Great Place To Work. Certified MAR 2021-FEB 2022 INDIA

UPES UNIVERSITY OF TOMORROW

# Major Project

## Image Depth Detection Using Deep Learning

Navkirat Singh

Arun Gera

Mentored By:
Richa Choudhary
Assistant Professor- Senior Scale
Department of Informatics
UPES, Dehradun

## Content

## Introduction

- Depth is among the most useful intermediate representations for action in physical environments.

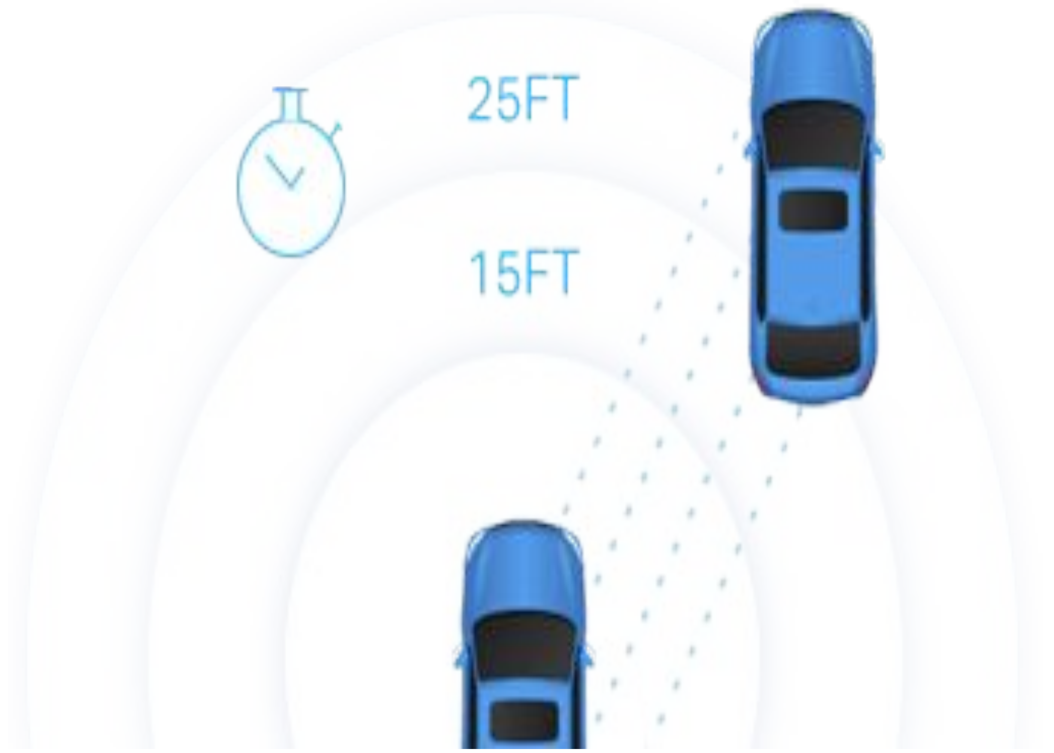
- Depth detection allows self driving cars to  predict trajectory and even robots to calculate the path.

- Over the years, with humanity becoming technologically advanced and more equipped with resources and knowledge of previous research, various algorithms have been proposed to detect the depth of the object from the source.

- But research always continues inventing new algorithms and improving on previous shortcomings.

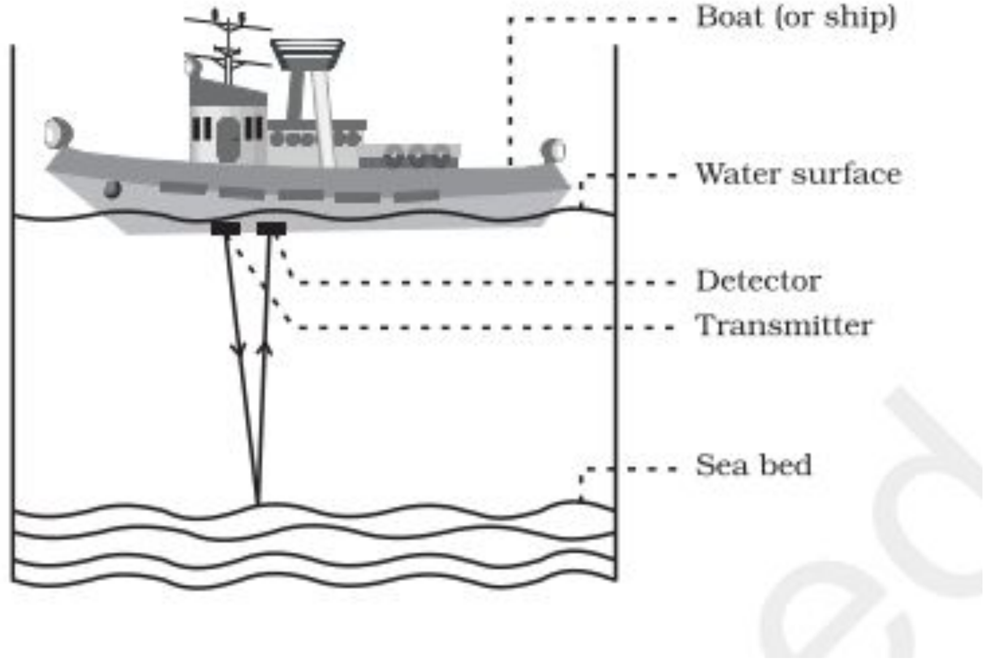
## Problem Statement

- Depth Detection using ultrasonic and lidar sensors is quite efficient especially with later
- But in recent times with massive necessity of detecting depth nearly everywhere
- for instance in autonomous cars, robots and drones
- use of lidar sensor makes the already expensive product more expensive



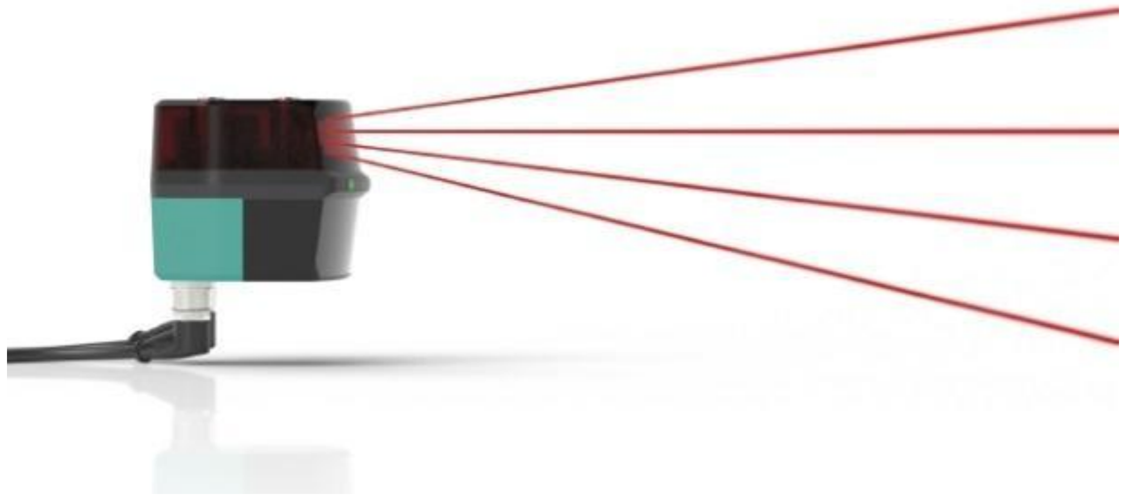
## Ultrasonic Depth Detection

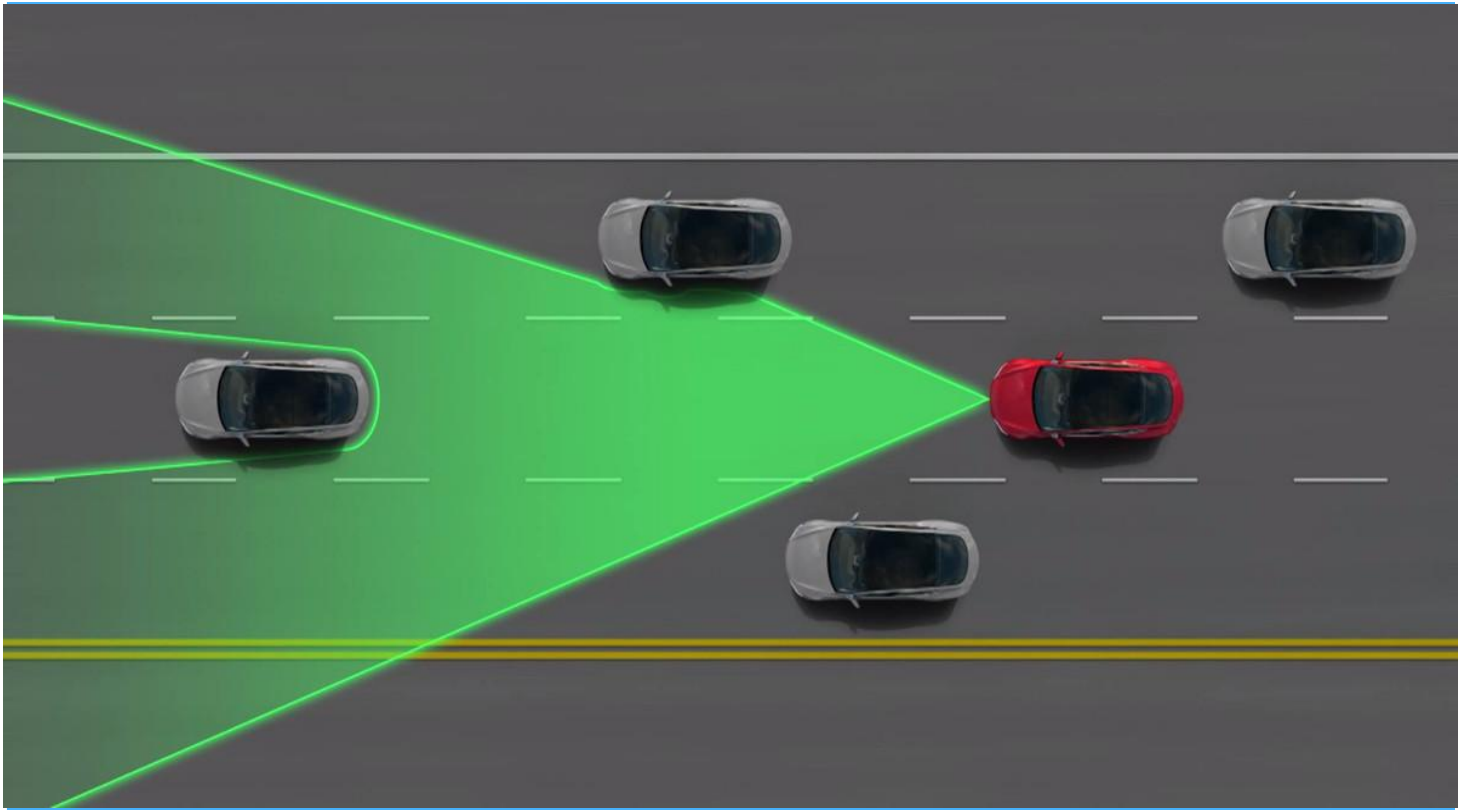
- optical sensor that transmitter and receive, an ultrasonic signal
- uses a single ultrasonic element for both emission and reception.
- Uses basic physics reflection model.
- Distance = ½ * Time * C
- Where C is speed of light.
- Not good for detecting fast moving objects



## Lidar Depth Detection

- Lidar (light detection and ranging)
- Uses same principle as Ultrasonic
- But uses eye-safe laser.
- Has ability to measure 3D structures.
- Can even detect small object and high speed objects with high accuracy
- But overly Expensive

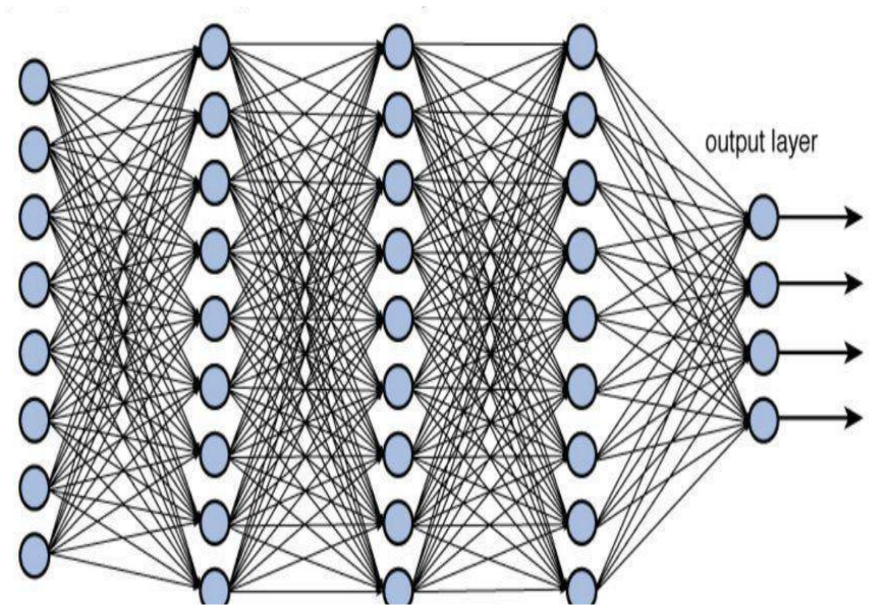




Vertical-cavity surface-emitting lasers (VCSELs) Or High Accuracy Lidar cost $75,000

# Objective - Neural Network

- Goal: build deep neural network to efficiently detect the relative depth of objects in image
- Eliminated the use of any hardware based device except for CPU or GPU (which is relatively inexpensive)
- Harness the power of massive datasets collected over the decades using lidar.
- Train and fine tune a model such that it works great on any image.



