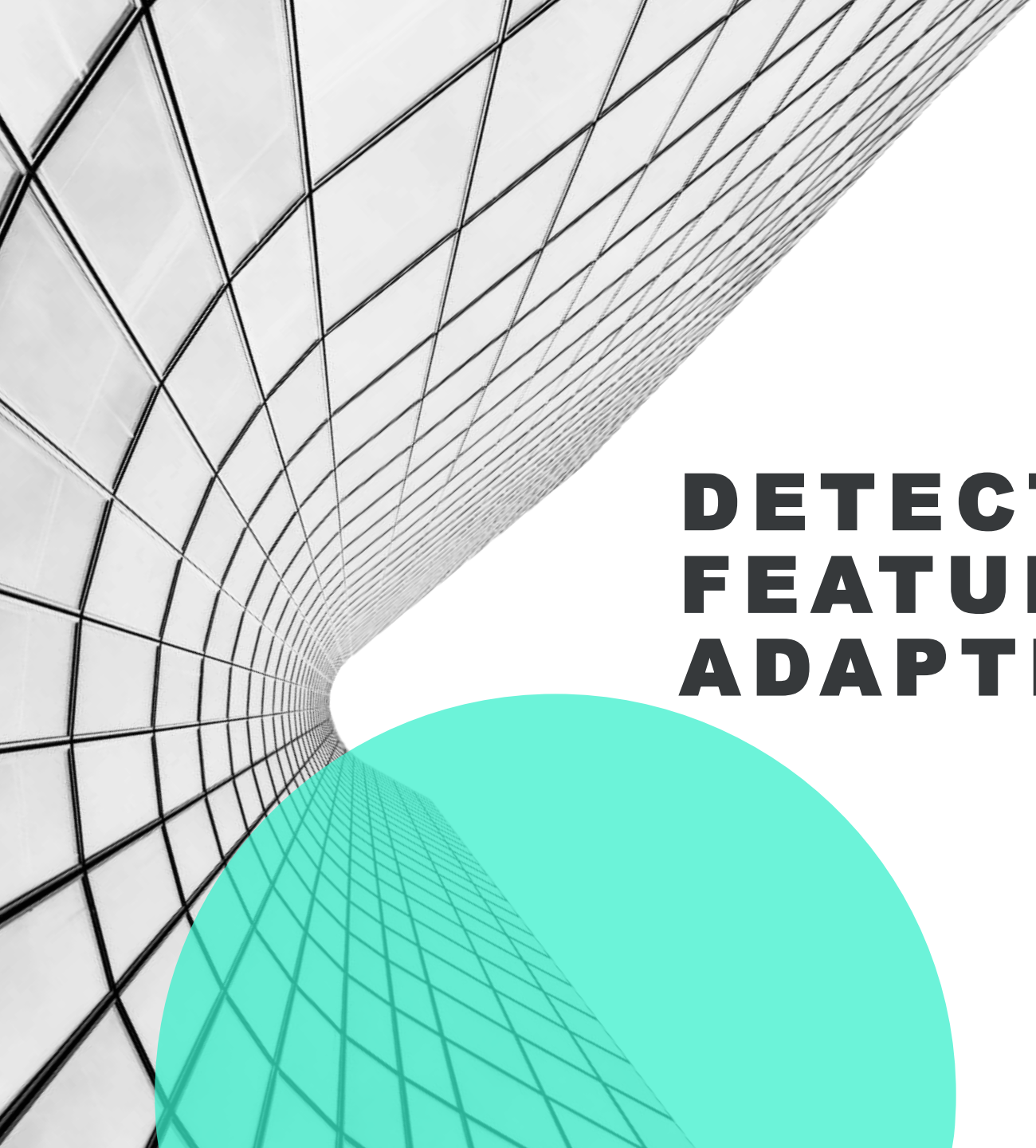


DETECTION OF FACE FEATURES USING ADAPTED TRIPLET LOSS

Prudhviraj Boddu
Mahesh Gundagoni

AGENDA

INTRODUCTION

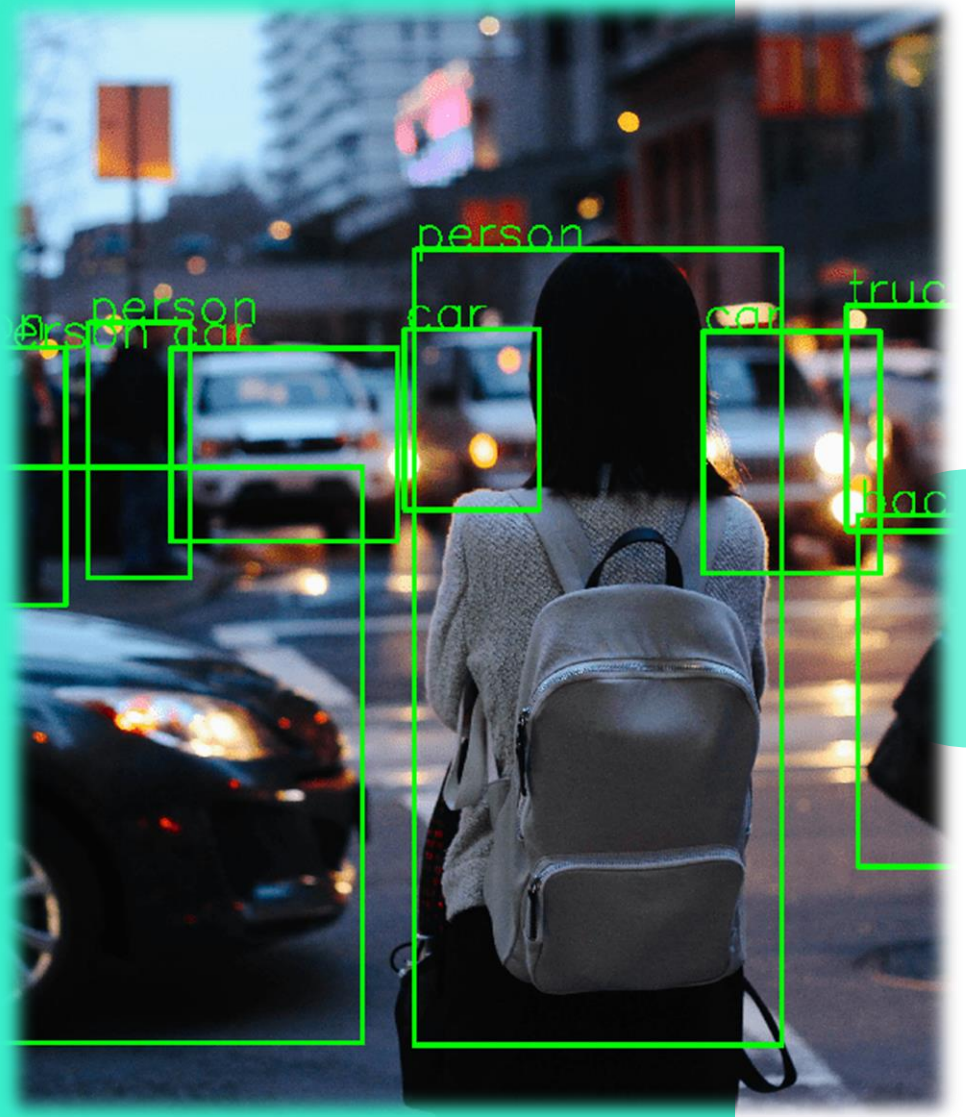
DATA PREPROCESSING

MODEL

OBJECTIVE FUNCTION

EXPERIMENTATION AND RESULTS

INTRO



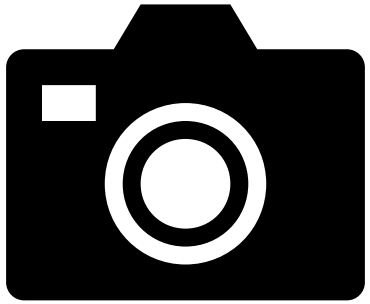
Our project aims to detect and recognize five major image classes of faces that's present in our dataset. We have used ResNet backbone for our project and PyTorch as Deep Learning Framework.

DATA PROCESSING

1. Data Collection
2. Annotations
3. Custom Dataset
4. Data Loading
5. Visualizing the batch of samples

DATA COLLECTION

- We collected a total of 200 samples to train the model consisting of five different classes. We captured pictures around the campus and collected some samples from the social media page of the university to make it more diverse.



  - Student
  - Security
  - Food Service Worker
  - Facility Operator
  - Staff

DATA ANNOTATION

- We have used Label studio as a annotation tool to draw bounding boxes around the pictures and label them using it.
- This tool will generate XML file for each image which contains the annotation data of all the bounding boxes which contains –
  - Xmax,Xmin,Ymax,Ymin of each box
  - Original image's height and width
  - Label for each bounding box

