Pedestrian Trajectory Forecasting

MIDTERM WORK REPORT

Stanford University, Vision and Learning Laboratory
Giulio Autelitano, 6 Jan 2020

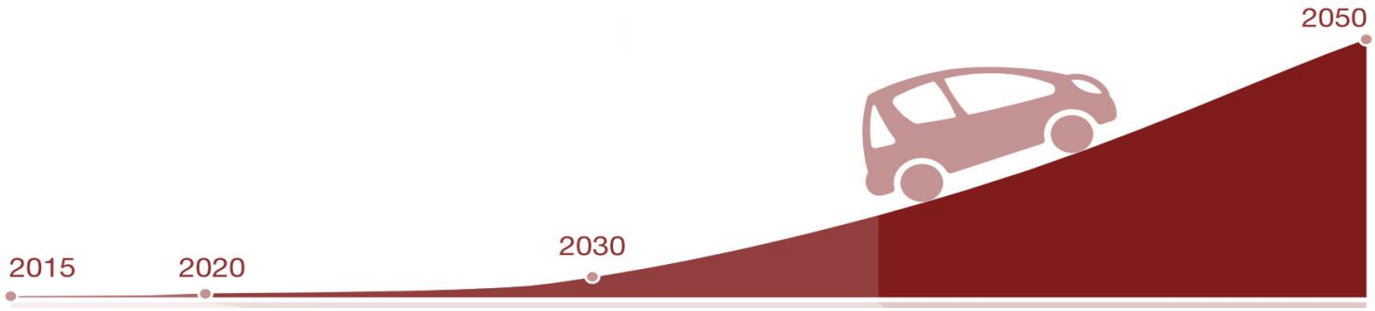
Stanford | ENGINEERING

ASI Agenzia Spaziale Italiana

Cultural Association of Italians at Fermilab

Why addressing pedestrian trajectory forecasting

Autonomous vehicles to become the major means of transport by 2050



First Stage: 2020
- Development of AVs for consumers.
- New mobility models begin to emerge.
- Car OEMs begin to assess strategic impact

Second Stage: 2030
- Consumers begin to adopt AVs

Third Stage: 2050
- AVs become the primary means of transport
- AVs free up to 50 mins/day
- Vehicles crashes fall by 90%, saving billions of dollars

McKinsey & Company, 2015

Video: Pedestrian are not easily predictable

Automation Levels

Pedestrian recognition is one of the key features for AVs next leap

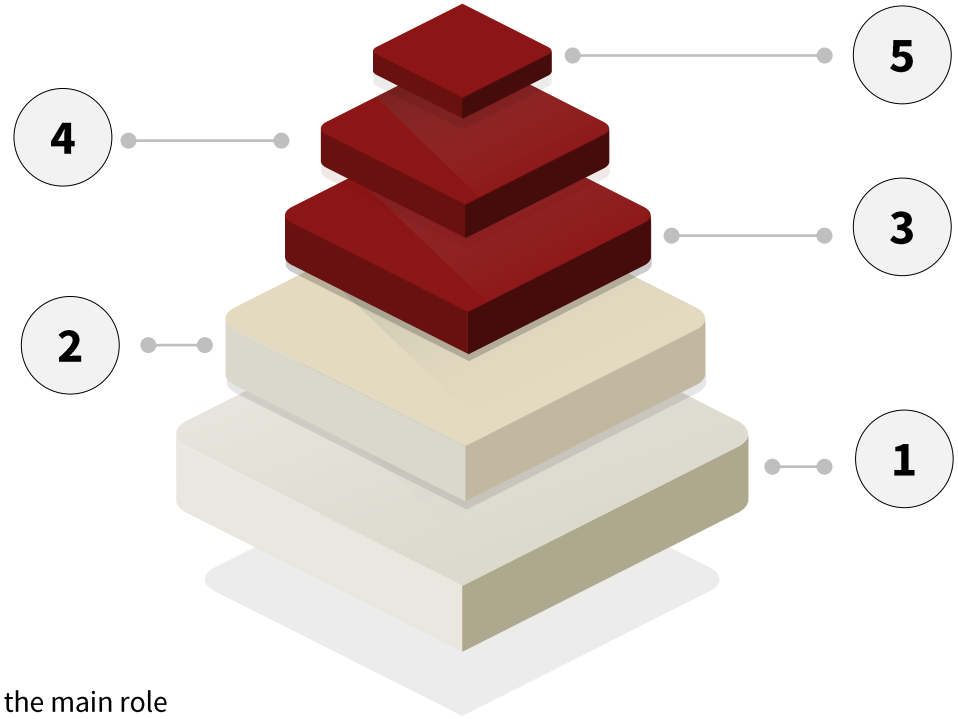
5 — Full Automation
No need for a driver. The vehicle is fully capable of handle every situation.

4 — High Automation
An automated driving system performs all aspects of the dynamic driving task in certain driving modes even if the human driver will do not respond appropriately to a request to intervene

3 — Conditional Automation
An automated driving system performs all aspects of the dynamic driving task in certain driving modes

2 — Partial Automation
The driver assistance system undertakes both of steering and acceleration/braking operations, with the expectation that the human driver performs all remaining aspects of the dynamic driving task

1 — Driver Assistance
The driver assistance system executes steering OR acceleration and braking operations, by using information about the driving environment. The human driver performs all remaining aspects of the dynamic driving task. The vehicle is fully under the driver's control.



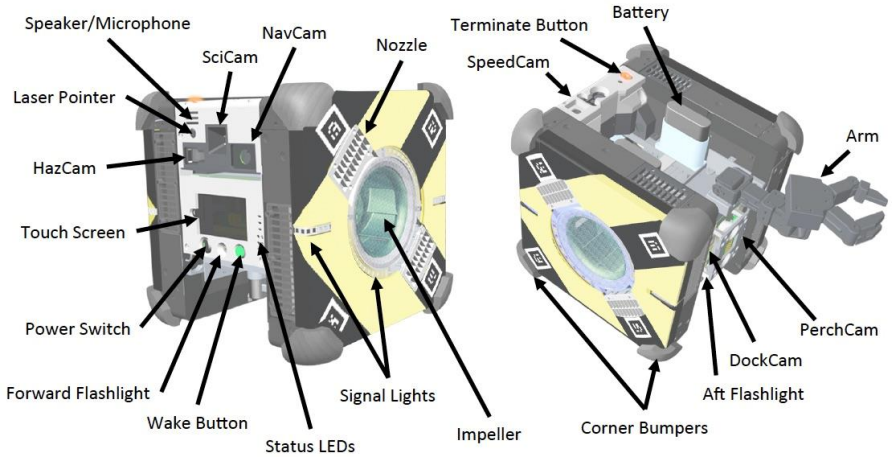
The car has the main role

The driver has the main role

Stanford University

Space Applications: Astrobee

Astrobee will help astronauts reduce time they spend on routine duties, leaving them to focus more on the things that only humans can do.



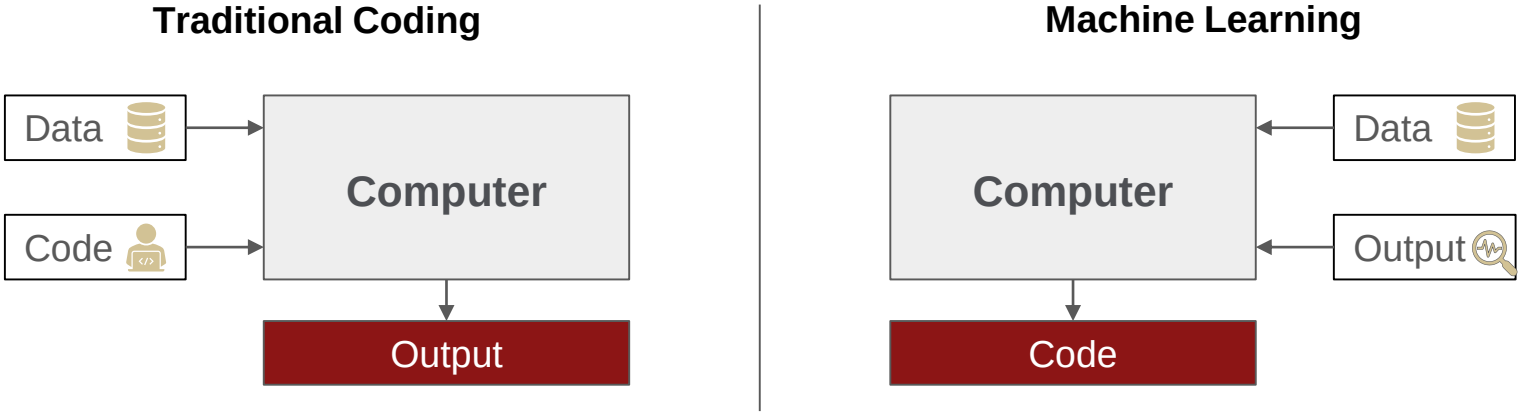
"It can fly autonomously throughout most of the US section of the ISS interior. […] Each Astrobee carries a suite of six cameras (including LIDAR sensors and a 21 MP RGB camera)"

NASA: Astrobee Science Guide

Machine Learning vs Traditional Coding

*"Machine Learning is a field of study that gives computers the ability to learn **without being explicitly programmed**."*

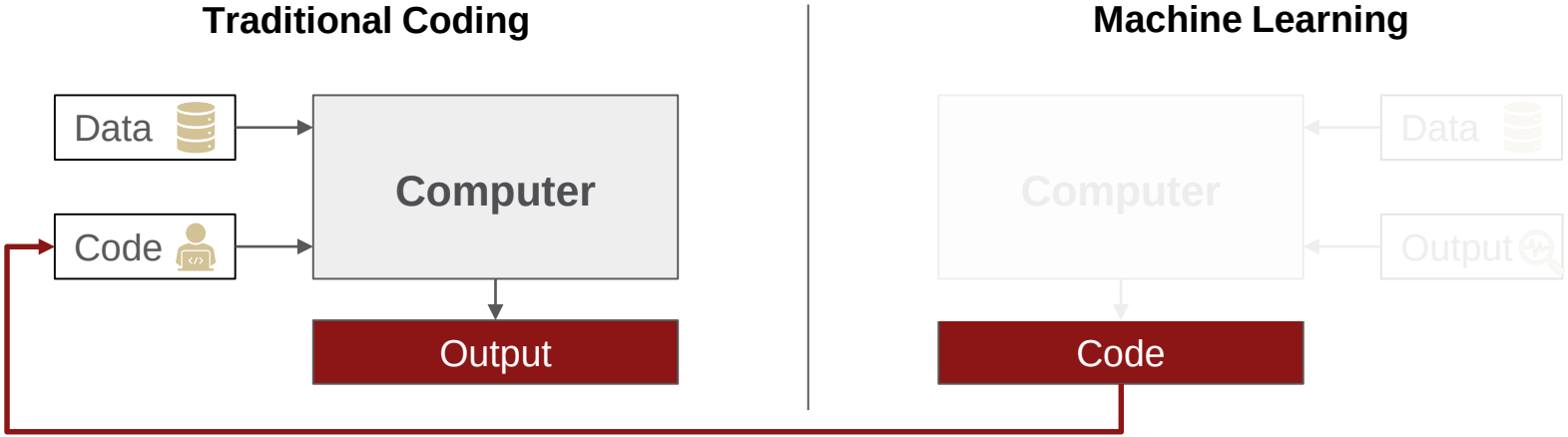
Arthur Samuel, 1959



Machine Learning vs Traditional Coding

*"Machine Learning is a field of study that gives computers the ability to learn **without being explicitly programmed**."*