## Literature Review

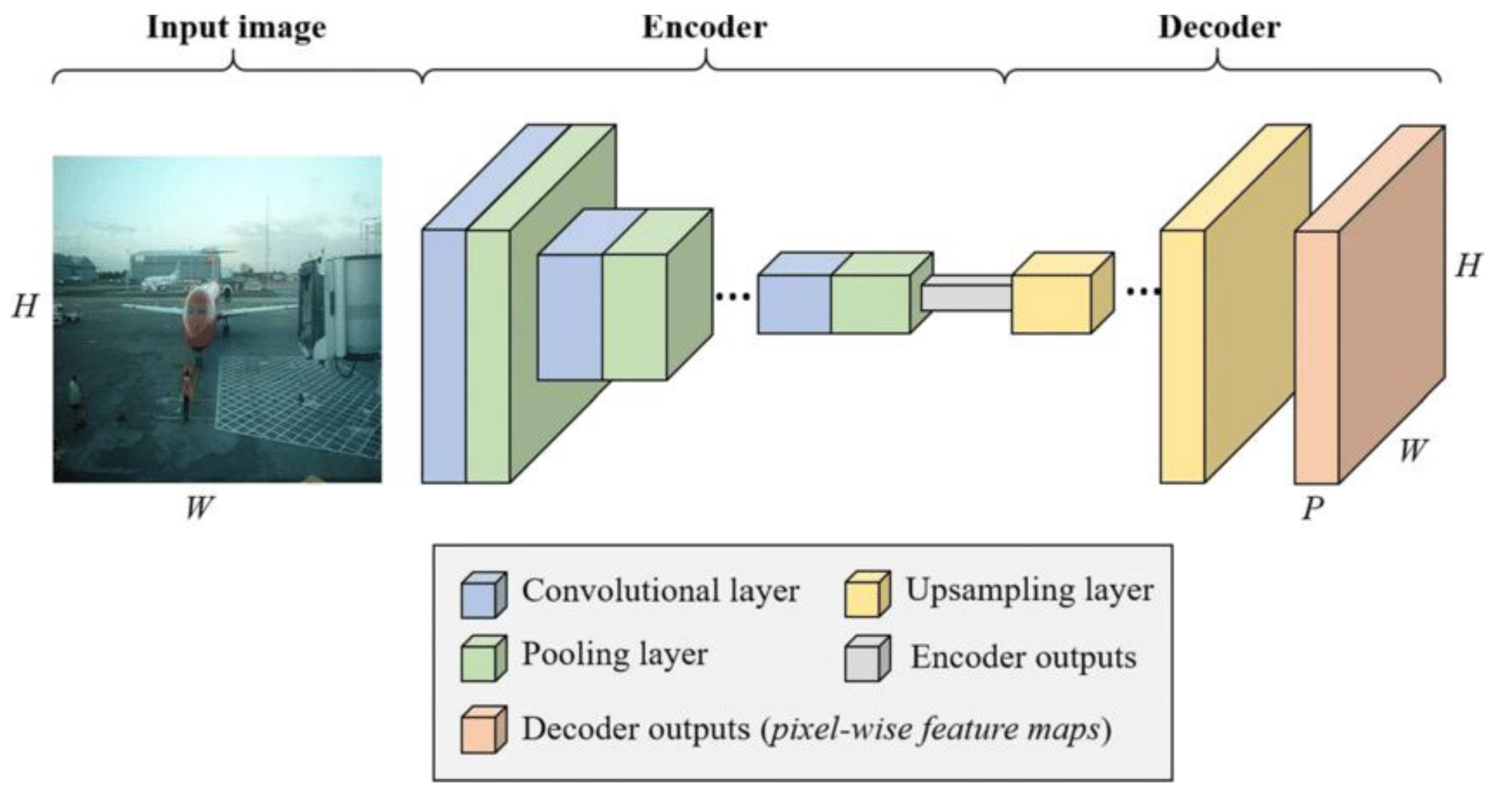
- Masked Image Modeling - Masked image modeling (MIM) is a sub-task of masked signal prediction,that masks a portion of input images, and lets the deep networks predict the masked signals conditioned on the visible ones.

- Image Segmentation - Image segmentation is the task of clustering parts of an image together that belong to the same object class. This process is also called pixel-level classification. In other words, it involves partitioning images (or video frames) into multiple segments or objects.

- We used Semantic segmentation instead of instance

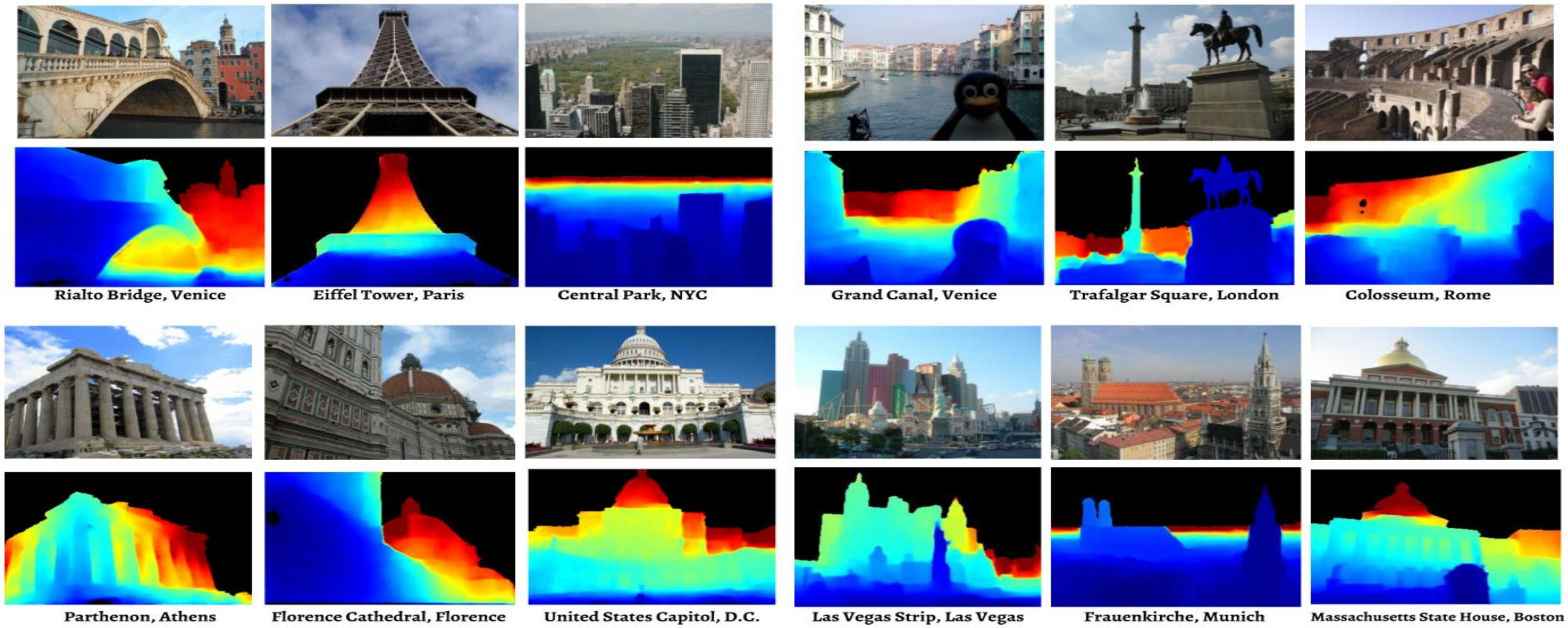
- Backbone Architectures - Masked image modeling is mostly studied in the Transformer architectures, thus the major understandings and experiments in this paper are performed on Vision

## Model Architecture



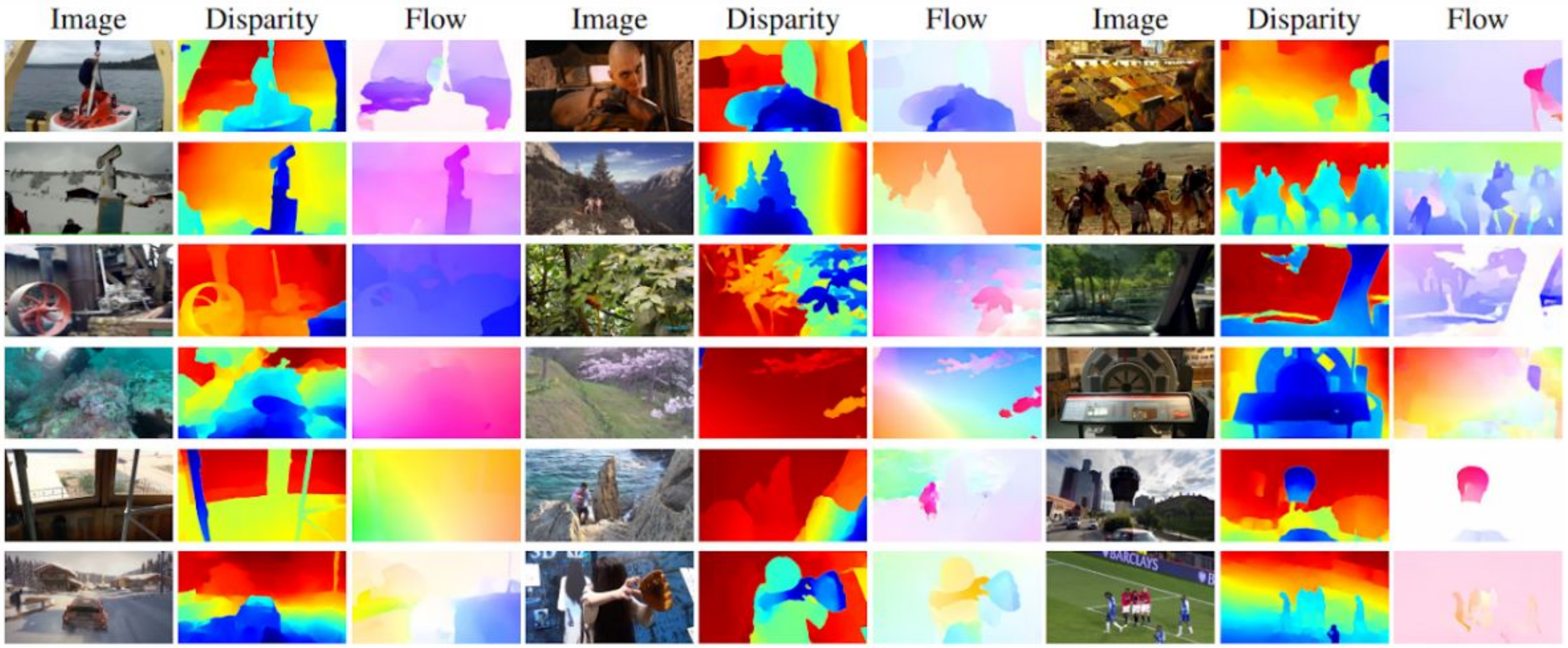
## Training Dataset

## MegaDepth: Learning Single-View Depth Prediction from Internet Photos



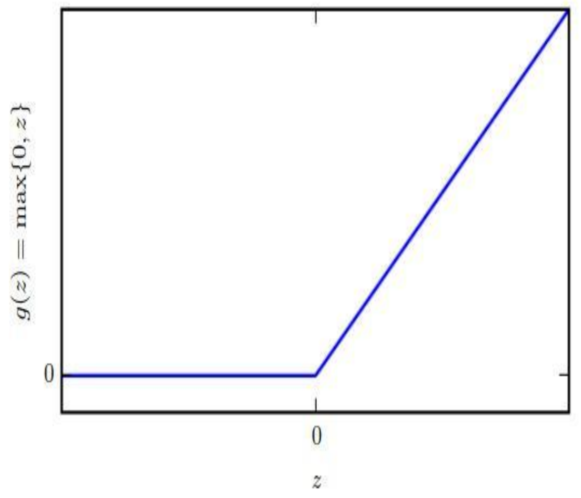
## Validation Dataset

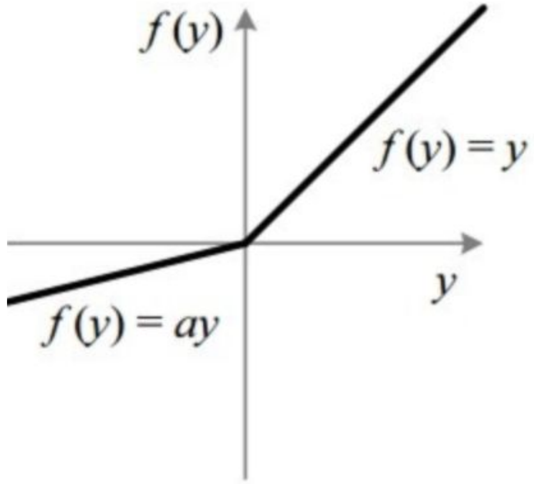
## Web Stereo Video Supervision for Depth Prediction from Dynamic Scenes (WSVD)



