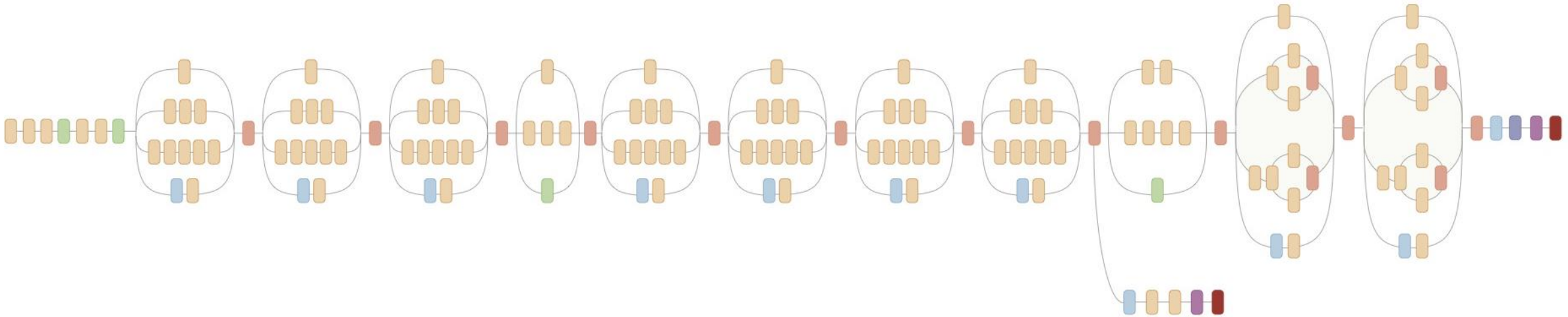
Transferência de Aprendizado em Deep Learning Utilizando Tensorflow Hub

IV MEETUP PYDATA MANAUS

FAGNER CUNHA

DEEP LEARNING

REDES NEURAIS COM MUITAS CAMADAS



Esquema da InceptionV3

Fonte: Adaptado de [1]