e958869b-compressed_Screenshot_72.xml

D: > DL_proj > dataset > train > e958869b-compressed_Screenshot_72.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<annotation>
<folder>images</folder>
<filename>e958869b-compressed_Screenshot_72.png</filename>
<source>
<database>MyDatabase</database>
<annotation>COCO2017</annotation>
<image>flickr</image>
<flickrid>NULL</flickrid>
<annotator>1</annotator>
</source>
<owner>
<flickrid>NULL</flickrid>
<name>Label Studio</name>
</owner>
<size>
<width>477</width>
<height>792</height>
<depth>4</depth>
</size>
<segmented>0</segmented>
<object>
<name>Staff</name>
<pose>Unspecified</pose>
<truncated>0</truncated>
<difficult>0</difficult>
<bndbox>
<xmin>273</xmin>
<ymin>254</ymin>
<xmax>443</xmax>
<ymax>513</ymax>
</bndbox>
</object>
</annotation>

```

CUSTOM DATASET& DATA LOADING

- We have defined the custom dataset to take image paths in the folders and XML files associated with the image in the same folder and cropped height and width to train the model of that image size size.
- We extracted information in XML file using inbuilt xml library in python, set bounding boxes according to the image height and width and add labels to the torch tensor of each bounding box in an image.
- We used Custom dataset with dataloader to feed the data to the model of batch size 8.
- This will load the data to the model as batch of tensors to the model to train on. Based on our GPU memory we set the batch size to 8.

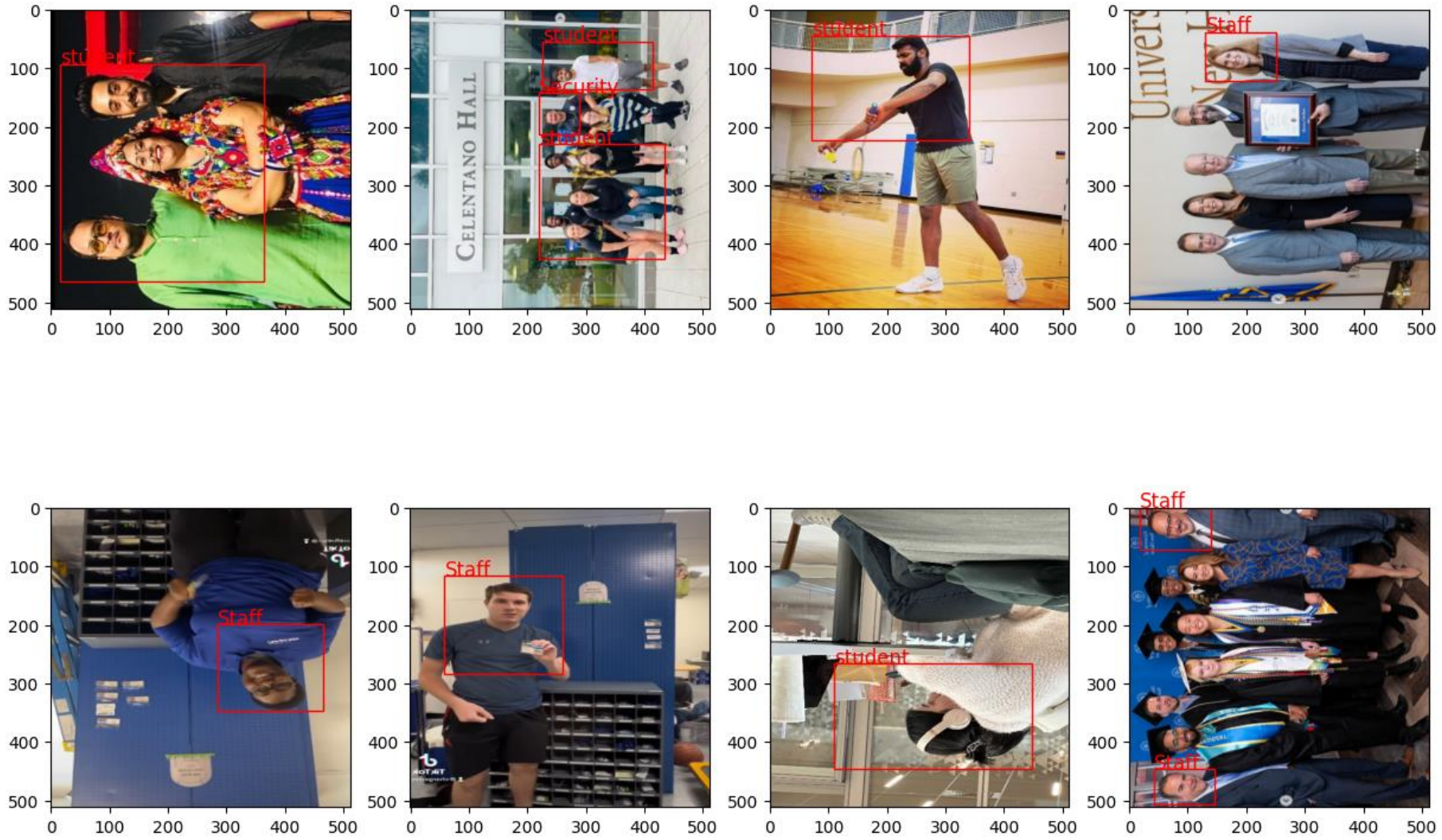
TRANSFORMATIONS

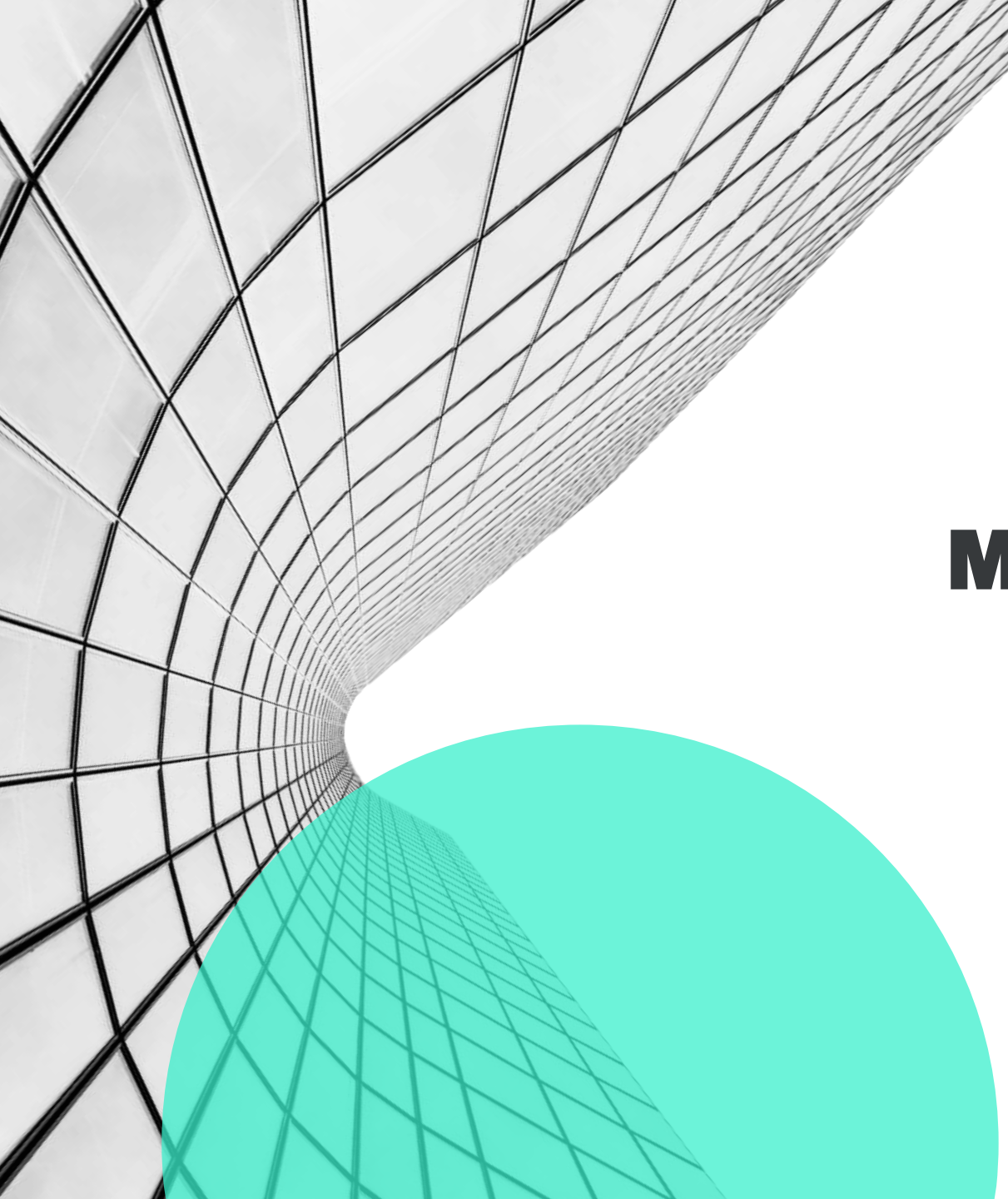
- We have used Albumentations library to transform the training and validation images, such as Flipping the image, Random rotation, blur the images randomly for the training dataset.

