Women in ML

Keras: Shortcut to AI Mastery

Divyashree Sreepathihalli (she/her)
Software Engineer, Google
@divyashreess

Introduction

Software Engineer
Keras Maintainer
KerasCV Lead and Key Maintainer
Specialized in Keras Modelling APIs
and KerasCV

Setup

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

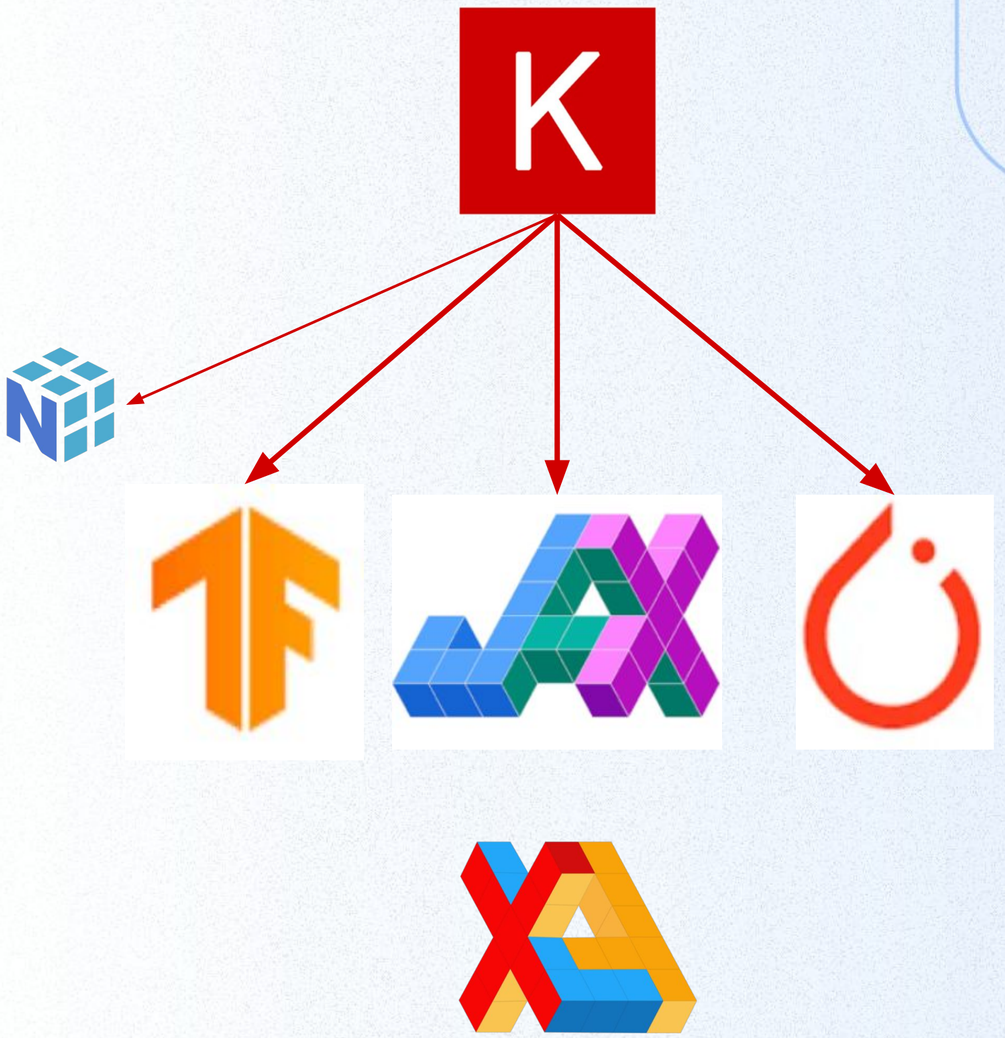
Github Repo for todays workshop - https://github.com/divyashreepathihalli/WIML_2023

Multi-backend Keras is back

1. Full rewrite of Keras

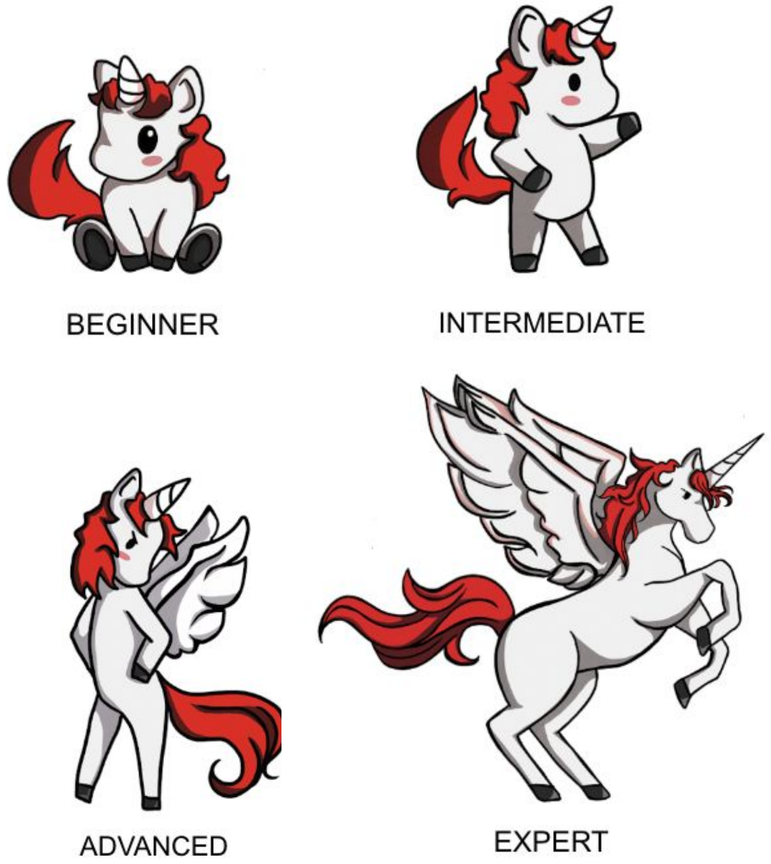
Now only 45k loc instead of 135k

2. Support for TensorFlow, JAX, PyTorch, NumPy backends(NumPy backend is inference-only)
3. Drop-in replacement for tf.keras when using TensorFlow backend
4. XLA compilation by default in TF and JAX



