Summer Internship Program 2020-21

ART PARK

Investigations on Autonomous vehicle tracking and Medical Imaging

By

Rithik Mali

1. Model Predictive Control for trajectory tracking

Abstract

Model predictive control (MPC) is an advanced method of process control that is used to control the given plant while satisfying a set of constraints. In this work,we implement this method to efficiently track a robot along the reference trajectory while adhering to the given motion constraints. We also investigate the effect of tunable MPC parameters and report the corresponding results.

Introduction

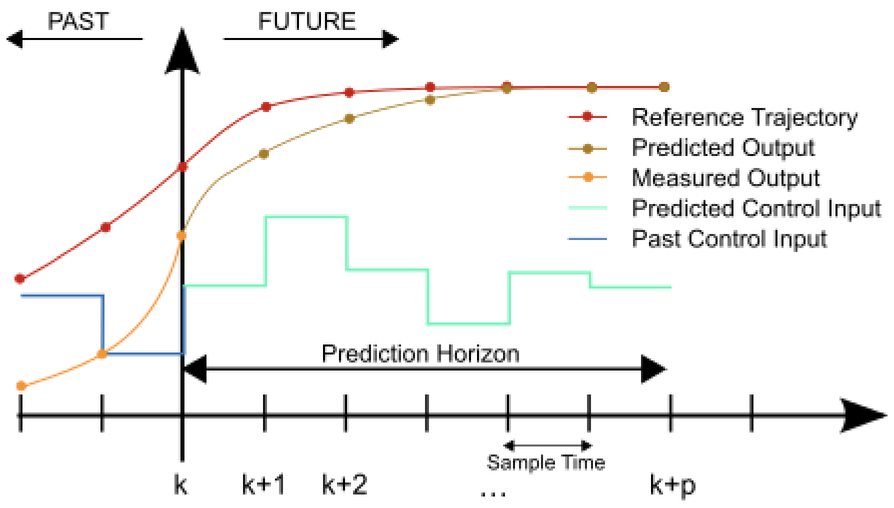
Trajectory tracking control algorithms are needed to precisely realize the desired trajectory of a wheeled robot. **Model Predictive Control (MPC)** is a general method to determine control inputs while satisfying a set of constraints. Assuming a kinematics motion model for the vehicle, the trajectory for a *prediction horizon* can be computed. The optimization problem involves choosing the control inputs that selects the least-cost path.
We implemented the MPC framework as it has the following advantages over other trajectory tracking methods:

- MPC provides the possibility to flexibly introduce system constraints
- Allows us to implicitly observe the unknown system dynamics. As a result, our decisions can factor in non-idealities such as time delays in motor response, surface friction, contact area of the tyre, etc.

In short, the advantages of predictive control are to reduce the variance (more precision and optimality), to be able to change the setpoint (more degree of freedom in the control of the systems.

We develop the necessary software framework to study MPC for the trajectory tracking problem. We also provide the ability to finetune the configurations of various parameters through a configuration template. We also provide a visualizer tool to analyze the results and draw insights.

Model Predictive Control



MPC is based on iterative, finite-horizon optimization of a robot model. At time *t* the current robot state is assumed to be available to us. For instance, this can be from a localization module that fuses several on-board sensors. A cost-minimizing control strategy is computed via a numerical optimization technique for a relatively short time-horizon in the future:*[t,t+T]*. Only the first step of this control strategy is implemented. Then the robot state is measured again and the calculations are repeated starting from this new robot position. In turn, this yields a new control action. The prediction horizon keeps shifting forward and for this reason MPC is also called **receding horizon control**. In practice, MPC control is observed to produce near-optimal policies in the absence of system dynamics knowledge.

At each time step, the reference trajectories of a vessel are assumed to be known over a finite time horizon; the MPC controller computes the optimal velocity and steer the robot needs in order to efficiently track the reference trajectory. Since the optimal control problem is solved online, it is straightforward to add state and control saturation constraints as penalty functions to the cost function. For the optimization part, the algorithm can make use of any off-the-shelf solver. For the purpose of this work, the algorithm makes use of SLSQP (Sequential Least Squares Programming).

The cost function of the algorithm focuses on the deviation from the original path, difference from target velocity, angle of heading, acceleration, angular velocity and jerk.

Implementation details

Initialize the variables, `x`, `y` and `yaw` to their initial positions and their derivatives to zero

1. Plan a trajectory by formulating the non-linear optimization problem taking into account the kinematic and possible velocity/ curvature constraints.
2. Execute the first step of the planned trajectory
3. Keep iterating the trajectory planning until the goal state is reached

This algorithm is quite robust to Gaussian noise at different speeds and follows the reference path well. At high speeds, the MPC algorithm starts to deviate slightly from the reference path in order to maintain the target speed. It must be noted that a separate collision avoidance algorithm is needed.

Cost function

The cost function used is

$$C = \sum_{i=1}^{N} K_X||x_i - x_i^{ref}||^2 + K_Y||y_i - y_i^{ref}||^2 + K_V||v_i - v_{max}||^2 + K_{yaw}||\theta_i - \theta_i^{ref}||^2$$
$$+K_{a,lat}||a_i||^2 + K_{a,long}||a_i^y||^2 + K_J||\ddot{v}_i||^2$$

where K_X, K_Y, K_V, K_{yaw}, $K_{a,lat}$, $K_{a,long}$,K_J are tunable weight parameters.

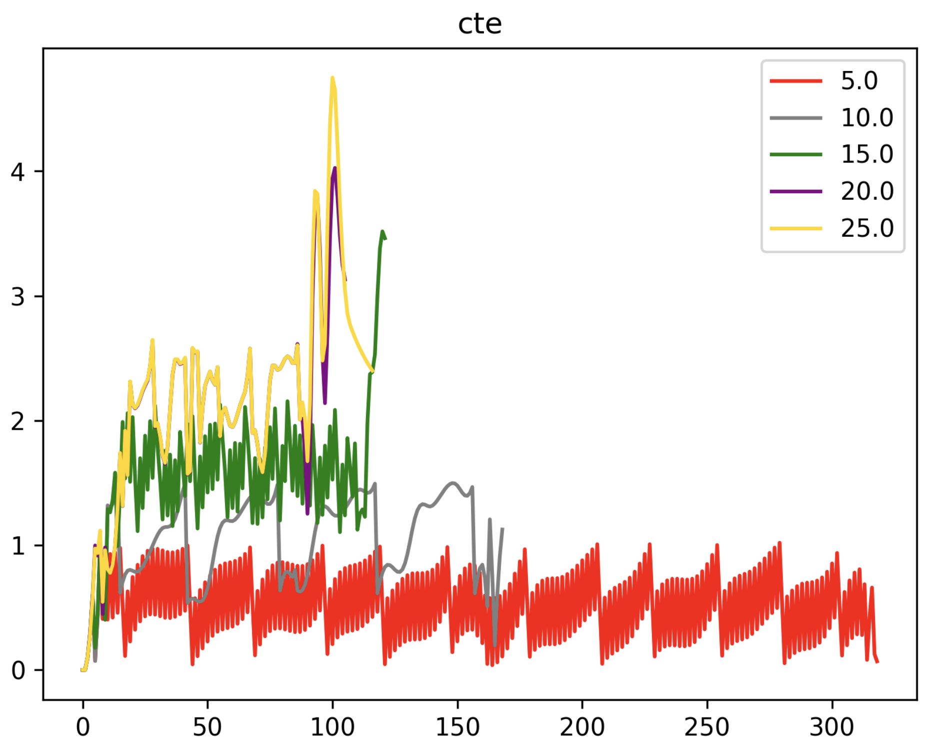
$||x - x_{ref}||$and $||y - y_{ref}||$, $||\theta - \theta_{ref}||$ are the deviations between current and reference trajectories.