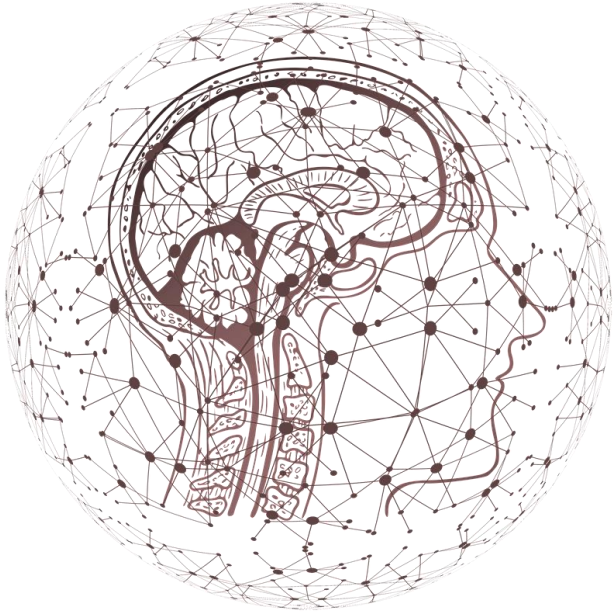
Arthur Samuel, 1959



Artificial Intelligence

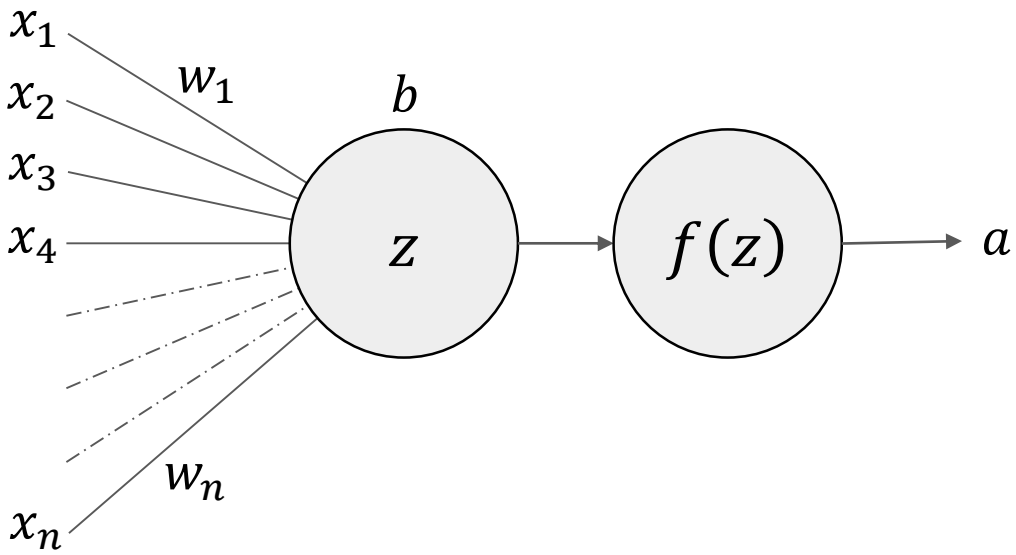
- Introductory Concepts
  - Perceptron
  - Activation Functions
  - Artificial Neural Networks
  - Gradient Descent
  - Back Propagation

- Advanced Networks
  - Convolutional Neural Networks
  - Generative Antagonist Networks
  - Long-Short Term Memory LSTM Networks
  - Attention Networks



Perceptron: ML Elementary Unit

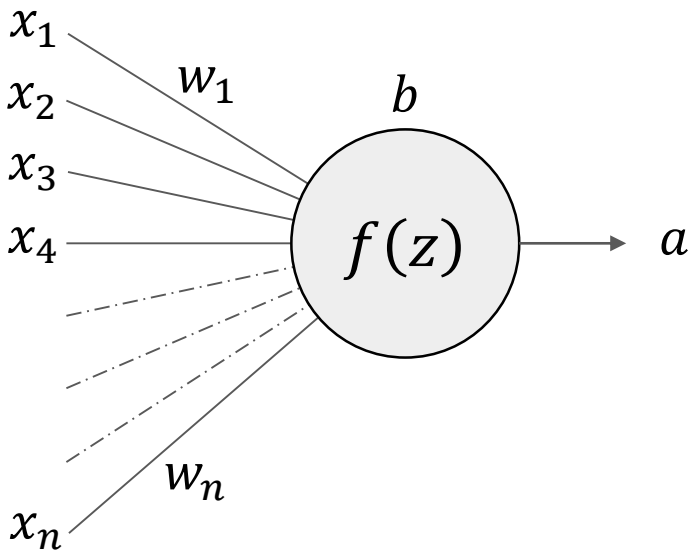
A perceptron is the simplest neural network: a computational model of a single neuron



$$z = \sum_{i=1}^{n} w_i x_i + b$$

Perceptron: ML Elementary Unit

A perceptron is the simplest neural network: a computational model of a single neuron

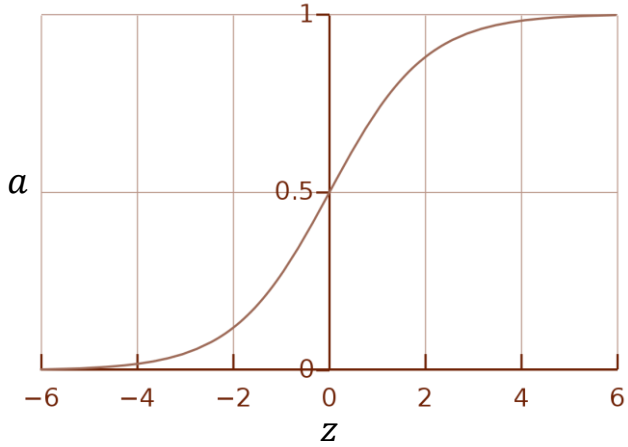


$$z = \sum_{i=1}^{n} w_i x_i + b$$

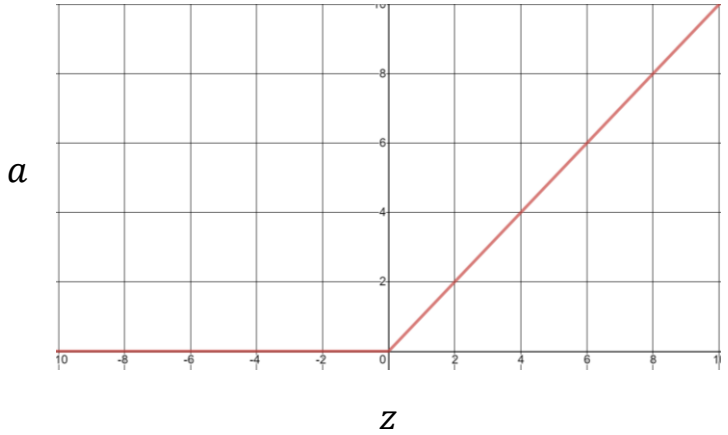
Activation Functions

The activation function allows to have control over the weighted sum.

$$a = \sigma(z) = \frac{1}{1 + e^{-z}}$$

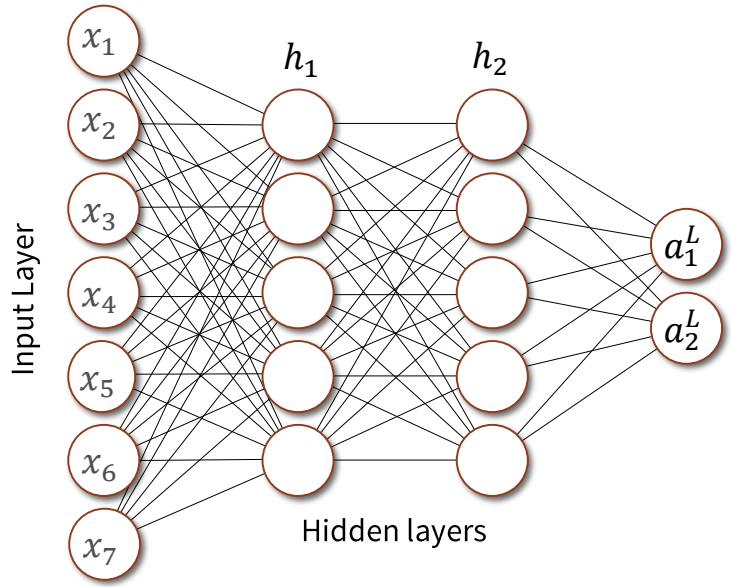


$$a = ReLu(z) = \max(0, z)$$



Artificial Neural Networks

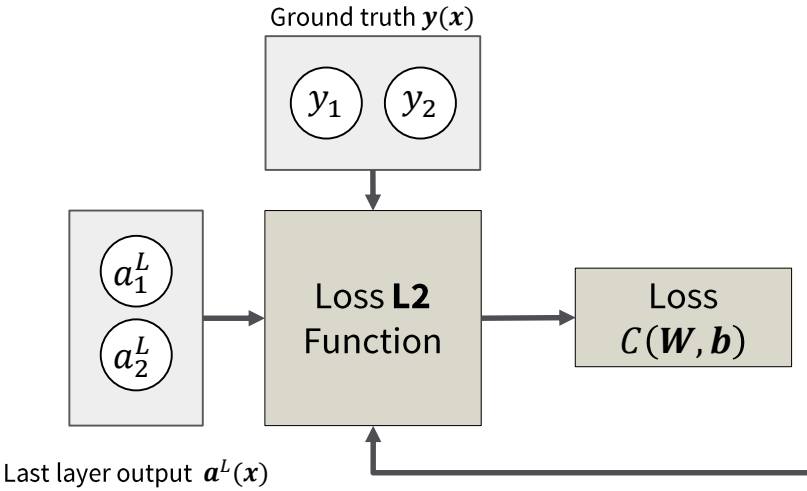
Perceptrons are fully connected to create a network with hidden layers.
Weights and biases are initialized randomly



- $a_j^l = f\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right)$
- $\boldsymbol{a}^l = \boldsymbol{f}\left(\boldsymbol{W}^l \boldsymbol{a}^{l-1} + \boldsymbol{b}^l\right)$

Error Function

The loss function tracks network's performance and it is used to compute gradients for the learning process.



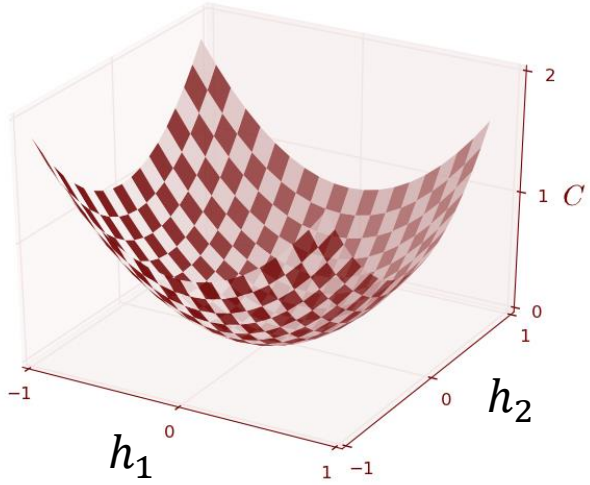
$$C(\boldsymbol{W}, \boldsymbol{b}) = \frac{1}{2n} \sum_{\boldsymbol{x}} \left\| \boldsymbol{y}(\boldsymbol{x}) - \boldsymbol{a}^L(\boldsymbol{x}) \right\|^2$$

Gradient Descent

We aim at training the network to minimize the cost function tuning the weight and biases.

