CSI 4133 FINAL PRESENTATION

Aiden Stevenson Bradwell

November 28th, 2021

300064655 || abrado60@uottawa.ca

University of Ottawa, Ottawa, Ontario

PART B: HAND TRACKING

Goal: Tracking markerless hand in a video by using object detection and object tracking.

External Code

- https://techvidvan.com/tutorials/hand-gesture-recognition-tensorflow-opencv/
  - Provided pretrained hand-recognition TensorFlow network, implemented through Google Mediapipe
    - Taken network
    - Taken stages to get network to return hand bone locations
  - No author information available
  - No git information available.

Method Steps

- Input video and output video handlers
- Initialize 2 empty lists, to store previous hand-positions
- For each frame grabbed:
- Convert frame to RBG
- Using pretrained hand-detection **mp_hand_gesture,** detect hands in frame
- Detect palm of hand, and add to appropriate hand-list
- Add point to curve of detected hand
- Draw curve of previous palm-locations
  - Limit size of these lists to 75 data-points per hand
  - Two points are considered sequential if they are within 75px of each other
- Display and Save Frame

Run-time Example

PART A: DIGIT DETECTION

Goal: Creating a method for detecting digits on series of images

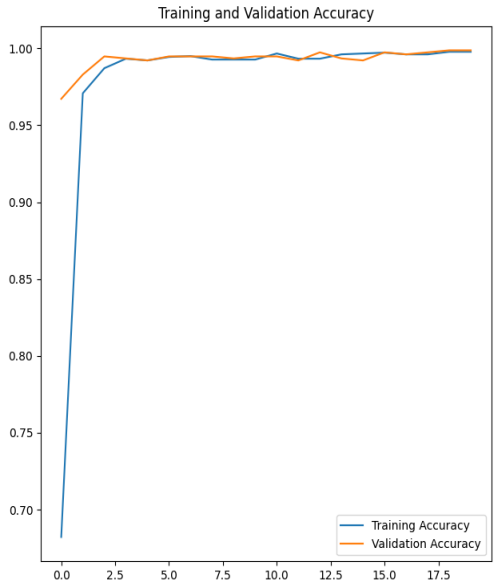
Neural Network Structure

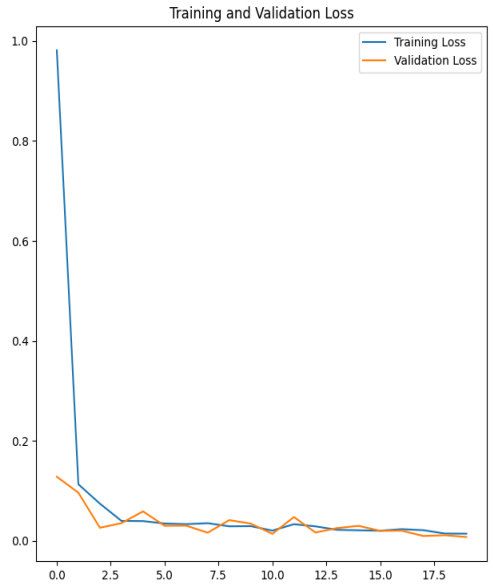
Class	# of Training Images
Zero	1368
One	588
Two	144
Three	109
Four	105
Five	184
Six	119
Seven	324
Eight	558
Nine	493

Layer	Specific Type	Settings
0	Random Flip	
1	Random Rotation	
2	Random Zoom	
3	Rescaling	
4	2D Convolutional	• 16 filters, 3 kernel size • Relu activation
5	2D Max Pooling	
6	2D Convolutional	• 32 filters , 3 kernel size • Relu activation
7	2D Max Pooling	
8	2D Convolutional	• 64filters , 3 kernel size • Relu activation
9	2D Max Pooling	
10	Dropout	• Rate of 0.2
11	Dense Layer	• 128 units • Relu activation
12	Dense Layer	• 10 units • Relu activation

