Real Time Content Moderation

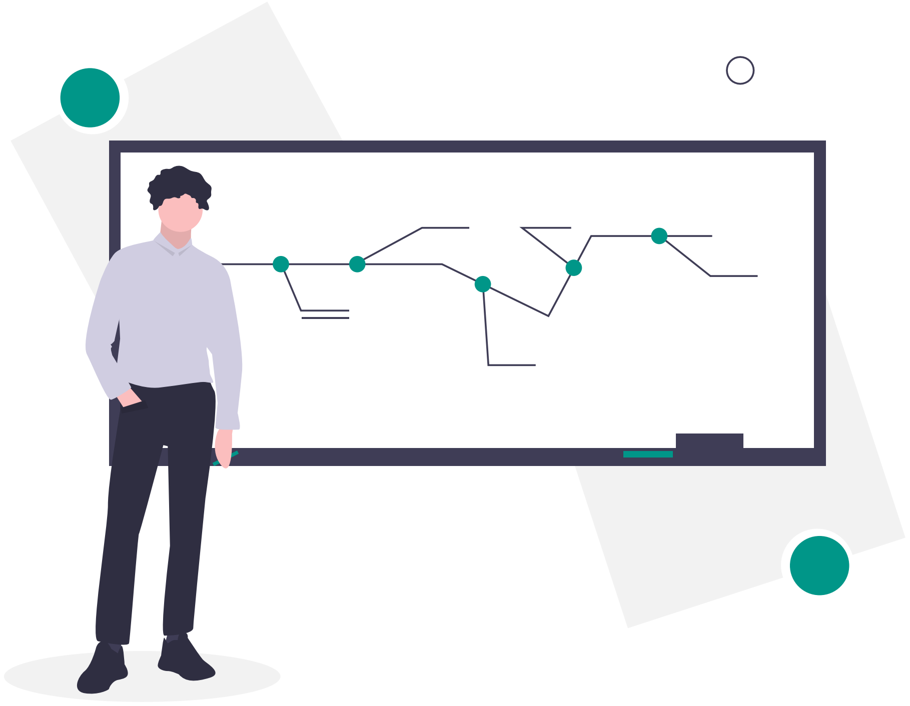
Group Members

Anuj Dalvi

Animesh Srivastava

Cyrus Britto

Contents



Problem Statements

In the recent years, there has been a massive increase in the generation and consumption of media content which are freely available to different types of users. There might be scenarios where the inappropriate content, which is not suitable for a specific category of users like kids, are present offline on the user's device. Due to the limited or no restrictions on access of such content, there is always a huge risk that underage users might be exposed to inappropriate content or pornographic content. This effects young minds in a negative way and can cause various mental disorders.

The traditional content censoring systems work by the technique of blocking URLs of unsafe websites which is both inefficient and can also be fooled easily by the use of proxies. In order to prevent users from using alternative ways to access the explicit content, there is a need of an application which is monitoring the type of content accessed.

Existing System

- URL Blacklisting
- Safe Search – Bing, Google, Yahoo (Same as URL Blacklisting)

Proposed System

Our Real time content moderation will use deep neural networks for analyzing different labels according to the body parts revealed. We are opting for multi-label classification of these images. We are further compressing this model using deep compression to reduce the space required by the application without affecting the efficiency of the model.

The application runs in the background at the scheduled time and scans for explicit images continuously for that duration and censors them.

This is implemented on the local drive itself and can be further developed for mobile platforms.

H/W requirements

Intel Core i7-8700 Processor:

As we are working in Deep Learning Domain, we need a high computation power. As the clock speed and cores of in is higher at its range, it will be very helpful.

128GB SSD + 1TB HDD:

Model predicts accurately only when the dataset is strong and large enough. The 1TB HDD will be very useful for storing a large amount of Whereas, 128GB SSD will be useful to perform the read-write operation fast.

16GB DDR4 RAM:

During the training of model, the data is stored in the primary memory (RAM). Larger the memory, faster the execution.

12GB NVidia GeForce GTX 1080Ti Graphic Card:

The training of models, requires a large amount of computing power, where GPU out performs the processor in terms of training the models.

S/W requirements

Linux (Ubuntu or any Debian-base): Due to wide availability of open source libraries, and huge community support is the main reason for using Linux Operating System.