In today's workshop you will

1. Explore multi backend Keras 3.0 - how to write a simple model, and run it with a backend of your choice, including PyTorch, TensorFlow, or JAX
2. Learn deep learning best practices
3. Learn about the KerasNLP and KerasCV modules and how to build powerful AI applications



Keras mission

Our goal is to ease the process of building and training your model by offering consistent & simple APIs

Keras Ecosystem

Modelling API

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

Training API

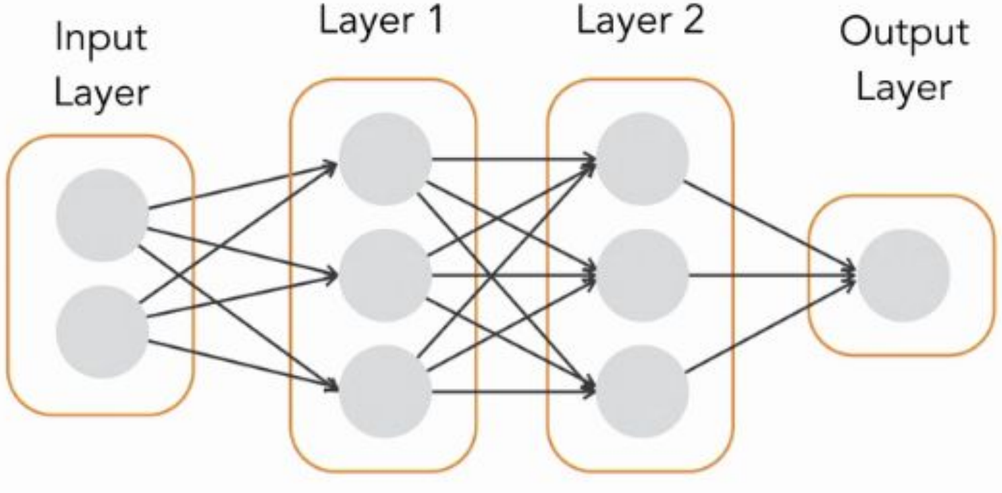
- model.fit(), model.predict()
- Loss functions, metrics
- Optimizers
- Callbacks
- Preprocessing layers

