UMKC
UNIVERSITY OF MISSOURI
KANSAS CITY

Self Driving Car in 3D game world using Neural Network

Group Name: Techletes
Andrew Abbott
Kapil Sharma
Gopesh Thakur
Jinal Patel

CS5542 – CS5590/490: Python and Deep Learning Programming

Content

- Introduction
- Overcoming Challenges
- Background and Related Works
- Architecture and Methodology
- Data, Code Snippet, and Results
- Conclusion
- References
- Q & A

Introduction

- Every year, traffic accidents account for 2.2% of global deaths. That stacks up to roughly 1.3 million a year — 3,287 a day. On top of this, some 20–50 million people are seriously injured in auto-related accidents each year. The root of these accidents? Human error.

- From distracted driving to drunk driving to reckless driving to careless driving, one poor or inattentive decision could be the difference between a typical drive and a life-threatening situation. But what if we could neutralize human error from the equation?

- “Autonomous cars are no longer beholden to Hollywood sci-fi films” — Elon Musk, the founder of Tesla Inc. and SpaceX believes within a decade, self-driving cars will be as common as elevators.

Overcoming Challenges

- Train an end-to-end deep learning model that would let a car drive by itself around the track in a driving simulator. It is a supervised regression problem between the car steering angles and the road images in real-time from the cameras of a car.

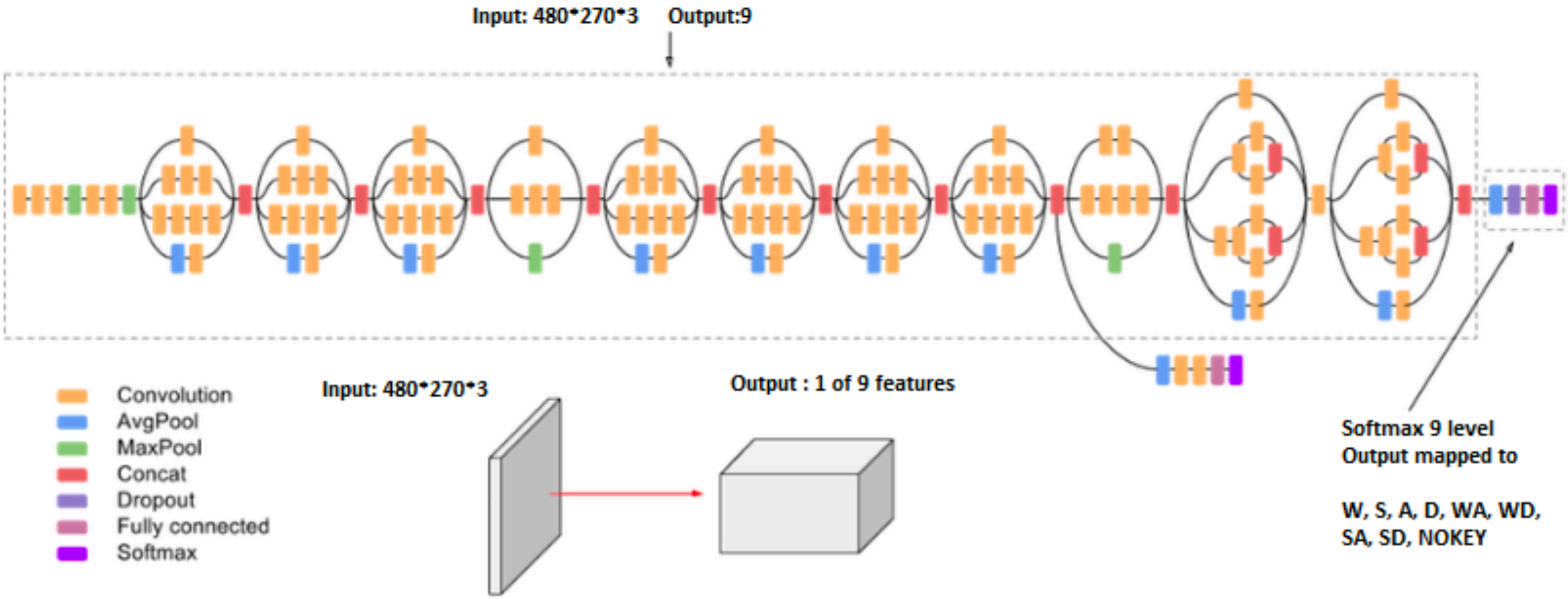
- Due to the large computational requirements extended data collection and training is needed to accurately model this system.