Programming Languages (Python, C/C++):

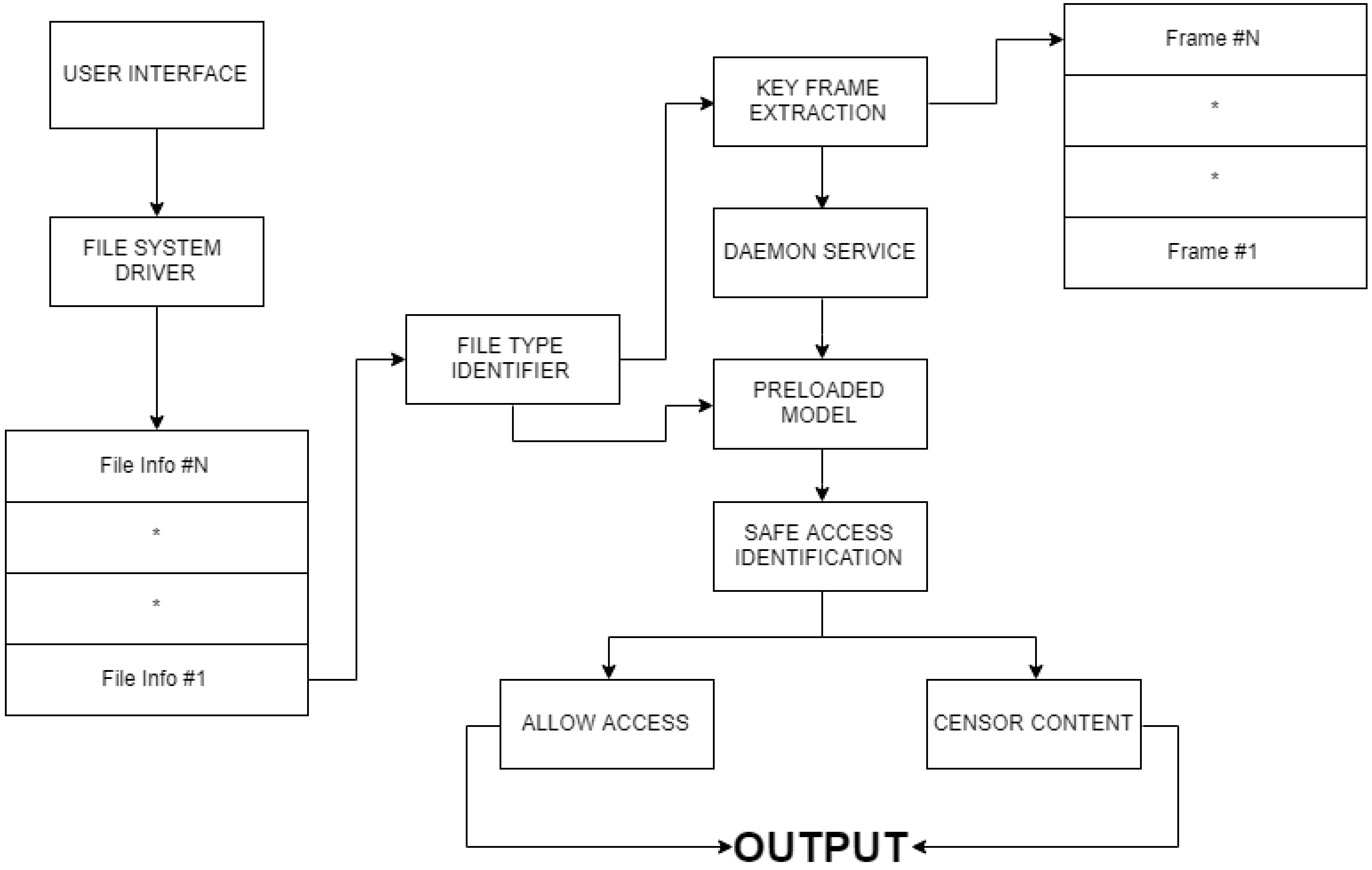
- Python is a widely used programming language for Deep Learning, due to its vast support for deep learning libraries.
- C/C++ is the programming language which will be used for Driver Development.