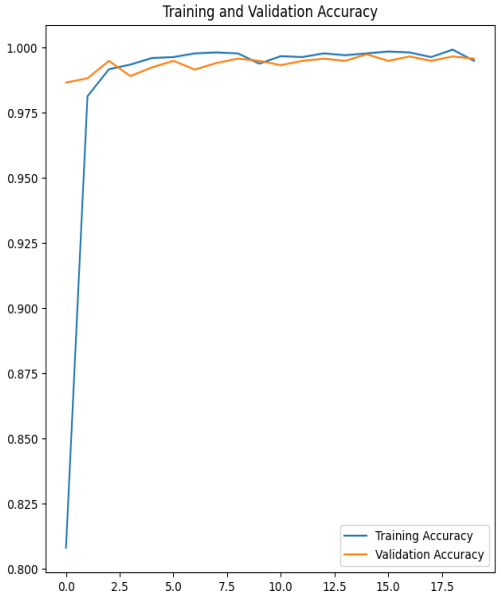
Neural Network Efficiency

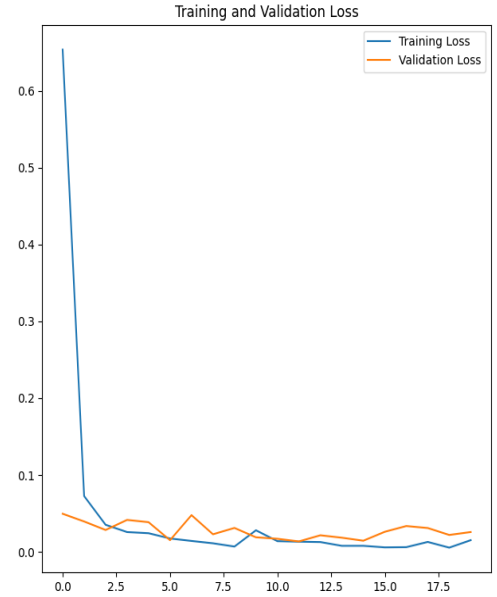
First Training Iteration





Second Training Iteration

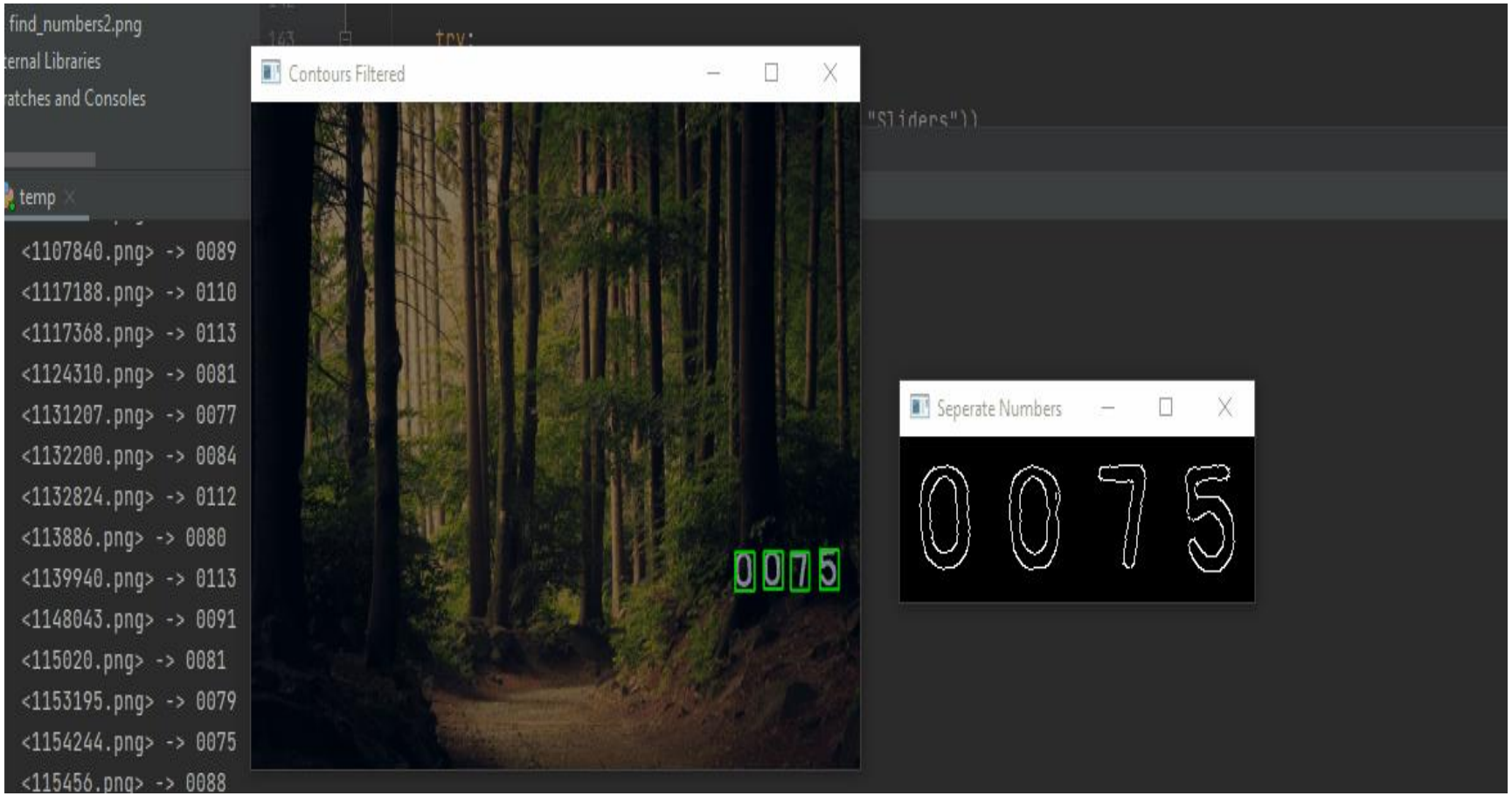




Method Steps

- Request image name and open as frame
- Gaussian Blur image with a kernel of (3, 3)
- convert image to grayscale
- Run canny edge detection on image with low_thresh=177 and high_thresh=204
- Detect external contours on canny image with cv2.RETR_EXTERNAL and cv2.CHAIN_APPROX_NONE
- Filter 4 largest contours by area
- Sort contours by furthest left
- For each contour found
    - Separate the contour's bounding box from the original image
        - Gaussian Blur image with a kernel of (5, 5)
        - Convert image to Grayscale
        - Run canny edge detection with a low_thresh=177 and high_thresh=204
        - Convert image to RBG
        - Resize image to 70w x 84h
        - Run TensorFlow Neural Network on image
        - return strongest prediction of number value
- Output combined 4 strongest predictions in terminal