- $\boldsymbol{h} = \{\boldsymbol{W}, \boldsymbol{b}\} \in \mathbb{R}^m$
- $\Delta \mathrm{C} \approx \frac{\partial C}{\partial h_1} dh_1 + \cdots \frac{\partial C}{\partial h_m} dh_m$
- $\Delta \mathrm{C} \approx \nabla C \cdot \Delta \boldsymbol{h}$

- $\Delta \boldsymbol{h} = -\eta \nabla C \implies \Delta \mathrm{C}$
- $\boldsymbol{h}' = \boldsymbol{h} + \Delta \boldsymbol{h} = \boldsymbol{h} - \eta \nabla C$



Setting the **learning rate η** to high would result in excessive overshooting

Backpropagation

We use stochastic gradient descent to minimize a cost function. To update weights and biases (**learning process**) we rely on backpropagation:

Gradients calculation via BP

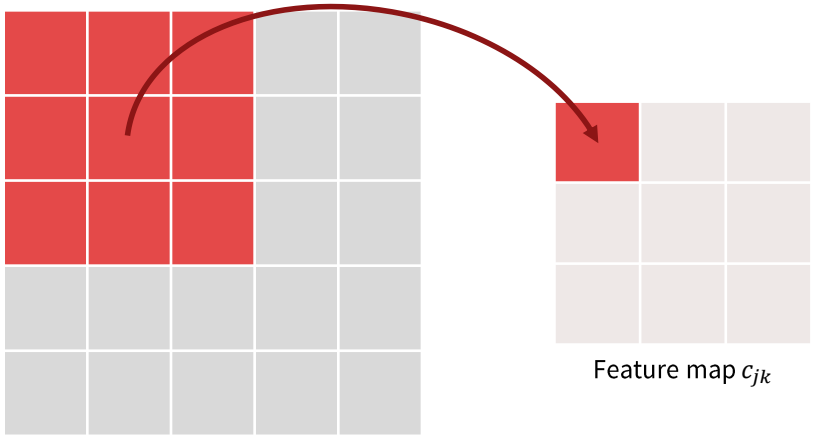
- $\boldsymbol{\delta}^L = \nabla_a C \odot \sigma'(\boldsymbol{z}^L)$
- $\boldsymbol{\delta}^l = \left[\left(w^{l+1}\right)^T \delta^{l+1}\right] \odot \sigma'(\boldsymbol{z}^l)$
- $\frac{\partial C}{\partial b_j^l} = \delta_j^l$
- $\frac{\partial C}{\partial w_{jk}^l} = a_k^l \delta_j^l$

Network parameters update

- $\boldsymbol{h}' = \boldsymbol{h} + \Delta\boldsymbol{h} = \boldsymbol{h} - \eta \nabla C$
- $w_k' = w - \eta \frac{\partial C}{\partial w_k}$
- $b_k' = b - \eta \frac{\partial C}{\partial b_k}\ a_k^l \delta_j^l$

Convolutional Neural Networks

CNNs are the standard in image processing for artificial intelligence. The output of CNNs layers generalizes the most common patterns highlighting key aspects



Input Image x_{jk}

Feature map c_{jk}

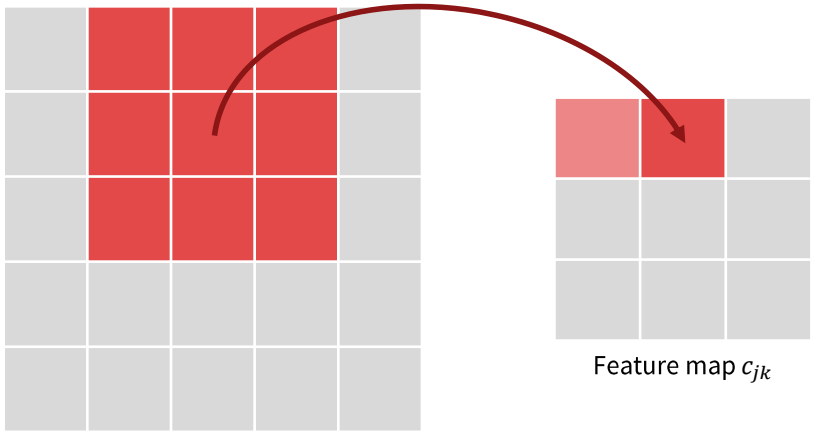
$$c_{jk} = \sigma\left(\sum_{l}\sum_{m} w_{l,m} x_{j+l,k+m} + b\right)$$

Convolution with shared weights

Input: 4× 4
Kernel: 3×3 $(l \times m)$
Stride: 1

Convolutional Neural Networks

CNNs are the standard in image processing for artificial intelligence. The output of CNNs layers generalizes the most common patterns highlighting key aspects



Input Image x_{jk}

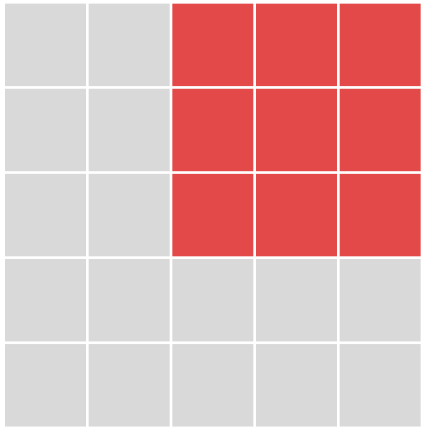
Feature map c_{jk}

$$a_{jk} = \sigma\left(\sum_{l}\sum_{m} w_{l,m} a_{j+l,k+m} + b\right)$$

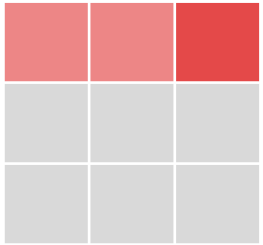
Convolution with shared weights

Convolutional Neural Networks

CNNs are the standard in image processing for artificial intelligence. The output of CNNs layers generalizes the most common patterns highlighting key aspects



Input Image x_{jk}



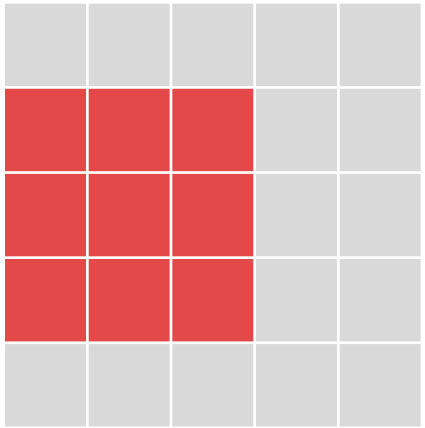
Feature map c_{jk}

$$a_{jk} = \sigma\left(\sum_{l}\sum_{m} w_{l,m} a_{j+l,k+m} + b\right)$$

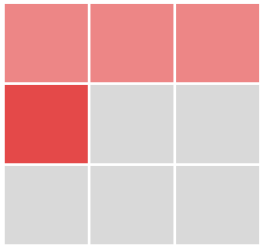
Convolution with shared weights

Convolutional Neural Networks

CNNs are the standard in image processing for artificial intelligence. The output of CNNs layers generalizes the most common patterns highlighting key aspects



Input Image x_{jk}



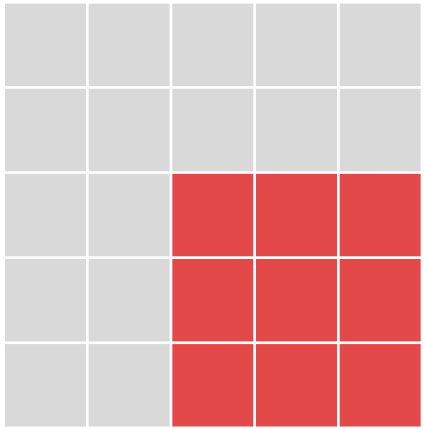
Feature map c_{jk}

$$a_{jk} = \sigma\left(\sum_{l}\sum_{m} w_{l,m} a_{j+l,k+m} + b\right)$$

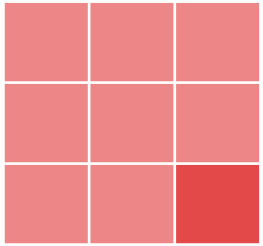
Convolution with shared weights

Convolutional Neural Networks

CNNs are the standard in image processing for artificial intelligence. The output of CNNs layers generalizes the most common patterns highlighting key aspects



Input Image x_{jk}



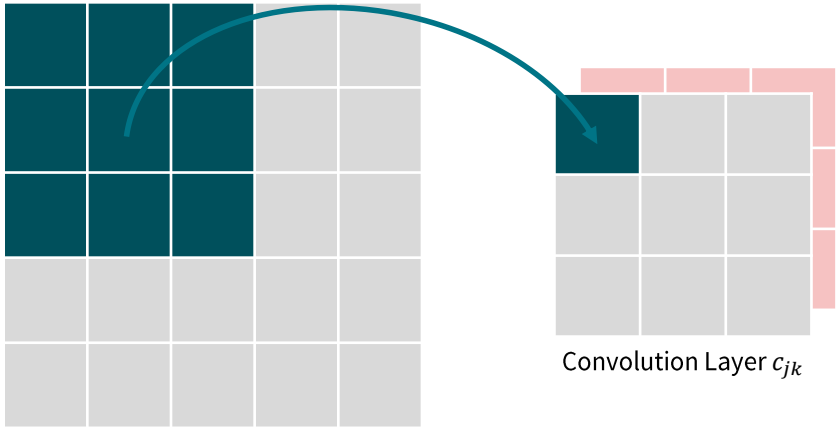
Feature map c_{jk}

$$a_{jk} = \sigma\left(\sum_{l}\sum_{m} w_{l,m} a_{j+l,k+m} + b\right)$$

Convolution with shared weights

Convolutional Neural Networks

CNNs are the standard in image processing for artificial intelligence. The output of CNNs layers generalizes the most common patterns highlighting key aspects



Input Image x_{jk}

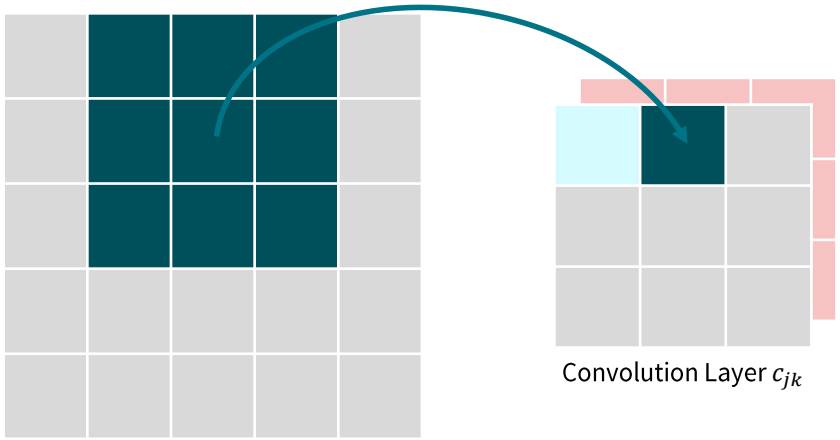
Convolution Layer c_{jk}

$$a_{jk} = \sigma\left(\sum_{l}\sum_{m} w_{l,m} a_{j+l,k+m} + b\right)$$

Convolution with shared weights

Convolutional Neural Networks

CNNs are the standard in image processing for artificial intelligence. The output of CNNs layers generalizes the most common patterns highlighting key aspects



Input Image x_{jk}

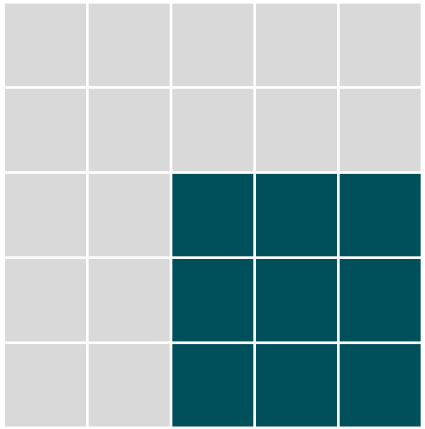
Convolution Layer c_{jk}

$$a_{jk} = \sigma\left(\sum_{l}\sum_{m} w_{l,m} a_{j+l,k+m} + b\right)$$

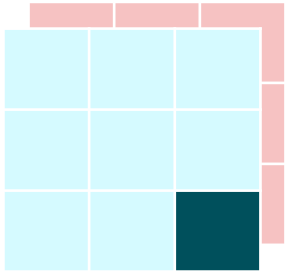
Convolution with shared weights

Convolutional Neural Networks

CNNs are the standard in image processing for artificial intelligence. The output of CNNs layers generalizes the most common patterns highlighting key aspects



