Google Developers

Writing Models in Keras

ML Bootcamp India (2022)

Sayak Paul
ML Engineer at Carted
@RisingSayak

$whoami

- ML Engineer at Carted
- Open-source 🥑 (Keras, KerasCV, 🤗 Transformers etc.)
- Netflix nerd
- Coordinates at sayak.dev

Agenda

- Several ways to write models in Keras
- When to use what
- Code
- QnA

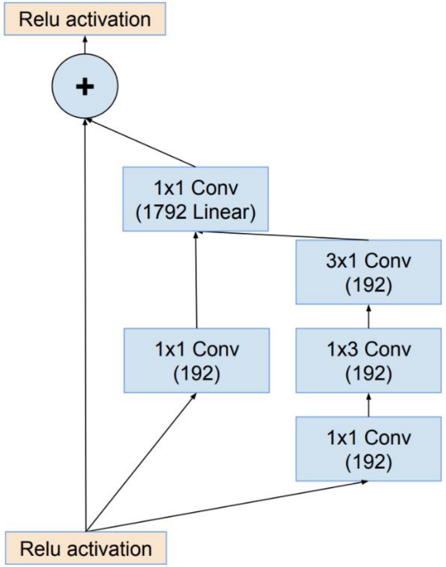
Materials are here:

bit.ly/code-mlp-2022

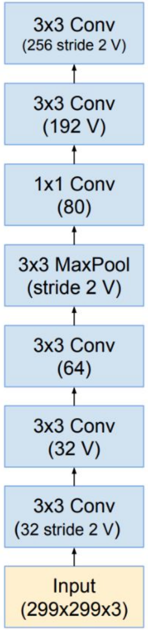
Two broad styles: Symbolic and Imperative

- Symbolic:
  - Your model is sequential stack of layers (Sequential)
  - Your model is basically a DAG (Functional)

Two broad styles: Symbolic and Imperative



DAG



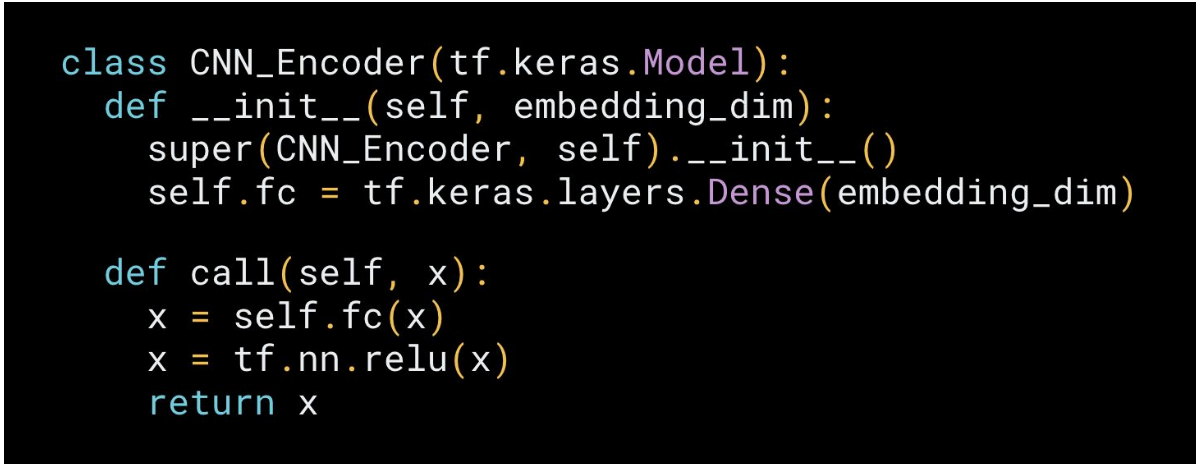
Stack

Credits: [What are Symbolic and Imperative APIs in TensorFlow 2.0?]

Two broad styles: Symbolic and Imperative

- Symbolic
- Imperative:
    - You need to have granular controls over how each component in your model is executed.

