Poor Man’s Rekognition

CCExtractor

Mentor : Amit Kumar

Personal Details:

Name : Pulkit Mishra

University : Indian Institute of Information Technology Kalyani
Course : Bachelor of Technology in Computer Science and Engineering
Course Term: 4 Years (2017 - 2021)
Current Year: 3rd Year
CGPA : 9.18

Portfolio : https://pulkitmishra.github.io/
GitHub : https://github.com/PulkitMishra
LinkedIn : https://www.linkedin.com/in/pulkit-mishra/
CV :
https://github.com/PulkitMishra/PulkitMishra.github.io/blob/master/resources/Pulkit_Hackathon_Resume.pdf
Medium : https://medium.com/@pulkitmishra

Projects:

1. ‘HereToHear’ - Artificial Intelligence powered Counsellor that detects and treats Depression
    - Mood evaluation by the Alexa bot along with suicidal behaviour detection and prevention.
    - Social Media Sentiment Analysis to dig up evidence of mental illness, if any.
    - Intelligent Suggestions by the Alexa AI to improve user's mood.
    - Informs emergency contact via text if the user is sad or, in extreme cases, shows suicidal tendencies.
    - Tech Stack : Alexa Skills, Flask-ASK, Machine Learning
    - Link

2. ‘Jinkies’ - CI and CE exclusively built for Machine Learning
    - Fills the gap between traditional CIs and Data Science
    - Provides data visualization to help developers in making better models
    - Provides Version Controlling of Models
    - Tech Stack : Jenkins, Docker, Flask, MongoDB
    - Link

3. Correction of annotation affected words in document images
    - Used Text Imputation followed by grammar checking for post-OCR text correction in annotation affected documents

    - Tech Stack : Tensorflow, RNNs, NLTK

4. ‘HackTenders’ - E-Tendering on blockchain
    - Automates the tendering process
    - Makes the process more secure and transparent
    - Tech Stack : BigChainDB, Inter Planetary FIle System, Flask, MongoDB
    - Link

5. ‘killall_queue’ - Smart and Secure Shopping System
    - Eliminates the need of a queue in shopping mall by automating the billing process
    - Also provides support for inventory management
    - Tech Stack : Internet Of Things, Raspberry Pi, RFID, Android Studio, Firebase
    - Link

6. ‘eye.ai’ - Artificial Intelligence powered wearable to assist the visually impaired in interacting with the environment
    - Environmental Analysis to give the visually impaired a brief summary of his/her surroundings.
    - Identifies faces of known people(friends and relatives) and informs about their presence
    - Object Identification (like clocks)
    - Identification and option to read out texts in surroundings, i.e- banners, information at stations, etc.
    - Alexa for voice interface
    - Tech Stack : Deep Learning(CNN and LSTM), Raspberry Pi, Alexa Skills
    - Link

7. Social Media for Literature Enthusiasts
    - Platform to share and read poems, novel extracts and other literary pieces
    - Tech Stack : Django, React

8. Developed Website of Indian Institute of Information Technology Kalyani Free and Open Source Club
    - Front end of College’s Free and Open Source Club’s website
    - Tech Stack : HTML, CSS, JS
    - Link

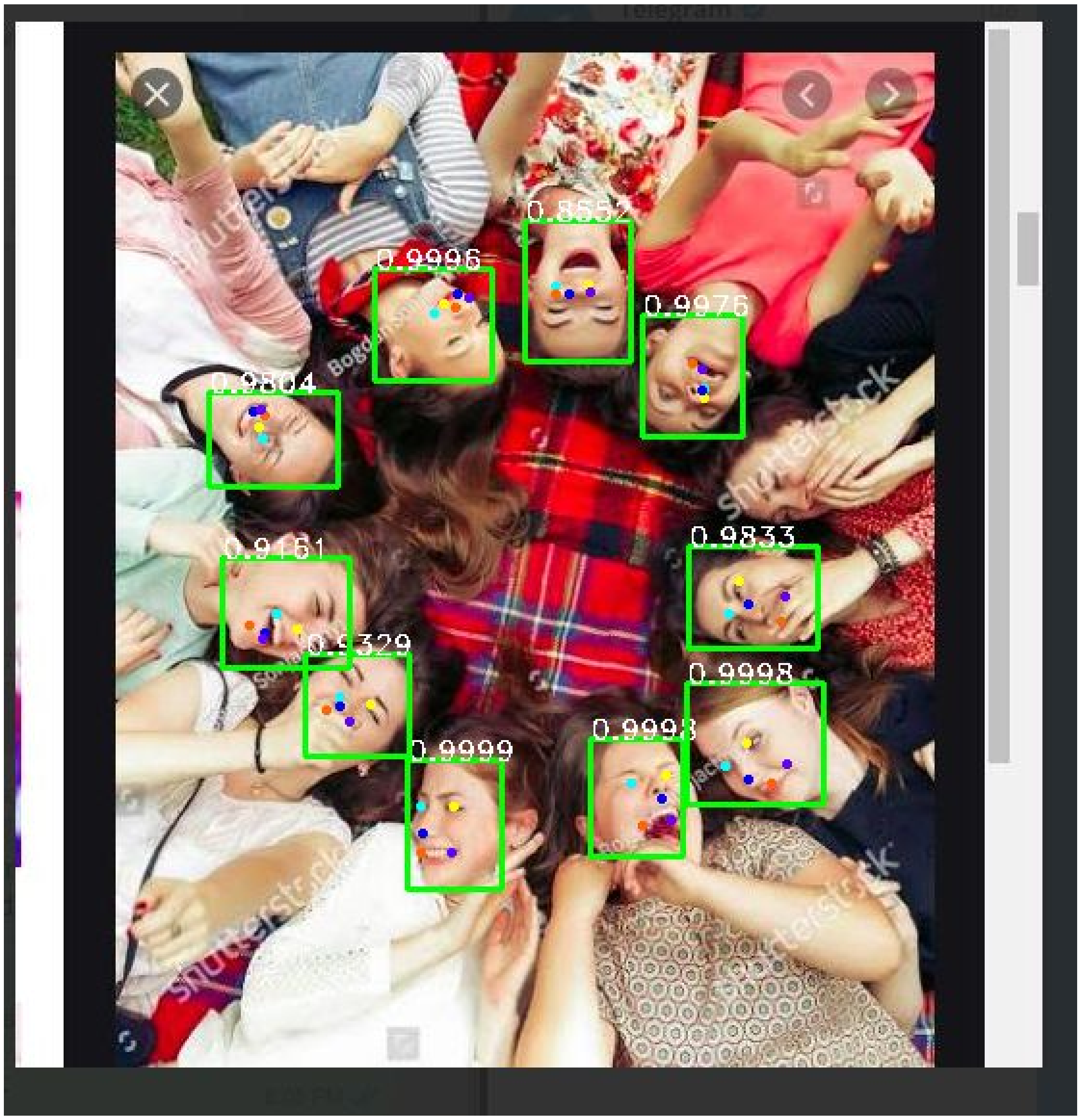
Contact Details:

Time Zone : IST (GMT +5:30)
Email : pulkit@iiitkalyani.ac.in
Skype : pulkitmishra007
Slack : @pulkit
Mobile : +91 - 7860564879 (India)
Facebook : https://www.facebook.com/pulkitmishra007
Instagram : https://www.instagram.com/pulkitmishra_/
Twitter : https://twitter.com/its_only_words_

Open source Contributions :

1. To CCExtractor
    a. Issues
        - Failed to build scikit-learn and scipy (#61)
        - Add Documentation (#64)
        - Error when no faces present in reference image (#70)
        - Resize method of skimage throws waning (#68)
        - More to come
    b. Pull Request
        - Documentation for entire project (#73)
        - Trained a better Facial Expression Recognition model and deployed on TFS(#72)
        - Display error message when no face in reference image(#71)
        - Documentation for nsfwClassifier added (#63)
        - Fix for warning thrown by skimage.resize(#69)
        - Change dependency version of numpy, scipy and scikit-learn(#62)
        - More to come
    c. https://ccextractor.org/public:gsoc:takehome
        - Deep Learning Task : Improve the current face detection model. I did this by implementing RetinaFace, the current state of the art in face detection.

Following Image was supposed to be given as the test input.



- Code Organization Task : [Meetup auto-RVSP]
  Code https://github.com/PulkitMishra/meetup-auto-rsvp

2. Contribution to other
   a. Popping a mini browser for wiki (PR #73)
      - The link for wiki used to open in the default browser window.
      - One of the tasks marked #TODO was to make the same happen by popping a mini browser.

   b. Adding Custom Equation List (PR #83)
      - One of the tasks marked #TODO was to help user in selecting equations from file and adding them to input space

c. Started work on LaTeX to text parser (PR #76)
    - Issue #63 desired that Visma support LateX expressions as inputs
    - Thus PR added parsing for /frac and also a test case for the same

d. Added functionality to the matrix module (PR #135 & #154)
    - Transpose Calculation
    - Equality check for matrices
    - Check for Square Matrix
    - Added Diagonal Matrix class and checks for the same
    - Changed the structure of Identity Matrix Class and added checks for the same

e. Operator Overloading for Constant and Variable classes (PR #181)
    - Overloading add and subtract for Constant and Variable
    - Adding Multiplication and Division overloading for Constant Class
    - Unit testing

f. Add support for png files (PR #609)
    - Added support for PNG files in Labelme for converting Labelme datset to COCO style dataset

g. Bug Fixes (PR #135, #154, #79, #80, #94, #97,)
    - Fixed Deprecation warning as highlighted in issue #136 raised by me
    - Fixed .gitignore to avoid the local log.txt from getting uploaded
    - Avoided crashing of LaTeX parser when input was given byte by byte
    - Prevented Visma from crashing when simplify button was pressed with no input and thereby fixed issue #66
    - Edge case checks for adding custom equations (1 and 2)
    - Pressing enter with nothing else in input space resulted in Visma exiting abruptly.

Hackathons :

1. Hack-A-BIT
    - BIT Mesra, Ranchi

- November 2018
- Most Innovative Hardware Hack

2. Codeutsava 3.0
    - National Institute of Technology, Raipur
    - February 2019
    - Second Runners Up
3. Smart India Hackathon
    - March 2020
    - Regional Level : Winner
    - National Level : Finalist

4. Hack In The North
    - Indian Institute of Information Technology, Allahabad
    - March 2019
    - Among the Top Ten Hacks
5. Hackfest
    - Indian Institute of Technology, Dhanbad
    - March 2019
    - Among the Top Twenty Hacks
6. iHack
    - Indian Institute of Technology, Bombay
    - January 2019
    - Finalist

Work/Intern Experience :

1. Superbolter
   Software Development Intern
   June 2019 - August 2019
    - Recommending the best room images to customer on the basis of text description depending on interior design style and furniture.
    - Conversion of floorplan images into vector-graphics representation.
    - Improved the webcrawler by bypassing crawlblock
2. Machine Origin
   Deep Learning Intern
   September 2019 - October 2019
    - Detection of action sequences and celebrities in videos.
    - Humour detection in videos. Highlight generation of sports videos.
    - Video genre classification.
3. Invigilo Technologies
   Computer Vision Intern
   January 2020 - Present

- Detection of excavator body and arm in videos
- Detection of Trenches and shorings in videos

Why I am best suited for this project :

Firstly, I have been following Poor Man's Rekognition especially Amit's work closely for about an year now. I have seen the project being developed from day 1 with all the commits piling up one by one.

Secondly, I believe that I have the necessary skills required for the project.

- I have taken the following courses at my university
    1. Programming in Python
    2. Artificial Intelligence
    3. Machine Learning
    4. Image Processing and Computer Vision
- I have done projects that involve
    1. Writing code in python
    2. Working with OpenCV
    3. Django
    4. Deep Learning
    5. Docker
- By following PMR for such a long time I have developed a strong command over Tensorflow Serving

Thirdly, I have done a lot of research on the project and in my humble opinion it will not be wrong to say that I have gone through almost everything on the web that had Rekognition on it -

- Features offered by Amazon's Rekognition
- Blogs from all the previous contributors and gone through the work done by them in the past and report for the same is provided below
- Read Amit's proposal for GSOC 2019
- PMR's documentation, contributing guidelines, code base and kanban board on github

Lastly and most importantly, I have a well structured plan with the time bound deliverables and implementation details clearly provided below.

Report on previous year's work :

	https://github.com/pymit/	https://gitlab.com	https://github.co	https://github.co

	Rekognition	/drcpmkeyi/poor-man-rekognition	m/thelastpolaris/poors_man_rekognition	m/sziraqui/pmr-server
The Good	Most features that were proposed are implemented. Face Detection works decently but is based on MTCNN and can be improved to the current state of the art i.e. RetinaNet (which is a part of my proposal) Face Recognition works great and can easily be scaled by adding more embeddings. DjangoRestFamework has been beautifully integrated making development work very easy and fast. Using Tensorflow serving is an excellent idea as it helps a lot in shifting models to production. Documentation is great. Both inline as well as the one for each API and so are the blogs. Didn't have to bug Amit for stuff. Easy to understand code with the documentation made it very easy for me to get familiar with the codebase. NSFW classifier is excellent. Worked great with no bugs on all images that I tried. Video	Mohsin has two face detectors in place in the plug and play mode. I borrowed the idea of having both MTCNN and RetinaFace in the same fashion from his approach. He also had implemented face alignment which is something that I found really interesting and will be using for developing a better Facial Expression Recognition module. I have already incorporated the idea in the preprocessing steps in the PR that I have made for the same purpose. Documentation provided by Mohsin is seriously amazing. Every function, every variable has been explained along with	Artem also has two face detectors in place in the plug and play mode. However he is using MTCNN along with MobileNet. This further solidified my belief of having two separate models for face detection. One model will be state of the art and the other an average one that can provide inference in real time. Artem followed a deep learning approach instead of a ML one for face recognition and as a result his Face Recognition service can identify a lot of celebrity faces as opposed to Amit or Mohsin. However, no support has been added for obtaining custom embeddings of faces.	Sarfaraz's projects add a lot of value for the open source community wanting to do something awesome with JS in the field of deep learning.