- Novas técnicas de treinamento
- Poder computacional
- **Muitos** dados

BASES DE DADOS

1,2M IMAGENS

330K IMAGENS
1,5M INSTÂNCIAS

2500M PALAVRAS

BASES DE DADOS

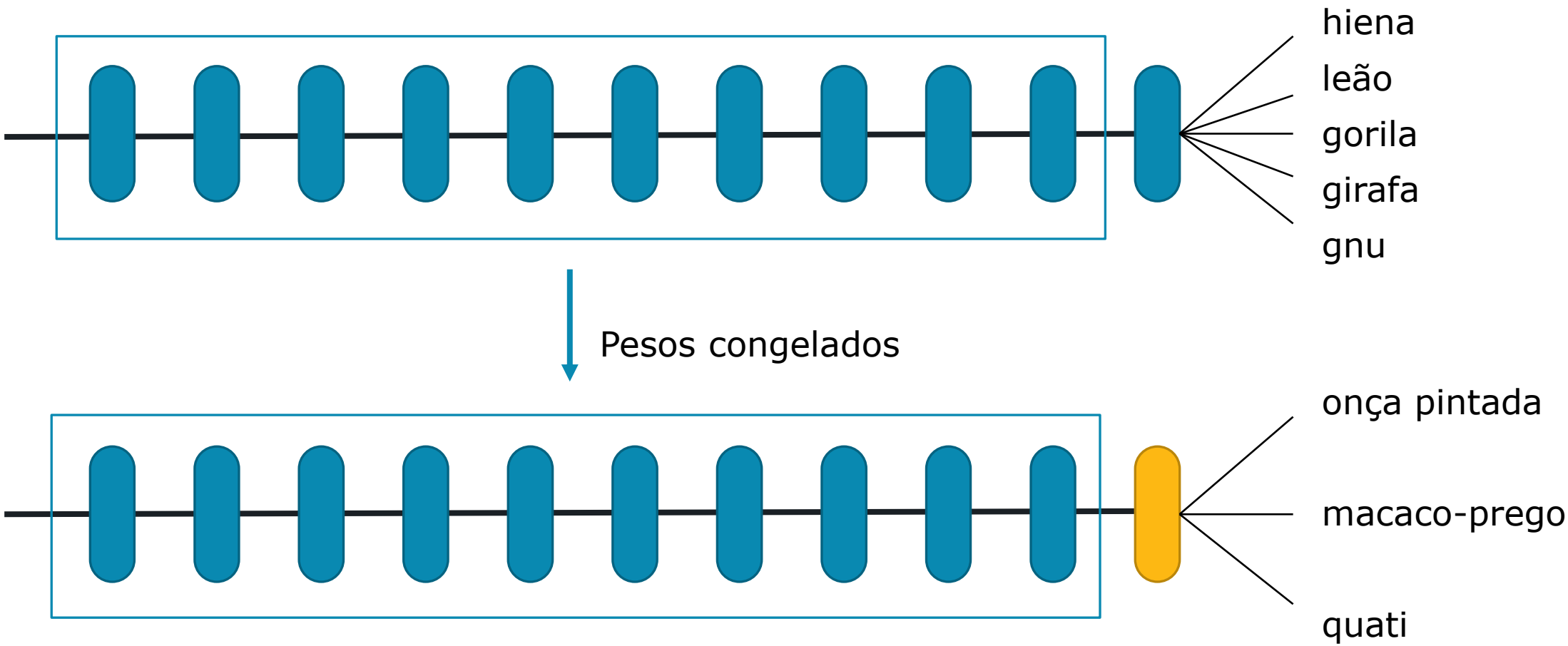
1,2M IMAGENS

330K IMAGENS
1,5M INSTÂNCIAS

2500M PALAVRAS

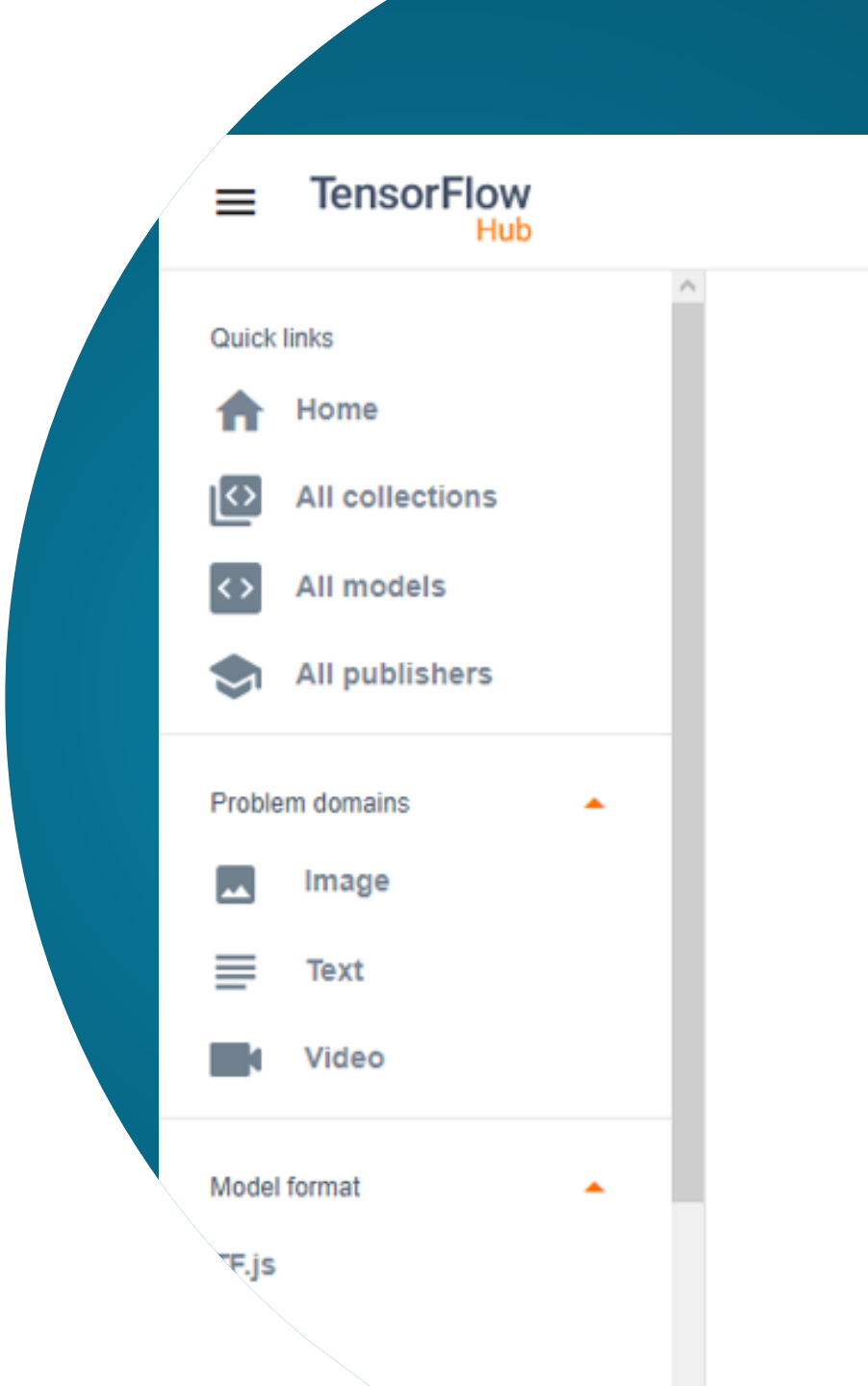
Minha base é **pequena!**

TRANSFERÊNCIA DE APRENDIZADO

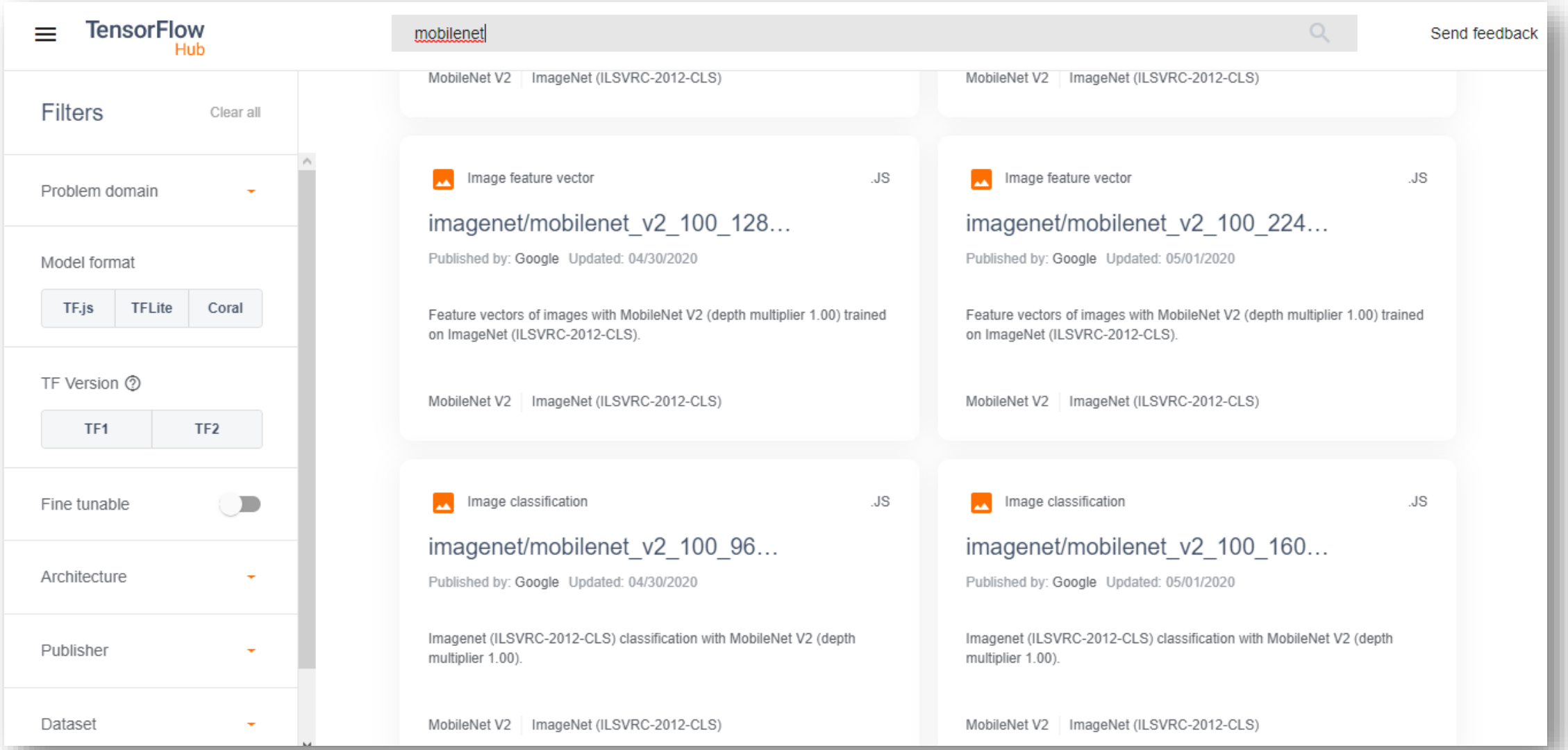


TENSORFLOW HUB

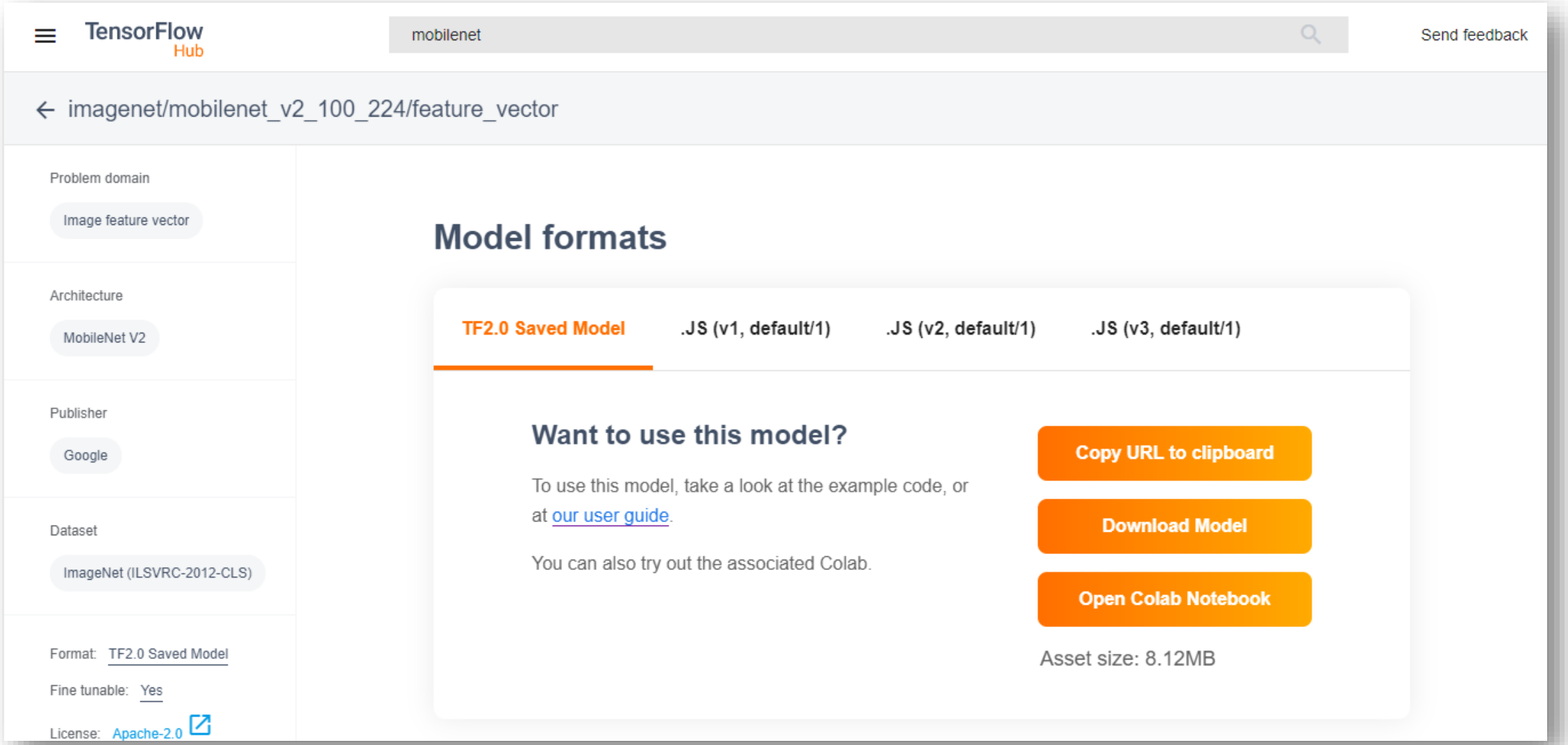
- Facilita o reuso de módulos pré-treinados
- Mais de 500 módulos pra problemas nos domínios de texto, imagem, vídeo e áudio
- tfhub.dev



TFHUB.DEV


TensorFlow Hub
mobilenet
Send feedback
Filters
Clear all
Problem domain
Model format
TF.js
TFLite
Coral
TF Version
TF1
TF2
Fine tunable
Architecture
Publisher
Dataset
MobileNet V2 ImageNet (ILSVRC-2012-CLS)
MobileNet V2 ImageNet (ILSVRC-2012-CLS)
Image feature vector .JS
imagenet/mobilenet_v2_100_128…