Two broad styles: Symbolic and Imperative



```
class CNN_Encoder(tf.keras.Model):
  def __init__(self, embedding_dim):
    super(CNN_Encoder, self).__init__()
    self.fc = tf.keras.layers.Dense(embedding_dim)

  def call(self, x):
    x = self.fc(x)
    x = tf.nn.relu(x)
    return x
```

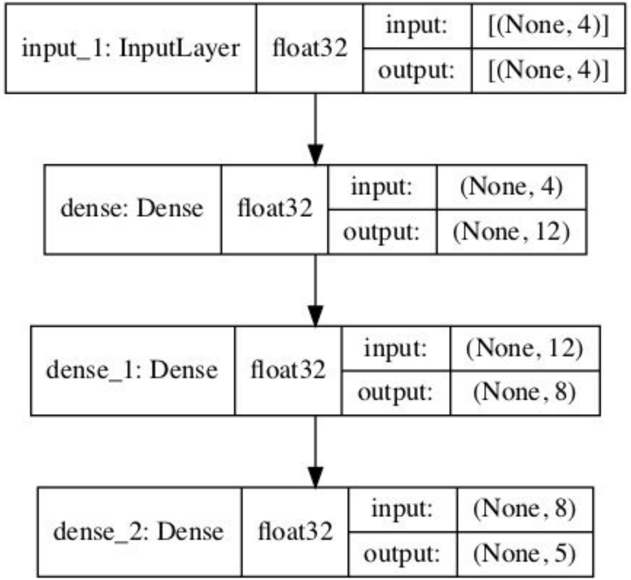
Credits: [What are Symbolic and Imperative APIs in TensorFlow 2.0?]()

Two broad styles: Symbolic and Imperative

For a more rigorous treatment of these two styles, check out:

What are Symbolic and Imperative APIs in TensorFlow 2.0?.

Models with `tf.keras.Sequential`



input_1: InputLayer | float32 | input: [(None, 4)] | output: [(None, 4)]

dense: Dense | float32 | input: (None, 4) | output: (None, 12)

dense_1: Dense | float32 | input: (None, 12) | output: (None, 8)

dense_2: Dense | float32 | input: (None, 8) | output: (None, 5)

A sequential stack of layers

```python
num_features = 4
num_classes = 5

sequential_model = keras.Sequential([
    keras.layers.InputLayer(input_shape=(num_features,)),
    keras.layers.Dense(12, activation="relu"),
    keras.layers.Dense(8, activation="relu"),
    keras.layers.Dense(num_classes, activation="softmax")
], name="sequential_model")
```

Image classification model

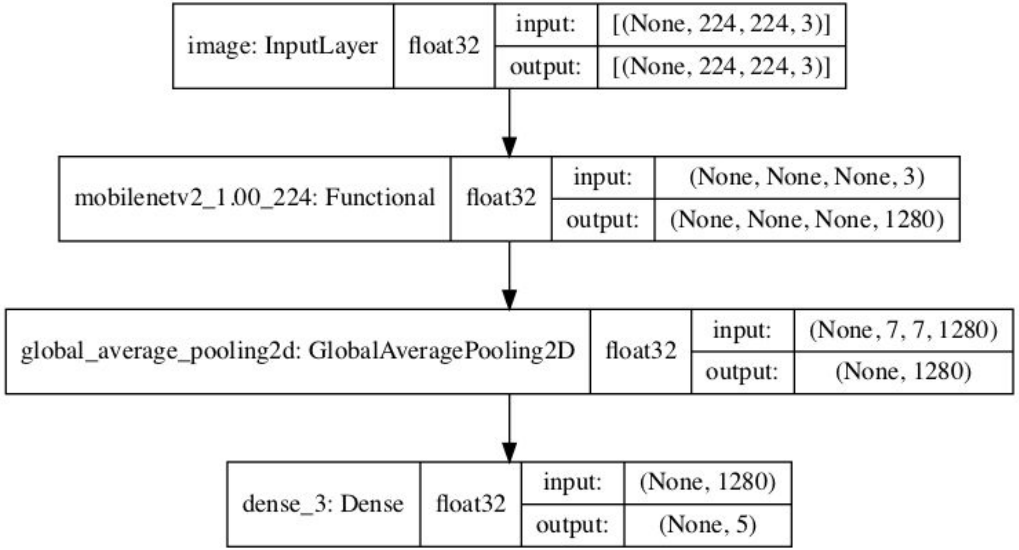
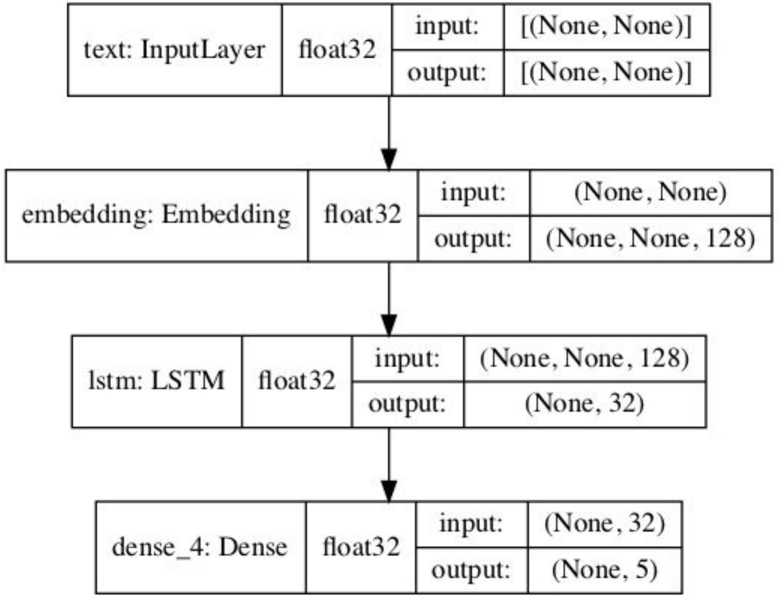


