In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
import tensorflow as tf

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
import pathlib
import pandas as pd
from PIL import Image
from PIL.ImageDraw import Draw
```

In [2]:
```
width = 480
height = 480

num_classes = 1
classes = ["sign"]
```

Processing Images and Annotations

In [3]:
```
TRAINING_CSV_FILE = 'Data/training_data.csv'
VALIDATION_CSV_FILE = 'Data/validation_data.csv'
TRAINING_IMAGE_DIR = 'Images/training'
VALIDATION_IMAGE_DIR = 'Images/validation'

training_image_records = pd.read_csv(TRAINING_CSV_FILE)
validation_image_records = pd.read_csv(VALIDATION_CSV_FILE)

train_image_path = os.path.join(os.getcwd(), TRAINING_IMAGE_DIR)
validation_image_path = os.path.join(os.getcwd(), VALIDATION_IMAGE_DIR)

train_images = []
train_targets = []
train_labels = []

for index, row in training_image_records.iterrows():

    (filename, width, height, class_name, xmin, ymin, xmax, ymax) = row

    train_image_fullpath = os.path.join(train_image_path, filename)
    train_img = keras.preprocessing.image.load_img(train_image_fullpath, tar
    train_img_arr = keras.preprocessing.image.img_to_array(train_img)

    # Here we make sure the bounding boxes' vertices are always at the same
    xmin = round(xmin/ width, 2)
    ymin = round(ymin/ height, 2)
    xmax = round(xmax/ width, 2)
    ymax = round(ymax/ height, 2)
```

```
    train_images.append(train_img_arr)
    train_targets.append((xmin, ymin, xmax, ymax))
    train_labels.append(classes.index(class_name))


validation_images = []
validation_targets = []
validation_labels = []

for index, row in validation_image_records.iterrows():

    (filename, width, height, class_name, xmin, ymin, xmax, ymax) = row

    validation_image_fullpath = os.path.join(validation_image_path, filename
    validation_img = keras.preprocessing.image.load_img(validation_image_ful
    validation_img_arr = keras.preprocessing.image.img_to_array(validation_i

    xmin = round(xmin/ width, 2)
    ymin = round(ymin/ height, 2)
    xmax = round(xmax/ width, 2)
    ymax = round(ymax/ height, 2)

    validation_images.append(validation_img_arr)
    validation_targets.append((xmin, ymin, xmax, ymax))
    validation_labels.append(classes.index(class_name))
```

Building the Model

In [4]:
```
#create the common input layer
input_shape = (height, width, 3)
input_layer = tf.keras.layers.Input(input_shape)
```

In [5]:
```
base_layers = layers.experimental.preprocessing.Rescaling(1./255, name='bl_1
base_layers = layers.Conv2D(16, 3, padding='same', activation='relu', name='
base_layers = layers.MaxPooling2D(name='bl_3')(base_layers)
base_layers = layers.Conv2D(32, 3, padding='same', activation='relu', name='
base_layers = layers.MaxPooling2D(name='bl_5')(base_layers)
base_layers = layers.Conv2D(64, 3, padding='same', activation='relu', name='
base_layers = layers.MaxPooling2D(name='bl_7')(base_layers)
base_layers = layers.Flatten(name='bl_8')(base_layers)

locator_branch = layers.Dense(128, activation='relu', name='bb_1')(base_laye
locator_branch = layers.Dense(64, activation='relu', name='bb_2')(locator_br
locator_branch = layers.Dense(32, activation='relu', name='bb_3')(locator_br
locator_branch = layers.Dense(4, activation='sigmoid', name='bb_head')(locat
```

In [6]:
```
model = tf.keras.Model(input_layer, outputs=[locator_branch])
```

In [7]:
```
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 480, 480, 3)]     0

 bl_1 (Rescaling)            (None, 480, 480, 3)       0

 bl_2 (Conv2D)               (None, 480, 480, 16)      448

 bl_3 (MaxPooling2D)         (None, 240, 240, 16)      0

 bl_4 (Conv2D)               (None, 240, 240, 32)      4640

 bl_5 (MaxPooling2D)         (None, 120, 120, 32)      0

 bl_6 (Conv2D)               (None, 120, 120, 64)      18496

 bl_7 (MaxPooling2D)         (None, 60, 60, 64)        0

 bl_8 (Flatten)              (None, 230400)            0

 bb_1 (Dense)                (None, 128)               29491328

 bb_2 (Dense)                (None, 64)                8256

 bb_3 (Dense)                (None, 32)                2080

 bb_head (Dense)             (None, 4)                 132

=================================================================
Total params: 29525380 (112.63 MB)
Trainable params: 29525380 (112.63 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [8]:
```
losses = tf.keras.losses.MSE
```

In [9]:
```
model.compile(loss=losses, optimizer='Adam', metrics=['accuracy'])
```

In [10]:
```
training_epochs = 50
print(len(train_images))

train_images = np.array(train_images)
print(np.shape(train_images))
train_targets = np.array(train_targets)
train_labels = np.array(train_labels)

validation_images = np.array(validation_images)
validation_targets = np.array(validation_targets)
validation_labels = np.array(validation_labels)
```

```
100
(100, 480, 480, 3)
```

In [11]:
```
trainTargets = {
```

```
    "bb_head": train_targets
}

validationTargets = {
    "bb_head": validation_targets
}
```

