Applied ML with keras

Divyashree Sreepathihalli
Software Engineer, Google

Setup

Google Colab Signup - https://colab.research.google.com/

Github Repo for todays workshop - https://github.com/divyashreepathihalli/women_tech_makers

Introduction to Keras - overview

Keras mission

- What is Keras
- Why Keras

Modeling with Keras

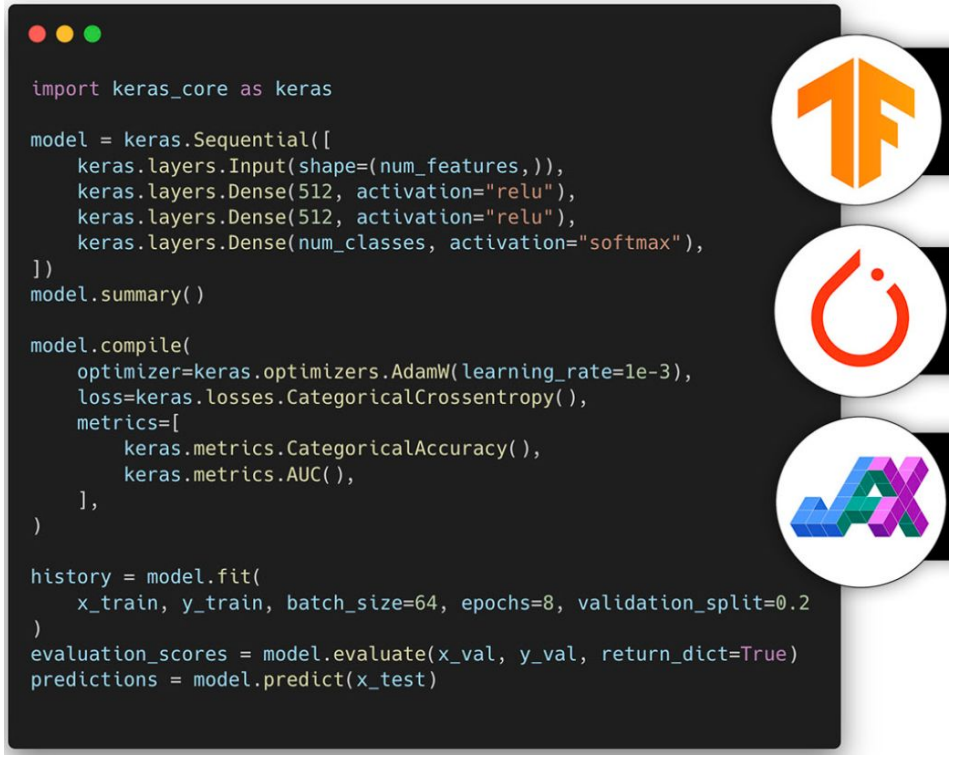
- Keras modeling API. layers, functional and so on.
- Best practice of building models with Keras modeling APIs.

Training with Keras

- Keras training API.
- Best practice on training models with Keras training API
- Data pipeline and distributed training.

What is Keras and Keras Mission

- Keras is built on top of auto-differentiation libraries: Tensorflow, JAX, PyTorch and numpy and provides high-level modeling and training APIs. (numpy backend is inference only)
- Our goal is to ease the process of building and training your model by offering consistent & simple APIs .
- Keras went multi-backend in 07/2023, with same API contract as `tf.keras`. We will keep maintaining tf keras, but shifting the focus to multi-backend Keras.



```
import keras_core as keras

model = keras.Sequential([
    keras.layers.Input(shape=(num_features,)),
    keras.layers.Dense(512, activation="relu"),
    keras.layers.Dense(512, activation="relu"),
    keras.layers.Dense(num_classes, activation="softmax"),
])
model.summary()

model.compile(
    optimizer=keras.optimizers.AdamW(learning_rate=1e-3),
    loss=keras.losses.CategoricalCrossentropy(),
    metrics=[
        keras.metrics.CategoricalAccuracy(),
        keras.metrics.AUC(),
    ],
)

history = model.fit(
    x_train, y_train, batch_size=64, epochs=8, validation_split=0.2
)
evaluation_scores = model.evaluate(x_val, y_val, return_dict=True)
predictions = model.predict(x_test)
```