image: InputLayer	float32	input:	[(None, 224, 224, 3)]
		output:	[(None, 224, 224, 3)]
mobilenetv2_1.00_224: Functional	float32	input:	(None, None, None, 3)
		output:	(None, None, None, 1280)
global_average_pooling2d: GlobalAveragePooling2D	float32	input:	(None, 7, 7, 1280)
		output:	(None, 1280)
dense_3: Dense	float32	input:	(None, 1280)
		output:	(None, 5)

Pretrained backbone +
Classification top

```
mobilenet = keras.applications.MobileNetV2(weights="imagenet", include_top=False)
image_resolution = 224

custom_mobilenet_sequential = keras.Sequential(
    [
        keras.layers.InputLayer(
            input_shape=(image_resolution, image_resolution, 3), name="image"
        ),
        mobilenet,
        keras.layers.GlobalAveragePooling2D(),
        keras.layers.Dense(num_classes, activation="softmax"),
    ],
    name="custom_mobilenet_sequential",
)
```

Text classification model



text: InputLayer | float32 | input: [(None, None)] | output: [(None, None)]

embedding: Embedding | float32 | input: (None, None) | output: (None, None, 128)

lstm: LSTM | float32 | input: (None, None, 128) | output: (None, 32)

dense_4: Dense | float32 | input: (None, 32) | output: (None, 5)

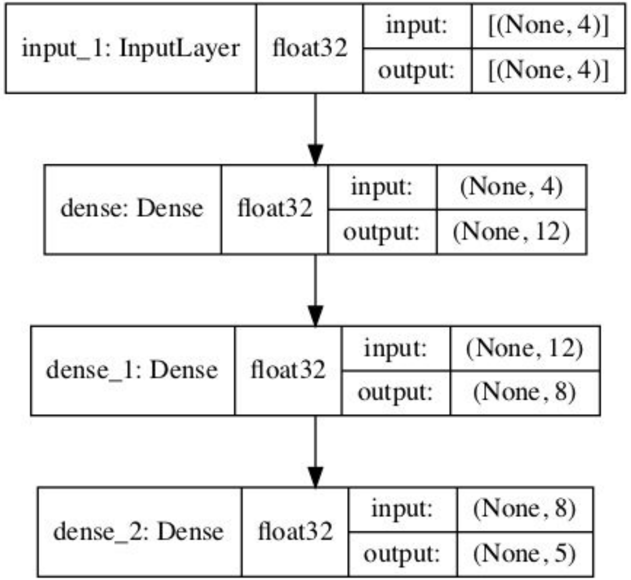
A sequential stack of layers

```
embedding_dim = 128
input_dim = 512
lstm_units = 32

text_model_sequential = keras.Sequential(
    [
        keras.layers.InputLayer(input_shape=(None,), name="text"),
        keras.layers.Embedding(input_dim=input_dim, output_dim=embedding_dim),
        keras.layers.LSTM(units=lstm_units),
        keras.layers.Dense(num_classes, activation="softmax"),
    ],
    name="text_model_sequential",
)
```

Models with the `Functional` API

- What if you need to return multiple outputs from a single model?
- What if your model needs to accept multiple inputs of multiple modalities (image, text, audio, etc.)?
- What if you need to add a skip connection in between the intermediate layer outputs?
- ...

