X-Ray Anomaly Detection using CNNs

Team PerXeptron
ITSP 2020

Richeek Das
Harshit Varma
Ankit Kumar Misra
Prapti Kumar

Idea Behind the Project

Chest X-Ray analysis to detect diseases is a manually time-taking task and requires a lot of skill. If not done properly, deadly diseases may go unnoticed, with severe consequences for the patient.

With this project, we aim to automate this task and **reduce the False Negative Cases around the globe.** We have created a model ensemble that can effectively predict whether a patient has each of 5 diseases, and we plan to deploy it completely on our website soon.

Workflow

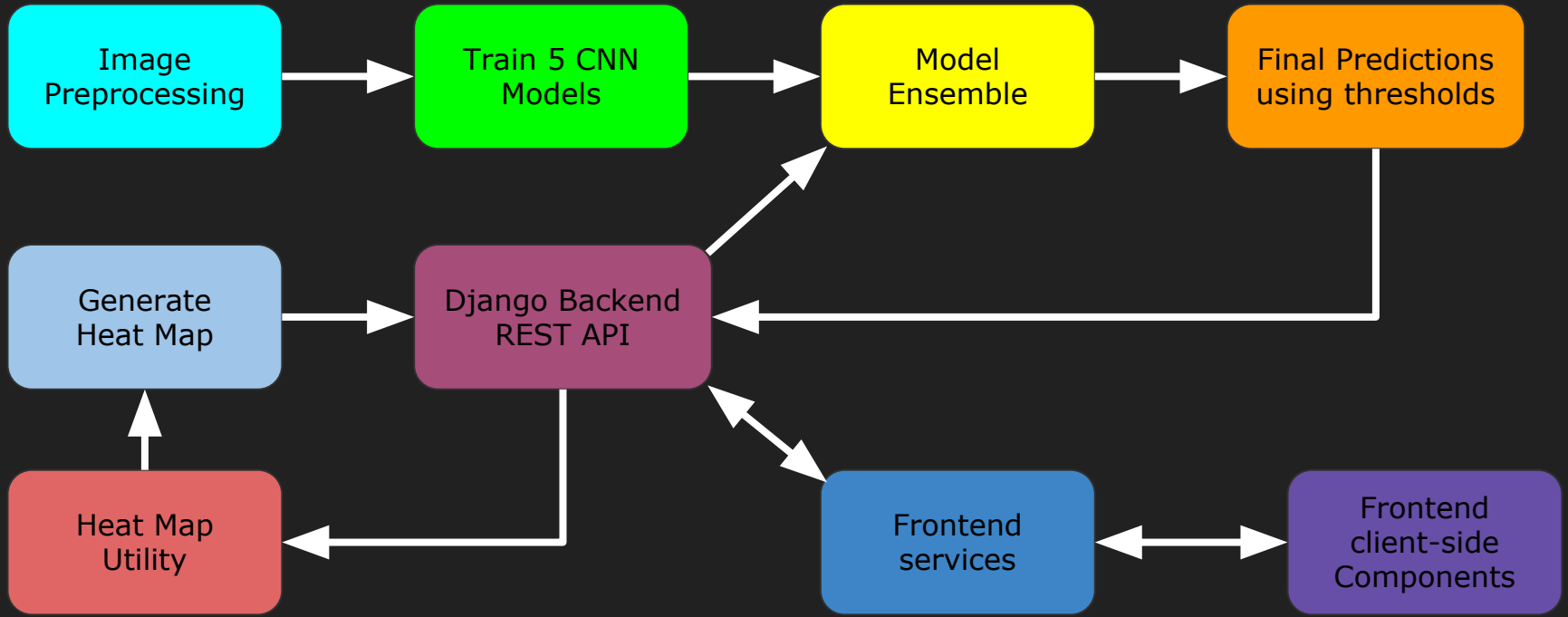
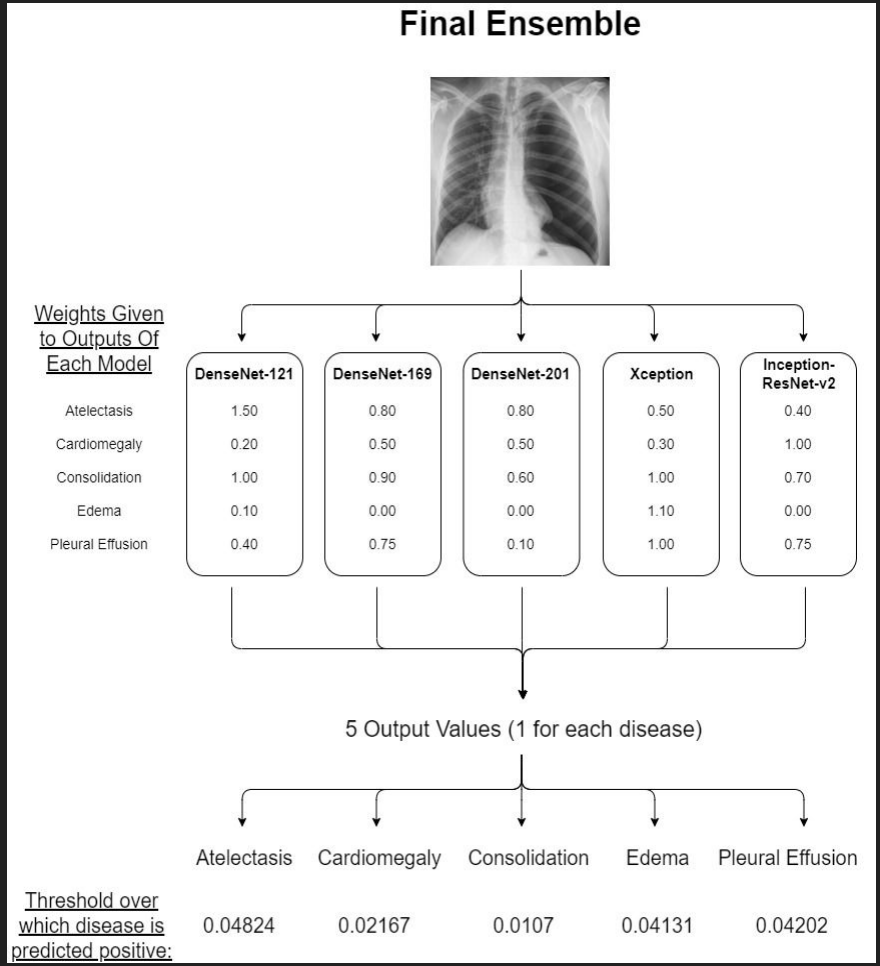