	part is incomplete and will be done this GSoC. Similar Face Search also does great but had a small bug which I fixed in one of the PRs. Several utility methods such as downloading youtube videos are already implemented	detailed blog posts. I will be following his approach for writing documentations.		
The Bad	Facial Emotion Recognition had several problems such as dataset being skewed(as Amit himself pointed out), the index and data files were missing from the TFS model and the preprocessing code also warranted some changes (I have done all of this and made a Pull Request for the same) A major issue is with that of memory consumption while processing videos as scikit video uses one hot vector. This has been acknowledged by Amit also in his project kanban board. This needs to be resolved and will be taken care of by breaking down video into parts. Project as of now is structured around faces. I believe the reason is that the first GSoC for PMR was about getting the faces part of Rekognition right. But	Mohsin chose flask and while I appreciate the lightweight framework and believe that it is great for, I genuinely believe that if PMR is to grow then flask is not the way forward. Extremely poor handling of video processing. Amit did a much better job. Video processing was slow and consumed a lot of memory. Very few features were implemented as opposed to Amit.	Again, anything other than Django will become a hurdle as the project begins to scale. Django has even introduced async views now. Tonado, that Artem used is great but has similar shortcomings as Flask. Very few features were implemented as opposed to Amit.	While I appreciate the unconventional path taken by Sarfaraz, I need to point out that the Deep learning frameworks and tools available for python are far more superior than any other . Sarfaraz's is a great building block for other libraries to build up on but definitely not what PMR needs at the current stage.

	now that more features such as object and scene detection have to be added including an in the wild text extraction, code needs to undergo some major refactoring			
Additional Comments	Project also has a UI and a feedback feature in place but they don't excite me much and I will restrict my work towards creating REST APIs and improving the performance of the existing ones. One thing that really bummed me off was manual testing. The project did not have a CI/CD pipeline in place and hence no automated testing. I have added that in my proposal.	CI/CD was working here which was great since it was hosted on gitlab.	The pipeline and kernel based approach was fascinating and is something I will definitely try while working on resolving memory issues with Amit's video processing code.	Took me an entire day to setup his project only to learn that an important part of the project - creating the REST API is incomplete.

The reason why I decided to build up on Amit's work has more or less been highlighted above. Other than the choice of framework, number of features implemented, quality of features implemented and model deployment pipeline, I also found him to be the only one among the four who has been extremely active in the community and has hence been helping me all along with the same.

Deliverables :

Under the project I promise that I will have made the following changes in the three month time period to improve PMR:

- Refactoring code and setting up CI pipeline with static code analysis
    - ➔ **Reason** : Will be essential as the project starts maturing and starts getting more traction from the open source community.
    - ➔ Implementation details:

- Making code modular and object oriented
- Exception handling
- Adding documentation
- Writing unit tests
- Connect Travis CI to the project
- Adding .travis.yml to the repo
- PEP8 check
- Switch all models to tensorflow serving

- Add NSFW Detection in videos
  - **Reason** : NSFW classifier works but only for images as mentioned here

  - Implementation details: Videos will be processed using the same model and pipeline as for images but with added multithreading.

    Following methods will be added to coreapi/main_api.py

    1. def NSFWInVideo(request, filename):
       """   NSFW Recognition in Video

       Args:
       *   request: Post https request containing a video file
       *   filename: filename of the video

       Workflow
       *   Video file is first saved into videos which is subfolder of MEDIA_ROOT directory.

       *   Information about the video is saved into database
       *   Using skvideo meta information of the video is extracted
       *   With the help of extracted metadata frame/sec (fps) is calculated and with this frame_hop is calculated.
       Now this frame_hop is actually useful in decreasing the number of frames to be processed,
       say for example if a video is of 30 fps then with frame_hop of 2, every third frame is processed, It reduces
       the computation work. Of Course for more accuracy the frame_hop can be reduced, but It is observed that this
       affect the output very little.
       *   Now videofile is read using skvideo.io.vreader(), Now, each frame is read from the videogen and the already implemented nsfwClassifier is called for each of the frames

* Now a local dictionary is maintained which keeps the all timestamps and the corresponding predicted classes with their confidence score

* Using the concept of weighted moving average a final class s calculated for the entire video

* After that a dictionary is created which keeps the class id as key and final calculated class as the value

Returns:

* Dictionary having class id as key and final calculated class as the value

“””

Following class will be modified in coreapi/views.py

1. class NSFW_Recognise(views.APIView):
   * Inside post method NSFWInVideo() will be called for video files and NSFWclassiifer method will be called for images.

- Detect and extract text in images and videos.

➔ **Reason** : Present in Amazon’s Rekognition Service

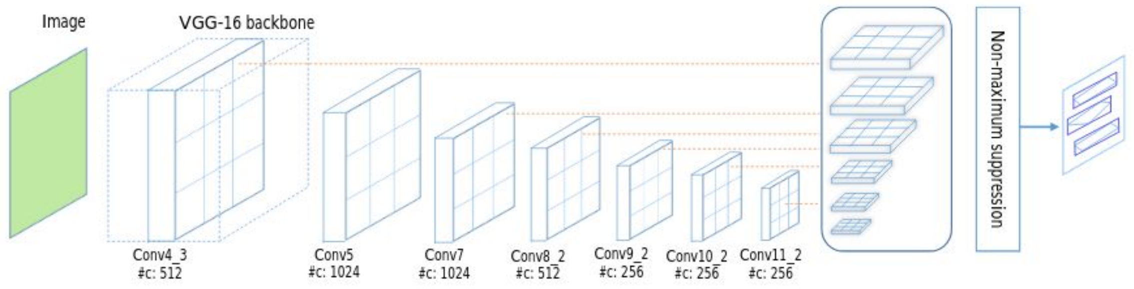
➔ Implementation details:

Reading text from “in the wild” natural images is far more difficult than from images of scanned documents. Scene text images have a complicated background where some patterns are visually indistinguishable from the true text. These images have different colors, sizes, orientations, sometimes curvy, fonts and languages. The images also suffer from a lot of interferences like low resolution, exposure, noise, motion blur, out of focus, varying illumination, etc. In the view of the aforementioned challenges, the problem is split as a two-step process