Multi-output model with the `Functional` API



input_1: InputLayer | float32 | input: [(None, 4)] | output: [(None, 4)]

dense: Dense | float32 | input: (None, 4) | output: (None, 12)

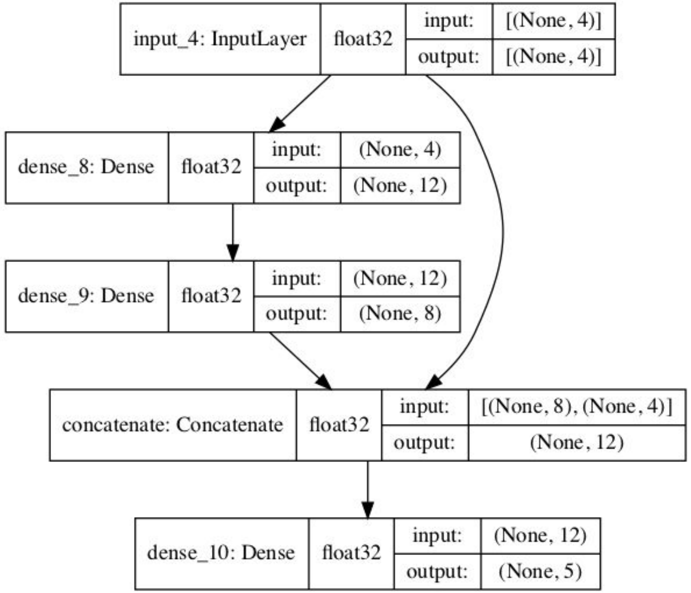
dense_1: Dense | float32 | input: (None, 12) | output: (None, 8)

dense_2: Dense | float32 | input: (None, 8) | output: (None, 5)

Model computation graph unchanged

```python
num_features = 4
num_classes = 5

inputs = keras.Input((num_features,))
x1 = keras.layers.Dense(12, activation="relu")(inputs)
x2 = keras.layers.Dense(8, activation="relu")(x1)
outputs = keras.layers.Dense(num_classes, activation="softmax")(x2)
functional_model_multi = keras.Model(
    inputs, outputs=[x1, x2, outputs], name="functional_model"
)
```

Model with combined intermediate outputs



input_4: InputLayer | float32 | input: [(None, 4)] | output: [(None, 4)]

dense_8: Dense | float32 | input: (None, 4) | output: (None, 12)

dense_9: Dense | float32 | input: (None, 12) | output: (None, 8)

concatenate: Concatenate | float32 | input: [(None, 8), (None, 4)] | output: (None, 12)

dense_10: Dense | float32 | input: (None, 12) | output: (None, 5)

The computational flow is not entirely sequential

```
num_features = 4
num_classes = 5

inputs = keras.Input((num_features,))
x = keras.layers.Dense(12, activation="relu")(inputs)
x = keras.layers.Dense(8, activation="relu")(x)
concatenated_features = keras.layers.Concatenate()([x, inputs])

outputs = keras.layers.Dense(num_classes, activation="softmax")(concatenated_features)
functional_model = keras.Model(inputs, outputs, name="functional_model")
```