In [12]:
```
print(type(trainTargets))
```

```
<class 'dict'>
```

In [13]:
```
history = model.fit(train_images, trainTargets,
                    validation_data=(validation_images, validationTargets),
                    batch_size=4,
                    epochs=training_epochs,
                    shuffle=True,
                    verbose=1)
```

```
Epoch 1/50
25/25 [==============================] - 5s 167ms/step - loss: 0.0630 - accur
acy: 0.5900 - val_loss: 0.0186 - val_accuracy: 0.5854
Epoch 2/50
25/25 [==============================] - 4s 158ms/step - loss: 0.0154 - accur
acy: 0.6400 - val_loss: 0.0169 - val_accuracy: 0.5854
Epoch 3/50
25/25 [==============================] - 4s 156ms/step - loss: 0.0091 - accur
acy: 0.8000 - val_loss: 0.0096 - val_accuracy: 0.7805
Epoch 4/50
25/25 [==============================] - 4s 160ms/step - loss: 0.0040 - accur
acy: 0.9200 - val_loss: 0.0092 - val_accuracy: 0.7805
Epoch 5/50
25/25 [==============================] - 4s 158ms/step - loss: 0.0030 - accur
acy: 0.8900 - val_loss: 0.0080 - val_accuracy: 0.8780
Epoch 6/50
25/25 [==============================] - 4s 156ms/step - loss: 0.0014 - accur
acy: 0.9100 - val_loss: 0.0074 - val_accuracy: 0.8049
Epoch 7/50
25/25 [==============================] - 4s 169ms/step - loss: 7.8116e-04 - a
ccuracy: 0.9400 - val_loss: 0.0084 - val_accuracy: 0.8049
Epoch 8/50
25/25 [==============================] - 4s 166ms/step - loss: 7.3395e-04 - a
ccuracy: 0.9500 - val_loss: 0.0077 - val_accuracy: 0.8293
Epoch 9/50
25/25 [==============================] - 4s 160ms/step - loss: 6.4111e-04 - a
ccuracy: 0.9500 - val_loss: 0.0074 - val_accuracy: 0.8293
Epoch 10/50
25/25 [==============================] - 4s 173ms/step - loss: 5.2509e-04 - a
ccuracy: 0.9500 - val_loss: 0.0076 - val_accuracy: 0.8049
Epoch 11/50
25/25 [==============================] - 4s 162ms/step - loss: 3.8873e-04 - a
ccuracy: 0.9800 - val_loss: 0.0077 - val_accuracy: 0.8293
Epoch 12/50
25/25 [==============================] - 4s 164ms/step - loss: 3.8200e-04 - a
ccuracy: 0.9500 - val_loss: 0.0074 - val_accuracy: 0.8537
Epoch 13/50
25/25 [==============================] - 4s 157ms/step - loss: 3.3092e-04 - a
ccuracy: 0.9900 - val_loss: 0.0076 - val_accuracy: 0.8293
Epoch 14/50
25/25 [==============================] - 4s 155ms/step - loss: 3.2849e-04 - a
ccuracy: 0.9700 - val_loss: 0.0075 - val_accuracy: 0.7561
Epoch 15/50
25/25 [==============================] - 4s 156ms/step - loss: 3.2339e-04 - a
ccuracy: 0.9400 - val_loss: 0.0072 - val_accuracy: 0.8049
Epoch 16/50
25/25 [==============================] - 4s 155ms/step - loss: 2.6963e-04 - a
ccuracy: 0.9600 - val_loss: 0.0074 - val_accuracy: 0.8293
Epoch 17/50
25/25 [==============================] - 4s 161ms/step - loss: 3.0304e-04 - a
ccuracy: 0.9700 - val_loss: 0.0075 - val_accuracy: 0.8293
Epoch 18/50
25/25 [==============================] - 4s 157ms/step - loss: 3.4072e-04 - a
ccuracy: 0.9500 - val_loss: 0.0076 - val_accuracy: 0.8293
Epoch 19/50
25/25 [==============================] - 4s 155ms/step - loss: 4.4070e-04 - a
```