```python
# define the training tranforms
def get_train_transform():
    """
    This function defines the training transforms that are applied to the the training dataset
    """

    return A.Compose([
        A.Flip(0.5),
        A.RandomRotate90(0.5),
        A.MotionBlur(p=0.2),
        A.MedianBlur(blur_limit=3, p=0.1),
        A.Blur(blur_limit=3, p=0.1),
        ToTensorV2(p=1.0),
    ], bbox_params={
        'format': 'pascal_voc',
        'label_fields': ['labels']
    })

# define the validation transforms
def get_valid_transform():

    """
    This function defines the validation transforms that are applied to the the validation dataset

    """

    return A.Compose([
        ToTensorV2(p=1.0),
    ], bbox_params={
        'format': 'pascal_voc',
        'label_fields': ['labels']
    })
```

VISUALIZING BATCH OF IMAGES

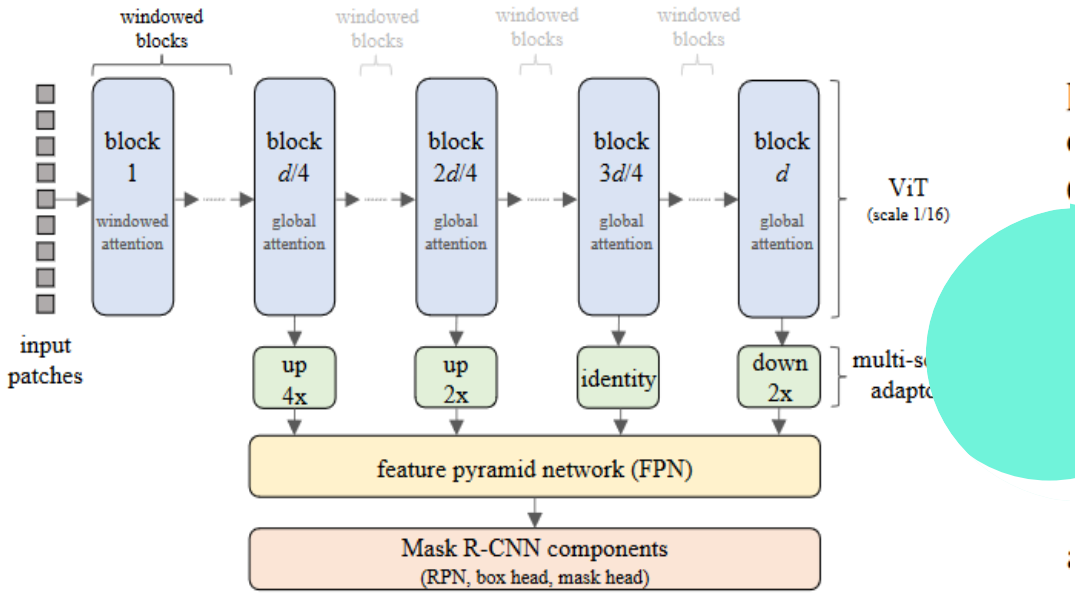
- Here's a batch of transformed training images.





METHOD / MODEL

MODEL



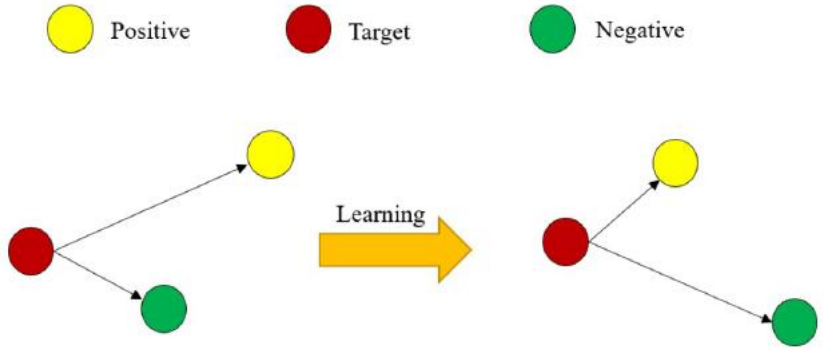
