RGB Input

Pix2Pix Depth

Our Result

Output taken from our implementation of Pix2Pix.

Simulating DoF by Monocular Depth Estimation

Tinn Kukamjad, Ta Chaimongkol
CS 188 – Team 40

Problem

Depth of Field (DoF) is a term in optics which, in layman's terms, describes the distance between the focal point and the farthest object which is in focus.

In other words, how blurry the background is.

DOF = 0.8cm
(Photoskop, 2017)

DOF = 2.2cm

DOF = 12.4cm

Problem

Smartphones usually have very small focal lengths due to their small cameras so their DoF is usually deep. This is great as smartphones are usually used to capture images in a variety of environments.

However, sometimes photographers want a shallower DoF. This is not possible by using phone hardware.

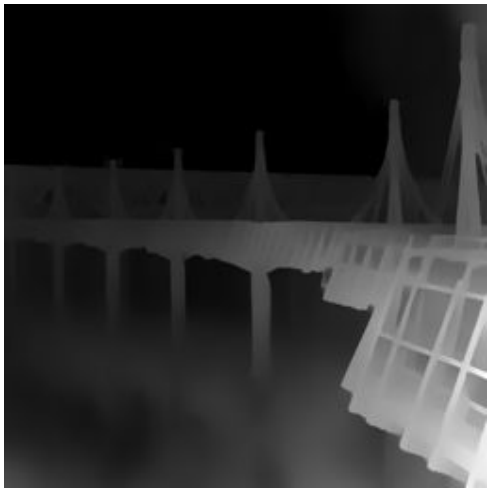
Example of Shallow DoF (Koebke, 2019)

Objective

We aim to develop a machine learning model which converts RGB data to a depth map. By using this depth map, we can generate artificial blur which allows phones to artificially replicate a camera with lower DoF.

A depth map is a 1 channel output where the brighter the pixel, the closer it is to the camera.

RGB (feature)

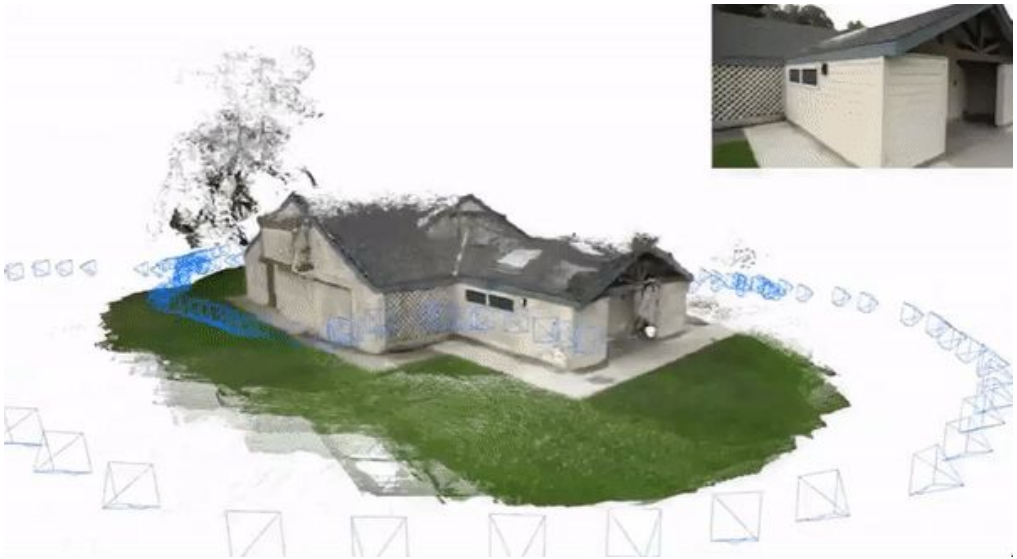


depth map (label)

Other Use-Cases

Depth map based imagery can be used in various other fields

- Simultaneous Localization and Mapping (SLaM)
- Photogrammetry
- Maneuvering such as in robotics and self driving cars



(Bhalerao, 2022)

Datasets

We compared a multitude of datasets. In the end we picked this to train the model on:

DIML (indoor): https://dimlrgbd.github.io/

- Realistic scenarios of data
- Accurate depth map ground truth
- Sourced from Kinect v2 & Zed stereo camera (reliable)
- Very large dataset, good to prevent overfitting

Classical Appr.