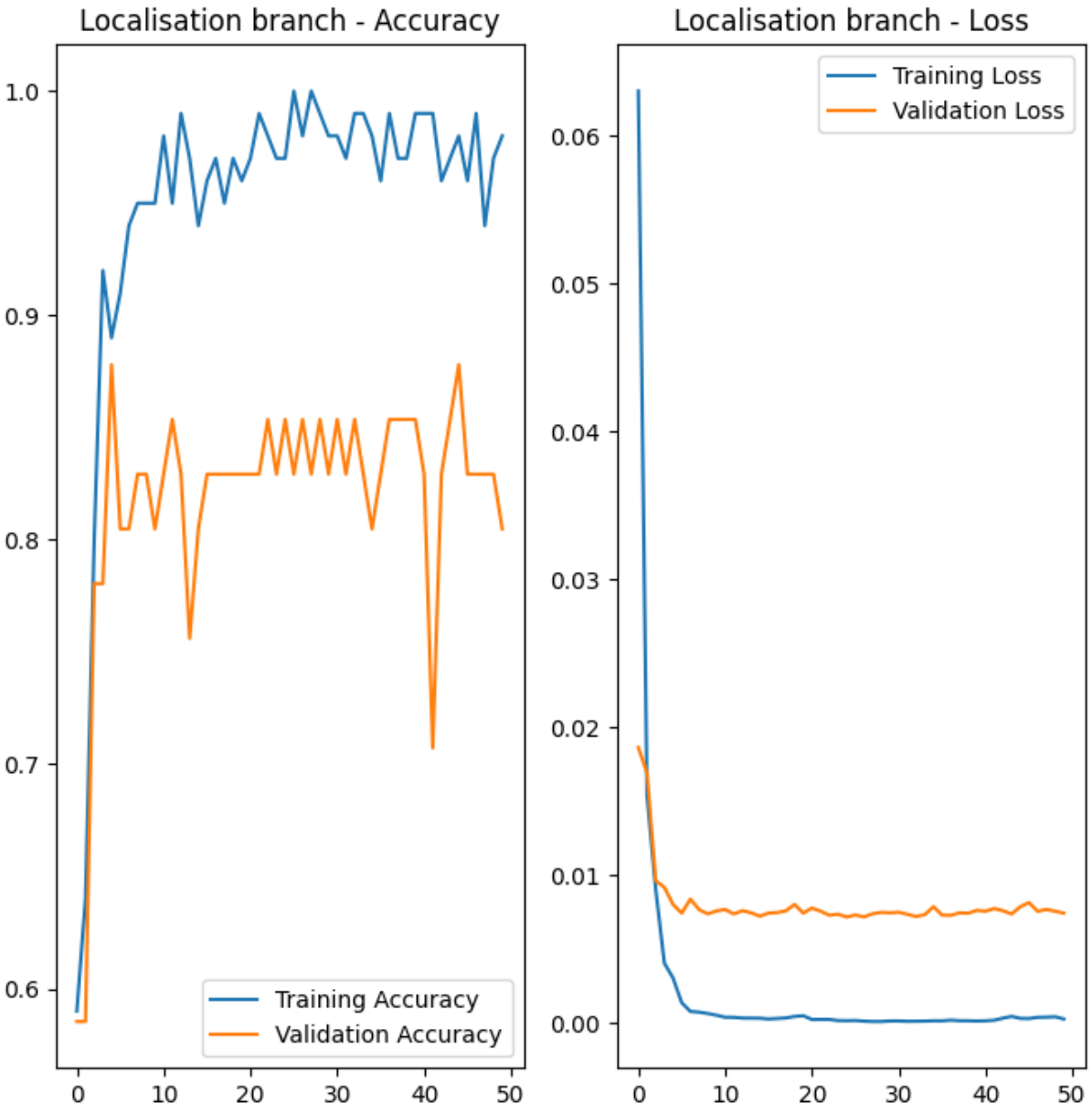
```
ccuracy: 0.9700 - val_loss: 0.0080 - val_accuracy: 0.8293
Epoch 20/50
25/25 [==============================] - 4s 155ms/step - loss: 4.9040e-04 - a
ccuracy: 0.9600 - val_loss: 0.0074 - val_accuracy: 0.8293
Epoch 21/50
25/25 [==============================] - 4s 157ms/step - loss: 2.3586e-04 - a
ccuracy: 0.9700 - val_loss: 0.0078 - val_accuracy: 0.8293
Epoch 22/50
25/25 [==============================] - 4s 157ms/step - loss: 2.3914e-04 - a
ccuracy: 0.9900 - val_loss: 0.0076 - val_accuracy: 0.8293
Epoch 23/50
25/25 [==============================] - 4s 158ms/step - loss: 2.3998e-04 - a
ccuracy: 0.9800 - val_loss: 0.0073 - val_accuracy: 0.8537
Epoch 24/50
25/25 [==============================] - 4s 159ms/step - loss: 1.6565e-04 - a
ccuracy: 0.9700 - val_loss: 0.0074 - val_accuracy: 0.8293
Epoch 25/50
25/25 [==============================] - 4s 164ms/step - loss: 1.5617e-04 - a
ccuracy: 0.9700 - val_loss: 0.0072 - val_accuracy: 0.8537
Epoch 26/50
25/25 [==============================] - 4s 158ms/step - loss: 1.6890e-04 - a
ccuracy: 1.0000 - val_loss: 0.0073 - val_accuracy: 0.8293
Epoch 27/50
25/25 [==============================] - 4s 157ms/step - loss: 1.2552e-04 - a
ccuracy: 0.9800 - val_loss: 0.0072 - val_accuracy: 0.8537
Epoch 28/50
25/25 [==============================] - 4s 164ms/step - loss: 1.0159e-04 - a
ccuracy: 1.0000 - val_loss: 0.0074 - val_accuracy: 0.8293
Epoch 29/50
25/25 [==============================] - 4s 164ms/step - loss: 1.0221e-04 - a
ccuracy: 0.9900 - val_loss: 0.0075 - val_accuracy: 0.8537
Epoch 30/50
25/25 [==============================] - 4s 160ms/step - loss: 1.3653e-04 - a
ccuracy: 0.9800 - val_loss: 0.0075 - val_accuracy: 0.8293
Epoch 31/50
25/25 [==============================] - 4s 161ms/step - loss: 1.3919e-04 - a
ccuracy: 0.9800 - val_loss: 0.0075 - val_accuracy: 0.8537
Epoch 32/50
25/25 [==============================] - 4s 156ms/step - loss: 1.1490e-04 - a
ccuracy: 0.9700 - val_loss: 0.0074 - val_accuracy: 0.8293
Epoch 33/50
25/25 [==============================] - 4s 160ms/step - loss: 1.2204e-04 - a
ccuracy: 0.9900 - val_loss: 0.0072 - val_accuracy: 0.8537
Epoch 34/50
25/25 [==============================] - 4s 157ms/step - loss: 1.2920e-04 - a
ccuracy: 0.9900 - val_loss: 0.0073 - val_accuracy: 0.8293
Epoch 35/50
25/25 [==============================] - 4s 156ms/step - loss: 1.4859e-04 - a
ccuracy: 0.9800 - val_loss: 0.0079 - val_accuracy: 0.8049
Epoch 36/50
25/25 [==============================] - 4s 159ms/step - loss: 1.4652e-04 - a
ccuracy: 0.9600 - val_loss: 0.0073 - val_accuracy: 0.8293
Epoch 37/50
25/25 [==============================] - 4s 156ms/step - loss: 1.9214e-04 - a
ccuracy: 0.9900 - val_loss: 0.0073 - val_accuracy: 0.8537
Epoch 38/50
```

```
25/25 [==============================] - 4s 158ms/step - loss: 1.5317e-04 - a
ccuracy: 0.9700 - val_loss: 0.0074 - val_accuracy: 0.8537
Epoch 39/50
25/25 [==============================] - 4s 155ms/step - loss: 1.5015e-04 - a
ccuracy: 0.9700 - val_loss: 0.0074 - val_accuracy: 0.8537
Epoch 40/50
25/25 [==============================] - 4s 156ms/step - loss: 1.3490e-04 - a
ccuracy: 0.9900 - val_loss: 0.0076 - val_accuracy: 0.8537
Epoch 41/50
25/25 [==============================] - 4s 161ms/step - loss: 1.4752e-04 - a
ccuracy: 0.9900 - val_loss: 0.0076 - val_accuracy: 0.8293
Epoch 42/50
25/25 [==============================] - 4s 166ms/step - loss: 1.8806e-04 - a
ccuracy: 0.9900 - val_loss: 0.0077 - val_accuracy: 0.7073
Epoch 43/50
25/25 [==============================] - 4s 161ms/step - loss: 3.2610e-04 - a
ccuracy: 0.9600 - val_loss: 0.0076 - val_accuracy: 0.8293
Epoch 44/50
25/25 [==============================] - 4s 155ms/step - loss: 4.4658e-04 - a
ccuracy: 0.9700 - val_loss: 0.0074 - val_accuracy: 0.8537
Epoch 45/50
25/25 [==============================] - 4s 154ms/step - loss: 3.1221e-04 - a
ccuracy: 0.9800 - val_loss: 0.0079 - val_accuracy: 0.8780
Epoch 46/50
25/25 [==============================] - 4s 156ms/step - loss: 3.0187e-04 - a
ccuracy: 0.9600 - val_loss: 0.0081 - val_accuracy: 0.8293
Epoch 47/50
25/25 [==============================] - 4s 153ms/step - loss: 3.8423e-04 - a
ccuracy: 0.9900 - val_loss: 0.0076 - val_accuracy: 0.8293
Epoch 48/50
25/25 [==============================] - 4s 151ms/step - loss: 3.9240e-04 - a
ccuracy: 0.9400 - val_loss: 0.0077 - val_accuracy: 0.8293
Epoch 49/50
25/25 [==============================] - 4s 153ms/step - loss: 4.2223e-04 - a
ccuracy: 0.9700 - val_loss: 0.0076 - val_accuracy: 0.8293
Epoch 50/50
25/25 [==============================] - 4s 156ms/step - loss: 2.7108e-04 - a
ccuracy: 0.9800 - val_loss: 0.0074 - val_accuracy: 0.8049
```

In [14]:

```
bb_accuracy = history.history['accuracy']
bb_val_acc = history.history['val_accuracy']

bb_loss = history.history['loss']
bb_val_loss = history.history['val_loss']



epochs_range = range(training_epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, bb_accuracy, label='Training Accuracy')
plt.plot(epochs_range, bb_val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Localisation branch - Accuracy')
```

```
plt.subplot(1, 2, 2)
plt.plot(epochs_range, bb_loss, label='Training Loss')
plt.plot(epochs_range, bb_val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Localisation branch - Loss')
plt.show()
```



