Sequence models in Keras 3 and TensorFlow

The same ideas the TensorFlow way: Keras 3 layers, `compile` / `fit`, callbacks, and `tf.GradientTape` when you want your own training loop. TensorFlow 2.21 is stable and mostly receives fixes now; Google recommends Keras 3, JAX or PyTorch for new generative-AI work, and Keras 3 code like this also runs on JAX or PyTorch by changing one line.

Open in Google Colab · part of the inPace course library. Slides © DeepLearning.AI (CC BY-SA 2.0); this notebook: inPace (CC BY-SA 4.0).

Run it top to bottom (Runtime → Run all in Colab). Each part says what to look for.

```
# Works in Google Colab as it is (TensorFlow and Keras are already installed
there).
# On your own computer:  pip install tensorflow keras scikit-learn scipy
matplotlib
# One framework per notebook: PyTorch and TensorFlow can clash when loaded into
the same Python process.
import os
os.environ["KERAS_BACKEND"] = "tensorflow"   # Keras 3 also runs on "jax" or
"torch": change it here
os.environ.setdefault("TF_CPP_MIN_LOG_LEVEL", "3")
import numpy as np, matplotlib.pyplot as plt
import tensorflow as tf, keras
keras.utils.set_random_seed(0)
print("TensorFlow", tf.__version__, "| Keras", keras.__version__, "| backend:",
keras.backend.backend(),
      "| GPU:", bool(tf.config.list_physical_devices("GPU")))
```

1. RNN, GRU and LSTM forecasting

```
t = np.arange(0, 400, 0.1)
signal = np.sin(t) + 0.5 * np.sin(3.1 * t) + 0.1 *
np.random.default_rng(0).normal(size=t.size)
W = 40
Xs = np.stack([signal[i:i + W] for i in range(len(signal) - W)])[:, :,
None].astype("float32"); ys = signal[W:].astype("float32")
n = int(0.8 * len(Xs))
for name, layer in (("SimpleRNN", keras.layers.SimpleRNN), ("GRU",
keras.layers.GRU), ("LSTM", keras.layers.LSTM)):
    keras.utils.set_random_seed(0)
    m = keras.Sequential([keras.Input((W, 1)), layer(32), keras.layers.Dense(1)])
    m.compile(keras.optimizers.Adam(3e-3), "mse")
    h = m.fit(Xs[:n], ys[:n], epochs=3, batch_size=64, verbose=0)
    print(f"{name:<10} test MSE {m.evaluate(Xs[n:], ys[n:], verbose=0):.4f}")
```

2. A tiny transformer with MultiHeadAttention

The counting task: for every position, how many times its token appears in the whole sequence.

```
V_, L_ = 6, 12
rng = np.random.default_rng(0)
def batch(n):
    x = rng.integers(0, V_, (n, L_)); return x, (x[:, :, None] == x[:, None,
:]).sum(-1) - 1
inp = keras.Input((L_,), dtype="int32")
h = keras.layers.Embedding(V_, 64)(inp) + keras.layers.Embedding(L_, 64)
(keras.ops.arange(L_))
for _ in range(2):
    h = keras.layers.LayerNormalization()(h +
keras.layers.MultiHeadAttention(num_heads=4, key_dim=16)(h, h))
    h = keras.layers.LayerNormalization()(h + keras.layers.Dense(64)
(keras.layers.Dense(128, activation="relu")(h)))
model = keras.Model(inp, keras.layers.Dense(L_)(h))
model.compile(keras.optimizers.AdamW(1e-3),
keras.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=
["accuracy"])
x, y = batch(20000); model.fit(x, y, epochs=8, batch_size=64, verbose=0)
x, y = batch(1000); print("accuracy:", round(model.evaluate(x, y, verbose=0)[1],
3))
```

3. Text in, numbers out: TextVectorization

```
vec = keras.layers.TextVectorization(max_tokens=50, output_sequence_length=6)
vec.adapt(["the dessert is excellent", "completely lacking in good taste", "good
service"])
print(vec.get_vocabulary()[:12]); print(vec(["the service is good"]).numpy())
```