Model with multiple inputs

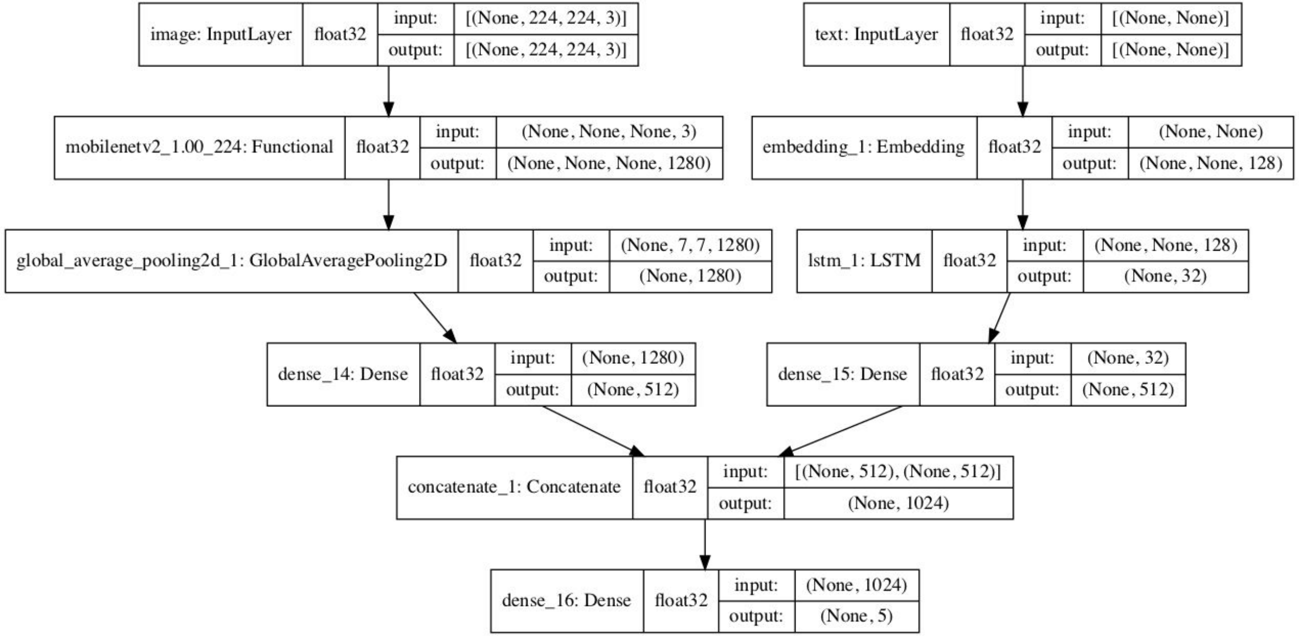


image: InputLayer | float32 | input: [(None, 224, 224, 3)] | output: [(None, 224, 224, 3)]

mobilenetv2_1.00_224: Functional | float32 | input: (None, None, None, 3) | output: (None, None, None, 1280)

global_average_pooling2d_1: GlobalAveragePooling2D | float32 | input: (None, 7, 7, 1280) | output: (None, 1280)

dense_14: Dense | float32 | input: (None, 1280) | output: (None, 512)

text: InputLayer | float32 | input: [(None, None)] | output: [(None, None)]

embedding_1: Embedding | float32 | input: (None, None) | output: (None, None, 128)

lstm_1: LSTM | float32 | input: (None, None, 128) | output: (None, 32)

dense_15: Dense | float32 | input: (None, 32) | output: (None, 512)

concatenate_1: Concatenate | float32 | input: [(None, 512), (None, 512)] | output: (None, 1024)

dense_16: Dense | float32 | input: (None, 1024) | output: (None, 5)

A model accepting images and texts as inputs

```
# Inputs
image_inputs = keras.Input((image_resolution, image_resolution, 3), name="image")
text_inputs = keras.Input((None,), name="text")

# MobileNet model for images
image_model = sequential_image_model()

# Image representations
image_representations = image_model(image_inputs)
projected_image_representations = keras.layers.Dense(projection_dim, activation="relu")(
    image_representations
)

# Text representations
lstm_output = functional_text_model(text_inputs)
projected_text_representations = keras.layers.Dense(projection_dim, activation="relu")(
    lstm_output
)
…
```

```
…

# Concatenate image and text representations
concatenated_projections = keras.layers.Concatenate()(
    [projected_image_representations, projected_text_representations]
)

# Classification top
outputs = keras.layers.Dense(num_classes, activation="softmax")(concatenated_projections)
multimodal_model = keras.Model([image_inputs, text_inputs], outputs, name="multimodal_model")
```

Models with subclassing from `keras.Model`

- Allows you to take more granular controls of what happens inside your model.
- Particularly useful if your model has a tree-like or recursive structure.

```
class ShallowMLP(keras.Model):

    def __init__(self, num_classes=num_classes, **kwargs):
        super().__init__(**kwargs)
        self.dense1 = keras.layers.Dense(12, activation="relu")
        self.dense2 = keras.layers.Dense(8, activation="relu")
        self.classification_layer = keras.layers.Dense(
            num_classes, activation="softmax"
        )

    def call(self, inputs):
        x = self.dense1(inputs)
        x = self.dense2(x)
        outputs = self.classification_layer(x)
        return outputs
```

Some gotchas

- No structured outputs when you call `summary()`.
- No concrete model computation graph with `plot_model()`.
- More during the code walkthrough and here.