In [19]:

```
test_dir = pathlib.Path('Images/test')
img_paths = list(test_dir.glob('*.jpg'))


plt.figure(figsize=(20, 20))

for i, img_path in enumerate(img_paths):
    test_img = keras.preprocessing.image.load_img(img_path, target_size=(hei
    img_arr = keras.preprocessing.image.img_to_array(test_img)
    img_arr = tf.expand_dims(img_arr, 0)

    predictions = model.predict(img_arr)
```

```
    print(predictions)

    bbox = predictions[0]
    bbox = [bbox[0] * width, bbox[1] * height, bbox[2] * width, bbox[3] * he

    draw1 = Draw(test_img)
    draw1.rectangle(bbox, outline='cyan', width=2)
        #test_img

    ax = plt.subplot(6, 5, i+1)
    plt.imshow(test_img)
    plt.axis("off")
```

```
1/1 [==============================] - 0s 25ms/step
[[0.46281928 0.4482857  0.64412576 0.5189963 ]]
1/1 [==============================] - 0s 22ms/step
[[0.45809156 0.43809587 0.64807606 0.5175914 ]]
1/1 [==============================] - 0s 22ms/step
[[0.4651276  0.4345678  0.5910957  0.50153315]]
1/1 [==============================] - 0s 22ms/step
[[0.4927985 0.4561605 0.6270964 0.5141671]]
1/1 [==============================] - 0s 20ms/step
[[0.3678292  0.1978642  0.67193484 0.40086251]]
1/1 [==============================] - 0s 21ms/step
[[0.5134349  0.48509794 0.6603877  0.52138096]]
1/1 [==============================] - 0s 20ms/step
[[0.52950895 0.45523557 0.6484508  0.50636464]]
1/1 [==============================] - 0s 21ms/step
[[0.24008419 0.17058705 0.62376064 0.42314884]]
1/1 [==============================] - 0s 21ms/step
[[0.2577669  0.27012616 0.668748   0.45836544]]
1/1 [==============================] - 0s 21ms/step
[[0.41723317 0.42936638 0.7196637  0.55487907]]
1/1 [==============================] - 0s 21ms/step
[[0.24884035 0.3457871  0.5452062  0.49743208]]
1/1 [==============================] - 0s 21ms/step
[[0.40952894 0.38085672 0.6494406  0.49572852]]
1/1 [==============================] - 0s 21ms/step
[[0.5340578 0.4925578 0.6788949 0.5500652]]
1/1 [==============================] - 0s 22ms/step
[[0.3045203  0.1668789  0.5639035  0.37595075]]
1/1 [==============================] - 0s 23ms/step
[[0.3875958 0.4362633 0.6177927 0.5619406]]
1/1 [==============================] - 0s 22ms/step
[[0.38838744 0.41352844 0.5455041  0.5310311 ]]
1/1 [==============================] - 0s 22ms/step
[[0.3652346  0.30933797 0.6295046  0.48354056]]
1/1 [==============================] - 0s 20ms/step
[[0.34637856 0.37179923 0.7962475  0.5659767 ]]
1/1 [==============================] - 0s 21ms/step
[[0.48236397 0.37327507 0.65130836 0.46984568]]
1/1 [==============================] - 0s 22ms/step
[[0.22652459 0.46971732 0.72277296 0.6946776 ]]
1/1 [==============================] - 0s 22ms/step
[[0.29687464 0.5012101  0.6083357  0.65008557]]
1/1 [==============================] - 0s 21ms/step
[[0.3219837  0.2806773  0.58761704 0.4613152 ]]
1/1 [==============================] - 0s 21ms/step
[[0.2716597  0.39990032 0.5207666  0.5366165 ]]
1/1 [==============================] - 0s 21ms/step
[[0.37162805 0.3941619  0.5627825  0.50594443]]
1/1 [==============================] - 0s 23ms/step
[[0.2327291  0.49224025 0.6758387  0.7111752 ]]
1/1 [==============================] - 0s 21ms/step
[[0.27799928 0.32281303 0.53691804 0.5130597 ]]
1/1 [==============================] - 0s 21ms/step
[[0.3106796  0.50874597 0.6331433  0.6627335 ]]
1/1 [==============================] - 0s 22ms/step
[[0.5508148  0.46227247 0.71086633 0.5262203 ]]
```

```
1/1 [==============================] - 0s 21ms/step
[[0.43896824 0.3708446  0.5743124  0.4768832 ]]
1/1 [==============================] - 0s 21ms/step
[[0.3138904  0.3374242  0.59048593 0.5068008 ]]
```

In [18]:
```
test_dir = pathlib.Path('Images/test2')
img_paths = list(test_dir.glob('*.jpg'))


plt.figure(figsize=(20, 20))

for i, img_path in enumerate(img_paths):
    test_img = keras.preprocessing.image.load_img(img_path, target_size=(hei
    img_arr = keras.preprocessing.image.img_to_array(test_img)
    img_arr = tf.expand_dims(img_arr, 0)

    predictions = model.predict(img_arr)
    print(predictions)
```

```
    bbox = predictions[0]
    bbox = [bbox[0] * width, bbox[1] * height, bbox[2] * width, bbox[3] * he

    draw1 = Draw(test_img)
    draw1.rectangle(bbox, outline='cyan', width=2)
        #test_img

    ax = plt.subplot(6, 5, i+1)
    plt.imshow(test_img)
    plt.axis("off")
```

```
1/1 [==============================] - 0s 24ms/step
[[0.5508148  0.46227247 0.71086633 0.5262203 ]]
1/1 [==============================] - 0s 21ms/step
[[0.27948442 0.45392978 0.5624292  0.5707499 ]]
1/1 [==============================] - 0s 23ms/step
[[0.23795603 0.48662624 0.49212334 0.597246  ]]
1/1 [==============================] - 0s 21ms/step
[[0.5766924  0.4337504  0.73119575 0.5269951 ]]
1/1 [==============================] - 0s 21ms/step
[[0.5021373 0.432592  0.7046188 0.507299 ]]
1/1 [==============================] - 0s 22ms/step
[[0.25016764 0.45876962 0.5029787  0.5773083 ]]
1/1 [==============================] - 0s 22ms/step
[[0.22644201 0.41780722 0.47954834 0.538838  ]]
1/1 [==============================] - 0s 22ms/step
[[0.7196603  0.300886   0.91185826 0.48843247]]
1/1 [==============================] - 0s 20ms/step
[[0.56389827 0.45096198 0.75064075 0.5453222 ]]
1/1 [==============================] - 0s 21ms/step
[[0.37162805 0.3941619  0.5627825  0.50594443]]
1/1 [==============================] - 0s 21ms/step
[[0.61357653 0.48915058 0.6906426  0.53975666]]
1/1 [==============================] - 0s 22ms/step
[[0.26372528 0.4091444  0.48885912 0.51359975]]
1/1 [==============================] - 0s 22ms/step
[[0.2808542  0.4140245  0.50214106 0.5298429 ]]
1/1 [==============================] - 0s 23ms/step
[[0.5884565  0.49607006 0.66516316 0.5386007 ]]
1/1 [==============================] - 0s 22ms/step
[[0.15064299 0.43101126 0.589345   0.6377273 ]]
1/1 [==============================] - 0s 30ms/step
[[0.30288565 0.3947372  0.9362964  0.61624897]]
1/1 [==============================] - 0s 25ms/step
[[0.56475455 0.45758483 0.66402054 0.52319425]]
1/1 [==============================] - 0s 22ms/step
[[0.6728495 0.5120985 0.7318277 0.5422966]]
1/1 [==============================] - 0s 28ms/step
[[0.72825605 0.501818   0.78219754 0.55108356]]
1/1 [==============================] - 0s 21ms/step
[[0.569398   0.46273518 0.65873605 0.52646035]]
1/1 [==============================] - 0s 24ms/step
[[0.67778647 0.50654614 0.7285546  0.5578932 ]]
1/1 [==============================] - 0s 22ms/step
[[0.38943353 0.47383836 0.5040472  0.5269981 ]]
1/1 [==============================] - 0s 21ms/step
[[0.59510404 0.46457112 0.6982028  0.53145826]]
1/1 [==============================] - 0s 21ms/step
[[0.55704635 0.44272104 0.6525481  0.5143303 ]]
1/1 [==============================] - 0s 20ms/step
[[0.5339154  0.4883845  0.67312926 0.58119315]]
1/1 [==============================] - 0s 20ms/step
[[0.28143165 0.35995865 0.51439726 0.538551  ]]
1/1 [==============================] - 0s 22ms/step
[[0.22698306 0.44876617 0.49433452 0.55972886]]
1/1 [==============================] - 0s 21ms/step
[[0.5340578 0.4925578 0.6788949 0.5500652]]
```

```
1/1 [==============================] - 0s 23ms/step
[[0.5870225  0.46006936 0.7226593  0.5456031 ]]
1/1 [==============================] - 0s 22ms/step
[[0.58754927 0.4627597  0.70279837 0.5408365 ]]
```

