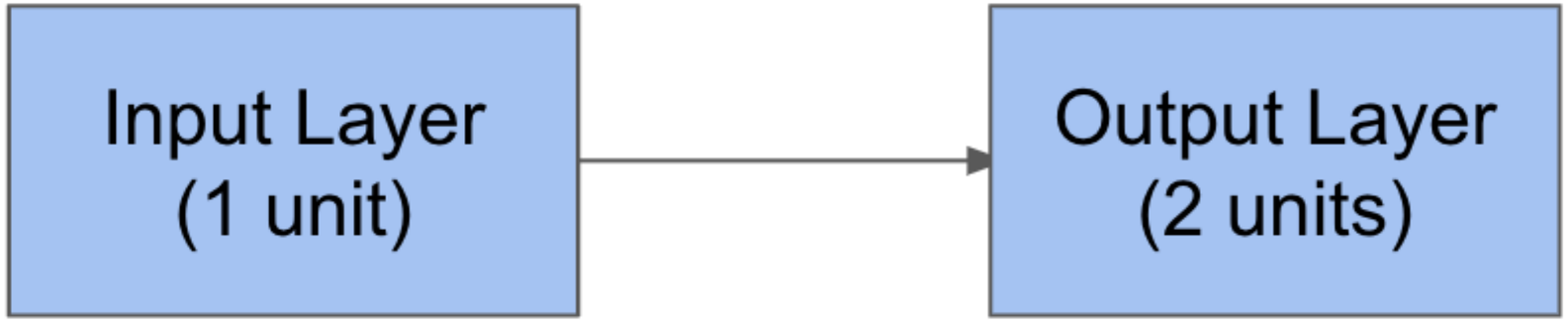
Two-output models

ADVANCED DEEP LEARNING WITH KERAS

Zach Deane-Mayer
Data Scientist

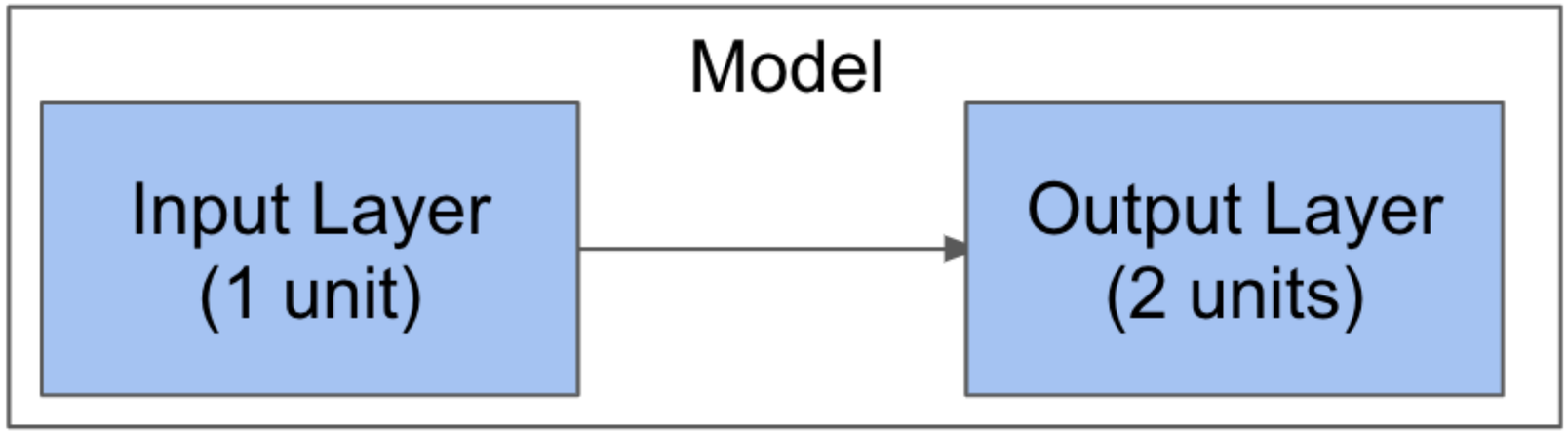
Simple model with 2 outputs

```
from keras.layers import Input, Concatenate, Dense
input_tensor = Input(shape=(1,))
output_tensor = Dense(2)(input_tensor)
```



Simple model with 2 outputs

```
from keras.models import Model
model = Model(input_tensor, output_tensor)
model.compile(optimizer='adam', loss='mean_absolute_error')
```