◆ **Text detection**
- To use single shot detectors instead of region based detectors (as in region-based methods).
  - Region based detectors are slow due to a two-staged approach:
    - ◆ Find the bounding box

        - ◆ Find the class of the bounding box
- To implement a version of TextBoxes++ trained on *SynthText* in the Wild Dataset to detect arbitrary oriented scene text with both high accuracy and efficiency in a single network forward pass
    - No post-processing other than an efficient non-maximum suppression is required.
    - Superior performance to competitors with respect to text localization accuracy and runtime.
- TextBoxes++ architecture
    - Fully convolutional network
    - 13 layers from VGG-16 followed by 10 extra convolutional layers
    - 6 Text-box layers connected to 6 intermediate convolutional layers
        - ◆ Each predicts a n-dimensional vector for each default box consisting of one of the following:
            - text presence scores (2 dimensions)
            - horizontal bounding rectangles offsets (4 dimensions)
            - rotated rectangle bounding box offsets (5 dimensions)
            - quadrilateral bounding box offsets (8 dimensions)
    - Non-maximum suppression to merge the results of all Text-box layers.

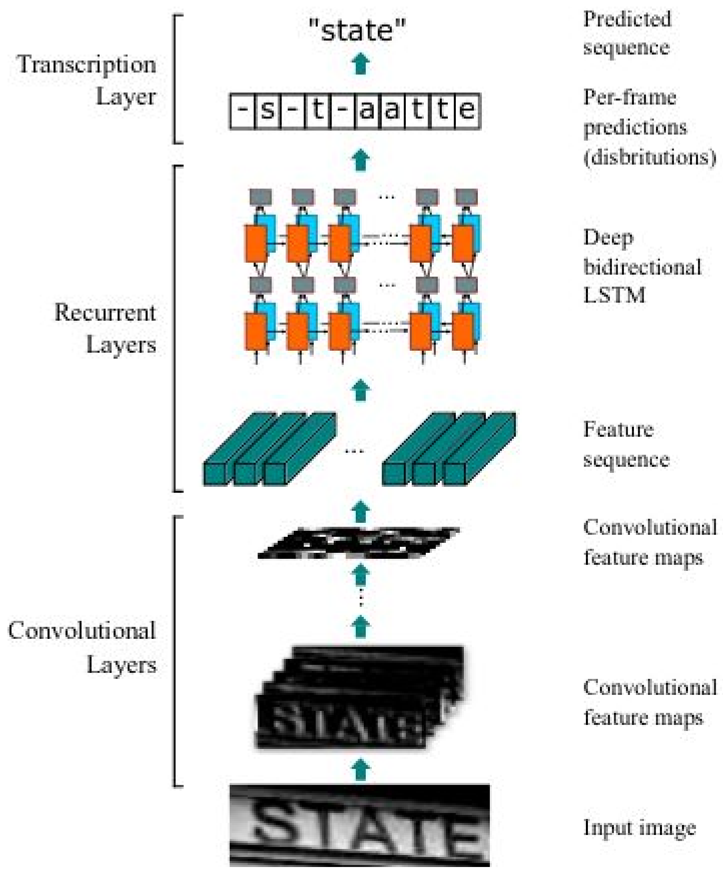
[4]



◆ **Text recognition**

- The bounding box around the detected text is identified in the last step

- CRNN, a combination of CNN, RNN, and CTC(Connectionist Temporal Classification) loss, to recognise detected text. CRNN is chosen for several specifics:
    - Efficient desirable performance in scene text recognition and in OCR, when trained on *SynthText* in the Wild Dataset
    - End-to-end trainability. This is in stark contrast with most of the existing algorithms whose components are separately trained and tuned
    - Handling sequences in arbitrary lengths, without involving character segmentation or horizontal scale normalization
    - Not confined to any predefined lexicon
    - Achieves remarkable performances in both lexicon-free and lexicon-based scene text recognition tasks.
- CRNN generates an effective yet smaller model, which is always desirable
- Architectural specifics:
    - Convolutional layers: extract a feature sequence from the input image
    - Recurrent layers: predict label distribution for each frame
    - Transcription layer: Translates the per-frame predictions into the final label sequence



Following methods will be added to coreapi/main_api.py

1. def textExtractImage(request, filename):
   """ ExtractText in images

   Args:
   * request: Post https request containing a video file
   * filename: filename of the imaage

   Workflow
   * will call textDetectImage(filename) and store the bounding boxes it returns in a local list
   * will call textRecognizeImage(bounding_box) for each bounding box present in the local list and store the string returned to a list
   * After that a dictionary is created which keeps extracted text as key and list as the value

   Returns:
   * Dictionary having extracted text as key and list as the value

“””

2. def textDetect(filename):
   """ Detect bounding boxes around text

   Args:
   * filename: filename of the image

   Returns:
   * Dictionary having bounding boxes as key and list of bounding boxes as the value

“””

3. def textExtract(boundingboxes):
   """ Detect text from bounding boxes

   Args:
   * boundingboxes: bounding boxes detected by textDetect(filename)

   Returns:
   * Dictionary having detected text as key and list of strings detected as the value

“””

4. def textInVideo(request, filename):
   """ NSFW Recognition in Video

   Args:
   * request: Post https request containing a video file
   * filename: filename of the video

   Workflow
   * Video file is first saved into videos which is subfolder of MEDIA_ROOT directory.

   * Information about the video is saved into database
   * Using skvideo meta information of the video is extracted
   * With the help of extracted metadata frame/sec (fps) is calculated
   * Now videofile is read using skvideo.io.vreader(), Now, each frame is read from the videogen and the already implemented textExtractImage(request, filename) is called for each of the frames
   * Now a local dictionary is maintained which keeps the all timestamps and the corresponding predicted text
   * After that a dictionary is created which keeps the detected text as key and local dictionary generated above as the value

Returns:
* Dictionary having detected text as key and dictionary generated above as the value

“””