Libraries (TensorFlow, Keras, OpenCV, Numpy, PIL): These are the most commonly used libraries in the deep learning domain. These libraries make the code simpler to its predefined algorithms.

Software (Anaconda, Microsoft Visual Studio 2017, XAMPP, Sublime Text Editor, Oracle, Virtual Box):

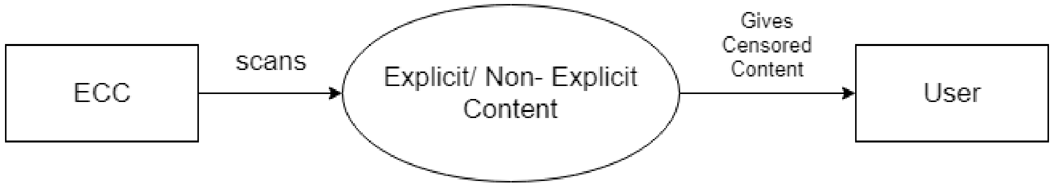
- Anaconda is the package manager for python. It makes easy solving of package dependencies.
- Microsoft Visual Studio 2017 and Sublime Text Editor are the IDE used for general programming.
- XAMPP's functionality of providing web server solution stack will be used for testing of browser extension for web pages.
- Oracle Virtual Box will be used as the testing environment for the developed windows driver.