In [20]:

```
test_dir = pathlib.Path('Images/test3')
img_paths = list(test_dir.glob('*.jpg'))


plt.figure(figsize=(20, 20))

for i, img_path in enumerate(img_paths):
    test_img = keras.preprocessing.image.load_img(img_path, target_size=(hei
    img_arr = keras.preprocessing.image.img_to_array(test_img)
    img_arr = tf.expand_dims(img_arr, 0)

    predictions = model.predict(img_arr)
    print(predictions)
```

```
    bbox = predictions[0]
    bbox = [bbox[0] * width, bbox[1] * height, bbox[2] * width, bbox[3] * he

    draw1 = Draw(test_img)
    draw1.rectangle(bbox, outline='cyan', width=2)
        #test_img

    ax = plt.subplot(6, 5, i+1)
    plt.imshow(test_img)
    plt.axis("off")
```

```
1/1 [==============================] - 0s 25ms/step
[[0.27562454 0.51960427 0.76914907 0.7453672 ]]
1/1 [==============================] - 0s 21ms/step
[[0.3427732  0.40936208 0.6974351  0.6188924 ]]
1/1 [==============================] - 0s 21ms/step
[[0.16135658 0.4131523  0.58814675 0.62416255]]
1/1 [==============================] - 0s 23ms/step
[[0.05002226 0.30050468 0.5418818  0.63023776]]
1/1 [==============================] - 0s 20ms/step
[[0.2646662  0.46822667 0.64990705 0.64745367]]
1/1 [==============================] - 0s 20ms/step
[[0.48236397 0.37327507 0.65130836 0.46984568]]
1/1 [==============================] - 0s 21ms/step
[[0.4635744  0.40470418 0.6127713  0.48057014]]
1/1 [==============================] - 0s 21ms/step
[[0.3106796  0.50874597 0.6331433  0.6627335 ]]
1/1 [==============================] - 0s 21ms/step
[[0.29687464 0.5012101  0.6083357  0.65008557]]
1/1 [==============================] - 0s 21ms/step
[[0.5011715  0.36606988 0.606852   0.43000513]]
1/1 [==============================] - 0s 21ms/step
[[0.47196296 0.40942663 0.5799865  0.49884665]]
1/1 [==============================] - 0s 21ms/step
[[0.42158067 0.42885673 0.51993716 0.49721068]]
1/1 [==============================] - 0s 22ms/step
[[0.35445973 0.45656577 0.5844166  0.5606887 ]]
1/1 [==============================] - 0s 23ms/step
[[0.0377108  0.2653488  0.92587966 0.6370258 ]]
1/1 [==============================] - 0s 23ms/step
[[0.27335805 0.5359581  0.59752095 0.74745506]]
1/1 [==============================] - 0s 22ms/step
[[0.32500944 0.21977612 0.6413764  0.45205694]]
1/1 [==============================] - 0s 22ms/step
[[0.14872396 0.39000696 0.5304915  0.6055774 ]]
1/1 [==============================] - 0s 21ms/step
[[0.300133   0.4062915  0.67116624 0.601342  ]]
1/1 [==============================] - 0s 21ms/step
[[0.27226096 0.40797213 0.52191436 0.5422552 ]]
1/1 [==============================] - 0s 25ms/step
[[0.38838744 0.41352844 0.5455041  0.5310311 ]]
1/1 [==============================] - 0s 22ms/step
[[0.19434287 0.3881554  0.5858703  0.6046384 ]]
1/1 [==============================] - 0s 21ms/step
[[0.44096434 0.43422946 0.7089658  0.5315649 ]]
1/1 [==============================] - 0s 21ms/step
[[0.15326469 0.14976402 0.8252921  0.43398252]]
1/1 [==============================] - 0s 21ms/step
[[0.362608   0.3997101  0.42381474 0.48810494]]
1/1 [==============================] - 0s 20ms/step
[[0.3534399  0.41195378 0.53806984 0.5258205 ]]
1/1 [==============================] - 0s 21ms/step
[[0.32357442 0.5424925  0.5683681  0.76464134]]
1/1 [==============================] - 0s 23ms/step
[[0.2716597  0.39990032 0.5207666  0.5366165 ]]
1/1 [==============================] - 0s 22ms/step
[[0.35690656 0.39211142 0.8712833  0.7201662 ]]
```

```
1/1 [==============================] - 0s 23ms/step
[[0.40456724 0.5567611  0.79680705 0.7988434 ]]
1/1 [==============================] - 0s 22ms/step
[[0.27190286 0.38744965 0.51739466 0.531398  ]]
```

