Water Bottle Classification

Import all the Dependencies

```
import tensorflow as tf
from tensorflow.keras import models, layers
import matplotlib.pyplot as plt
from IPython.display import HTML
```

Set all the Constants

```
BATCH_SIZE = 16
IMAGE_SIZE = 256
CHANNELS=3
EPOCHS=100
```

Import data into tensorflow dataset object

```
ls
```

```
drive/  flagged/  sample_data/
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
dataset = tf.keras.preprocessing.image_dataset_from_directory("/content/drive/MyDrive/Water bottle dataset",
    seed=123,
    shuffle=True,#
    image_size=(IMAGE_SIZE,IMAGE_SIZE),
    batch_size=BATCH_SIZE
)
```

```
Found 81 files belonging to 3 classes.
```

```
class_names = dataset.class_names
class_names
```

```
['Full_water_level_', 'Half_water_level', 'overflowing']
```

```
for image_batch, labels_batch in dataset.take(1):
    print(image_batch.shape)
    print(labels_batch.numpy())
```

```
(16, 256, 256, 3)
[1 1 1 1 0 0 2 1 1 2 0 0 0 1 1 1]
```

As you can see above, each element in the dataset is a tuple. First element is a batch of 32 elements of images. Second element is a batch of 32 elements of class labels

Visualize some of the images from our dataset

```
plt.figure(figsize=(10, 10))
for image_batch, labels_batch in dataset.take(1):
    for i in range(12):
        ax = plt.subplot(3, 4, i + 1)
        plt.imshow(image_batch[i].numpy().astype("uint8"))
        plt.title(class_names[labels_batch[i]])
        plt.axis("off")
```

Function to Split Dataset

Dataset should be bifurcated into 3 subsets, namely:

1. Training: Dataset to be used while training
2. Validation: Dataset to be tested against while training
3. Test: Dataset to be tested against after we trained a model

```
len(dataset)
```

```
6
```

```
train_size = 0.7
len(dataset)*train_size
```

```
4.199999999999999
```

```
train_ds = dataset.take(4)
len(train_ds)
```

```
4
```

```
test_ds = dataset.skip(4)
len(test_ds)
```

```
2
```

```
val_size=0.2
len(dataset)*val_size
```

```
1.2000000000000002
```

```
val_ds = test_ds.take(1)
len(val_ds)
```

```
1
```

```
test_ds = test_ds.skip(1)

len(test_ds)
```

```
1
```

```
def get_dataset_partitions_tf(ds, train_split=0.6, val_split=0.2, test_split=0.2, shuffle=True, shuffle_size=10000):
    assert (train_split + test_split + val_split) == 1

    ds_size = len(ds)

    if shuffle:
        ds = ds.shuffle(shuffle_size, seed=12)

    train_size = int(train_split * ds_size)
    val_size = int(val_split * ds_size)

    train_ds = ds.take(train_size)
    val_ds = ds.skip(train_size).take(val_size)
    test_ds = ds.skip(train_size).skip(val_size)

    return train_ds, val_ds, test_ds
```

```
train_ds, val_ds, test_ds = get_dataset_partitions_tf(dataset)
```

```
len(train_ds)
```

```
     3
```

```
len(val_ds)
```

```
     1
```

```
len(test_ds)
```

```
     2
```

Cache, Shuffle, and Prefetch the Dataset

```
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
test_ds = test_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
```

Building the Model

Creating a Layer for Resizing and Normalization

Before we feed our images to network, we should be resizing it to the desired size. Moreover, to improve model performance, we should normalize the image pixel value (keeping them in range 0 and 1 by dividing by 256). This should happen while training as well as inference. Hence we can add that as a layer in our Sequential Model.

You might be thinking why do we need to resize (256,256) image to again (256,256). You are right we don't need to but this will be useful when we are done with the training and start using the model for predictions. At that time somone can supply an image that is not (256,256) and this layer will resize it

```
resize_and_rescale = tf.keras.Sequential([
  layers.experimental.preprocessing.Resizing(IMAGE_SIZE, IMAGE_SIZE),
  layers.experimental.preprocessing.Rescaling(1./255),
])
```

Data Augmentation

Data Augmentation is needed when we have less data, this boosts the accuracy of our model by augmenting the data.

```
data_augmentation = tf.keras.Sequential([
  layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
  layers.experimental.preprocessing.RandomRotation(0.2),
])
```

Applying Data Augmentation to Train Dataset

```
train_ds = train_ds.map(
    lambda x, y: (data_augmentation(x, training=True), y)
).prefetch(buffer_size=tf.data.AUTOTUNE)
```

```
from sklearn.utils import class_weight
import numpy as np

class_weight = {0: 0.22,
                1: 0.62,2:0.16}
```