Keras Ecosystem

Modeling API

- Keras layers
- Functional/Sequential API
- Keras model interface
- Keras applications

Training API

- model.fit(), model.predict()
- Loss functions, metrics
- Optimizers
- Callbacks
- Preprocessing layers

Domain packages

- KerasCV
- KerasNLP

Keras Modeling API

Goal: reduces the difficulty of defining your model.

- `keras.layers.Layer`
- `keras.Model`

Keras layers

Layer is the main building block for Keras models, you can view a model as graph of nodes, and each node is a layer.

Layer is a combination of computations so that you can reuse it.

`keras.layers.Layer` does auto variables tracking and comes with support on initializers, constraints, regularizers.

```
from keras_core import layers

layer = layers.Dense(32, activation='relu')
inputs = np.random.uniform(shape=(10, 20))
outputs = layer(inputs)
```

```
keras.layers.Dense(
    units,
    activation=None,
    use_bias=True,
    kernel_initializer="glorot_uniform",
    bias_initializer="zeros",
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    **kwargs
)
```

Build Keras model - Sequential API

Keras sequential API is suitable for models that:

- Every layer has only 1 output and 1 input
- The output of each layer is only taken by one layer.

```
sequential_model = keras.Sequential([
    keras.Input([28, 28, 3]),
    keras.layers.Conv2D(8, 2),
    keras.layers.MaxPool2D(2),
    keras.layers.Flatten(),
    keras.layers.Dense(2),
    keras.layers.Softmax(),
])

print(sequential_model.summary())
print(sequential_model(np.random.uniform(size=[1, 28, 28, 3])))
```

Build Keras model - Functional API (My favorite)

The functional API can handle models with non-linear topology, shared layers, and even multiple inputs or outputs.

The functional API is a way to build graphs of layers.

```
inputs = keras.Input(shape=(784,))
x = keras.layers.Dense(64, activation="relu")(x)
x = keras.layers.Dense(10)(x)
outputs = keras.layers.Softmax()(x)

model = keras.Model(inputs=inputs, outputs=outputs)
```

```
token_ids = keras.Input(shape=(None,), dtype="int32", name="token_ids")
padding_mask = keras.Input(
    shape=(None,), dtype="int32", name="padding_mask"
)

outputs = keras.some_magic_layer()(token_ids, padding_mask)
model = keras.Model(
    inputs={
        "token_ids": token_ids,
        "padding_mask": padding_mask,
    },
    outputs=outputs,
)
```

Build Keras model - Subclassing model

For more flexibility, you can write your custom model by subclassing `keras.Model`.

If you are familiar with Torch, this is very similar.

```
class MyModel(keras.Model):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self._conv = keras.layers.Conv2D(8, 2)
        self._max_pool = keras.layers.MaxPool2D(2)
        self._flatten = keras.layers.Flatten()
        self._dense = keras.layers.Dense(2)
        self._softmax = keras.layers.Softmax()

    def call(self, inputs):
        x = self._conv(inputs)
        x = self._max_pool(x)
        x = self._flatten(x)
        x = self._dense(x)
        return self._softmax(x)
```

Build Keras model - Takeaways

1. By default go with functional model
2. For simple use case (single input, simple model), such as quick testing, use Sequential API.
3. Only use subclassing model when the above 2 do not fit, or you prefer the Torch style.

