Keras Functional Model-Non-linear Neural Networks

- The **Functional API** in Keras is a more flexible way to build neural networks than the simpler `Sequential` model.

```
from keras.models import Model
```

- 👆 It needs `input` & `output` layers
  - There can be multiple input & output layers

💡 Difference from `Sequential` :

Feature	Sequential API	Functional API
Structure	Straight line	Any shape (branches, merges, etc.)
Suitable for	Simple models	Complex, custom models
Limitations	Only one input/output, no branching	Multiple inputs/outputs, shared layers, skip connections

What is a Non-linear Neural Network?

A **non-linear neural network** is just a regular neural network that uses **non-linear activation functions** like:

- `ReLU`
- `tanh`
- `sigmoid`
- `softmax`

Without these, a neural network is just a **linear function**, which can't learn complex patterns.

Why Use Functional API?

You **must** use Functional API if you want:

- **Multiple inputs or outputs**
- **Layers that share weights**
- **Skip connections (like in ResNet)**
- **Combining outputs from different layers**
- **Non-linear topologies (not just a straight line)**

Python code

> 💡 **You have to name each layer & state where it's getting input from.**

Simple Structure:

Define our input & output layers:

```
from keras.models import Model

model = Model(inputs = x ,outputs = [output1,output2])
```

Import all layers:

```
from keras.layers import *
```

Define input layer:

```
x = Input(shape=(3,))
```

- We have 3 columns in our input layer

Make Dense & output layers:

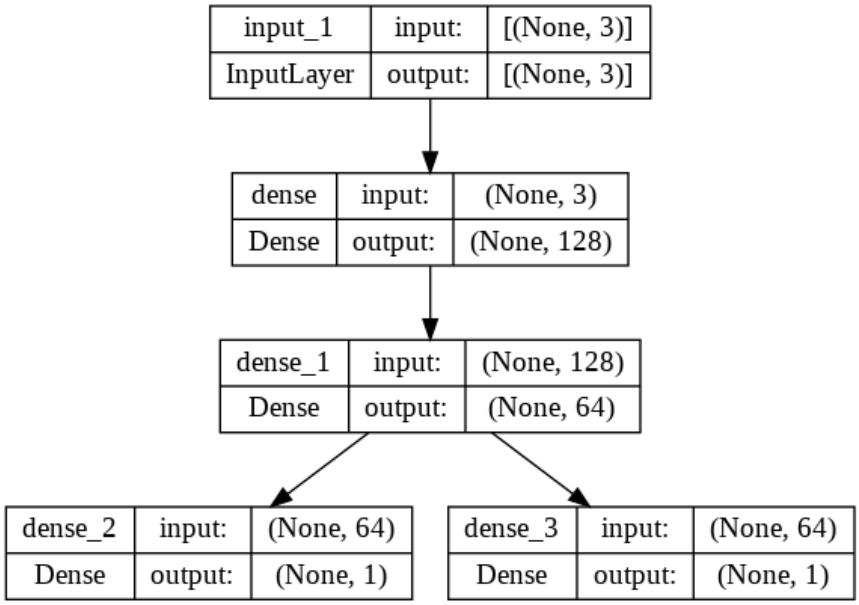
```
hidden1 = Dense(128,activation='relu')(x)
hidden2 = Dense(64,activation='relu')(hidden1)

output1 = Dense(1,activation='linear')(hidden2)
output2 = Dense(1,activation='sigmoid')(hidden2)
```

- `(x)`, `(hidden1)`, etc. are **input tensors** to the current layer (output of the previous layer)
  - Meaning: `hidden1` is getting input from `x`
  - `hidden1` is getting input from `hidden2`

Visualize this:

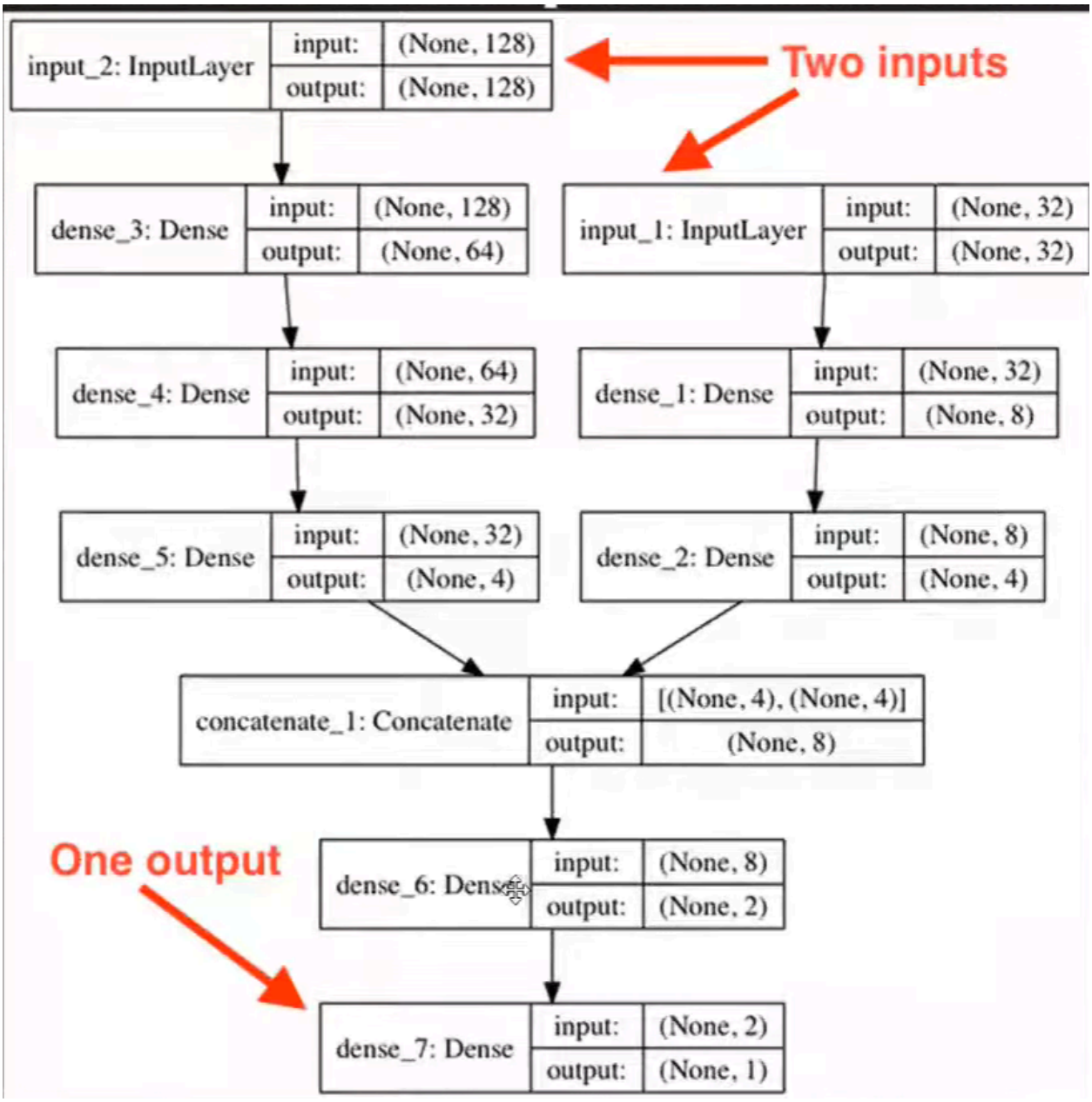
```
from keras.utils import plot_model
plot_model(model,show_shapes=True)
```



2 Inputs, 1 Output:

Flow:

1. Define `InputA` & `InputB`
2. There are 2 branches
    a. Branch 1: Gets `InputA`
    b. Branch 2: Gets `InputB`
3. **Combine (concatenate)** the output of the two branches
4. Make Dense & Output layer
5. Make a model



1. Define Input Layers:

```
inputA = Input(shape=(32,))  # First input (e.g., tabular data)
inputB = Input(shape=(128,)) # Second input (e.g., text embeddings)
```

- **Purpose**: Creates two separate input tensors for different data types.
- **Note**:

- `inputA` expects 32-dimensional vectors.
- `inputB` expects 128-dimensional vectors.

---

2. Branch 1: `inputA`

```
x = Dense(8, activation="relu")(inputA)  # First dense layer
x1 = Dense(4, activation="relu")(x)     # Second dense layer
```

- **Flow**: `inputA` → Dense (8 units) → Dense (4 units).
- **Output Shape**: `(None, 4)` (4-dimensional features).