The classical approach has drawbacks

- Dual camera technology
  - Expensive
- Dedicated camera
  - Expensive
- Manually edit the photos in post
  - Time consuming
  - High skill required

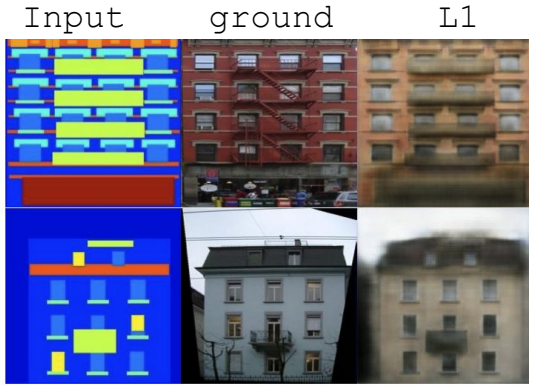
With deep learning

- Less expensive
- Minimal skill required for user to run

- More computationally intensive & has limitations

Naive DL Appr.

The naive deep learning approach is to have CNN which has a loss on the Euclidean distance between predicted and ground truth pixels. This usually produces blurry results. This looks unrealistic.



(Isola P., Et Al. 2018)

Our Appr.

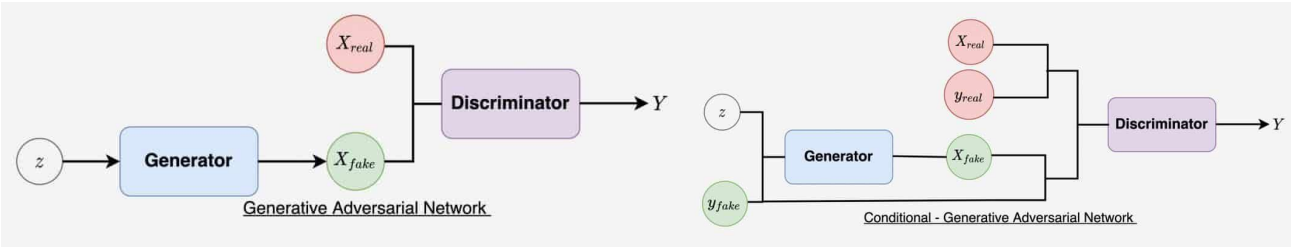
Our model implements and adapts `Pix2Pix` (Isola P., Et Al., 2018)

which originally was an Image-to-Image translation which utilizes a conditional General Adversarial Network (GAN).

GANs learn a loss function that adapts based on the data in order to produce more realistic results.

Traditional GANs have two model, a generator and discriminator.

- Generator generates image from noise
- Discriminator attempts to identify fake images
- Generator attempt to trick Discriminator



(Sharma, 2021)

