Carvana Image Segmentation

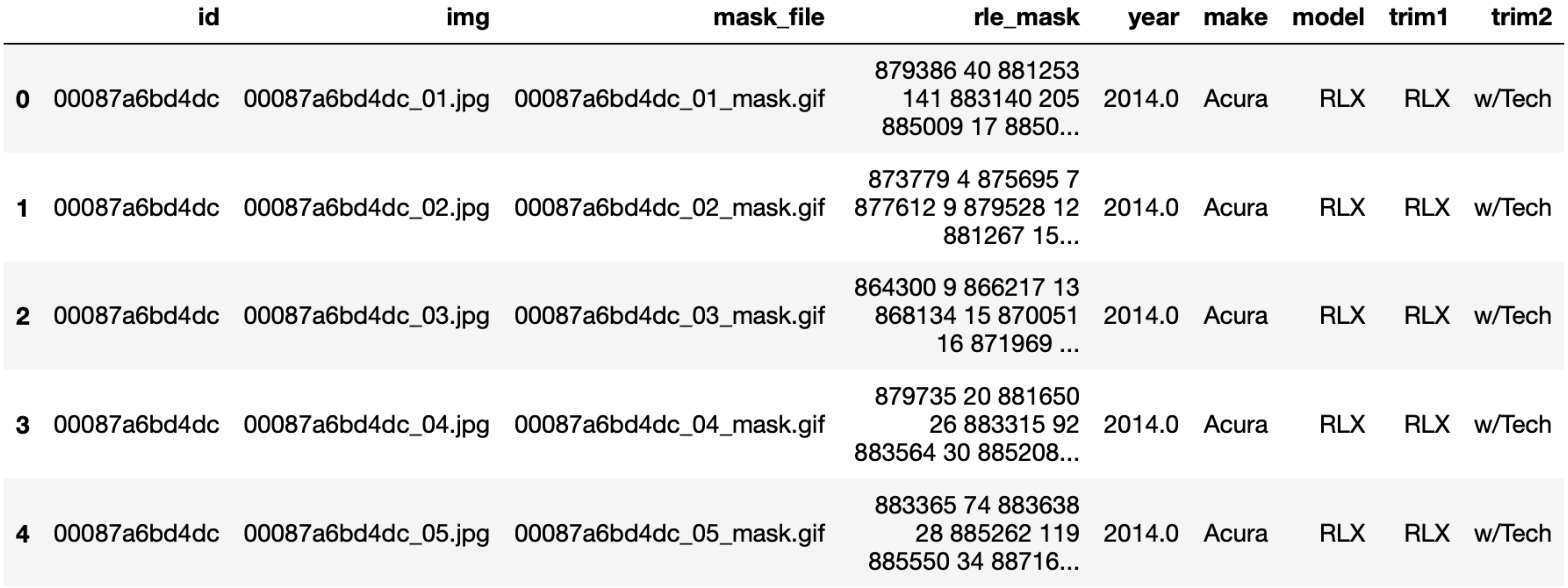
Holmusk - Coding Assignment

Zaini Chia, 11th March 2021

Overview

- Exploratory Data Analysis
- Run-Length-Encoding
- Model Training (CNN)
- Model Training (MobileNetV2)
- Predictions
- Evaluation and Conclusion

EDA



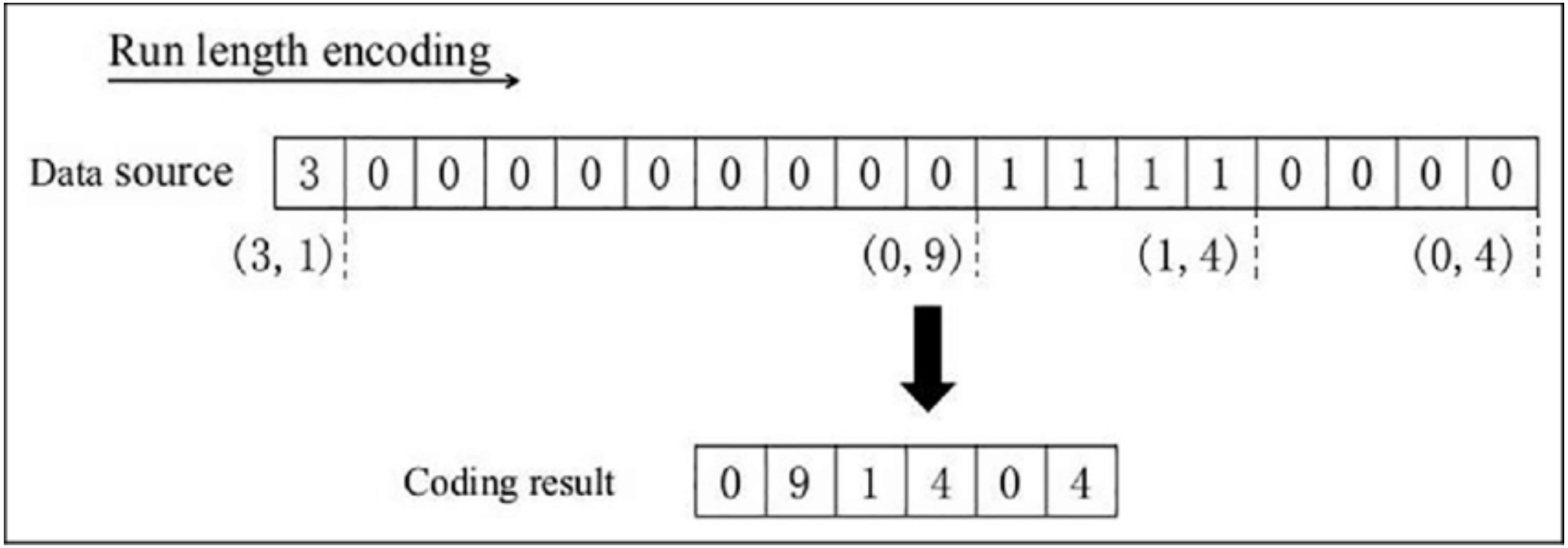
	id	img	mask_file	rle_mask	year	make	model	trim1	trim2
0	00087a6bd4dc	00087a6bd4dc_01.jpg	00087a6bd4dc_01_mask.gif	879386 40 881253 141 883140 205 885009 17 8850...	2014.0	Acura	RLX	RLX	w/Tech
1	00087a6bd4dc	00087a6bd4dc_02.jpg	00087a6bd4dc_02_mask.gif	873779 4 875695 7 877612 9 879528 12 881267 15...	2014.0	Acura	RLX	RLX	w/Tech
2	00087a6bd4dc	00087a6bd4dc_03.jpg	00087a6bd4dc_03_mask.gif	864300 9 866217 13 868134 15 870051 16 871969 ...	2014.0	Acura	RLX	RLX	w/Tech
3	00087a6bd4dc	00087a6bd4dc_04.jpg	00087a6bd4dc_04_mask.gif	879735 20 881650 26 883315 92 883564 30 885208...	2014.0	Acura	RLX	RLX	w/Tech
4	00087a6bd4dc	00087a6bd4dc_05.jpg	00087a6bd4dc_05_mask.gif	883365 74 883638 28 885262 119 885550 34 88716...	2014.0	Acura	RLX	RLX	w/Tech