Input Image x_{jk}



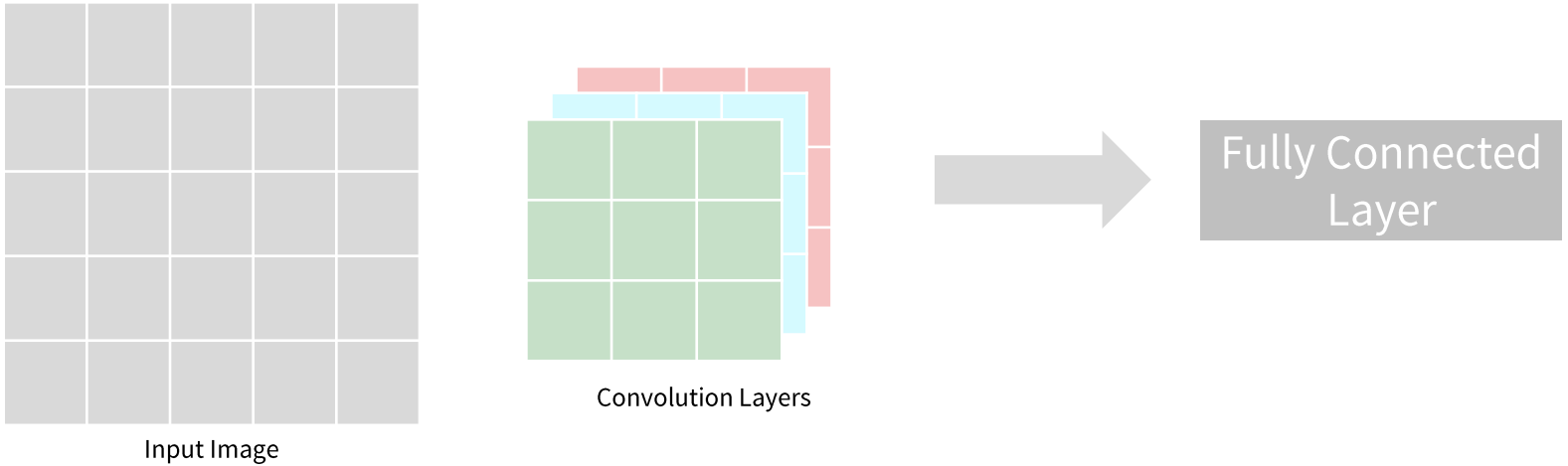
Convolution Layer c_{jk}

$$a_{jk} = \sigma\left(\sum_{l}\sum_{m} w_{l,m} a_{j+l,k+m} + b\right)$$

Convolution with shared weights

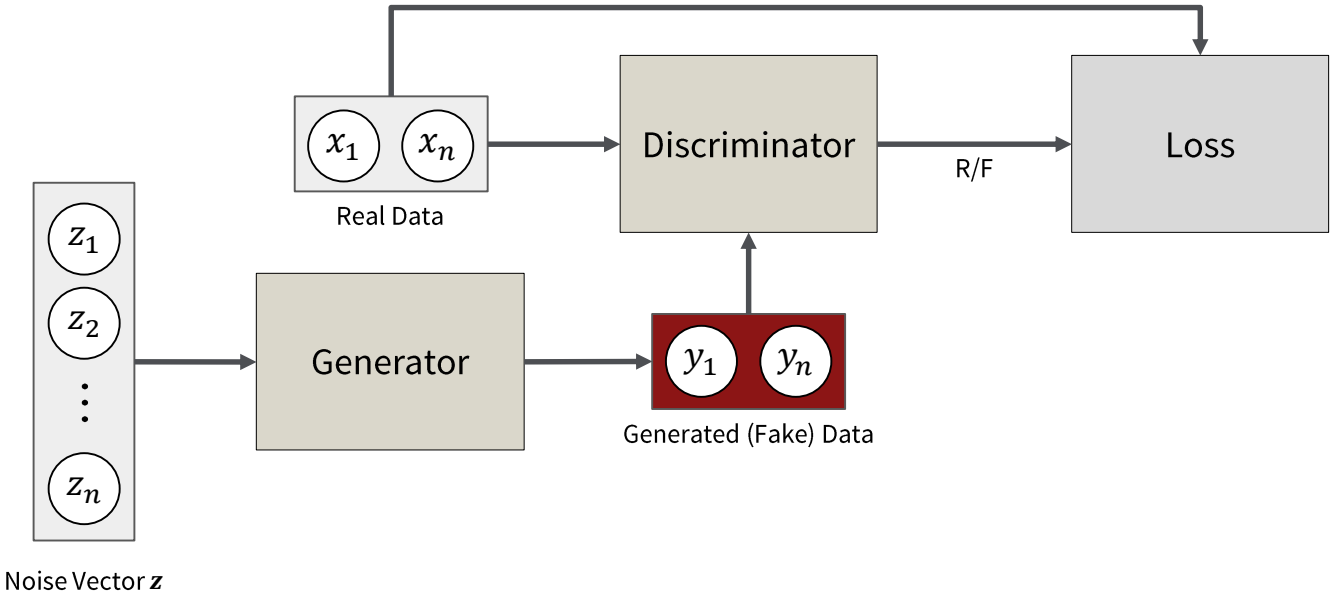
Convolutional Neural Networks

CNNs are the standard in image processing for artificial intelligence. The output of CNNs layers generalizes the most common patterns highlighting key aspects



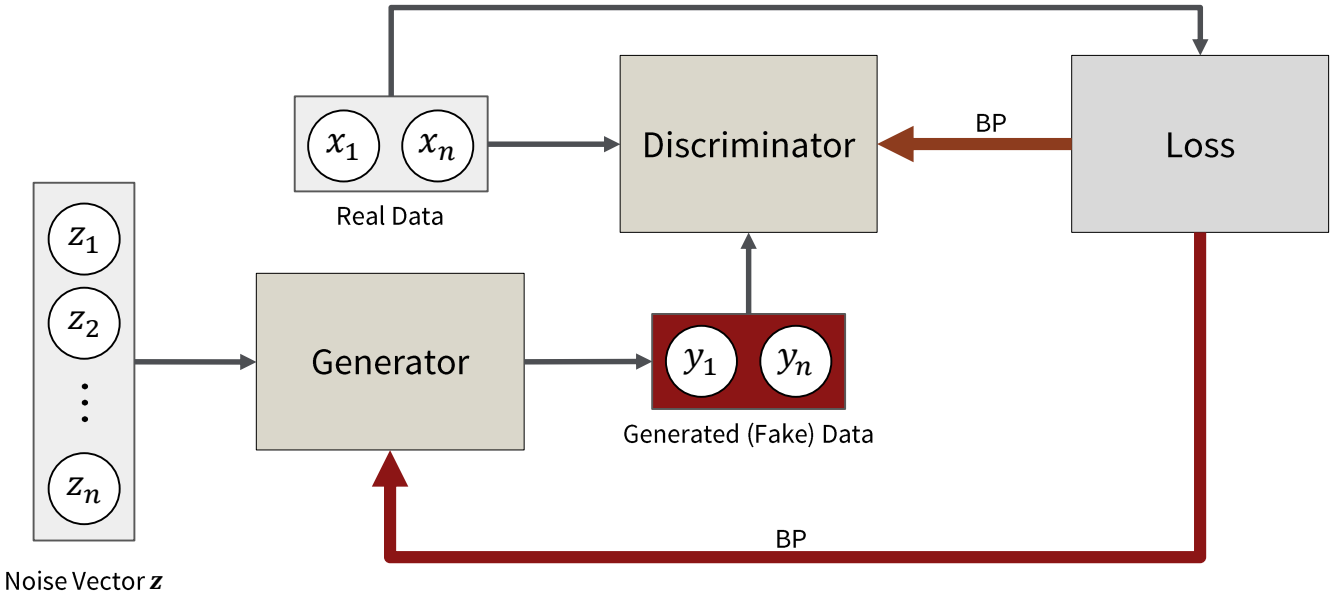
Generative Antagonist Networks

The network is trained until the discriminator cannot distinguish real data from fake data.



Generative Antagonist Networks: Architecture

The network is trained until the discriminator cannot distinguish real data from fake data.



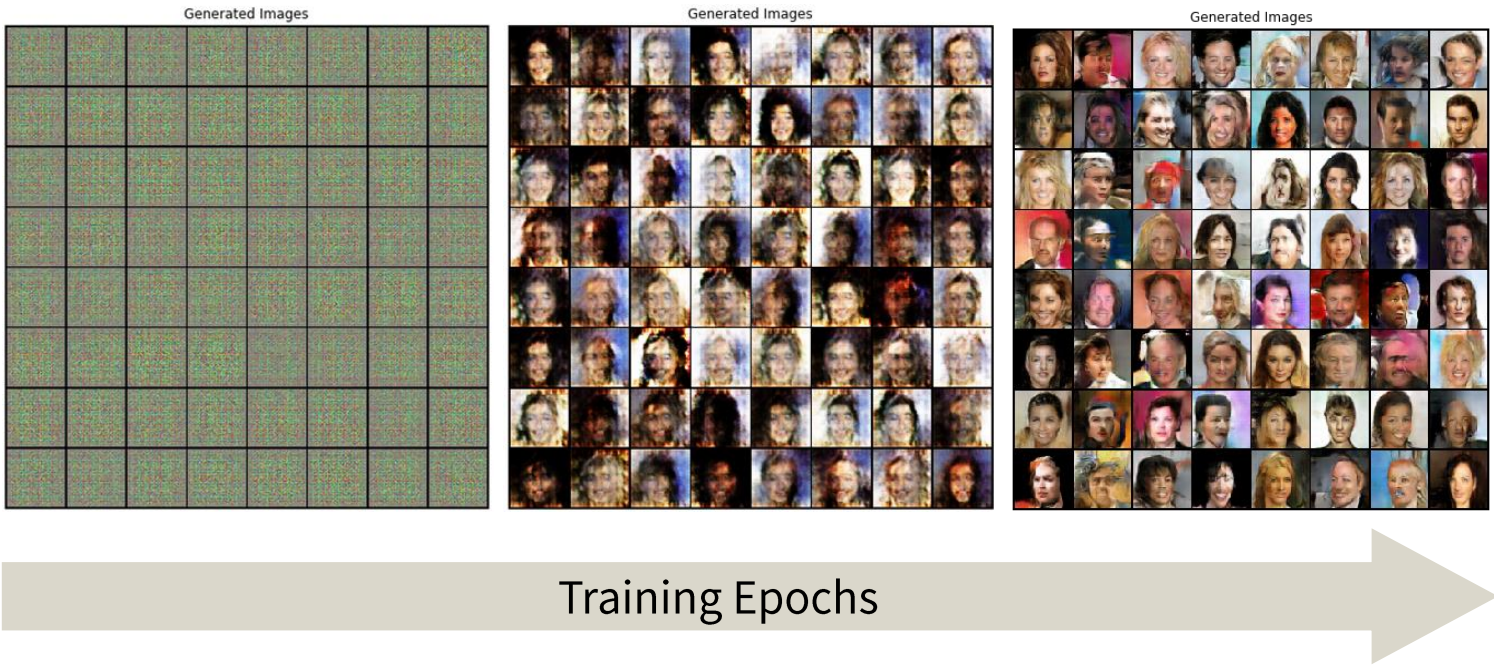
Generative Antagonist Networks: Problem Space

Giving some colors to a child hoping he will draw Caravaggio's "I Bari" fooling an art critic

Michelangelo Merisi, 1594

GAN Results seem promising

A first experiment shows remarkable results after only 5 training epochs



Code Example: Generator

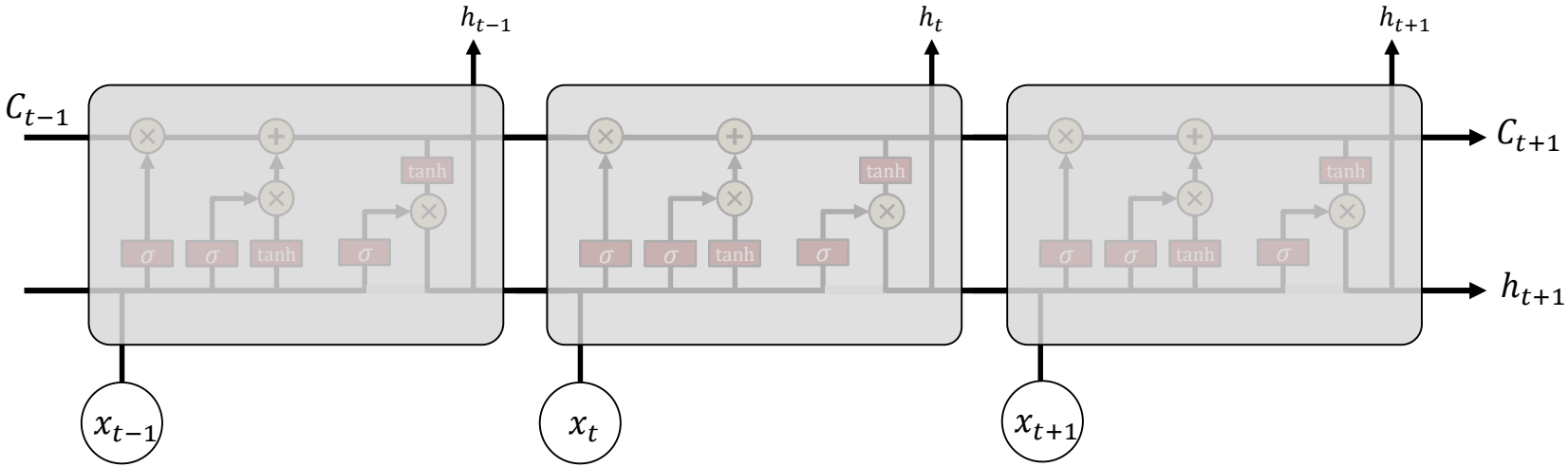
```
class Generator(nn.Module):
    def __init__(self, ngpu):

        self.main = nn.Sequential(
            nn.ConvTranspose2d(in_channels = nz, out_channels = ngf * 8, kernel_size = 4,
                               stride = 1, padding = 0, bias=False),
            nn.BatchNorm2d(num_features = ngf * 8),
            nn.ReLU(True),

            nn.ConvTranspose2d(ngf * 8, ngf * 4, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf * 4),
            nn.ReLU(True),

            nn.ConvTranspose2d( ngf * 4, ngf * 2, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf * 2),
            nn.ReLU(True),

            nn.ConvTranspose2d(ngf * 2, ngf, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf),
            nn.ReLU(True),

            nn.ConvTranspose2d(ngf, nc, 4, 2, 1, bias=False),
            nn.Tanh()
```