Model Architecture

We use a CNN coupled with a Softmax activation in the output layer. We also add the initial layers for resizing, normalization and Data Augmentation.

```
input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
n_classes = 3
```

```
model = models.Sequential([
    resize_and_rescale,
    layers.Conv2D(32, kernel_size = (3,3), activation='relu', input_shape=input_shape),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(n_classes, activation='softmax'),
])

model.build(input_shape=input_shape)
```

```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 sequential (Sequential)     (16, 256, 256, 3)         0         
                                                                 
 conv2d (Conv2D)             (16, 254, 254, 32)        896       
                                                                 
 max_pooling2d (MaxPooling2D  (16, 127, 127, 32)       0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (16, 125, 125, 64)        18496     
                                                                 
 max_pooling2d_1 (MaxPooling  (16, 62, 62, 64)         0         
 2D)                                                             
                                                                 
 conv2d_2 (Conv2D)           (16, 60, 60, 64)          36928     
                                                                 
 max_pooling2d_2 (MaxPooling  (16, 30, 30, 64)         0         
 2D)                                                             
                                                                 
 conv2d_3 (Conv2D)           (16, 28, 28, 64)          36928     
                                                                 
 max_pooling2d_3 (MaxPooling  (16, 14, 14, 64)         0         
 2D)                                                             
                                                                 
 conv2d_4 (Conv2D)           (16, 12, 12, 64)          36928     
                                                                 
 max_pooling2d_4 (MaxPooling  (16, 6, 6, 64)           0         
 2D)                                                             
                                                                 
 conv2d_5 (Conv2D)           (16, 4, 4, 64)            36928     
                                                                 
 max_pooling2d_5 (MaxPooling  (16, 2, 2, 64)           0         
 2D)                                                             
                                                                 
 flatten (Flatten)           (16, 256)                 0         
                                                                 
 dense (Dense)               (16, 64)                  16448     
                                                                 
 dense_1 (Dense)             (16, 3)                   195       
                                                                 
=================================================================
Total params: 183,747
Trainable params: 183,747
Non-trainable params: 0
_________________________________________________________________
```

Compiling the Model

We use `adam` Optimizer, `SparseCategoricalCrossentropy` for losses, `accuracy` as a metric

```
model.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)
```

```
history = model.fit(
    train_ds,
    batch_size=BATCH_SIZE,
    validation_data=val_ds,
    verbose=1,
    epochs=100,
    class_weight=class_weight
)
```

```
Epoch 72/100
3/3 [==============================] - 0s 173ms/step - loss: 0.2283 - accuracy: 0.6364 - val_loss: 1.3197 - val_accuracy: 0.5625
Epoch 73/100
3/3 [==============================] - 0s 172ms/step - loss: 0.2266 - accuracy: 0.6364 - val_loss: 1.2824 - val_accuracy: 0.5625
Epoch 74/100
3/3 [==============================] - 0s 95ms/step - loss: 0.2246 - accuracy: 0.6364 - val_loss: 1.3093 - val_accuracy: 0.5625
Epoch 75/100
3/3 [==============================] - 0s 94ms/step - loss: 0.2275 - accuracy: 0.6364 - val_loss: 1.3239 - val_accuracy: 0.5625
Epoch 76/100
3/3 [==============================] - 0s 98ms/step - loss: 0.2296 - accuracy: 0.6364 - val_loss: 1.3165 - val_accuracy: 0.5625
Epoch 77/100
3/3 [==============================] - 0s 95ms/step - loss: 0.2271 - accuracy: 0.6364 - val_loss: 1.3210 - val_accuracy: 0.5625
Epoch 78/100
3/3 [==============================] - 0s 99ms/step - loss: 0.2263 - accuracy: 0.6364 - val_loss: 1.3386 - val_accuracy: 0.5625
Epoch 79/100
3/3 [==============================] - 0s 177ms/step - loss: 0.2248 - accuracy: 0.6364 - val_loss: 1.3297 - val_accuracy: 0.5625
Epoch 80/100
3/3 [==============================] - 0s 176ms/step - loss: 0.2238 - accuracy: 0.6364 - val_loss: 1.3396 - val_accuracy: 0.5625
Epoch 81/100
3/3 [==============================] - 0s 91ms/step - loss: 0.2271 - accuracy: 0.6364 - val_loss: 1.3192 - val_accuracy: 0.5625
Epoch 82/100
3/3 [==============================] - 0s 92ms/step - loss: 0.2229 - accuracy: 0.6364 - val_loss: 1.2673 - val_accuracy: 0.5625
Epoch 83/100
3/3 [==============================] - 0s 95ms/step - loss: 0.2263 - accuracy: 0.6364 - val_loss: 1.3391 - val_accuracy: 0.5625
Epoch 84/100
3/3 [==============================] - 0s 98ms/step - loss: 0.2318 - accuracy: 0.6364 - val_loss: 1.4557 - val_accuracy: 0.5625
Epoch 85/100
3/3 [==============================] - 0s 93ms/step - loss: 0.2261 - accuracy: 0.6364 - val_loss: 1.2625 - val_accuracy: 0.5625
Epoch 86/100
3/3 [==============================] - 0s 182ms/step - loss: 0.2201 - accuracy: 0.6364 - val_loss: 1.2469 - val_accuracy: 0.5625
Epoch 87/100
3/3 [==============================] - 0s 177ms/step - loss: 0.2166 - accuracy: 0.6364 - val_loss: 1.3310 - val_accuracy: 0.5625
Epoch 88/100
3/3 [==============================] - 0s 174ms/step - loss: 0.2183 - accuracy: 0.6364 - val_loss: 1.3694 - val_accuracy: 0.5625
Epoch 89/100
3/3 [==============================] - 0s 95ms/step - loss: 0.2132 - accuracy: 0.6667 - val_loss: 1.2964 - val_accuracy: 0.5625
Epoch 90/100
3/3 [==============================] - 0s 91ms/step - loss: 0.2135 - accuracy: 0.6667 - val_loss: 1.5592 - val_accuracy: 0.5625
Epoch 91/100
3/3 [==============================] - 0s 96ms/step - loss: 0.2098 - accuracy: 0.6667 - val_loss: 1.1474 - val_accuracy: 0.5625
Epoch 92/100
3/3 [==============================] - 0s 95ms/step - loss: 0.2215 - accuracy: 0.6667 - val_loss: 1.7058 - val_accuracy: 0.5625
Epoch 93/100
3/3 [==============================] - 0s 104ms/step - loss: 0.2499 - accuracy: 0.6364 - val_loss: 1.4159 - val_accuracy: 0.5625
Epoch 94/100
3/3 [==============================] - 0s 91ms/step - loss: 0.2123 - accuracy: 0.6364 - val_loss: 1.0390 - val_accuracy: 0.5625
Epoch 95/100
3/3 [==============================] - 0s 178ms/step - loss: 0.2214 - accuracy: 0.7273 - val_loss: 1.1746 - val_accuracy: 0.5625
Epoch 96/100
3/3 [==============================] - 0s 104ms/step - loss: 0.1934 - accuracy: 0.6970 - val_loss: 1.5333 - val_accuracy: 0.5625
Epoch 97/100
3/3 [==============================] - 0s 97ms/step - loss: 0.1963 - accuracy: 0.6970 - val_loss: 1.4890 - val_accuracy: 0.5625
Epoch 98/100
3/3 [==============================] - 0s 98ms/step - loss: 0.1771 - accuracy: 0.7879 - val_loss: 1.5723 - val_accuracy: 0.5625
Epoch 99/100
3/3 [==============================] - 0s 175ms/step - loss: 0.1704 - accuracy: 0.7879 - val_loss: 1.6017 - val_accuracy: 0.5625
Epoch 100/100
3/3 [==============================] - 0s 170ms/step - loss: 0.1696 - accuracy: 0.7879 - val_loss: 1.4910 - val_accuracy: 0.5625
```

```
scores = model.evaluate(test_ds)
```

```
2/2 [==============================] - 1s 24ms/step - loss: 0.7800 - accuracy: 0.6562
```

```
scores
```

```
[0.7799946069717407, 0.65625]
```