Domain Packages

- KerasCV
- KerasNLP

Keras Modelling APIs

Keras layers API




```
from tensorflow.keras import layers

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

Keras Models API

`Sequential`

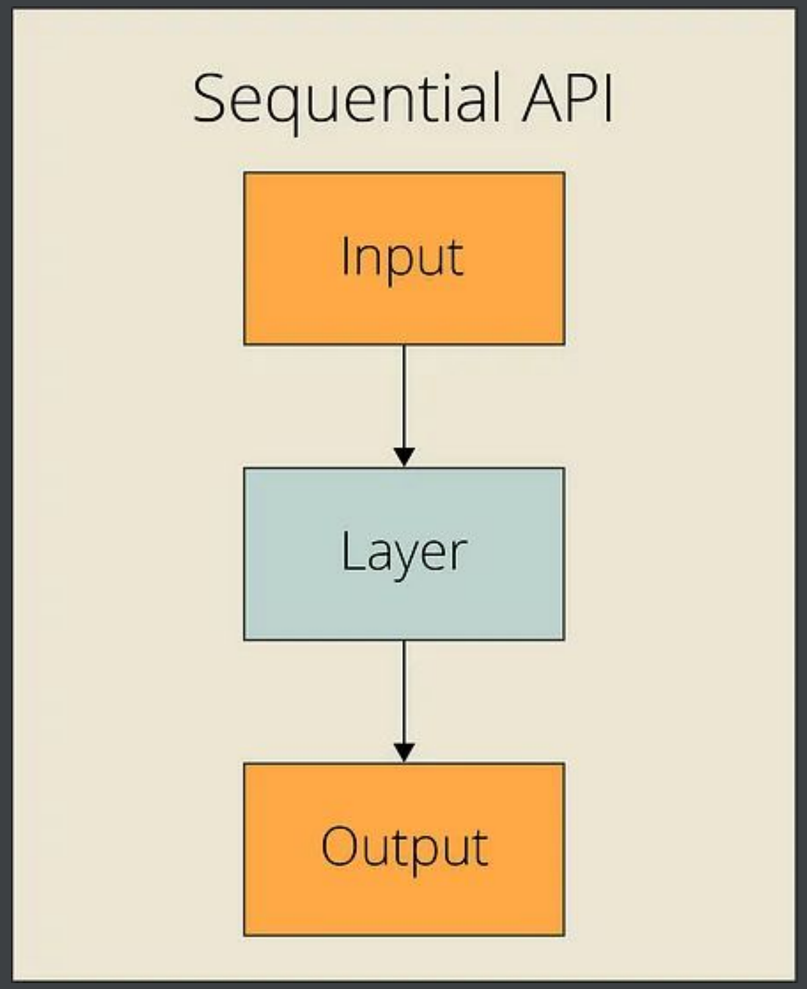
which is very straightforward (a simple list of layers), but is limited to single-input, single-output stacks of layers (as the name gives away)

`Functional`

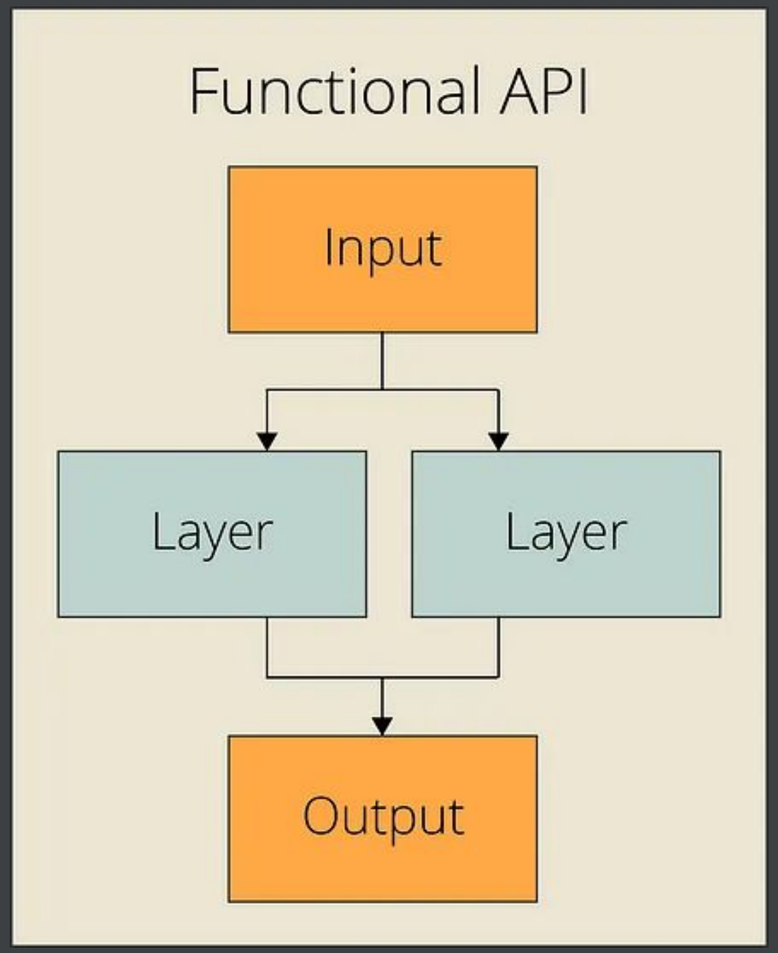
an easy-to-use, fully-featured API that supports arbitrary model architectures. For most people and most use cases, this is what you should be using. This is the Keras "industry strength" model.

`Subclass Model`

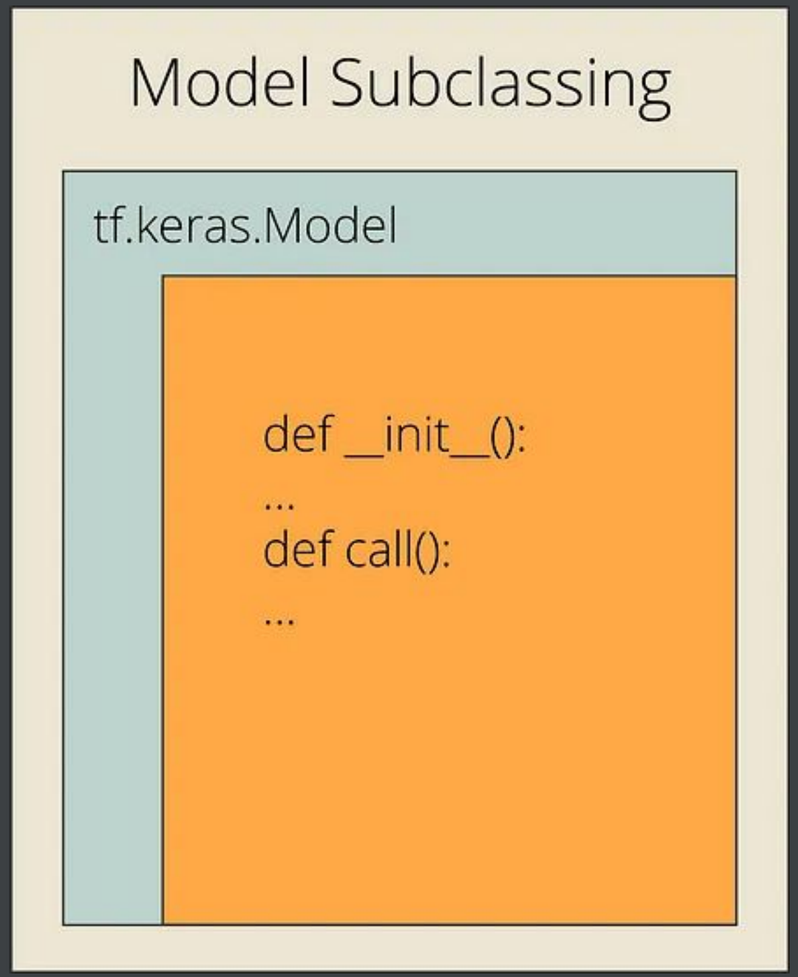
where you implement everything from scratch on your own. Use this if you have complex, out-of-the-box research use cases.