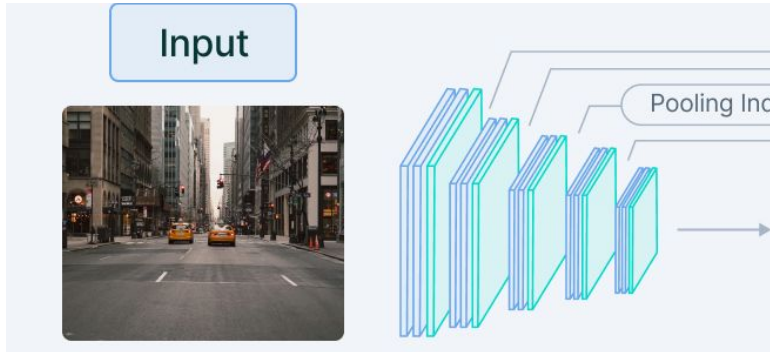
## Activation Function

- most vital component of neural network
- used to map non-linear function using neural network.
- used ReLU and leakyReLU in depthNet





## Encoder Component

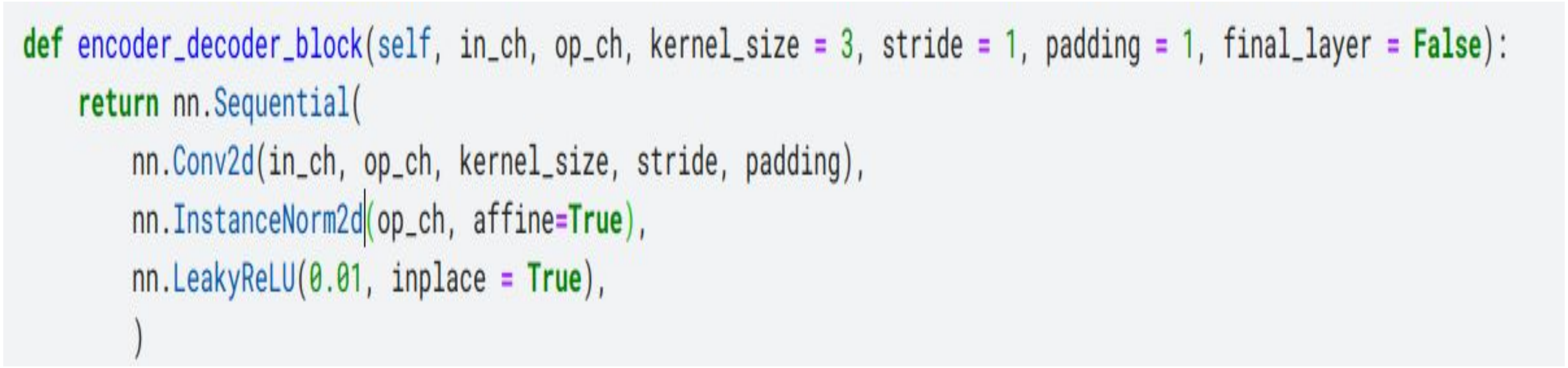


- Maps the input image of to latent vector

- Takes input an image of size 128x128x3
- Applies 11 convolutional layers along with ReLU activation to learn the patterns in image

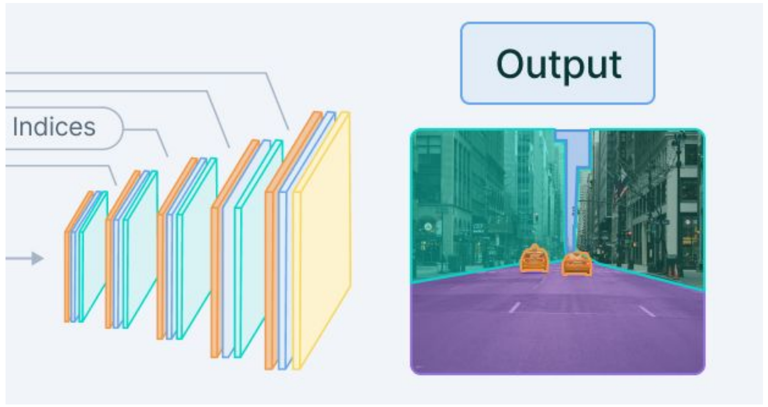
- Applies 11 average pooling layer to decrease dimension of image, as its passed through network
- Maps the input image to 1x1x64 latent vector to be passed in decoder

## Encoder Decoder Network



```
def encoder_decoder_block(self, in_ch, op_ch, kernel_size = 3, stride = 1, padding = 1, final_layer = False):
    return nn.Sequential(
        nn.Conv2d(in_ch, op_ch, kernel_size, stride, padding),
        nn.InstanceNorm2d(op_ch, affine=True),
        nn.LeakyReLU(0.01, inplace = True),
        )
```

## Decoder Component

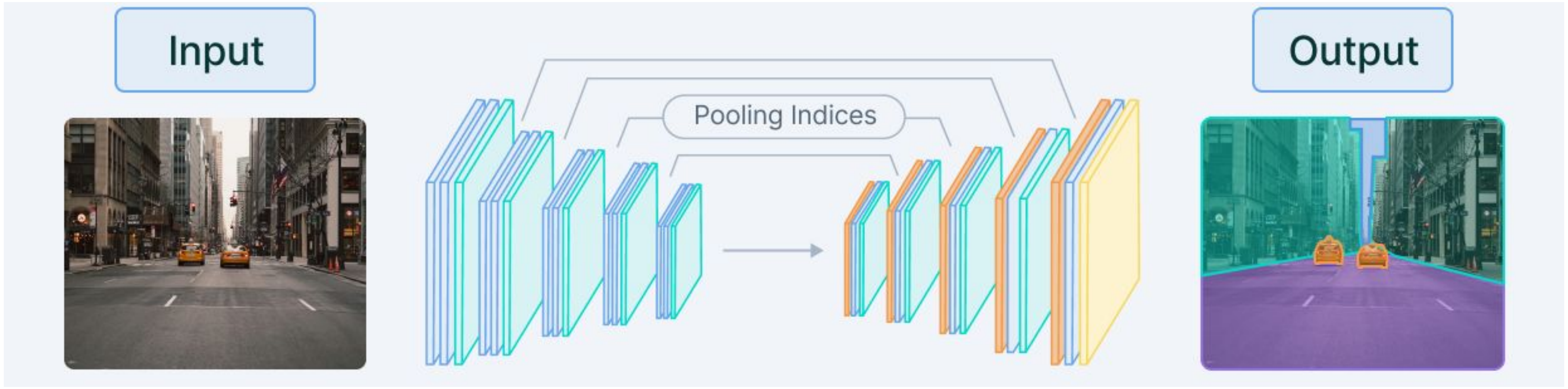


- Maps the latent vector to output image

- Takes input an latent vector of size 1x1x64
- Applies 7 deconvolutional layers along with leaky ReLU activation to learn the patterns in image

- Applies 7 upsampling layer to get original dimension of image, as its passed through network
- Maps the input latent vector of 1x1x64 to image of size 128x128x3

## Concatenating encoder decoder together

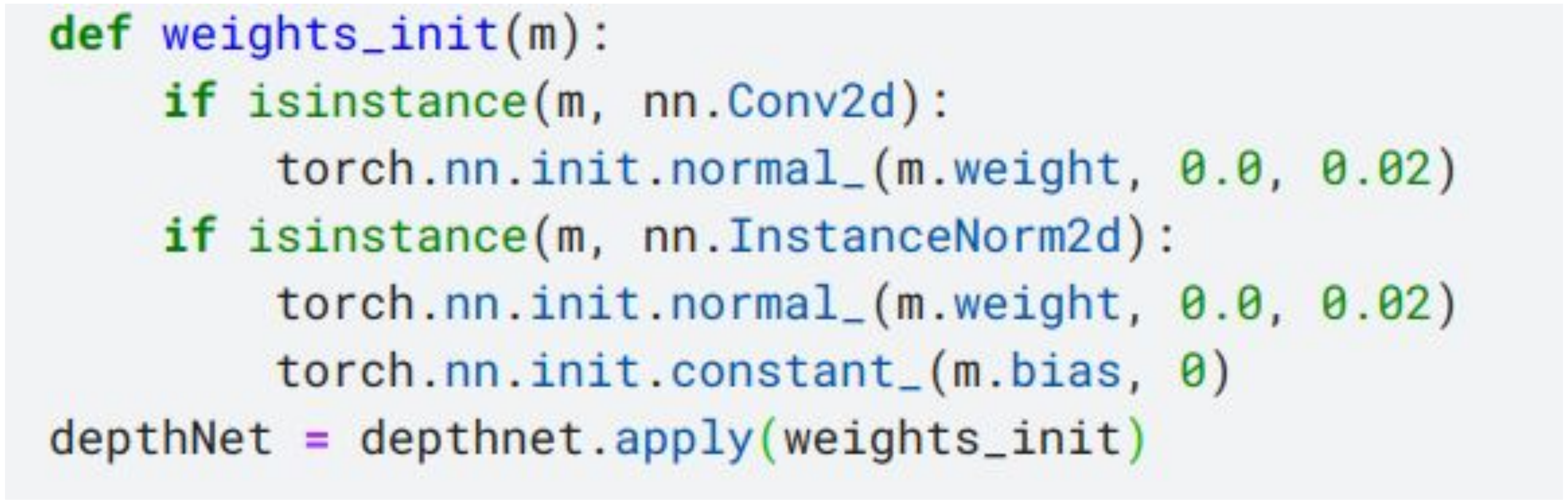


## Model and Weight Initialization

- For model to learn properly weights must be initialized carefully

- nor too large nor too small
- hence initialized weight from normal distribution

- with mean 0 and variance 0.02



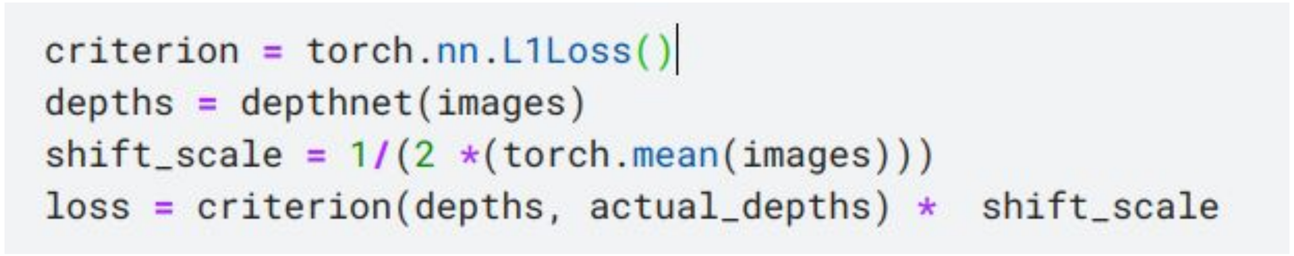
```
def weights_init(m):
    if isinstance(m, nn.Conv2d):
        torch.nn.init.normal_(m.weight, 0.0, 0.02)
    if isinstance(m, nn.InstanceNorm2d):
        torch.nn.init.normal_(m.weight, 0.0, 0.02)
        torch.nn.init.constant_(m.bias, 0)
depthNet = depthnet.apply(weights_init)
```

## Loss Function : Scale and shift invariant Loss Function

- The choice of the loss function is important as it plays a vital role in providing feedback to the neural networ

$$\mathcal{L}_{ssi}(\hat{\mathbf{d}}, \hat{\mathbf{d}}^*) = \frac{1}{2M}\sum_{i=1}^{M}\rho\left(\hat{\mathbf{d}}_i - \hat{\mathbf{d}}_i^*\right),$$