Fitting a model with 2 outputs

```
games_tourney_train[['seed_diff', 'score_1', 'score_2']].head()
```

```
   seed_diff  score_1  score_2
0         -3       41       50
1          4       61       55
2          5       59       63
3          3       50       41
4          1       54       63
```

```
X = games_tourney_train[['seed_diff']]
y = games_tourney_train[['score_1', 'score_2']]
model.fit(X, y, epochs=500)
```

Inspecting a 2 output model

```
model.get_weights()
```

```
[array([[ 0.60714734, -0.5988793 ]], dtype=float32),
 array([70.39491, 70.39306], dtype=float32)]
```

Evaluating a model with 2 outputs

```
X = games_tourney_test[['seed_diff']]
y = games_tourney_test[['score_1', 'score_2']]
model.evaluate(X, y)
```

```
11.528035634635021
```

Let's practice!

ADVANCED DEEP LEARNING WITH KERAS

Single model for classification and regression

ADVANCED DEEP LEARNING WITH KERAS

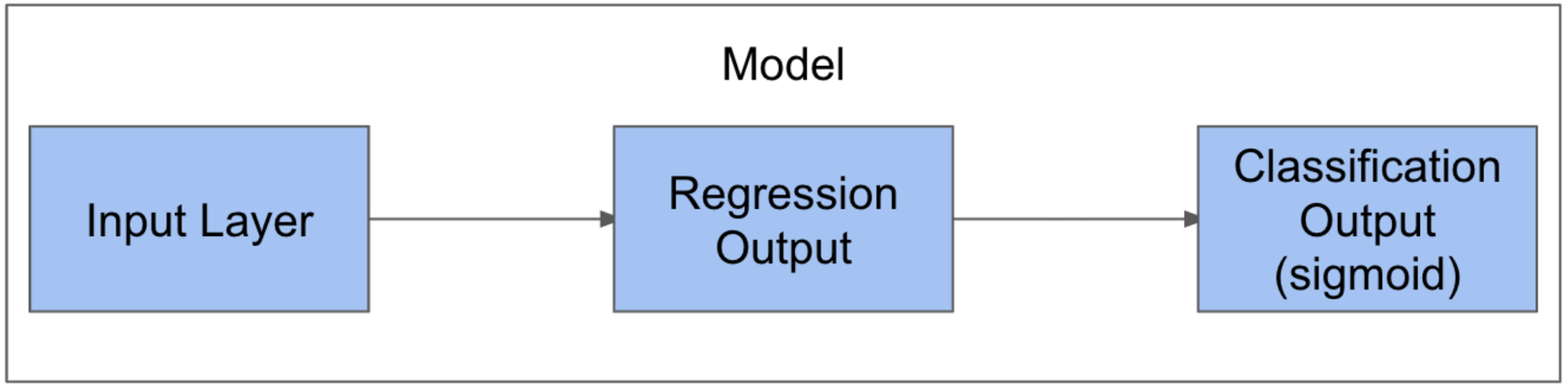
Zach Deane-Mayer
Data Scientist

Build a simple regressor/classifier

```
from keras.layers import Input, Dense
input_tensor = Input(shape=(1,))
output_tensor_reg = Dense(1)(input_tensor)
output_tensor_class = Dense(1, activation='sigmoid')(output_tensor_reg)
```

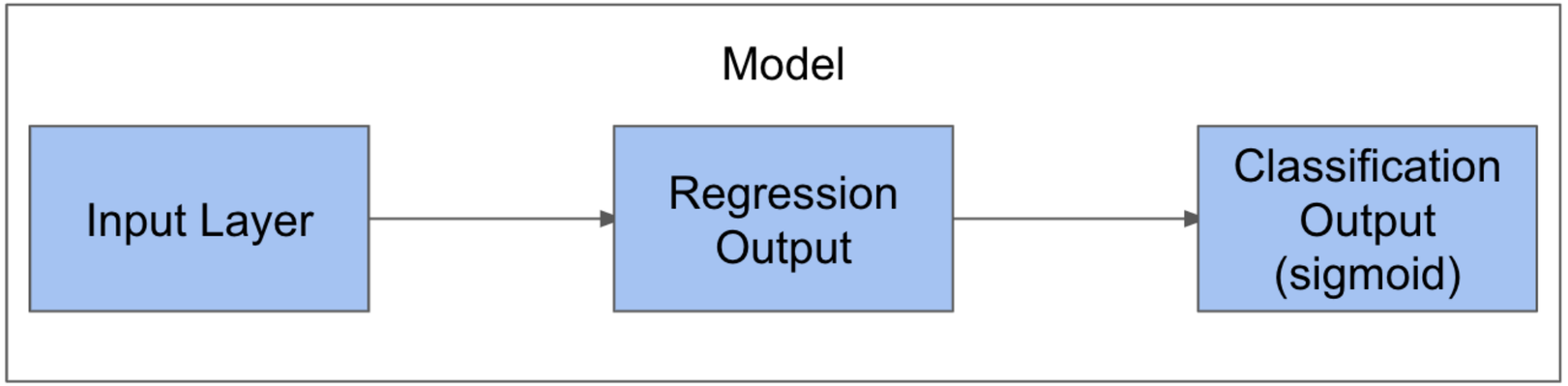
Make a regressor/classifier model

```
from keras.models import Model
model = Model(input_tensor, [output_tensor_reg, output_tensor_class])
model.compile(loss=['mean_absolute_error', 'binary_crossentropy'],
              optimizer='adam')
```