```
model = tf.keras.Sequential()
model.add(tf.keras.Input(shape=(16,)))
model.add(tf.keras.layers.Dense(8))
model.summary()
```



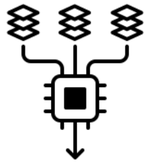

```
inputs = keras.Input([28, 28, 3])
x = keras.layers.Conv2D(8, 2)(inputs)
x = keras.layers.MaxPool2D(2)(x)
x = keras.layers.Flatten()(x)
x = keras.layers.Dense(2)(x)
outputs = keras.layers.Softmax()(x)

functional_model =
keras.Model(inputs=inputs,
outputs=outputs)
functional_model.summary()
```



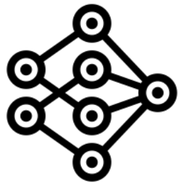

```
class MyModel(keras.Model):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self._dense = keras.layers.Dense(2)
        self._softmax = keras.layers.Softmax()

    def call(self, inputs):
        x = self._dense(x)
        return self._softmax(x)

subclass_model = MyModel()


subclass_model.summary()
```

Keras Training APIs

Training workflow



Training Data



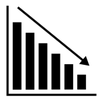
model

Forward pass

Backward pass

logging

Keras Training components



Loss

Optimizers

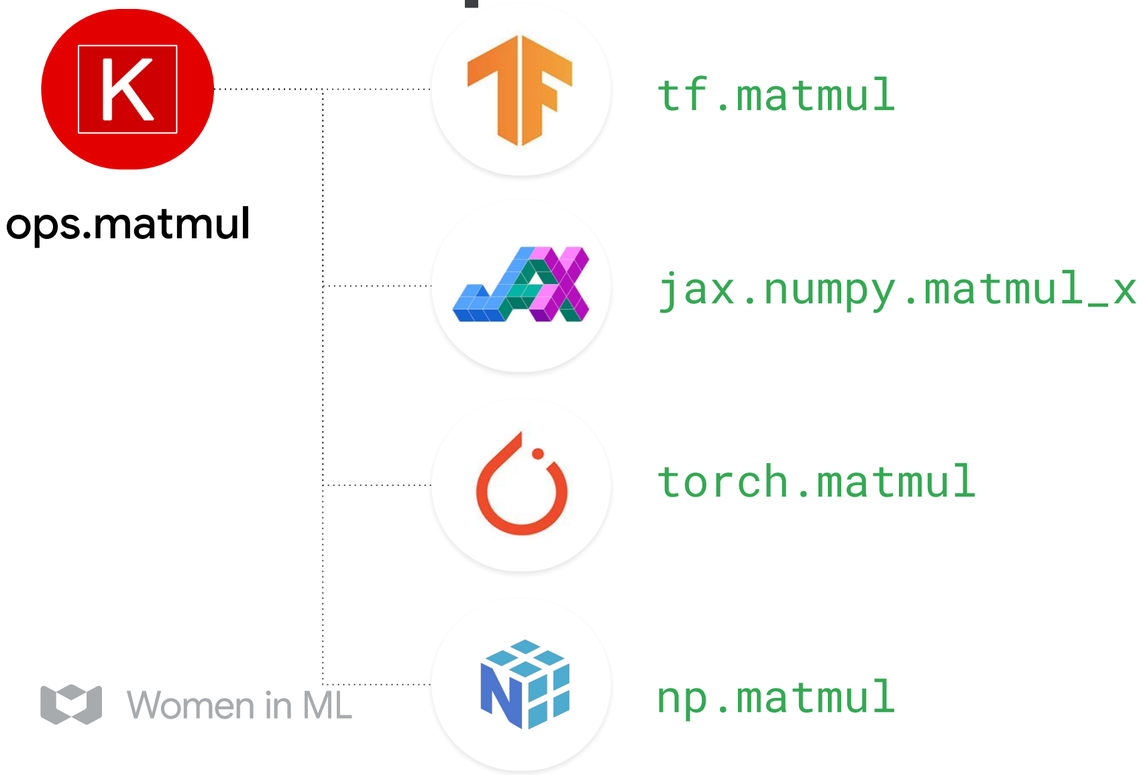
Metrics

callbacks

```
Model.compile(
    optimizer="rmsprop",
    loss=None,
    metrics=None,
    loss_weights=None,
    weighted_metrics=None,
    run_eagerly=None,
    steps_per_execution=None,
    jit_compile=None,
    pss_evaluation_shards=0,
    **kwargs
)
```

```
Model.fit(
    x=None,
    y=None,
    batch_size=None,
    epochs=1,
    callbacks=None,
    validation_data=None,
    steps_per_epoch=None,
    validation_steps=None,
)
```

Develop cross-framework components with keras.ops



Includes the NumPy API – same functions, same arguments

- `ops.matmul`, `ops.sum`, `ops.stack`, `ops.einsum`, etc.

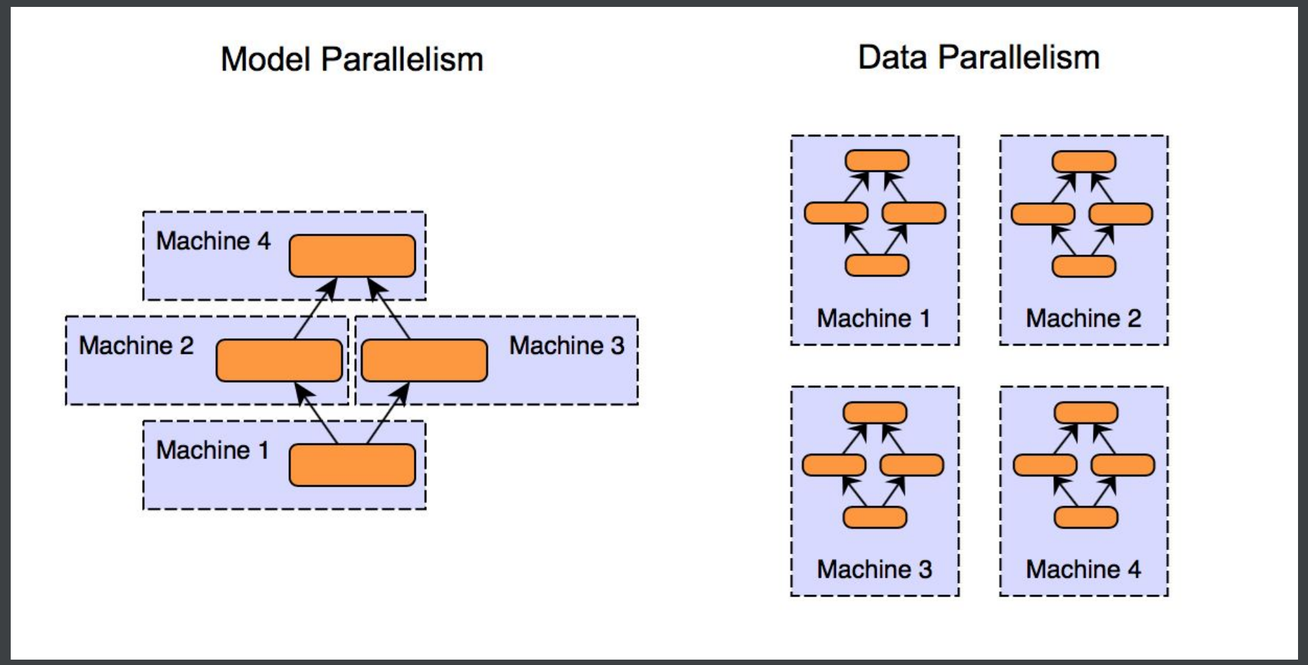
IPlus neural network-specific functions absent from NumPy

- `ops.softmax`, `oops.binary_crossentropy`, `ops.conv`, etc.

Models / layers / losses / metrics / optimizers written with Keras APIs **work the same with any framework**

- They can even be used outside of Keras workflows!

Unified Distribution API