Scores is just a list containing loss and accuracy value

Plotting the Accuracy and Loss Curves

```
history
```

```
<keras.callbacks.History at 0x7f744f84f220>
```

```
history.params
```

```
{'verbose': 1, 'epochs': 100, 'steps': 3}
```

```
history.history.keys()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

loss, accuracy, val loss etc are a python list containing values of loss, accuracy etc at the end of each epoch

```
type(history.history['loss'])
```

```
list
```

```
len(history.history['loss'])
```

```
100
```

```
history.history['loss'][:5] # show loss for first 5 epochs
```

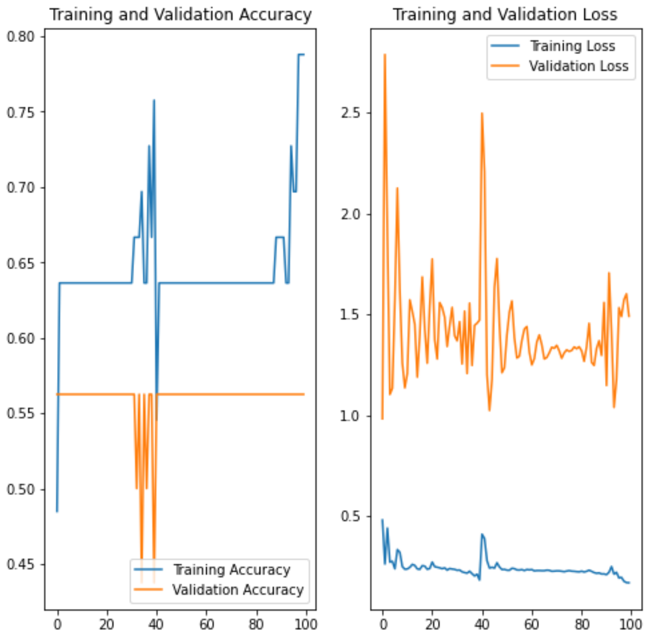
```
[0.48026713728904724,
 0.26245275139808655,
 0.44075724482536316,
 0.2717439830303192,
 0.275861531496048]
```

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']
```

```
plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(range(EPOCHS), acc, label='Training Accuracy')
plt.plot(range(EPOCHS), val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(range(EPOCHS), loss, label='Training Loss')
plt.plot(range(EPOCHS), val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



Run prediction on a sample image

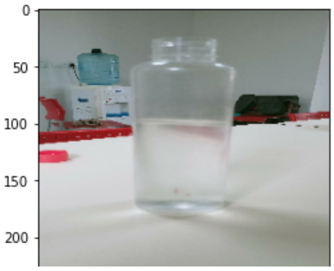
```
import numpy as np
for images_batch, labels_batch in test_ds.take(1):

    first_image = images_batch[0].numpy().astype('uint8')
    first_label = labels_batch[0].numpy()

    print("first image to predict")
    plt.imshow(first_image)
    print("actual label:",class_names[first_label])

    batch_prediction = model.predict(images_batch)
    print("predicted label:",class_names[np.argmax(batch_prediction[0])])
```

```
first image to predict
actual label: Half_water_level
1/1 [==============================] - 0s 135ms/step
predicted label: Half_water_level
```



Write a function for inference

```
def predict(model, img):
    img_array = tf.keras.preprocessing.image.img_to_array(images[i].numpy())
    img_array = tf.expand_dims(img_array, 0)

    predictions = model.predict(img_array)

    predicted_class = class_names[np.argmax(predictions[0])]
    confidence = round(100 * (np.max(predictions[0])), 2)
    return predicted_class, confidence
```

Now run inference on few sample images

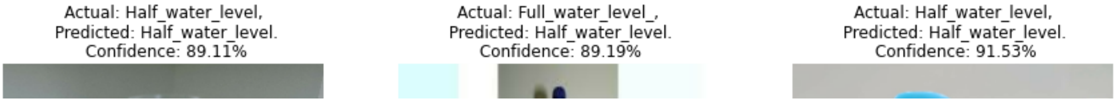
```
plt.figure(figsize=(15, 15))
for images, labels in test_ds.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))

        predicted_class, confidence = predict(model, images[i].numpy())
        actual_class = class_names[labels[i]]

        plt.title(f"Actual: {actual_class},\n Predicted: {predicted_class}.\n Confidence: {confidence}%")

        plt.axis("off")
```

```
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 21ms/step
```


Actual: Half_water_level,
Predicted: Half_water_level.
Confidence: 89.11%
Actual: Full_water_level_,
Predicted: Half_water_level.
Confidence: 89.19%
Actual: Half_water_level,
Predicted: Half_water_level.
Confidence: 91.53%

Saving the Model

We append the model to the list of models as a new version