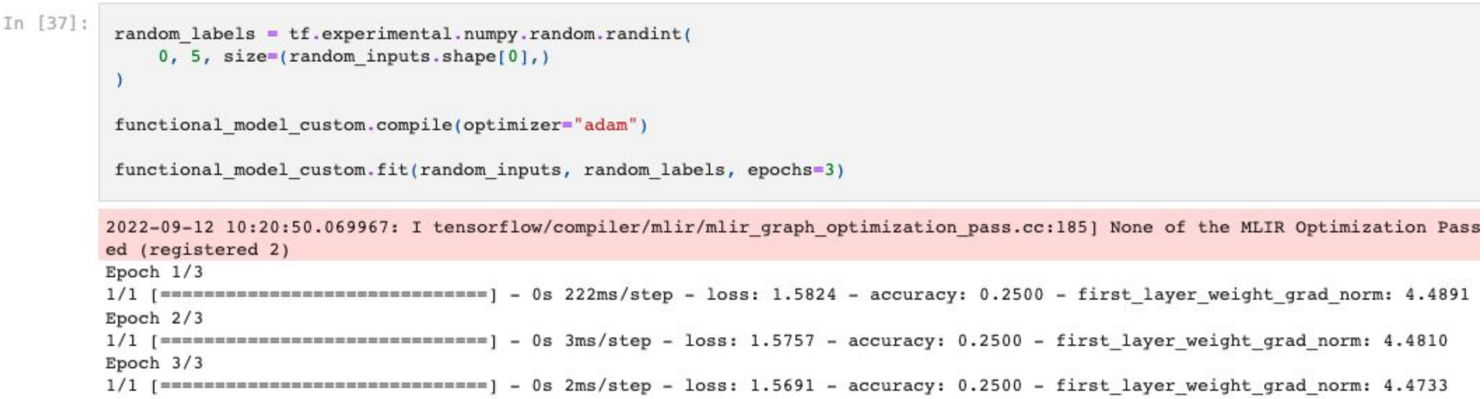
Customize the training behaviour

- Subclass from `keras.Model` overriding the `train_step()` method.
- Control the evaluation loop by overriding `test_step()`.

```python
class Trainer(keras.Model):
    def train_step(self, data):
        x, y = data

        with tf.GradientTape() as tape:
            # Forward pass
            y_pred = self(x, training=True)
            # Compute loss
            loss = keras.losses.sparse_categorical_crossentropy(y, y_pred)

        # Compute gradients
        trainable_vars = self.trainable_variables
        gradients = tape.gradient(loss, trainable_vars)
        # Update weights
        self.optimizer.apply_gradients(zip(gradients, trainable_vars))
        # Compute gradient norm
        norm_tracker.update_state(tf.norm(gradients[0]))
        # Compute our own metrics
        loss_tracker.update_state(loss)
        accuracy_metric.update_state(y, y_pred)

        return {
            "loss": loss_tracker.result(),
            "accuracy": accuracy_metric.result(),
            "first_layer_weight_grad_norm": norm_tracker.result(),
        }
```

```python
inputs = keras.Input((num_features,))
x = keras.layers.Dense(12, activation="relu")(inputs)
x = keras.layers.Dense(8, activation="relu")(x)
outputs = keras.layers.Dense(num_classes, activation="softmax")(x)

# Notice that we're using `Trainer` instead of `keras.Model`.
functional_model_custom = Trainer(inputs, outputs, name="functional_model_custom")
```

Now call `fit()`



```
In [37]:
random_labels = tf.experimental.numpy.random.randint(
    0, 5, size=(random_inputs.shape[0],)
)

functional_model_custom.compile(optimizer="adam")

functional_model_custom.fit(random_inputs, random_labels, epochs=3)
```

```
2022-09-12 10:20:50.069967: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:185] None of the MLIR Optimization Pass
ed (registered 2)
Epoch 1/3
1/1 [==============================] - 0s 222ms/step - loss: 1.5824 - accuracy: 0.2500 - first_layer_weight_grad_norm: 4.4891
Epoch 2/3
1/1 [==============================] - 0s 3ms/step - loss: 1.5757 - accuracy: 0.2500 - first_layer_weight_grad_norm: 4.4810
Epoch 3/3
1/1 [==============================] - 0s 2ms/step - loss: 1.5691 - accuracy: 0.2500 - first_layer_weight_grad_norm: 4.4733
```

Enough talk, show me the code!

bit.ly/models-colab

Questions?

Google Developers

Thank you!

Sayak Paul
ML Engineer at Carted
@RisingSayak