Run-time Example



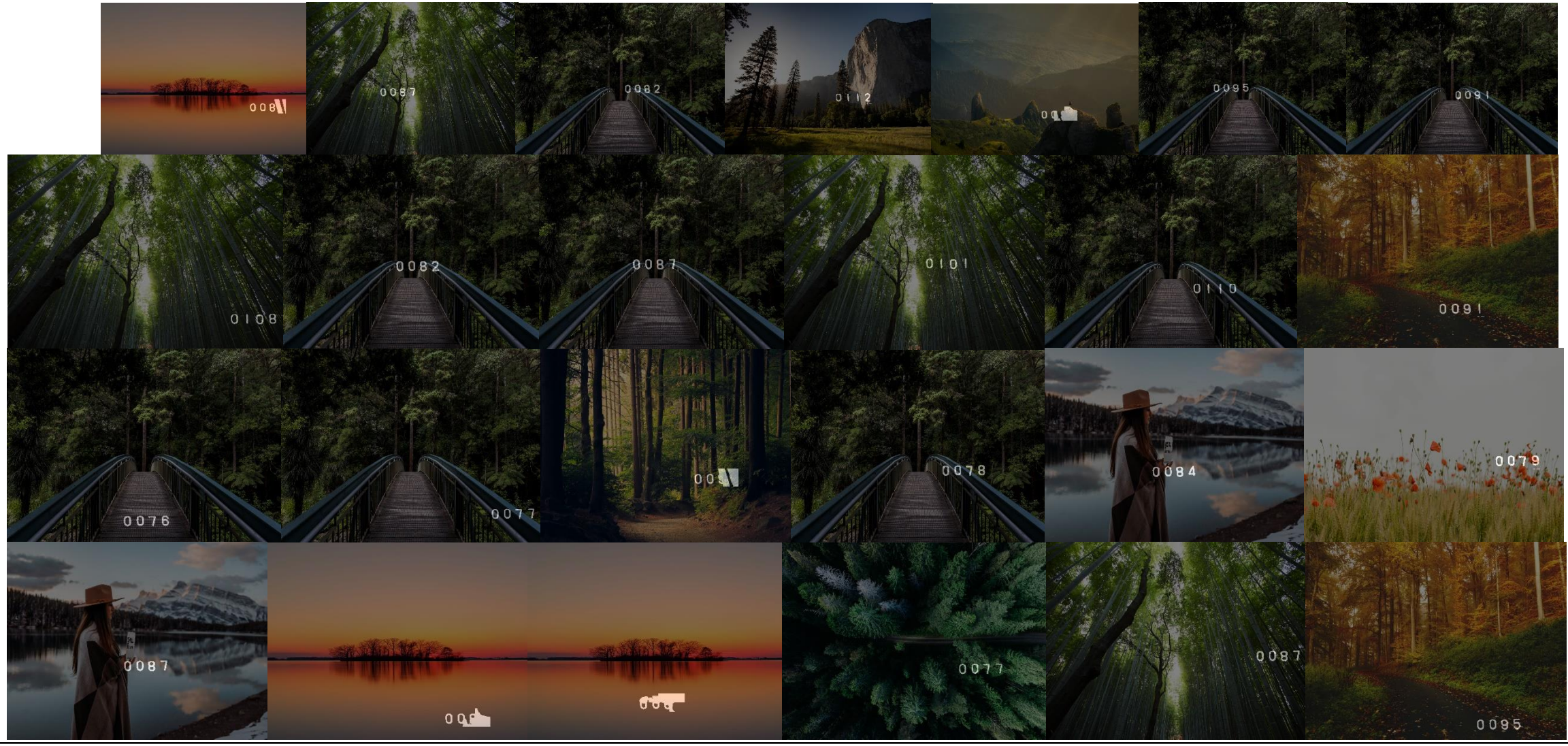
Limitations

- Correctly identified all 4 numbers
  - **1294/1324 (97.734%)**

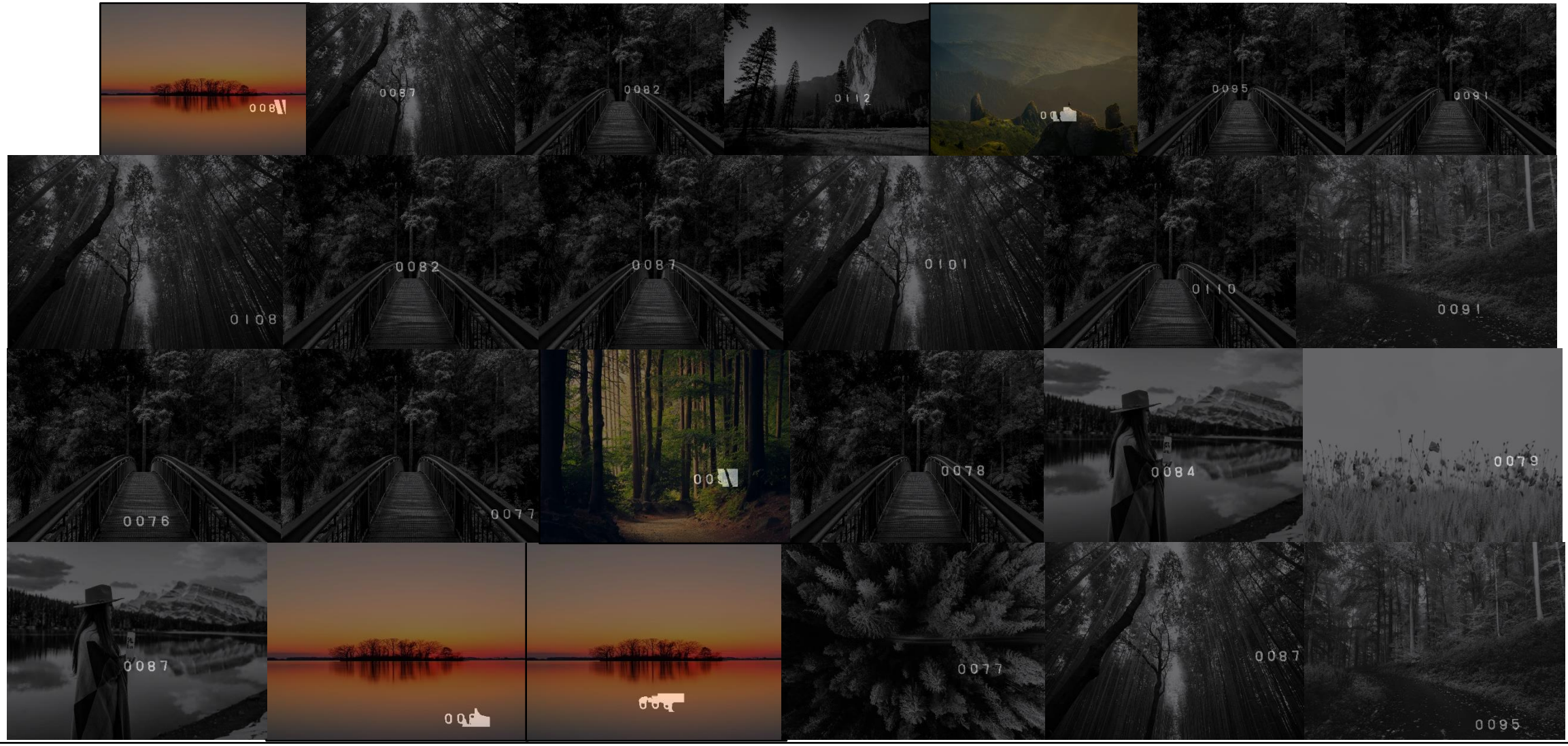
- Correctly identified all 4 non-corrupted numbers
  - **1300/1324 (98.187%)**

With the accuracy of the neural network, it appears that the error lies in incorrectly canny-edged images. Different threshvalues or a blurring method may provide an increase in accuracy