- We merge the metadata files with the mask files, and create filenames for both the images and corresponding masks

- This will help us in the creation of the Tensorflow Dataset later on

- We can also use this to do further evaluation (misclassified data, etc)

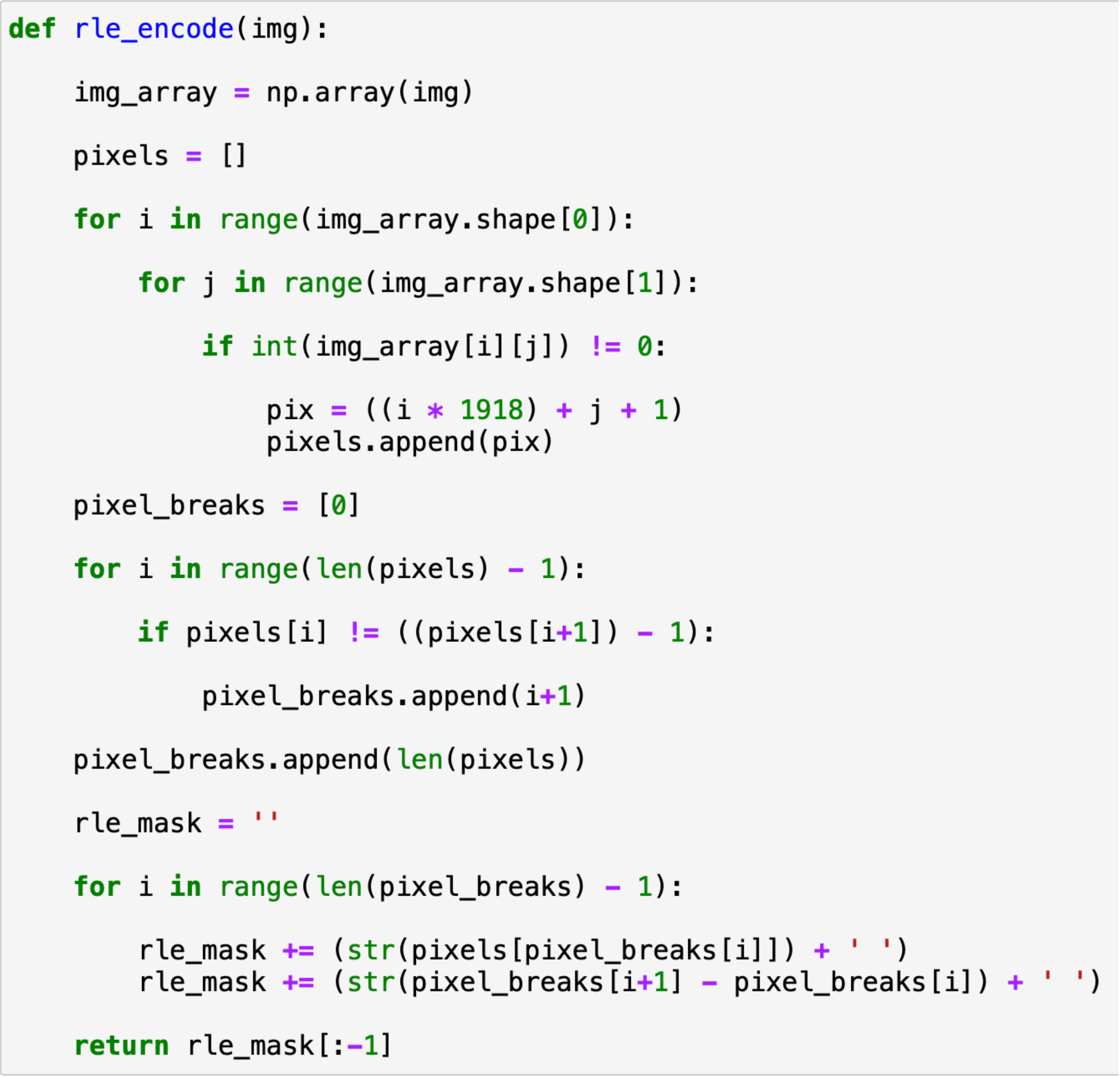
Run-Length-Encoding



- Kaggle submission requires images to be Run-Length-Encoded

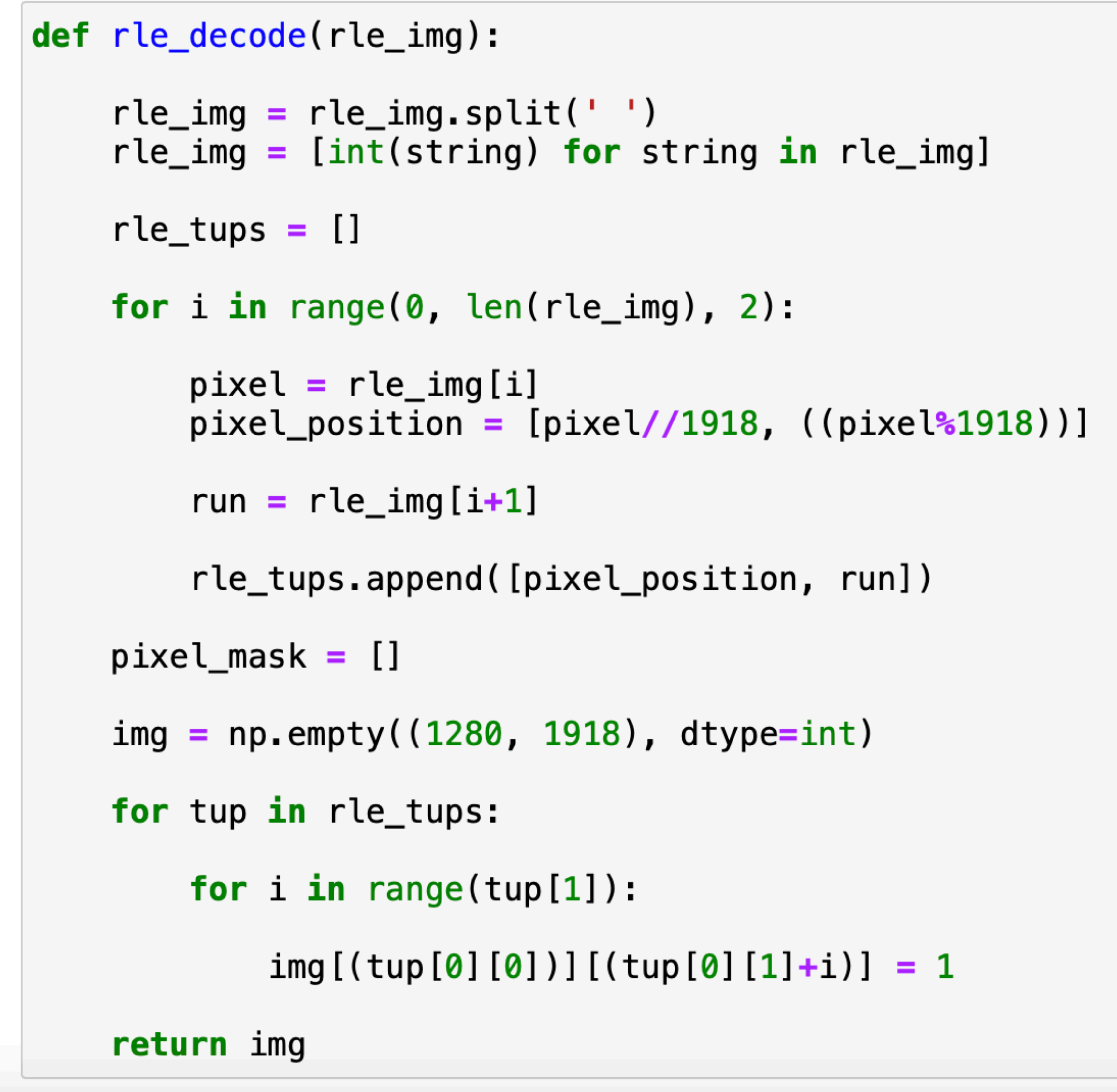
- “The competition format requires a space delimited list of pairs.
  For example, '1 3 10 5' implies pixels 1,2,3,10,11,12,13,14 are to be included in the mask”

Run-Length-Encoding



```python
def rle_encode(img):

    img_array = np.array(img)

    pixels = []

    for i in range(img_array.shape[0]):

        for j in range(img_array.shape[1]):

            if int(img_array[i][j]) != 0:

                pix = ((i * 1918) + j + 1)
                pixels.append(pix)

    pixel_breaks = [0]

    for i in range(len(pixels) - 1):

        if pixels[i] != ((pixels[i+1]) - 1):

            pixel_breaks.append(i+1)

    pixel_breaks.append(len(pixels))

    rle_mask = ''

    for i in range(len(pixel_breaks) - 1):

        rle_mask += (str(pixels[pixel_breaks[i]]) + ' ')
        rle_mask += (str(pixel_breaks[i+1] - pixel_breaks[i]) + ' ')

    return rle_mask[:-1]
```

Run-Length-Decoding



```
def rle_decode(rle_img):

    rle_img = rle_img.split(' ')
    rle_img = [int(string) for string in rle_img]

    rle_tups = []

    for i in range(0, len(rle_img), 2):

        pixel = rle_img[i]
        pixel_position = [pixel//1918, ((pixel%1918))]

        run = rle_img[i+1]

        rle_tups.append([pixel_position, run])

    pixel_mask = []

    img = np.empty((1280, 1918), dtype=int)

    for tup in rle_tups:

        for i in range(tup[1]):

            img[(tup[0][0])][(tup[0][1]+i)] = 1

    return img
```