Code Example: Discriminator

```
class Discriminator(nn.Module):
    def __init__(self, ngpu):

        self.main = nn.Sequential(

            nn.Conv2d(nc, ndf, 4, 2, 1, bias=False),
            nn.LeakyReLU(0.2, inplace=True),

            nn.Conv2d(ndf, ndf * 2, 4, 2, 1,      False),
            nn.BatchNorm2d(ndf * 2),
            nn.LeakyReLU(0.2, inplace=True),

            nn.Conv2d(ndf * 2, ndf * 4, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ndf * 4),
            nn.LeakyReLU(0.2, inplace=True),

            nn.Conv2d(ndf * 4, ndf * 8, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ndf * 8),
            nn.LeakyReLU(0.2, inplace=True),

            nn.Conv2d(ndf * 8, 1, 4, 1, 0, bias=False),
            nn.Sigmoid()
```

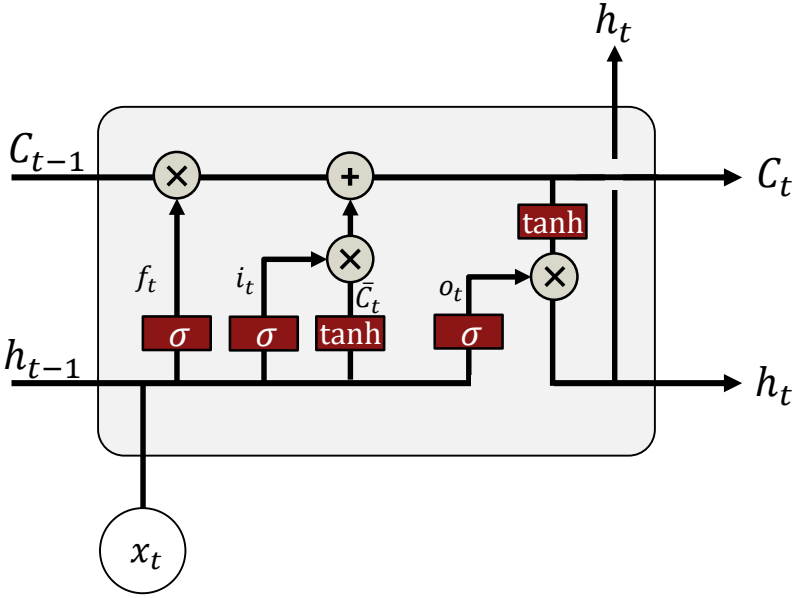
Long-Short Term Memory (LSTM) Networks

Humans don't start their thinking from scratch every second. Recurrent Neural Networks (RNN) take account of history to understand the context



Long-Short Term Memory (LSTM) Networks

We analyze a single block of an LSTM network

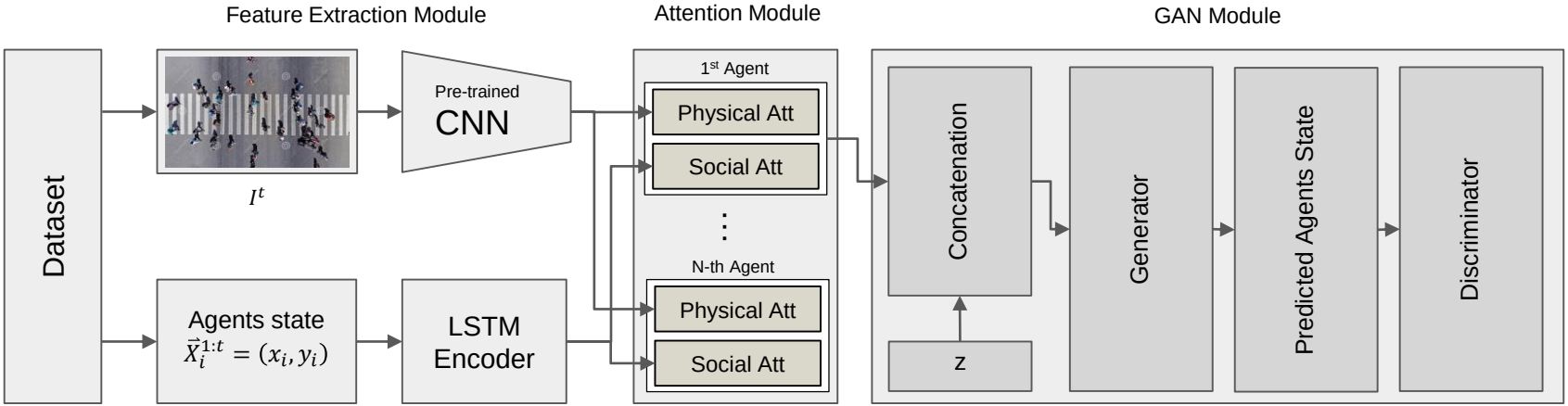


- x_t = Input Vector
- C_t = Context Vector
- $f_t = \sigma\left(W_f \cdot [x_t, h_{t-1}] + b_f\right)$
- $i_t = \sigma(W_i \cdot [x_t, h_{t-1}] + b_i)$
- $\bar{C}_t = \tanh(W_C \cdot [x_t, h_{t-1}] + b_C)$
- $o_t = \sigma(W_i \cdot [x_t, h_{t-1}] + b_i)$

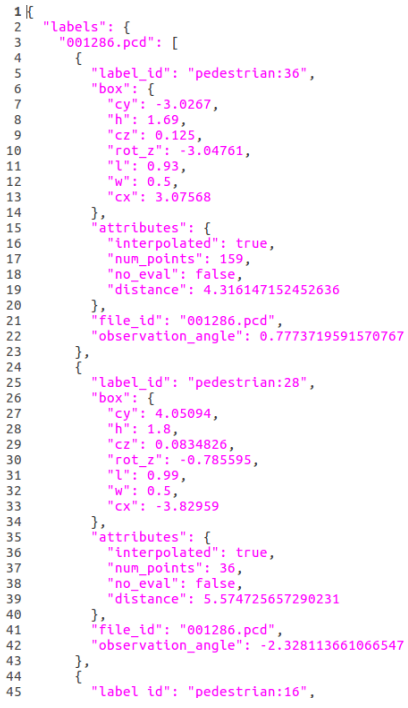
- $C_t = f_t \cdot C_{t-1} + i_t \cdot \bar{C}_t$
- $h_t = o_t \cdot \tanh(C_t)$

SoPhie: Top-View Approach

SoPhie algorithm considers both physical and social constraints using as input the top view scene and coordinates of each pedestrian



JRDB Converter



```
{
  "labels": {
    "001286.pcd": [
      {
        "label_id": "pedestrian:36",
        "box": {
          "cy": -3.0267,
          "h": 1.69,
          "cz": 0.125,
          "rot_z": -3.04761,
          "l": 0.93,
          "w": 0.5,
          "cx": 3.07568
        },
        "attributes": {
          "interpolated": true,
          "num_points": 159,
          "no_eval": false,
          "distance": 4.316147152452636
        },
        "file_id": "001286.pcd",
        "observation_angle": 0.7773719591570767
      },
      {
        "label_id": "pedestrian:28",
        "box": {
          "cy": 4.05094,
          "h": 1.8,
          "cz": 0.0834826,
          "rot_z": -0.785595,
          "l": 0.99,
          "w": 0.5,
          "cx": -3.82959
        },
        "attributes": {
          "interpolated": true,
          "num_points": 36,
          "no_eval": false,
          "distance": 5.574725657290231
        },
        "file_id": "001286.pcd",
        "observation_angle": -2.328113661066547
      },
      {
        "label id": "pedestrian:16",
```

.json

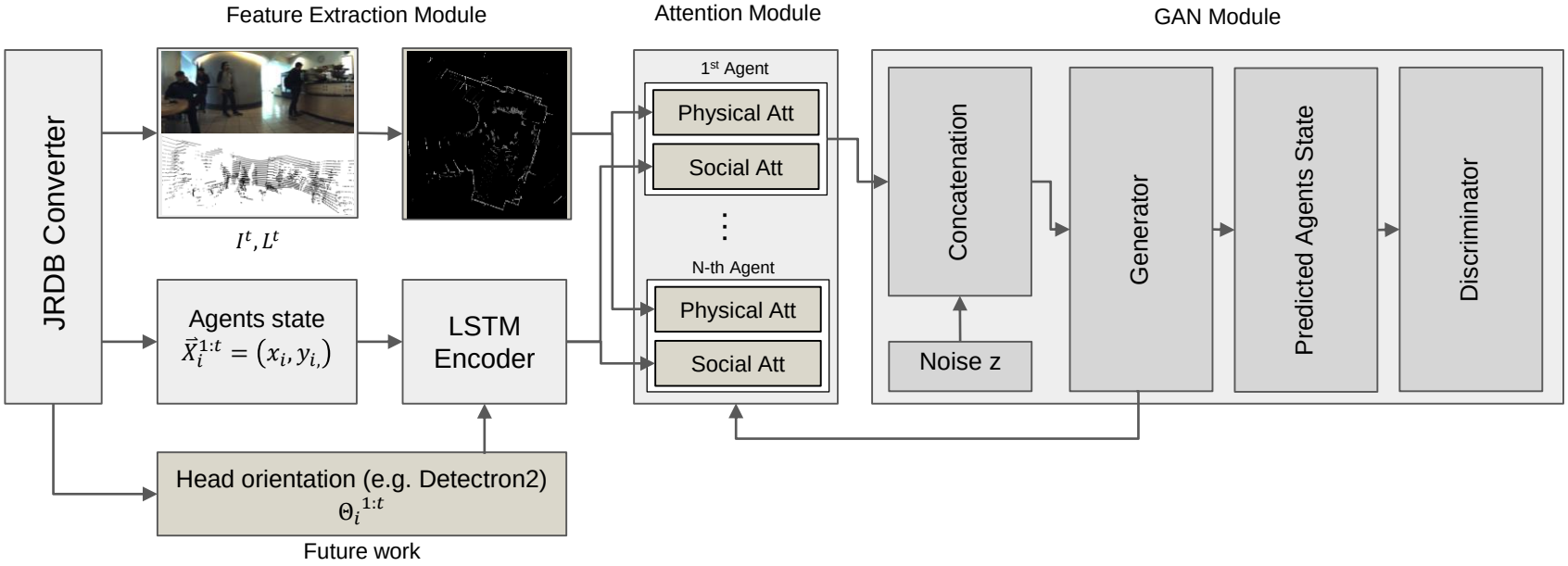

```
0.0 1.0 0.4 2.85
0.0 3.0 -0.56 1.7
0.0 5.0 -1.27 1.63
0.0 6.0 -0.55 -1.16
0.0 7.0 -1.51 -0.7
0.0 8.0 0.98 -1.8
0.0 9.0 -7.72 1.96
0.0 10.0 -7.68 3.4
0.0 12.0 -6.36 0.53
0.0 14.0 -4.2 0.96
0.0 17.0 -7.33 0.12
0.0 20.0 -4.84 -0.29
0.0 21.0 -2.0 0.24
0.0 22.0 -0.43 3.42
0.0 23.0 -0.04 -3.11
0.0 24.0 1.89 1.84
0.0 25.0 -3.55 -0.08
0.0 26.0 -1.48 7.88
0.0 27.0 -3.17 -1.14
0.0 35.0 3.1 1.52
1.0 1.0 0.4 2.85
1.0 3.0 -0.56 1.7
1.0 5.0 -1.27 1.62
1.0 6.0 -0.55 -1.16
1.0 7.0 -1.51 -0.7
1.0 8.0 0.98 -1.8
1.0 9.0 -7.72 1.96
1.0 10.0 -7.68 3.4
1.0 12.0 -6.37 0.54
1.0 14.0 -4.2 0.95
1.0 17.0 -7.33 0.12
1.0 20.0 -4.84 -0.29
1.0 21.0 -1.99 0.29
1.0 22.0 -0.45 3.42
1.0 23.0 -0.04 -3.1
1.0 24.0 1.9 1.85
1.0 25.0 -3.55 -0.01
1.0 26.0 -1.48 7.88
1.0 27.0 -3.16 -1.12
1.0 35.0 3.1 1.51
2.0 1.0 0.4 2.85
2.0 3.0 -0.56 1.69
2.0 5.0 -1.27 1.62
2.0 6.0 -0.55 -1.16
2.0 7.0 -1.51 -0.7
```

