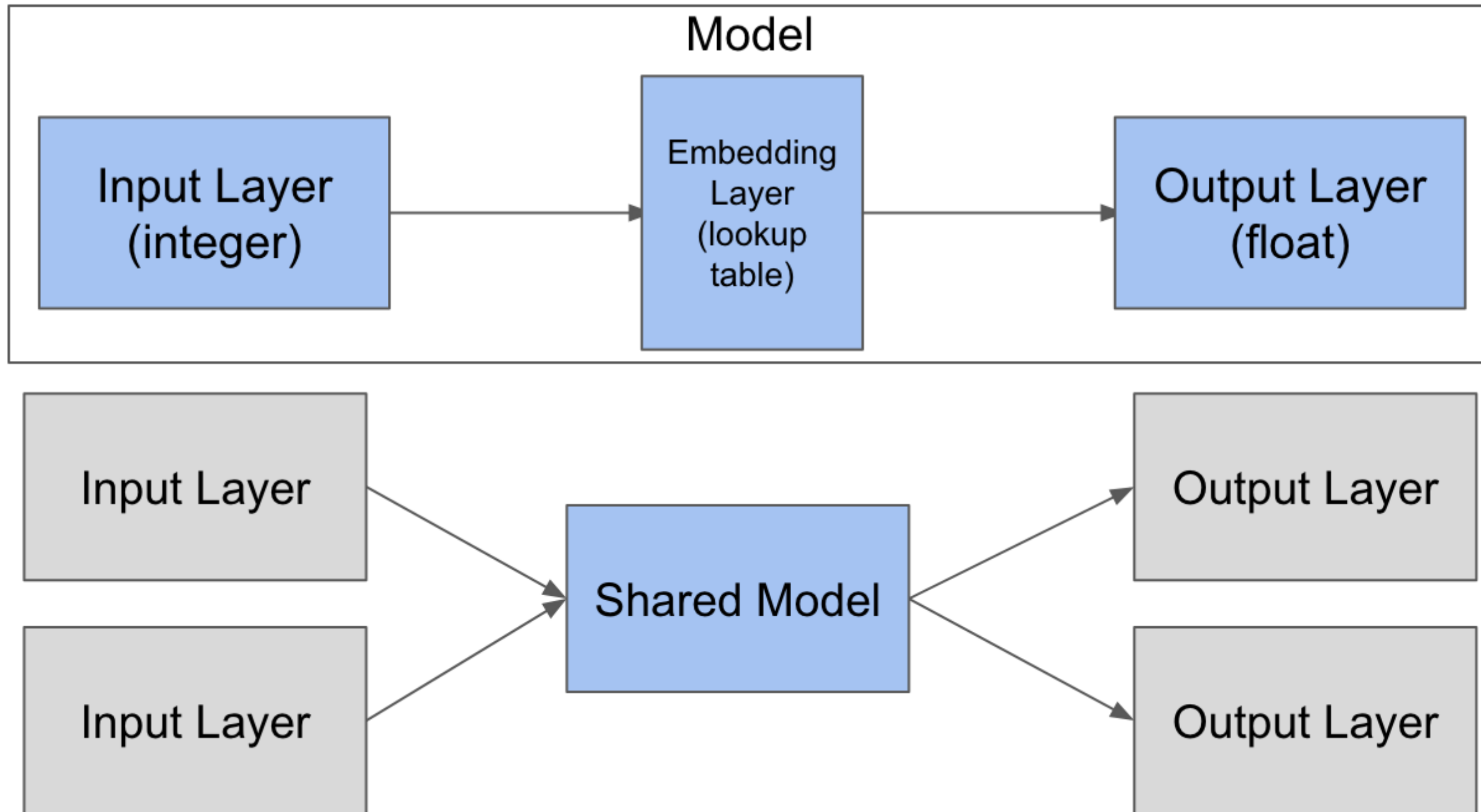
```python
input_tensor_1 = Input((1,))
input_tensor_2 = Input((1,))
output_tensor_1 = model(input_tensor_1)
output_tensor_2 = model(input_tensor_2)
```

Sharing multiple layers as a model

ADVANCED DEEP LEARNING WITH KERAS IN PYTHON

Let's practice!