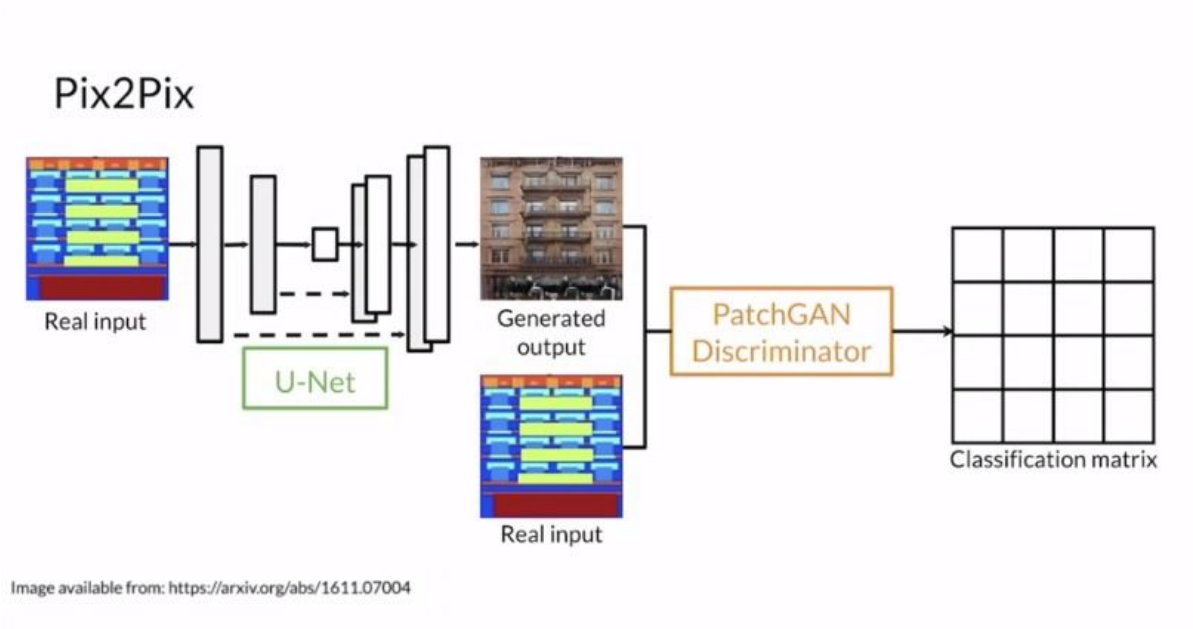
Method

Conditional GANs are modified GANs.

- Discriminators are also trained on the fake prediction and features.
- This means that the discriminator is aware of the style it is trying to produce.
- Gave better results than traditional GANs at modifying image styles (Redford A., Et Al., 2016)

(Reford A., Et Al., 2016)

Method



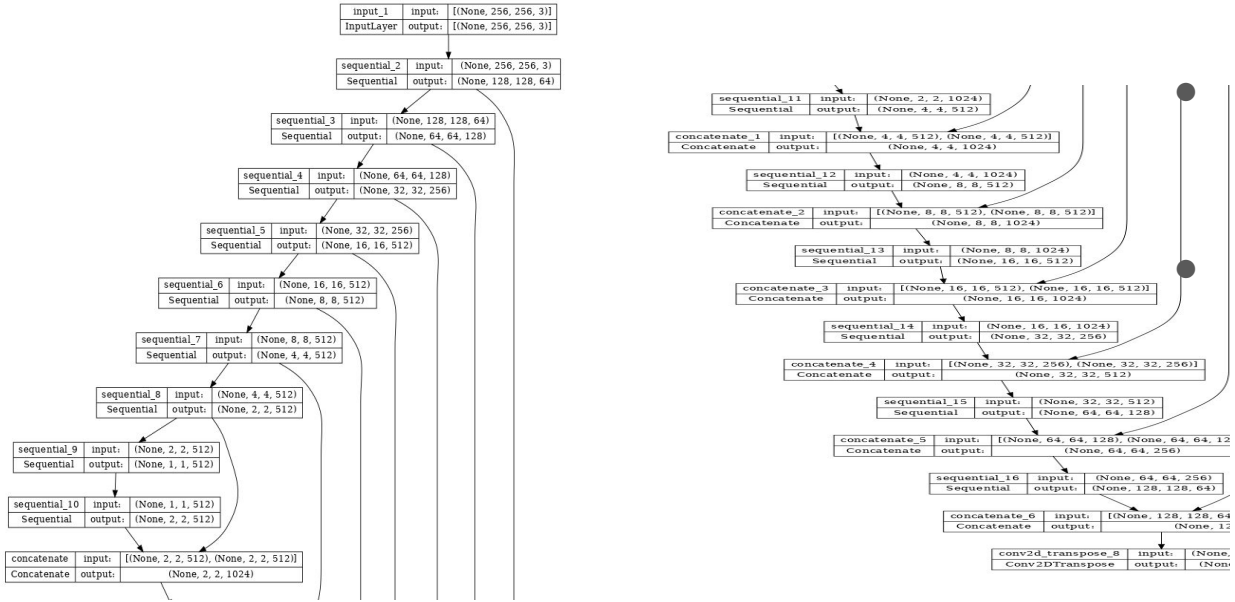
(Haiku Tech Center, 2020)

Method

Generator: “U-net”

(Ronneberger, Et al., 2015)

- Won 2012 EM segmentation challenge
- Upsampling has large number of feature channels to keep context info.
- Minibatch SGD, Adam Solver