.txt

GitHub: https://github.com/GiulioAutel/jrdb_converter

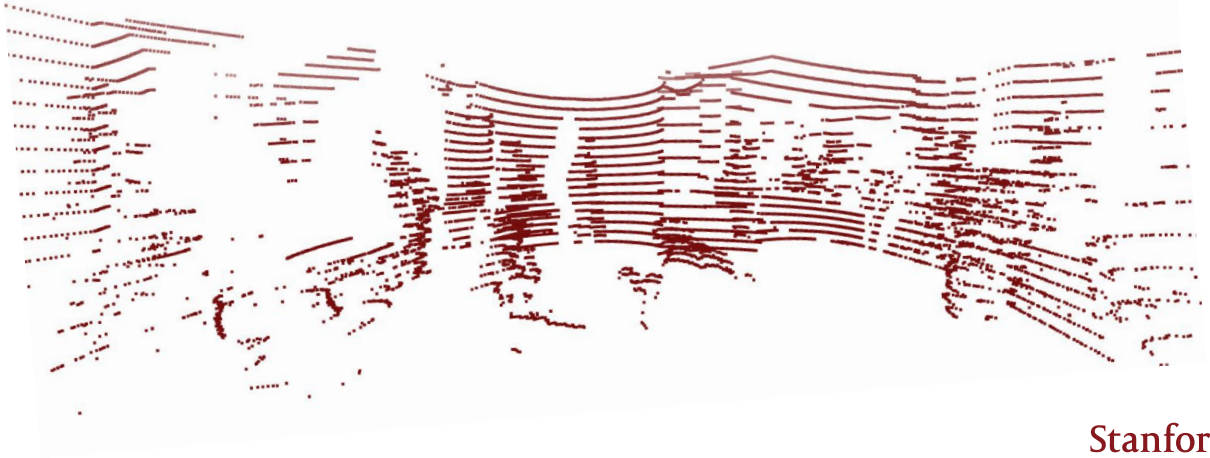
SoPhie: Extension to first-person view

Utilizing onboard sensors to forecast pedestrian interaction is crucial for social navigation. Primary sensors are RGB cameras and LiDAR.



JRDB Dataset

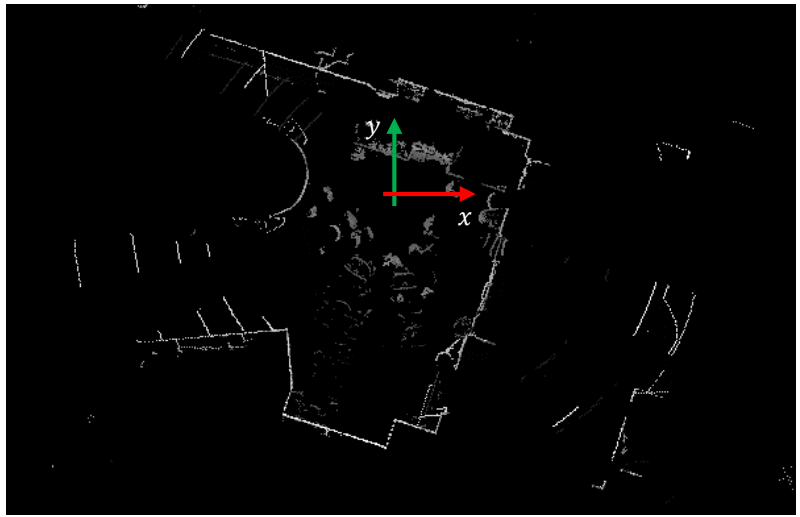
The dataset is composed of both indoor and outdoor scenarios. Interactions between pedestrians span from trivial to highly unpredictable



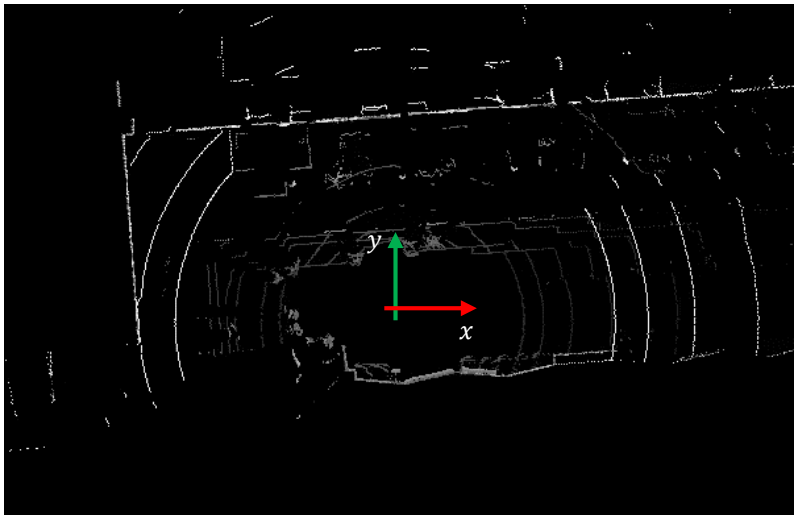
Birdseye View Transformation

Using **LiDAR point-cloud** it is possible to infer the **birds-eye view** scene. In this manner occlusion problems are highly mitigated.

Cafeteria: people ordering and eating

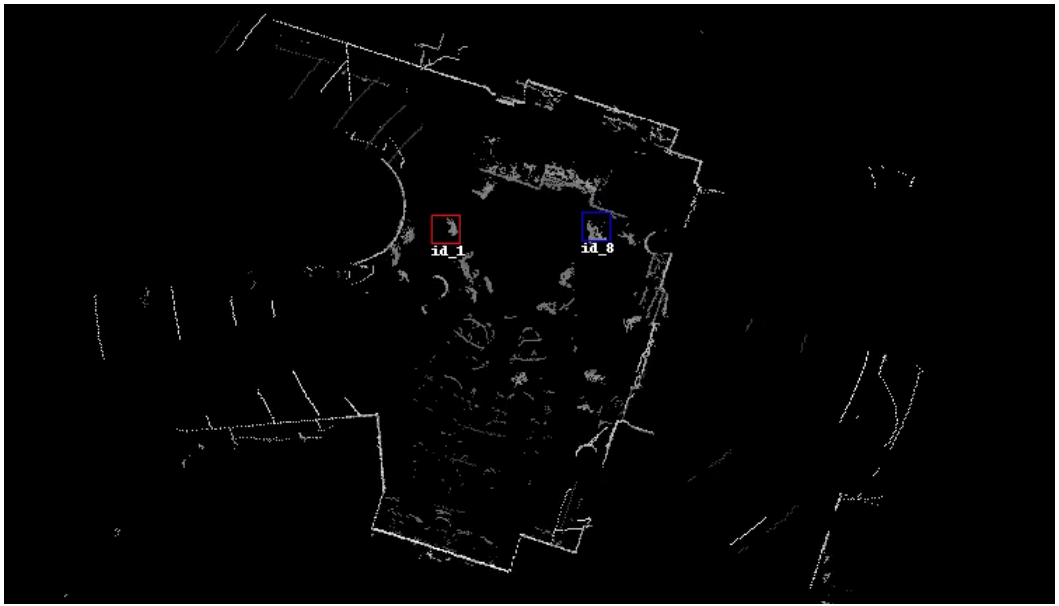


Hall: students leaving class



Pedestrian Tracking

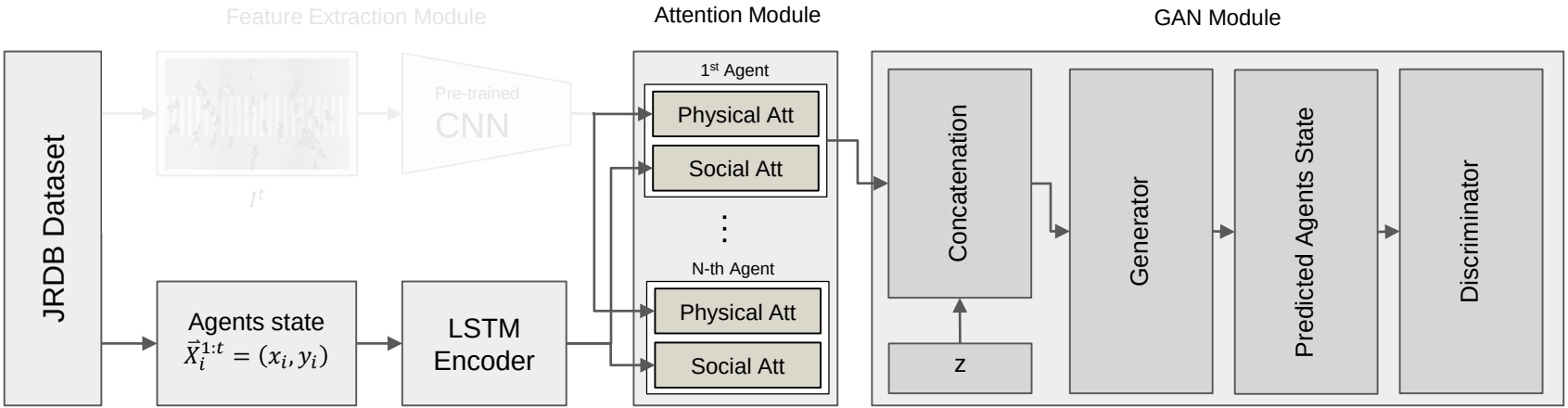
Applying a transformation to labels position it is possible to track pedestrian movements and thus feeding the input to the neural network.



GitHub: https://github.com/GiulioAutel/jrdb_converter

S-GAN: Baseline Evaluation

As a first step I trained the network with a cleaner and richer dataset coming from the JRDB Converter