- Our Paper focussed on **ResNet** Architecture to develop the model.
- We have used COCO_dataset pretrained weights for ResNet Architecture.
- We froze all the layers except the last layer which contains number of layers.
- We have taken input weights to the last layer and added number of classes for the box head that's present in the last layer.

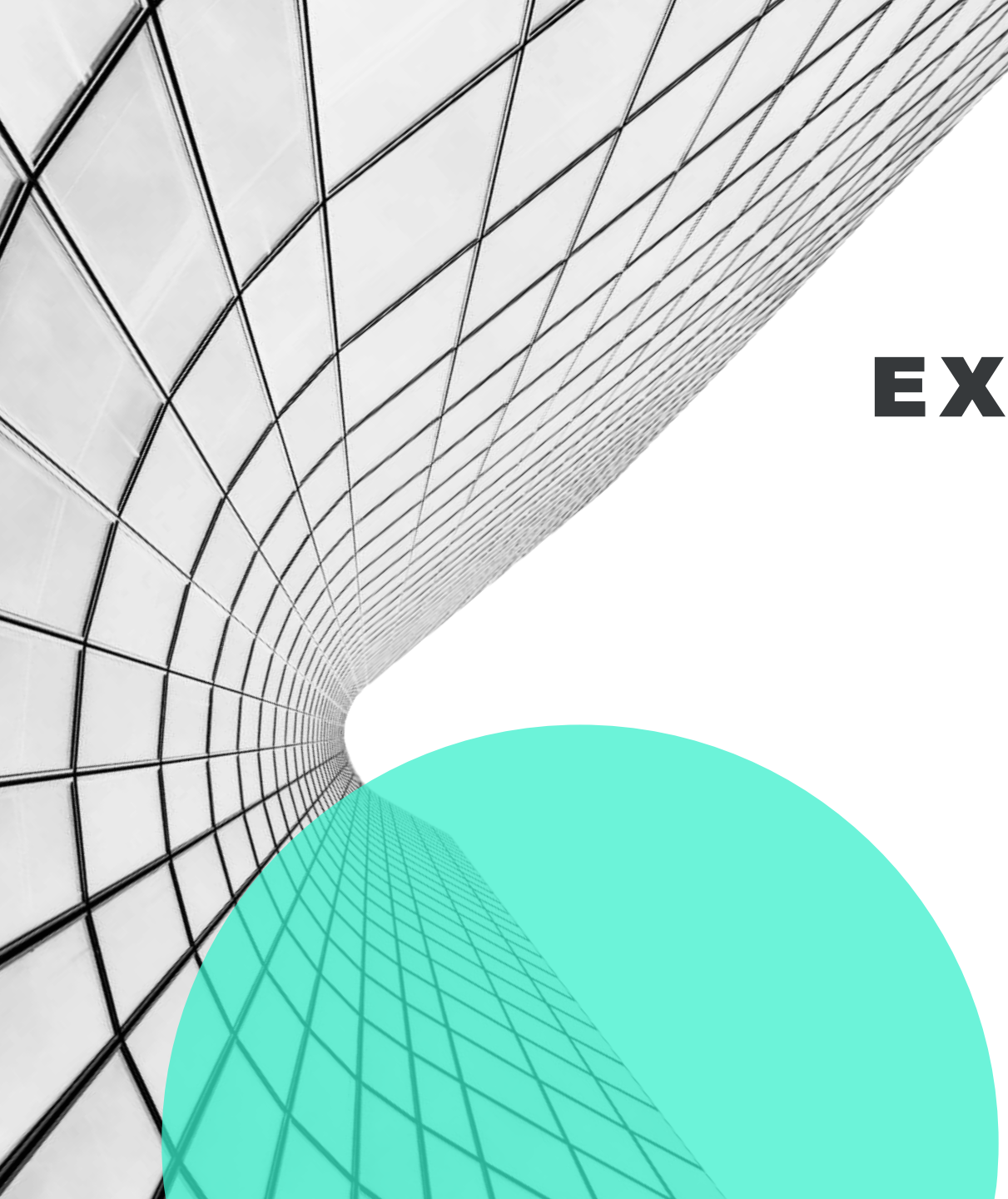
OBJECTIVE FUNCTION

- In this project, we are using Triplet Loss function as a base of our model. This is widely used in algorithms of machine learning.

- A triplet is collection of three inputs: anchor point, negative point against anchor and a positive point similar to anchor.

- We have to pass all of these data points through the model and outputs of these models to calculate the loss. Throughout the training the loss distance between anchor-positive will be smaller than the distance between anchor-negative. This helps us to develop a distance function that allocates distance between set of dissimilar and similar images in the dataset.





EXPERIMENTATION & RESULTS