- Gradient over the loss helps minimise the loss and help network learn

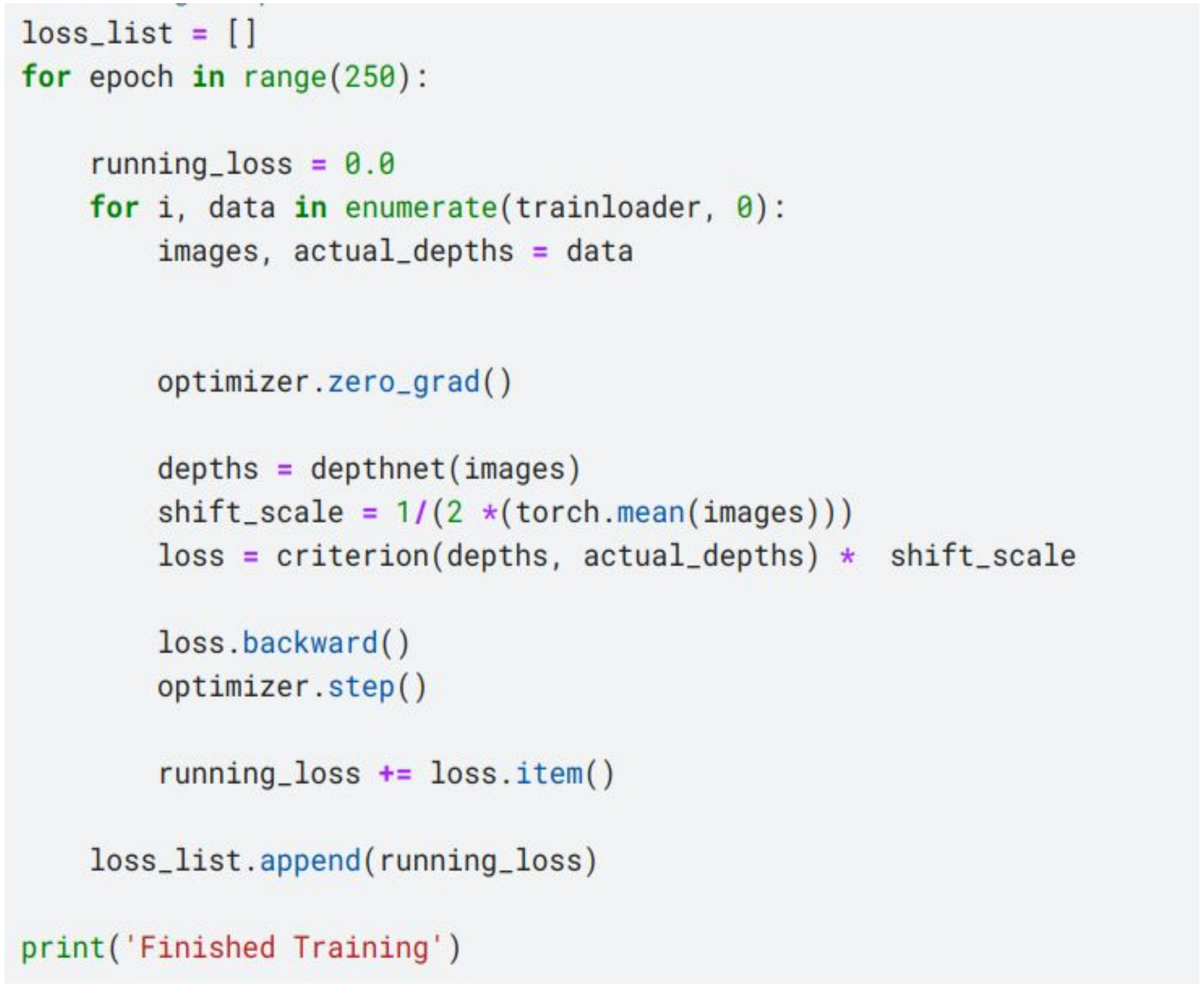


```
criterion = torch.nn.L1Loss()
depths = depthnet(images)
shift_scale = 1/(2 *(torch.mean(images)))
loss = criterion(depths, actual_depths) *  shift_scale
```

- We take scaling factor p = 1 because only required in video

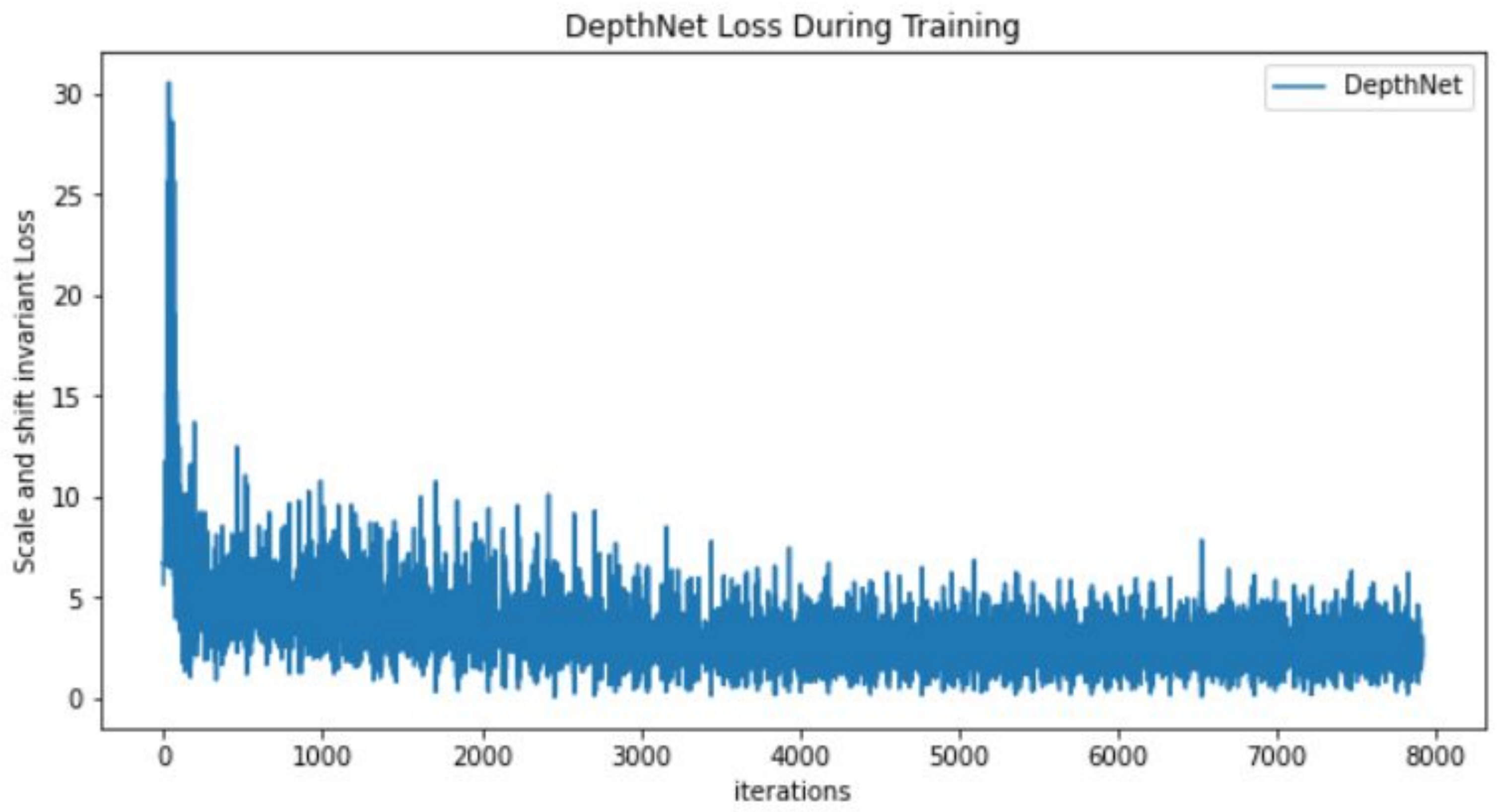
## Training Phase

- Trained model for 10 hrs daily for 30 days
- For training purpose variation of pure gradient descent was used i.e RMSProp.
- Trained on Nvidia T4 GPU 16 GB

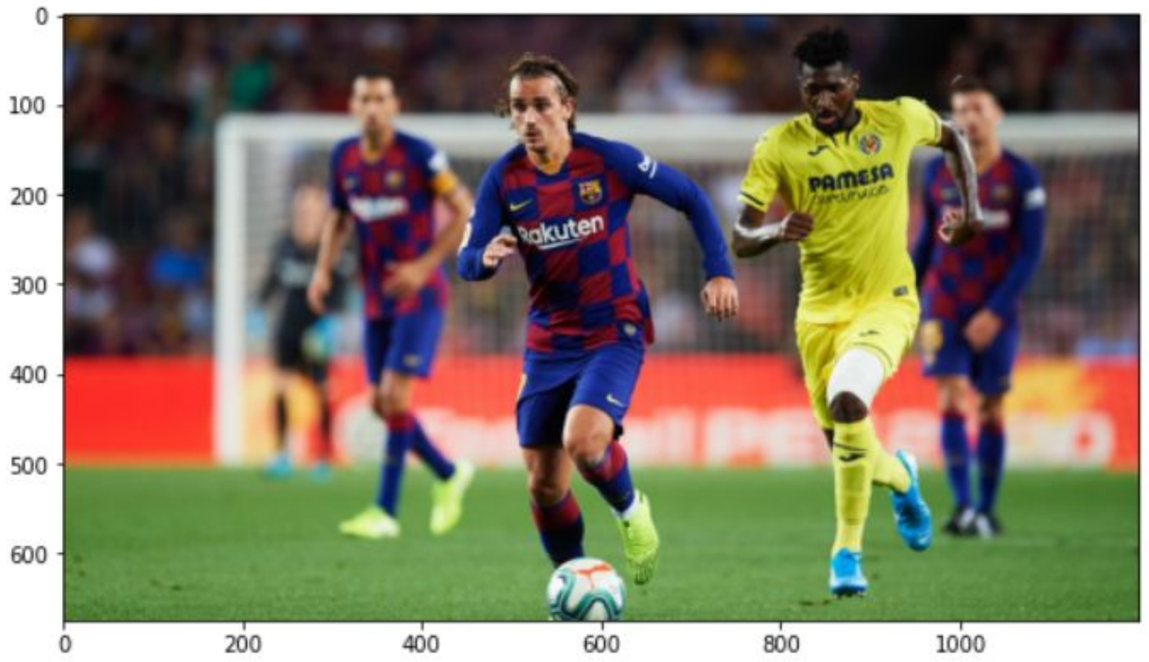


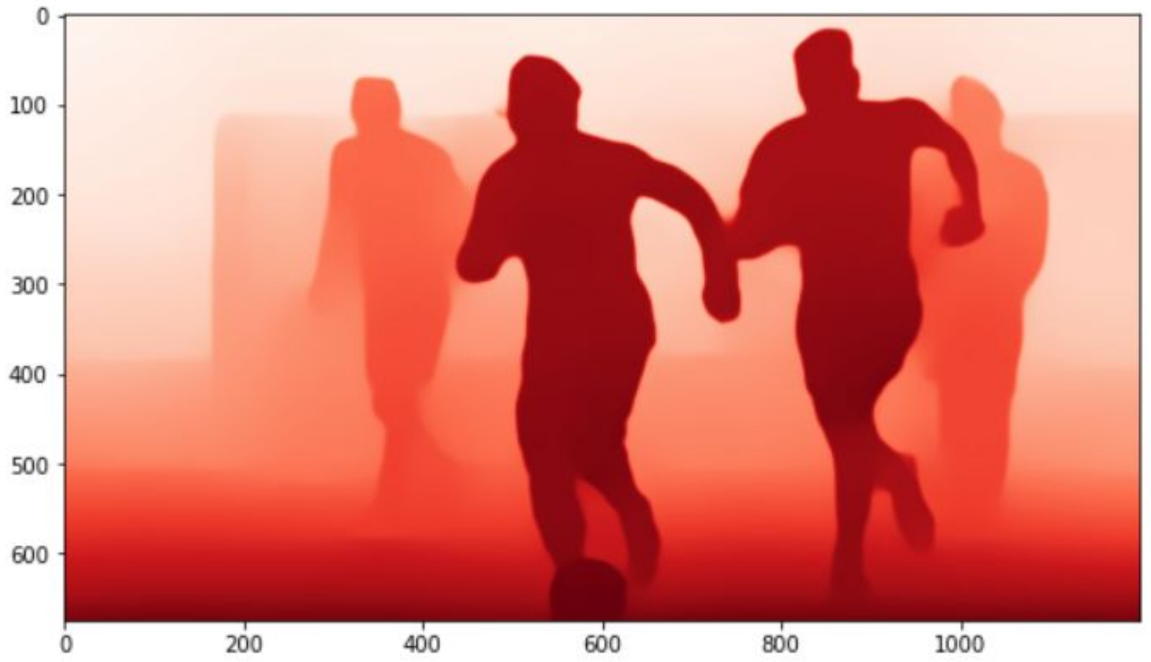
```
loss_list = []
for epoch in range(250):

    running_loss = 0.0
    for i, data in enumerate(trainloader, 0):
        images, actual_depths = data


        optimizer.zero_grad()

        depths = depthnet(images)
        shift_scale = 1/(2 *(torch.mean(images)))
        loss = criterion(depths, actual_depths) *  shift_scale

        loss.backward()
        optimizer.step()

        running_loss += loss.item()

    loss_list.append(running_loss)

print('Finished Training')
```