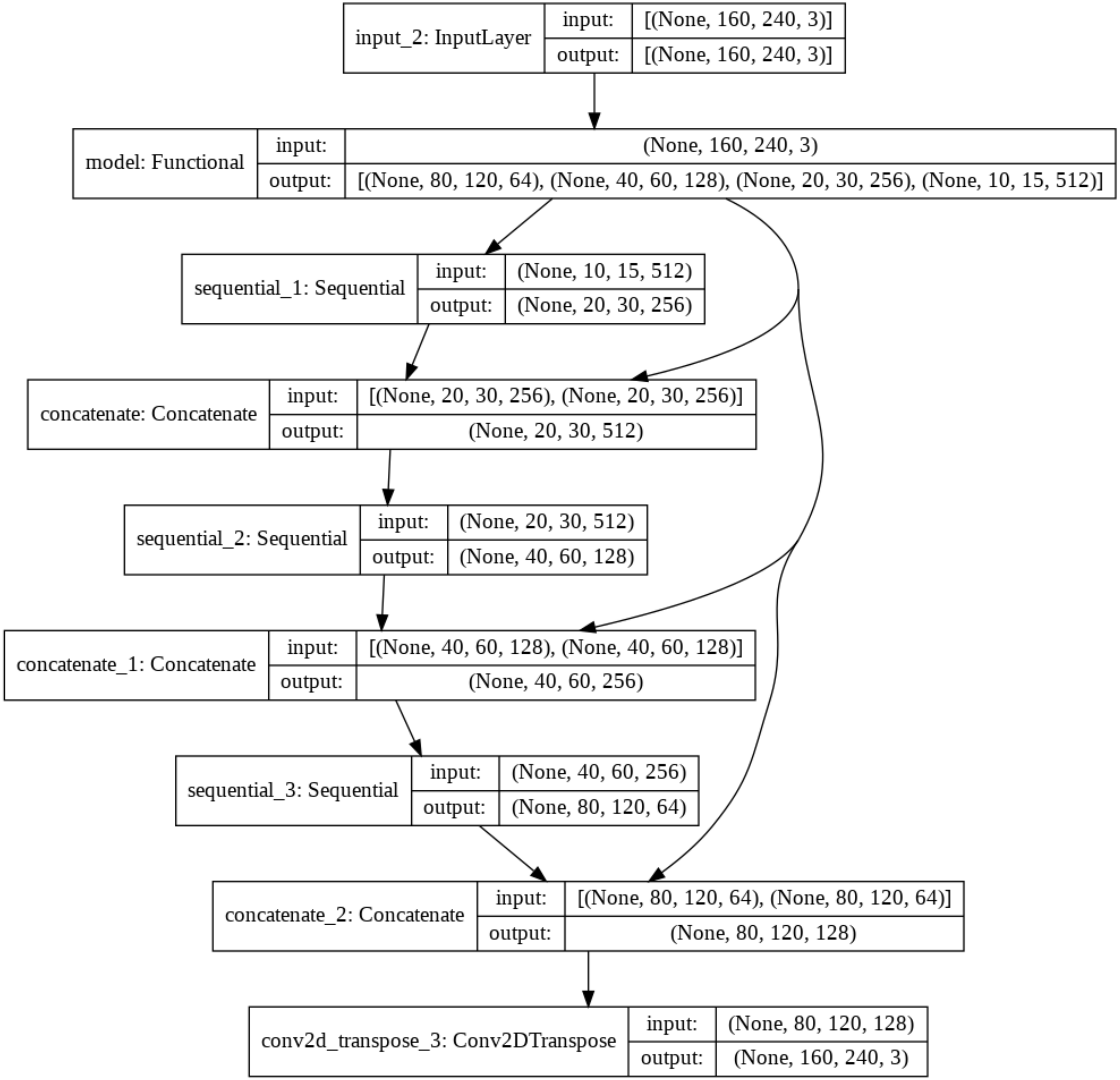
Model Training - CNN

Downsampling

- 4 Convolution layers:
  - Kernel size = 2,
  - Padding = Same
- MaxPooling layers (2X2)
- No batchnormalization