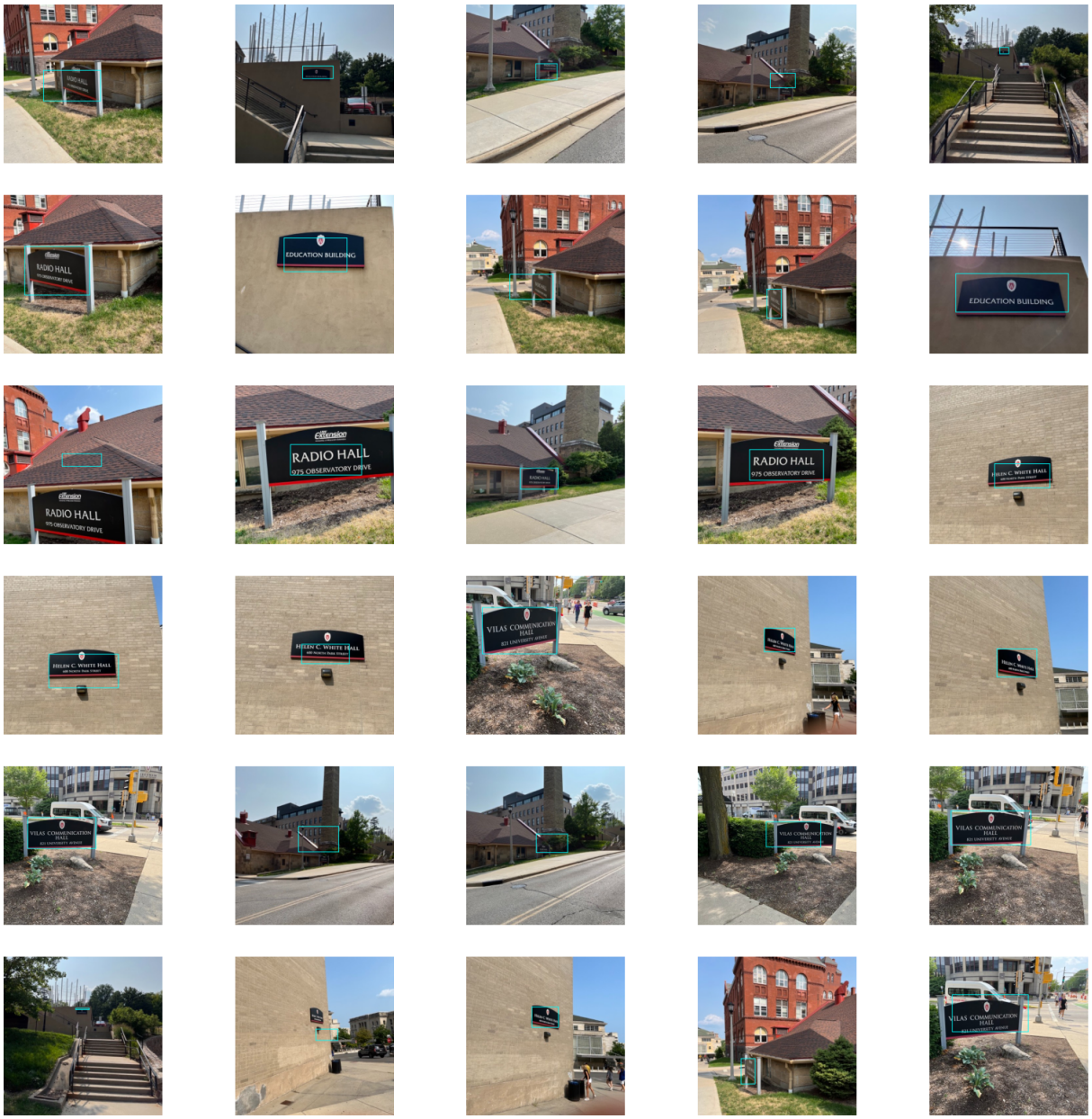
In [21]:

```
test_dir = pathlib.Path('Images/test4')
img_paths = list(test_dir.glob('*.jpg'))


plt.figure(figsize=(20, 20))

for i, img_path in enumerate(img_paths):
    test_img = keras.preprocessing.image.load_img(img_path, target_size=(hei
    img_arr = keras.preprocessing.image.img_to_array(test_img)
    img_arr = tf.expand_dims(img_arr, 0)

    predictions = model.predict(img_arr)
    print(predictions)
```

```
    bbox = predictions[0]
    bbox = [bbox[0] * width, bbox[1] * height, bbox[2] * width, bbox[3] * he

    draw1 = Draw(test_img)
    draw1.rectangle(bbox, outline='cyan', width=2)
        #test_img

    ax = plt.subplot(6, 5, i+1)
    plt.imshow(test_img)
    plt.axis("off")
```

```
1/1 [==============================] - 0s 24ms/step
[[0.2510562 0.4137792 0.6104995 0.6087847]]
1/1 [==============================] - 0s 23ms/step
[[0.4235955  0.38619325 0.6166845  0.4671047 ]]
1/1 [==============================] - 0s 21ms/step
[[0.43896824 0.3708446  0.5743124  0.4768832 ]]
1/1 [==============================] - 0s 21ms/step
[[0.4546889  0.4315351  0.61661077 0.52640486]]
1/1 [==============================] - 0s 21ms/step
[[0.44193968 0.26979816 0.50061476 0.3123401 ]]
1/1 [==============================] - 0s 22ms/step
[[0.13265258 0.32318234 0.54286706 0.6323796 ]]
1/1 [==============================] - 0s 21ms/step
[[0.3074138  0.27073023 0.70546246 0.48378888]]
1/1 [==============================] - 0s 22ms/step
[[0.27490455 0.49958822 0.5489544  0.6595868 ]]
1/1 [==============================] - 0s 21ms/step
[[0.43471214 0.5918256  0.52591103 0.7780924 ]]
1/1 [==============================] - 0s 21ms/step
[[0.16464922 0.49580422 0.87658507 0.7375429 ]]
1/1 [==============================] - 0s 21ms/step
[[0.36771014 0.42996195 0.61807287 0.5121916 ]]
1/1 [==============================] - 0s 20ms/step
[[0.34637856 0.37179923 0.7962475  0.5659767 ]]
1/1 [==============================] - 0s 21ms/step
[[0.34122595 0.52074367 0.5807272  0.65524465]]
1/1 [==============================] - 0s 20ms/step
[[0.32584146 0.40360156 0.7925955  0.5999362 ]]
1/1 [==============================] - 0s 21ms/step
[[0.4102288  0.4936819  0.75883186 0.64611745]]
1/1 [==============================] - 0s 22ms/step
[[0.28509676 0.49246094 0.72467595 0.7076998 ]]
1/1 [==============================] - 0s 23ms/step
[[0.41723317 0.42936638 0.7196637  0.55487907]]
1/1 [==============================] - 0s 21ms/step
[[0.10070655 0.1993469  0.572769   0.49313492]]
1/1 [==============================] - 0s 20ms/step
[[0.4169833  0.32767287 0.6140481  0.48056242]]
1/1 [==============================] - 0s 22ms/step
[[0.42469275 0.45854744 0.67899555 0.6386448 ]]
1/1 [==============================] - 0s 22ms/step
[[0.15188938 0.32100046 0.58275294 0.5196867 ]]
1/1 [==============================] - 0s 21ms/step
[[0.39707357 0.37734184 0.65339696 0.54004484]]
1/1 [==============================] - 0s 21ms/step
[[0.44108802 0.42685682 0.6404031  0.54515934]]
1/1 [==============================] - 0s 21ms/step
[[0.43056828 0.3485165  0.77590543 0.50858355]]
1/1 [==============================] - 0s 20ms/step
[[0.13379371 0.28053227 0.6385824  0.49647635]]
1/1 [==============================] - 0s 20ms/step
[[0.45289746 0.32408655 0.540896   0.33418804]]
1/1 [==============================] - 0s 21ms/step
[[0.5065976  0.45509338 0.646822   0.52911615]]
1/1 [==============================] - 0s 22ms/step
[[0.41429737 0.31618136 0.58271533 0.44585806]]
```

```
1/1 [==============================] - 0s 22ms/step
[[0.27150837 0.6410867  0.3619813  0.8036262 ]]
1/1 [==============================] - 0s 21ms/step
[[0.14276077 0.23916848 0.6249135  0.47387236]]
```