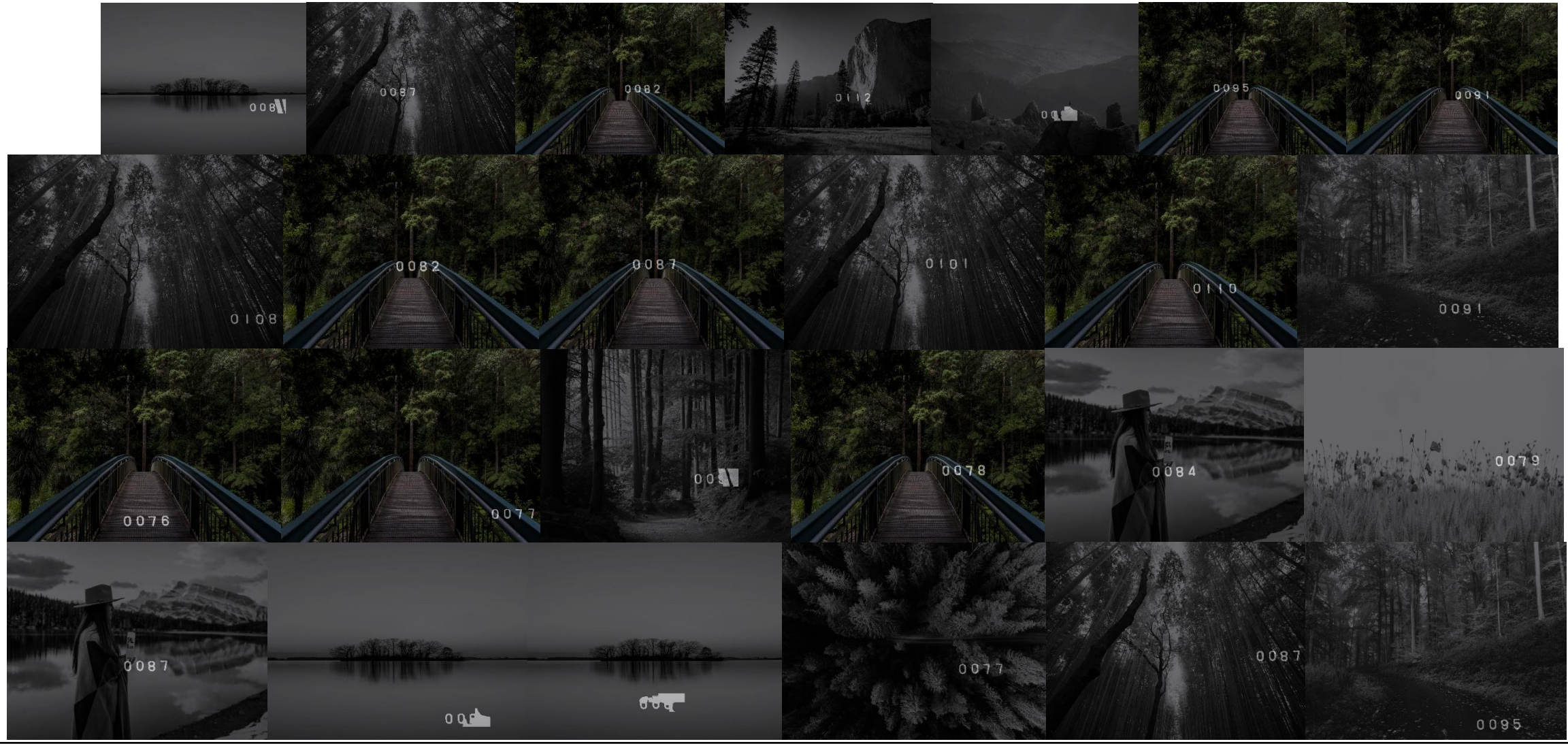
Images of Difficulty



Images of Difficulty



Images of Difficulty



THANK YOU!

Feel free to ask any questions.

References

1. https://www.tensorflow.org/tutorials/images/classification

- TensorFlow. (2021, November 11). *Image classification : Tensorflow Core*. TensorFlow. Retrieved November 29, 2021, from https://www.tensorflow.org/tutorials/images/classification.

1. https://techvidvan.com/tutorials/hand-gesture-recognition-tensorflow-opencv/

- *Real-time hand gesture recognition using tensorflow & opencv*. TechVidvan. (2021, July 21). Retrieved November 29, 2021, from https://techvidvan.com/tutorials/hand-gesture-recognition-tensorflow-opencv/.