Image Preprocessing
Train 5 CNN Models
Model Ensemble
Final Predictions using thresholds
Generate Heat Map
Django Backend REST API
Heat Map Utility
Frontend services
Frontend client-side Components

Project Details

- Models and the Ensemble:
  - We have used the following models for making the predictions: Xception, Inception-ResNet-v2, DenseNet-121, DenseNet-169, and DenseNet-201.
  - After getting the individual predictions we combine them via a weighted average and get the final ensembled predictions (probabilities).
  - We have then maximized the Youden's Index to find the appropriate thresholds for making the predictions. Using these thresholds, we get the final predictions (binary).



- Django Backend and REST API :
  - We have used Django REST Framework on top of Django 3.0.7 for the Backend.
  - It is a RESTful API which, handles all the login, authentication, signup, and prediction in a 2 layer deep abstracted module.

- Front-end Web Application:
  - After discussion and mutual agreement, we decided to use Angular 9 for the front end part. Angular Material was also used for some elements.
  - Home page, Upload page, Login page, Results page and Profile page was created and routing between all of the pages was enabled.
  - The web app was made responsive and is deployed on github pages.

Web App

perXeptron

Upload XRay Anonymously | Login | Sign Up

perXeptron

What do we aim to do?

We are living in unprecedented times. The COVID-19 pandemic brought a struggle to our already overwhelmed healthcare systems and billions of us have to stay at home to practice social distancing.This is our effort to help the society in these times of need. Chest radiography is one of the most common examinations in the diagnosis of several common thoracic diseases. The failure to detect such diseases quickly and accurately can have severe consequences. We aim to solve this problem with a tool that helps detect common thoracic problems by deploying the use of Neural Networks. This tool will be able to detect as many as 5 common thoracic diseases.

?