Design: Block Diagram

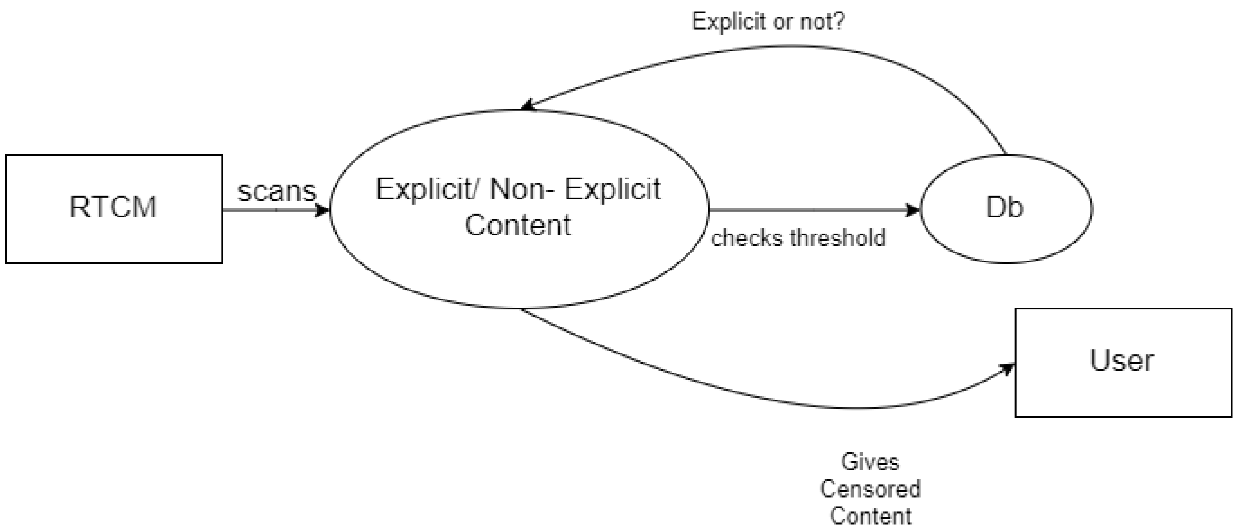


Design -a

Level 0 DFD

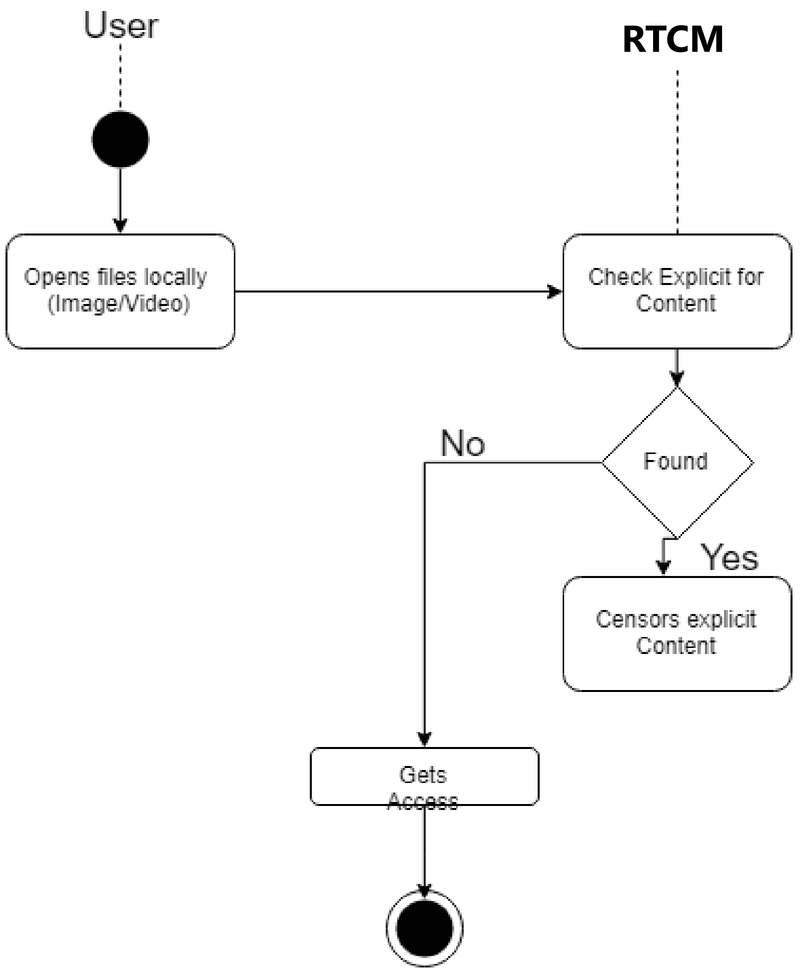


Level 1 DFD

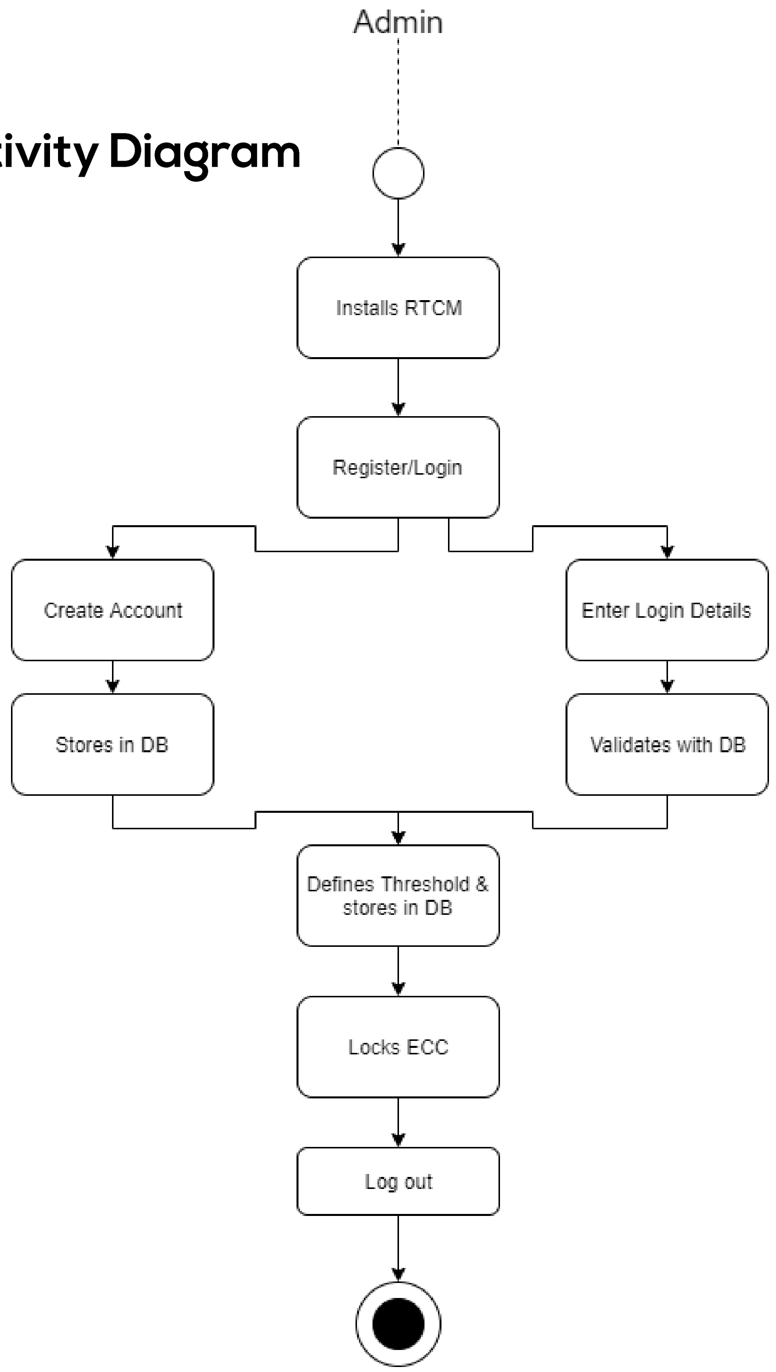


Design-b

User Activity Diagram

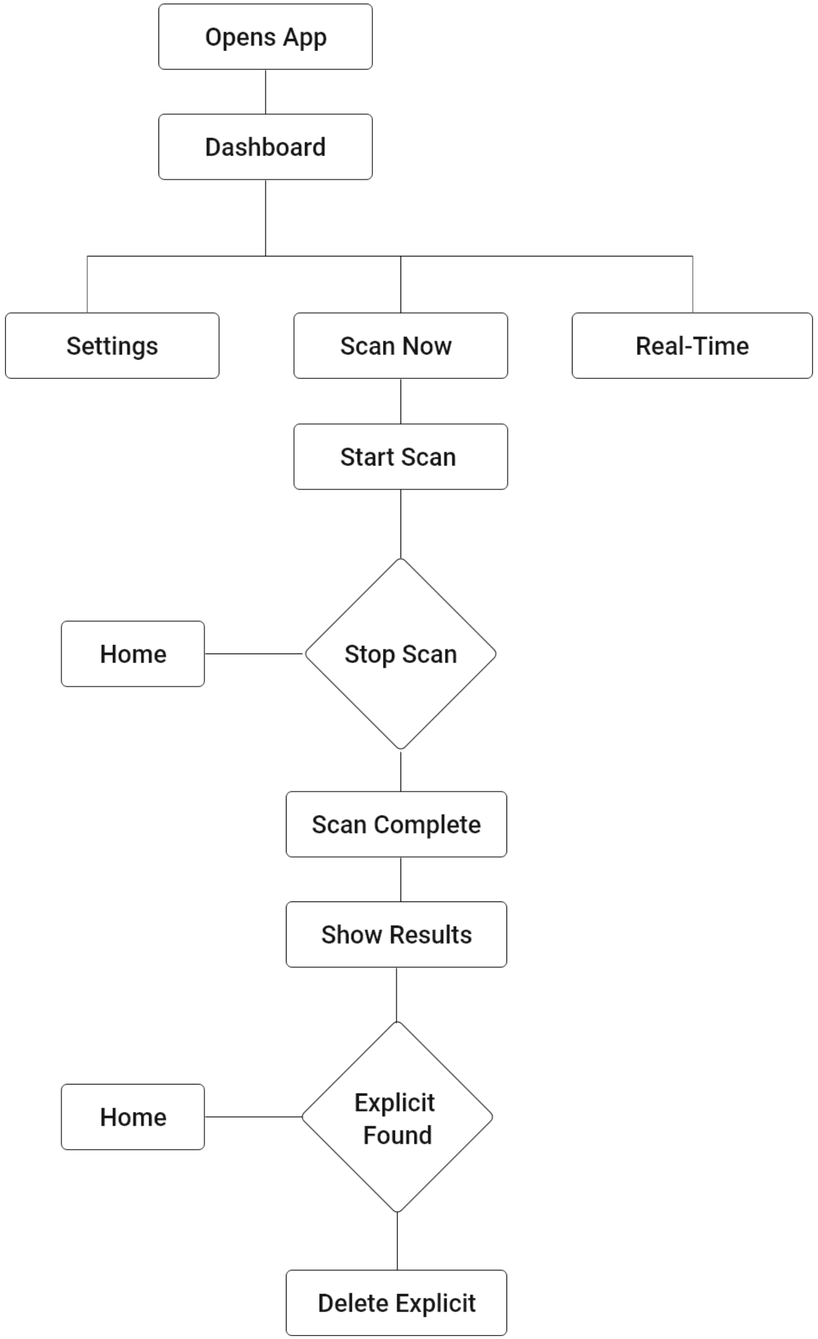


Admin Activity Diagram



Design-b

User Flow Diagram



Classification Models Tested

```
Use tf.cast instead.
Epoch 1/10
2019-06-18 17:23:27.968740: I tensorflow/stream_executor/dso_loader.cc:152] successfully opened CUDA library cublas64_100.dll locally