$||v - v_{max}||$ is the term to improve linear velocity of the robot. $||a||$, $||a^y||$, $||\ddot{v}||$ are the terms to moderate lateral acceleration, longitudinal acceleration and jerk, respectively.

Results

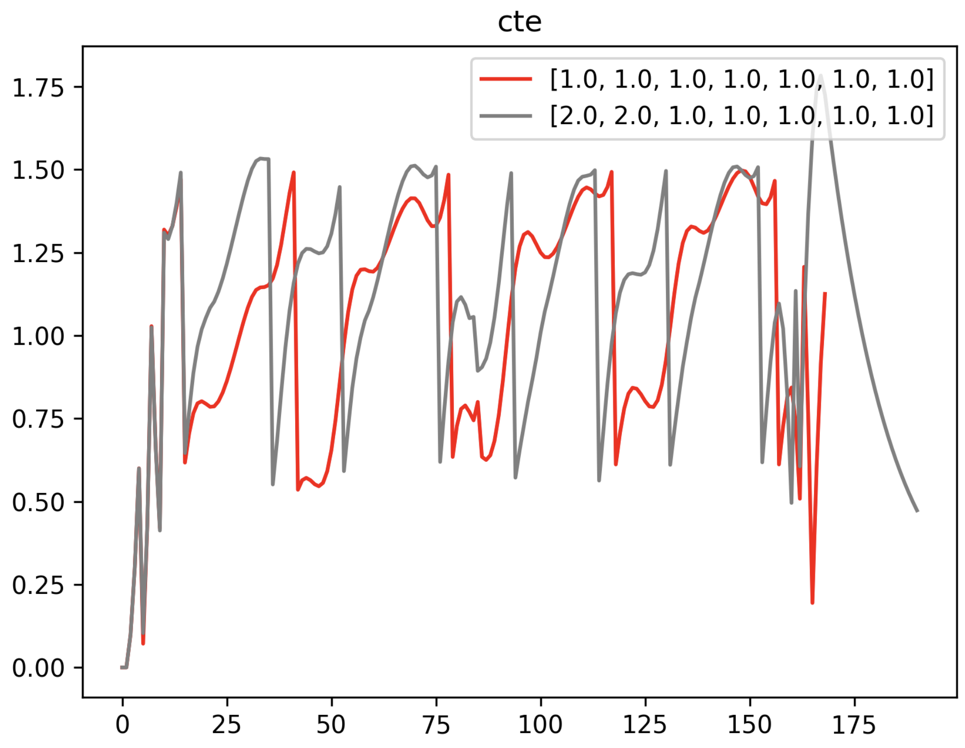
Effects of different parameters on MPC

- We tried changing the maximum permissible velocity



We observe that moving at lower velocities give better accuracy for a fixed time horizon. This is not surprising as we should expect to execute the MPC algorithm at a faster time scale as velocities increase, in order to obtain similar accuracy

- Next we experimented with the tunable weight parameters in the cost function

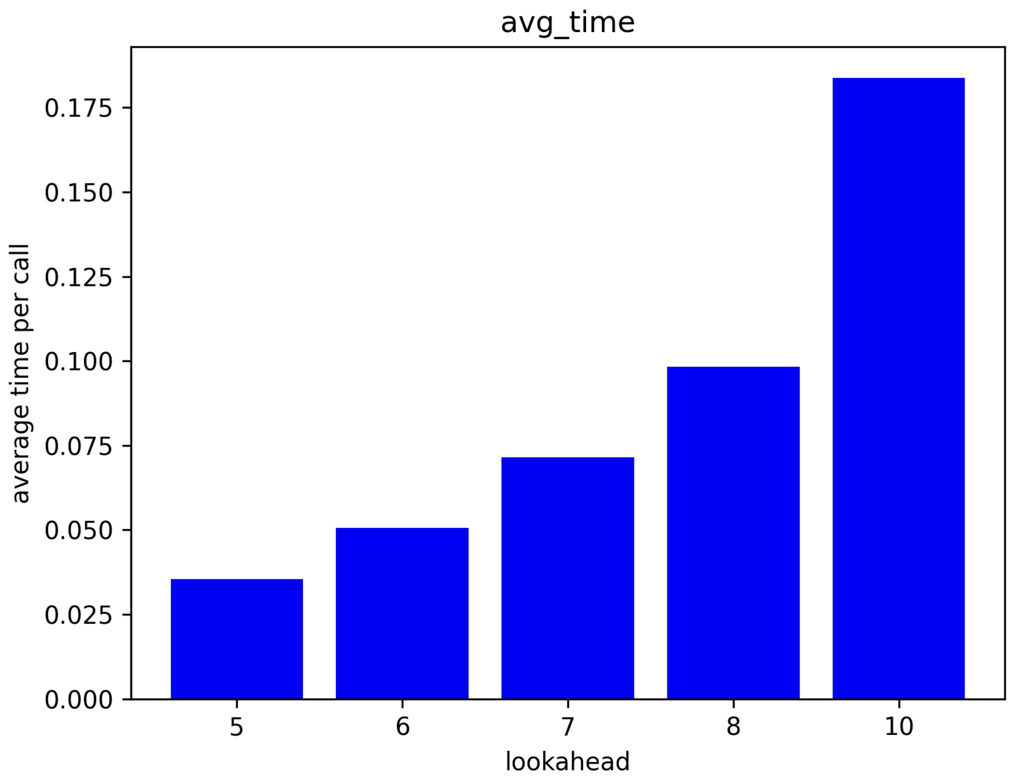


We observed that

- giving equal weight to every parameter in the cost function gave the best result.
- Higher weight to x, y increased CTE(Cross Track Error) at the end.