Fit the combination classifier/regressor

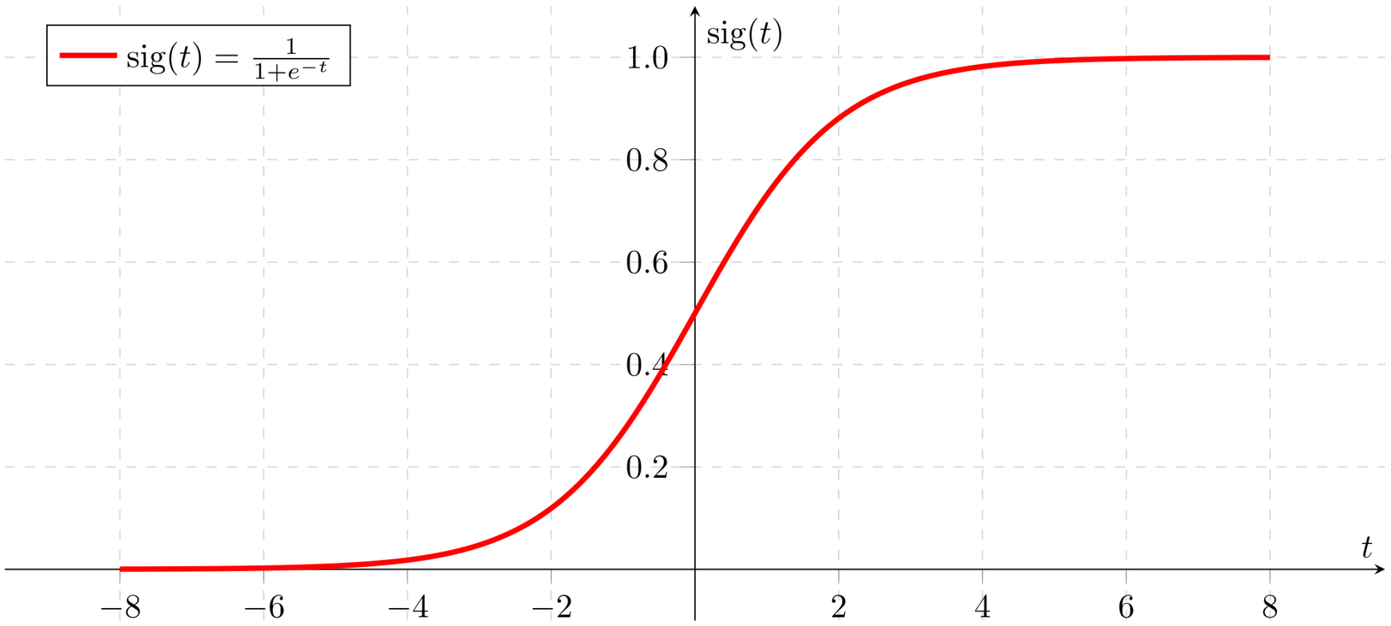
```
X = games_tourney_train[['seed_diff']]
y_reg = games_tourney_train[['score_diff']]
y_class = games_tourney_train[['won']]
model.fit(X, [y_reg, y_class], epochs=100)
```



Look at the model's weights

```
model.get_weights()
```

```
[array([[1.2371823]], dtype=float32),
 array([-0.05451894], dtype=float32),
 array([[0.13870609]], dtype=float32),
 array([0.00734114], dtype=float32)]
```



Look at the model's weights

```
model.get_weights()
```

```
[array([[1.2371823]], dtype=float32),
 array([-0.05451894], dtype=float32),
 array([[0.13870609]], dtype=float32),
 array([0.00734114], dtype=float32)]
```

```
from scipy.special import expit as sigmoid
print(sigmoid(1 * 0.13870609 + 0.00734114))
```

```
0.5364470465211318
```

Evaluate the model on new data

```
X = games_tourney_test[['seed_diff']]
y_reg = games_tourney_test[['score_diff']]
y_class = games_tourney_test[['won']]
model.evaluate(X, [y_reg, y_class])
```

```
[9.866300069455413, 9.281179495657208, 0.585120575627864]
```

Now you try!