## Training Phase

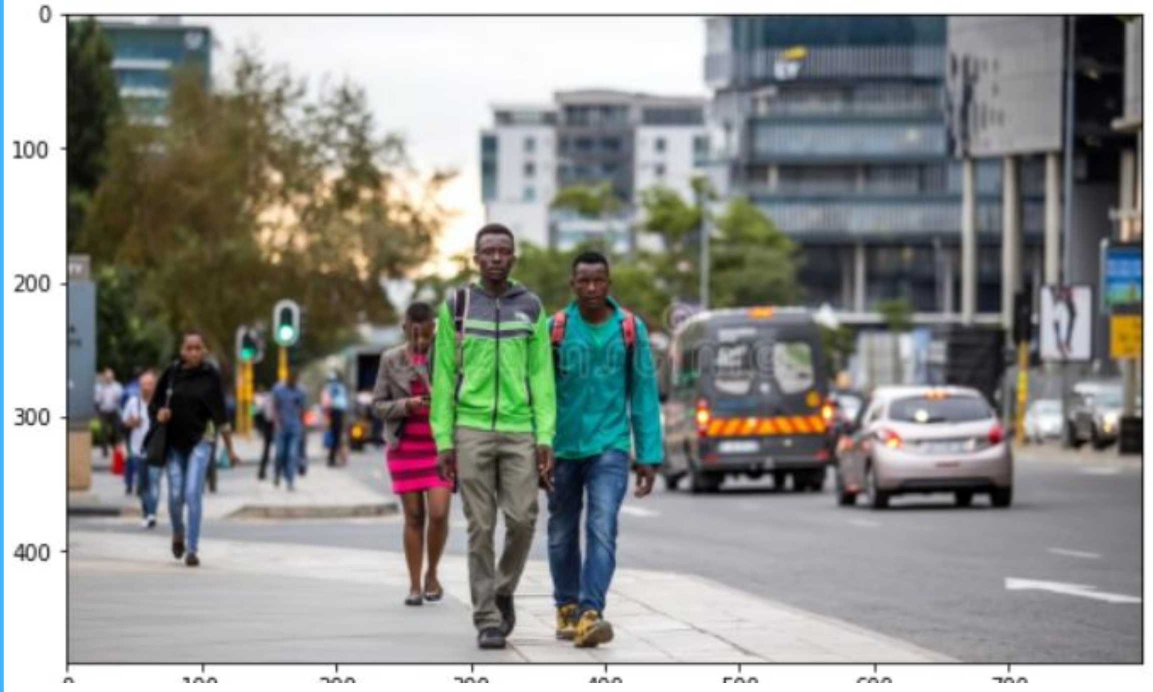


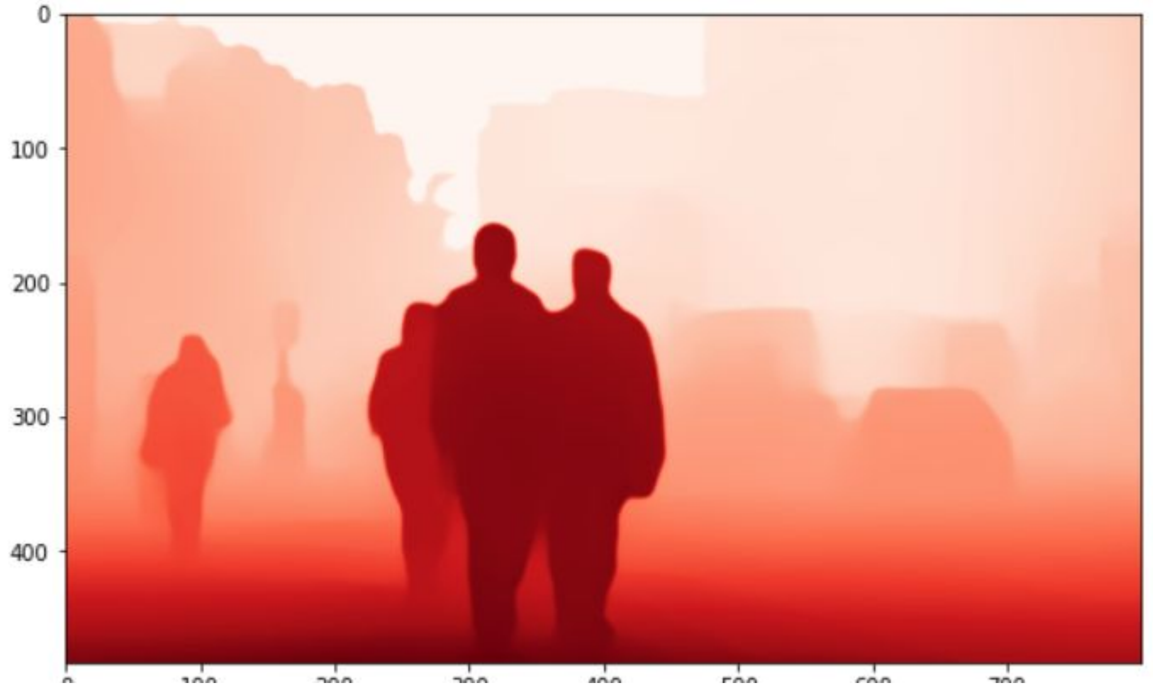
## Predictions and Testing on Random Internet Images



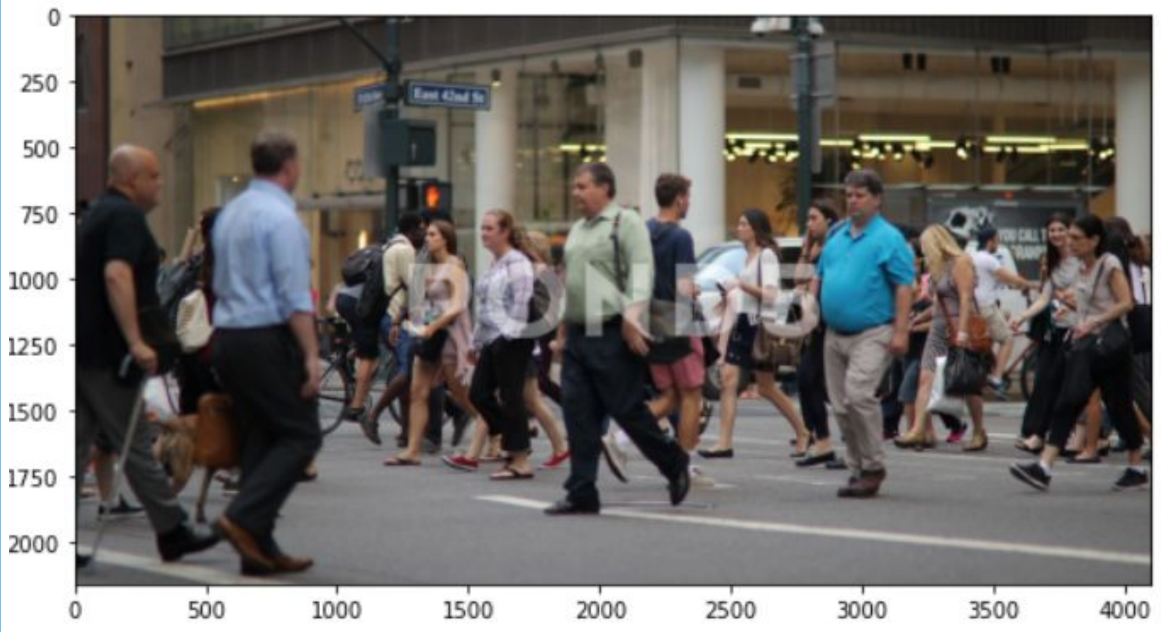


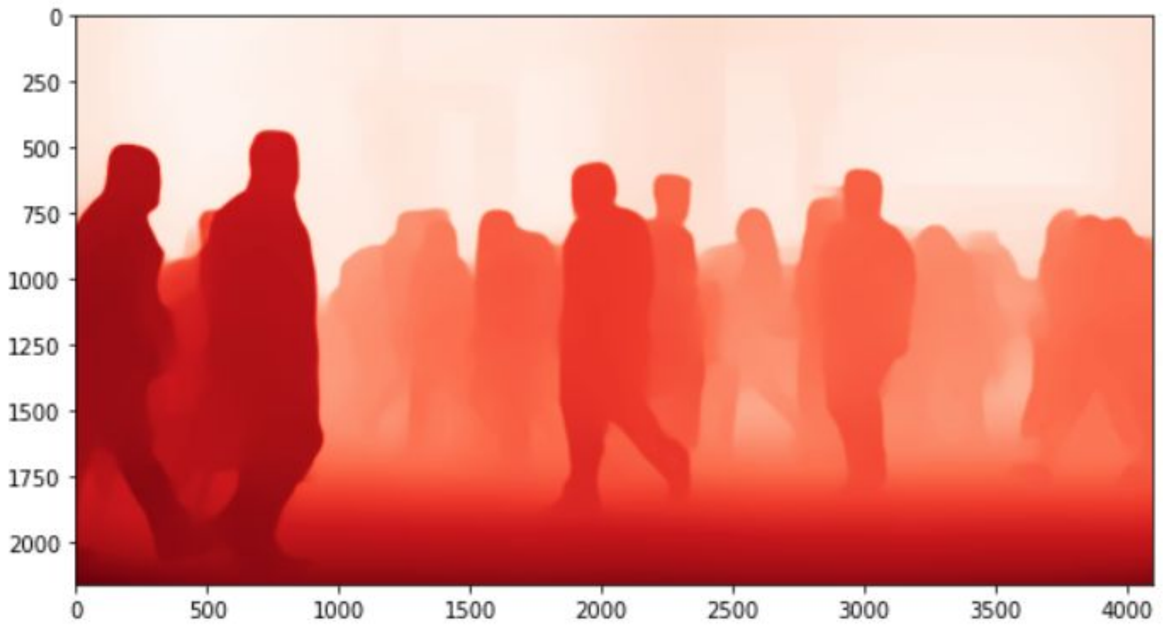
## Predictions and Testing on Random Internet Images