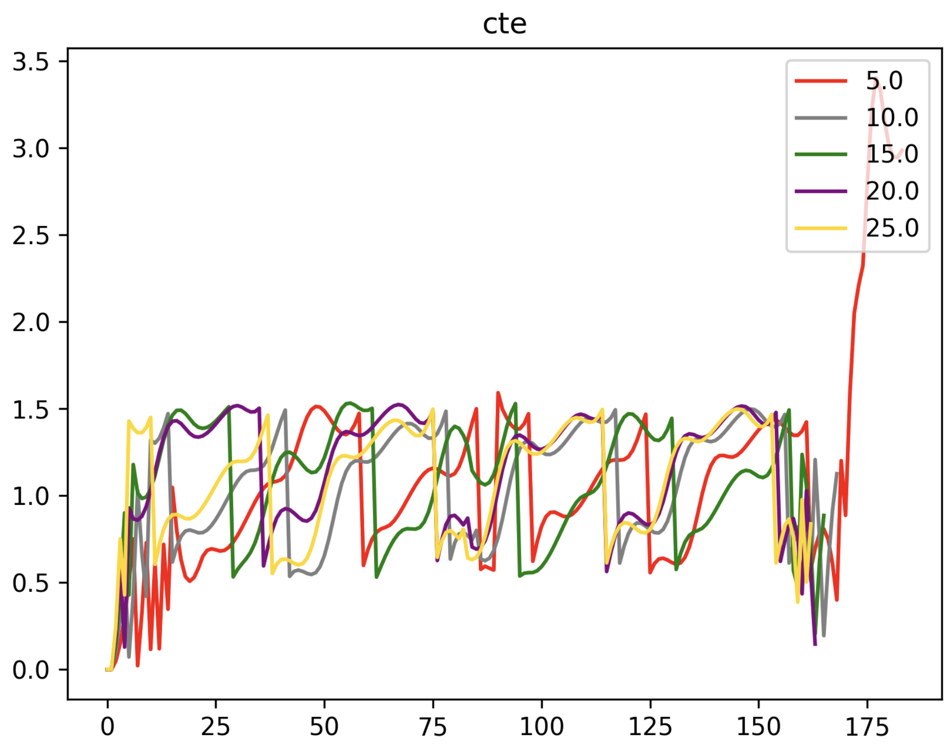
  We have not been able to reason the observed trend

- Lookahead time steps:



  - bigger lookahead ⇒ higher average time per call to solve().

- Acceleration:



- higher is better as we can vary the velocity faster ⇒ decrease in CTE.
- Increasing acceleration will introduce a lot of jerk (can be taken care by increasing the weight of jerk cost)

References

- Model Predictive Control - Wikipedia
- MathWorks - Understanding Model Predictive Control
- PythonRobotics: a Python code collection of robotics algorithms

2. Classifying WhatsApp X-ray images for CoVID detection

Abstract

In this project we explored and experimented with various techniques for classifying X-ray images for CoVID detection. This classification was separated into 2 separate problems, classifying whether a given image is an X-ray or not and classifying whether the image classified as X-ray is a usable X-ray. These classification models will be used in the preprocessing pipeline to filter out unusable images from being passed to the CoVID inference model of XraySETU. We explored the following pre-trained models for the problem on hand: EfficientNet, BigTransfer and SimCLRv2. We observed that SimCLRv2 produced the best results for both the classification problems.

Introduction

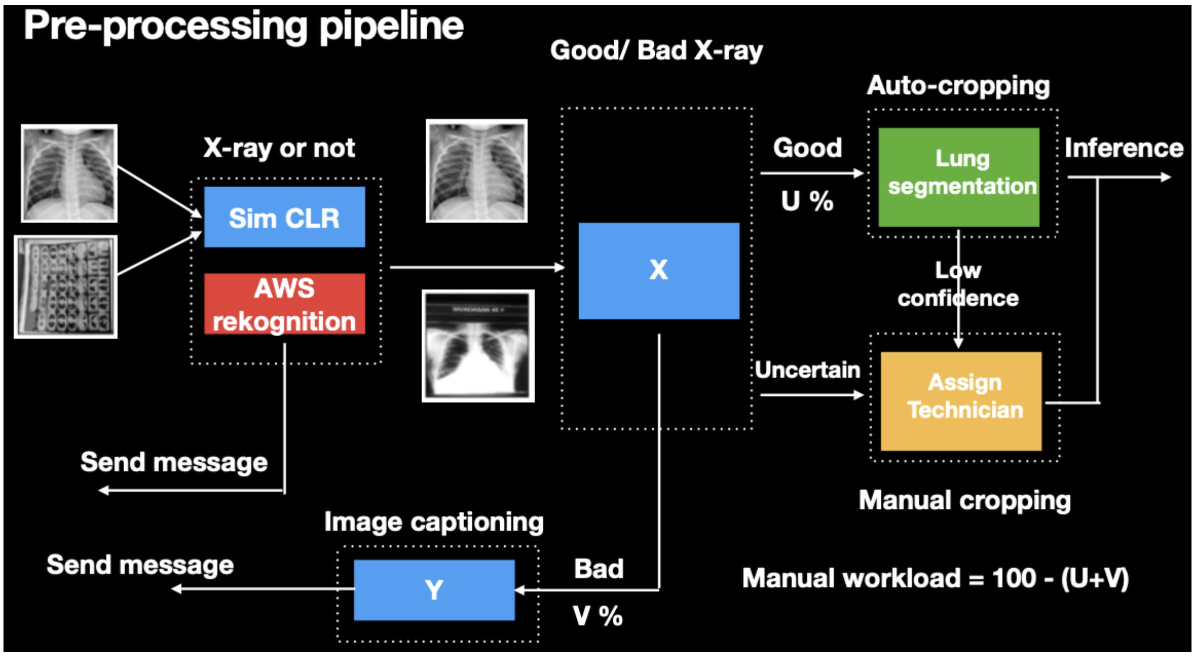
The XraySetu project is a joint initiative between ARTPARK, Niramai and IISc. The objective is to provide AI-driven Xray prediction of the onset of CoVID. This in turn can aid in early intervention. Built on the back of 125000 Xray images data over last 10 months, the AI algorithms are working even with low resolution Whatsapp images.

The XraySetu platform only accepts chest Xray. So any other image submitted by the end user is to be rejected. This filtering of images is currently being done by onsite technicians who manually go through each image. Additionally, any photos with chest X-ray in which the chest area is not properly visible, improper lighting, objects in front / behind the of X-ray, extremely high / low contrast, X-rays with improper light source etc. are also to be identified and rejected by the technicians.

As the XraySetu service scales, the number of images that the technicians will have to classify will also increase. The increase in manual burden could result in slow turnover and errors. The objective of our problem is to

- Classify given image as X-rays or not
- Classify an X-ray as a good/usable X-ray or bad/unusable X-ray.

Pre-processing Pipeline



The above image shows the flow of the image in the new Pre-processing pipeline that is in development. Before the image from WhatsApp reaches the Inference model, it has to go through multi-stage filtering and an auto cropping model to ensure that the image is a chest X-ray and only the chest area is presented to the inference model

Here we focus on the first model SimCLRv2 that will filter out all the non X-ray images and pass only the X-ray images to the second filter model. If the image is not an X-ray then it is rejected. We also compare the performance with an off-the-shelf AWS rekognition

The second model is an Ensemble model of various classifiers that will tell us whether the images identified as X-ray by the previous model in the pipeline is a good X-ray image(chest X-ray taken in good lighting) or a bad X-ray image(improper lighting, improper contrast, non chest X-rays). Depending on the confidence of the prediction one of the following will happen:

- If this model says with a high confidence that the image is a good chest X-ray, then it is passed on to the auto cropping model that will crop the image keeping only the chest part and forwards it to the inference model.
- If the model is uncertain with the confidence of the prediction that image is forwarded to a Technician to verify and crop the image manually.
- If the model has a high confidence that the image is a bad X-ray, then it rejects the image and provides a message that has instructions which will help take a good image of the X-ray.

If the auto cropping model outputs an image with low confidence then it is again forwarded to the technician to crop it manually.

By forwarding uncertain images to technicians instead of the user, we make sure that the number of false negatives are kept to the minimum.