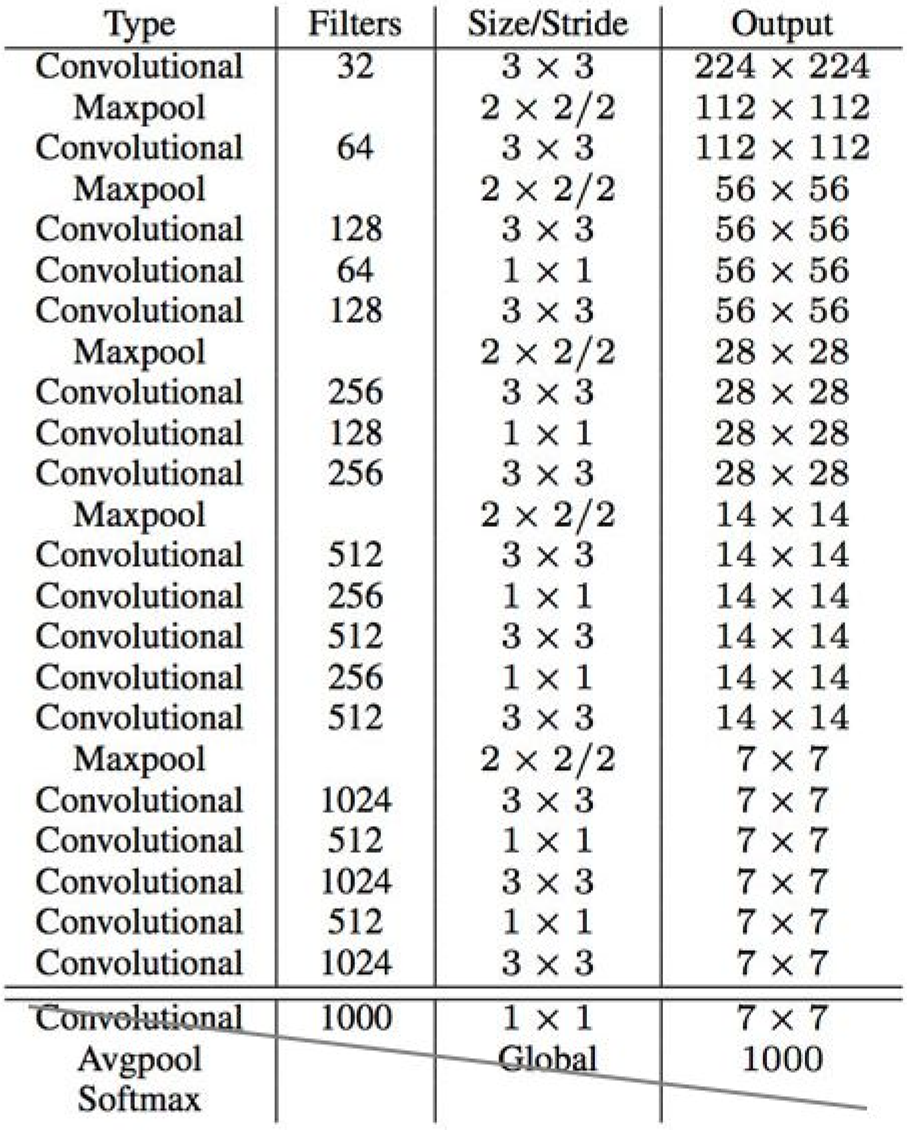
Following class will be created in coreapi/views.py

1. class Text_Recognize(views.APIView):
    * Inside the post method textInVideo(request, filename) will be called for video files and textInImage(request, filename) method will be called for images.

- Detect objects in images and videos

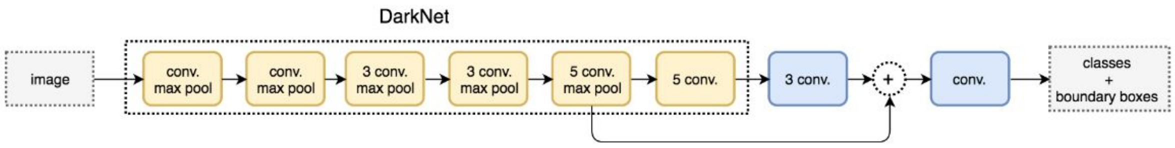
    ➔ **Reason** : Present in Amazon’s Rekognition Service

    ➔ Implementation Details
        - Using YoloV3 + DarkNet
        - Why DarkNet? DarkNet is an efficient backbone for YoloV3 because of the following specifics:
            - Requires 5.58 billion operations only.
            - Helps YOLO achieve 72.9% top-1 accuracy and 91.2% top-5 accuracy on ImageNet
            - Efficient architectural combinations:
                - 3 × 3 filters to extract features
                - 1 × 1 filters to reduce output channels
                - global average pooling

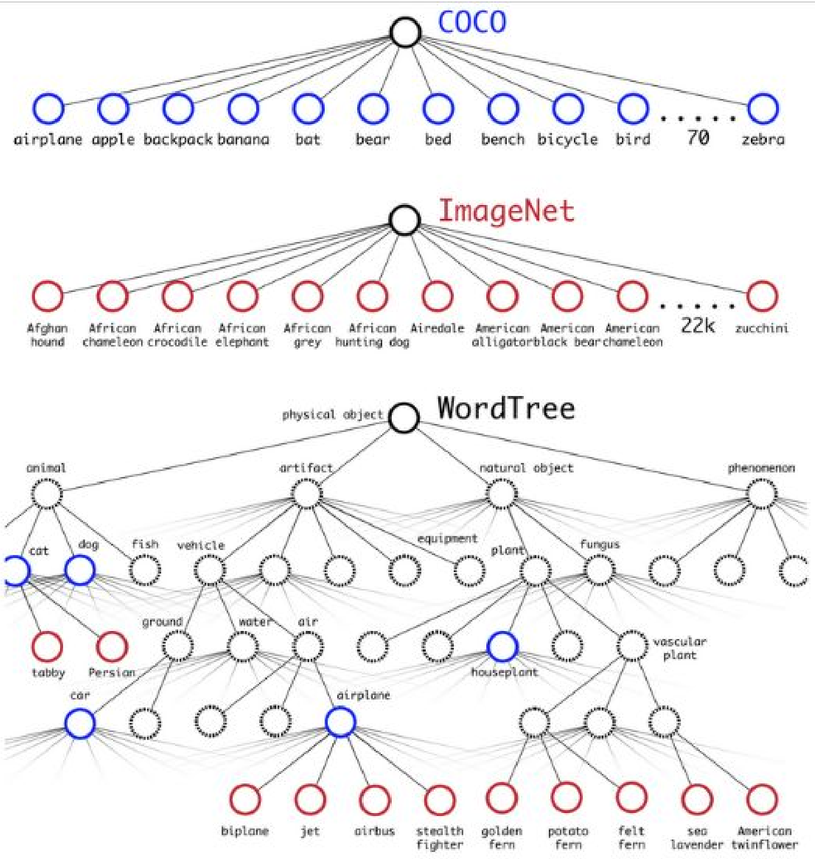


Type	Filters	Size/Stride	Output
Convolutional	32	3 × 3	224 × 224
Maxpool		2 × 2/2	112 × 112
Convolutional	64	3 × 3	112 × 112
Maxpool		2 × 2/2	56 × 56
Convolutional	128	3 × 3	56 × 56
Convolutional	64	1 × 1	56 × 56
Convolutional	128	3 × 3	56 × 56
Maxpool		2 × 2/2	28 × 28
Convolutional	256	3 × 3	28 × 28
Convolutional	128	1 × 1	28 × 28
Convolutional	256	3 × 3	28 × 28
Maxpool		2 × 2/2	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Maxpool		2 × 2/2	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
~~Convolutional~~	~~1000~~	~~1 × 1~~	~~7 × 7~~
~~Avgpool~~		~~Global~~	~~1000~~
~~Softmax~~			

The last convolution later (crossed-out section) will be replaced with three 3 × 3 convolutional layers each outputting 1024 output channels. Post this, a final 1 × 1 convolutional layer will be applied to convert the 7 × 7 × 1024 output into 7 × 7 × 125. (5 boundary boxes each with 4 parameters for the box, 1 objectness score and 20 conditional class probabilities).