---

3. Branch 2: `inputB`

```
y = Dense(64, activation="relu")(inputB)  # First dense layer
y1 = Dense(32, activation="relu")(y)     # Second dense layer
y2 = Dense(4, activation="relu")(y1)     # Final layer (matches Branch 1's outpu
t)
```

- **Flow**: `inputB` → Dense (64 units) → Dense (32 units) → Dense (4 units).
- **Output Shape**: `(None, 4)` (4-dimensional features, same as Branch 1).

---

4. Combine Branches (Concatenation)

```
combined = concatenate([x1, y2])  # Merges outputs from both branches
```

- **Purpose**: Combines features from both inputs into a single tensor.
- **Output Shape**: `(None, 8)` (4 + 4 = 8 dimensions).

---

5. Final Dense Layers for Prediction

```
z = Dense(2, activation="relu")(combined)  # Intermediate layer
z1 = Dense(1, activation="linear")(z)     # Output layer (regression)
```

- **Flow**: Combined features → Dense (2 units) → Dense (1 unit).
- **Activation**:
  - `relu` for intermediate layer (non-linearity).
  - `linear` for output (regression task).

---

6. Define the Model

```
model = Model(inputs=[inputA, inputB], outputs=z1)
```

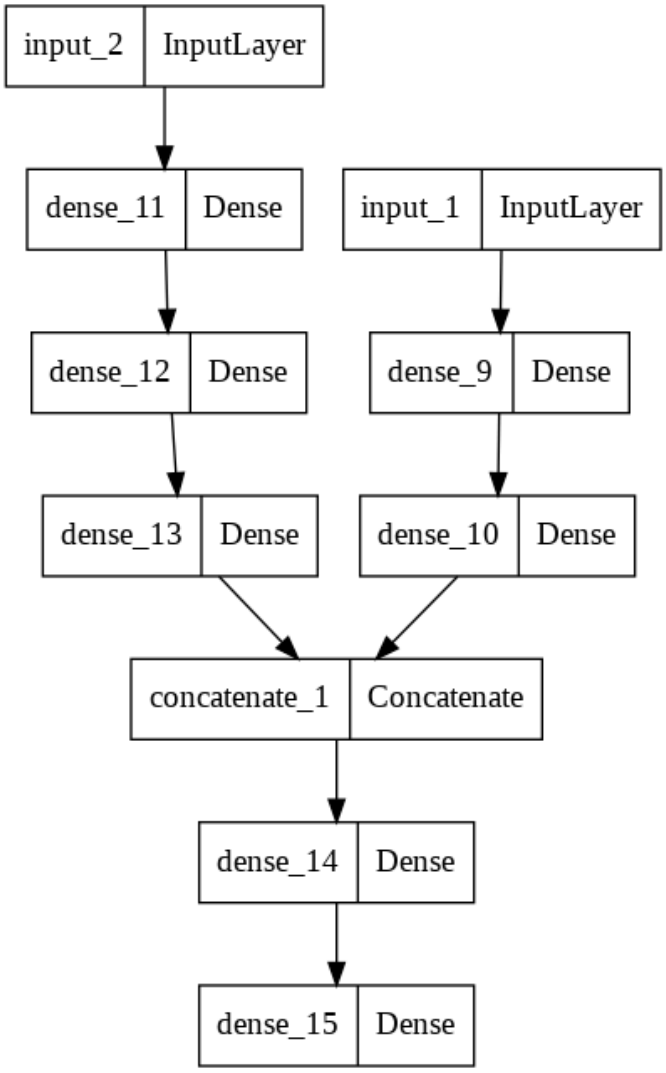
- **Inputs**: Accepts both `inputA` and `inputB` as a list.
- **Output**: Single regression value (`z1`).

---

7. Visualize the Model

```
from keras.utils import plot_model
plot_model(model, show_shapes=True)
```

- **Output**: A diagram showing the two input branches, concatenation, and output layers.


input_2 InputLayer
dense_11 Dense
input_1 InputLayer
dense_12 Dense
dense_9 Dense
dense_13 Dense
dense_10 Dense
concatenate_1 Concatenate
dense_14 Dense
dense_15 Dense