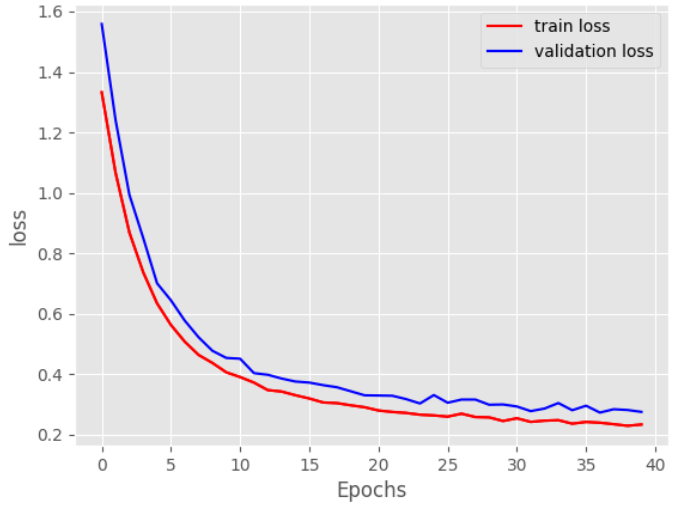
EXPERIMENTATION & RESULTS

- **Hyper Parameters –** Epochs, Learning rate, momentum, weight decay.
- We have trained the model for 40 Epochs and saving the model if the previous validation loss is less than the present validation loss to get the best model.
- **Loss –** In triplet loss, we have taken margin of 0.01 and reduce the over loss to average of all samples.
- **Optimizer –** Stochastic Gradient Descent **(SGD)**
- **Learning rate scheduler –** To adjust the learning rate according to the optimizer loss after waiting for three epochs.

EXPERIMENTATION & RESULTS

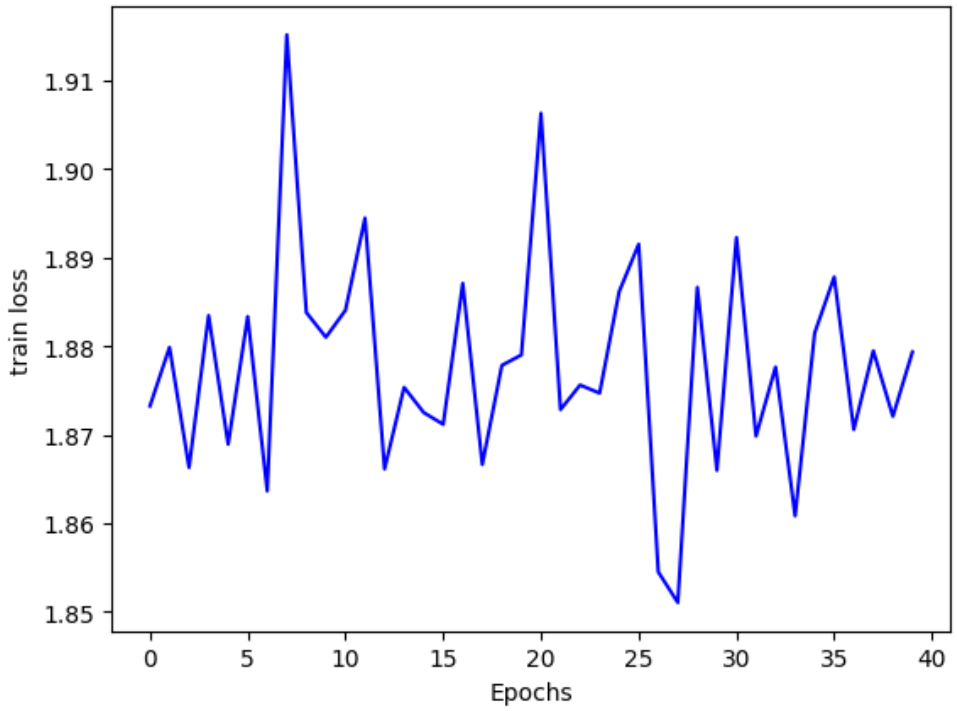
- We started with three learning rates **1e-7,1e-9,2e-9** which are very small learning rates and use the scheduler to reduce the learning rate if needed for the model to train and optimize slowly as we have very less data.
- We have used momentum of **0.4** and weight decay of **5e-8**, to speed up the training and to produce better gradients using the momentum.