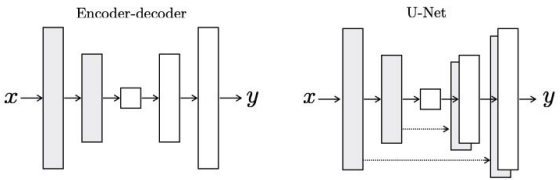
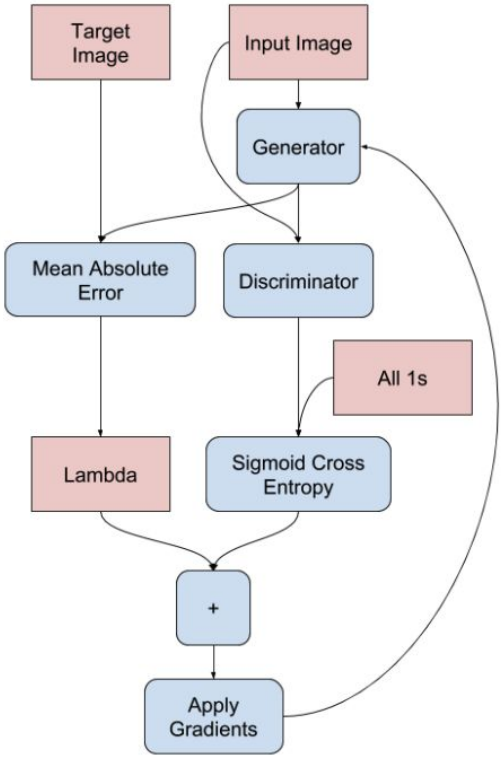


Figure 3: Two choices for the architecture of the generator. The “U-Net” [50] is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks.

Method



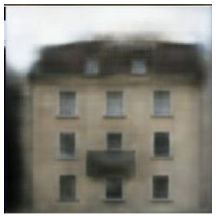
- As discussed earlier, L1 loss is not sufficient (blurry results)
- Generator must
  - Minimize L1 loss (low frequency correctness)
  - Fool discriminator (high frequency correctness)
- We use a discriminator which focuses only on patches
  - Thus, it looks at only high frequency correctness

(Isola et al. 2018)

Method

- *x* is the real rgb (real input)
- *y* is the real depth map (real output)
- z is noise vector
- G(x, z) is the generator's depth map (fake output)

L1



cGAN

cGAN + L1

L1 Loss (Naive method, using only CNN)

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x,z)\|_1].$$

GAN Loss (Introduce discriminator)

$$\mathcal{L}_{GAN}(G,D) = \mathbb{E}_y[\log D(y)] + \mathbb{E}_{x,z}[\log(1 - D(G(x,z))].$$

cGAN Loss (Introduce conditional D.)

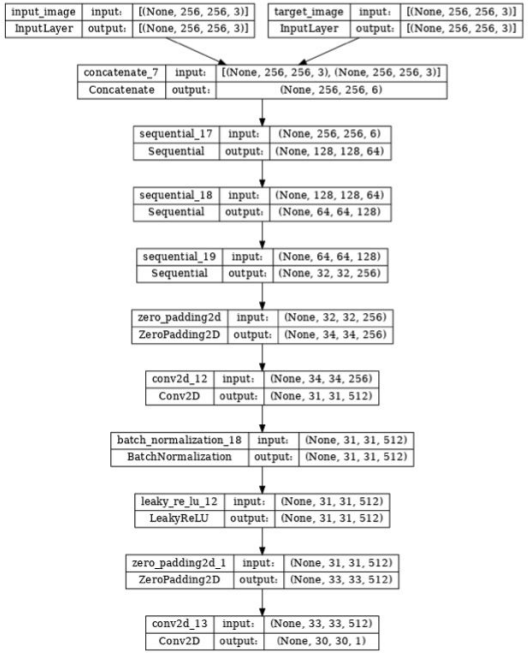
$$\mathcal{L}_{cGAN}(G,D) = \mathbb{E}_{x,y}[\log D(x,y)] + \mathbb{E}_{x,z}[\log(1 - D(x,G(x,z))],$$

cGAN + L1 Loss

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G,D) + \lambda \mathcal{L}_{L1}(G).$$