Classification with SimCLRv2

Contrastive Learning

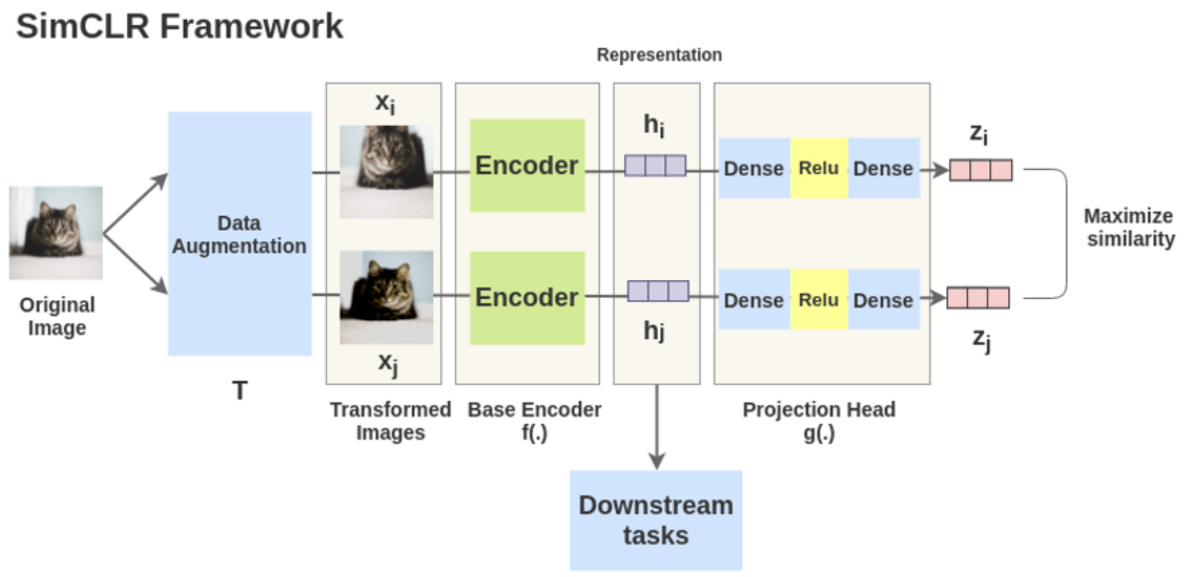
The idea behind contrastive learning is surprisingly simple: the model learns to encode images in a lower dimensional space in such a way that images that are similar semantically will be close to each other in the low dimensional space, and at the same time far away from other images. In contrastive learning, we are trying to find the parameters of our encoder that minimize the contrastive loss on our dataset. Contrastive loss implements the "contrastive" idea verbatim: in the learned representation, we want images that are similar to be close to each other, and images that are different to be far away from each other. The inner working of contrastive learning can be formulated as a score function, which is a metric that measures the similarity between two features. Over this, a softmax classifier can be built that classifies positive and negative samples correctly. A similar application of this technique can be found in the recently introduced framework SimCLR.

Introduction - SimCLR

Google has introduced a framework called "SimCLR" that uses contrastive learning. This framework first learns generic representations of images on an unlabeled dataset and then is fine-tuned with a dataset of labelled images for a given classification task. In the unsupervised representation setting, we do not have access to any labels. In SimCLR, this problem is addressed with data augmentation. It follows the simple reasoning that augmentations of the same image are similar to each other and augmentations of different images are not. SimCLR achieves a big performance gain compared to previous methods in the unsupervised learning space. It is able to achieve similar performance compared to the standard supervised pre-training and fine tuning pipeline. This suggests that the gap between unsupervised and supervised representation learning is largely closed in many vision tasks.

Architecture

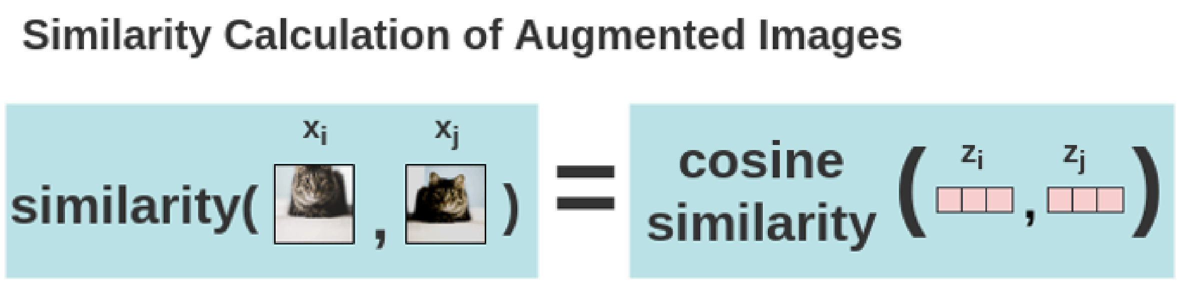
A random transformation is applied on each input image to obtain a pair of two augmented images x_i and x_j. Each image in that pair is passed through an encoder to get representations. Then a non-linear fully connected layer is applied to get representations z. The task is to maximize the similarity between these two representations, z_i and z_j for the same image.



After we get the vectors z_i and z_j for a given image we calculate its similarity score with all the other images in the batch. The similarity metric used here is cosine similarity.

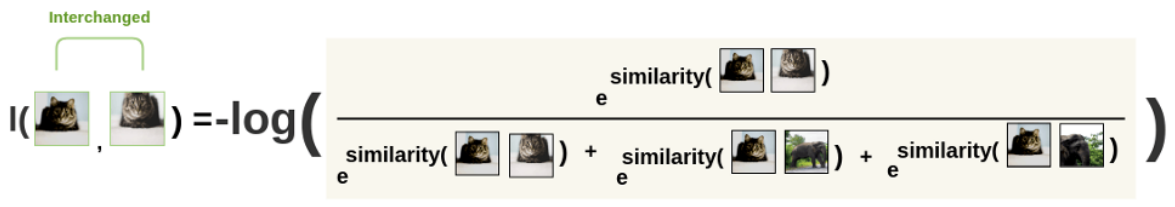
$$S(x_i, x_j) = \frac{z_i z_j^T}{|z_i||z_j|}$$

Below is an illustration of the similarity metric of two cat images x_i and x_j.



After getting the similarity, the model needs to calculate the loss. SimCLR uses a contrastive loss called Normalized Temperature-Scaled Cross-Entropy.

This loss is defined as the negative log of softMax of similarities. This softMax calculation is equivalent to getting the probability of the second augmented cat image being the most similar to the first cat image.



This loss is calculated for the other pairs in the batch and averaged out to give the overall loss.

$$L = \frac{1}{2N} \sum_{k=1}^{N} [l(2k-1, 2k) + l(2k, 2k-1)]$$

Based on this average loss, the encoder and projection head representations improve over time and the representations obtained place similar images closer in the space.

Finetuning the pretrained model

That is how the model is trained from scratch. But for our implementation we use transfer learning from the model that is pretrained with ‘ImageNet’ and finetuned on 100% of the labels. To use the pretrained model on downstream tasks like Image Classification, the saved model is loaded and the top most layer is discarded. We then add a new top layer with the outputs equal to the number of classes we want the outputs to be classified into(i.e. 2). This new model is then trained for a few epochs to finally end up with a model that is finetuned to our requirements.