```
EPOCH 39 of 40
Training
Loss: 1.8273: 100%|██████████| 10/10 [00:10<00:00,  1.09s/it]
Validating
Loss: 1.7512: 100%|██████████| 2/2 [00:01<00:00,  1.44it/s]
Epoch #39 train loss: 1.872
Epoch #39 validation loss: 1.748
Took 0.205 minutes for epoch 39

EPOCH 40 of 40
Training
Loss: 1.7915: 100%|██████████| 10/10 [00:10<00:00,  1.08s/it]
Validating
Loss: 1.7496: 100%|██████████| 2/2 [00:01<00:00,  1.48it/s]
Epoch #40 train loss: 1.879
Epoch #40 validation loss: 1.747
Took 0.202 minutes for epoch 40
```

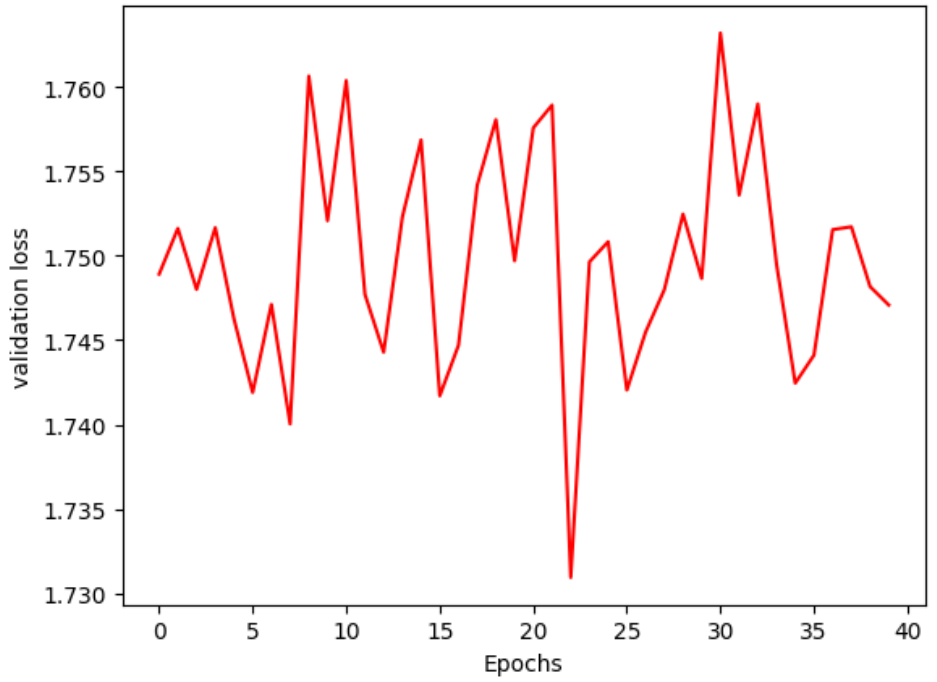


EXPERIMENTATION & RESULTS

- **Training & Validation Losses vs EPOCHS (40)**



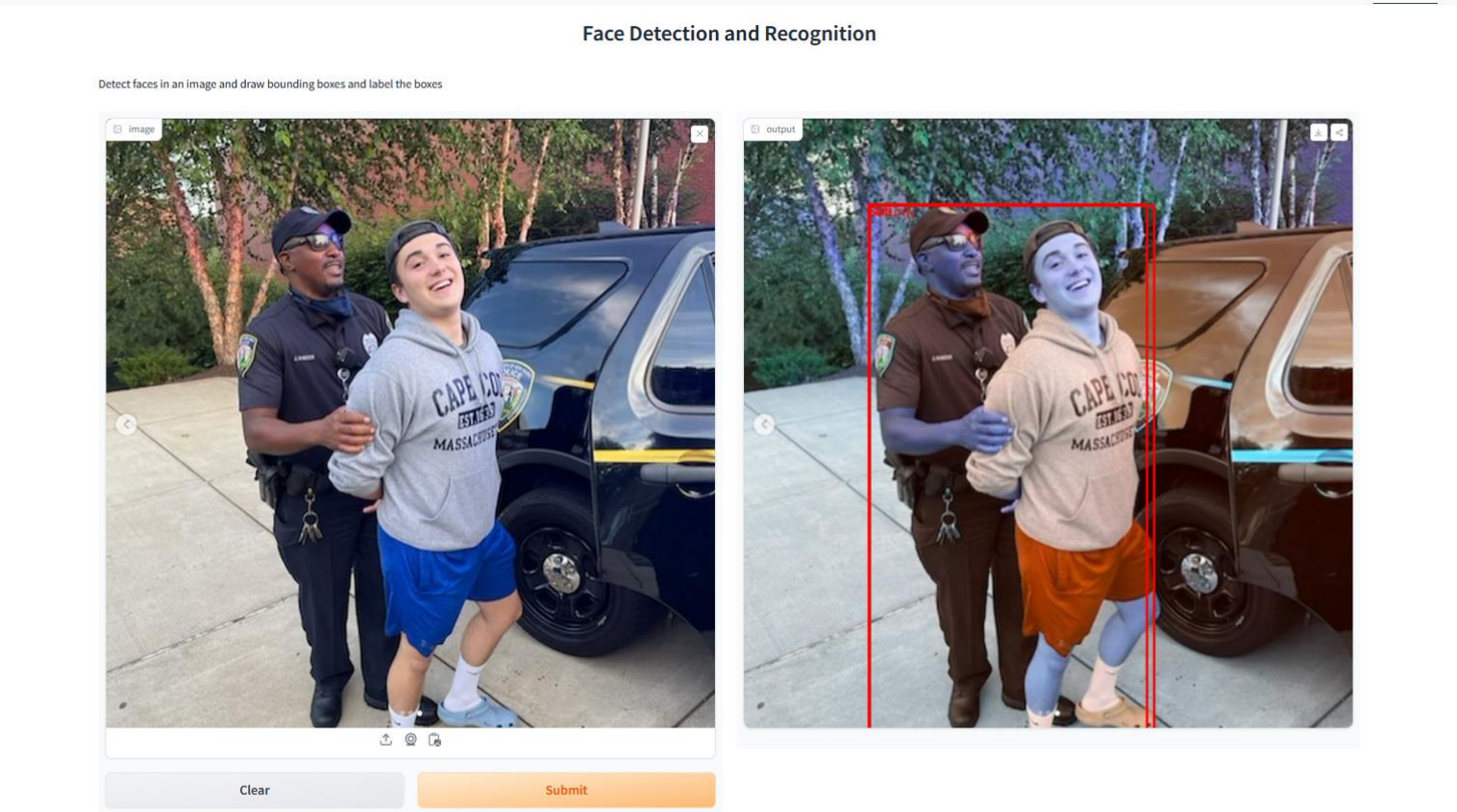
Training Loss



Validation Loss

GRADIO INTERFACE

- We deployed the app along with the model in Huggingface spaces using Gradio.
- Link to the app: https://huggingface.co/spaces/prudhvirajboddu/detectionmodel



THANK YOU