In [22]:
```
test_dir = pathlib.Path('Images/test5')
img_paths = list(test_dir.glob('*.jpg'))


plt.figure(figsize=(20, 20))

for i, img_path in enumerate(img_paths):
    test_img = keras.preprocessing.image.load_img(img_path, target_size=(hei
    img_arr = keras.preprocessing.image.img_to_array(test_img)
    img_arr = tf.expand_dims(img_arr, 0)

    predictions = model.predict(img_arr)
    print(predictions)
```

```
    bbox = predictions[0]
    bbox = [bbox[0] * width, bbox[1] * height, bbox[2] * width, bbox[3] * he

    draw1 = Draw(test_img)
    draw1.rectangle(bbox, outline='cyan', width=2)
        #test_img

    ax = plt.subplot(6, 5, i+1)
    plt.imshow(test_img)
    plt.axis("off")
```

```
1/1 [==============================] - 0s 22ms/step
[[0.33223802 0.59642285 0.5588345  0.70999163]]
1/1 [==============================] - 0s 22ms/step
[[0.3675995  0.58751374 0.57128257 0.70018375]]
1/1 [==============================] - 0s 21ms/step
[[0.4237034  0.46278632 0.6399201  0.5874194 ]]
1/1 [==============================] - 0s 21ms/step
[[0.23885894 0.5395471  0.68205005 0.7528144 ]]
1/1 [==============================] - 0s 20ms/step
[[0.36012992 0.45259076 0.5929761  0.5779837 ]]
1/1 [==============================] - 0s 21ms/step
[[0.3875958 0.4362633 0.6177927 0.5619406]]
1/1 [==============================] - 0s 21ms/step
[[0.37523946 0.6474228  0.53258884 0.72801536]]
1/1 [==============================] - 0s 24ms/step
[[0.3045203  0.1668789  0.5639035  0.37595075]]
1/1 [==============================] - 0s 20ms/step
[[0.3678292  0.1978642  0.67193484 0.40086251]]
1/1 [==============================] - 0s 20ms/step
[[0.29797414 0.5170383  0.54529923 0.64228535]]
1/1 [==============================] - 0s 22ms/step
[[0.28343782 0.51995844 0.51164806 0.6388029 ]]
1/1 [==============================] - 0s 20ms/step
[[0.20106408 0.30971932 0.5635595  0.47781274]]
1/1 [==============================] - 0s 20ms/step
[[0.2704828  0.1510944  0.611092   0.39982355]]
1/1 [==============================] - 0s 21ms/step
[[0.2918726  0.5422186  0.56276655 0.6653347 ]]
1/1 [==============================] - 0s 23ms/step
[[0.37064472 0.18047646 0.7691673  0.35927618]]
1/1 [==============================] - 0s 23ms/step
[[0.29021102 0.15173197 0.740669   0.36299196]]
1/1 [==============================] - 0s 22ms/step
[[0.17506792 0.4739464  0.48851663 0.6256444 ]]
1/1 [==============================] - 0s 21ms/step
[[0.26169574 0.20288815 0.56908613 0.4264687 ]]
1/1 [==============================] - 0s 21ms/step
[[0.18574457 0.05897149 0.66875285 0.39790234]]
1/1 [==============================] - 0s 22ms/step
[[0.17245556 0.43035227 0.6041787  0.6466182 ]]
1/1 [==============================] - 0s 21ms/step
[[0.24008419 0.17058705 0.62376064 0.42314884]]
1/1 [==============================] - 0s 22ms/step
[[0.28219068 0.12538248 0.63712496 0.37756842]]
1/1 [==============================] - 0s 22ms/step
[[0.24884035 0.3457871  0.5452062  0.49743208]]
1/1 [==============================] - 0s 21ms/step
[[0.18829297 0.45852417 0.55081743 0.62568253]]
1/1 [==============================] - 0s 23ms/step
[[0.191195   0.10952079 0.67063695 0.42874798]]
1/1 [==============================] - 0s 21ms/step
[[0.24957253 0.16560918 0.7232698  0.42074484]]
1/1 [==============================] - 0s 21ms/step
[[0.37475926 0.368758   0.6742454  0.51245457]]
1/1 [==============================] - 0s 22ms/step
[[0.35140514 0.15997148 0.7068531  0.45858818]]
```

```
1/1 [==============================] - 0s 23ms/step
[[0.32535937 0.26131204 0.7143306  0.45321685]]
1/1 [==============================] - 0s 22ms/step
[[0.29776475 0.30409893 0.7900158  0.48628432]]
```