ADVANCED DEEP LEARNING WITH KERAS IN PYTHON
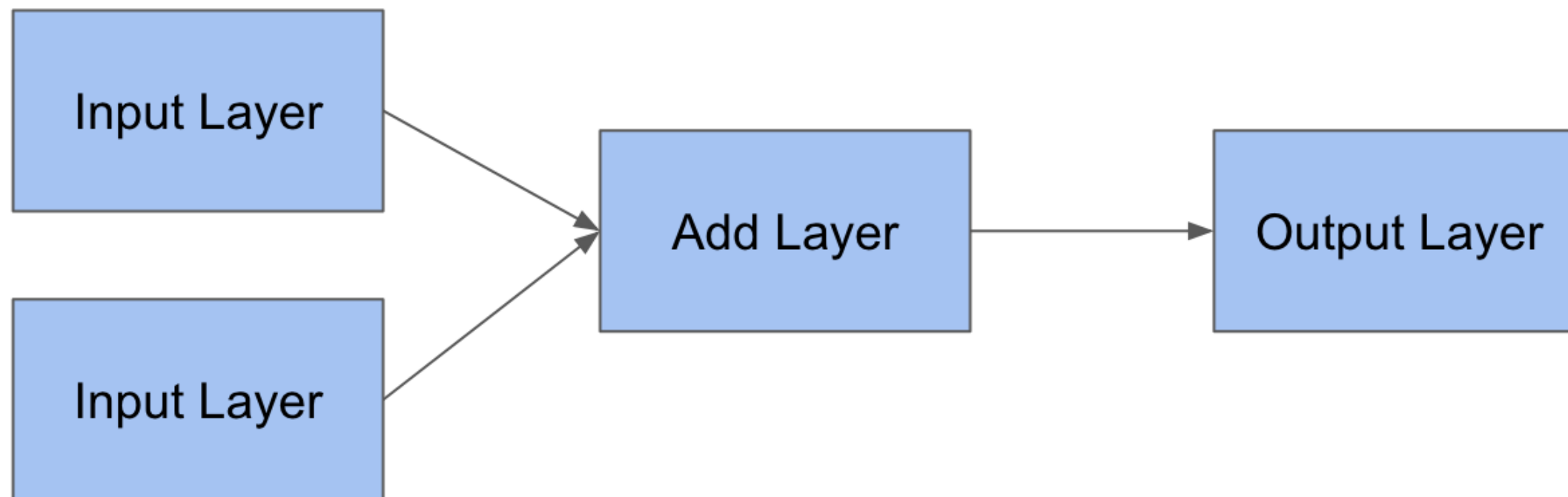
Merge layers