Tinn, Ta, 2024

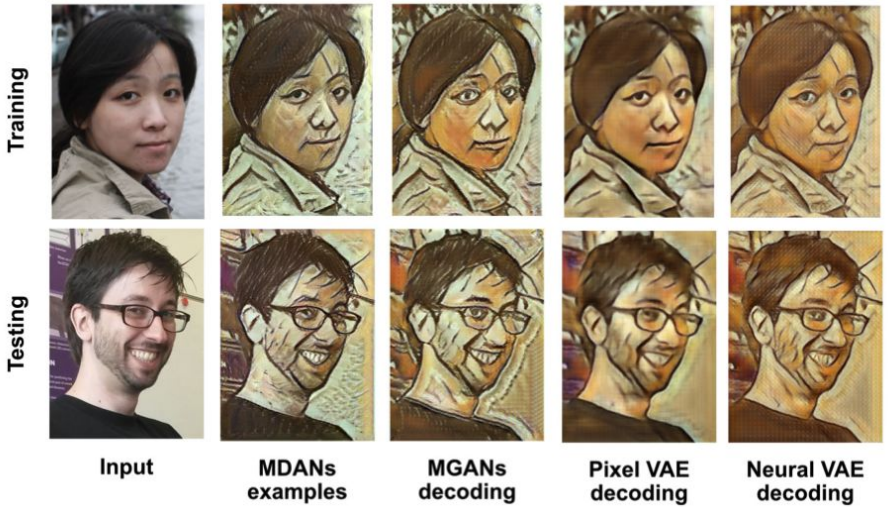
Method



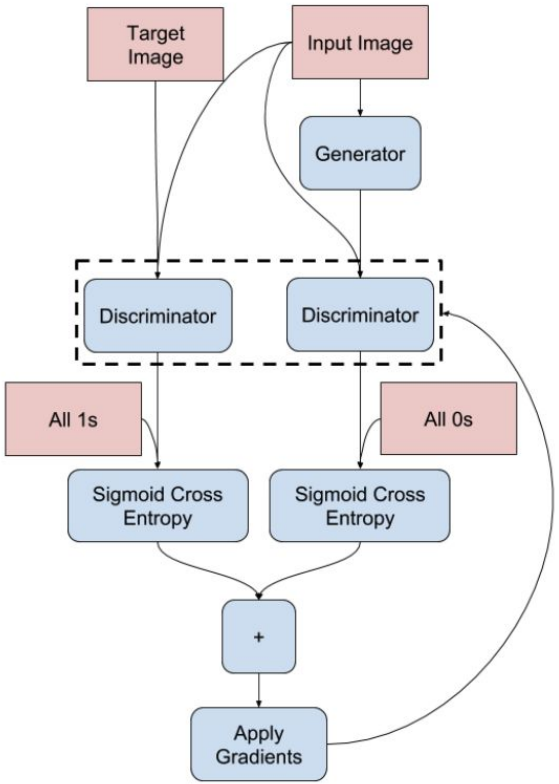
Discriminator: “PatchGAN”

(Li C., Wand M., 2016)

- Designed to synthesize textures
- Achieved stylized images using VGG-16
- Markovian random field to