Results

Classification of image as X-ray or not

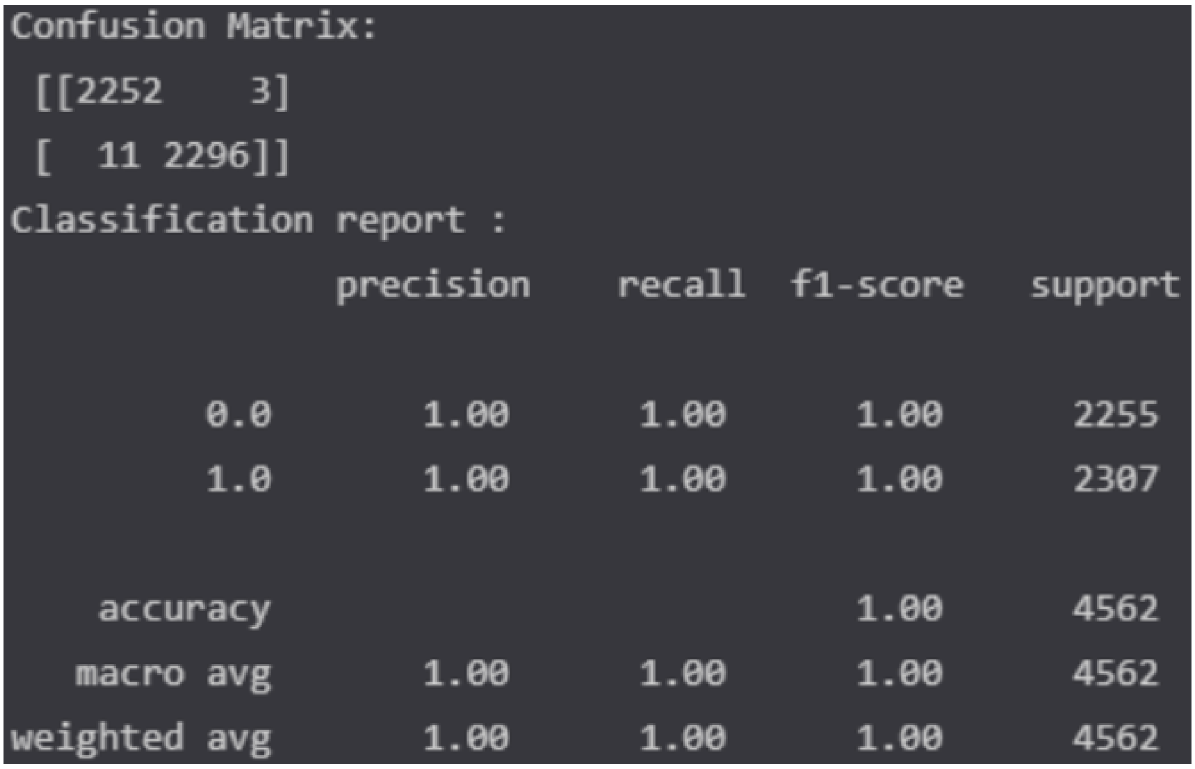
The model achieved exceptional results on the training dataset with accuracy of 100% and loss of 0.

0 => X-ray

1 => not X-ray

1. Test split

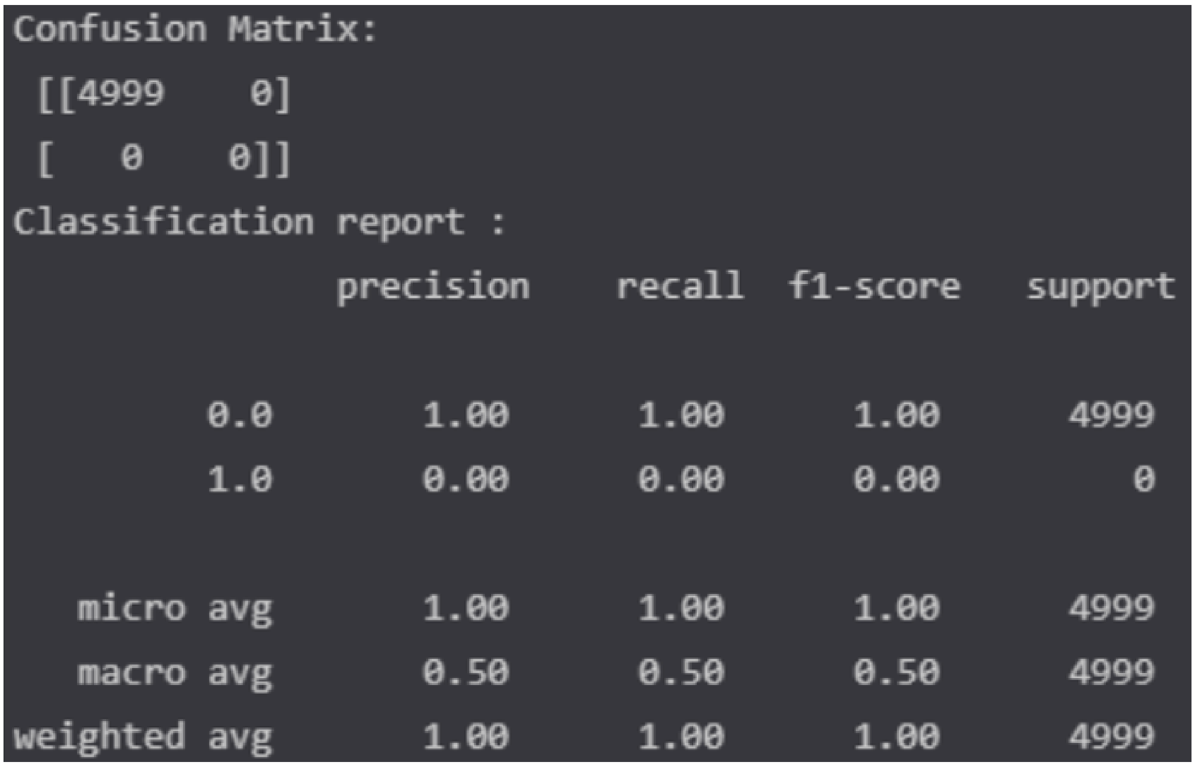
The results on test split of the dataset also shows accuracy of 100%



```
Confusion Matrix:
 [[2252    3]
 [  11 2296]]
Classification report :
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      2255
         1.0       1.00      1.00      1.00      2307

    accuracy                           1.00      4562
   macro avg       1.00      1.00      1.00      4562
weighted avg       1.00      1.00      1.00      4562
```