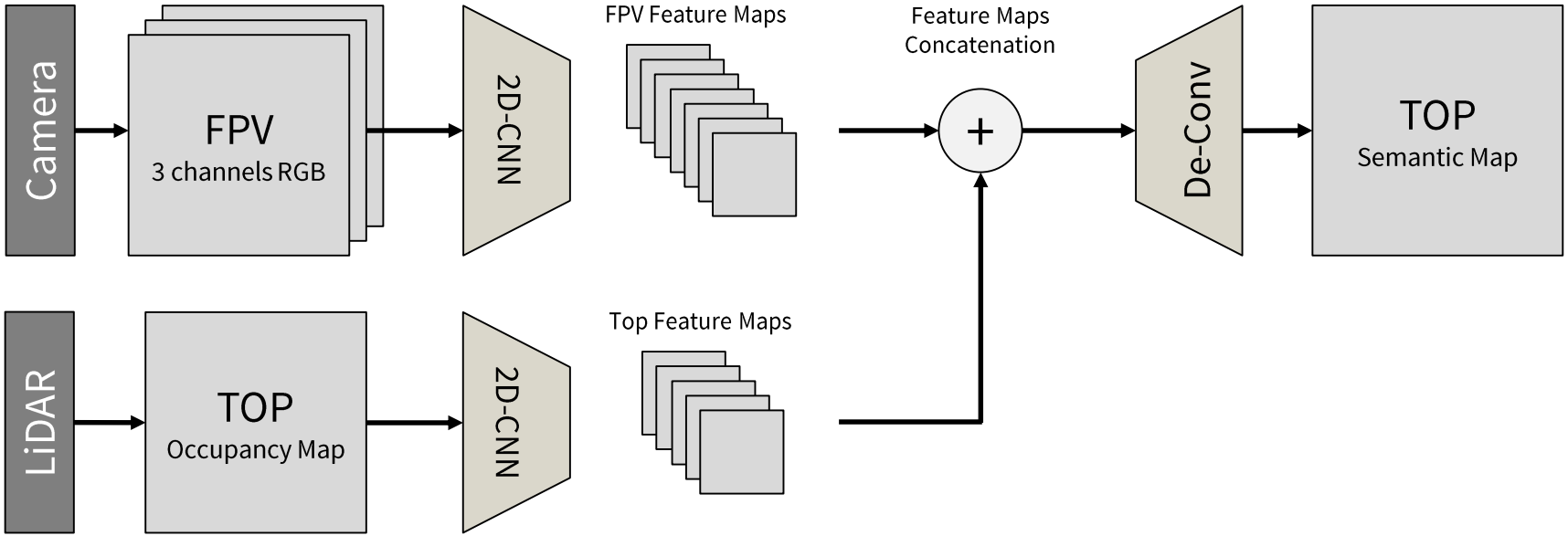


Baseline: Social-GAN Preliminary Results on JRDB

		Model (ADE/FDE [m])			
		JRDB-In	JRDB-Out	ETH (Worst)	ZARA2 (Best)
Dataset	JRDB-In	**0.08 / 0.18**	**0.08 / 0.18**	0.14 / 0.28	0.12 / 0.24
	JRDB-Out	**0.07 / 0.15**	**0.06 / 0.13**	0.14 / 0.29	0.12 / 0.24
	ETH	1.66 / 2.85	1.02 / 1.82	0.72 / 1.31	0.52 / 0.94
	ZARA2	0.64 / 1.28	0.37 / 0.78	0.32 / 0.67	0.31 / 0.64

ADE: Average Displacement Error
FDE: Finale Displacement Error

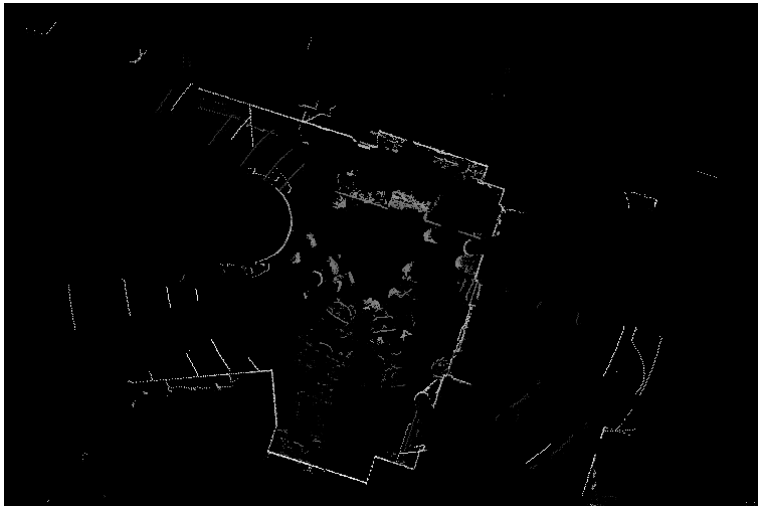
Top View Net Architecture



Top View Net

The network's input will be the RGB video stream and the LiDAR pointcloud projected on a 2D plane.

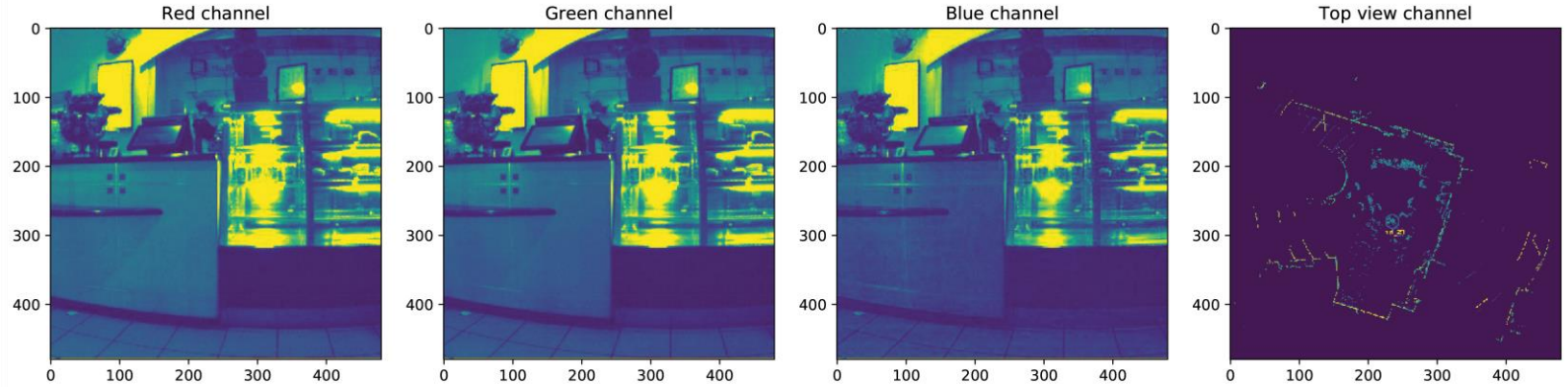
RGB Camera Video Stream



LiDAR projected pointcloud

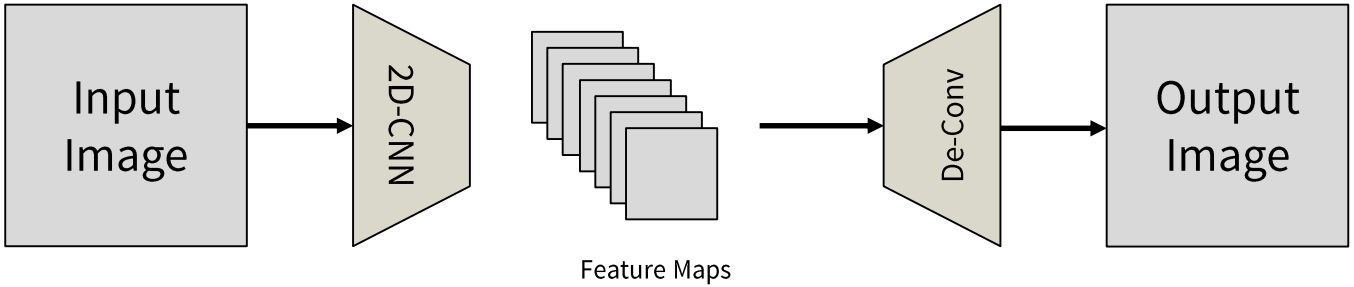
Input Data Loader

To generate the input and shuffle the dataset, a data-loader has been coded. All the inputs are tensors storing pixel intensity values bounded between [-1, 1] with 255 steps (8-bit). The images are also cropped.



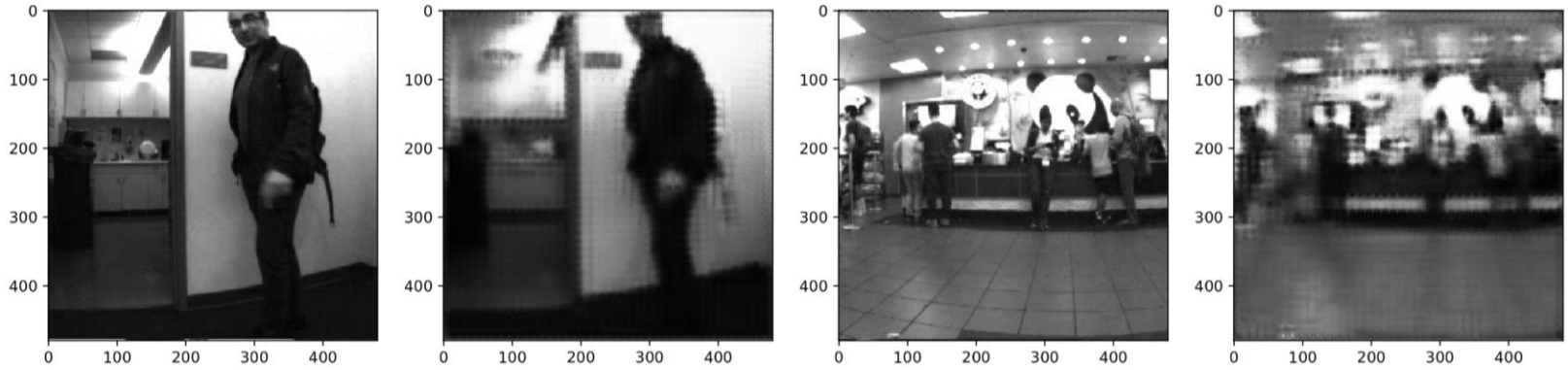
Autoencoder Test

To test the network architecture the encoder-decoder module has been trained as an autoencoder to obtain a visual proof as a sanity check



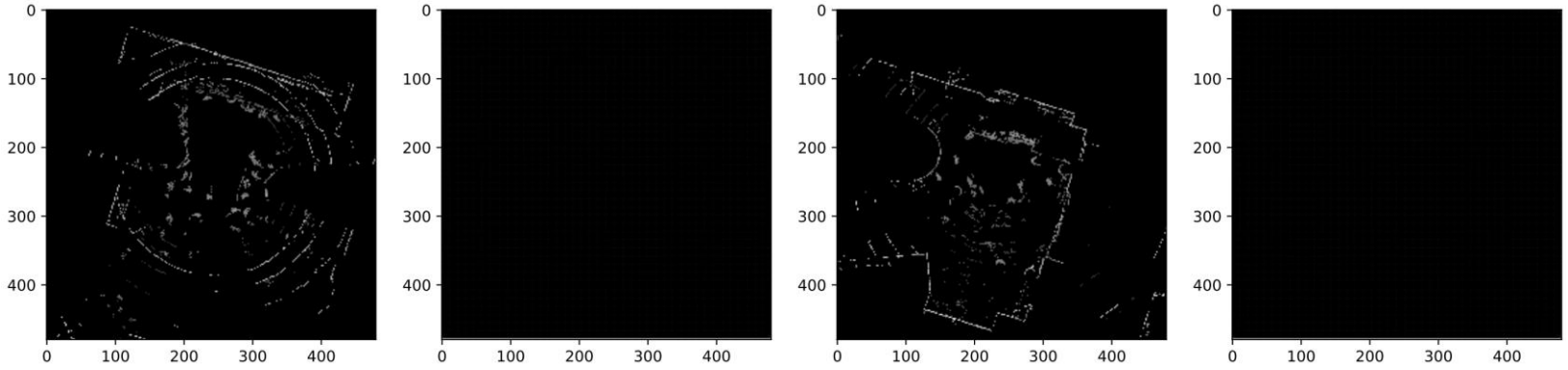
Autoencoder Results on FPV

The autoencoder module can extract the most relevant features and reconstruct a final image based on the latter information. The overall result is satisfactory as a sanity check.



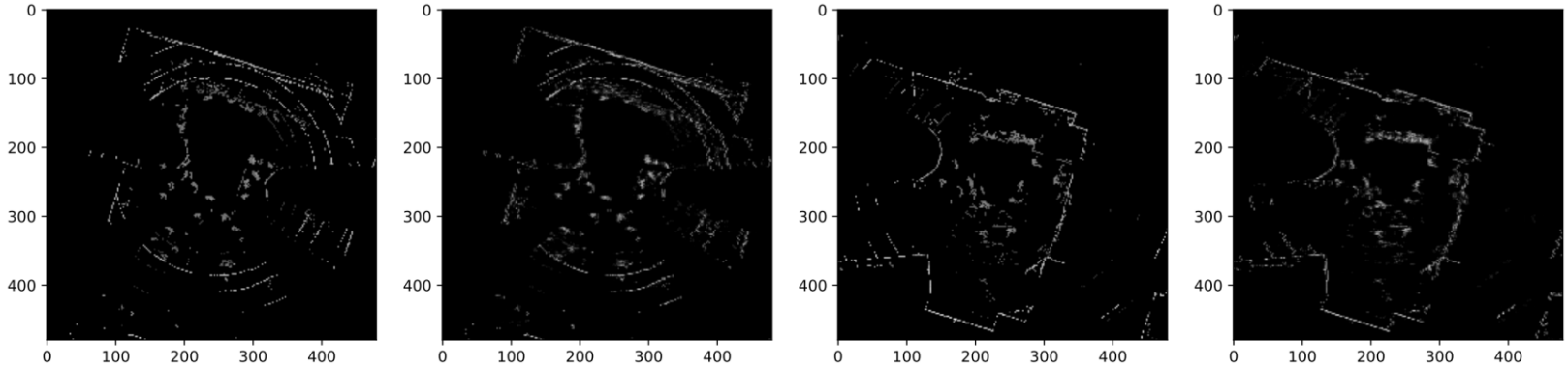
Autoencoder Results on LiDAR Top View (2D-Conv)

The autoencoder module is not capable of extracting the most relevant features and reconstruct a final image based on LiDAR information. The 2D convolution layers have not proven to be the correct solution.