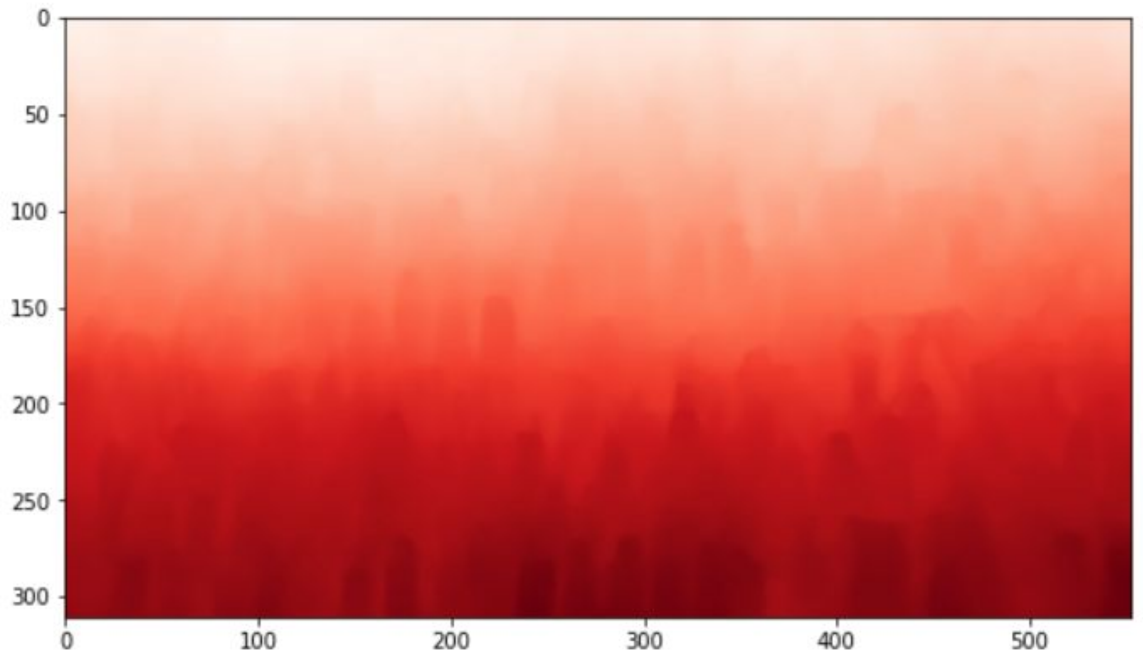


## Predictions and Testing on Random Internet Images

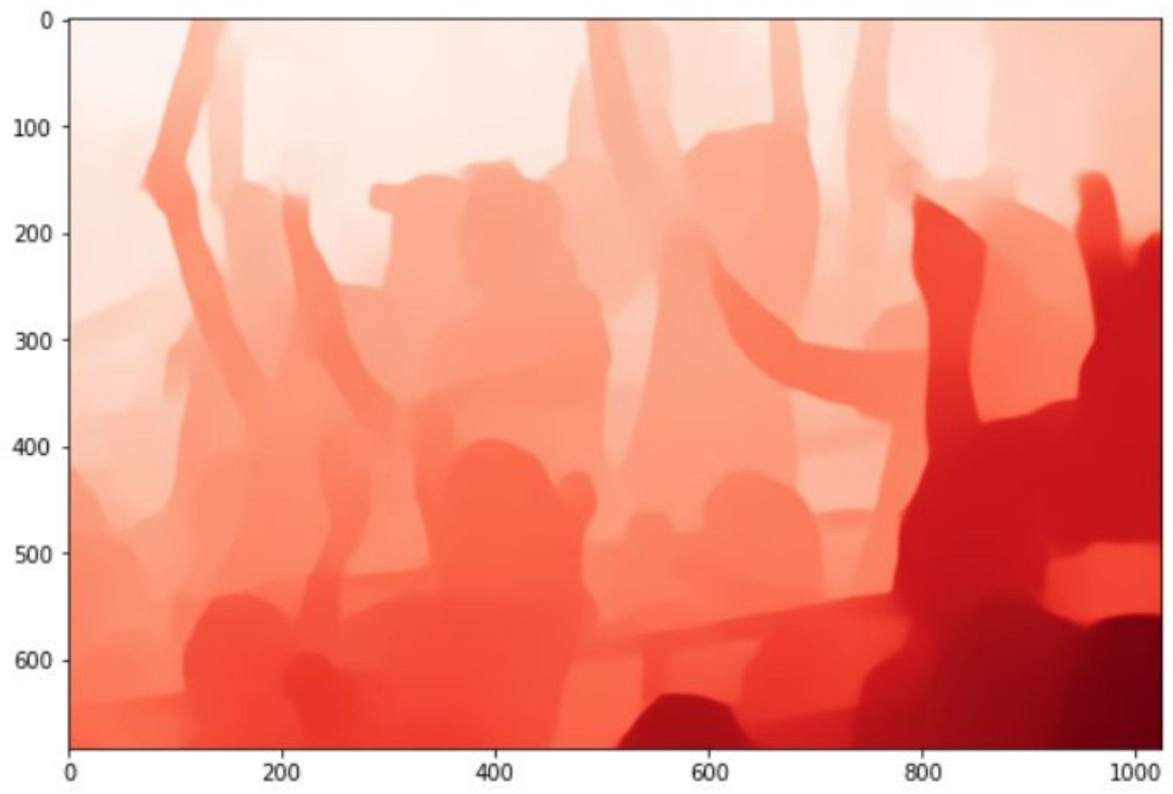




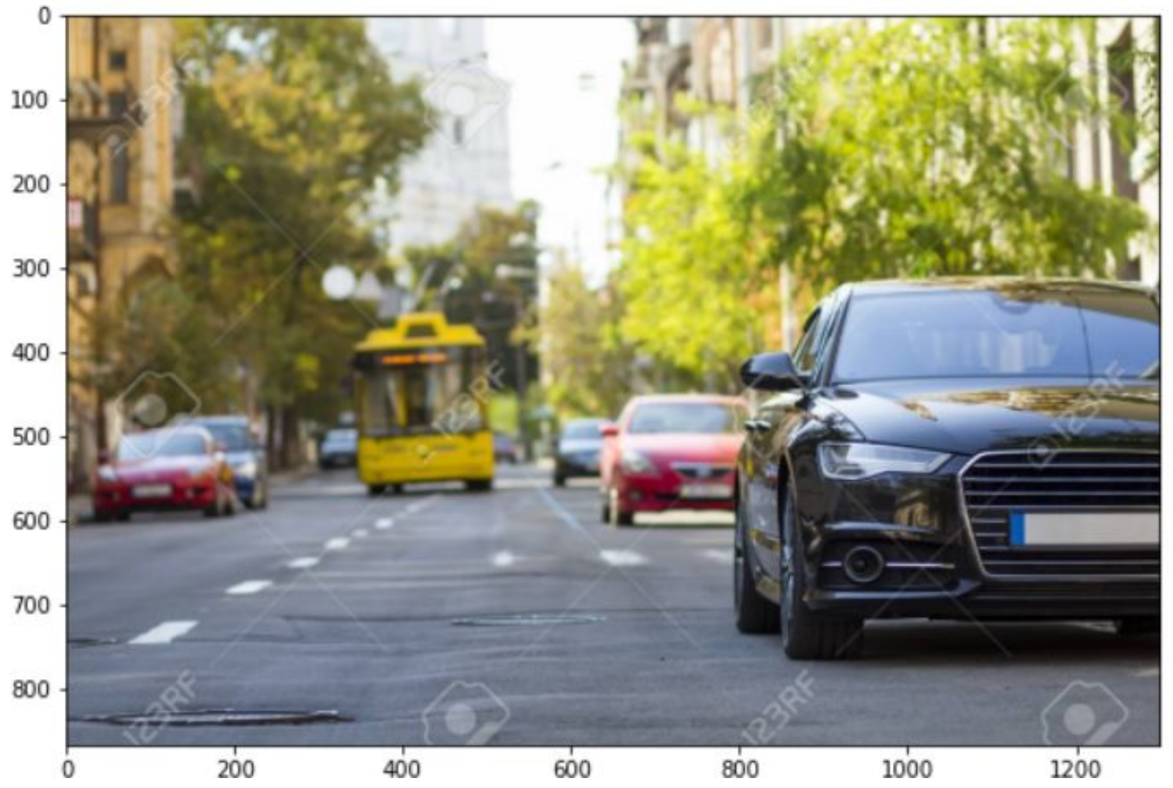
## Predictions and Testing on Random Internet Images

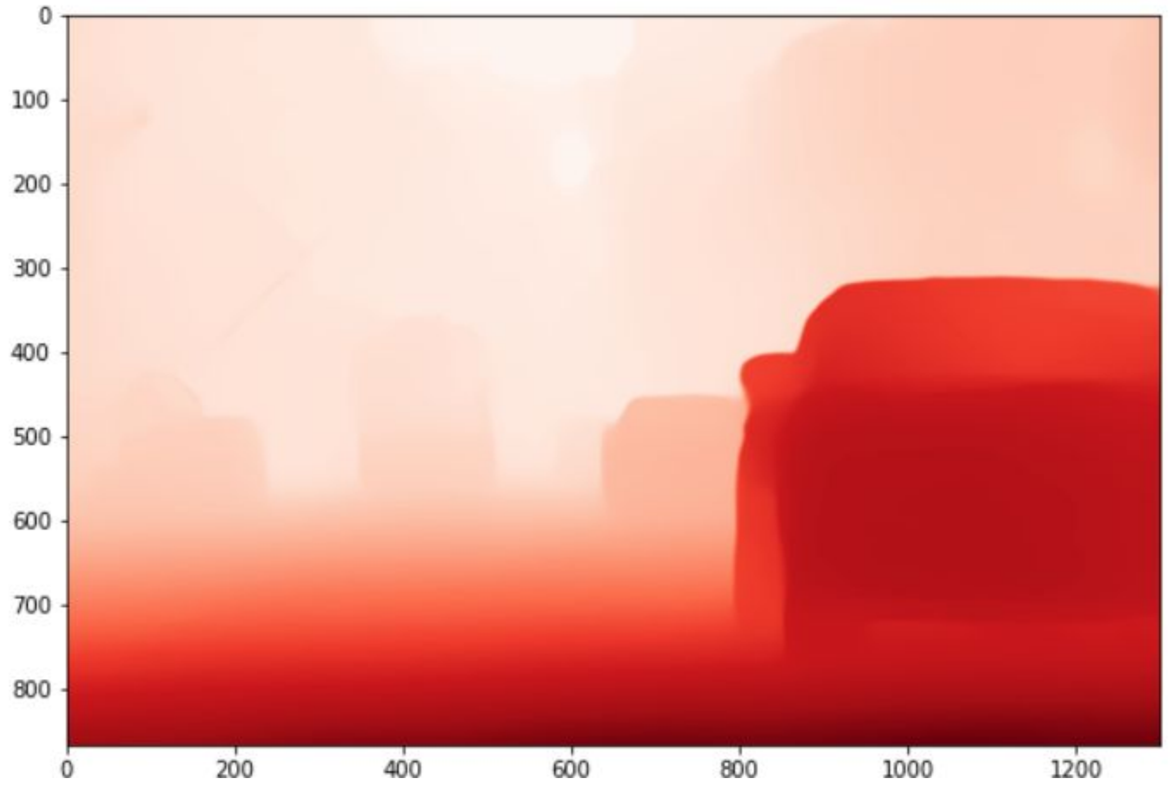


## Predictions and Testing on Random Internet Images

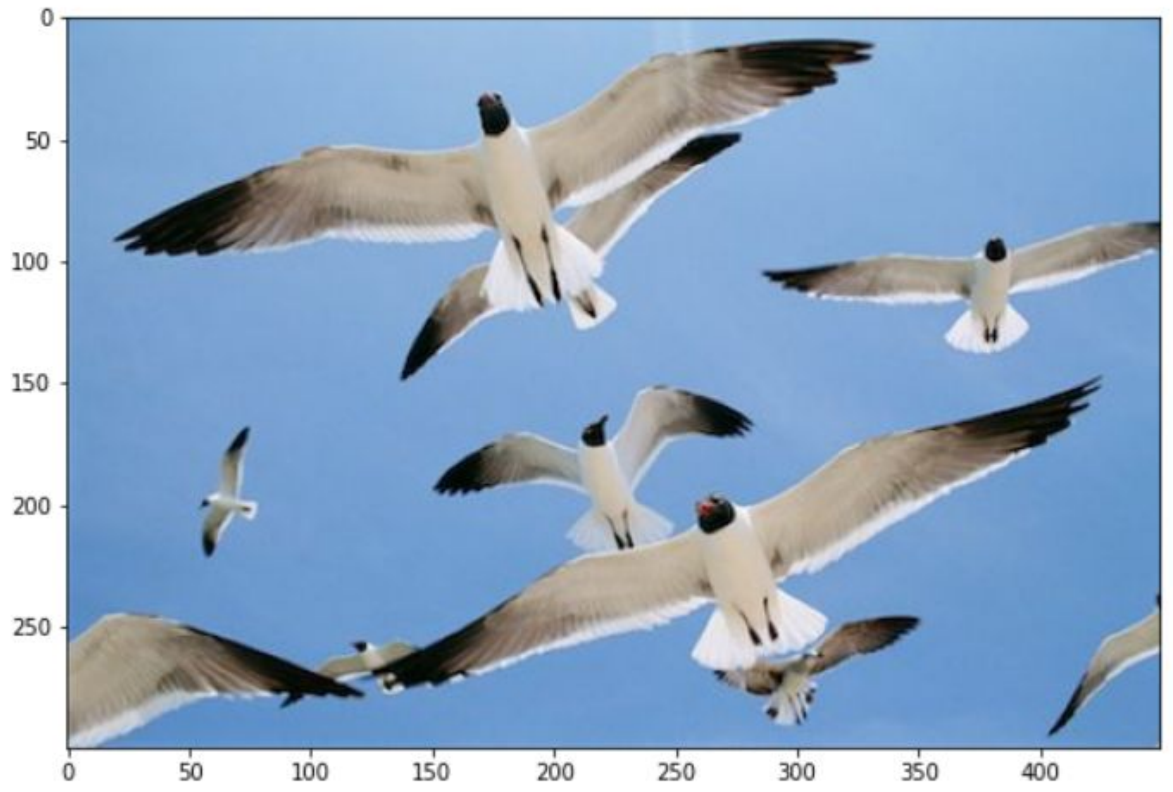


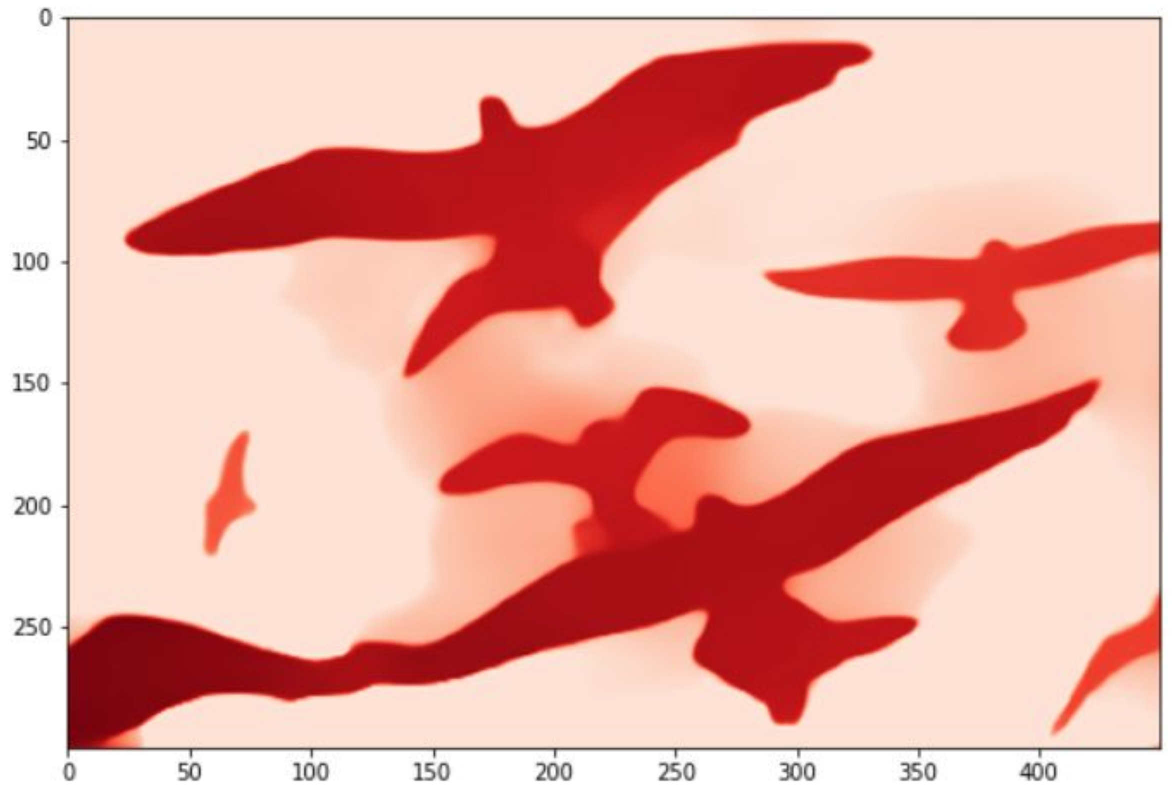
## Predictions and Testing on Random Internet Images



