In [ ]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.image import load_img
import tensorflow.lite as tflite
from io import BytesIO
from urllib import request
import requests
from PIL import Image
import numpy as np
```

Convert Keras to TF-Lite

Question 1

Now convert this model from Keras to TF-Lite format.

What's the size of the **converted** model?

- 21 Mb
- **43 Mb**
- 80 Mb
- 164 Mb

In [ ]:
```
model = keras.models.load_model('model-bees-wasps.h5')
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

with open('model-bees-wasps.tflite', 'wb') as f_out:
    f_out.write(tflite_model)
```

```
INFO:tensorflow:Assets written to: /tmp/tmpnliq_l00/assets
```

In [ ]:
```
# check file size
!ls -lh
```

```
total 129M
-rw-rw-r-- 1 marcos marcos  288 nov 27 07:16 Dockerfile
-rw-rw-r-- 1 marcos marcos  90K nov 27 07:41 homework-9.ipynb
-rw-rw-r-- 1 marcos marcos 1,8K nov 27 07:38 lambda_function.py
-rw-rw-r-- 1 marcos marcos  86M nov 25 08:31 model-bees-wasps.h5
-rw-rw-r-- 1 marcos marcos  43M nov 27 07:43 model-bees-wasps.tflite
```

Question 2

To be able to use this model, we need to know the index of the input and the index of the output.

What's the output index for this model?

- 3
- 7
- **13**
- 24

In [ ]:
```
# Instantiate a TensorFlow Lite Interpreter with the specified model file.
interpreter = tf.lite.Interpreter(model_path="model-bees-wasps.tflite")
# Allocate memory for the model's tensors.
interpreter.allocate_tensors()

# Retrieve and print details about the model's input/output tensor(s), such as shape and type.
print("Input details:\n",interpreter.get_input_details())

print("\nOutput details:\n",interpreter.get_output_details())
```

```
Input details:
 [{'name': 'serving_default_conv2d_input:0', 'index': 0, 'shape': array([  1, 150, 150,   3], dtype=int32), 'shap
e_signature': array([ -1, 150, 150,   3], dtype=int32), 'dtype': <class 'numpy.float32'>, 'quantization': (0.0,
0), 'quantization_parameters': {'scales': array([], dtype=float32), 'zero_points': array([], dtype=int32), 'quant
ized_dimension': 0}, 'sparsity_parameters': {}}]

Output details:
 [{'name': 'StatefulPartitionedCall:0', 'index': 13, 'shape': array([1, 1], dtype=int32), 'shape_signature': arra
y([-1,  1], dtype=int32), 'dtype': <class 'numpy.float32'>, 'quantization': (0.0, 0), 'quantization_parameters':
{'scales': array([], dtype=float32), 'zero_points': array([], dtype=int32), 'quantized_dimension': 0}, 'sparsity_
parameters': {}}]
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
```

Input

The shape of the input tensor is $[1, 150, 150, 3]$, which suggests that the model expects a single input of size 150×150 pixels with 3 color channels (RGB image). The `shape_signature` with a -1 in the first dimension indicates that the model can accept a batch of any size, but each image within the batch must be $150 \times 150 \times 3$.

Output

The shape of the output tensor is $[1, 1]$, which implies that the model outputs a single value. This is in accordance with a binary classification (e.g., bee or wasp). The `shape_signature` indicates that the model can produce an output batch of any size, but each output will be a single value.

The index refers to the position of a tensor within the internal list of tensors managed by the interpreter. Each tensor, whether it is an input or output tensor, is assigned an index number that the interpreter uses to identify it.

Preparing the image

You'll need some code for downloading and resizing images. You can use this code:

```
from io import BytesIO
from urllib import request

from PIL import Image

def download_image(url):
    with request.urlopen(url) as resp:
        buffer = resp.read()
    stream = BytesIO(buffer)
    img = Image.open(stream)
    return img


def prepare_image(img, target_size):
    if img.mode != 'RGB':
        img = img.convert('RGB')
    img = img.resize(target_size, Image.NEAREST)
    return img
```

For that, you'll need to have `pillow` installed:

```
pip install pillow
```

Let's download and resize this image:

https://habrastorage.org/webt/rt/d9/dh/rtd9dhsmhwrdezeldzoqgijdg8a.jpeg

Based on the previous homework, what should be the target size for the image?

In [ ]:
```
def download_image(url):
    # Open the URL, read the content and store it in a buffer (temporarily holding data in memory)
    with request.urlopen(url) as resp:
        buffer = resp.read()
    # Create a BytesIO stream from the buffer
    stream = BytesIO(buffer)
    # Open the stream as an Image object
    img = Image.open(stream)
    return img


def prepare_image(img, target_size):
    """Convert image to 'RGB' mode and resize it to the target size.