```
# Save the model
model.save("/content/drive/MyDrive/Modified_cnn_new_model/saved_model.h5")
```

New Section

Model Deployment

```
pip install gradio
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: gradio in /usr/local/lib/python3.8/dist-packages (3.15.0)
Requirement already satisfied: markdown-it-py[linkify,plugins] in /usr/local/lib/python3.8/dist-packages (from gradio) (2.1.0)
Requirement already satisfied: uvicorn in /usr/local/lib/python3.8/dist-packages (from gradio) (0.20.0)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.8/dist-packages (from gradio) (3.8.3)
Requirement already satisfied: python-multipart in /usr/local/lib/python3.8/dist-packages (from gradio) (0.0.5)
Requirement already satisfied: websockets>=10.0 in /usr/local/lib/python3.8/dist-packages (from gradio) (10.4)
Requirement already satisfied: altair>=4.2.0 in /usr/local/lib/python3.8/dist-packages (from gradio) (4.2.0)
Requirement already satisfied: pandas in /usr/local/lib/python3.8/dist-packages (from gradio) (1.3.5)
Requirement already satisfied: ffmpy in /usr/local/lib/python3.8/dist-packages (from gradio) (0.3.0)
Requirement already satisfied: pydantic in /usr/local/lib/python3.8/dist-packages (from gradio) (1.10.2)
Requirement already satisfied: httpx in /usr/local/lib/python3.8/dist-packages (from gradio) (0.23.1)
Requirement already satisfied: pydub in /usr/local/lib/python3.8/dist-packages (from gradio) (0.25.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-packages (from gradio) (1.21.6)
Requirement already satisfied: fastapi in /usr/local/lib/python3.8/dist-packages (from gradio) (0.88.0)
Requirement already satisfied: pycryptodome in /usr/local/lib/python3.8/dist-packages (from gradio) (3.16.0)
Requirement already satisfied: fsspec in /usr/local/lib/python3.8/dist-packages (from gradio) (2022.11.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.8/dist-packages (from gradio) (3.2.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.8/dist-packages (from gradio) (2.11.3)
Requirement already satisfied: markupsafe in /usr/local/lib/python3.8/dist-packages (from gradio) (2.0.1)
Requirement already satisfied: pillow in /usr/local/lib/python3.8/dist-packages (from gradio) (7.1.2)
Requirement already satisfied: orjson in /usr/local/lib/python3.8/dist-packages (from gradio) (3.8.3)
Requirement already satisfied: requests in /usr/local/lib/python3.8/dist-packages (from gradio) (2.23.0)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.8/dist-packages (from gradio) (6.0)
Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.8/dist-packages (from altair>=4.2.0->gradio) (4.3.3)
Requirement already satisfied: toolz in /usr/local/lib/python3.8/dist-packages (from altair>=4.2.0->gradio) (0.12.0)
Requirement already satisfied: entrypoints in /usr/local/lib/python3.8/dist-packages (from altair>=4.2.0->gradio) (0.4)
Requirement already satisfied: attrs>=17.4.0 in /usr/local/lib/python3.8/dist-packages (from jsonschema>=3.0->altair>=4.2.0->gradio) (22.1.0)
Requirement already satisfied: importlib-resources>=1.4.0 in /usr/local/lib/python3.8/dist-packages (from jsonschema>=3.0->altair>=4.2.0->gradic
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in /usr/local/lib/python3.8/dist-packages (from jsonschema>=3.0->al
Requirement already satisfied: zipp>=3.1.0 in /usr/local/lib/python3.8/dist-packages (from importlib-resources>=1.4.0->jsonschema>=3.0->altair>=
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.8/dist-packages (from pandas->gradio) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.8/dist-packages (from pandas->gradio) (2022.6)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.8/dist-packages (from python-dateutil>=2.7.3->pandas->gradio) (1.15.0)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.8/dist-packages (from aiohttp->gradio) (1.3.1)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.8/dist-packages (from aiohttp->gradio) (1.3.3)
Requirement already satisfied: charset-normalizer<3.0,>=2.0 in /usr/local/lib/python3.8/dist-packages (from aiohttp->gradio) (2.1.1)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.8/dist-packages (from aiohttp->gradio) (6.0.3)
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /usr/local/lib/python3.8/dist-packages (from aiohttp->gradio) (4.0.2)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.8/dist-packages (from aiohttp->gradio) (1.8.2)
Requirement already satisfied: idna>=2.0 in /usr/local/lib/python3.8/dist-packages (from yarl<2.0,>=1.0->aiohttp->gradio) (2.10)
Requirement already satisfied: starlette==0.22.0 in /usr/local/lib/python3.8/dist-packages (from fastapi->gradio) (0.22.0)
Requirement already satisfied: anyio<5,>=3.4.0 in /usr/local/lib/python3.8/dist-packages (from starlette==0.22.0->fastapi->gradio) (3.6.2)
Requirement already satisfied: typing-extensions>=3.10.0 in /usr/local/lib/python3.8/dist-packages (from starlette==0.22.0->fastapi->gradio) (4.
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.8/dist-packages (from anyio<5,>=3.4.0->starlette==0.22.0->fastapi->gradio)
Requirement already satisfied: httpcore<0.17.0,>=0.15.0 in /usr/local/lib/python3.8/dist-packages (from httpx->gradio) (0.16.3)
Requirement already satisfied: certifi in /usr/local/lib/python3.8/dist-packages (from httpx->gradio) (2022.12.7)
Requirement already satisfied: rfc3986[idna2008]<2,>=1.3 in /usr/local/lib/python3.8/dist-packages (from httpx->gradio) (1.5.0)
Requirement already satisfied: h11<0.15,>=0.13 in /usr/local/lib/python3.8/dist-packages (from httpcore<0.17.0,>=0.15.0->httpx->gradio) (0.14.0)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.8/dist-packages (from markdown-it-py[linkify,plugins]->gradio) (0.1.2)
Requirement already satisfied: linkify-it-py~=1.0 in /usr/local/lib/python3.8/dist-packages (from markdown-it-py[linkify,plugins]->gradio) (1.0.
Requirement already satisfied: mdit-py-plugins in /usr/local/lib/python3.8/dist-packages (from markdown-it-py[linkify,plugins]->gradio) (0.3.3)
Requirement already satisfied: uc-micro-py in /usr/local/lib/python3.8/dist-packages (from linkify-it-py~=1.0->markdown-it-py[linkify,plugins]->
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.8/dist-packages (from matplotlib->gradio) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib->gradio) (1.4.4)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib->gradio) (3.0
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.8/dist-packages (from requests->gradio) (1.24.3
from tensorflow.keras.models import load_model