Zach Deane Mayer

Data Scientist

Merge layers

- Add

- Subtract

- Multiply

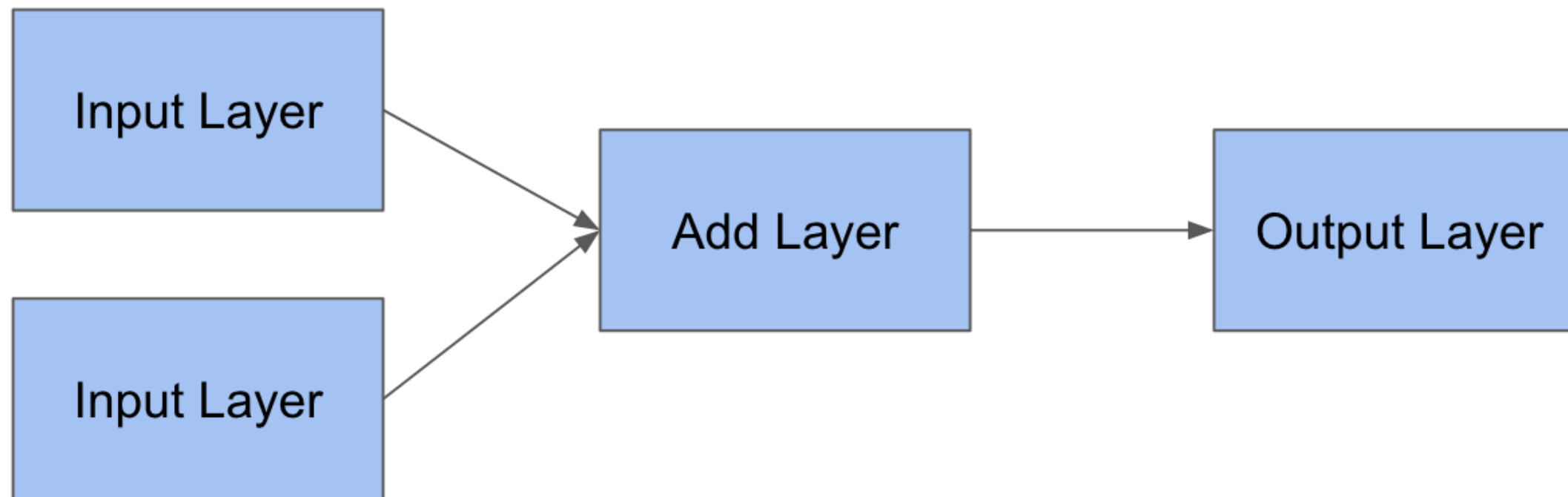- Concatenate

Merge layers

```python
from keras.layers import Input, Add
in_tensor_1 = Input((1,))
in_tensor_2 = Input((1,))
out_tensor = Add()([in_tensor_1, in_tensor_2])
```