    Args:
        img: Image object to be converted and resized.
        target_size : A tuple (width, height) for the target size.

    Returns:
        The converted and resized image
    """
    if img.mode != 'RGB':
        img = img.convert('RGB')
    # Resize the image to the target size using the NEAREST filter
    img = img.resize(target_size, Image.NEAREST)
    return img
```

In [ ]:
```
url_img = 'https://habrastorage.org/webt/rt/d9/dh/rtd9dhsmhwrdezeldzoqgijdg8a.jpeg'
img = download_image(url_img)
```

```
img_resized = prepare_image(img, (150, 150))
display(img_resized)
```

Question 3

Now we need to turn the image into numpy array and pre-process it.

> Tip: Check the previous homework. What was the pre-processing we did there?

After the pre-processing, what's the value in the first pixel, the R channel?

- 0.3450980
- 0.5450980
- 0.7450980
- **0.9450980**

In [ ]:
```
# scales the pixel values to the range [0, 1]
img_array = np.array(img_resized)/255.0
print('\nValue first pixel from R chanel:\n', img_array[0,0,0])
```

```
Value first pixel from R chanel:
 0.9450980392156862
```

Question 4

Now let's apply this model to this image. What's the output of the model?

- 0.258
- 0.458
- **0.658**
- 0.858

In [ ]:
```
model = keras.models.load_model('model-bees-wasps.h5')
predictions = model.predict(img_array[None,:,:,:])

print('\nPrediction:\n', predictions)
```

```
1/1 [==============================] - 0s 118ms/step

Prediction:
 [[0.6592143]]
1/1 [==============================] - 0s 118ms/step

Prediction:
[[0.6592143]]
```

Prepare the lambda code

Now you need to copy all the code into a separate python file. You will need to use this file for the next two questions.

Tip: you can test this file locally with `ipython` or Jupyter Notebook by importing the file and invoking the function from this file.

Docker

For the next two questions, we'll use a Docker image that we already prepared. This is the Dockerfile that we used for creating the image:

```
FROM public.ecr.aws/lambda/python:3.10
COPY bees-wasps-v2.tflite .
```

And pushed it to `agrigorev/zoomcamp-bees-wasps:v2`.

A few notes:

- The image already contains a model and it's not the same model as the one we used for questions 1-4.

- The version of Python is 3.10, so you need to use the right wheel for TF-Lite. For Tensorflow 2.14.0, it's https://github.com/alexeygrigorev/tflite-aws-lambda/raw/main/tflite/tflite_runtime-2.14.0-cp310-cp310-linux_x86_64.whl

Question 5

Download the base image `agrigorev/zoomcamp-bees-wasps:v2`. You can easily make it by using docker pull command.

So what's the size of this base image?

- 162 Mb
- 362 Mb
- **662 Mb**
- 962 Mb

`sudo docker images`

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
agrigorev/zoomcamp-bees-wasps	v2	b9f6c13de368	8 days ago	**662MB**
marcosbenicio/diabetes_prediction	latest	ef8921c2e46d	2 weeks ago	474MB
diabetes_prediction	latest	ef8921c2e46d	2 weeks ago	474MB
homework5	latest	fd36a47c9cc5	5 weeks ago	431MB
python	3.10-slim	a9e021b7cfa1	5 weeks ago	128MB
svizor/zoomcamp-model	3.10.12-slim	08266c8f0c4b	6 weeks ago	147MB

Question 6

Now let's extend this docker image, install all the required libraries and add the code for lambda.

You don't need to include the model in the image. It's already included. The name of the file with the model is `bees-wasps-v2.tflite` and it's in the current workdir in the image (see the Dockerfile above for the reference).

Now run the container locally.

Score this image: https://habrastorage.org/webt/rt/d9/dh/rtd9dhsmhwrdezeldzoqgijdg8a.jpeg

What's the output from the model?

- 0.2453
- **0.4453**
- 0.6453
- 0.8453

Build docker:

```
docker build -t agrigorev/zoomcamp-bees-wasps:v2 .
```

Run docker:

```
docker run -it --rm -p 8080:8080 agrigorev/zoomcamp-bees-wasps:v2
```

In [ ]:
```
import requests

url = 'http://localhost:8080/2015-03-31/functions/function/invocations'
data = {'url': 'https://habrastorage.org/webt/rt/d9/dh/rtd9dhsmhwrdezeldzoqgijdg8a.jpeg'}

result = requests.post(url, json=data).json()
print(result)
```

```
{'prediction': 0.4453350603580475}
```