ADVANCED DEEP LEARNING WITH KERAS

Wrap-up

ADVANCED DEEP LEARNING WITH KERAS

Zach Deane-Mayer
Data Scientist

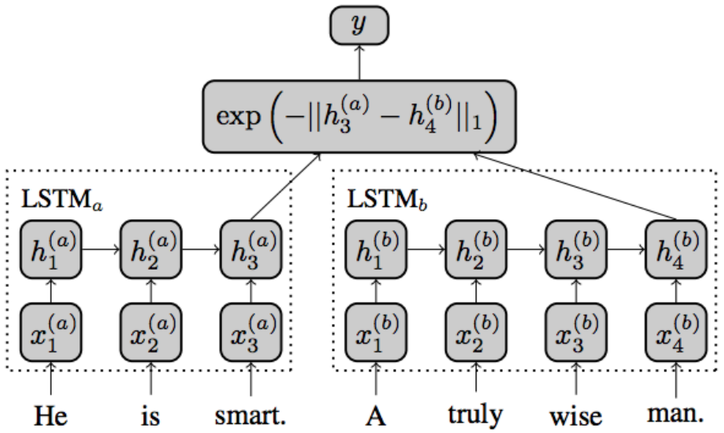
So far...

- Functional API
- Shared layers
- Categorical embeddings
- Multiple inputs
- Multiple outputs
- Regression / Classification in one model

Shared layers

Useful for making comparisons

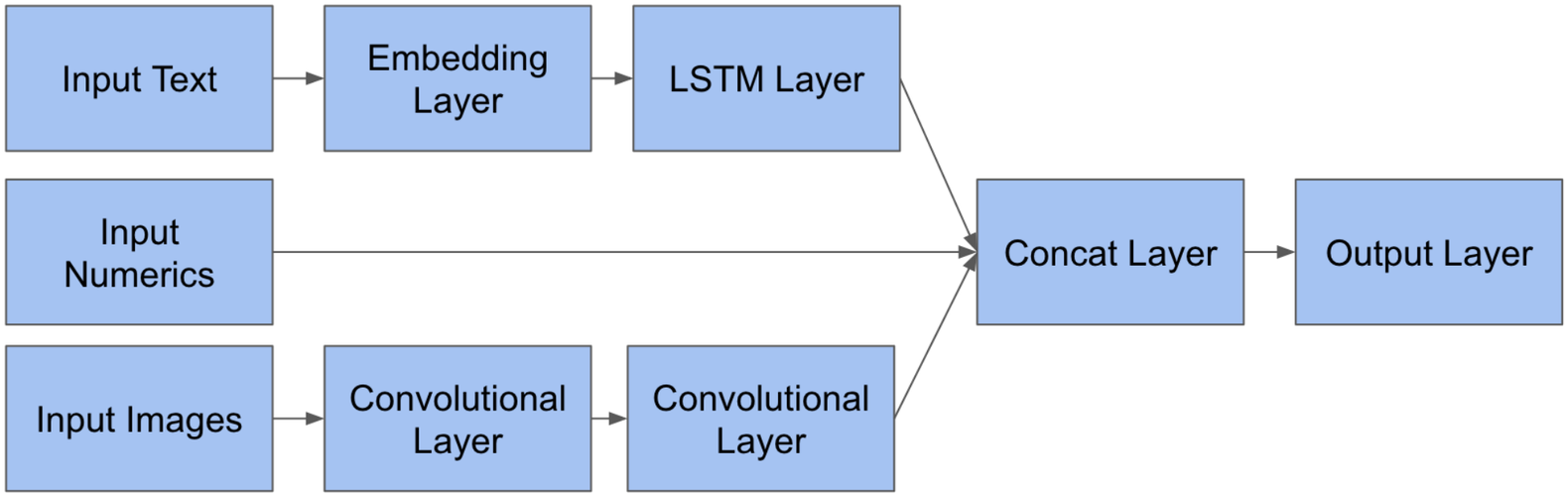
- Basketball teams
- Image similarity / retrieval
- Document similarity