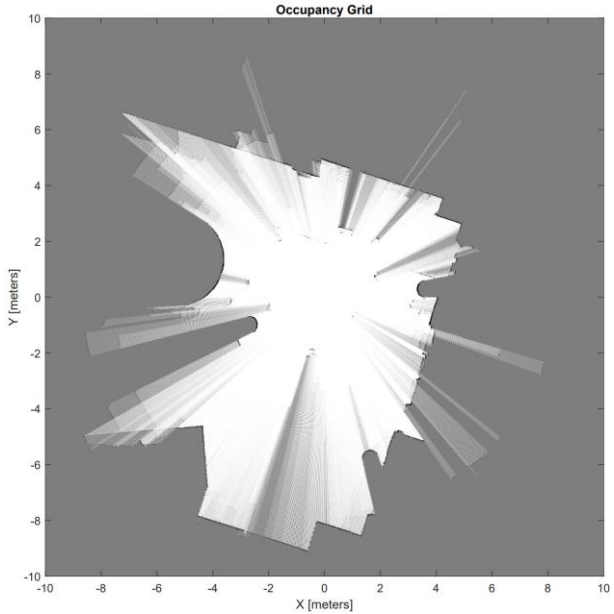
Autoencoder Results on LiDAR Top View (1D-Conv)

The 2D input image tensor ($n \times n$) has been reshaped to obtain a single column vector of sizes: $n^2 \times 1$. A subsequent 1D-Conv series of layers have been utilized to extract the most relevant features.



Occupancy Grid Map

The basic concept behind an occupancy grid map is the representation of three different domains: free space, solid boundaries and unknown space.

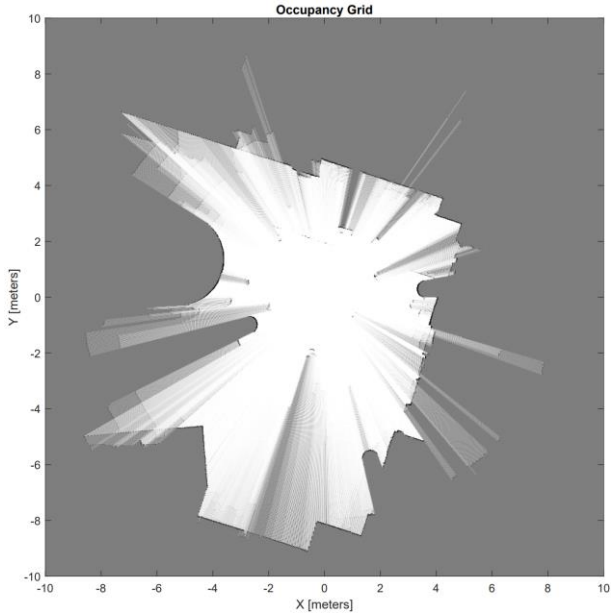


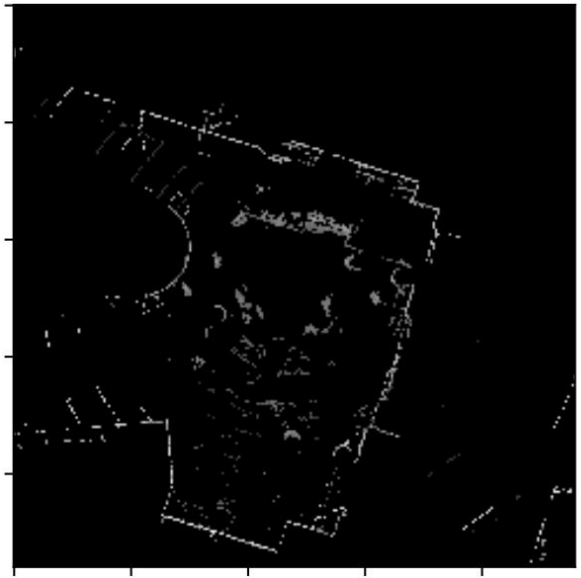
- $\vec{r_i}(t) = (x_i, y_y, z_i)$
- $P_t = \{\vec{r_i}(t)\}_{i=1}^{n}$
- $n =$ Number of LiDAR rays

- $\overline{r_i} < \vec{r_i}$ Free Space (White)
- $\overline{r_i} = \vec{r_i}$ Solid Boundary (Black)
- $\overline{r_i} > \vec{r_i}$ Unknown Space (Gray

Occupancy Grid Map

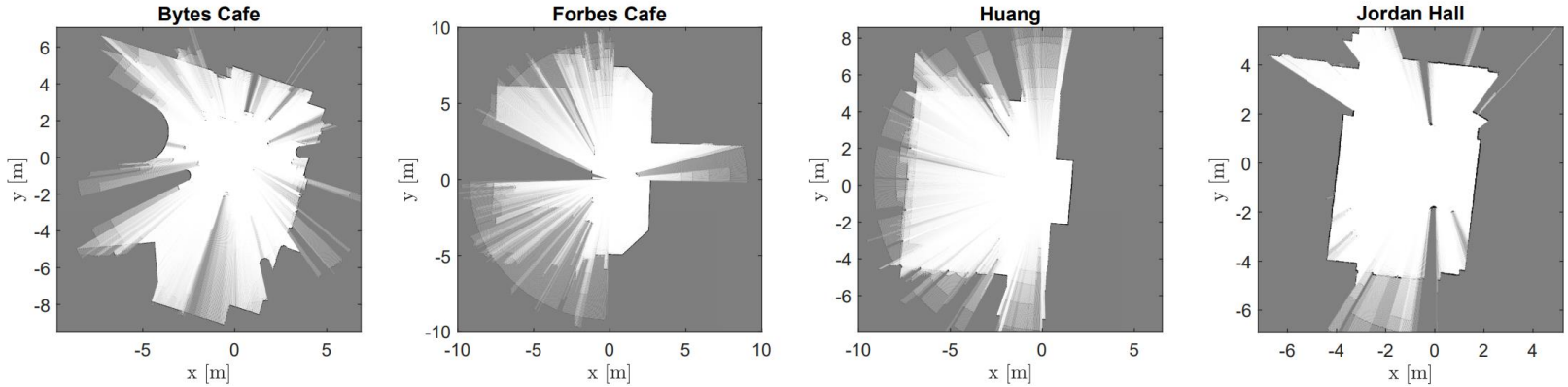
The basic concept behind an occupancy grid map is the representation of three different domains: the free space, the solid boundaries and the unknown space.





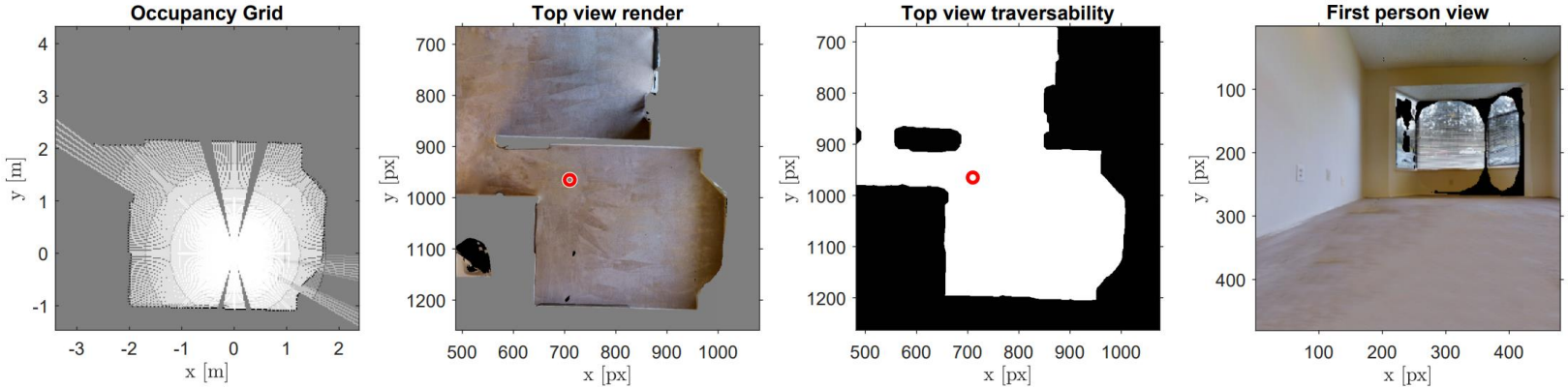
Occupancy Map Examples

Other examples of occupancy maps relative to different environments. For each map it is mandatory to identify a “zone of interest”.

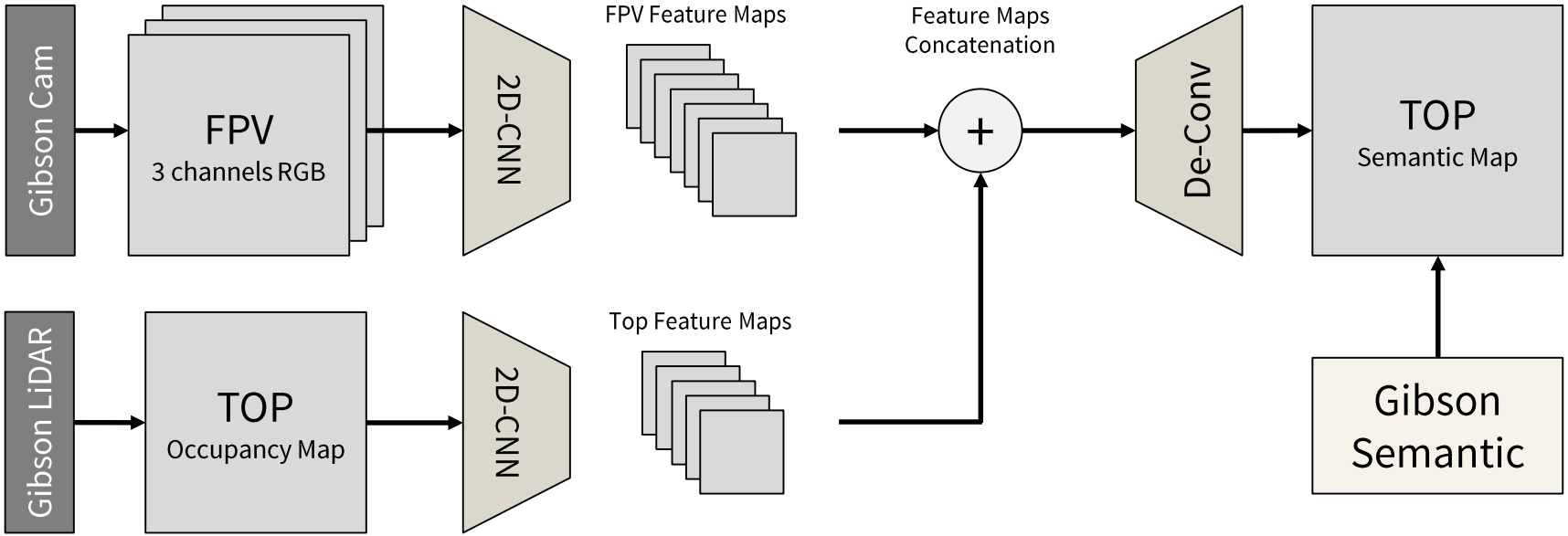


Gibson Environment Dataset: Proof of Concept

The final step before the training process is the creation of a dataset to obtain the ground-truth for the loss function evaluation. A Gibson Environment has been modified to output the desired results.



Final Training



Pedestrian Trajectory Forecasting

MIDTERM WORK REPORT

Stanford University, Vision and Learning Laboratory
Giulio Autelitano, 6 Jan 2020

Stanford | ENGINEERING

ASI Agenzia Spaziale Italiana

Cultural Association of Italians at Fermilab