 - 715s - loss: 0.0851 - acc: 0.9689 - val_loss: 0.0140 - val_acc: 1.0000
Epoch 2/10
 - 691s - loss: 0.0270 - acc: 0.9919 - val_loss: 0.0057 - val_acc: 1.0000
Epoch 3/10
 - 690s - loss: 0.0155 - acc: 0.9956 - val_loss: 0.0021 - val_acc: 1.0000
Epoch 4/10
 - 687s - loss: 0.0156 - acc: 0.9957 - val_loss: 0.0076 - val_acc: 1.0000
Epoch 5/10
 - 686s - loss: 0.0120 - acc: 0.9967 - val_loss: 0.0015 - val_acc: 1.0000
Epoch 6/10
 - 693s - loss: 0.0093 - acc: 0.9971 - val_loss: 3.3543e-04 - val_acc: 1.0000
Epoch 7/10
 - 691s - loss: 0.0075 - acc: 0.9975 - val_loss: 0.0139 - val_acc: 0.9950
Epoch 8/10
 - 687s - loss: 0.0058 - acc: 0.9981 - val_loss: 0.0020 - val_acc: 1.0000
Epoch 9/10
 - 690s - loss: 0.0044 - acc: 0.9988 - val_loss: 0.0041 - val_acc: 0.9975
Epoch 10/10
 - 694s - loss: 0.0086 - acc: 0.9970 - val_loss: 6.8721e-04 - val_acc: 1.0000

(ten) C:\Users\legend698\Desktop\ANIMESH\python>
```