Training API - overview

A complete training workflow consists of:

- Training data (not by Keras)
- Model
- Forward pass
- Backward pass
- Logging

Keras Training API

- Training components
  - Loss, e.g, `keras.losses.SparseCategoricalCrossentropy`.
  - Optimizers, e.g., `keras.optimizers.Adam`.
  - Metrics, e.g., `keras.metrics.Accuracy`.
  - Callbacks, e.g., `keras.callbacks.TensorboardCallback`.
- Training entry point `compile()` and `fit()`
  - Configure your training with `compile()`.
  - Train with `fit()`.

Training API - configure with `model.compile`

- Set optimizer
- Set training target/loss
- Set metrics
- Run eagerly or in graph mode
- Use XLA or not

```
model.compile(
    optimizer=keras.optimizers.Adam(1e-3),
    loss=keras.losses.BinaryCrossentropy(),
    metrics=[keras.metrics.BinaryAccuracy()],
    run_eagerly=False,
    jit_compile=True,
)
```

Training API - train with `model.fit`

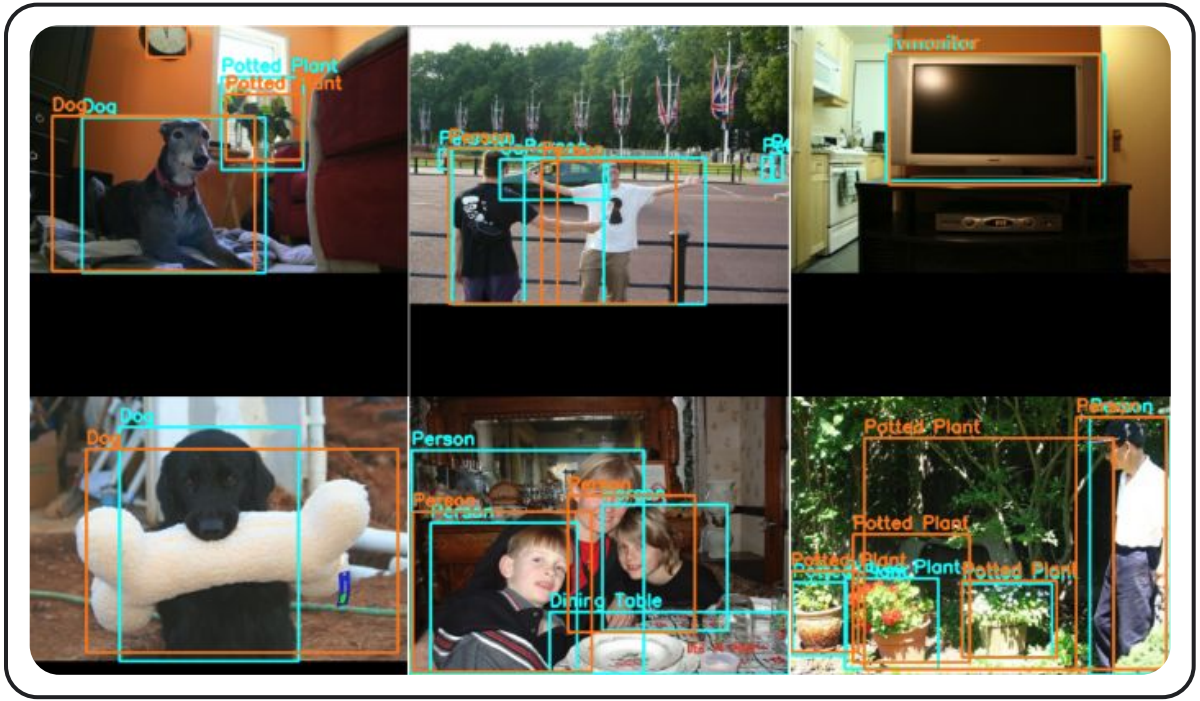
- Convert data into `tf.data.Dataset`, batching, shuffling...
- Trace the computation graph with `@tf.function`.
- Distributed training support.
- Validation after each epoch.
- Training hooks via callbacks.
- Log the training stats.

Applied ML with KerasCV & KerasNLP

What can you do with KerasCV?