- Different environments such as tracks, day or night view, First person or Third Person views, obstructions etc. makes it hard to train the model.

Related Works

- According to a fascinating report from **Bloomberg Technology**, scientists at Darmstadt University of Technology and Intel Labs worked out a way extract visual information from the game in 2016, and various companies are using the game for research.

Architecture and Methodology



Input: 480*270*3 Output:9

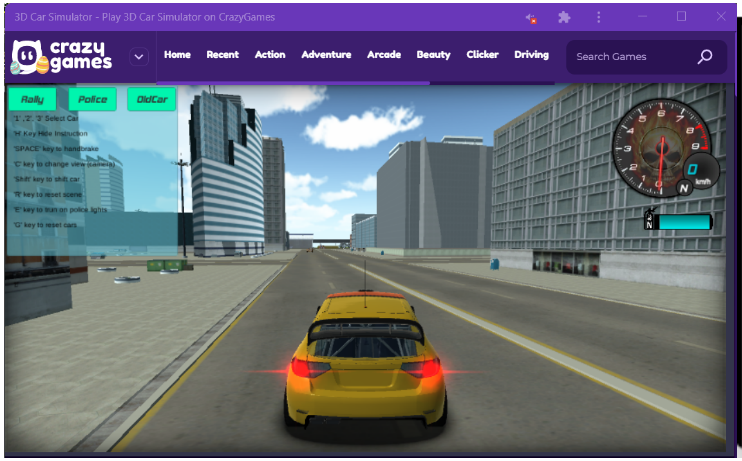
Input: 480*270*3

Output : 1 of 9 features

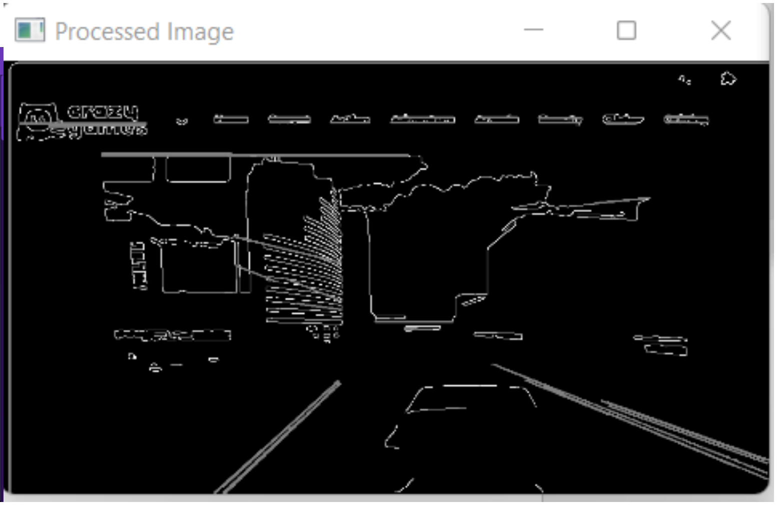
Softmax 9 level
Output mapped to

W, S, A, D, WA, WD,
SA, SD, NOKEY

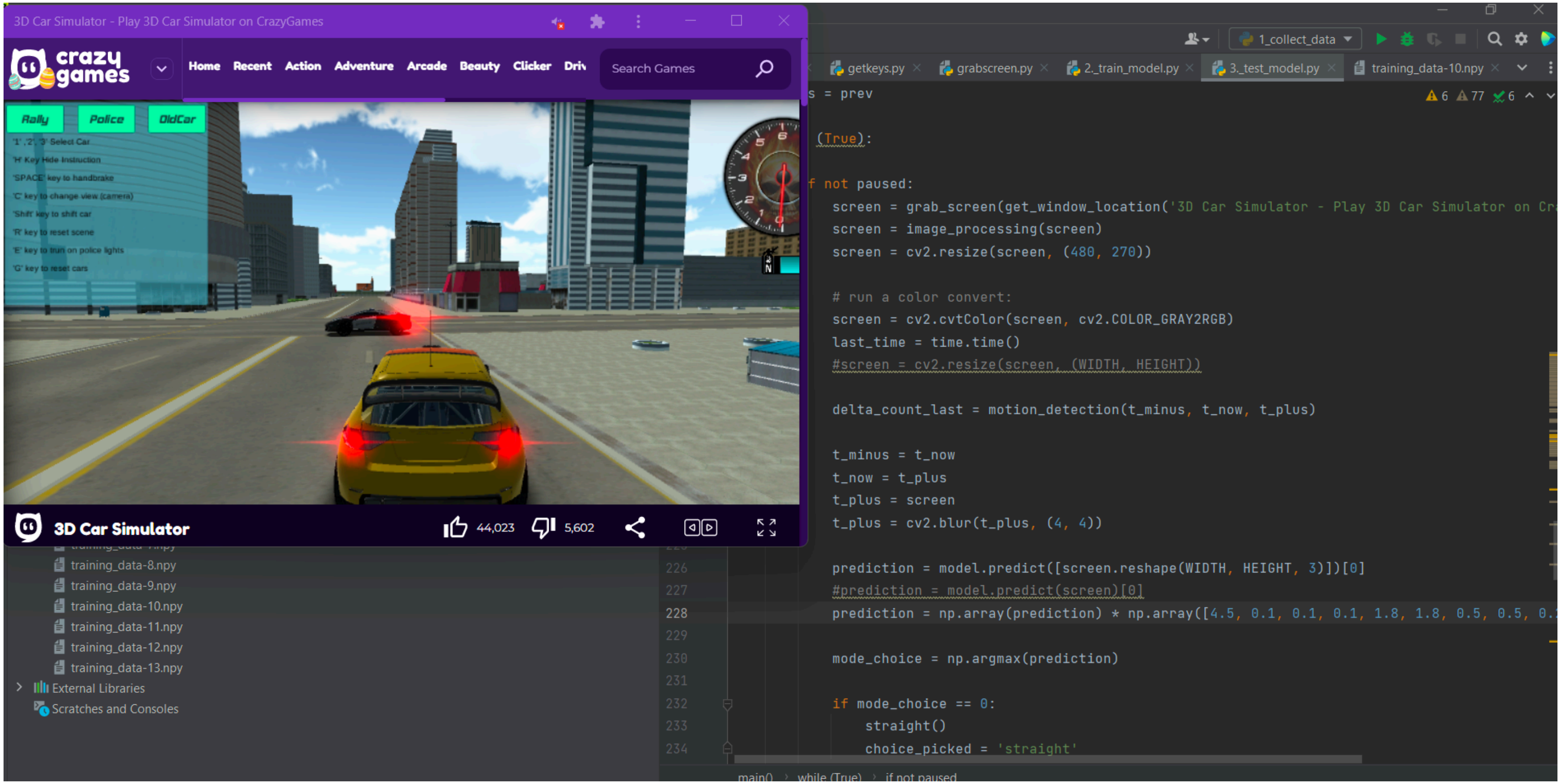
Image Pre-Processing



'1' ,'2', '3' Select Car
'H' Key Hide Instruction
'SPACE' key to handbrake
'C' key to change view (camera)
'Shift' key to shift car
'R' key to reset scene
'E' key to trun on police lights
'G' key to reset cars