Method



Discriminator training

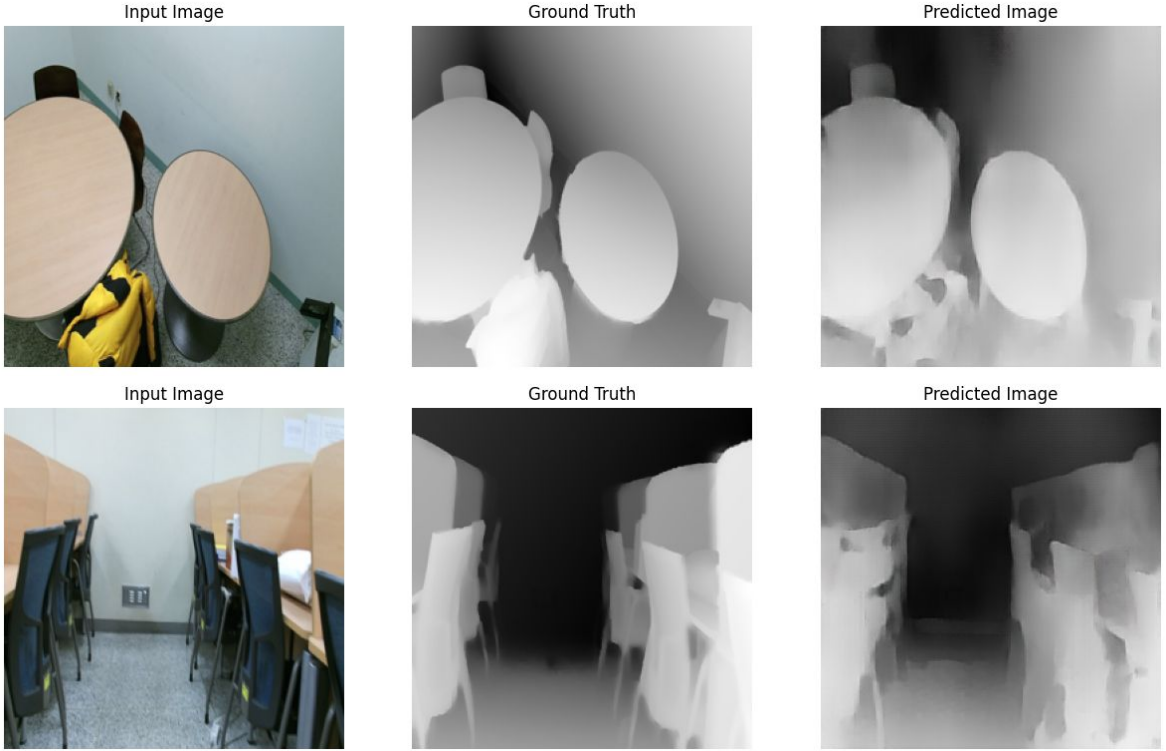
- Takes in real image and generated images
- `real_loss` is a SCE loss of real images
  - and array of 1s
- `generated_loss` is a SCE of generated images
  - and array of 0s
- `total_loss` is sum of the losses

```
def discriminator_loss(disc_real_output, disc_generated_output):
  real_loss = loss_object(tf.ones_like(disc_real_output), disc_real_output)

  generated_loss = loss_object(tf.zeros_like(disc_generated_output), disc_generated_output)

  total_disc_loss = real_loss + generated_loss

  return total_disc_loss
```