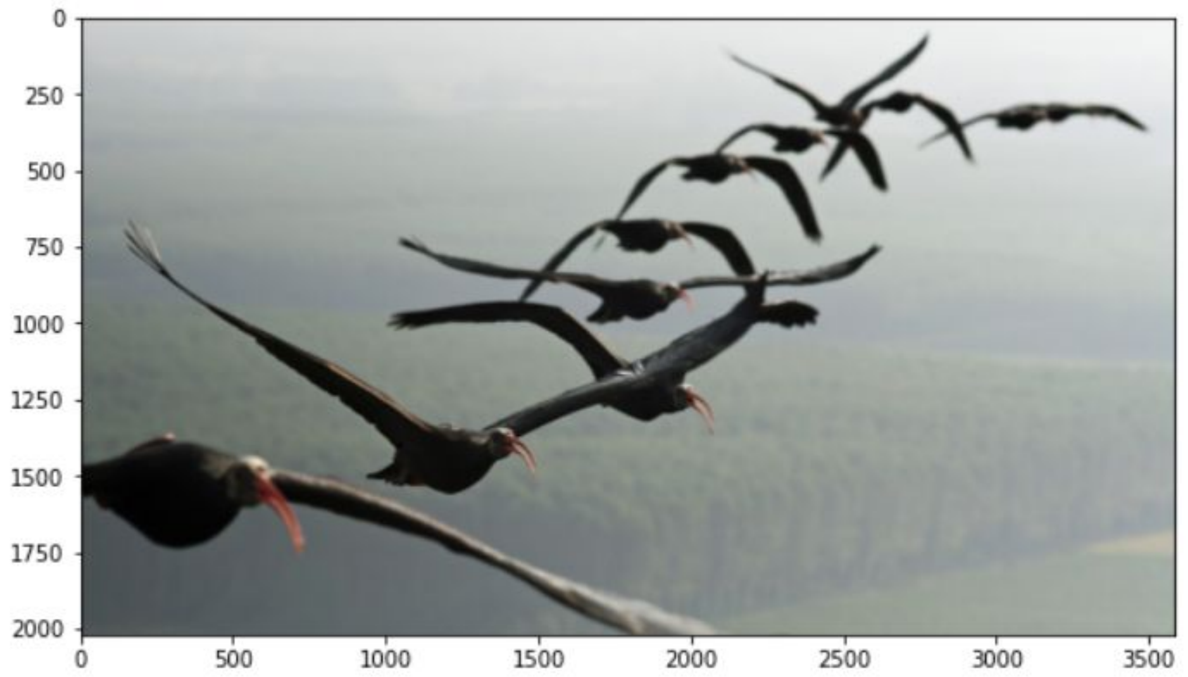


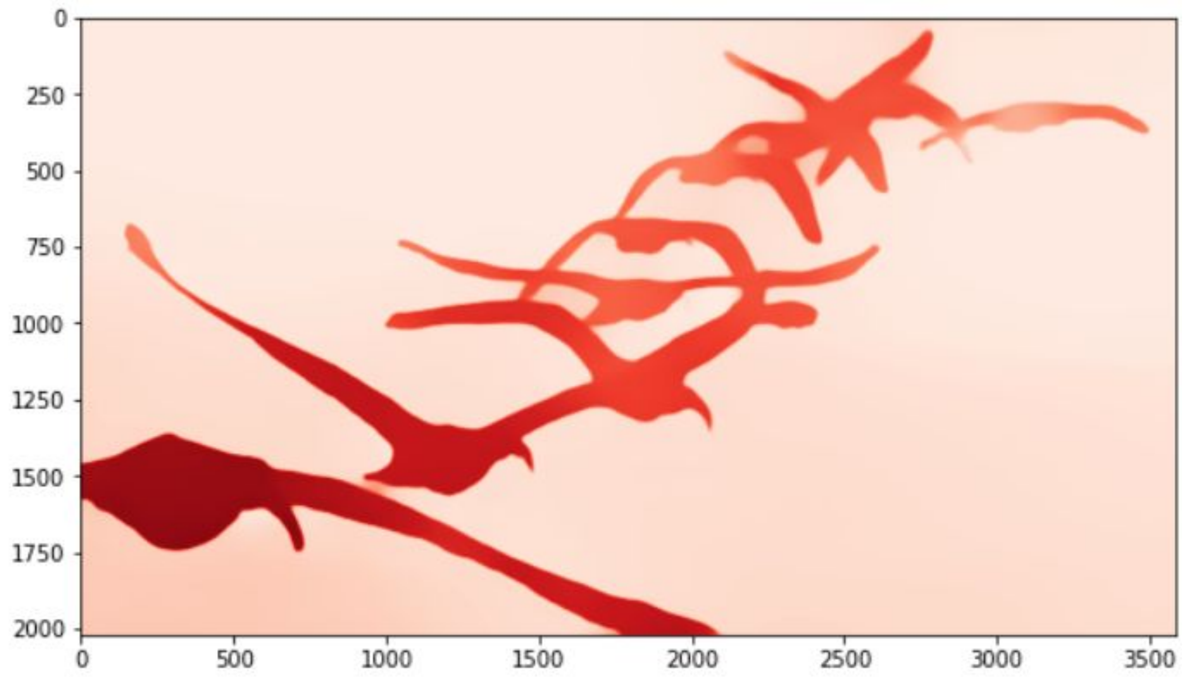
## Predictions and Testing on Random Internet Images



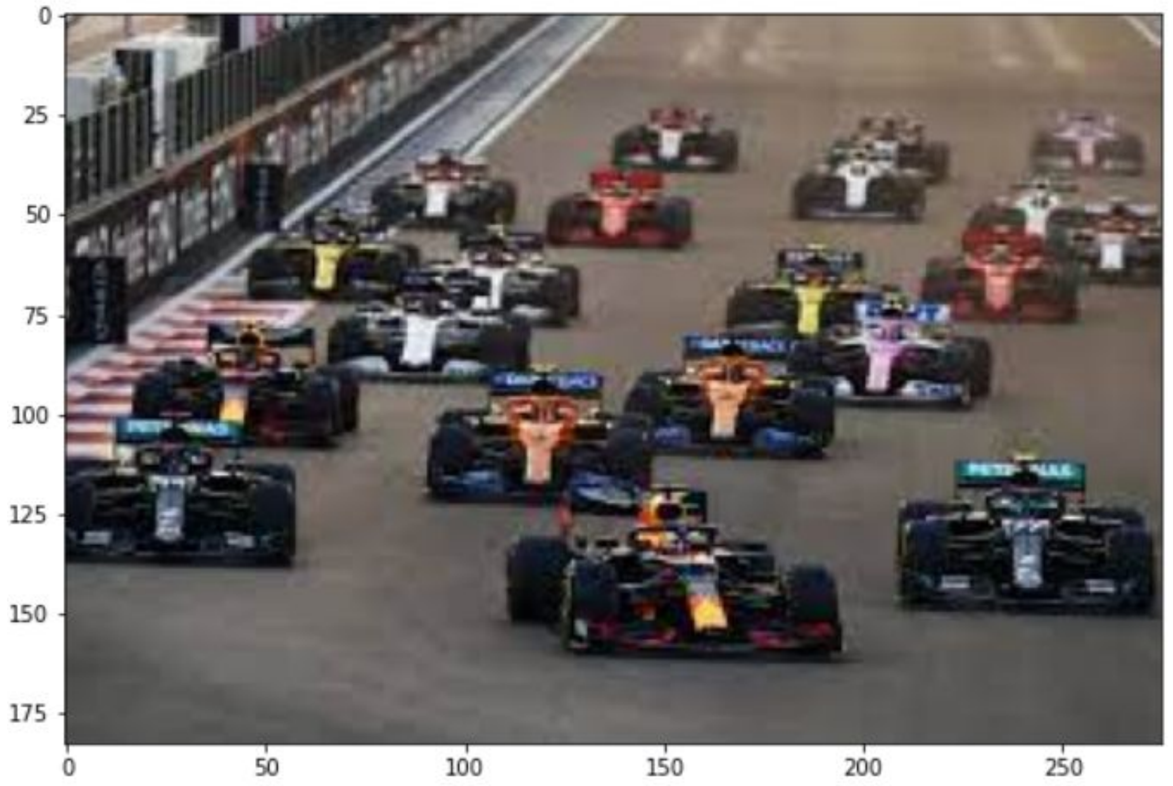


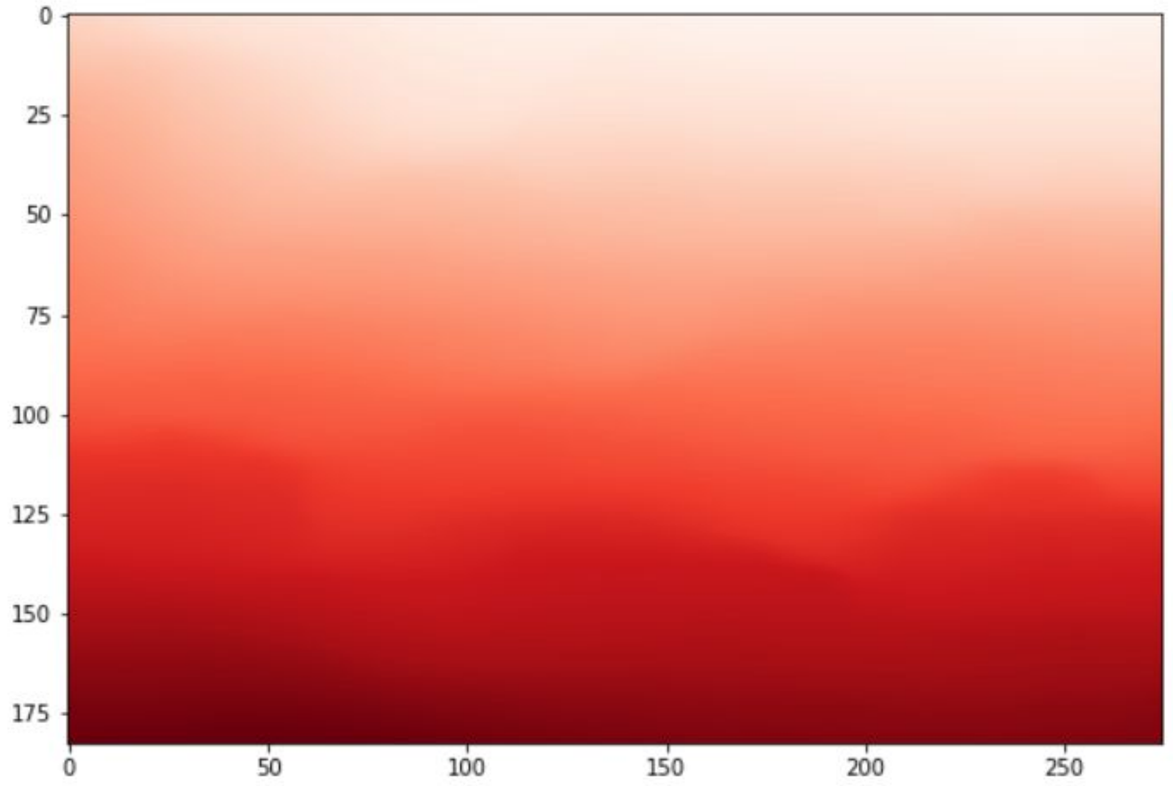
## Predictions and Testing on Random Internet Images