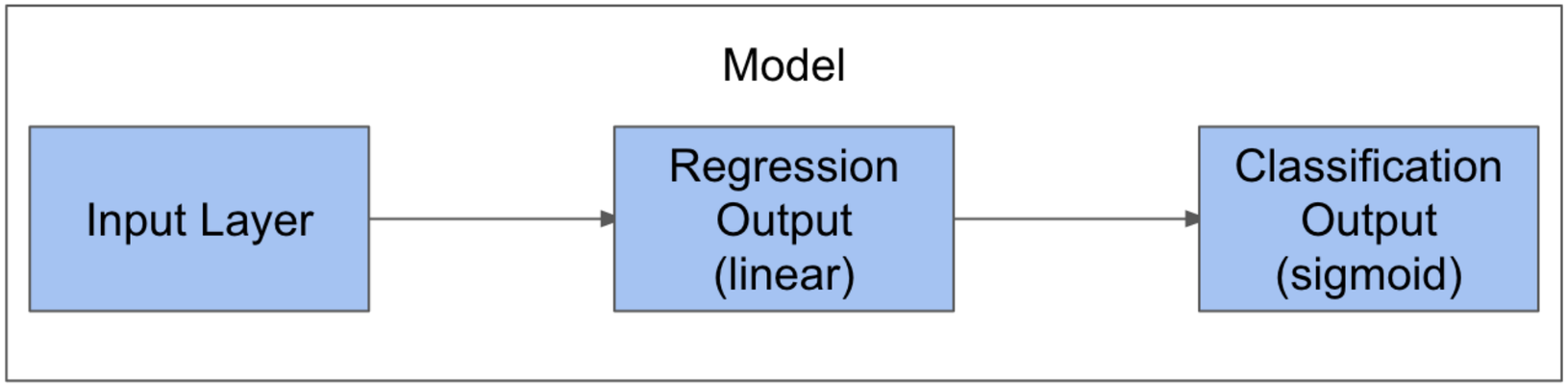
Known in the academic literature as Siamese networks

- **Link to blog post**
- **Link to academic paper**

Multiple inputs



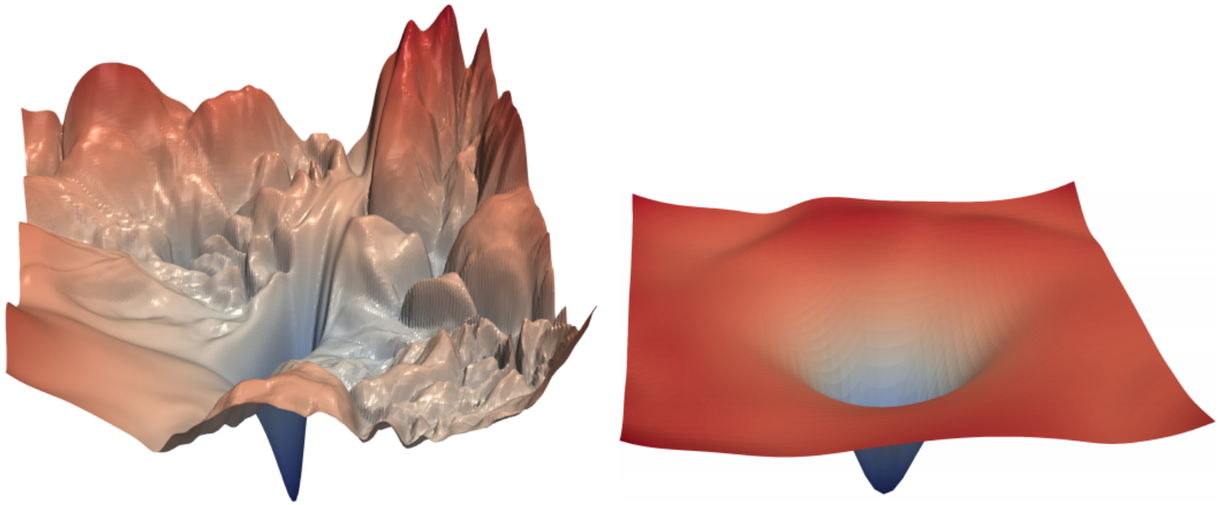
Multiple outputs



Skip connections

```
input_tensor = Input((100,))
hidden_tensor = Dense(256, activation='relu')(input_tensor)
hidden_tensor = Dense(256, activation='relu')(hidden_tensor)
hidden_tensor = Dense(256, activation='relu')(hidden_tensor)
output_tensor = Concatenate()([input_tensor, hidden_tensor])
output_tensor = Dense(256, activation='relu')(output_tensor)
```

Visualizing the Loss Landscape of Neural Nets



(a) without skip connections

(b) with skip connections

Best of luck!

ADVANCED DEEP LEARNING WITH KERAS