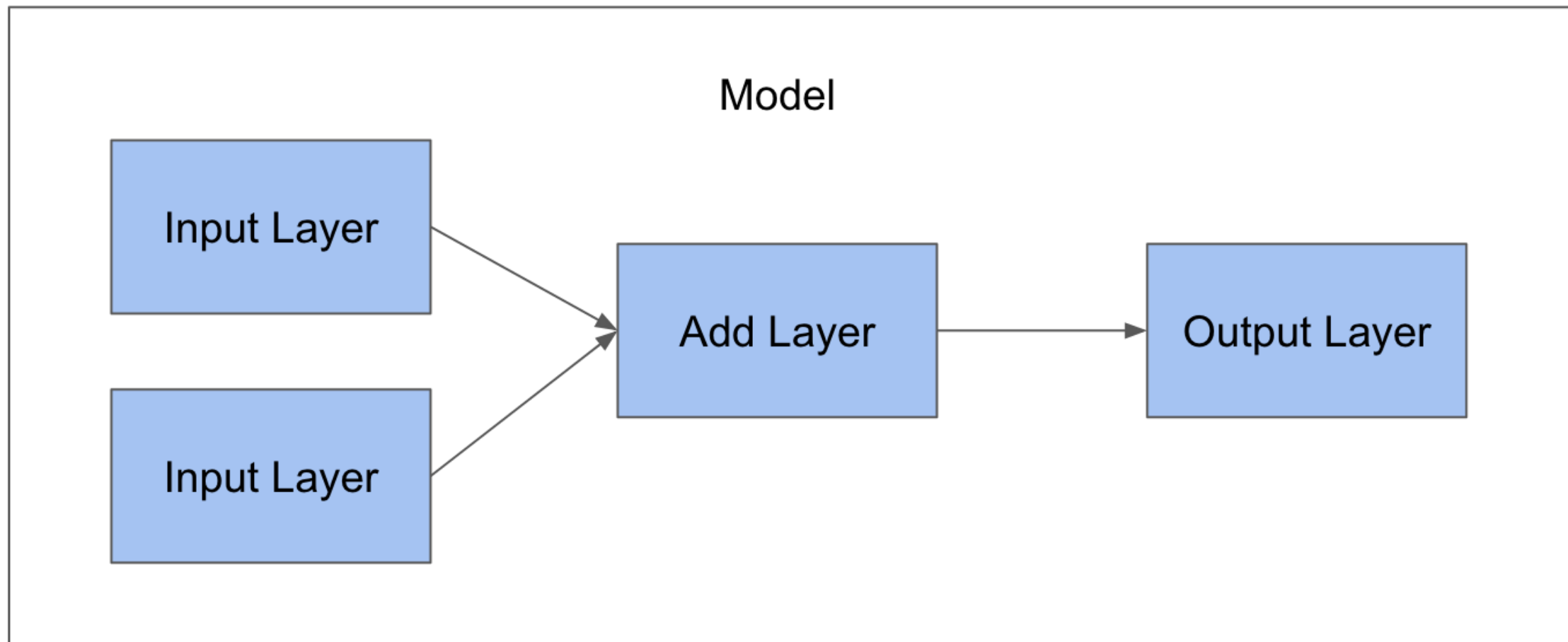
Merge layers

```python
in_tensor_3 = Input((1,))
out_tensor = Add()([in_tensor_1, in_tensor_2, in_tensor_3]
```
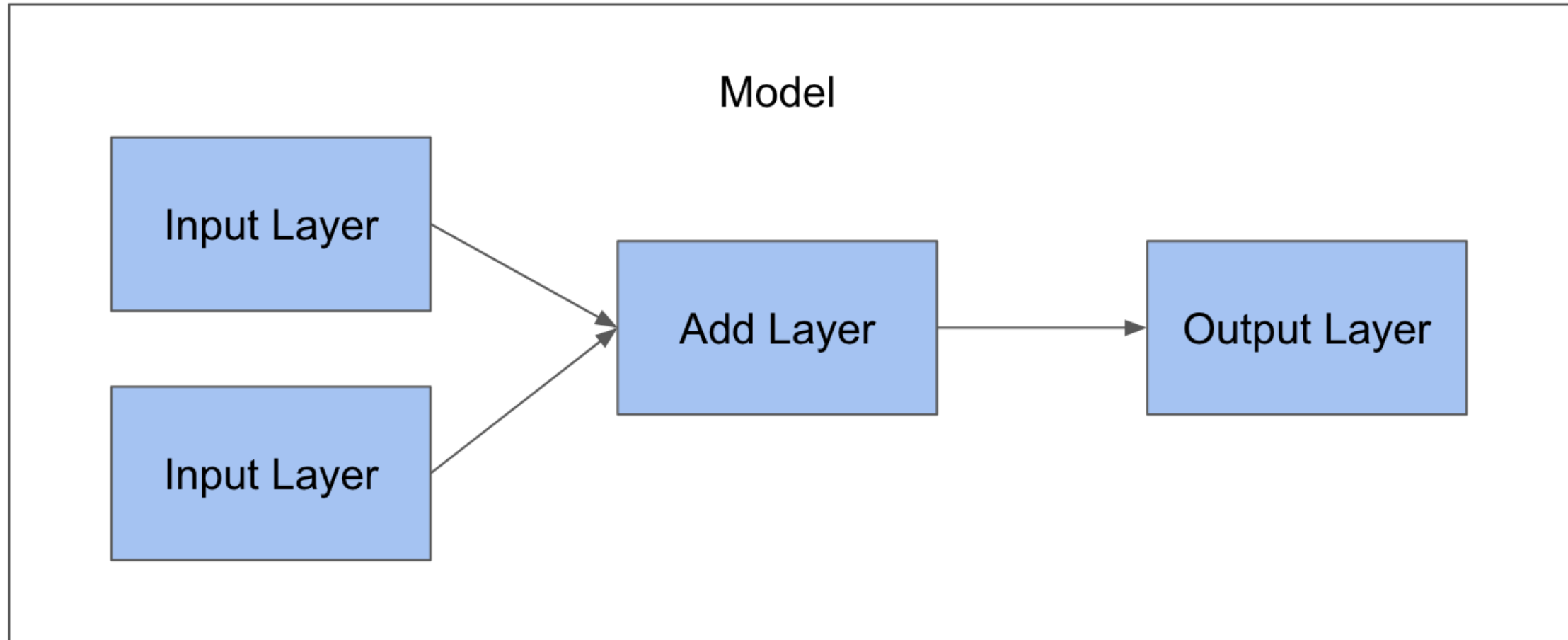
Create the model

```python
from keras.models import Model
model = Model([in_tensor_1, in_tensor_2], out_tensor)
```

