**Keras Subclassing: Custom Layers & Models – Quick Notes**

**✅ Key Concepts**

📌 **1. What is Keras Subclassing?**

* A flexible way to **define custom layers & models** in TensorFlow using Python classes.
* Subclass keras.layers.Layer for **custom layers** and keras.Model for **custom models**.

📌 **2. Key Methods in Subclassing**

* \_\_init\_\_(): Initialize the layer/model.
* build(input\_shape): Create **trainable weights** dynamically.
* call(inputs): Defines the **forward pass** logic.
* get\_config(): Enables **serialization** (saving & loading).

📌 **3. Difference Between Layer and Model**

| **Feature** | **keras.layers.Layer** | **keras.Model** |
| --- | --- | --- |
| Used For | Custom layers | Full models |
| Has fit()? | ❌ No | ✅ Yes |
| Has save()? | ❌ No | ✅ Yes |

📌 **4. How to Create a Custom Layer?**

python

CopyEdit

class CustomLayer(keras.layers.Layer):

def \_\_init\_\_(self, units=32):

super().\_\_init\_\_()

self.units = units

def build(self, input\_shape):

self.w = self.add\_weight(shape=(input\_shape[-1], self.units), initializer="random\_normal", trainable=True)

self.b = self.add\_weight(shape=(self.units,), initializer="zeros", trainable=True)

def call(self, inputs):

return tf.matmul(inputs, self.w) + self.b

📌 **5. How to Create a Custom Model?**

python

CopyEdit

class CustomModel(keras.Model):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.dense1 = keras.layers.Dense(64, activation='relu')

self.dense2 = keras.layers.Dense(10, activation='softmax')

def call(self, inputs):

x = self.dense1(inputs)

return self.dense2(x)

📌 **6. How to Save & Load a Model?**

python

CopyEdit

model.save("custom\_model.keras") # Save

loaded\_model = keras.models.load\_model("custom\_model.keras") # Load

**🎯 Interview Questions & Answers**

✅ **Q1: What is the benefit of using subclassing in Keras?**  
🔹 More flexibility in defining **complex models & custom layers**.

✅ **Q2: What is the purpose of the call() method?**  
🔹 Defines the **forward pass** (computations) of the layer/model.

✅ **Q3: How do you add trainable weights in a custom layer?**  
🔹 Use self.add\_weight() inside build().

✅ **Q4: What is add\_loss() used for?**  
🔹 Adds **custom loss** (e.g., **regularization**) during training.

✅ **Q5: How do you differentiate training & inference in a custom layer?**  
🔹 Pass training=True in call(). Example:

python

CopyEdit

def call(self, inputs, training=False):

return tf.nn.dropout(inputs, rate=0.5) if training else inputs

✅ **Q6: What’s the difference between build() and call()?**

| **Method** | **Purpose** |
| --- | --- |
| build() | Creates weights **once** |
| call() | Executes the **forward pass** |

✅ **Q7: How do you make a custom layer serializable?**  
🔹 Implement get\_config() method.

✅ **Q8: What is the use of from\_config()?**  
🔹 Helps in reloading custom layers/models from **saved configurations**.

✅ **Q9: How do you add non-trainable weights?**  
🔹 Use self.add\_weight(trainable=False).

✅ **Q10: Can you use model.fit() with a custom model?**  
🔹 ✅ Yes, if the model subclasses keras.Model.

**📌 Best Practices**

✔ **Use build() for weight creation** (lazy initialization).  
✔ **Use call() for forward pass logic**.  
✔ **Add get\_config() for serialization**.  
✔ **Use add\_loss() for custom loss terms**.  
✔ **Use training argument to differentiate between training & inference**.

**ChatGPT said:**

Here are some **must-know interview questions and answers** based on the key points from the model saving and loading process:

**1. What are the different ways to save a TensorFlow model?**

**Answer:**

* **Model Checkpoints** (ModelCheckpoint callback) to save weights during training.
* **Saving the entire model** using model.save() in:
  + .keras format (recommended)
  + HDF5 .h5 format (legacy)
  + **SavedModel** format (for TensorFlow Serving, deployment)

**2. How do you load a saved model in TensorFlow?**

**Answer:**  
Use tf.keras.models.load\_model('model\_path') to load a model from the .keras, .h5, or SavedModel format. For custom objects, pass them using the custom\_objects parameter.

**3. What is the difference between model.save\_weights() and model.save()?**

**Answer:**

* model.save\_weights(): Saves only the model's weights (parameters).
* model.save(): Saves the entire model, including architecture, weights, optimizer state, and training configuration.

**4. What is the use of the ModelCheckpoint callback?**

**Answer:**  
ModelCheckpoint saves the model's weights during training, either at regular intervals (every few epochs) or when the model's performance improves, allowing you to resume training from the saved state.

**5. How do you load weights into a new model?**

**Answer:**  
Use model.load\_weights('checkpoint\_path') to load the weights from a saved checkpoint into a new model with the same architecture.

**6. What is the recommended format for saving Keras models?**

**Answer:**  
The .keras format is the recommended format for saving Keras models due to its simplicity and efficient, name-based saving.

**7. What do you need to do when saving/loading custom objects (models or layers)?**

**Answer:**

* Implement get\_config() and from\_config() methods for custom layers/models.
* Register custom layers with @tf.keras.utils.register\_keras\_serializable for easier saving/loading.

**8. Explain the difference between SavedModel format and HDF5 format.**

**Answer:**

* **SavedModel**: A directory containing a TensorFlow graph and checkpoint; ideal for deployment with TensorFlow Serving.
* **HDF5**: A single file that stores the entire model, including architecture and weights; more commonly used in Keras.

**9. How can you resume training from the last checkpoint?**

**Answer:**

* Use ModelCheckpoint to save the model weights regularly.
* Use model.load\_weights('checkpoint\_path') to load the latest checkpoint and resume training.

**10. What is the .keras format, and why is it preferred?**

**Answer:**  
The .keras format is a new high-level format that is efficient and uses name-based saving, making it easier to debug. It is recommended for saving Keras models over legacy formats like .h5.