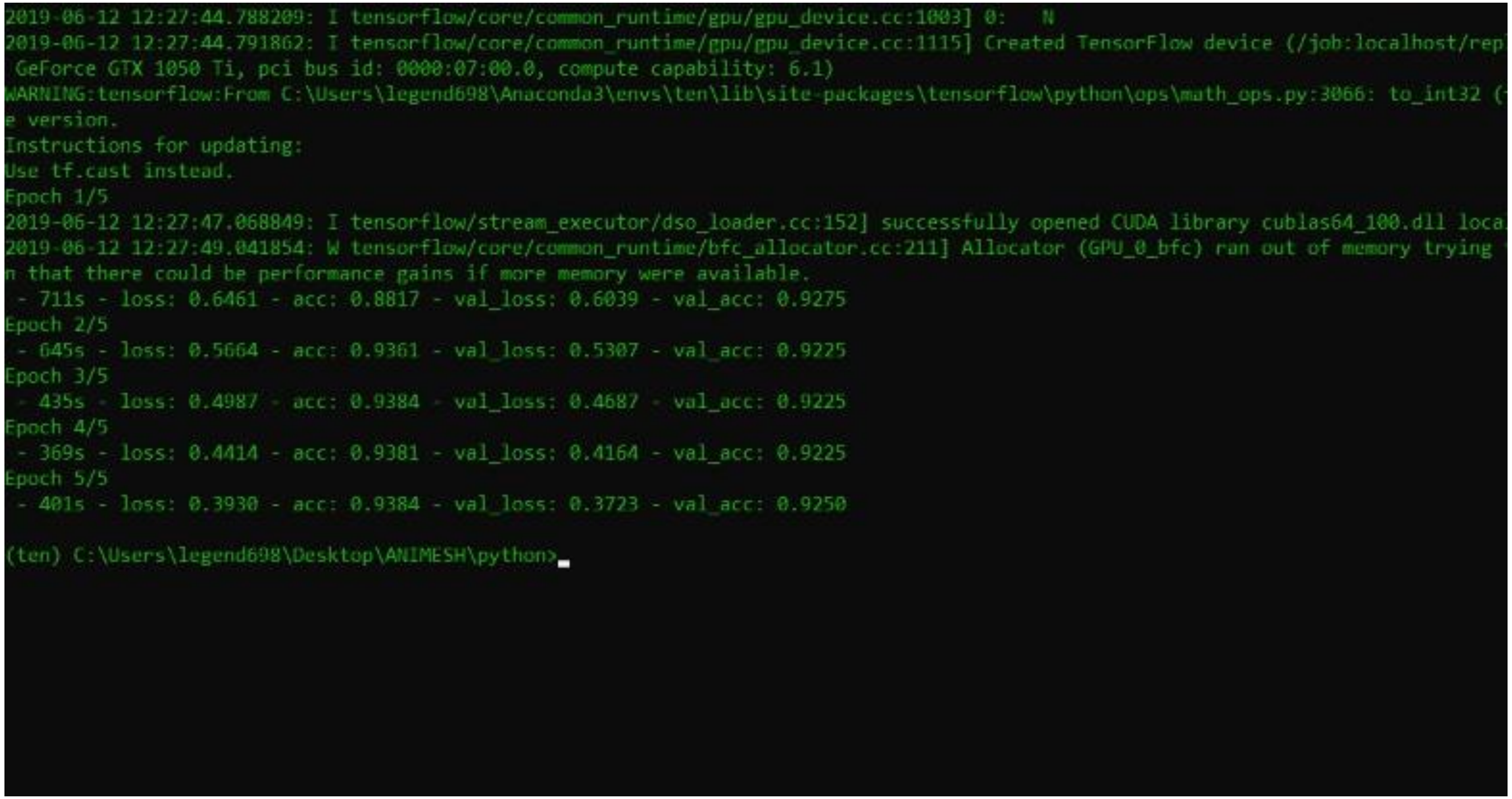
EfficientNetB0 ACC-99.7% LOSS-0.86% OVER 10 EPOCHS, SIZE-46.8mb

Classification Models Tested

```
Instructions for updating:
Use tf.cast instead.
Epoch 1/10
2019-06-15 18:05:34.452046: I tensorflow/stream_executor/dso_loader.cc:152] successfully opened CUDA library cublas64_100.dll locally
 - 652s - loss: 0.1255 - acc: 0.9494 - val_loss: 0.1570 - val_acc: 0.9625
Epoch 2/10
 - 630s - loss: 0.0819 - acc: 0.9682 - val_loss: 0.0390 - val_acc: 0.9850
Epoch 3/10
 - 636s - loss: 0.0666 - acc: 0.9760 - val_loss: 0.0240 - val_acc: 0.9875
Epoch 4/10
 - 635s - loss: 0.0492 - acc: 0.9819 - val_loss: 0.0267 - val_acc: 0.9875
Epoch 5/10
 - 485s - loss: 0.0365 - acc: 0.9878 - val_loss: 0.0207 - val_acc: 0.9925
Epoch 6/10
 - 297s - loss: 0.0333 - acc: 0.9894 - val_loss: 0.0184 - val_acc: 0.9950
Epoch 7/10
 - 296s - loss: 0.0243 - acc: 0.9928 - val_loss: 0.0366 - val_acc: 0.9925
Epoch 8/10
 - 694s - loss: 0.0179 - acc: 0.9949 - val_loss: 0.0358 - val_acc: 0.9900
Epoch 9/10
 - 678s - loss: 0.0167 - acc: 0.9950 - val_loss: 0.0172 - val_acc: 0.9950
Epoch 10/10
 - 302s - loss: 0.0156 - acc: 0.9948 - val_loss: 0.0185 - val_acc: 0.9950
```

