Convolutional networks 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. A CNN on digits

Keras images are channels-last: `(batch, height, width, channels)`.

```
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
d = load_digits()
Xa, Xb, ya, yb = train_test_split((d.images / 16.0).astype("float32")[..., None],
d.target, test_size=0.3, random_state=0)
cnn = keras.Sequential([keras.Input((8, 8, 1)),
    keras.layers.Conv2D(16, 3, padding="same", activation="relu"),
keras.layers.MaxPooling2D(),
    keras.layers.Conv2D(32, 3, padding="same", activation="relu"),
keras.layers.MaxPooling2D(),
    keras.layers.Flatten(), keras.layers.Dense(10)])
cnn.compile("adamw", keras.losses.SparseCategoricalCrossentropy(from_logits=True),
metrics=["accuracy"])
cnn.fit(Xa, ya, epochs=60, batch_size=64, verbose=0)
print(f"parameters {cnn.count_params():,} | test accuracy {cnn.evaluate(Xb, yb,
verbose=0)[1]:.3f}")
```

2. A residual block with the functional API

```
def residual_block(x, filters):
    shortcut = x
    x = keras.layers.Conv2D(filters, 3, padding="same", use_bias=False)(x)
    x = keras.layers.ReLU()(keras.layers.BatchNormalization()(x))
    x = keras.layers.BatchNormalization()(keras.layers.Conv2D(filters, 3,
padding="same", use_bias=False)(x))
    return keras.layers.ReLU()(keras.layers.Add()([x, shortcut]))     # the skip
connection
inp = keras.Input((8, 8, 16)); print("shape kept:", keras.Model(inp,
residual_block(inp, 16)).output_shape)
```

3. Depthwise separable convolution

```
x = keras.Input((32, 32, 64))
normal = keras.Model(x, keras.layers.Conv2D(128, 3, padding="same")(x))
separable = keras.Model(x, keras.layers.SeparableConv2D(128, 3, padding="same")
(x))
print(f"weights: normal {normal.count_params():,} vs separable
{separable.count_params():,}")
```

4. Augmentation layers

```
aug = keras.Sequential([keras.layers.RandomRotation(0.05),
keras.layers.RandomZoom(0.15), keras.layers.RandomContrast(0.3)])
fig, ax = plt.subplots(1, 8, figsize=(10, 1.5))
for i, a in enumerate(ax):
    a.imshow((Xa[0] if i == 0 else aug(Xa[:1], training=True)[0])[..., 0],
cmap="gray_r"); a.axis("off")
plt.show()
```

5. Non-max suppression with TensorFlow

TensorFlow's boxes are `[y1, x1, y2, x2]`.

```
boxes = tf.constant([[30, 30, 120, 130], [38, 40, 128, 140], [25, 35, 125, 125],
[60, 200, 160, 290], [70, 210, 165, 300]], tf.float32)
scores = tf.constant([0.9, 0.75, 0.6, 0.85, 0.7])
print("kept:", tf.image.non_max_suppression(boxes, scores, max_output_size=10,
iou_threshold=0.5).numpy().tolist())
```

6. Transfer learning with keras.applications

Needs internet to download the ImageNet weights (runs in Colab).

```
try:
    base = keras.applications.MobileNetV3Small(input_shape=(224, 224, 3),
include_top=False, weights="imagenet")
    base.trainable = False
    inp = keras.Input((224, 224, 3))
```

```
    out = keras.layers.Dense(3)(keras.layers.GlobalAveragePooling2D()(base(inp,
training=False)))
    model = keras.Model(inp, out)
    print("trainable weights:", sum(int(np.prod(w.shape)) for w in
model.trainable_weights))
except Exception as e:
    print("Needs internet to download the weights (run this in Colab):",
type(e).__name__)
```