Published by: Google Updated: 04/30/2020
Feature vectors of images with MobileNet V2 (depth multiplier 1.00) trained on ImageNet (ILSVRC-2012-CLS).
MobileNet V2 ImageNet (ILSVRC-2012-CLS)
Image feature vector .JS
imagenet/mobilenet_v2_100_224…
Published by: Google Updated: 05/01/2020
Feature vectors of images with MobileNet V2 (depth multiplier 1.00) trained on ImageNet (ILSVRC-2012-CLS).
MobileNet V2 ImageNet (ILSVRC-2012-CLS)
Image classification .JS
imagenet/mobilenet_v2_100_96…
Published by: Google Updated: 04/30/2020
Imagenet (ILSVRC-2012-CLS) classification with MobileNet V2 (depth multiplier 1.00).
MobileNet V2 ImageNet (ILSVRC-2012-CLS)
Image classification .JS
imagenet/mobilenet_v2_100_160…
Published by: Google Updated: 05/01/2020
Imagenet (ILSVRC-2012-CLS) classification with MobileNet V2 (depth multiplier 1.00).
MobileNet V2 ImageNet (ILSVRC-2012-CLS)

TFHUB.DEV


TensorFlow Hub
mobilenet
Send feedback
imagenet/mobilenet_v2_100_224/feature_vector
Problem domain
Image feature vector
Architecture
MobileNet V2
Publisher
Google
Dataset
ImageNet (ILSVRC-2012-CLS)
Format: TF2.0 Saved Model
Fine tunable: Yes
License: Apache-2.0
Model formats
TF2.0 Saved Model
.JS (v1, default/1)
.JS (v2, default/1)
.JS (v3, default/1)
Want to use this model?
To use this model, take a look at the example code, or at our user guide.
You can also try out the associated Colab.
Copy URL to clipboard
Download Model
Open Colab Notebook
Asset size: 8.12MB

USANDO O TFHUB

In [5]:
```
import tensorflow_hub as hub

from tensorflow.keras.layers import Dense
from tensorflow.keras import Sequential
```

In [6]:
```
MODEL_URL = 'https://tfhub.dev/google/imagenet/mobilenet_v2_100_224/feature_vector/4'
feature_extractor = hub.KerasLayer(MODEL_URL,
                                   input_shape=(224, 224, 3),
                                   trainable=False)
```