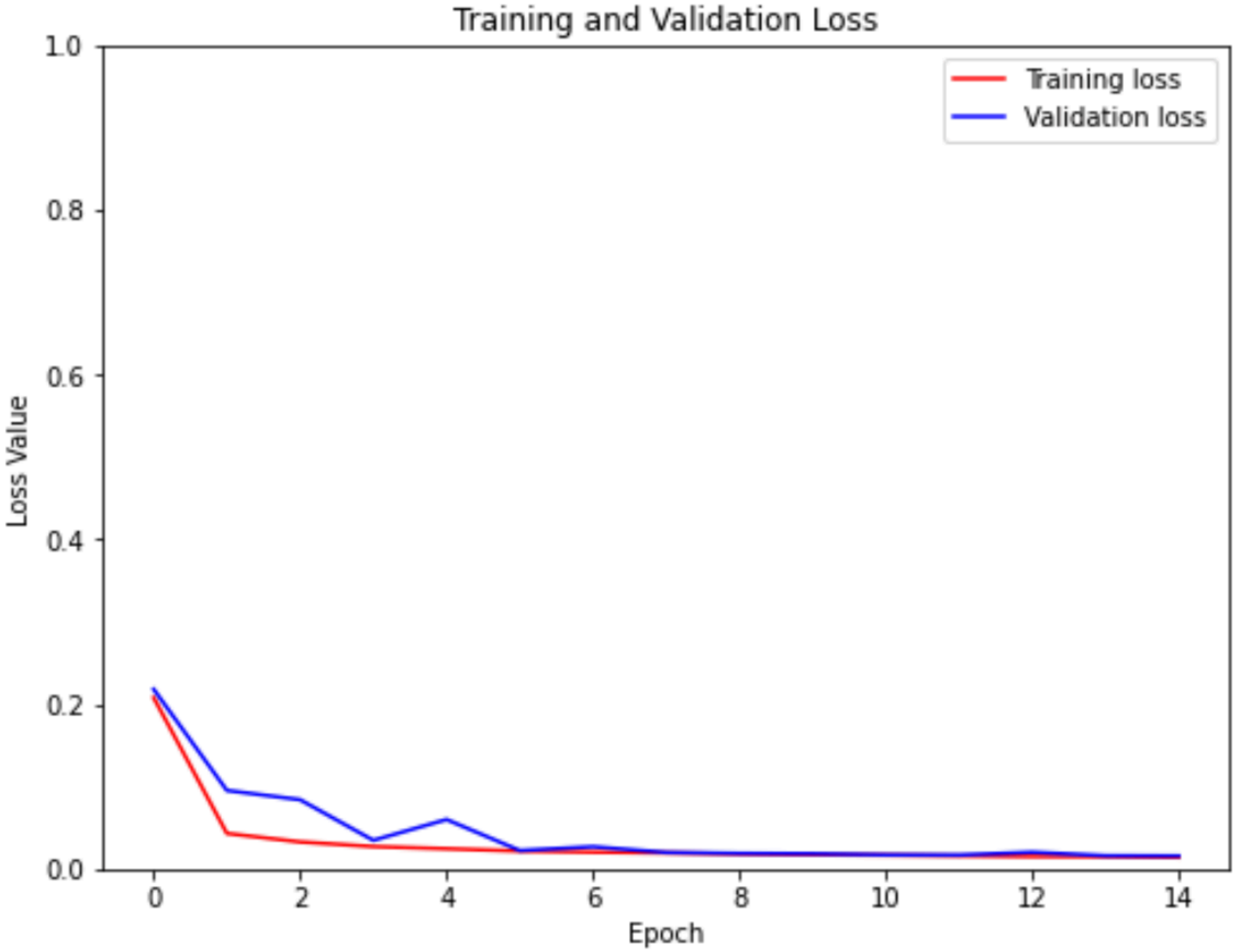
Upsampling

- Using Tensorflow Pix2Pix example
- 3 upsampling steps


input_2: InputLayer | input: | [(None, 160, 240, 3)]
output: | [(None, 160, 240, 3)]
model: Functional | input: | (None, 160, 240, 3)
output: | [(None, 80, 120, 64), (None, 40, 60, 128), (None, 20, 30, 256), (None, 10, 15, 512)]
sequential_1: Sequential | input: | (None, 10, 15, 512)
output: | (None, 20, 30, 256)
concatenate: Concatenate | input: | [(None, 20, 30, 256), (None, 20, 30, 256)]
output: | (None, 20, 30, 512)
sequential_2: Sequential | input: | (None, 20, 30, 512)
output: | (None, 40, 60, 128)
concatenate_1: Concatenate | input: | [(None, 40, 60, 128), (None, 40, 60, 128)]
output: | (None, 40, 60, 256)
sequential_3: Sequential | input: | (None, 40, 60, 256)
output: | (None, 80, 120, 64)
concatenate_2: Concatenate | input: | [(None, 80, 120, 64), (None, 80, 120, 64)]
output: | (None, 80, 120, 128)
conv2d_transpose_3: Conv2DTranspose | input: | (None, 80, 120, 128)
output: | (None, 160, 240, 3)