Image Classification

Object Detection



Semantic segmentation

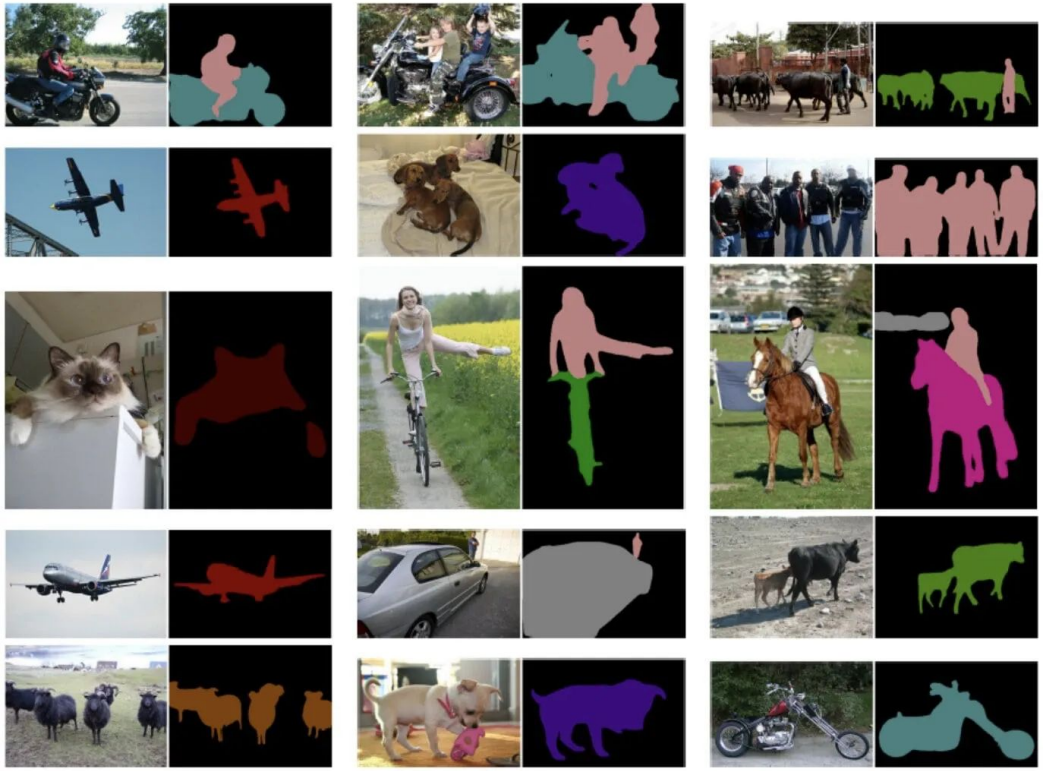
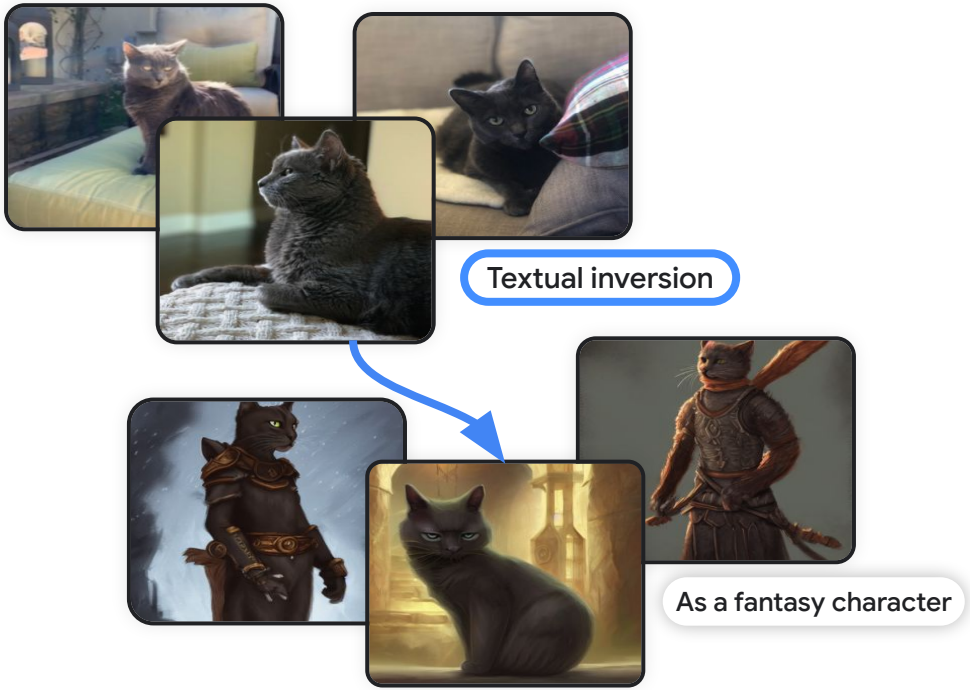