1. NIH dataset

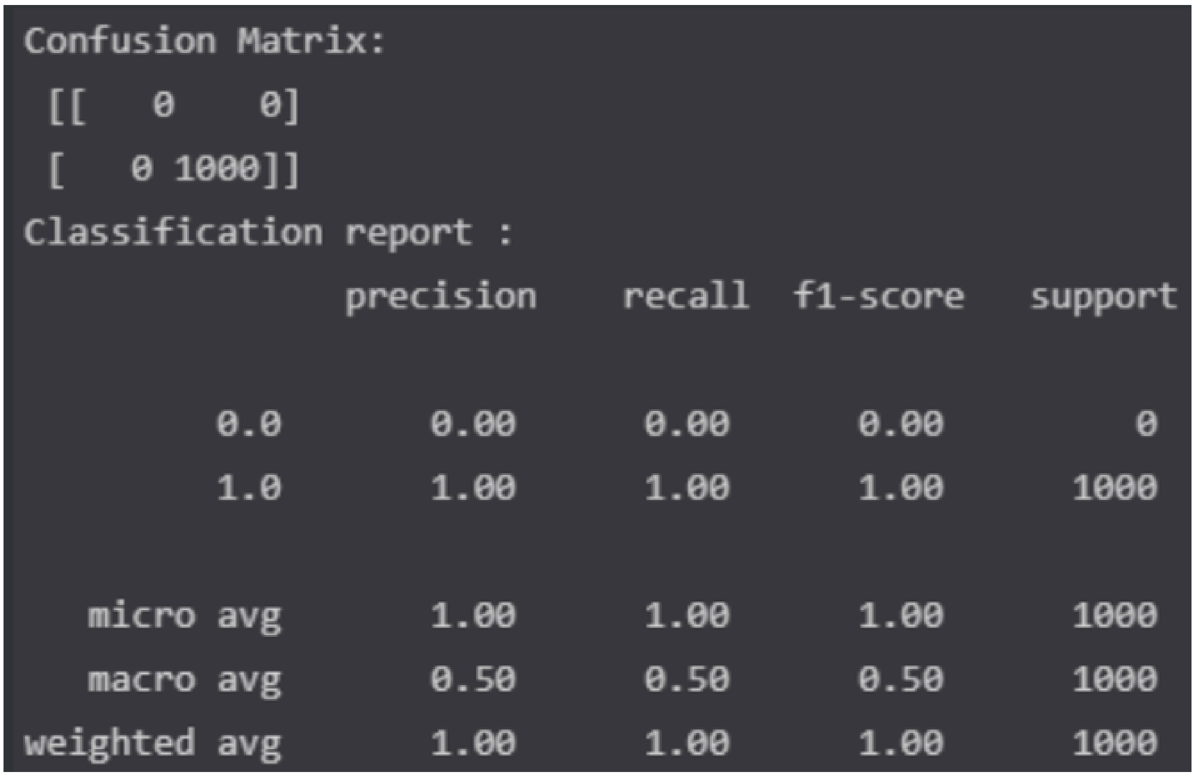
The NIH dataset contains all X-ray images, hence the model classifies all of them correctly as “X-rays”



```
Confusion Matrix:
 [[4999    0]
 [   0    0]]
Classification report :
               precision    recall  f1-score   support


         0.0       1.00      1.00      1.00      4999
         1.0       0.00      0.00      0.00         0


   micro avg       1.00      1.00      1.00      4999
   macro avg       0.50      0.50      0.50      4999
weighted avg       1.00      1.00      1.00      4999
```

1. People Dataset (from Kaggle)

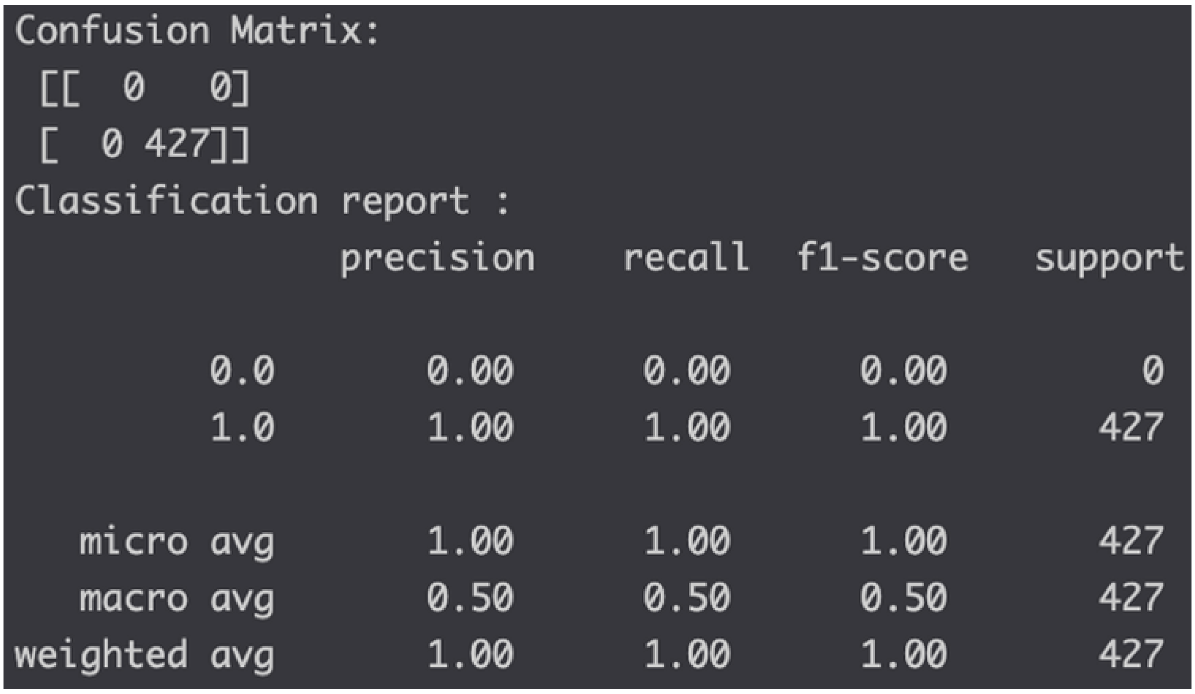
This dataset contains images of some people, hence it classifies these images as “not X-rays”



```
Confusion Matrix:
 [[   0    0]
 [   0 1000]]
Classification report :
               precision    recall  f1-score   support


         0.0       0.00      0.00      0.00         0
         1.0       1.00      1.00      1.00      1000


   micro avg       1.00      1.00      1.00      1000
   macro avg       0.50      0.50      0.50      1000
weighted avg       1.00      1.00      1.00      1000
```