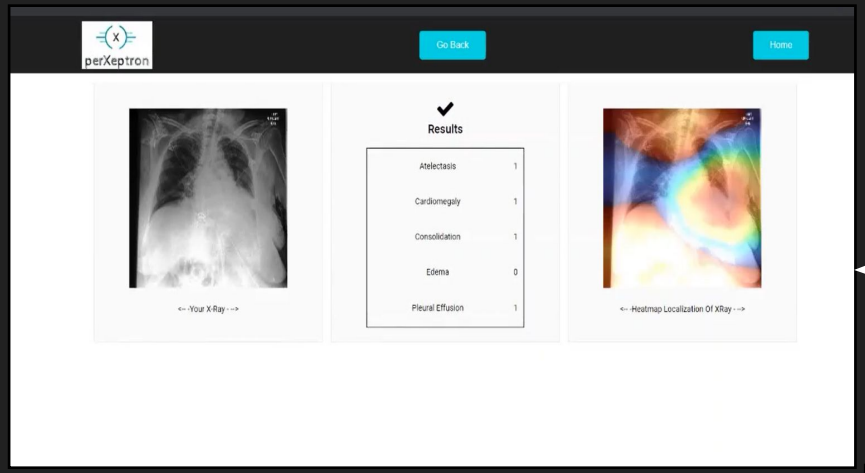
How to Use It?

We have created an easy to use interface for the diagnosis. All you have to do is upload a picture of your X Ray File in PNG, JPEG or JPG format, After which you will be redirected to the diagnostic results page.

Upload XRay

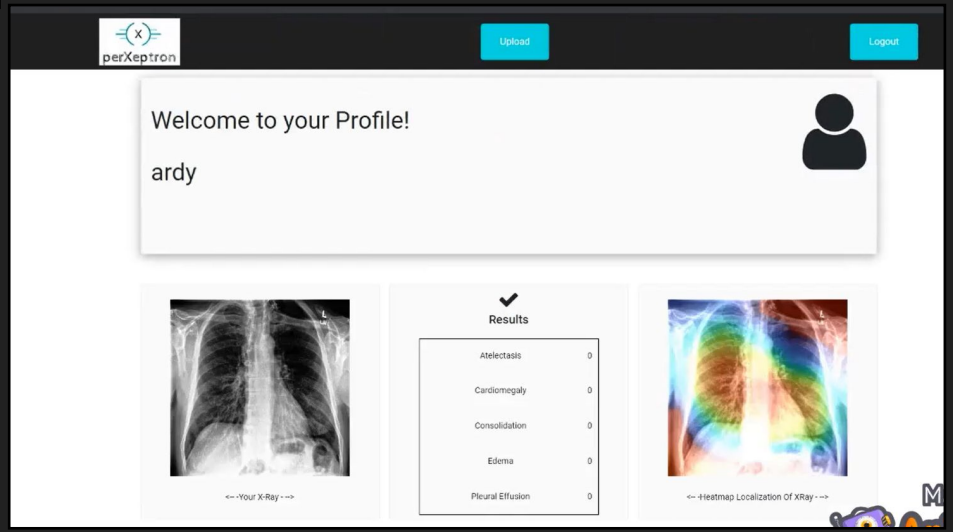
To save your data for future use,you can also register with us:

Register

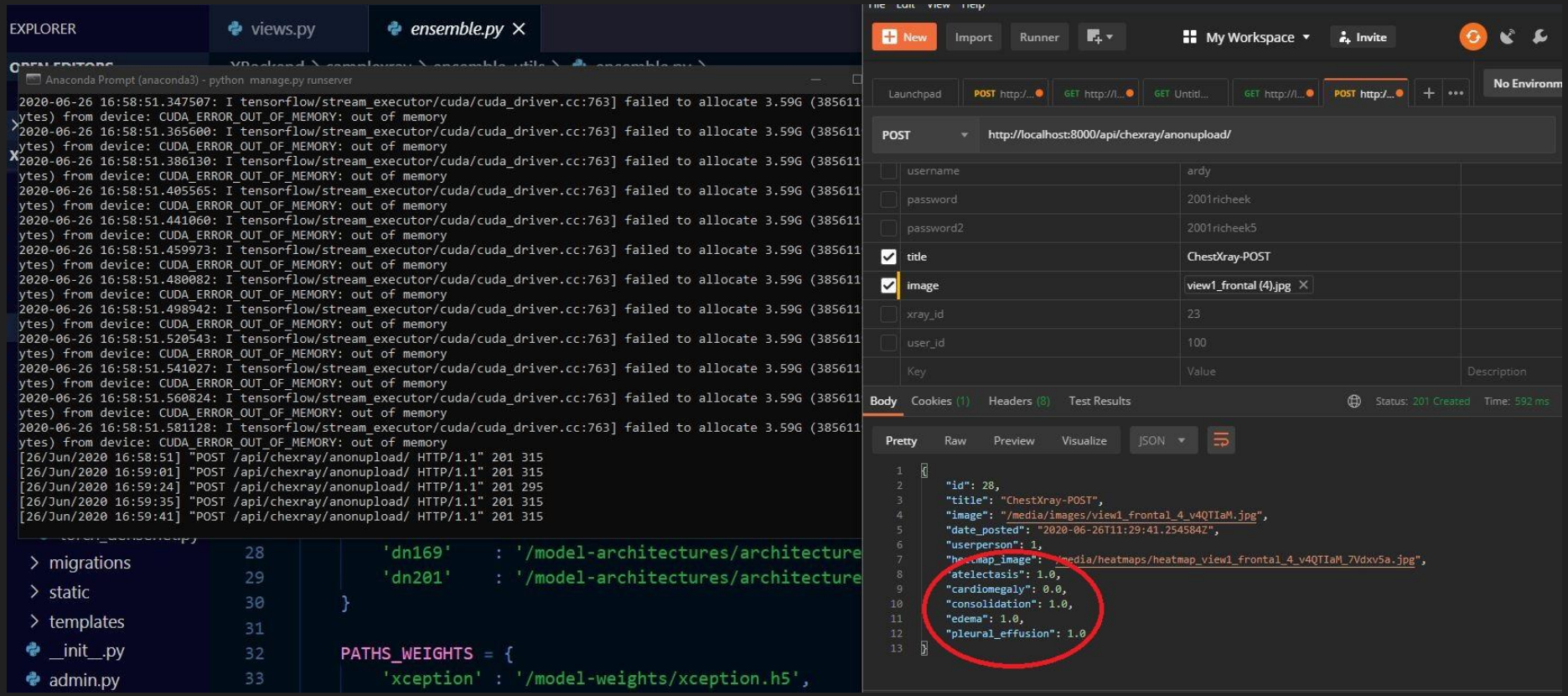


Results page

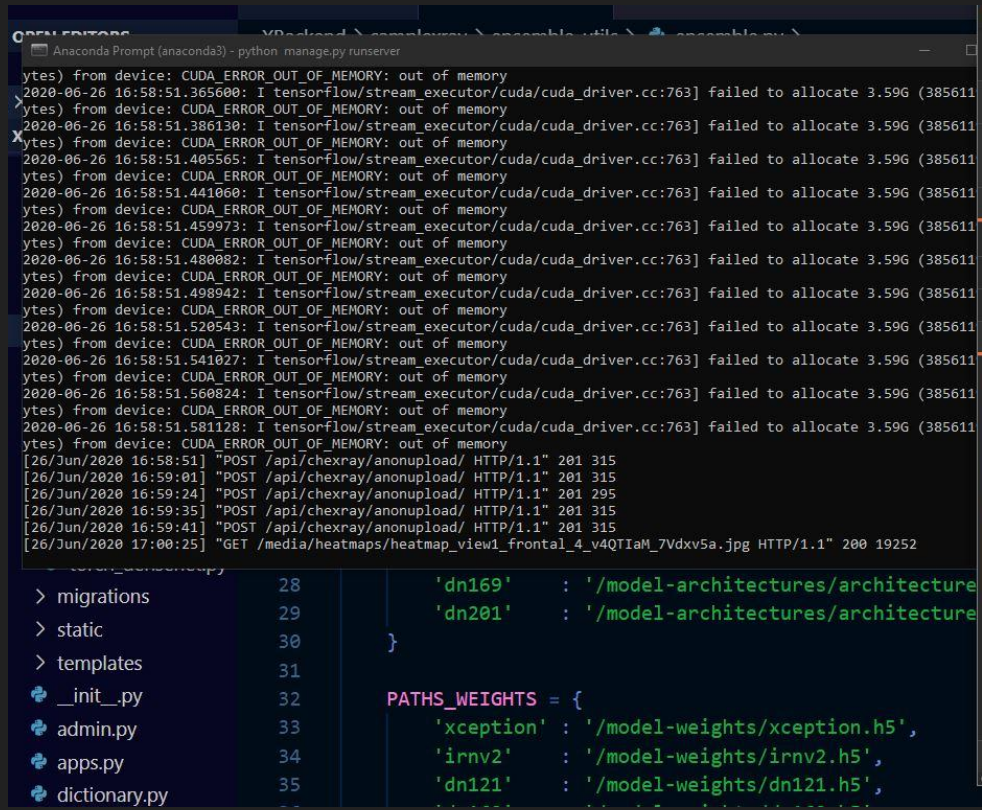
Profile page



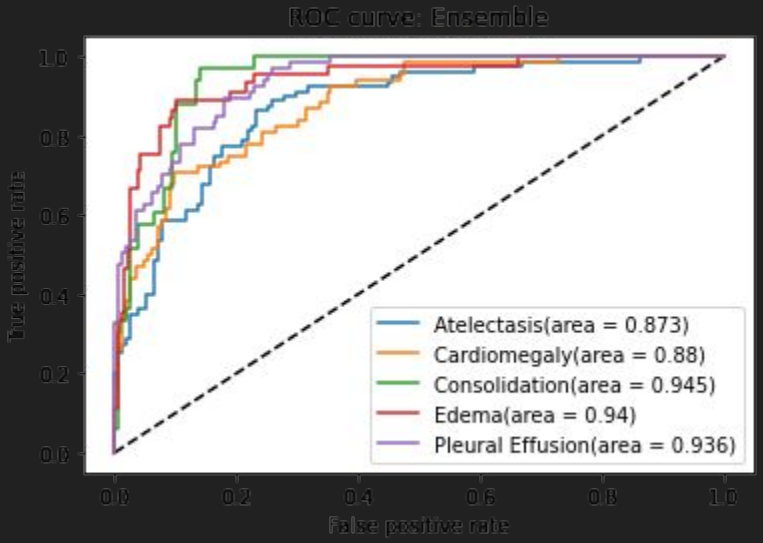
Open-API

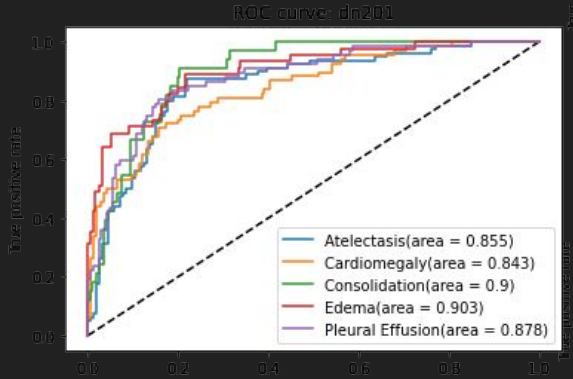

EXPLORER
views.py
ensemble.py
Anaconda Prompt (anaconda3) - python manage.py runserver
2020-06-26 16:58:51.347507: I tensorflow/stream_executor/cuda/cuda_driver.cc:763] failed to allocate 3.59G (385611
ytes) from device: CUDA_ERROR_OUT_OF_MEMORY: out of memory
2020-06-26 16:58:51.365600: I tensorflow/stream_executor/cuda/cuda_driver.cc:763] failed to allocate 3.59G (385611
ytes) from device: CUDA_ERROR_OUT_OF_MEMORY: out of memory
2020-06-26 16:58:51.386130: I tensorflow/stream_executor/cuda/cuda_driver.cc:763] failed to allocate 3.59G (385611
ytes) from device: CUDA_ERROR_OUT_OF_MEMORY: out of memory
2020-06-26 16:58:51.405565: I tensorflow/stream_executor/cuda/cuda_driver.cc:763] failed to allocate 3.59G (385611
ytes) from device: CUDA_ERROR_OUT_OF_MEMORY: out of memory
2020-06-26 16:58:51.441060: I tensorflow/stream_executor/cuda/cuda_driver.cc:763] failed to allocate 3.59G (385611
ytes) from device: CUDA_ERROR_OUT_OF_MEMORY: out of memory
