Training Day-92 Report:

Functional Composition in Keras

Definition:

Functional Composition in Keras refers to building deep learning models with a flexible and expressive API. It allows the creation of complex architectures, such as models with multiple inputs, multiple outputs, shared layers, or non-linear connections.

Core Concepts

1. **Input Layer:**
    - Define the shape and type of the input using tf.keras.Input.
    - Example:
    - inputs = tf.keras.Input(shape=(784,))
2. **Layer Composition:**
    - Each layer is treated as a function that transforms its input into output.
    - Example:
    - x = tf.keras.layers.Dense(128, activation='relu')(inputs)
3. **Model Definition:**
    - Use tf.keras.Model to specify the input and output of the entire model.
    - Example:
    - model = tf.keras.Model(inputs=inputs, outputs=outputs)

Steps to Build a Functional Model

1. **Define Inputs:**
    - Specify the input shape and type using tf.keras.Input.
2. **Stack Layers:**
    - Use each layer as a function and connect it to the previous layer.
3. **Output Layer:**
    - Specify the final output layer for your task (e.g., classification or regression).
4. **Compile the Model:**
    - Set the optimizer, loss function, and metrics.
5. **Train and Evaluate:**
    - Use model.fit() for training and model.evaluate() for testing.

Example: Building a Functional Model

```
import tensorflow as tf

# Define inputs
inputs = tf.keras.Input(shape=(784,))
```

```
# Build hidden layers
x = tf.keras.layers.Dense(128, activation='relu')(inputs)
x = tf.keras.layers.Dropout(0.2)(x)

# Define output layer
outputs = tf.keras.layers.Dense(10, activation='softmax')(x)

# Create the model
model = tf.keras.Model(inputs=inputs, outputs=outputs)

# Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Summary of the model
model.summary()
```

When to Use Functional Composition

1. **Complex Architectures:**
    - Networks with multiple inputs or outputs.
    - Example: A model that takes both text and image data as input.
2. **Shared Layers:**
    - Reusing the same layers for different parts of the model.
    - Example: Siamese networks for comparing two inputs.
3. **Non-Sequential Connections:**
    - Models with skip connections, branching, or merging.
    - Example: Residual Networks (ResNets) or Inception Networks.

Example: Multi-Input and Multi-Output Model

```
# Input 1: Text data
text_input = tf.keras.Input(shape=(100,), name='text_input')
x1 = tf.keras.layers.Embedding(input_dim=10000, output_dim=64)(text_input)
x1 = tf.keras.layers.LSTM(128)(x1)

# Input 2: Numerical data
numerical_input = tf.keras.Input(shape=(10,), name='numerical_input')
x2 = tf.keras.layers.Dense(64, activation='relu')(numerical_input)

# Concatenate features
combined = tf.keras.layers.concatenate([x1, x2])
```

```
# Output 1: Classification
output1 = tf.keras.layers.Dense(10, activation='softmax', name='output1')(combined)

# Output 2: Regression
output2 = tf.keras.layers.Dense(1, activation='linear', name='output2')(combined)

# Define the model
model = tf.keras.Model(inputs=[text_input, numerical_input], outputs=[output1, output2])

# Compile the model
model.compile(optimizer='adam',
              loss={'output1': 'sparse_categorical_crossentropy', 'output2': 'mse'},
              metrics={'output1': 'accuracy', 'output2': 'mae'})

# Summary of the model
model.summary()
```

Advantages of Functional Composition

1. **Flexibility:**
    - Build complex and dynamic architectures.
2. **Readability:**
    - Clear mapping of inputs to outputs.
3. **Reusability:**
    - Easily reuse layers or sub-models.
4. **Custom Architectures:**
    - Suitable for non-linear, branched, or multi-task networks.

.