

Seed shuffling in TensorFlow



```
shuffled_tensor = tf.random.shuffle(  
    tensor_2D  
)  
shuffled_tensor
```

```
shuffled_tensor = tf.random.shuffle(  
    tensor_2D,seed=42  
)  
shuffled_tensor
```

```
tf.random.set_seed(1234)  
shuffled_tensor = tf.random.shuffle(  
    tensor_2D  
)  
shuffled_tensor
```

```
tf.random.set_seed(1234)  
shuffled_tensor = tf.random.shuffle(  
    tensor_2D,seed=42  
)  
shuffled_tensor
```

Explain me the difference between these different ways of seed shuffling



Great question! TensorFlow provides two levels of randomness control: **global seed** (`tf.random.set_seed()`) and **operation-level seed** (`seed` argument in functions like tf.random.shuffle`). Let's go through each case you mentioned and explain what's happening.

✓ Understanding the Two Types of Seeds

- **Global Seed** → Set using `tf.random.set_seed()`. It influences all random operations for deterministic results **only when used with operation-level seeds**.
- **Operation-Level Seed** → Set directly on the random function (e.g., `seed=42` inside `tf.random.shuffle`). If set alone, it affects only that operation.

To make randomness **deterministic**, both seeds should be set.

📖 Your Examples Explained

Assume `tensor_2D` is a 2D tensor like:

```
tensor_2D = tf.constant([[1, 2], [3, 4], [5, 6]])
```

◆ 1. No Seed at All

```
shuffled_tensor = tf.random.shuffle(tensor_2D)
```

✓ Effect:

- No seed = random every time.
- Output will be different on every run.
- **Non-deterministic**

◆ 2. Only Operation-Level Seed

```
shuffled_tensor = tf.random.shuffle(tensor_2D, seed=42)
```

✓ Effect:


- Same operation-level seed gives **same output** on same run (inside a session).
 - But **different output** across runs of the program.
 - So, it's **deterministic only within the session**, not across multiple runs.
-

◆ 3. Only Global Seed

```
tf.random.set_seed(1234)  
shuffled_tensor = tf.random.shuffle(tensor_2D)
```

✓ Effect:

- Global seed is set, but no operation-level seed.
- This won't guarantee deterministic behavior across runs.
- Different shuffles on different runs.

 TensorFlow requires **both** seeds to be set for true reproducibility.

◆ 4. Global + Operation-Level Seed

```
tf.random.set_seed(1234)  
shuffled_tensor = tf.random.shuffle(tensor_2D, seed=42)
```

✓ Effect:

- This ensures **full reproducibility**.
- You will get the **exact same output every time** you run the code.
- This is the recommended way to get **repeatable** randomness.

✓ Summary Table

Global Seed	Op Seed	Reproducible Across Runs?	Reproducible Within Run?
✗	✗	✗	✗
✗	✓	✗	✓
✓	✗	✗	✗
✓	✓	✓	✓

💡 Best Practice

To ensure the same shuffling every time (important for reproducible training/testing):

```
tf.random.set_seed(1234)
shuffled_tensor = tf.random.shuffle(tensor_2D, seed=42)
```

Let me know if you want to test this live with a sample tensor and check outputs!



show me the results for each operation



It looks like TensorFlow isn't available in the current environment, so I can't execute the code here directly. But you can easily run this locally on your machine. Here's the complete code you can use to observe the different shuffling results:

✓ Code to Try in Local Python Environment

```
import tensorflow as tf

# Define a 2D tensor
tensor_2D = tf.constant([[1, 2], [3, 4], [5, 6]])

# 1. No Seed
shuffled_1 = tf.random.shuffle(tensor_2D)
print("1. No Seed:")
print(shuffled_1.numpy())

# 2. Only Operation-Level Seed
shuffled_2 = tf.random.shuffle(tensor_2D, seed=42)
print("\n2. Only Operation-Level Seed:")
print(shuffled_2.numpy())

# 3. Only Global Seed
tf.random.set_seed(1234)
shuffled_3 = tf.random.shuffle(tensor_2D)
print("\n3. Only Global Seed:")
print(shuffled_3.numpy())

# 4. Global + Operation-Level Seed
tf.random.set_seed(1234)
shuffled_4 = tf.random.shuffle(tensor_2D, seed=42)
```

```
print("\n4. Global + Operation-Level Seed:")
print(shuffled_4.numpy())
```

✅ What You Should Observe

- `1. No Seed:` → Different output every time you run the script.
- `2. Only Operation-Level Seed:` → Same result **within one run**, but changes across runs.
- `3. Only Global Seed:` → Still gives different output across runs.
- `4. Global + Operation-Level Seed:` → Always gives the **same output** every time you run the script.