2020-06-26 16:58:51.459973: I tensorflow/stream_executor/cuda/cuda_driver.cc:763] failed to allocate 3.59G (385611
ytes) from device: CUDA_ERROR_OUT_OF_MEMORY: out of memory
2020-06-26 16:58:51.480082: I tensorflow/stream_executor/cuda/cuda_driver.cc:763] failed to allocate 3.59G (385611
ytes) from device: CUDA_ERROR_OUT_OF_MEMORY: out of memory
2020-06-26 16:58:51.498942: I tensorflow/stream_executor/cuda/cuda_driver.cc:763] failed to allocate 3.59G (385611
ytes) from device: CUDA_ERROR_OUT_OF_MEMORY: out of memory
2020-06-26 16:58:51.520543: I tensorflow/stream_executor/cuda/cuda_driver.cc:763] failed to allocate 3.59G (385611
ytes) from device: CUDA_ERROR_OUT_OF_MEMORY: out of memory
2020-06-26 16:58:51.541027: I tensorflow/stream_executor/cuda/cuda_driver.cc:763] failed to allocate 3.59G (385611
ytes) from device: CUDA_ERROR_OUT_OF_MEMORY: out of memory
2020-06-26 16:58:51.560824: I tensorflow/stream_executor/cuda/cuda_driver.cc:763] failed to allocate 3.59G (385611
ytes) from device: CUDA_ERROR_OUT_OF_MEMORY: out of memory
2020-06-26 16:58:51.581128: I tensorflow/stream_executor/cuda/cuda_driver.cc:763] failed to allocate 3.59G (385611
ytes) from device: CUDA_ERROR_OUT_OF_MEMORY: out of memory
[26/Jun/2020 16:58:51] "POST /api/chexray/anonupload/ HTTP/1.1" 201 315