YOLO is trained with the ImageNet 1000 class classification dataset in 160 epochs: using stochastic gradient descent with a starting learning rate of 0.1, polynomial rate decay with a power of 4, weight decay of 0.0005 and momentum of 0.9. In the initial training, YOLO uses 224 × 224 images, and then returns it with 448× 448 images for 10 epochs at a 10−3 learning rate. After the training, the classifier achieves a top-1 accuracy of 76.5% and a top-5 accuracy of 93.3%. Then the fully connected layers and the last convolution layer is removed for a detector. YOLO adds three 3 × 3 convolutional layers with 1024 filters each followed by a final 1 × 1 convolutional layer with 125 output channels. (5 box predictions each with 25 parameters) YOLO also adds a passthrough layer. YOLO trains the network for 160 epochs with a starting learning rate of 10−3 , dividing it by 10 at 60 and 90 epochs. Datasets for object detection have far fewer class categories than those for classification. To expand the classes that YOLO can detect, YOLO proposes a method to mix images from both detection and classification datasets during training. It trains the end-to-end network with the object detection samples while backpropagates the classification loss from the classification samples to train the classifier path. YOLO also combines labels in different datasets to form a tree-like structure **WordTree**. The children form an is-a relationship with its parent like biplane is a plane.



Following methods will be added to coreapi/main_api.py

1. def objectImage(request, filename):
   """ Objects in images

   Args:
   * request: Post https request containing a video file
   * filename: filename of the imaage

   Returns:
   * Dictionary having detected objects as key and dictionary with detected objects and their confidence score as the vale
"""

2. def objectVideo(request, filename):
   """ Objects in Video

   Args:
   * request: Post https request containing a video file
   * filename: filename of the video

   Workflow
   * Video file is first saved into videos which is subfolder of MEDIA_ROOT directory.

   * Information about the video is saved into database
   * Using skvideo meta information of the video is extracted
   * With the help of extracted metadata frame/sec (fps) is calculated
   * Now videofile is read using skvideo.io.vreader(), Now, each frame is read from the videogen and the already implemented objectImage(request, filename) is called for each of the frames
   * Now a local dictionary is maintained which keeps the all timestamps and the corresponding predicted object with confidence score
   * After that a dictionary is created which keeps the detected object as key and local dictionary generated above as the value

   Returns:
   * Dictionary having detected objects as key and dictionary generated above as the value
"""

Following class will be created in coreapi/views.py

1. class Object_Detect(views.APIView):

* Inside the post method objectVideo(request, filename) will be called for video files and objectImage(request, filename) method will be called for images.

- Detect scenes in images and videos

  - ➔ **Reason** : Present in Amazon’s Rekognition Service

  - ➔ Implementation Details
    - ◆ Training CNN based classification model on Places365 dataset
    - ◆ Why Places365 dataset? Dataset specifics mentioned below make models trained on it better suited to scene detection.
      - Dataset design on principles of human visual cognition and is a core of visual knowledge that can be used to train artificial systems for high-level visual understanding tasks, such as scene context, object recognition, action and event prediction, and theory-of-mind inference.
      - More than 10 million images comprising 400+ unique scene categories, with 5000 to 30,000 training images per class, consistent with real-world frequencies of occurrence.
      - Semantic categories of Places are defined by their function: the labels represent the entry-level of an environment.
    - ◆ Models trained on Places365 allows learning of deep scene features for various scene recognition tasks.
    - ◆ Final prediction will be based on the combination of GoogLeNet and VGG16, given their impressive performance on ImageNet benchmark.

  Following methods will be added to coreapi/main_api.py

  1. def sceneClassifier(request, filename):

     """ scene classifier of images

     Args:

     * request: Post https request containing a image file
     * filename: filename of the video

     Workflow:

     * A numpy array of an image is taken as input (RGB). inference input dimension requires dimension of

(64,64,1) and therefore the RGB input is resized to the required input dimension.

* Now the processed output is further processed to make it a json format which is compatible to TensorFlow Serving input.

* Then a http post request is made at localhost:8501 . The post request contain data and headers.

* Incase of any exception, it return empty string.

* output from TensorFlow Serving is then parsed and a dictionary is defined which keeps classes and probabilities as the two keys. Value of Classes is the class with maximum probability and value of probabilities is a dictionary of classes with their respective probabilities

Returns:

* Dictionary having the class with maximum probability and probabilities of all classes.

"""

2. def sceneClassifierVideo(request, filename):

""" NSFW Recognition in Video

Args:

* request: Post https request containing a video file

* filename: filename of the video

Workflow

* Video file is first saved into videos which is subfolder of MEDIA_ROOT directory.

* Information about the video is saved into database

* Using skvideo meta information of the video is extracted

* With the help of extracted metadata frame/sec (fps) is calculated and with this frame_hop is calculated.

Now this frame_hop is actually useful in decreasing the number of frames to be processed,

say for example if a video is of 30 fps then with frame_hop of 2, every third frame is processed, It reduces

the computation work. Of Course for more accuracy the frame_hop can be reduced, but It is observed that this

affect the output very little.

* Now videofile is read using skvideo.io.vreader(), Now, each frame is read from the videogen and the already implemented sceneClassifier is called for each of the frames

    * Now a local dictionary is maintained which keeps the all timestamps and the corresponding predicted classes with their confidence score
    * Using the concept of weighted moving average a final class is calculated for the entire video
    * After that a dictionary is created which keeps the class id as key and final calculated class as the value

    Returns:
    * Dictionary having class id as key and final calculated class as the value
“””

Following class will be added in coreapi/views.py

1. class Scence_Recognise(views.APIView):
   * Inside post method sceneClassifierVideo will be called for video files and sceneClassifier method will be called for images.

