Training Day-100 Report:

Sequential Composition:

Sequential Composition

Sequential composition in deep learning refers to building a neural network layer-by-layer in a sequential order. TFLearn provides a straightforward way to implement sequential models, where each layer's output is directly passed as input to the next layer.

Key Features of Sequential Composition

1. **Layer-by-Layer Construction:**
    - Layers are stacked in a linear order, making it ideal for feedforward neural networks.
2. **Ease of Use:**
    - Simplifies model design, especially for beginners, by eliminating the need for complex connections between layers.
3. **Flexible Integration:**
    - While primarily linear, additional functionalities like dropout, activation, and batch normalization can be incorporated.

Steps for Sequential Composition in TFLearn

1. **Define the Input Layer:**
    - Specify the input data shape using the input_data() function.
    - Example:
    - input_layer = input_data(shape=[None, 784])
2. **Add Hidden Layers:**
    - Use predefined layers such as fully_connected, conv_2d, or dropout to build the network.
    - Example:
    - hidden_layer = fully_connected(input_layer, 128, activation='relu')
3. **Add the Output Layer:**
    - Choose an activation function suitable for the problem (e.g., softmax for classification).
    - Example:

   - output_layer = fully_connected(hidden_layer, 10, activation='softmax')
4. **Define the Training Objective:**
   - Use the regression() function to specify the optimizer, loss function, and learning rate.
   - Example:
   - network = regression(output_layer, optimizer='adam', loss='categorical_crossentropy', learning_rate=0.001)
5. **Create and Train the Model:**
   - Use the DNN() method to compile the model and train it using the fit() function.
   - Example:
   - model = tflearn.DNN(network)
   - model.fit(X_train, y_train, n_epoch=10, batch_size=64, show_metric=True)

Example: Building a Sequential Model in TFLearn

```
import tflearn
from tflearn.layers.core import input_data, fully_connected
from tflearn.layers.estimator import regression

# Input layer
input_layer = input_data(shape=[None, 784])

# Hidden layers
hidden_layer1 = fully_connected(input_layer, 128, activation='relu')
hidden_layer2 = fully_connected(hidden_layer1, 64, activation='relu')

# Output layer
output_layer = fully_connected(hidden_layer2, 10, activation='softmax')

# Define the regression layer
network = regression(output_layer, optimizer='adam', loss='categorical_crossentropy',
learning_rate=0.01)

# Create and train the model
```

```
model = tflearn.DNN(network)
model.fit(X_train, y_train, n_epoch=10, batch_size=32, show_metric=True)
```

Advantages of Sequential Composition

- **Simplified Model Building:** Linear structure is intuitive and easy to debug.
- **Quick Prototyping:** Ideal for testing simple architectures.
- **Compatibility:** Easily integrates with additional regularization or dropout layers.

Applications

- Basic feedforward neural networks.
- Sequential image processing tasks (e.g., digit classification).
- Entry-level projects and educational purposes.

Sequential composition provides a structured and straightforward way to build neural networks, making it an excellent choice for beginners and prototyping.