[26/Jun/2020 16:59:01] "POST /api/chexray/anonupload/ HTTP/1.1" 201 315
[26/Jun/2020 16:59:24] "POST /api/chexray/anonupload/ HTTP/1.1" 201 295
[26/Jun/2020 16:59:35] "POST /api/chexray/anonupload/ HTTP/1.1" 201 315
[26/Jun/2020 16:59:41] "POST /api/chexray/anonupload/ HTTP/1.1" 201 315
migrations
static
templates
__init__.py
admin.py
'dn169' : '/model-architectures/architecture
'dn201' : '/model-architectures/architecture
}
PATHS_WEIGHTS = {
'xception' : '/model-weights/xception.h5',
New
Import
Runner
My Workspace
Invite
Launchpad
No Environm
POST
http://localhost:8000/api/chexray/anonupload/
username
ardy
password
2001richeek
password2
2001richeek5
title
ChestXray-POST
image
view1_frontal (4).jpg
xray_id
23
user_id
100
Key
Value
Description
Body
Cookies (1)
Headers (8)
Test Results
Status: 201 Created
Time: 592 ms
Pretty
Raw
Preview
Visualize
JSON
{
"id": 28,
"title": "ChestXray-POST",
"image": "/media/images/view1_frontal_4_v4QTIaM.jpg",
"date_posted": "2020-06-26T11:29:41.254584Z",
"userperson": 1,
"heatmap_image": "/media/heatmaps/heatmap_view1_frontal_4_v4QTIaM_7Vdxv5a.jpg",
"atelectasis": 1.0,
"cardiomegaly": 0.0,
"consolidation": 1.0,
"edema": 1.0,
"pleural_effusion": 1.0
}