- Replace MTCNN with RetinaFace

  - ➔ **Reason** : MTCNN was great but RetinaFace is better and faster and also is the current state-of-the-art for face detection

  - ➔ Implementation Details
    - ◆ RetinaFace specifics making it a better choice than MTCNN:
      - robust single-stage face detector
      - pixel-wise face localization on various scales of faces
      - takes advantage of jointextra-supervised and self-supervised multi-task learning
    - ◆ Use of RetinaFace trained on WIDER FACE dataset
    - ◆ WIDER FACE dataset specifics:
      - 32,203 images and 393,703 face bounding boxes
      - high degree of variability in scale, pose, expression, occlusion and illumination
      - Data split: training(40%), validation (10%) and testing (50%)
      - Split by random sampling from 61 scene categories

- EdgeBox detection rate is utilised in incorporating incrementally hard samples, leading to three levels of difficulty: *Easy, Medium*, and *Hard.*

Following methods will be modified in coreapi/main_api.py

1. def FaceRecogniseInImage(request, filename):
   """ Face Recognition in image

   Args:
   * request: Post https request containing a image file
   * filename: filename of the video
   * network: The network architecture to be used for face detection

   Workflow:
   * Image file is first saved into images which is subfolder of MEDIA_ROOT directory.
   * If there is any file in the post request and the file extension is mentioned in allowed_set then it is
     allowed for further processing else the returns an error.
   * Then, the image is converted into numpy array, followed by reshaping the image dimension if it contains 4 channels.
     It is actually done to process .png files.
   * Then, all the faces present in an image is extracted along with corresponding boundingbox using method get_face.
   * if number of faces is greater than 0 then for each face and corresponding bounding box is taken for further processing.
   * embedding for each face is created with the help of embed_image and returned to the vairable embedding
   * Now this embedding is compared with already available embeddings, It returns the name of the embedding which has the
     lowest difference in them else if it crosses the limit then 'unknown' id returned.
   * Next step is to get the facial expression of the face using the method FaceExp.
   * Then, all the information which currently includes face id, bounding box and facial expression is saved into a dictionary.
   * Information about the face is saved into database.

   Returns:
   * Dictionary having all the faces and corresponding bounding boxes with facial expression

"""

Following class will be added in coreapi/serializers.py

1. class IMAGE_FRSerializers(serializers.Serializer):
   file = serializers.ImageField()
   network =
serializers.ChoiceField(choices=IMAGE_FR_NETWORK_CHOICES,
default=IMAGE_FR_NETWORK_CHOICES[1])

Following class will be modified in coreapi/viewspy

1. class IMAGE_FR(views.APIView):

   * Inside post method sceneClassifierVideo will be called for video files and sceneClassifier method will be called for images.

A new file called retinanet.py will be created.

This file will have the class :

1. class FaceDetectionRetina(object):
   """
   Used for FaceDetectionRetina, also this class acts as a sub
module for embeddings and Video Recognition Modules
   """

The above class will have the following methods

a. def __init__(self, down_scale_factor=1.0):

b. @staticmethod
   def _pad_input_image(img, max_steps):
   """pad image to suitable shape"""

c. @staticmethod
   def _recover_pad_output(outputs, pad_params):
   """recover the padded output effect"""

d. @staticmethod
   def _draw_bbox_landm(img, ann, img_height, img_width):
   """draw bboxes and landmarks"""

e. @staticmethod
   def _draw_anchor(img, prior, img_height, img_width):
   """draw anchors"""

f. @staticmethod
   def _process_outputs(output, img_height_raw, img_width_raw):

g. def get_face(self, file):
   """ to get the face from the image

   Args:
   img: image file
   Returns:
   numpy arrays of faces and corresponding faces

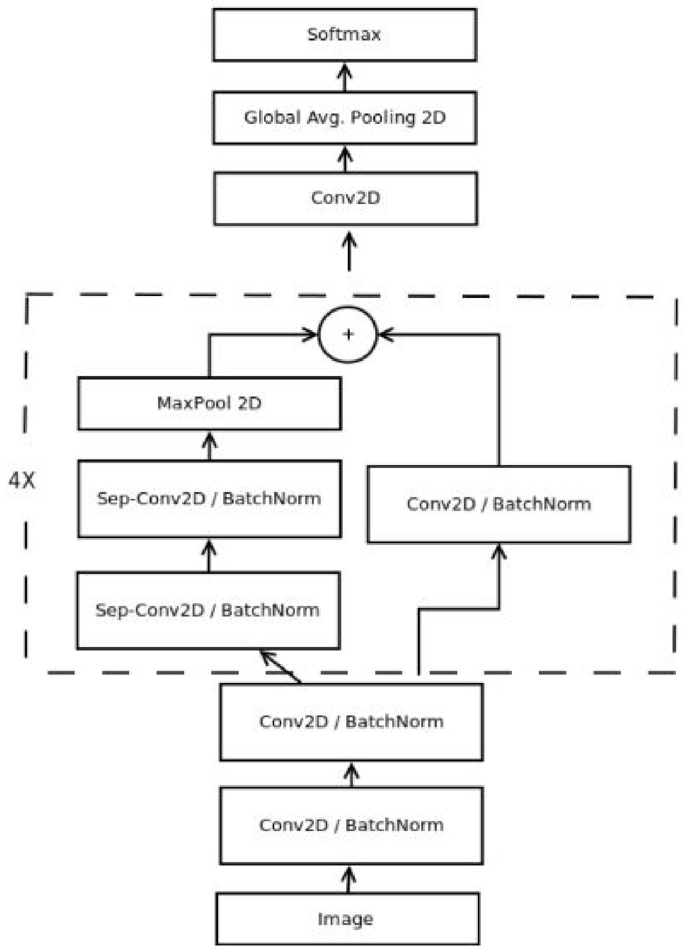
   """

h. def predict(self, img):
   """   Returns the bounding box coordinates along with the resultant
image """

- Improve performance of Facial Expression Recognition model

  ➔ **Reason** : Current Facial Expression Recognition model does not work well

  ➔ **Implementation Details** : I will be training a convolutional neural network on the FER-2013 emotion dataset. FER-2013 has 35,685 examples of 48x48 pixel grayscale images of faces with each image characterized based on the emotion shown in the facial expressions (happiness, neutral, sadness, anger, surprise, disgust, fear). VGG, Inception will be ensembled with Residual Masking approach for this purpose. The architecture of the same is given below.



Following methods will be modified in coreapi/main_api.py

1. def FaceExp(cropped_face):

""" Facical Expression Recognition of faces in image

Args:

* cropped_face: numpy array of cropped face

Workflow:

* A numpy array of a cropped face is taken as input (RGB). inference input dimension requires

a dimension of (1,100,100,3) therefore the RGB input is first converted to grayscale image followed by normalizing and resizing the required input dimension.

* Now the processed output is further processed to make it a json format which is compatible to TensorFlow Serving input.

* Then a http post request is made at localhost:8501 . The post request contain data and headers.

* Incase of any exception, it return empty string.