model = load_model("/content/drive/MyDrive/Modified_cnn_new_model/saved_model.h5")


def predict_image(img):
  img_4d=img.reshape(-1,256,256,3)
  prediction=model.predict(img_4d)[0]
  return {class_names[i]: float(prediction[i]) for i in range(3)}
```

class_names

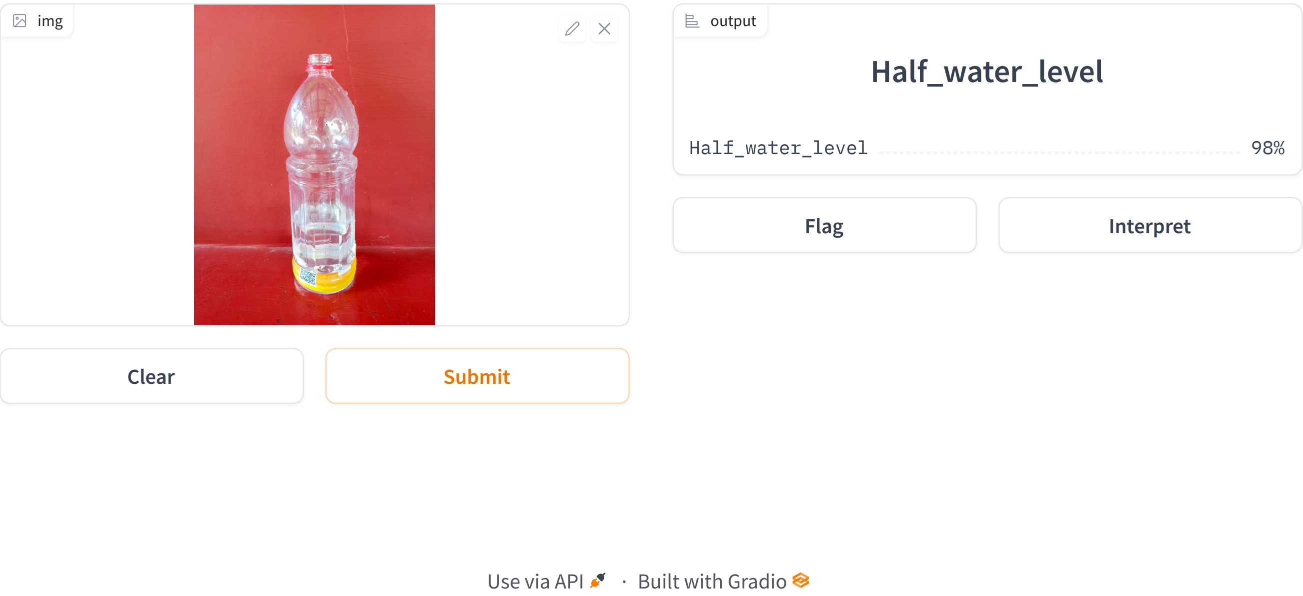
```
import gradio as gr
```

```
image = gr.inputs.Image(shape=(256,256))
label = gr.outputs.Label(num_top_classes=1)

gr.Interface(fn=predict_image, inputs=image, outputs=label,interpretation='default').launch(debug='True')
```

```
/usr/local/lib/python3.8/dist-packages/gradio/inputs.py:256: UserWarning: Usage of gradio.inputs is deprecated, and will not be supported in the f
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/gradio/deprecation.py:40: UserWarning: `optional` parameter is deprecated, and it has no effect
  warnings.warn(value)
/usr/local/lib/python3.8/dist-packages/gradio/outputs.py:196: UserWarning: Usage of gradio.outputs is deprecated, and will not be supported in the
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/gradio/deprecation.py:40: UserWarning: The 'type' parameter has been deprecated. Use the Number component i
  warnings.warn(value)
Colab notebook detected. This cell will run indefinitely so that you can see errors and logs. To turn off, set debug=False in launch().
Note: opening Chrome Inspector may crash demo inside Colab notebooks.