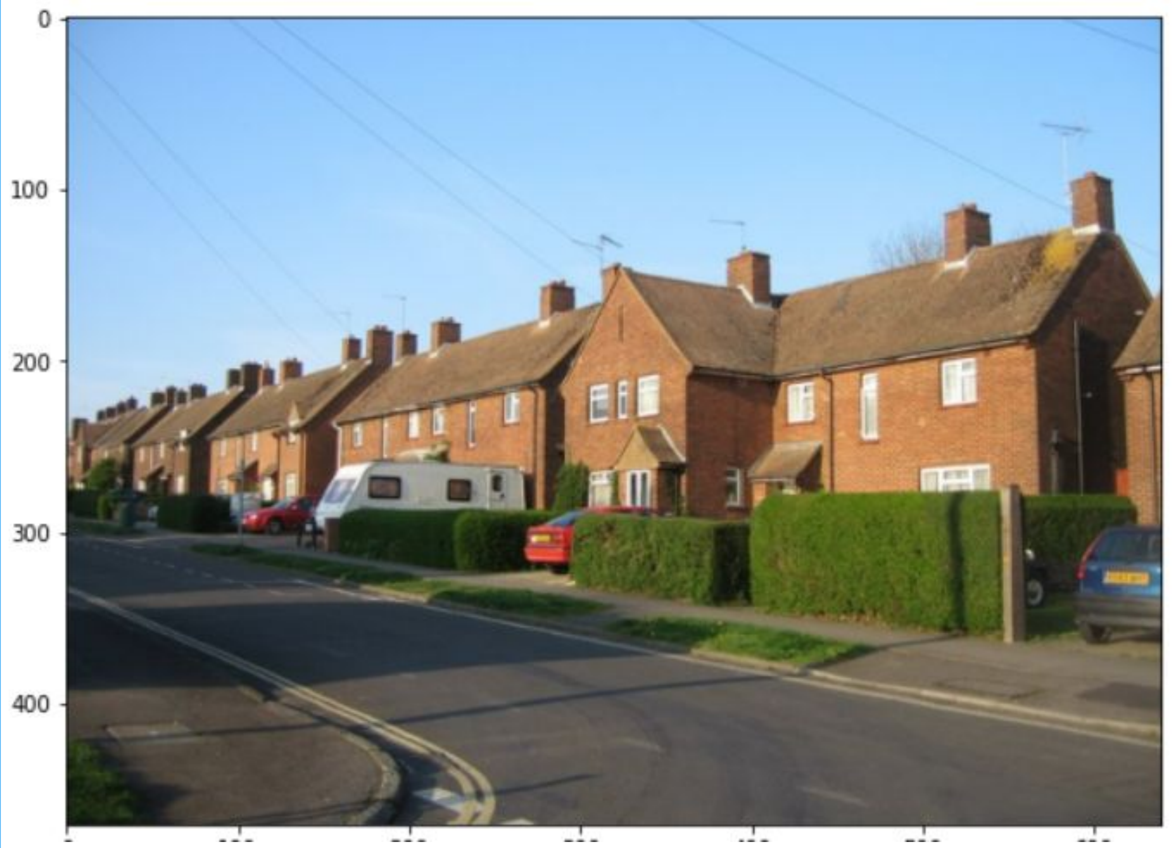


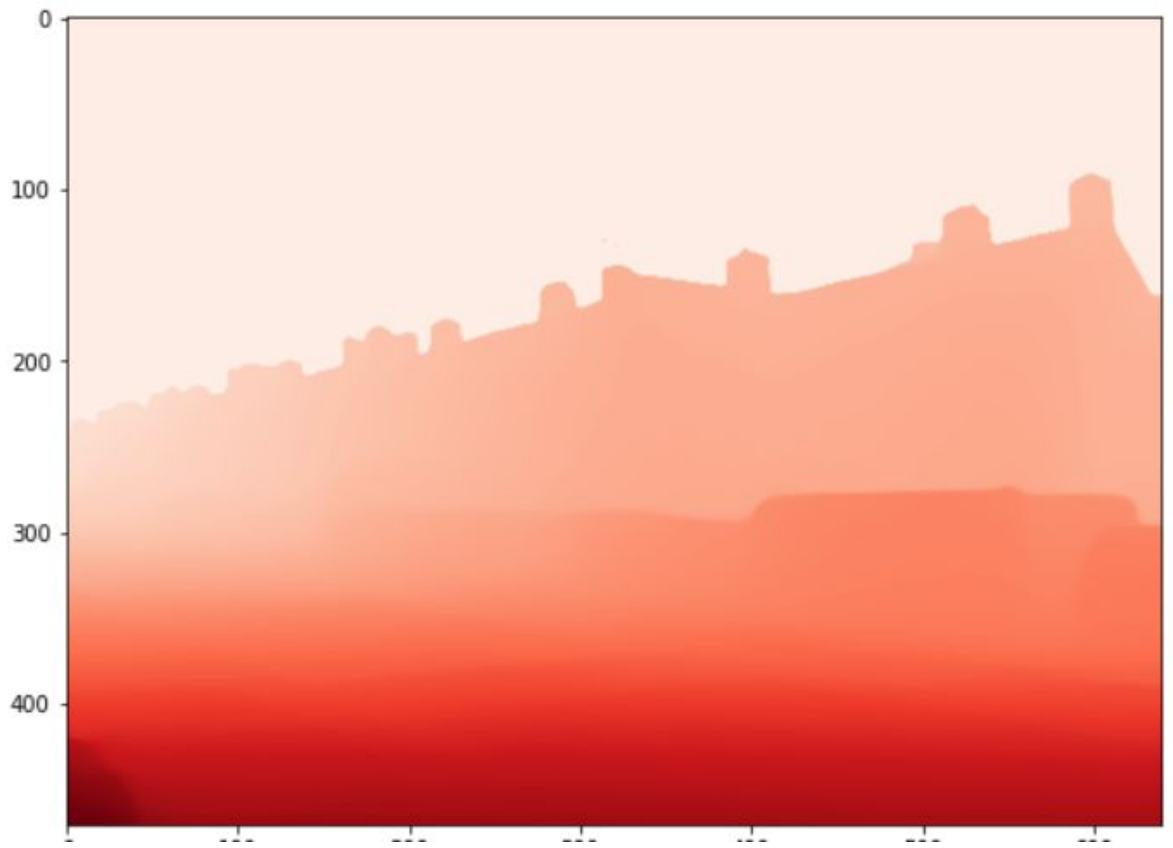
## Predictions and Testing on Random Internet Images



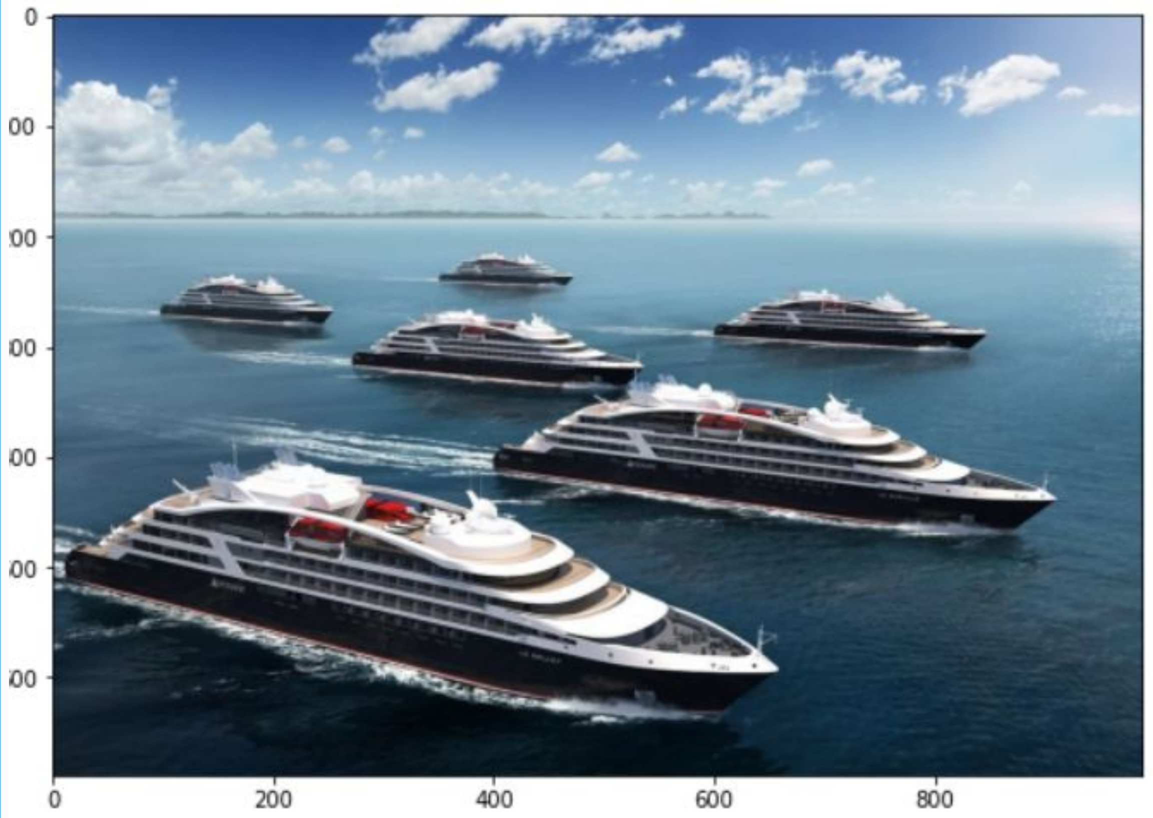


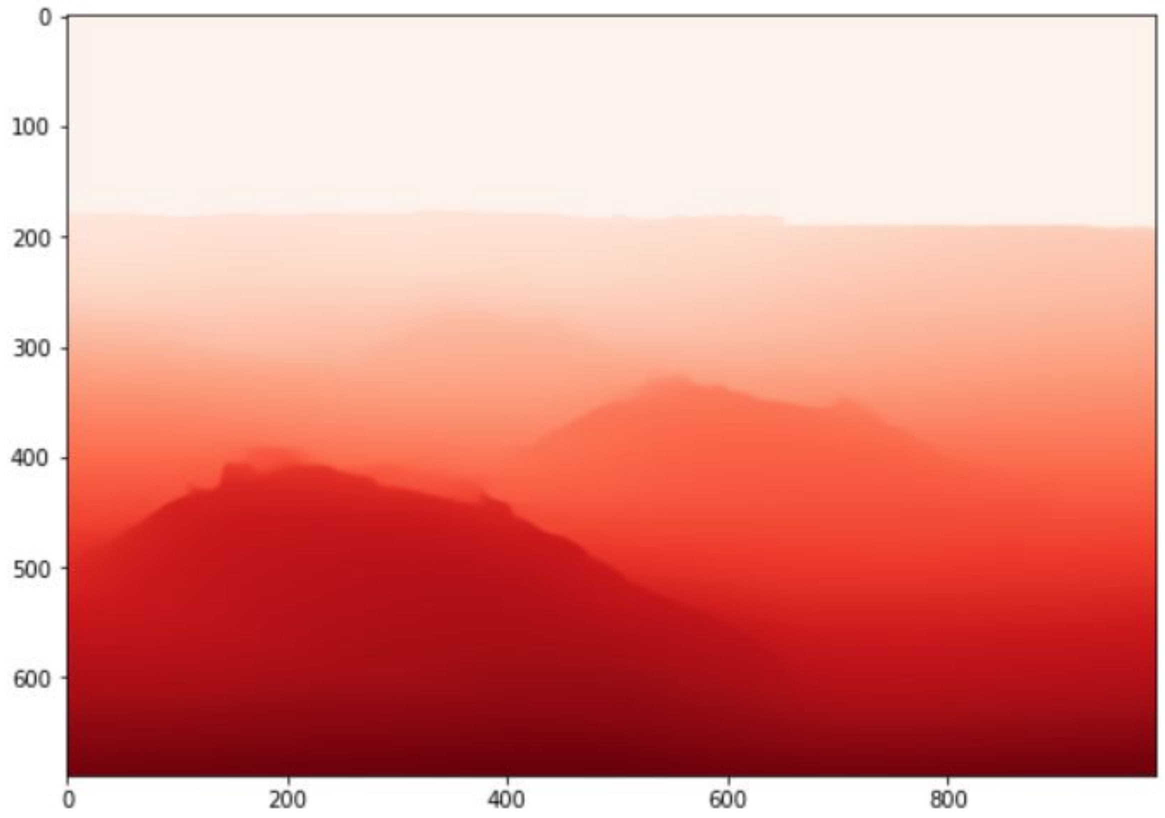
## Predictions and Testing on Random Internet Images





## Predictions and Testing on Random Internet Images





## Conclusion

- Our model is generalized well to dataset therefore performs well.
- Does Not performs well on inputs with too many subjects in them or some outlier inputs
- Therefore our model has medium Recall but medium Precision
- Fine tuning requires more work for s.o.a results
    - Train More
    - Search for better hyperparameter selection
- *Future Scope*
    - More narrow dataset can lead to much better model
        - Instead of generalized model if we train model specific to vehicle distance detection, it can outperform lidar

## References

[1] Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer
https://arxiv.org/pdf/1907.01341.pdf

[2] Vision Transformers for Dense Prediction
https://arxiv.org/pdf/2103.13413.pdf

[3] MegaDepth: Learning Single-View Depth Prediction from Internet Photos
https://www.cs.cornell.edu/projects/megadepth/

[4] IWeb Stereo Video Supervision for Depth Prediction from Dynamic Scenes
https://sites.google.com/view/wsvd/home?authuser=0

[5] Depth Estimation on NYU-Depth V2
https://arxiv.org/pdf/2205.13543v2.pdf

[6] SDC-Depth: Semantic Divide-and-Conquer Network for Monocular Depth Estimation
https://arxiv.org/pdf/2205.13543v2.pdf

UPES

UNIVERSITY OF TOMORROW

# Thank You

## Timeline

Our month wise plan to complete the project is as follows

**August**

- Finalizing the dataset.
- Research on generative adversarial networks.
- Deciding the architecture of the model.

**September- October**

- Developing the architecture using pytorch
- Training the model using pytorch.
- Debugging the code.
- Retraining the model using wasserstein loss.

**November**

- Concluding the result
- Publishing the results.
- Submitting the project.