* output from TensorFlow Serving is then parsed and a dictionary is defined which keeps the facial expression name as key and prediction's output as value. The prediction values are floating point values which tells the probability of the particular facial expression.

Returns:

* Dictionary having all the faces and corresponding facial expression and it's values.

"""

- Containerize the entire project

    ➔ **Reason** : To make sure that PMR works seamlessly in development, test and production environments.

    ➔ **Implementation Details** : Dockerfile with the following specifications will be added
        - ◆ Official Python runtime to be used as a parent image
        - ◆ Setting environment variable
        - ◆ Commands to make directories and install requirements

        docker-compose.yml file will also be added with the following specifications

- Designing Logo for PMR

Reason : Why not ?

Tentative timeline :

GSoC (May 4 - Forever)

No! That’s not a typo. Let me explain :)

Before GSoC	April 9 - May 4	I have already mentioned the work that has already been done by me. I will be spending the time after proposal submission and before accepted student proposal announcement to do the following: ● Adding Documentation ● Working on independent open issues and few bugs (not related to other issues and tractable enough not to be a part of this proposal) ● Getting more familiar with the codebase ● Continued discussion (relevant topic) with the mentor and previous contributors
Community Bonding Period	May 04 - June 01	● Contributing bug fixes/patches by working on the remaining issues ● Getting more familiar with the codebase ● Cleaning up the codebase ● Setting up blog for GSoC
Week 1	June 01- June 08	Refactoring Code ● Setting up CI/CD pipeline ● Writing unit tests
Week 2	June 08 - June 15	Refactoring Code ● Increasing code modularity and object-oriented based logic design ● Adding exception handling modules ● Porting all models to tensorflow serving

Week 3	June 15 - June 22	Text Detection and Recognition model training ● Downloading and Cleaning SynthText dataset ● Training model for Text Detection by implementing TextBoxes++ architecture ● Postprocessing the above output by non-maximum suppression ● Training model for Text Recognition by implementing a CRNN
Week 4	June 22 - June 29	Text Detection and Recognition model deployment ● Text Detection and Recognition model deployment for images ● Add support for video processing of text extraction ● Unit Testing
Phase I Evaluation 1. Refactored Code 2. CI/CD pipeline setup 3. REST API created for Text Extraction from ‘in the wild’ images and videos		
Week 5	June 29 - July 06	Object Detection Model Training ● Downloading and Cleaning ImageNet and COCO dataset ● Training model for Text Detection by implementing YOLOV3 with DarkNet as backbone.
Week 6	July 06 - July 13	Object Detection Model Deployment ● Object Detection model deployment for images ● Add support for video processing of object detection ● Unit Testing

Week 7	July 13 - July 20	Scene Classification Model Training ● Downloading and Cleaning Places365 dataset ● Training 2 CNN models for Scene Classification (VGG and InceptionV3) ● Ensembling the models
Week 8	July 20 - July 27	Scene Classification Model Deployment ● Scene Classification model deployment for images ● Add support for video processing of Scene Classification ● Unit Testing
Phase II Evaluation 1. REST API created for Object Detection in images and videos 2. REST API created for Scene Classification in images and videos		
Week 9	July 27 - August 03	RetinaFace Model Training RetinaFace Model Deployment ● RetinaFace model deployment for images ● Add support for video processing of RetinaFace ● Adding option to detect faces from either MTCNN or RetinaFace model
Week 10	August 03 - August 10	Facial Expression Recognition Model Training Facial Expression Recognition Model Deployment ● Facial Expression Recognition model deployment for images ● Add support for video processing of Facial Expression Recognition

		Add support for video processing of NSFW Classification
Week 11	August 10 - August 17	Containerizing the entire project using Docker
Week 12	August 17 - August 24	Buffer week ● Complete Pending work ● Fixing of bugs and issues that may have been produced in the previous weeks ● Discussion with mentor ● Designing Logo ● Documentation/Tutorial/Updating Wiki for future contributors
Final Evaluation 1. Improved the performance of face detection model 2. Improved the performance of face detection model 3. Added support for video processing of the existing NSFW Classification model 4. Containerized the entire project 5. Designed Logo		

After GSoC :

I plan on sticking with the community forever (assuming you people still keep me around, and don't start hating me for my over-excitement). There are a lot of things that I can think of that can be added to PMR - some of which I personally wanted to add but am not able to do so as the three month time period will not be enough for the same. I would really love to see all these features and would add some of them myself after GSoC gets over. A few examples would be an image captioning model, people pathing and a nice UI for PMR.

I also wish to contribute to the other awesome projects under CCExtractor after GSoC is over. I have already mentioned in my slack introduction that outside GSoC I'd love to help with Poor Man's Textract. So during GSoC or after that this is something that I can help CCExtractor with whenever required. Other than that I'd love to work on the core repository as well. I believe that with the help of such awesome people at CCExtractor like Carlos and Amit, I'd soon develop a quick grasp of the concepts required to start making valuable contributions to the core repository.

One thing that I would surely do regardless of my selection, would be to publicize the work done by CCExtractor. I genuinely believe that more people need to know and start talking about how you people are creating open space for everyone. I'm going to blog about the work done on Medium, tweet about it and post about it on Instagram whether or not I'm selected for GSoC.

Other Commitments :

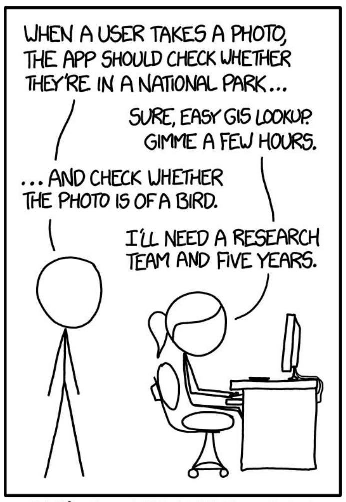
I will have my college exams from the first week of May to the second week of May as a result of which I will not be very active in the initial part of the community bonding period.

Other than that I will be able to give my 100% from May 15 onwards i.e. I will easily be able to work for eight hours per week and in fact much more than that.

There won't be any time zone issues as well, since the mentor (Amit Kumar) shares the same timezone as I.

Conclusion

I would like to thank you for taking out the time to read my proposal. I hope that I get a chance to be a part of this awesome community. Here’s a XKCD to end it on a light note:)


WHEN A USER TAKES A PHOTO, THE APP SHOULD CHECK WHETHER THEY'RE IN A NATIONAL PARK...
SURE, EASY GIS LOOKUP. GIMME A FEW HOURS.
... AND CHECK WHETHER THE PHOTO IS OF A BIRD.
I'LL NEED A RESEARCH TEAM AND FIVE YEARS.

IN CS, IT CAN BE HARD TO EXPLAIN THE DIFFERENCE BETWEEN THE EASY AND THE VIRTUALLY IMPOSSIBLE.