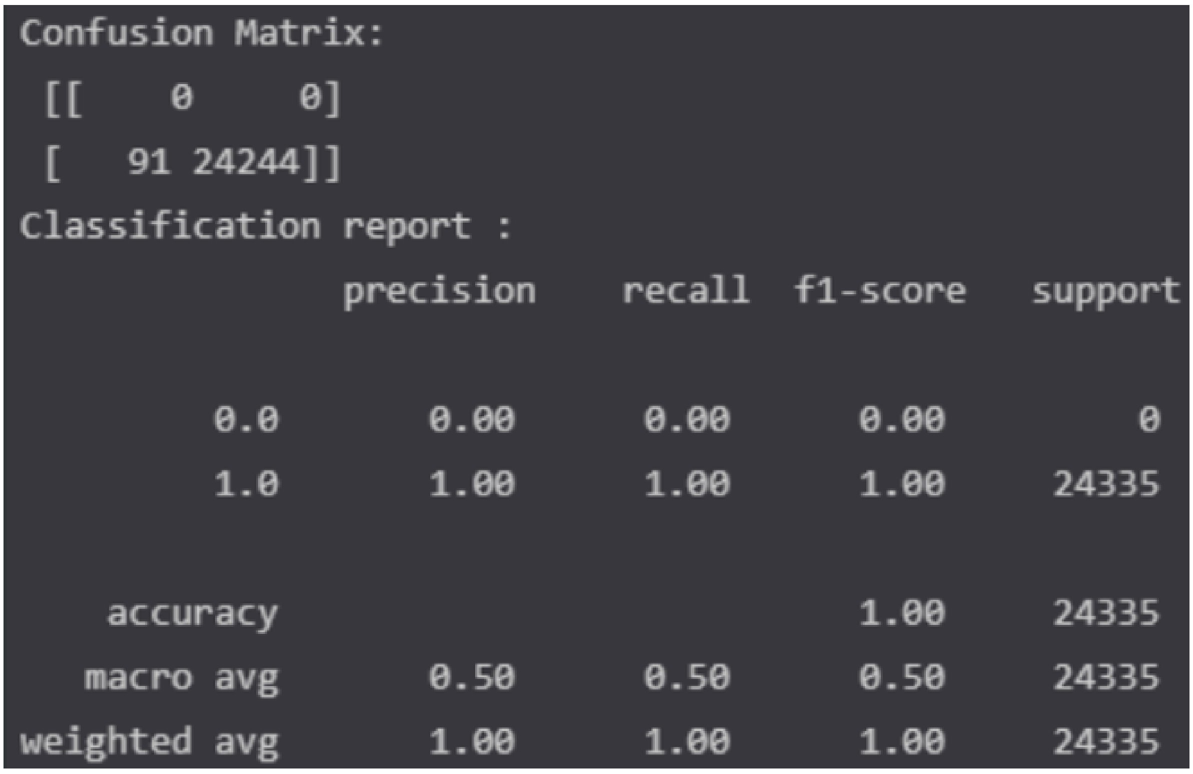
1. Document Dataset (from Kaggle)

Dataset containing a bunch of scanned documents. Model again correctly classifies them as “not X-rays”


```
Confusion Matrix:
 [[  0   0]
 [  0 427]]
Classification report :
              precision    recall  f1-score   support


         0.0       0.00      0.00      0.00         0
         1.0       1.00      1.00      1.00       427


   micro avg       1.00      1.00      1.00       427
   macro avg       0.50      0.50      0.50       427
weighted avg       1.00      1.00      1.00       427
```

1. Intel Image Classification Dataset (from Kaggle)

Dataset contains buildings, forest, glacier, mountain, sea, street. Hence we see that model classifies most of them correctly as “not X-rays”

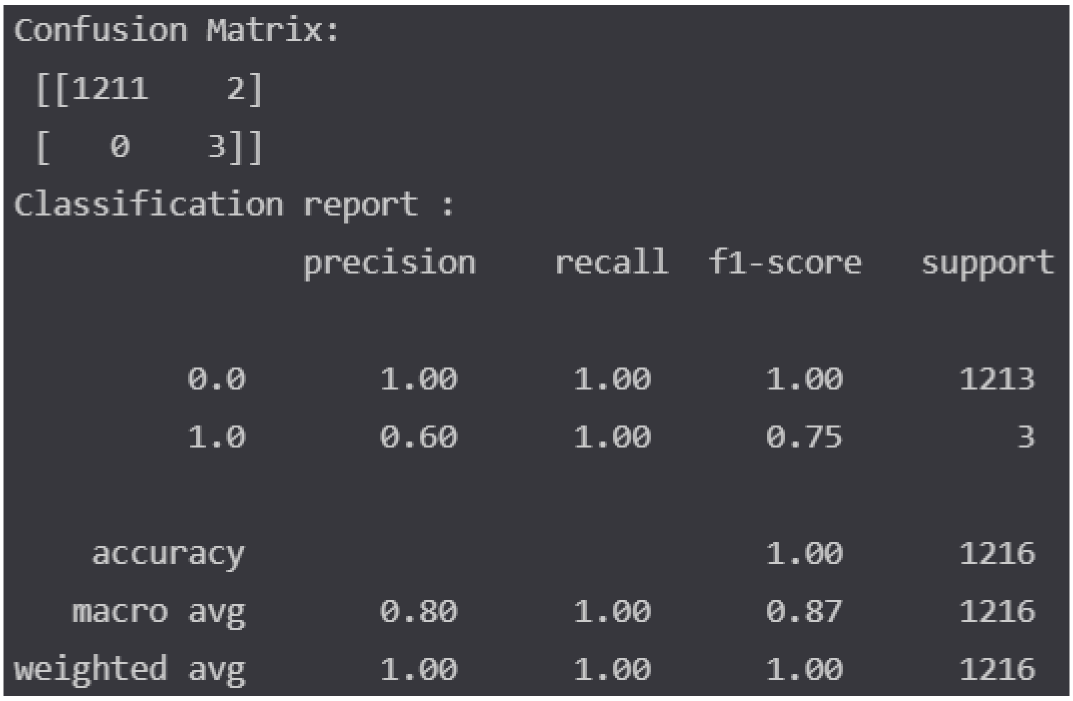

```
Confusion Matrix:
 [[    0     0]
 [   91 24244]]
Classification report :
              precision    recall  f1-score   support


         0.0       0.00      0.00      0.00         0
         1.0       1.00      1.00      1.00     24335


    accuracy                           1.00     24335
   macro avg       0.50      0.50      0.50     24335
weighted avg       1.00      1.00      1.00     24335
```

Comparison with Amazon Rekognition API

We tested the SimCLR model against the Amazon API in a real time setting and found that the simCLR model outperformed the Amazon API and was even able to reject CT scans which was not possible with Amazon Rekognition API.

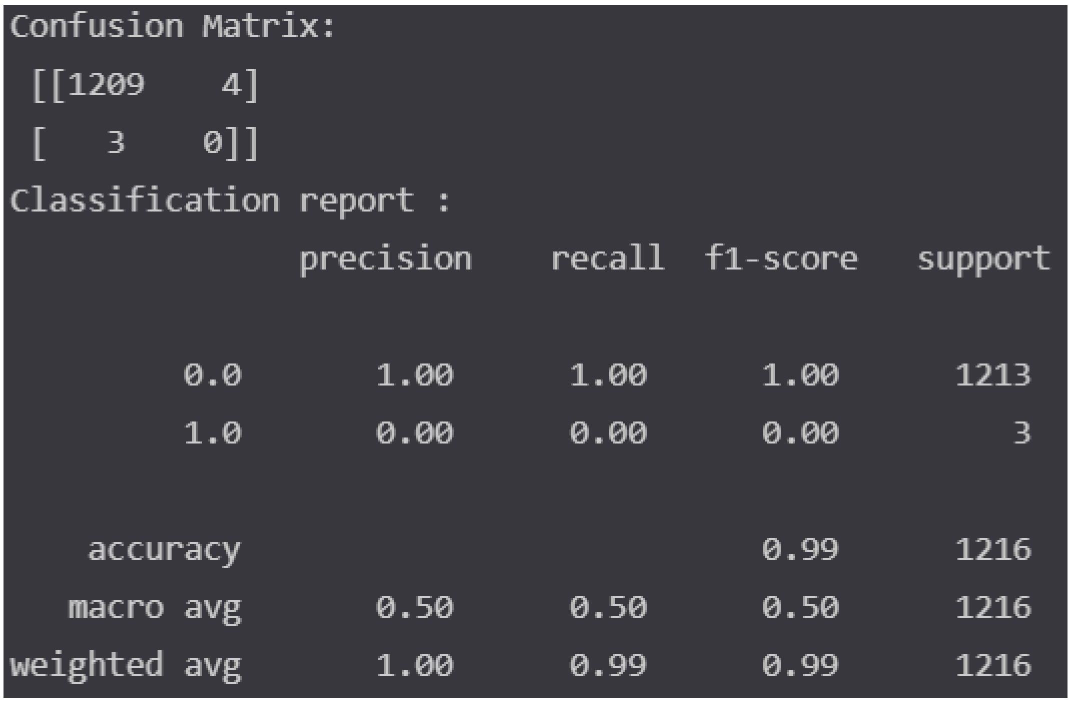
Of the 1216 images that were passed to both, model, and the API, SimCLR model misclassified only 2 incorrectly whereas Amazon Rekognition API got 7 of them wrong.