Tinn, Ta, 2024

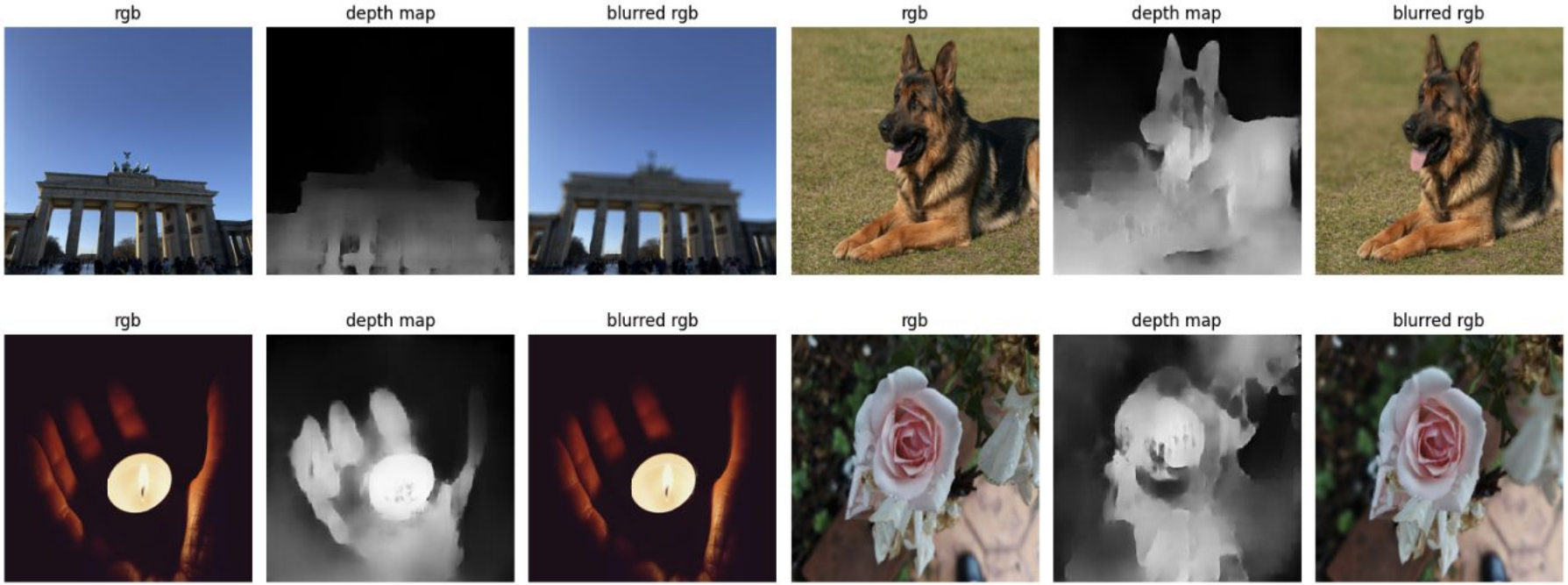
Results

This is what our implementation of Pix2Pix trained on depth map produces.



Results

After feeding in rgb as features and depth map as the labels here are some curated examples from the test set



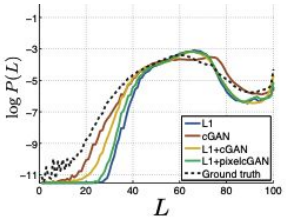
Tinn, Ta, 2024

Results

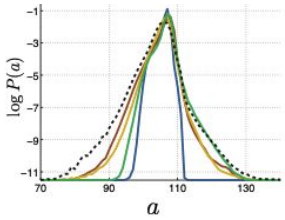
FCN-8s Cityscape image semantic segmentation results

Loss	Photo → Map % Turkers labeled *real*	Map → Photo % Turkers labeled *real*
L1	2.8% ± 1.0%	0.8% ± 0.3%
L1+cGAN	6.1% ± 1.3%	**18.9%** ± **2.5%**

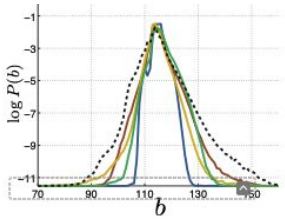
Loss	Per-pixel acc.	Per-class acc.	Class IOU
L1	0.42	0.15	0.11
GAN	0.22	0.05	0.01
cGAN	0.57	0.22	0.16
L1+GAN	0.64	0.20	0.15
L1+cGAN	**0.66**	**0.23**	**0.17**
Ground truth	0.80	0.26	0.21



(a)



(b)



(c)

Loss	Histogram intersection against ground truth L	a	b
L1	0.81	0.69	0.70
cGAN	**0.87**	0.74	**0.84**
L1+cGAN	0.86	**0.84**	0.82
PixelGAN	0.83	0.68	0.78

(d)

Tinn, Ta, 2024.

Works Cited

Koebke M. (2019). Picture of Rabbit. *Pixabay*.

Bhalerao C. (2022). Simultaneous Location and Mapping SLaM Systems

https://medium.com/geekculture/simultaneous-localization-and-mapping-slam-systems-44d4369fcb46

Photoskop. (2017).

https://www.photoskop.com/player.html?l=all&ch=3&sec=0

Haiku Tech Center. (2020). Pix2Pix GAN Architecture For Image.

https://www.haikutechcenter.com/2020/10/pix2pix-gan-architecture-for-image-to.html

Sharma A. (2021) Conditional GAN cGAN in Pytorch and TF.

https://learnopencv.com/conditional-gan-cgan-in-pytorch-and-tensorflow/

Isola P., Et Al. (2018) Image-to-Image Translation with Conditional Adversarial Networks.

https://arxiv.org/pdf/1611.07004.pdf

Radford A., Et Al. (2016) Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.

https://arxiv.org/pdf/1511.06434.pdf

Works Cited

Ronneberger O., Et Al. (2015). U-net: Convolutional networks for biomedical image segmentation.*MIC CAI.*

https://arxiv.org/pdf/1505.04597.pdf

Li C., Wand M., (2016) Precomputed real-time texture synthesis with markovian generative adversarial networks. *ECCV*.

https://arxiv.org/pdf/1604.04382.pdf

Tinn Kukamjad
Ta Chaimongkol