To create a public link, set `share=True` in `launch()`.
Running on https://localhost:7860/
```



```
1/1 [==============================] - 0s 106ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 88ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
Keyboard interruption in main thread... closing server.
```

```
image = gr.inputs.Image(shape=(256,256))
label = gr.outputs.Label(num_top_classes=1)

gr.Interface(fn=predict_image, inputs=image, outputs=label,interpretation='default').launch(debug='True')
```

```
/usr/local/lib/python3.8/dist-packages/gradio/inputs.py:256: UserWarning: Usage of gradio.inputs is deprecated, and will not be supported in the f
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/gradio/deprecation.py:40: UserWarning: `optional` parameter is deprecated, and it has no effect
  warnings.warn(value)
/usr/local/lib/python3.8/dist-packages/gradio/outputs.py:196: UserWarning: Usage of gradio.outputs is deprecated, and will not be supported in the
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/gradio/deprecation.py:40: UserWarning: The 'type' parameter has been deprecated. Use the Number component i
  warnings.warn(value)
Colab notebook detected. This cell will run indefinitely so that you can see errors and logs. To turn off, set debug=False in launch().
Note: opening Chrome Inspector may crash demo inside Colab notebooks.

To create a public link, set `share=True` in `launch()`.
Running on https://localhost:7860/
```

img

output

Half_water_level

Half_water_level 92%

Flag

Interpret

Clear

Submit

```
image = gr.inputs.Image(shape=(256,256))
label = gr.outputs.Label(num_top_classes=1)

gr.Interface(fn=predict_image, inputs=image, outputs=label,interpretation='default').launch(debug='True')
```

```
/usr/local/lib/python3.8/dist-packages/gradio/inputs.py:256: UserWarning: Usage of gradio.input
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/gradio/deprecation.py:40: UserWarning: `optional` parame
  warnings.warn(value)
/usr/local/lib/python3.8/dist-packages/gradio/outputs.py:196: UserWarning: Usage of gradio.out
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/gradio/deprecation.py:40: UserWarning: The 'type' parame
  warnings.warn(value)
Colab notebook detected. This cell will run indefinitely so that you can see errors and logs.
Note: opening Chrome Inspector may crash demo inside Colab notebooks.

To create a public link, set `share=True` in `launch()`.
Running on https://localhost:7860/
```

img

Clear

Submit

output

overflowing

overflowing 100%

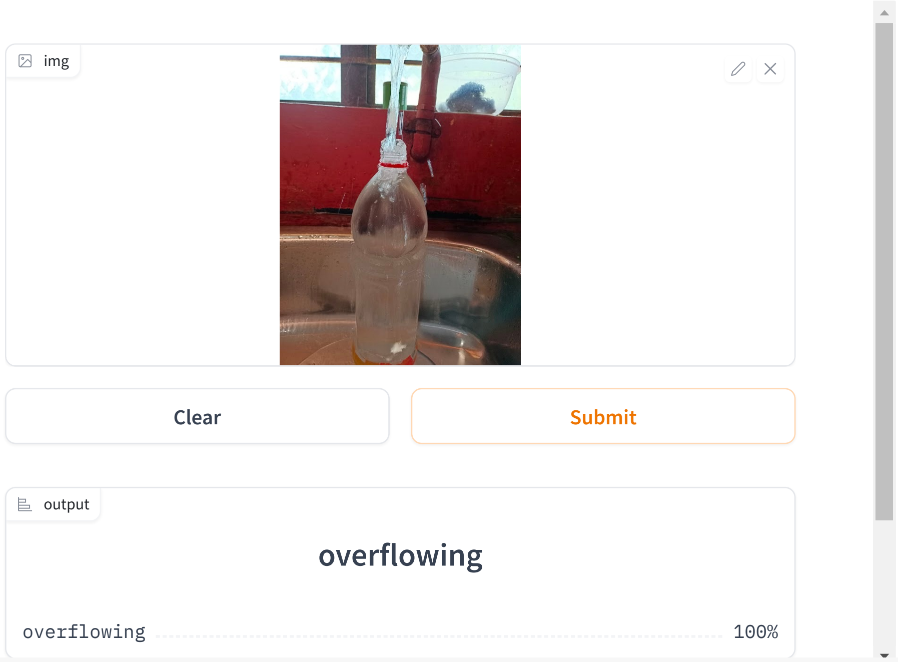
```
1/1 [==============================] - 0s 19ms/step
Keyboard interruption in main thread... closing server.
```

```
image = gr.inputs.Image(shape=(256,256))
label = gr.outputs.Label(num_top_classes=1)

gr.Interface(fn=predict_image, inputs=image, outputs=label,interpretation='default').launch(debug='True')
```

```
/usr/local/lib/python3.8/dist-packages/gradio/inputs.py:256: UserWarning: Usage of gradio.input
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/gradio/deprecation.py:40: UserWarning: `optional` parame
  warnings.warn(value)
/usr/local/lib/python3.8/dist-packages/gradio/outputs.py:196: UserWarning: Usage of gradio.outp
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/gradio/deprecation.py:40: UserWarning: The 'type' parame
  warnings.warn(value)
Colab notebook detected. This cell will run indefinitely so that you can see errors and logs. 
Note: opening Chrome Inspector may crash demo inside Colab notebooks.