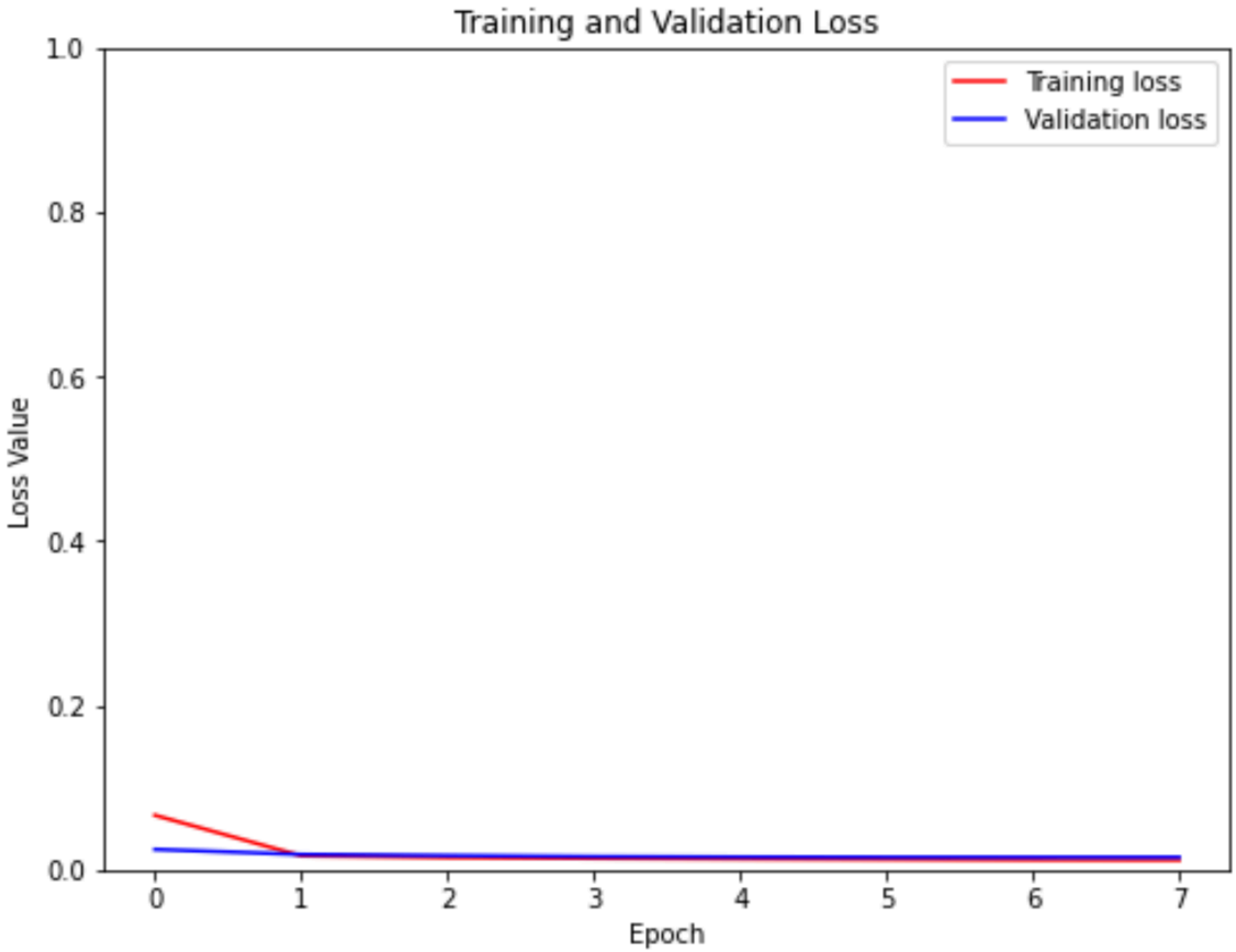
Modelling Results

Self-Built CNN

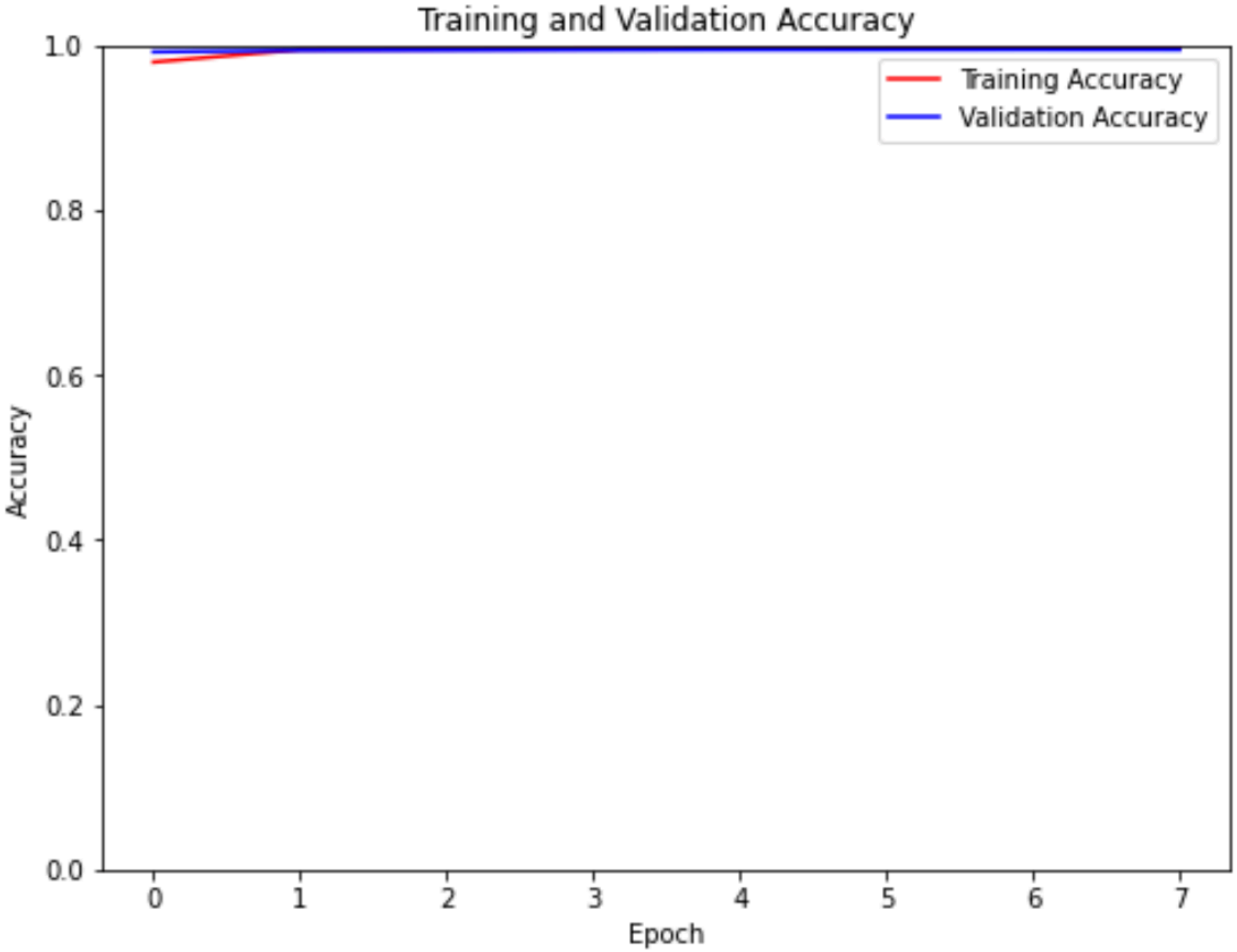


Modelling Results

MobileNetV2



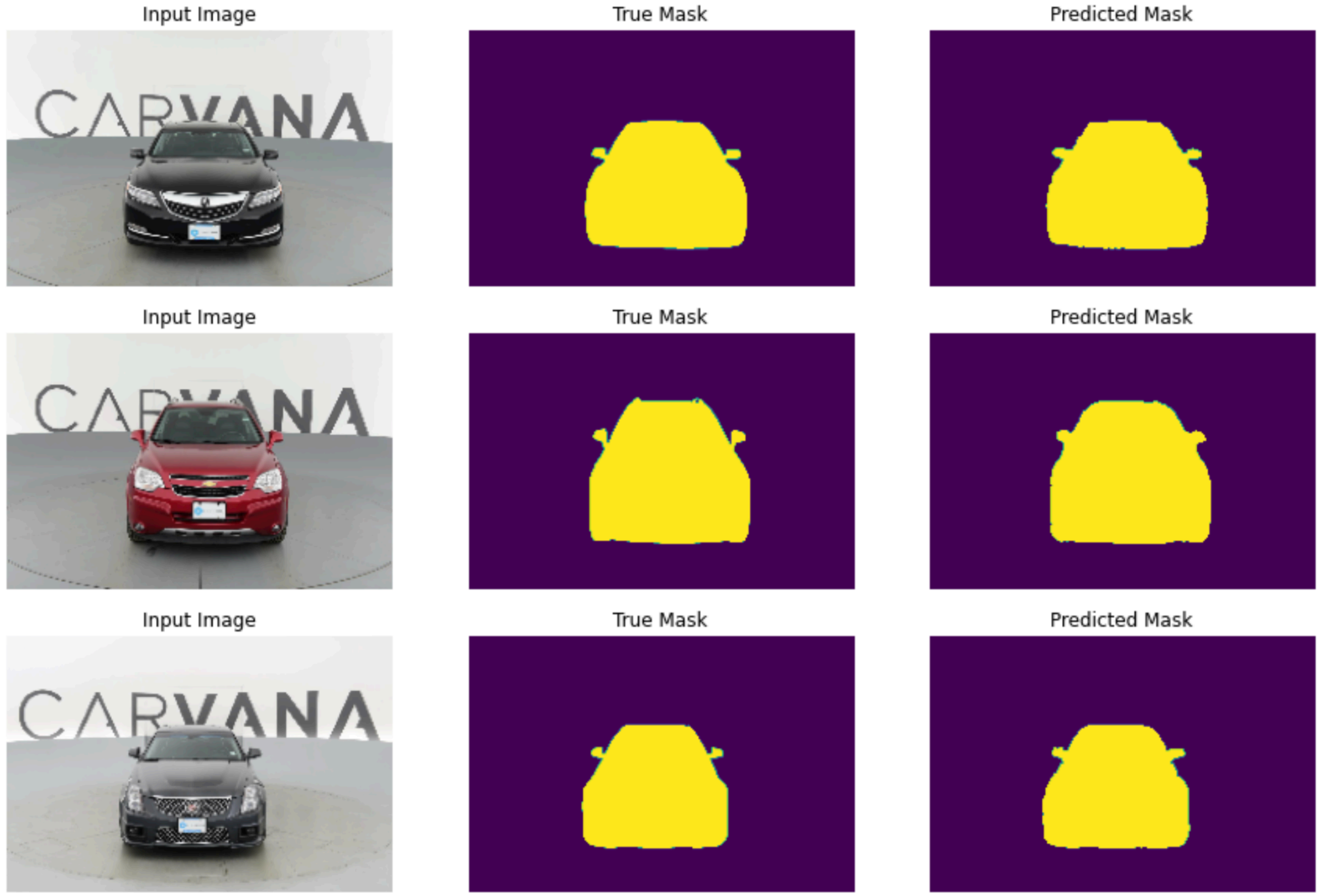
Modelling Results



- Both reached an accuracy of 99.4%
- However, MobilenetV2 was able to reach that level in 7 epochs, instead of 15 for the Self-built CNN