```
mesh_2d = keras.distribution.DeviceMesh(
    shape=(2, 4), axis_names=["data", "model"], devices=devices
)
layout_map = keras.distribution.LayoutMap(mesh_2d)
layout_map["d1/kernel"] = (None, "model")
layout_map["d1/bias"] = ("model",)
layout_map["d2/output"] = ("data", None)

model_parallel = keras.distribution.ModelParallel(
    mesh_2d, layout_map, batch_dim_name="data"
)

keras.distribution.set_distribution(model_parallel)

inputs = layers.Input(shape=(28, 28, 1))
y = layers.Flatten()(inputs)
y = layers.Dense(units=200, use_bias=False, activation="relu",
name="d1")(y)
y = layers.Dropout(0.4)(y)
y = layers.Dense(units=10, activation="softmax", name="d2")(y)
model = keras.Model(inputs=inputs, outputs=y)

model.compile(loss="mse")
model.fit(dataset, epochs=3)
model.evaluate(dataset)
```

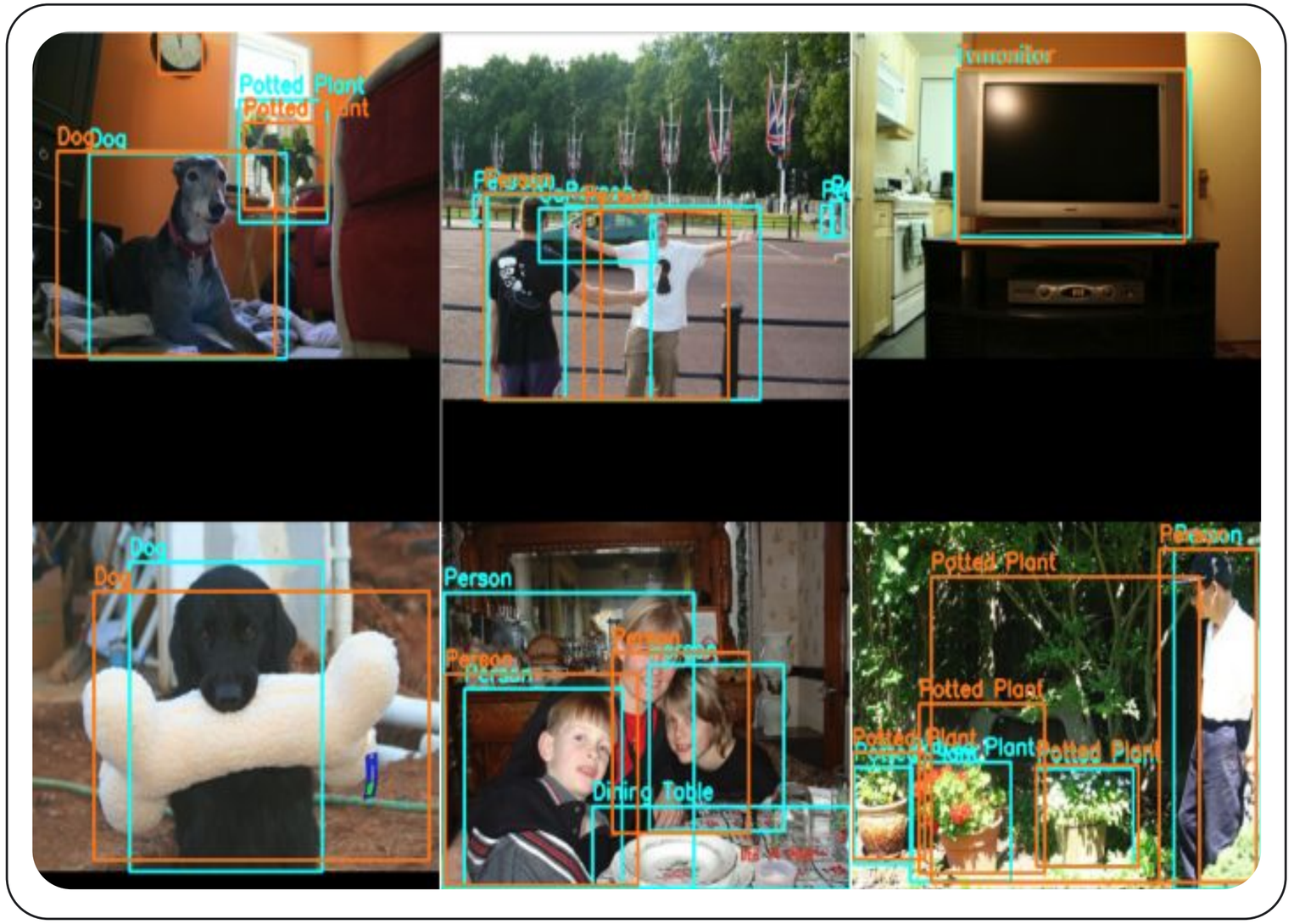
Keras Applications


efficientnetv2l
efficientnetv2b3
vgg19
resnet50
vgg16
efficientnetv2b0
efficientnetv2s
vgg16
convnexttiny
resnet101v2
resnet152
efficientnetb7
efficientnetb0
xception
resnet50v2
resnet50v2
densenet201
densenet169
vgg19
efficientnetv2s
convnextxlarge
efficientnetb5
nasnetlarge
efficientnetv2b0
efficientnetb7
efficientnetb3
densenet169
resnet101v2
efficientnetb6
nasnetlarge
efficientnetb1
mobilenetv2
efficientnetb5
vgg19
convnextlarge
xception
densenet201
efficientnetb1
vgg19
mobilenetv2
resnet152
nasnetmobile
vgg16
resnet152v2
efficientnetv2l
nasnetmobile
efficientnetv2m
resnet50
inceptionv3
convnextsmall
densenet121
resnet101
resnet101
efficientnetv2b3
densenet121
efficientnetv2b0
nasnetmobile
nasnetmobile
xception
resnet101
efficientnetv2b2
convnextlarge
resnet101
convnextsmall
efficientnetb2
resnet152
densenet121
convnextsmallefficientnetv2b0
nasnetlarge
inceptionv3
efficientnetb5
mobilenet
xception
efficientnetb0
resnet50
resnet152
efficientnetb6