In [7]:
```
model = Sequential([
  feature_extractor,
  Dense(NUM_CLASSES)
])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
keras_layer (KerasLayer)     (None, 1280)              2257984
_________________________________________________________________
dense (Dense)                (None, 3)                 3843
=================================================================
Total params: 2,261,827
Trainable params: 3,843
Non-trainable params: 2,257,984
_________________________________________________________________
```

TENSORFLOW DATASETS

- Mais de 170 datasets (áudio, imagem, vídeo, texto)
- https://www.tensorflow.org/datasets/catalog/
- Fornecidos como **tf.data.Datasets**
- Dataset de classificação de imagens: rock_paper_scissors [2]

USANDO O TENSORFLOW DATASETS

In [8]:
```
import tensorflow_datasets as tfds
```

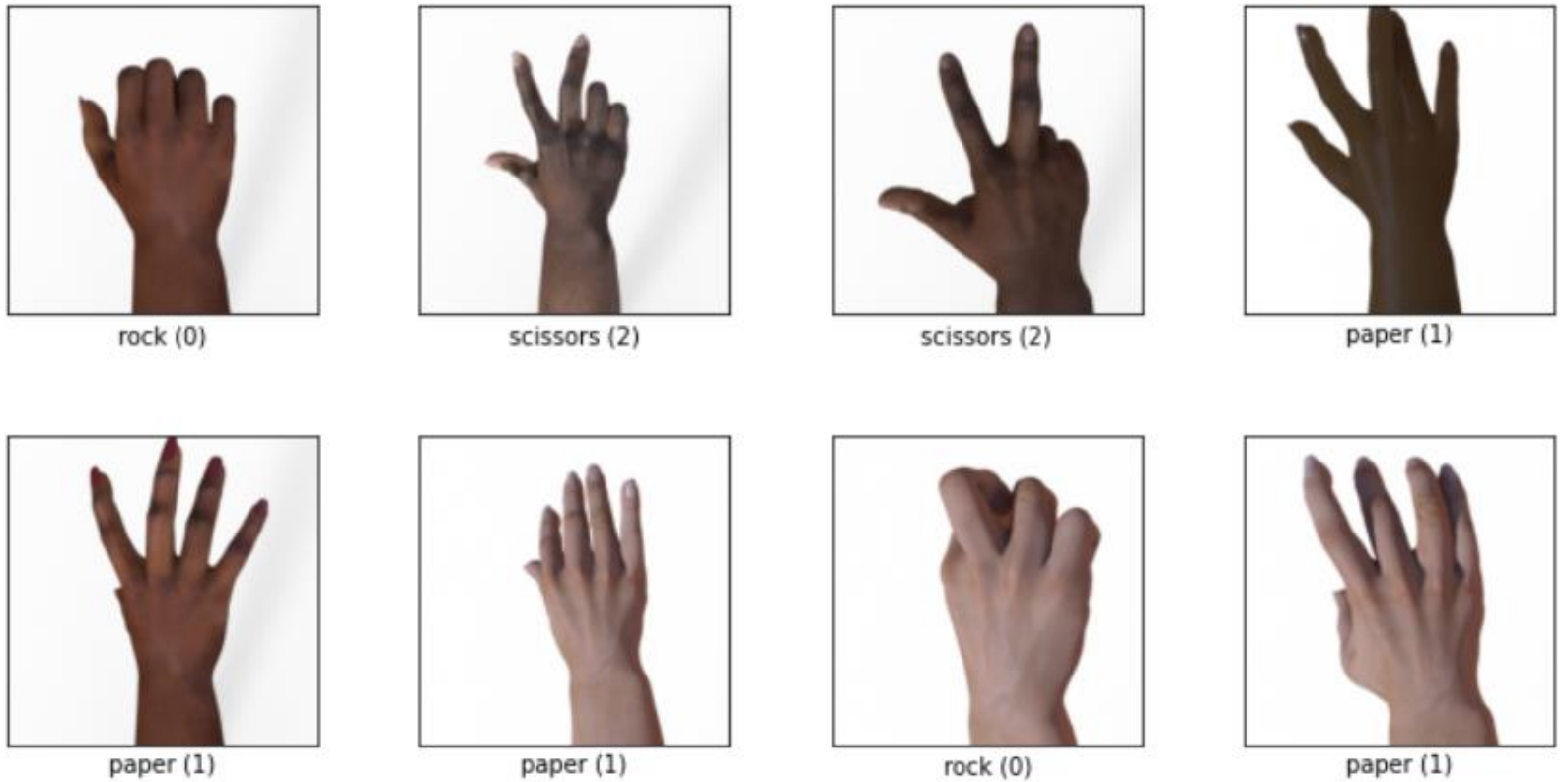
In [9]:
```
(jokempo_train, jokempo_test), info = tfds.load('rock_paper_scissors',
                                                split=['train', 'test'],
                                                shuffle_files=True,
                                                as_supervised=True,
                                                with_info=True)
```