In [23]:
```
test_dir = pathlib.Path('Images/test6')
img_paths = list(test_dir.glob('*.jpg'))


plt.figure(figsize=(20, 20))

for i, img_path in enumerate(img_paths):
    test_img = keras.preprocessing.image.load_img(img_path, target_size=(hei
    img_arr = keras.preprocessing.image.img_to_array(test_img)
    img_arr = tf.expand_dims(img_arr, 0)

    predictions = model.predict(img_arr)
    print(predictions)
```

```
    bbox = predictions[0]
    bbox = [bbox[0] * width, bbox[1] * height, bbox[2] * width, bbox[3] * he

    draw1 = Draw(test_img)
    draw1.rectangle(bbox, outline='cyan', width=2)
        #test_img

    ax = plt.subplot(6, 5, i+1)
    plt.imshow(test_img)
    plt.axis("off")
```

```
1/1 [==============================] - 0s 22ms/step
[[0.2327291  0.49224025 0.6758387  0.7111752 ]]
1/1 [==============================] - 0s 22ms/step
[[0.20829114 0.4550381  0.67430377 0.679453  ]]
1/1 [==============================] - 0s 20ms/step
[[0.3116277  0.5159067  0.60909283 0.6567496 ]]
1/1 [==============================] - 0s 22ms/step
[[0.40655857 0.53224236 0.6991336  0.6928544 ]]
1/1 [==============================] - 0s 20ms/step
[[0.22652459 0.46971732 0.72277296 0.6946776 ]]
1/1 [==============================] - 0s 21ms/step
[[0.447338   0.41562882 0.6397395  0.5252299 ]]
1/1 [==============================] - 0s 21ms/step
[[0.38051912 0.37776068 0.6190316  0.50985086]]
1/1 [==============================] - 0s 22ms/step
[[0.40348735 0.2992174  0.6130713  0.5062635 ]]
1/1 [==============================] - 0s 21ms/step
[[0.5063113  0.47302824 0.6295507  0.5510796 ]]
1/1 [==============================] - 0s 21ms/step
[[0.5431022  0.1553719  0.6701129  0.24248819]]
1/1 [==============================] - 0s 21ms/step
[[0.36141995 0.43957457 0.6358401  0.56771   ]]
1/1 [==============================] - 0s 21ms/step
[[0.37903833 0.48783568 0.66656053 0.61499786]]
1/1 [==============================] - 0s 21ms/step
[[0.31487933 0.31493482 0.5479151  0.5136556 ]]
1/1 [==============================] - 0s 21ms/step
[[0.31223208 0.3220937  0.5832852  0.5386773 ]]
1/1 [==============================] - 0s 21ms/step
[[0.40226215 0.47321102 0.62769467 0.57669985]]
1/1 [==============================] - 0s 21ms/step
[[0.25933856 0.49691162 0.6510583  0.6902875 ]]
1/1 [==============================] - 0s 22ms/step
[[0.3137554  0.55411386 0.5729551  0.68811035]]
1/1 [==============================] - 0s 21ms/step
[[0.42388058 0.4254233  0.60392374 0.5423428 ]]
1/1 [==============================] - 0s 21ms/step
[[0.45372295 0.41087252 0.61175925 0.5240063 ]]
1/1 [==============================] - 0s 21ms/step
[[0.26279107 0.56249434 0.491068   0.7000996 ]]
1/1 [==============================] - 0s 20ms/step
[[0.2715998 0.5455548 0.5926036 0.7566904]]
1/1 [==============================] - 0s 24ms/step
[[0.27725953 0.4809487  0.6444552  0.65112174]]
1/1 [==============================] - 0s 21ms/step
[[0.4303455  0.43262598 0.66636163 0.5441271 ]]
1/1 [==============================] - 0s 21ms/step
[[0.3775917  0.48036537 0.6998768  0.6233284 ]]
1/1 [==============================] - 0s 21ms/step
[[0.43839988 0.26942062 0.7315676  0.39840463]]
1/1 [==============================] - 0s 21ms/step
[[0.43161765 0.32296252 0.6498027  0.43224645]]
1/1 [==============================] - 0s 21ms/step
[[0.3652346  0.30933797 0.6295046  0.48354056]]
1/1 [==============================] - 0s 21ms/step
[[0.32782176 0.5228783  0.6363258  0.68263274]]
```

```
1/1 [==============================] - 0s 21ms/step
[[0.3855456  0.37322435 0.6015159  0.5206019 ]]
1/1 [==============================] - 0s 30ms/step
[[0.49604124 0.39397928 0.6233331  0.48209623]]
```

In [24]:

```
test_dir = pathlib.Path('Images/test7')
img_paths = list(test_dir.glob('*.jpg'))


plt.figure(figsize=(20, 20))

for i, img_path in enumerate(img_paths):
    test_img = keras.preprocessing.image.load_img(img_path, target_size=(hei
    img_arr = keras.preprocessing.image.img_to_array(test_img)
    img_arr = tf.expand_dims(img_arr, 0)

    predictions = model.predict(img_arr)
    print(predictions)
```

```
    bbox = predictions[0]
    bbox = [bbox[0] * width, bbox[1] * height, bbox[2] * width, bbox[3] * he

    draw1 = Draw(test_img)
    draw1.rectangle(bbox, outline='cyan', width=2)
        #test_img

    ax = plt.subplot(6, 5, i+1)
    plt.imshow(test_img)
    plt.axis("off")
```

```
1/1 [==============================] - 0s 26ms/step
[[0.2761278  0.5055767  0.48461014 0.61189854]]
1/1 [==============================] - 0s 26ms/step
[[0.13974635 0.21307302 0.75858974 0.49766526]]
1/1 [==============================] - 0s 20ms/step
[[0.1966689  0.32284376 0.74078894 0.5515433 ]]
1/1 [==============================] - 0s 21ms/step
[[0.36242515 0.42062062 0.6524133  0.5611263 ]]
1/1 [==============================] - 0s 20ms/step
[[0.22706397 0.32426634 0.61815417 0.50860494]]
1/1 [==============================] - 0s 21ms/step
[[0.29666585 0.37564605 0.72610617 0.5736548 ]]
1/1 [==============================] - 0s 20ms/step
[[0.34970286 0.32000878 0.57606405 0.46727005]]
1/1 [==============================] - 0s 22ms/step
[[0.3138904  0.3374242  0.59048593 0.5068008 ]]
1/1 [==============================] - 0s 22ms/step
[[0.3086381  0.2571434  0.6690693  0.48323816]]
1/1 [==============================] - 0s 20ms/step
[[0.700441   0.45190513 0.8020227  0.5788556 ]]
1/1 [==============================] - 0s 23ms/step
[[0.7763053 0.4280716 0.9097229 0.5252449]]
1/1 [==============================] - 0s 21ms/step
[[0.28498545 0.3396767  0.61296535 0.5038487 ]]
1/1 [==============================] - 0s 23ms/step
[[0.2577669  0.27012616 0.668748   0.45836544]]
1/1 [==============================] - 0s 30ms/step
[[0.3529035  0.2593301  0.8087755  0.42890564]]
1/1 [==============================] - 0s 21ms/step
[[0.53780156 0.4972745  0.81704575 0.6151823 ]]
1/1 [==============================] - 0s 20ms/step
[[0.30372196 0.26369703 0.6610698  0.45562196]]
1/1 [==============================] - 0s 20ms/step
[[0.40952894 0.38085672 0.6494406  0.49572852]]
1/1 [==============================] - 0s 20ms/step
[[0.6829286  0.3240939  0.86163557 0.44125754]]
1/1 [==============================] - 0s 22ms/step
[[0.38026133 0.3381979  0.6418481  0.47090608]]
1/1 [==============================] - 0s 21ms/step
[[0.3219837  0.2806773  0.58761704 0.4613152 ]]
1/1 [==============================] - 0s 21ms/step
[[0.52475876 0.4966131  0.80857474 0.62143785]]
1/1 [==============================] - 0s 21ms/step
[[0.06559242 0.23240717 0.64683884 0.4655144 ]]
1/1 [==============================] - 0s 22ms/step
[[0.03041572 0.22160062 0.56966877 0.46341726]]
1/1 [==============================] - 0s 22ms/step
[[0.46727416 0.47965804 0.7700626  0.5995648 ]]
1/1 [==============================] - 0s 22ms/step
[[0.2003838  0.24612685 0.37321794 0.399646  ]]
1/1 [==============================] - 0s 22ms/step
[[0.3941683  0.4436239  0.64734644 0.5379286 ]]
1/1 [==============================] - 0s 22ms/step
[[0.32886672 0.23317465 0.61476713 0.43255025]]
1/1 [==============================] - 0s 22ms/step
[[0.16727684 0.26254746 0.3716781  0.41988584]]
```

```
1/1 [==============================] - 0s 21ms/step
[[0.5077393  0.48862728 0.79542375 0.60954165]]
```

In [16]:
```
model.save('./Saved-Models/Locator')
```

```
INFO:tensorflow:Assets written to: ./Saved-Models/Locator/assets
INFO:tensorflow:Assets written to: ./Saved-Models/Locator/assets
```