efficientnetb3
mobilenet
inceptionresnetv2
inceptionresnetv2
nasnetlarge
xception
convnextxlarge
efficientnetv2b1
mobilenet
resnet101
resnet50v2
convnextbase
inceptionresnetv2
efficientnetv2b1
convnexttiny
inceptionv3
efficientnetv2b2
inceptionv3
mobilenetv2
resnet152v2
resnet152v2
efficientnetv2b3
xception
mobilenet
convnextxlarge
efficientnetb4
efficientnetb0
densenet121
efficientnetv2sconvnextxlarge
resnet101
resnet50
resnet50
inceptionresnetv2
efficientnetv2b1
densenet169
vgg16
vgg16
convnexttiny
efficientnetb0
resnet152
efficientnetb4
vgg19
resnet50v2
resnet50
resnet101v2
vgg19
resnet101v2
efficientnetv2m
efficientnetb2
efficientnetb2
vgg19
densenet169
efficientnetb1
resnet50v2
convnextsmall
inceptionv3
mobilenet
resnet152
resnet101v2
efficientnetb3
xception
resnet101
mobilenet
convnextlarge
vgg16
vgg16
efficientnetb4
resnet101
convnexttiny
efficientnetb5
efficientnetb6
xception
efficientnetb1
vgg16
convnextbase
densenet201
resnet50
mobilenetv2
vgg19
resnet50
efficientnetv2b2
resnet152v2
efficientnetb4
resnet152
efficientnetv2l
efficientnetb7
efficientnetb3
efficientnetb6

Applied ML with KerasCV

Image Classification

Object Detection



Semantic Segmentation

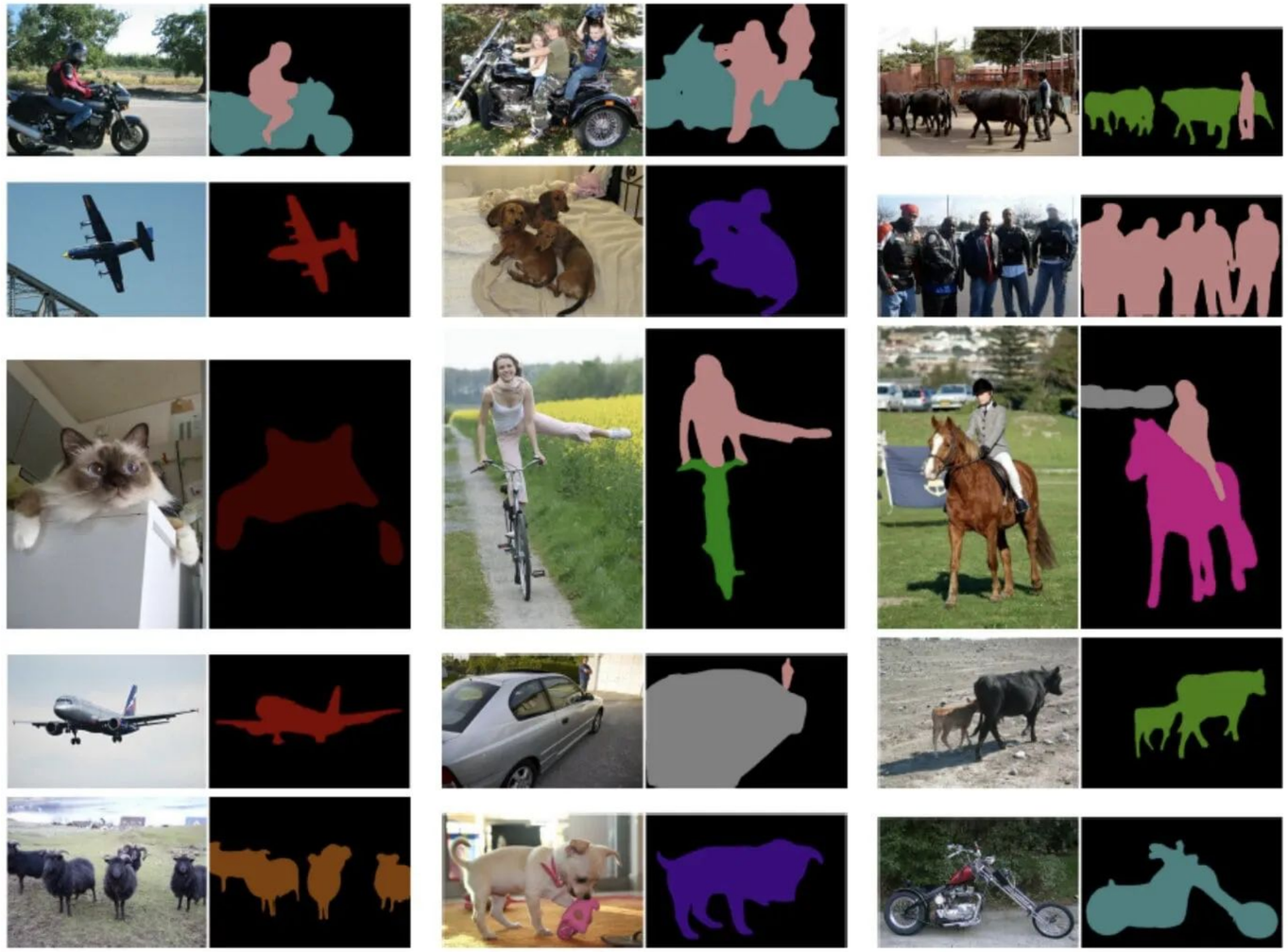
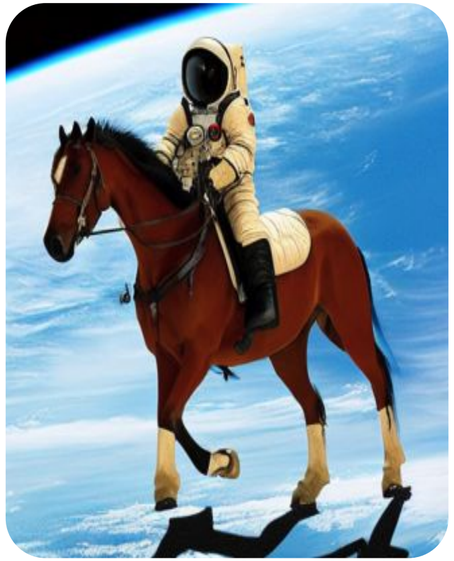
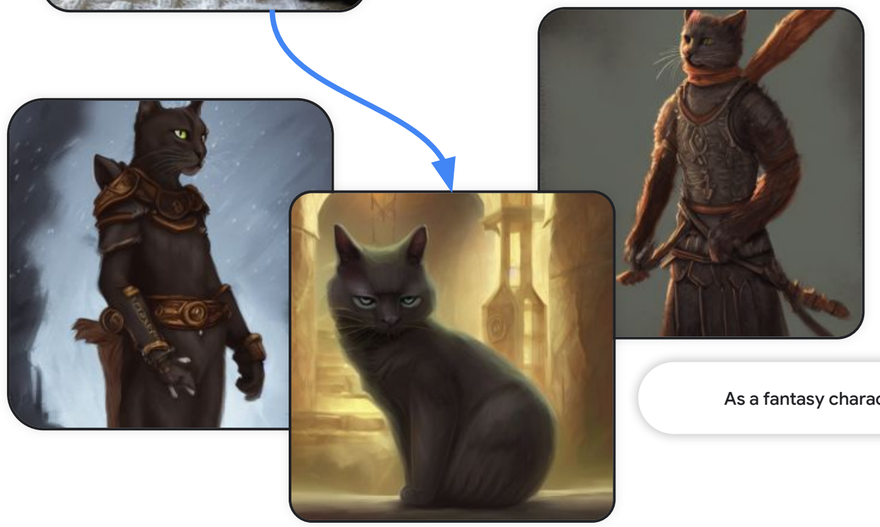


Image Generation



Textual inversion



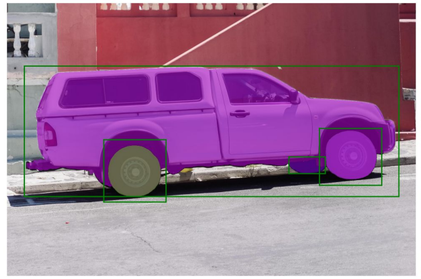
As a fantasy character

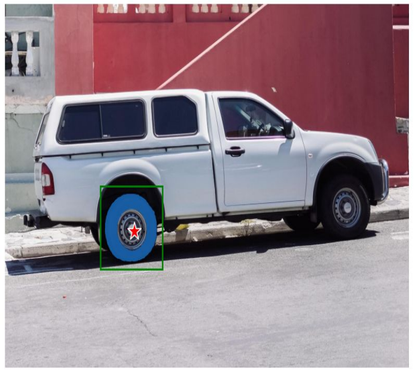
Prompt to prompt

cat

A photo of a ~~dog~~
with sunglasses

Text prompted Segment Anything




densenet169_imagenet
yolo_v8_xl_backbone_coco
densenet201_imagenet
efficientnetv2_s_imagenet
resnet50_v2_imagenet_classifier
yolo_v8_m_backbone_coco
efficientnetv2_s_imagenet
yolo_v8_xs_backbone_coco
yolo_v8_s_backbone_coco
yolo_v8_xs_backbone_coco
resnet50_v2_imagenet_classifier