Compile the model

```
model.compile(optimizer='adam', loss='mean_absolute_error')
```

ADVANCED DEEP LEARNING WITH KERAS IN PYTHON
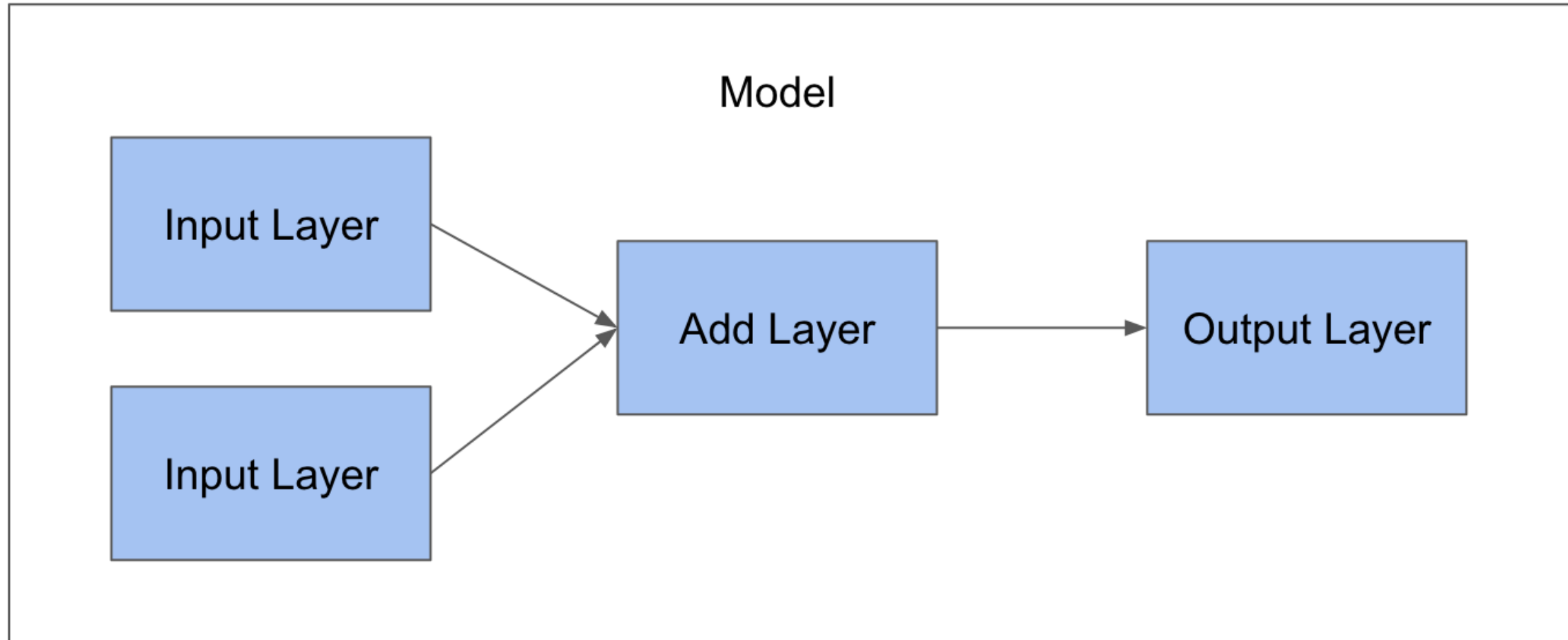
Let's practice!