Would you like me to generate simulated output here to demonstrate what you might see? [↗]



just some sample examples to understand



Sure! Here's a simulated example to help you **understand what to expect** when running each scenario.

🧱 Base Tensor:

```
tensor_2D = tf.constant([[1, 2], [3, 4], [5, 6]])
```

1. ❌ No Seed

```
shuffled_tensor = tf.random.shuffle(tensor_2D)
```

Each run will give **different order**. Example outputs:

```
Run 1: [[5 6]
        [3 4]
        [1 2]]
```

```
Run 2: [[3 4]
        [1 2]
        [5 6]]
```

2. ✅ Only Operation-Level Seed (`seed=42`)

```
shuffled_tensor = tf.random.shuffle(tensor_2D, seed=42)
```

Same order **within one script execution**, but changes across runs.

```
Run 1: [[3 4]
        [1 2]
        [5 6]]
```

```
Run 2: [[5 6]
        [1 2]
        [3 4]]
```

3. ✅ Only Global Seed (`tf.random.set_seed(1234)`)

```
tf.random.set_seed(1234)
shuffled_tensor = tf.random.shuffle(tensor_2D)
```

Still **not reproducible across runs**. You might get:

```
Run 1: [[1 2]
        [5 6]
        [3 4]]
```

```
Run 2: [[3 4]
        [1 2]
        [5 6]]
```

4. Global + Operation-Level Seed (`set_seed + seed=42`)

```
tf.random.set_seed(1234)
shuffled_tensor = tf.random.shuffle(tensor_2D, seed=42)
```

 Always gives **same result across all runs**:

```
Run 1: [[3 4]
        [1 2]
        [5 6]]
```

```
Run 2: [[3 4]
        [1 2]
        [5 6]]
```

 **TL;DR: Want reproducibility?**

Use **both**:

```
tf.random.set_seed(1234)
tf.random.shuffle(tensor_2D, seed=42)
```

Let me know if you want a visual plot or to try with larger tensors!



Same order within one script execution, but changes across runs.
I didn't get it . What is within script execution means and what is a run



Great question! Let's break this down in simple, clear terms:

✓ What is a Script Execution or a Run?

Think of your code as a Python script (e.g., `main.py`). A **run** or **script execution** means:

- “➤ You start your program (press Run or `python main.py`)
- It executes from top to bottom
- Then it stops (program ends)”

Every time you **re-run** the script, it's a **new execution** or a **new run**.

📌 What does "Same order within one script execution" mean?

If you use:

```
shuffled_tensor = tf.random.shuffle(tensor_2D, seed=42)
```

And you shuffle **multiple times in the same run**, like this:

```
shuffled1 = tf.random.shuffle(tensor_2D, seed=42)
shuffled2 = tf.random.shuffle(tensor_2D, seed=42)
print(shuffled1.numpy())
print(shuffled2.numpy())
```

✅ You will get the **same result for both**, because it's the same seed and same execution.

📌 What does "Changes across runs" mean?

If you **run the same code again later**, like:

```
$ python main.py    ← Run 1
$ python main.py    ← Run 2 (new run)
```

Even with the same seed (`seed=42`), the **output may change** across these runs — **because the global seed is not set**.

🔄 Simple Analogy

Imagine baking cookies:

- **Script execution = one baking session.**
- **Run = each time you bake.**

Using only `seed=42` is like using a random cookie mold:

- In **one session**, you'll get the same cookie shapes repeatedly.
- In a **different session** (next day), even with the same mold (seed), you may get different shapes due to some randomness (like oven temperature, timing) — **unless you fix everything else too** (global seed).

✅ How to Get Same Output Across Runs?

You need:

```
tf.random.set_seed(1234)
shuffled = tf.random.shuffle(tensor_2D, seed=42)
```

This ensures:

- ✅ Same output on every run.
- ✅ Same output inside the script, even if you shuffle multiple times.

Let me know if you'd like a visual explanation or animated diagram for this.