MOBILENET V1 ACC-99.48 ,LOSS-1.5% OVER 10 EPOCHS

Classification Models Tested


```
2019-06-12 12:27:44.788209: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1003] 0:   N
2019-06-12 12:27:44.791862: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1115] Created TensorFlow device (/job:localhost/rep
 GeForce GTX 1050 Ti, pci bus id: 0000:07:00.0, compute capability: 6.1)
WARNING:tensorflow:From C:\Users\legend698\Anaconda3\envs\ten\lib\site-packages\tensorflow\python\ops\math_ops.py:3066: to_int32 (
e version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/5
2019-06-12 12:27:47.068849: I tensorflow/stream_executor/dso_loader.cc:152] successfully opened CUDA library cublas64_100.dll loca
2019-06-12 12:27:49.041854: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying
n that there could be performance gains if more memory were available.
 - 711s - loss: 0.6461 - acc: 0.8817 - val_loss: 0.6039 - val_acc: 0.9275
Epoch 2/5
 - 645s - loss: 0.5664 - acc: 0.9361 - val_loss: 0.5307 - val_acc: 0.9225
Epoch 3/5
 - 435s - loss: 0.4987 - acc: 0.9384 - val_loss: 0.4687 - val_acc: 0.9225
Epoch 4/5
 - 369s - loss: 0.4414 - acc: 0.9381 - val_loss: 0.4164 - val_acc: 0.9225
Epoch 5/5
 - 401s - loss: 0.3930 - acc: 0.9384 - val_loss: 0.3723 - val_acc: 0.9250

(ten) C:\Users\legend698\Desktop\ANIMESH\python>
```

VGG16-93% ACCURACY,LOSS-39% OVER 5 EPOCHS

Conclusion

Work Done:

- Tested 3 models on Dog v/s cats dataset with 12000 training images.
    1. VGG 16
    2. MobileNet V1
    3. EffNet
- Object detection
    1. FRCNN
    2. SSD MobileNet V2
- GUI Design for desktop app using Adobe Xd.
- GUI Development using Electron Js.
- GUI and Python Backend Integration using tornado.

References

- Best Parental-Control Apps 2018: Paul Wagenseil & Brian S. Hall. Accesed at https://www.tomsguide.com/us/best-parental-control-apps,review-2258.html
- Yarn LeCuml, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jacket, Handwritten digit recognition with a back-propagation network, in: Proceedings of the Advances in Neural Information Processing Systems (NIPS), 1989, 396-404.
- Y. LeCun, L. Bottou, Y. Bengio, P. Hafiiier, Gradient-based learning applied to document recognition, Proceedings of IEEE 86 (11) (1998) 2278-2324.
- R. Hecht-Nielsen, Theory of the backpropagation neural network, Neural Networks 1 (Supplement-1) (1988) 445-448
- C. Y. Suen, A novel hybrid classifier for recognizing handwritten digits, Pattern Recognition 45 (4) (2012) 1318-1325.
- Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks
- K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: Proceedings of the International Conference on L R rest (ICLR), 2015. earning up en atlons
- C. Szegedy, W. Y. P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, 1-9.
- M. D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: Proceedings of the European Conference on Computer Vision 2014, 818-833.

References

- K. He, X. Zhang, S. Ren, J. Deep residual learning for image recognition in' Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2016, 770-778.
- Open Sourcing a Deep Learning Solution for Detecting NSFW Images. Accessed at /open-sourcing-a-deeplearning-solution-for
- Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid William J. Dally, Kurt Keutzer: SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size (2016). Accessed at
- S. Avila, E. Valle, and A. D. A. ArauJo. (May 15, 2017). NPDI pornography database, [Online]. Available: https://sites.google.com/site/ pomographydatabase/.
- F. Brian, T. Y. Wang, M. and J. "Pornographic image detection utilizing deep convolutional neural networks," Neurocomputing, vol. 210, 283 _293, 2016.
- M. Moustafa, "Applying deep learning to classify pornographic images and videos," preprint arXiv:1511.08899, 2015.
- Geoffrey Hinton, Oriol Vinyals, Jeff Dean: Distilling the Knowledge in a Neural Network (2015) Accessed at
- N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1):1929-1958, 2014.
- Song Han, Huizi William J. Dally: Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding (2015) Accessed at
- Diederik P. Kingma, Jimmy A Method for Stochastic Optimization (2014) Accessed at https://arxiv.org/abs/1412.6980