ADVANCED DEEP LEARNING WITH KERAS IN PYTHON

Fitting and Predicting with multiple inputs

Zach Deane Mayer

Data Scientist

Fit with multiple inputs

```
model.fit([data_1, data_2], target)
```

Predict with multiple inputs

```
model.predict([np.array([[1]]), np.array([[2]])])
array([[3.]], dtype=float32)
```

```
model.predict([np.array([[42]]), np.array([[119]])])
array([[161.]], dtype=float32)
```
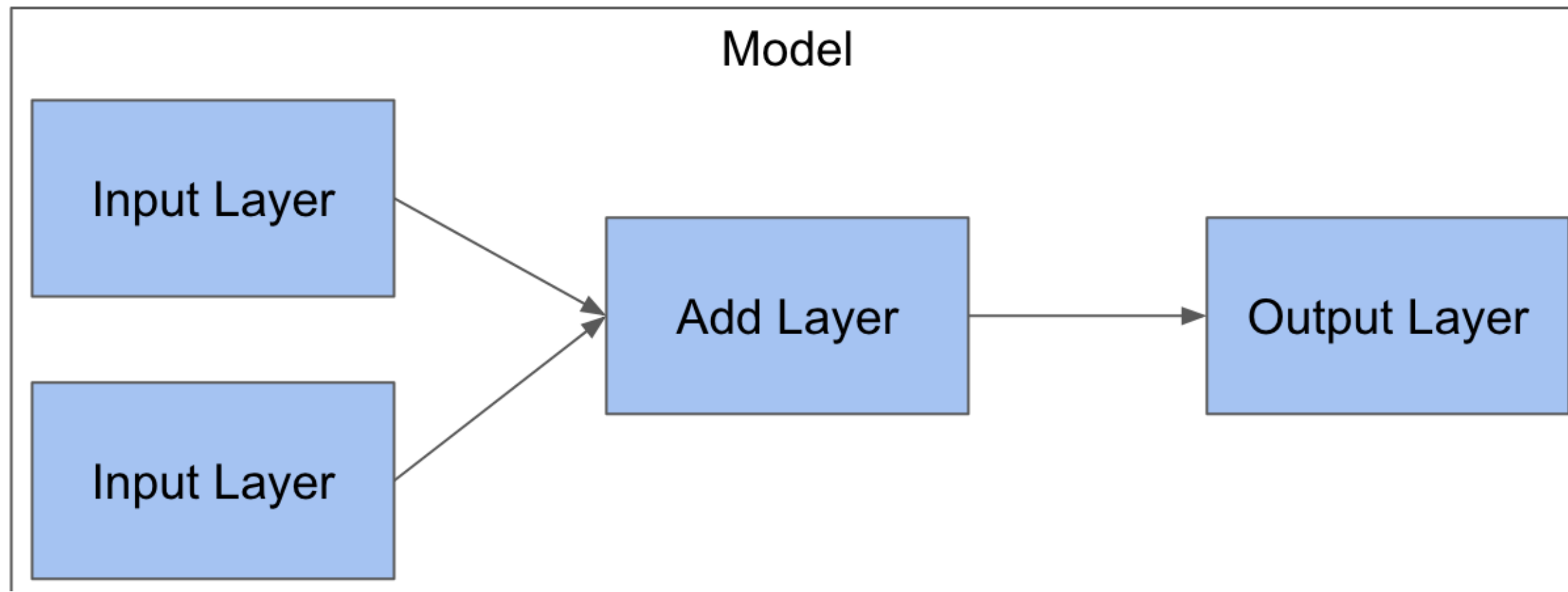
Evaluate with multiple inputs

```
model.evaluate([np.array([[-1]]), np.array([[-2]])], np.ar
```

```
1/1 [==============================] - 0s 801us/step
Out[21]: 0.0
```

ADVANCED DEEP LEARNING WITH KERAS IN PYTHON

Let's practice!