To create a public link, set `share=True` in `launch()`.
Running on https://localhost:7860/
```



overflowing

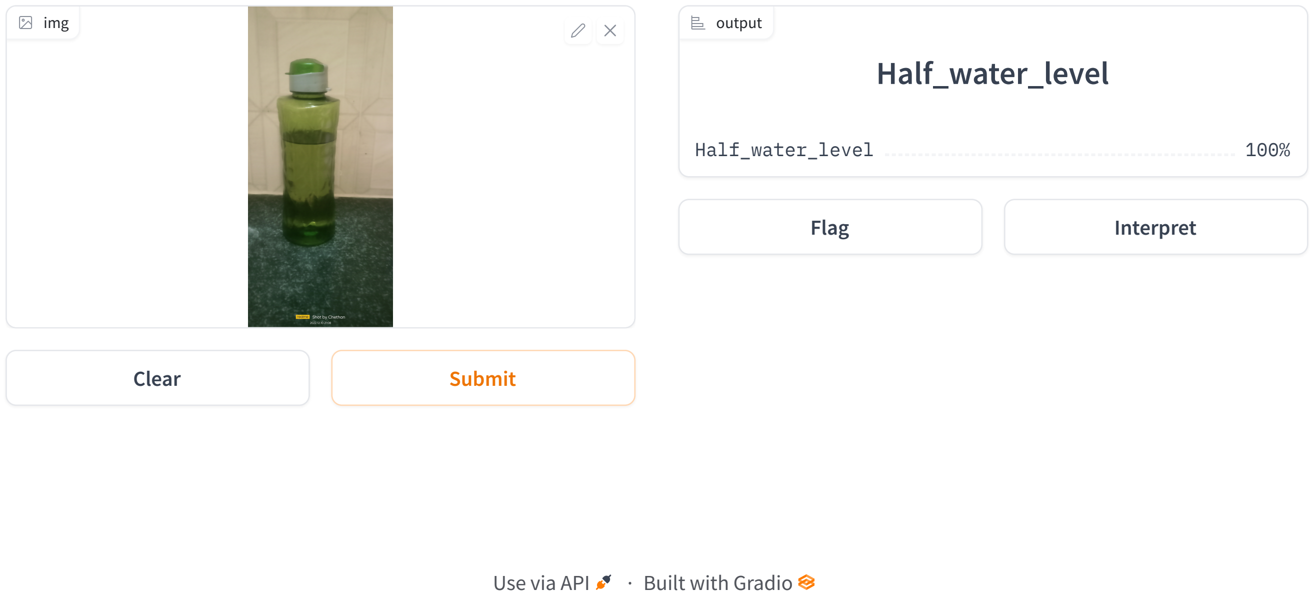
overflowing 100%

```
image = gr.inputs.Image(shape=(256,256))
label = gr.outputs.Label(num_top_classes=1)

gr.Interface(fn=predict_image, inputs=image, outputs=label,interpretation='default').launch(debug='True')
```

```
/usr/local/lib/python3.8/dist-packages/gradio/inputs.py:256: UserWarning: Usage of gradio.inputs is deprecated, and will not be supported in the f
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/gradio/deprecation.py:40: UserWarning: `optional` parameter is deprecated, and it has no effect
  warnings.warn(value)
/usr/local/lib/python3.8/dist-packages/gradio/outputs.py:196: UserWarning: Usage of gradio.outputs is deprecated, and will not be supported in the
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/gradio/deprecation.py:40: UserWarning: The 'type' parameter has been deprecated. Use the Number component i
  warnings.warn(value)
Colab notebook detected. This cell will run indefinitely so that you can see errors and logs. To turn off, set debug=False in launch().
Note: opening Chrome Inspector may crash demo inside Colab notebooks.

To create a public link, set `share=True` in `launch()`.
Running on https://localhost:7860/
```



Half_water_level

Half_water_level 100%

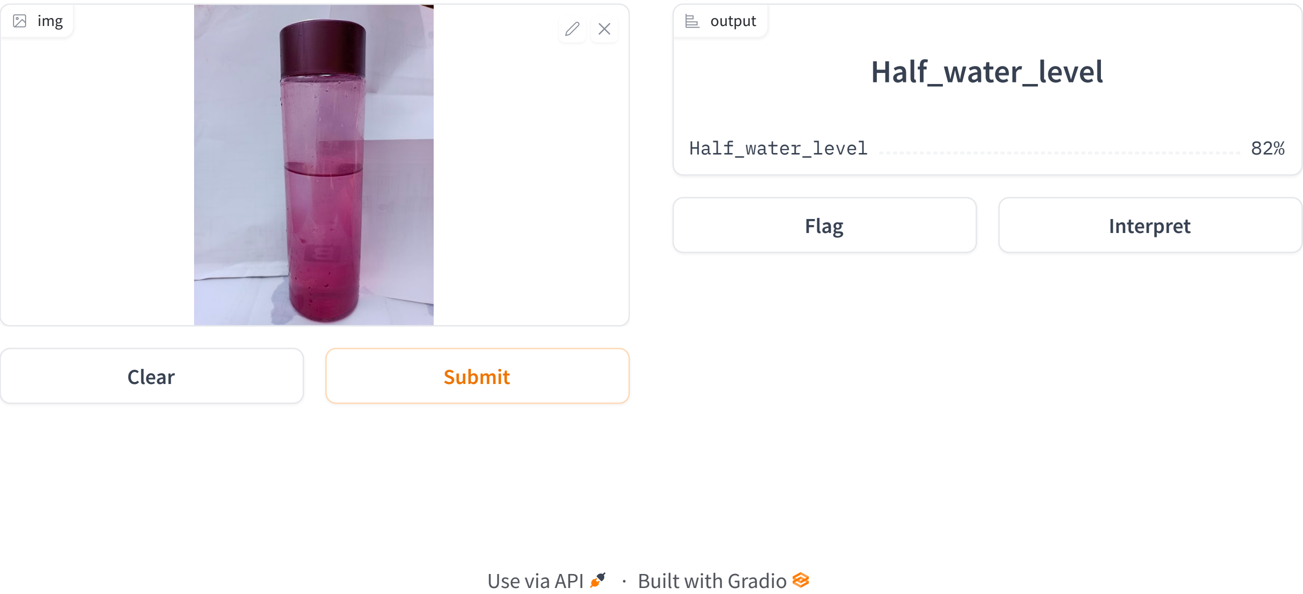
```
1/1 [==============================] - 0s 16ms/step
Keyboard interruption in main thread... closing server.
```