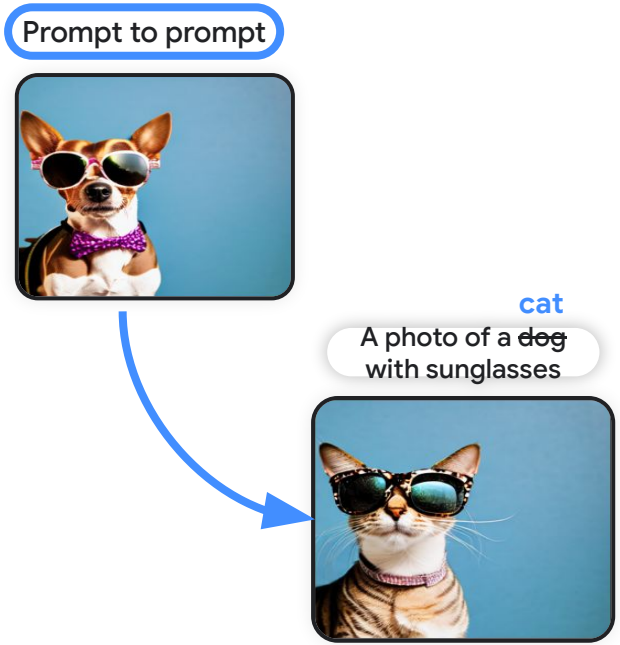
Image generation



```
from keras_cv.models import
(
    StableDiffusion,
)

model = StableDiffusion(
    img_width=512,
    img_height=512,
)
images =
model.text_to_image(
    "photograph of an
astronaut "
    "riding a horse",
    batch_size=3,
)
```





What can you do with KerasNLP?

Google

Text Classification

```
# Text classification
from keras_nlp.models import BertClassifier

model = BertClassifier.from_preset(
    "bert_base_en_uncased",
    num_classes=2,
    activation="softmax",
)

model.compile(...)

model.fit(imdb_movie_review_dataset)
model.predict(
    [
        "What an amazing movie!",
        "A total waste of my time.",
    ],
)
>>> array([[0.004173, 0.9956  ],
           [0.997   , 0.0028  ]],
dtype=float16)
```

Text generation

```
# Text generation
from keras_nlp.models import GPT2CausalLM

model = GPT2CausalLM.from_preset(
    "gpt2_base_en",
)

model.compile(...)

model.fit(cnn_dailymail_dataset)
model.generate(
    "Snowfall in Buffalo",
    max_length=40,
)

>>> 'Snowfall in Buffalo, New York, was
expected to reach '
    '2 feet by the end of the day, according
to the National '
    'Weather Service.'
```

Resources

Keras.io

- https://keras.io/keras_core/
- https://keras.io/keras_nlp/
- https://keras.io/keras_cv/

Github:

- https://github.com/keras-team/keras-core
- https://github.com/keras-team/keras-nlp
- https://github.com/keras-team/keras-cv