In [10]:
```
info
```

Out[10]:
```
tfds.core.DatasetInfo(
    name='rock_paper_scissors',
    version=3.0.0,
    description='Images of hands playing rock, paper, scissor game.',
    homepage='http://laurencemoroney.com/rock-paper-scissors-dataset',
    features=FeaturesDict({
        'image': Image(shape=(300, 300, 3), dtype=tf.uint8),
        'label': ClassLabel(shape=(), dtype=tf.int64, num_classes=3),
    }),
    total_num_examples=2892,
    splits={
        'test': 372,
        'train': 2520,
    },
    supervised_keys=('image', 'label'),
```

USANDO O TENSORFLOW DATASETS

```
In [11]: show_examples(jokempo_train, info)
```



TF.DATA

- API para construir pipelines de dados
- Pipelines complexos a partir de operações simples
- Flexibilidade
- Fornecer dados em tempo hábil para GPU

TF.DATA

In [13]:
```
IMAGE_HEIGHT = 224
IMAGE_WIDTH = 224
IMAGE_PRE_CROP_WIDTH = 240
IMAGE_PRE_CROP_HEIGHT = 240
ROTATION_ANGLE = 45
AUTOTUNE = tf.data.experimental.AUTOTUNE
```

In [14]:
```
def read_decode_image(image, label):
    image = tf.cast(image, tf.float32) / 255.

    return image, label
```

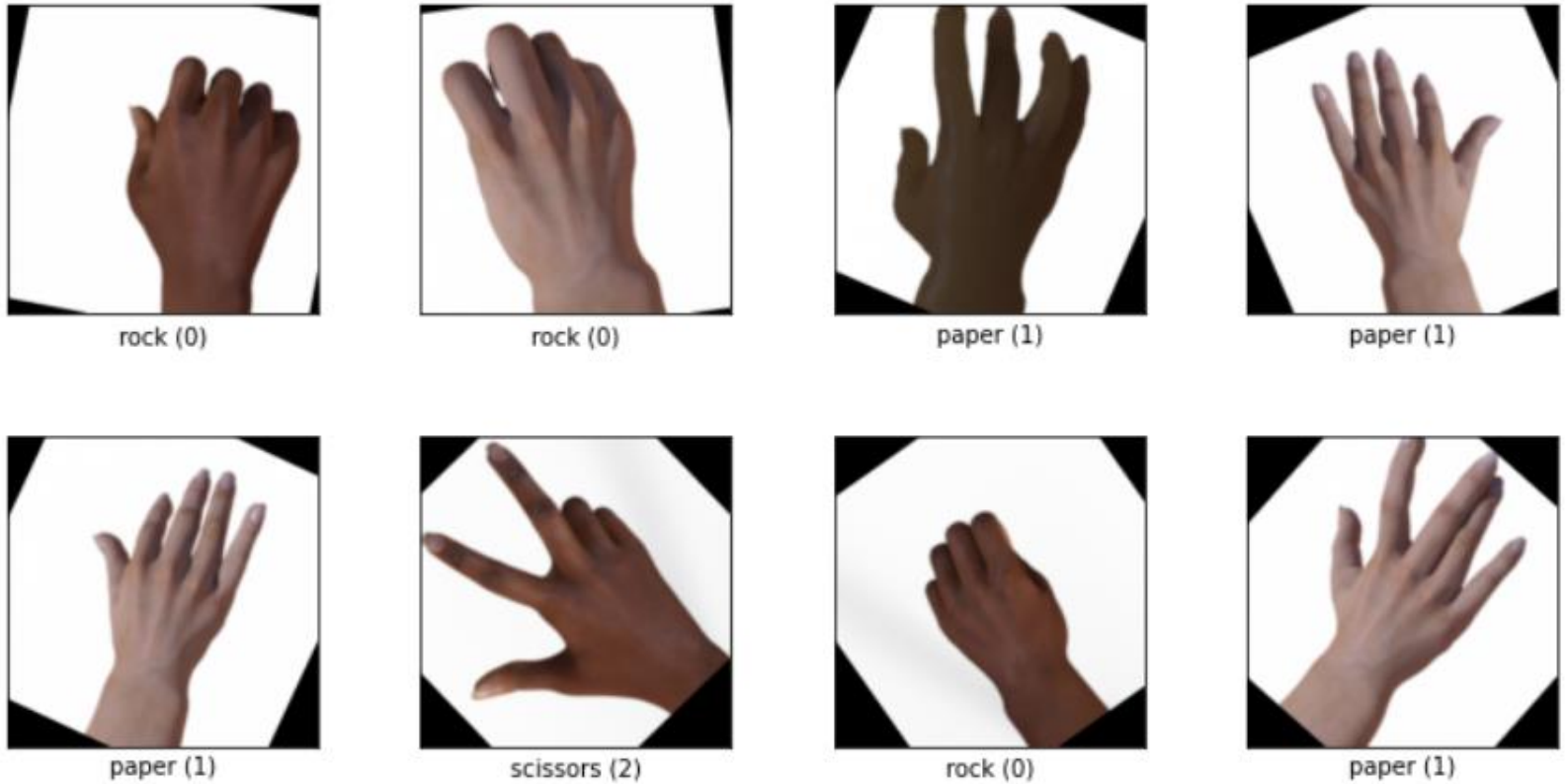
In [15]:
```
def apply_augmentation(image, label):
    rotation_theta = deg2rad(ROTATION_ANGLE)

    image = tf.image.resize(image, [IMAGE_PRE_CROP_WIDTH, IMAGE_PRE_CROP_HEIGHT])
    image = tf.image.random_flip_left_right(image)
    image = tfa.image.rotate(image,
                             tf.random.uniform(shape=[1],
                                               minval=-rotation_theta,
                                               maxval=rotation_theta),
                             interpolation='BILINEAR')
    image = tf.image.random_crop(image, size=[IMAGE_WIDTH, IMAGE_HEIGHT, 3])

    return image, label
```