```
image = gr.inputs.Image(shape=(256,256))
label = gr.outputs.Label(num_top_classes=1)

gr.Interface(fn=predict_image, inputs=image, outputs=label,interpretation='default').launch(debug='True')
```

```
/usr/local/lib/python3.8/dist-packages/gradio/inputs.py:256: UserWarning: Usage of gradio.inputs is deprecated, and will not be supported in the f
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/gradio/deprecation.py:40: UserWarning: `optional` parameter is deprecated, and it has no effect
  warnings.warn(value)
/usr/local/lib/python3.8/dist-packages/gradio/outputs.py:196: UserWarning: Usage of gradio.outputs is deprecated, and will not be supported in the
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/gradio/deprecation.py:40: UserWarning: The 'type' parameter has been deprecated. Use the Number component i
  warnings.warn(value)
Colab notebook detected. This cell will run indefinitely so that you can see errors and logs. To turn off, set debug=False in launch().
Note: opening Chrome Inspector may crash demo inside Colab notebooks.

To create a public link, set `share=True` in `launch()`.
Running on https://localhost:7860/
```



Half_water_level

Half_water_level 82%

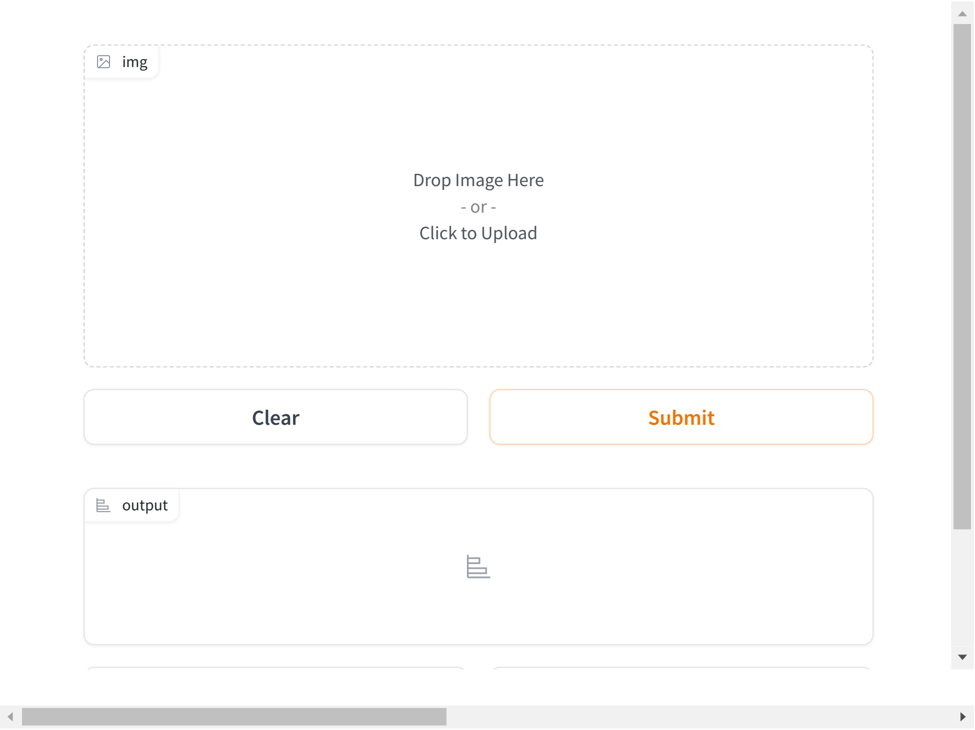
```
1/1 [==============================] - 0s 18ms/step
Keyboard interruption in main thread... closing server.
```

```
/usr/local/lib/python3.8/dist-packages/gradio/inputs.py:256: UserWarning: Usage of gradio.input
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/gradio/deprecation.py:40: UserWarning: `optional` parame
  warnings.warn(value)
/usr/local/lib/python3.8/dist-packages/gradio/outputs.py:196: UserWarning: Usage of gradio.outp
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/gradio/deprecation.py:40: UserWarning: The 'type' parame
  warnings.warn(value)
Colab notebook detected. This cell will run indefinitely so that you can see errors and logs.
Note: opening Chrome Inspector may crash demo inside Colab notebooks.

To create a public link, set `share=True` in `launch()`.
```

```
/usr/local/lib/python3.8/dist-packages/gradio/inputs.py:256: UserWarning: Usage of gradio.input
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/gradio/deprecation.py:40: UserWarning: `optional` parame
  warnings.warn(value)
/usr/local/lib/python3.8/dist-packages/gradio/outputs.py:196: UserWarning: Usage of gradio.outp
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/gradio/deprecation.py:40: UserWarning: The 'type' parame
  warnings.warn(value)
Colab notebook detected. This cell will run indefinitely so that you can see errors and logs.
Note: opening Chrome Inspector may crash demo inside Colab notebooks.

To create a public link, set `share=True` in `launch()`.
Running on https://localhost:7860/
```



Executing (5s) Cell > launch() > block_thread()