SimCLRv2



```
Confusion Matrix:
 [[1211    2]
 [   0    3]]
Classification report :
               precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      1213
         1.0       0.60      1.00      0.75         3

    accuracy                           1.00      1216
   macro avg       0.80      1.00      0.87      1216
weighted avg       1.00      1.00      1.00      1216
```

AWS Rekognition *API*



```
Confusion Matrix:
 [[1209    4]
 [   3    0]]
Classification report :
               precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      1213
         1.0       0.00      0.00      0.00         3

    accuracy                           0.99      1216
   macro avg       0.50      0.50      0.50      1216
weighted avg       1.00      0.99      0.99      1216
```

Classification of X-ray as “good” / “bad”

Since this an exceedingly difficult task for any model, we were mainly looking at increasing the recall for positive class so that we do not reject many “good X-rays”, else if the recall is less the end users will be affected and this will in turn reflect on the decrease of userbase.

Out of all the model that we tried out, SimCLR gave the best balance between accuracy and recall with acceptable threshold values.

1. Results on Niramai Dataset

   This dataset is nothing but the images that were collected in real time from the bot deployed on WhatsApp.

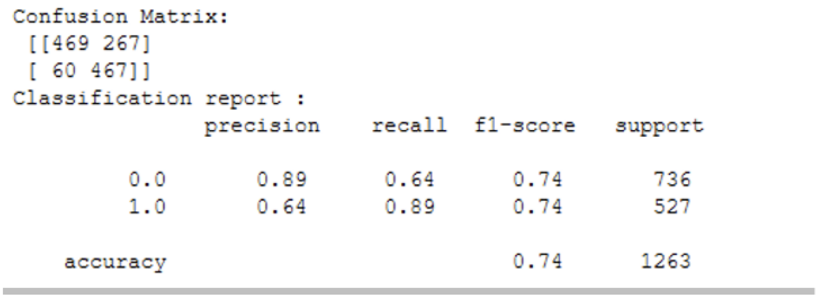
We see accuracy of 74% which is not very high, but the recall is 90% even with a threshold 0.5. This is way better than the other models that we tried. We see that the recall for negative class is 0.64 which means that with the help of this model the technicians will only have to go through 40% of the actual data instead of 100% of the data, thus saving a lot of time.

Niramai Dataset

Training set: 5100

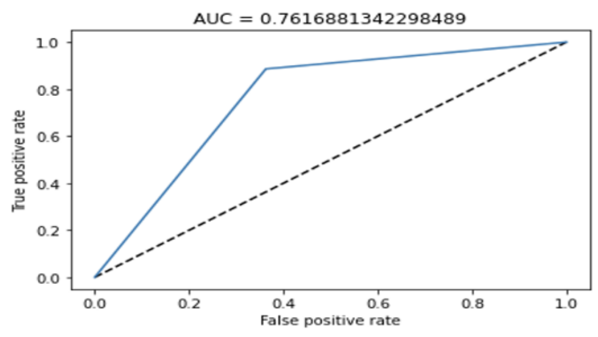
Testing set: 1200

Threshold = 0.5



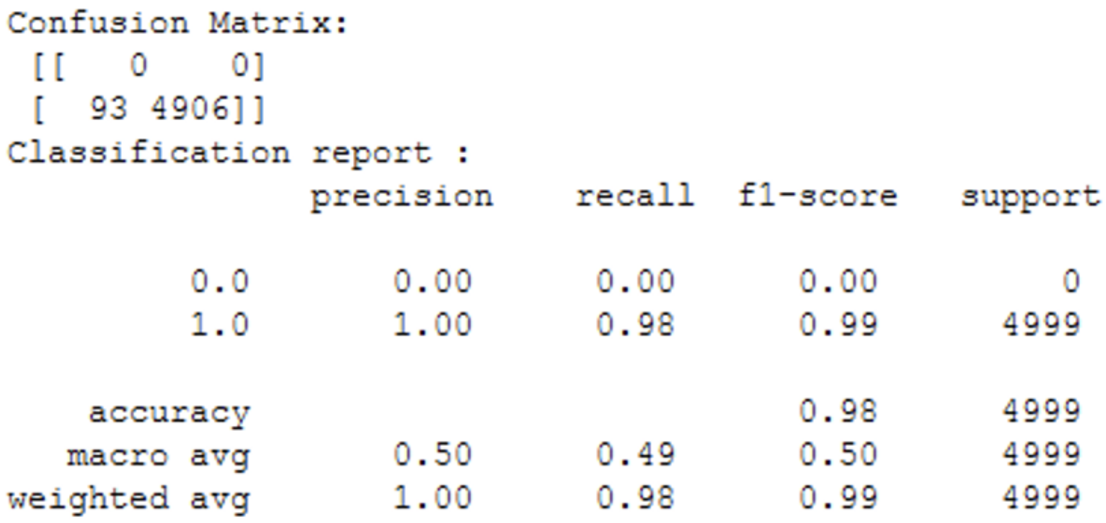
```
Confusion Matrix:
 [[469 267]
 [ 60 467]]
Classification report :
              precision    recall  f1-score   support

         0.0       0.89      0.64      0.74       736
         1.0       0.64      0.89      0.74       527

    accuracy                           0.74      1263
```

```
Text(0, 0.5, 'True positive rate')
```



1. Results on NIH dataset

   We also tested this model on the NIH dataset, and we see that the model had an accuracy and recall of 0.98.



```
Confusion Matrix:
 [[   0    0]
 [  93 4906]]
Classification report :
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00         0
         1.0       1.00      0.98      0.99      4999

    accuracy                           0.98      4999
   macro avg       0.50      0.49      0.50      4999
weighted avg       1.00      0.98      0.99      4999
```

Conclusions

The SimCLRv2 model for "classifying images as X-ray or not" shows exceptional results and can be put into production with little to no modifications.

The SimCLRv2 model for use in "classifying X-rays as Good/Bad" is being investigated and experimented with different techniques like ensemble learning techniques to further improve the performance and confidence of the predictions.

References

- A Simple Framework for Contrastive Learning of Visual Representations
- https://analyticsindiamag.com/contrastive-learning-self-supervised-ml/

- https://amitness.com/2020/03/illustrated-simclr/
- https://ml.berkeley.edu/blog/posts/contrastive_learning/