TF.DATA

```
In [17]: show_examples(jokempo_train, info)
```



TF.DATA

In [18]:
```
TRAIN_LENGHT = info.splits['train'].num_examples
TEST_LENGHT = info.splits['test'].num_examples
NUM_CLASSES = info.features['label'].num_classes
```

In [19]:
```
BATCH_SIZE = 32
EPOCHS = 10
AUTOTUNE = tf.data.experimental.AUTOTUNE
```

Pipeline de treinamento:

In [20]:
```
jokempo_train = jokempo_train.map(read_decode_image, num_parallel_calls=AUTOTUNE)
jokempo_train = jokempo_train.map(apply_augmentation, num_parallel_calls=AUTOTUNE)
jokempo_train = jokempo_train.repeat(EPOCHS)
jokempo_train = jokempo_train.shuffle(buffer_size=TRAIN_LENGHT)
jokempo_train = jokempo_train.batch(BATCH_SIZE, drop_remainder=True)
jokempo_train = jokempo_train.prefetch(buffer_size=AUTOTUNE)
```

TF.DATA

Pipeline de teste:

In [22]:
```
def resize_image(image, label):
    image = tf.image.resize(image, [IMAGE_WIDTH, IMAGE_HEIGHT])

    return image, label
```

In [23]:
```
jokempo_test = jokempo_test.map(read_decode_image, num_parallel_calls=AUTOTUNE)
jokempo_test = jokempo_test.map(resize_image, num_parallel_calls=AUTOTUNE)
jokempo_test = jokempo_test.repeat(EPOCHS)
jokempo_test = jokempo_test.batch(BATCH_SIZE, drop_remainder=True)
jokempo_test = jokempo_test.prefetch(buffer_size=AUTOTUNE)
```

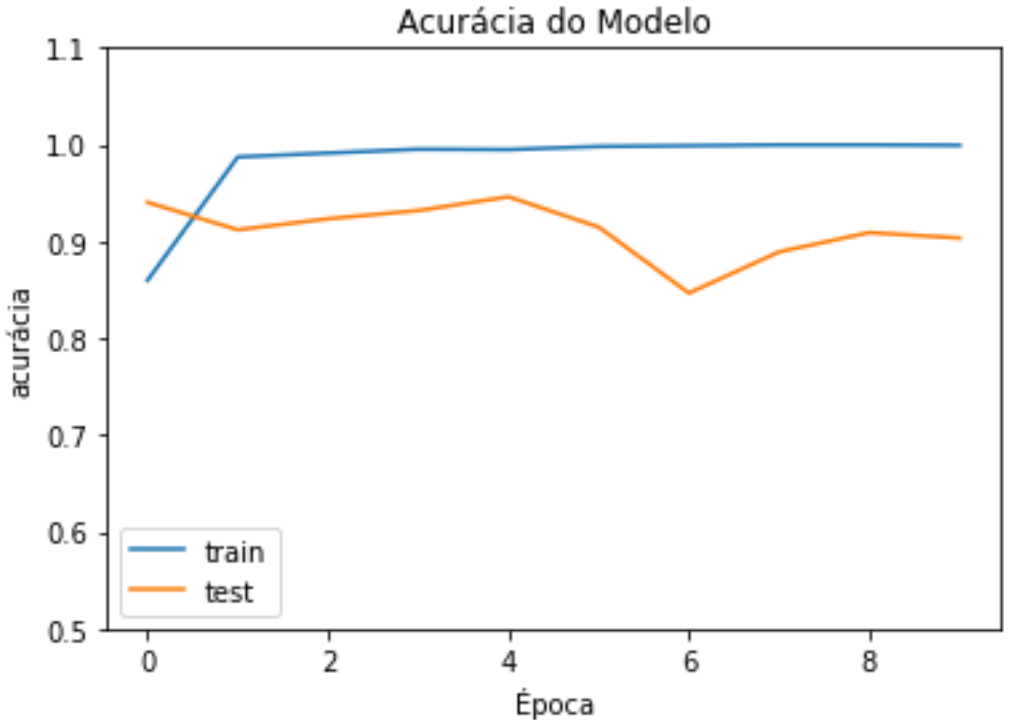
TREINANDO O MODELO

In [21]:
```
history = model.fit(jokempo_train,
                    steps_per_epoch=TRAIN_LENGHT // BATCH_SIZE,
                    epochs=EPOCHS,
                    validation_data=jokempo_test,
                    validation_steps=TEST_LENGHT // BATCH_SIZE,
                    initial_epoch=0)
```