efficientnetv2_b2_imagenet
yolo_v8_xl_backbone_coco
efficientnetv2_b2_imagenet
efficientnetv2_b1_imagenet_classifier
retinanet_resnet50_pascalvoc
efficientnetv2_b2_imagenet
mit_b0_imagenet
efficientnetv2_b0_imagenet
resnet50_imagenet
resnet50_v2_imagenet
mobilenet_v3_large_imagenet
yolo_v8_l_backbone_coco
efficientnetv2_b0_imagenet_classifier
mobilenet_v3_large_imagenet_classifier
yolo_v8_s_backbone_coco
resnet50_imagenet
mobilenet_v3_small_imagenet
mit_b0_imagenet
efficientnetv2_b1_imagenet
efficientnetv2_b2_imagenet_classifier
densenet201_imagenet
efficientnetv2_b0_imagenet_classifier
densenet121_imagenet
yolo_v8_xs_backbone_coco
efficientnetv2_b1_imagenet_classifier
efficientnetv2_b0_imagenet_classifier
mobilenet_v3_large_imagenet
yolo_v8_m_pascalvoc
mobilenet_v3_large_imagenet_classifier
efficientnetv2_b0_imagenet
efficientnetv2_b1_imagenet_classifier
yolo_v8_m_backbone_coco
efficientnetv2_s_imagenet_classifier
densenet121_imagenet
mobilenet_v3_large_imagenet_classifier
efficientnetv2_s_imagenet_classifier
efficientnetv2_s_imagenet_classifier
mit_b0_imagenet
densenet169_imagenet
efficientnetv2_b2_imagenet_classifier
resnet50_v2_imagenet_classifier
resnet50_v2_imagenet
retinanet_resnet50_pascalvoc
yolo_v8_xl_backbone_coco
efficientnetv2_b0_imagenet
yolo_v8_m_pascalvoc
efficientnetv2_b1_imagenet
yolo_v8_m_backbone_coco
yolo_v8_l_backbone_coco
yolo_v8_m_pascalvoc
efficientnetv2_s_imagenet
retinanet_resnet50_pascalvoc
resnet50_v2_imagenet
resnet50_imagenet
yolo_v8_s_backbone_coco
mobilenet_v3_small_imagenet
mobilenet_v3_small_imagenet
efficientnetv2_b2_imagenet_classifier
yolo_v8_l_backbone_coco
mobilenet_v3_large_imagenet
efficientnetv2_b1_imagenet
densenet121_imagenet
densenet169_imagenet
densenet201_imagenet

Applied ML with KerasNLP

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.'
```