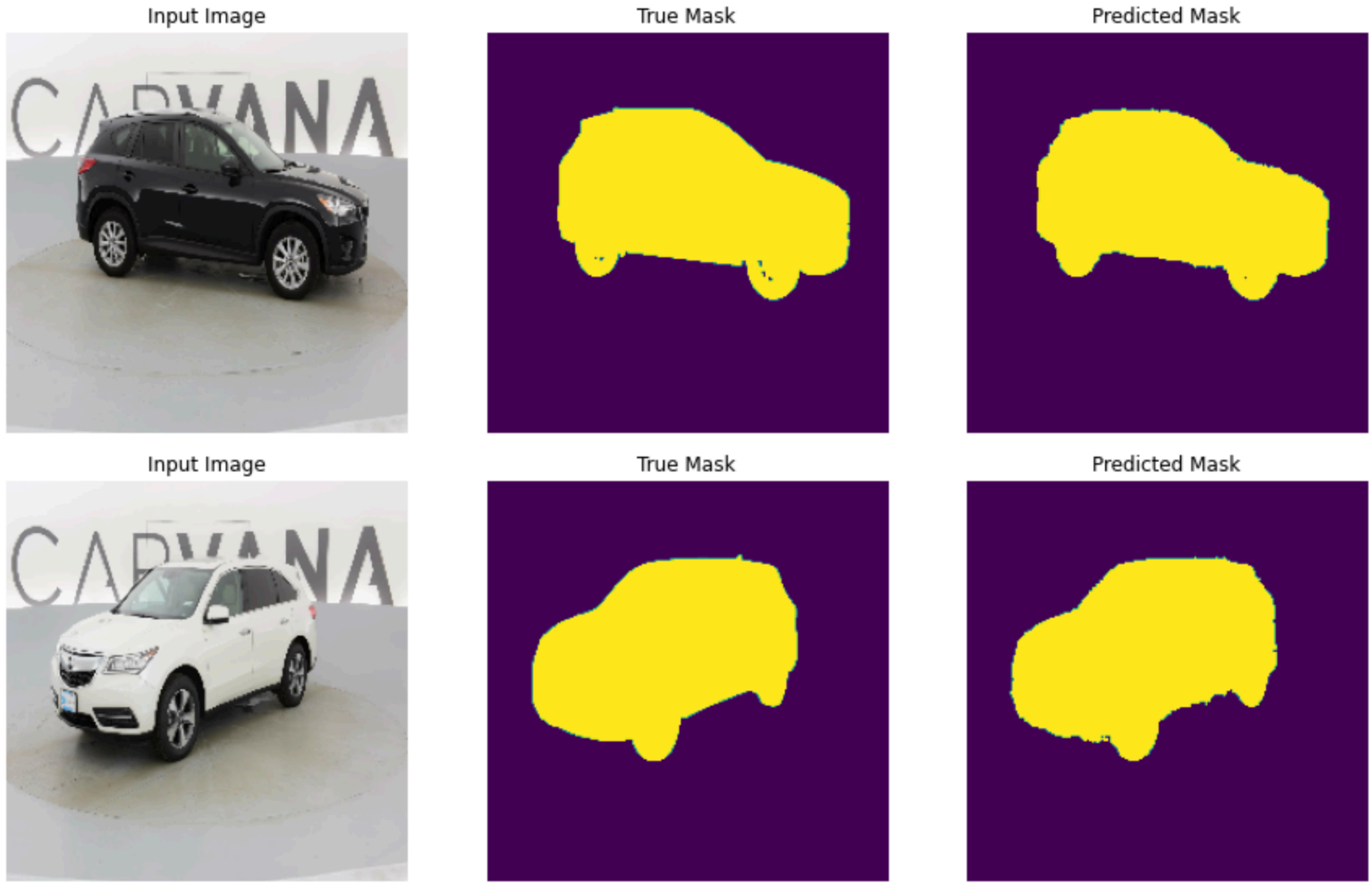
Predictions

Self-Built CNN

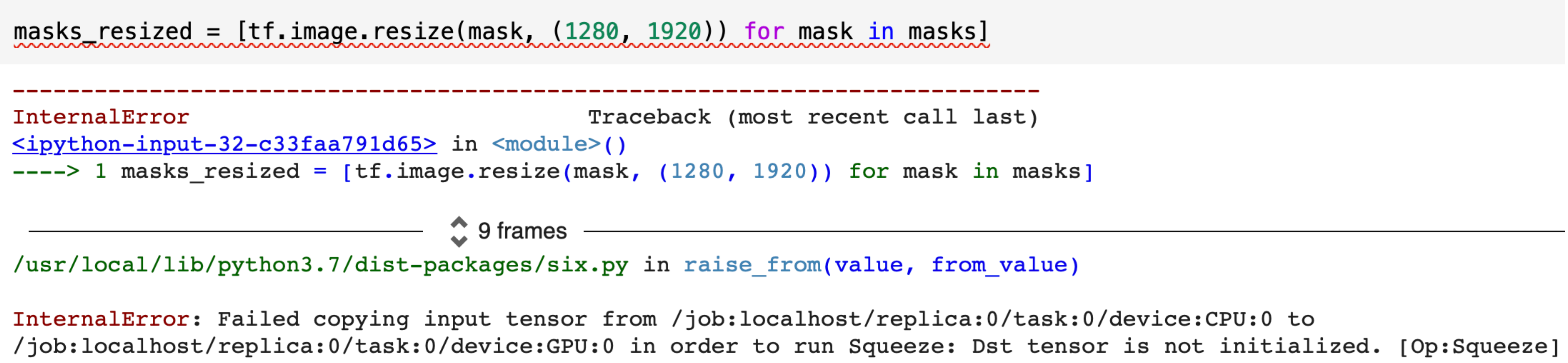


Predictions

MobileNetV2



Evaluation & Conclusions



```
masks_resized = [tf.image.resize(mask, (1280, 1920)) for mask in masks]
```

```
---------------------------------------------------------------------------
InternalError                             Traceback (most recent call last)
<ipython-input-32-c33faa791d65> in <module>()
----> 1 masks_resized = [tf.image.resize(mask, (1280, 1920)) for mask in masks]
```

```
/usr/local/lib/python3.7/dist-packages/six.py in raise_from(value, from_value)

InternalError: Failed copying input tensor from /job:localhost/replica:0/task:0/device:CPU:0 to
/job:localhost/replica:0/task:0/device:GPU:0 in order to run Squeeze: Dst tensor is not initialized. [Op:Squeeze]
```

- Since the model is trained on images of size 160 X 240, we had to resize the predicted images back to 1280 X 1920

- This turned out to be a computational challenge as Google Colab and Tensorflow would crash after resizing 1000 images

- The function to change the masks to Run-Length-Encoding was also not efficient, and was not able to cope with the large image size and quantity