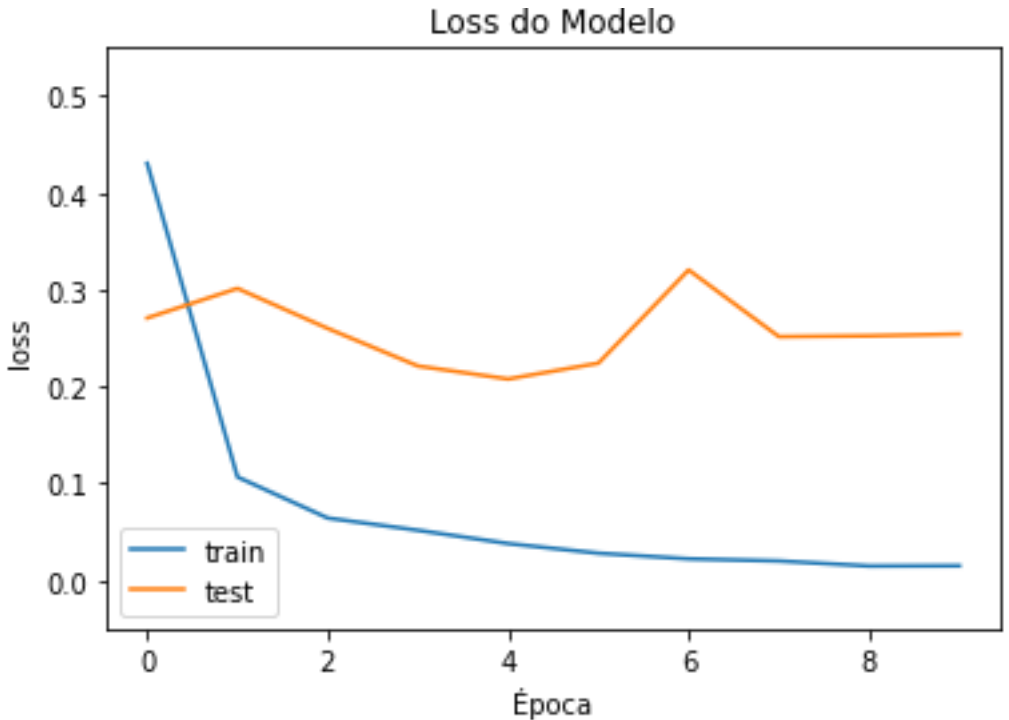
```
Train for 78 steps, validate for 11 steps
Epoch 1/10
78/78 [==============================] - 10s 134ms/step - loss: 0.4306 - accuracy: 0.8598 - val_loss:
0.2708 - val_accuracy: 0.9403
Epoch 2/10
78/78 [==============================] - 4s 49ms/step - loss: 0.1066 - accuracy: 0.9872 - val_loss:
0.3014 - val_accuracy: 0.9119
Epoch 3/10
78/78 [==============================] - 4s 51ms/step - loss: 0.0642 - accuracy: 0.9912 - val_loss:
0.2601 - val_accuracy: 0.9233
Epoch 4/10
78/78 [==============================] - 4s 51ms/step - loss: 0.0515 - accuracy: 0.9952 - val_loss:
0.2212 - val_accuracy: 0.9318
Epoch 5/10
```

VISUALIZANDO OS RESULTADOS

In [22]:
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.ylim([0.5, 1.1])
plt.title('Acurácia do Modelo')
plt.ylabel('acurácia')
plt.xlabel('Época')
plt.legend(['train', 'test'], loc='lower left')
plt.show()
```



In [23]:
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.ylim([-0.05, 0.55])
plt.title('Loss do Modelo')
plt.ylabel('loss')
plt.xlabel('Época')
plt.legend(['train', 'test'], loc='lower left')
plt.show()
```



SÓ MUDAR A ÚLTIMA CAMADA?

- Adicionar camadas intermediárias após o extrator de características
- Fine tuning
- Utilizar como gerador de features para outros modelos como o SVM
- Treinar modelos de deep learning é quase arte

OBRIGADO!

LinkedIn:
https://www.linkedin.com/in/cunhaf/

Twitter
@fagner_cunha

E-mail
fagner.cunha@icomp.ufam.edu.br

Github
@alcunha

REFERÊNCIAS

[1] Inception in TensorFlow. Disponível em: https://github.com/tensorflow/models/tree/master/research/inception. Acesso em: 30 de abril de 2020.

[2] Moroney, Laurence (2019). Rock, Paper, Scissors Dataset. Disponível em: http://laurencemoroney.com/rock-paper-scissors-dataset. Acesso em: 30 de abril de 2020.

[3] Transfer learning with TensorFlow Hub. Disponível em: https://www.tensorflow.org/tutorials/images/transfer_learning_with_hub. Acesso em: 30 de abril de 2020.

PERGUNTAS