bert_base_multi
distil_bert_base_en
gpt2_extra_large_en
albert_extra_extra_large_en_uncased
deberta_v3_base_en
gpt2_large_en
opt_6.7b_en
distil_bert_base_multi
opt_1.3b_en
opt_125m_en
gpt2_medium_en
bert_large_en_uncased
bert_large_en
bert_tiny_en_uncased
deberta_v3_base_en
bert_small_en_uncased
xlm_roberta_large_multi
bert_base_multi
deberta_v3_extra_small_en
bert_small_en_uncased
bert_large_en
f_net_base_en
bert_tiny_en_uncased
gpt2_base_en_cnn_dailymail
gpt2_extra_large_en
albert_large_en_uncased
bert_base_en
opt_2.7b_en
roberta_large_en
xlm_roberta_base_multi
gpt2_medium_en
deberta_v3_extra_small_en
deberta_v3_base_multi
opt_6.7b_en
bert_tiny_en_uncased
bert_large_en
opt_2.7b_en
albert_extra_extra_large_en_uncased
opt_6.7b_en
opt_125m_en
gpt2_base_en_cnn_dailymail
bert_base_en_uncased
albert_large_en_uncased
distil_bert_base_multi
albert_extra_large_en_uncased
deberta_v3_base_multi
bert_base_en
f_net_base_en
bert_base_en_uncased
deberta_v3_large_en
bert_large_en_uncased
gpt2_medium_en
albert_extra_extra_large_en_uncased
f_net_large_en
distil_bert_base_en
gpt2_large_en
f_net_large_en
distil_bert_base_en_uncased
deberta_v3_large_en
albert_extra_large_en_uncased
deberta_v3_base_en
albert_base_en_uncased
distil_bert_base_en_uncased
distil_bert_base_en
roberta_large_en
bert_medium_en_uncased
opt_1.3b_en
gpt2_base_en
bert_medium_en_uncased
albert_base_en_uncased
bert_base_zh
opt_125m_en
bert_base_multi
roberta_base_en
opt_2.7b_en
roberta_base_en
distil_bert_base_multi
xlm_roberta_base_multi
gpt2_extra_large_en
xlm_roberta_base_multi
bert_small_en_uncased
deberta_v3_extra_small_en
bert_base_en
gpt2_large_en
distil_bert_base_en_uncased
deberta_v3_small_en
albert_large_en_uncased
xlm_roberta_large_multi
deberta_v3_small_en
opt_1.3b_en
gpt2_base_en
albert_extra_large_en_uncased
gpt2_base_en_cnn_dailymail
bert_base_zh
bert_medium_en_uncased
bert_base_en_uncased
roberta_large_en
roberta_base_en
bert_base_zh
f_net_base_en
deberta_v3_base_multi
deberta_v3_small_en
f_net_large_en
bert_large_en_uncased
roberta_base_en
gpt2_base_en
deberta_v3_large_en
deberta_v3_small_en
xlm_roberta_large_multi
albert_base_en_uncased

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

Women in ML

Thank you!

Divyashree Sreepathihalli (she/her)
Software Engineer, Google
@divyashreess