Code Snippet & Result



```
s = prev

 (True):

f not paused:
    screen = grab_screen(get_window_location('3D Car Simulator - Play 3D Car Simulator on Cr
    screen = image_processing(screen)
    screen = cv2.resize(screen, (480, 270))

    # run a color convert:
    screen = cv2.cvtColor(screen, cv2.COLOR_GRAY2RGB)
    last_time = time.time()
    #screen = cv2.resize(screen, (WIDTH, HEIGHT))

    delta_count_last = motion_detection(t_minus, t_now, t_plus)

    t_minus = t_now
    t_now = t_plus
    t_plus = screen
    t_plus = cv2.blur(t_plus, (4, 4))

    prediction = model.predict([screen.reshape(WIDTH, HEIGHT, 3)])[0]
    #prediction = model.predict(screen)[0]
    prediction = np.array(prediction) * np.array([4.5, 0.1, 0.1, 0.1, 1.8, 1.8, 0.5, 0.5, 0.

    mode_choice = np.argmax(prediction)

    if mode_choice == 0:
        straight()
        choice_picked = 'straight'
```

Conclusions and Future Outlook

- Our first approach was to draw the lanes on the screen and drive based on the lanes found on the screen. If both lanes are on the left side, we need to steer left and if both lanes are on the right side, we need to steer right and if one lane is on the left and one is on the right, we need to drive straight.
- Second approach was to collect the raw images and keyboard strokes (we tried with actual gaming controller as well) to feed into the model for training and in actual game play we used model to project the output as one of the key strokes.
- Third approach was to mix of first two approaches where we collected lanes data, edges in image and keyboard strokes to feed into the mod . This approach gave us less noisy data to train the model and predict the better output.
- Due to time and limited computational resources, we were unable to train the model sufficiently which led to inaccurate predictions most of the time.
- As a future work, we will try to train the model with enough data to predict better output.

References

- 1] H. Kinsley, “Python Plays GTA V,” Python Programming Tutorials. Available: https://pythonprogramming.net/next-steps-python-plays-gta-v/
- [2] G. Research, “Colaboratory,” Google. Available: https://research.google.com/colaboratory/faq.html
- [3] K. Team, “Keras documentation: About Keras,” Keras. Available: https://keras.io/about/
- [4] Prabhu, “Understanding of Convolutional Neural Network (CNN) — Deep Learning,” Medium, 21-Nov-2019. Available: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148
- [5] K. Patel, “MNIST Handwritten Digits Classification using a Convolutional Neural Network (CNN),” Medium, 01-Dec-2019. Available : https://towardsdatascience.com/mnist-handwritten-digits-classification-using-a-convolutional-neural-network-cnn-af5fafbc35e9
- [6] B. Ramsundar and R. B. Zadeh, “TensorFlow for Deep Learning,” O’Reilly Online Learning. [Online]. Available: https://www.oreilly.com/library/view/tensorflow-for-deep/9781491980446/ch04.html
- [7] Miro.medium.com. 2020 Available at: https://miro.medium.com/max/1250/1*4ZEDRpFuClpUjNgjDdT2Lg.png
- [8] M. Bojarski, D. D. Test, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, “End to End Learning for Self-Driving Cars,” arXiv.org, 2016. Available: https://arxiv.org/abs/1604.07316
- [9] “Maxima and minima,” Wikipedia. Available : https://en.wikipedia.org/wiki/File:Extrema_example_original.svg
- [10] A. Kathuria, “Intro to optimization in deep learning: Gradient Descent,” Paperspace Blog, 02-Dec-2019. Available: https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/
- [11] V. Bushaev, “Adam — latest trends in deep learning optimization.,” Medium, 24-Oct-2018. Available : https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c.
- [12] J. Redmon and A. Farhadi, YOLO: Real-Time Object Detection. Available: https://pjreddie.com/darknet/yolo/.
- [13] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” arXiv.org, 08-Apr-2018. Available : https://arxiv.org/abs/1804.02767.
- [14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, You Only Look Once: Unified, Real-Time Object Detection — IEEE Conference Publication. Available: https://ieeexplore.ieee.org/document/7780460.