# Semantic segmentation transfer

# Motivation

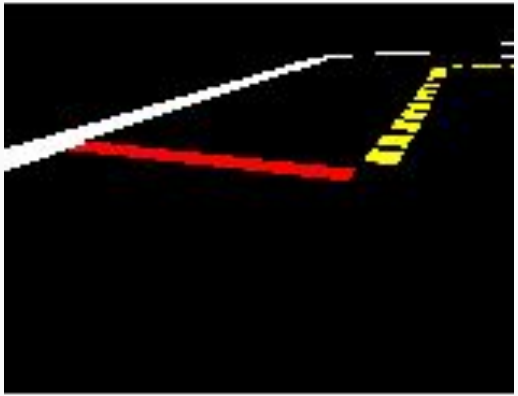
Interest towards unsupervised learning

Leverage data from simulator

Improve line detection in current Duckiebot system

# Line segmentation

# What is it?



# How was it done? – Threshold

Sources: [1] https://en.wikipedia.org/wiki/Thresholding_(image_processing)
[2] Nir Milstein, “Image Segmentation by Adaptive Thresholding”,Spring 1998.

# How is it done? – Parametric models

Learning a parametric model

$$\theta^* = \arg\min_{\theta^*} \mathcal{L}(Y, \hat{Y})$$

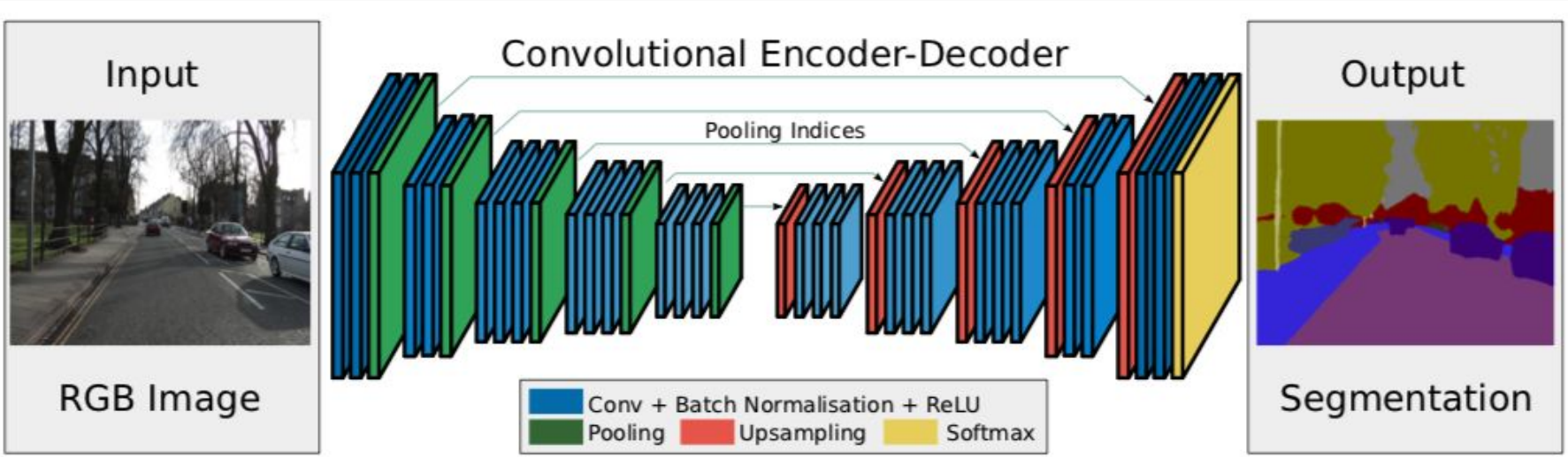
using **Gradient descent**

$$\theta' = \theta - \alpha \nabla_\theta f(x; \theta)$$

# How we did it?

Deep learning

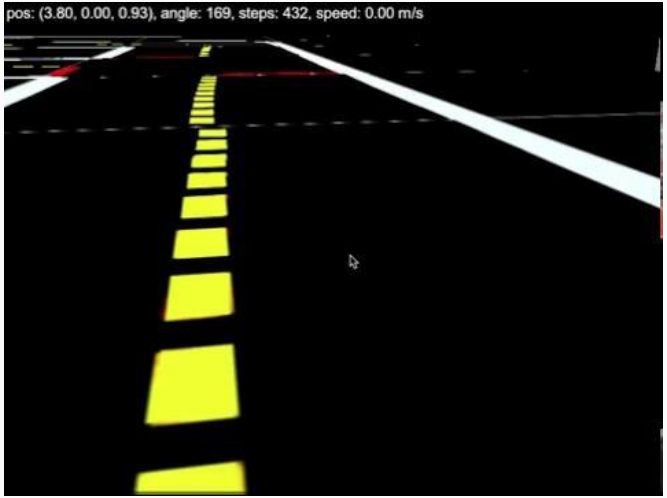
# How we did it? – Segnet




Source SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation https://arxiv.org/pdf/1511.00561.pdf

# How we did it? – Gathering data

1. Modify the simulator to generate segmented images.
2. Generate a couple of real and segmented images. (500k)
3. ???
4. Profit.



# Transferring to simulation

# Why transfer?

Relative cost of labelling real world / simulated data

Low cost of unlabelled real world Duckietown data

Safer training of models

# Why use a GAN?

Distributional approach

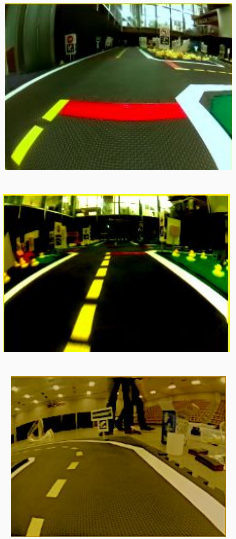
Don’t need the distribution, just a way of sampling it

Different unsupervised models have different goals

# How is it done? – in images

Distribution of *real* Duckietown images

Function that we control



Distribution of **fake** *simulated* images

Detective that we control

Distribution of **true** *simulated* images

# How is it done? – in simple math

$$\mathrm{P}(X_{real}) \rightarrow f_\theta(x) \rightarrow \mathrm{Q}(X_{sim}) \rightarrow g_\phi(x) \leftarrow \mathrm{P}(X_{sim})$$

Distribution of *real* Duckietown images

A function that we control

Distribution of **fake** *simulated* images

Detective that we control

Distribution of **true** *simulated* images

# How is it done? – in practice

```
import torch

data = load('wow_much_data.pth')
model = torch.nn.ImASuperModel()

model.fit(data)
```

# How is it done? – in practice

$f_\theta(x)$ : convolutional auto-encoder structure from [1]

$g_\phi(x)$ : convolutional encoder from DCGAN [2]

End-to-End training procedure from [3] (as seen in class!):

- Iteratively trains segmentation and generator/discriminator

[1] Perceptual Losses for Real-Time Style Transfer and Super-Resolution (Johnson *et al.*)
[2] Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks (Radford *et al.*)
[3] Real-to-Virtual Domain Unification for End-to-End Autonomous Driving (Yang *et al.*)

# What’s next?

# Next steps & hurdles to come

Controlling the GAN

Ensuring **sufficient** generalization in transfer

Smallest model that offers best performance

Running models on Duckiebot

Inserting line segmentation in ROS pipeline

# One more thing...