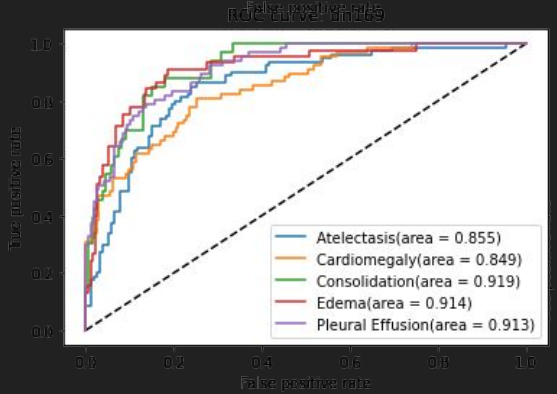
Reading Heat Map from the API Endpoint

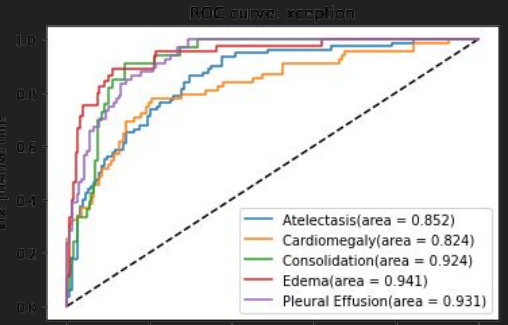


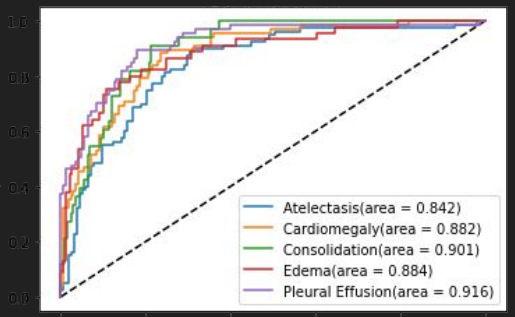
Results (Images/Videos)

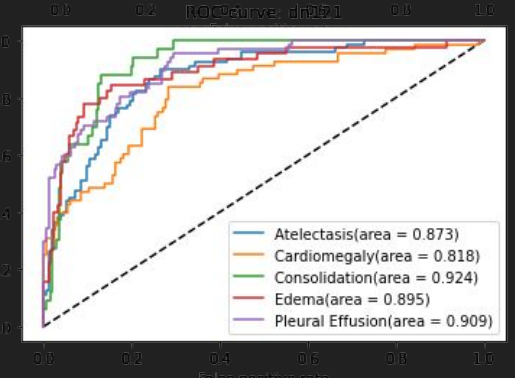












Final Ensemble Results:
Avg AUC: 0.915

Skills Learnt

- Fundamentals of Deep Learning and Computer Vision
- Fundamentals of Web development
- Image Pre-processing Techniques like Histogram Equalization and Center-Cropping
- Transfer Learning
- Model Ensembling
- Experience of building deep learning models on large(~11GB) datasets with limited computational resources.
- Continuous Integration and Deployment

Conclusion/Future plans

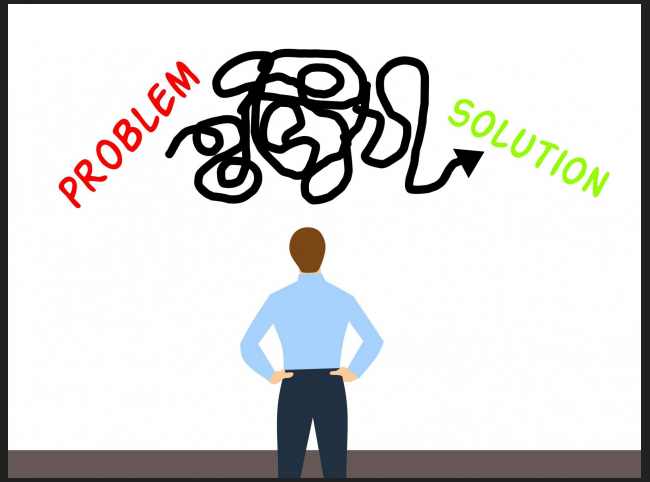
Conclusion:

We were able to get close to the current state-of-the-art (AUC: 0.94) by achieving an AUC of 0.915.

Future Plans:

- Submit our solutions to the official competition and see where we stand.
- Further improving the AUCs by trying out more models, and other approaches which we couldn’t try due to the lack of time.
- Expanding to all 14 diseases instead of the current 5.
- Beautification of the frontend.
- Make the Angular Frontend Cross Platform.
- Expand the Open API, using session authentication for developers.

Challenges faced



- Accuracy and AUC Saturation: After about 25% of the first epoch, the AUC value saturated to 0.70
- Featurewise Centering and Standard normalization: Colab cannot handle (RAM gets exhausted) the entire training set at once for fitting the ImageDataGenerators.
- Colab runtime disconnect after a period of inactivity
- Google Drive’s download limit exceeds after some time.
- 12 hour limit on GPU usage (Some models are very big and require about 1.5-2 hrs for a single epoch)
- Running heavy ensemble models on local servers : Faced problems while running a 5 Keras Model Ensemble alongside a PyTorch Model in the Backend.