Self drives me crazy: from 0 to autonomous car in 150 hours

Felipe Salvatore

Paula Moraes

Quem somos:

- **Felipe Salvatore**: doutorando em Deep Learning/NLP
- **Paula Moraes**: mestranda com foco em Robótica Probabilística

Laboratório de IA do IME-USP

Por que montar um carro autônomo?

- Desmistificar a implementação dessa tecnologia
- Porque é divertido :)

Desafio

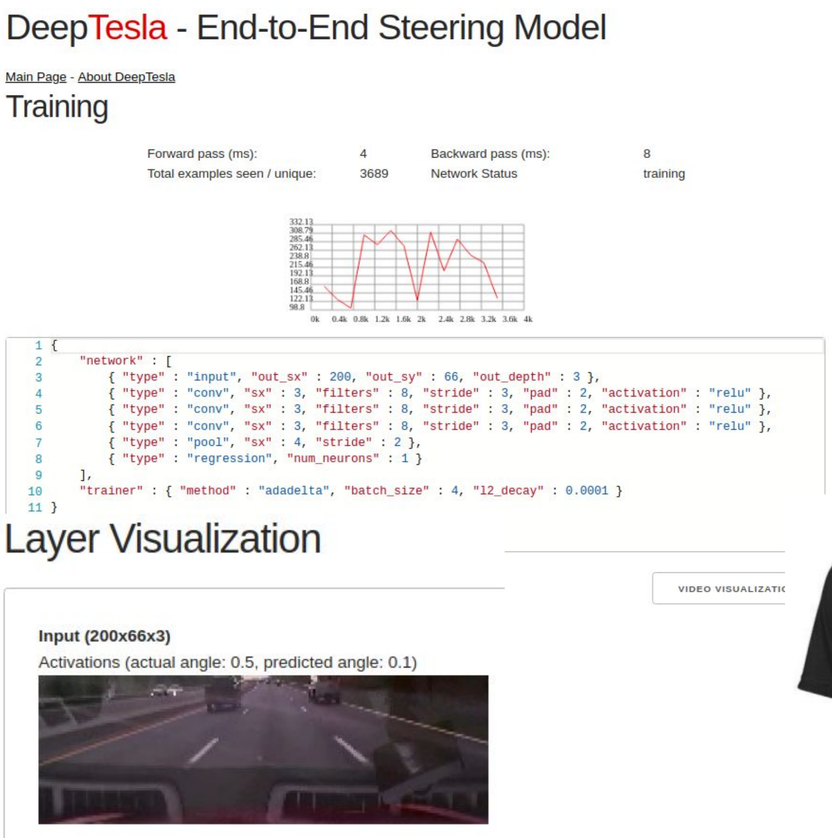
Implementar o artigo: End to End Learning for Self-Driving Cars (2016)

Por que carros autônomos?

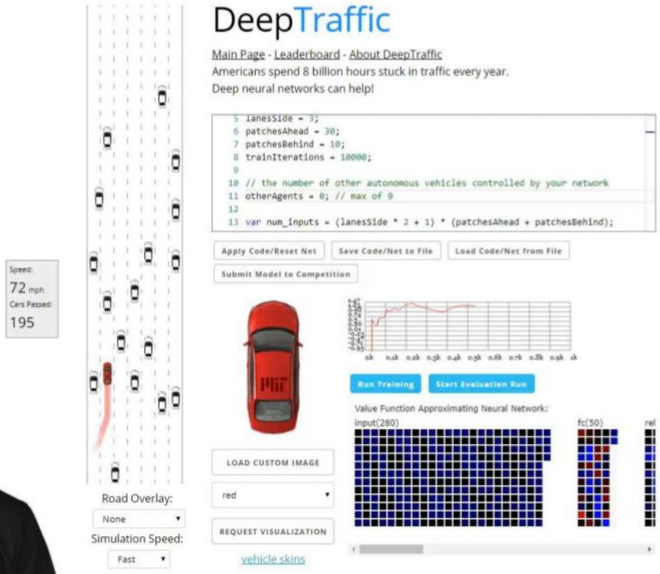
- Carros autônomos são os robôs que vão diretamente impactar nosso **dia a dia no futuro próximo**: atualmente há 51,3 milhões de veículos rodando no Brasil.

- Esse problema é rico para a inteligência artificial, pois junta diferentes áreas como **machine learning, computer vision, path planning** e **reinforcement learning**

Carro autônomo na universidade

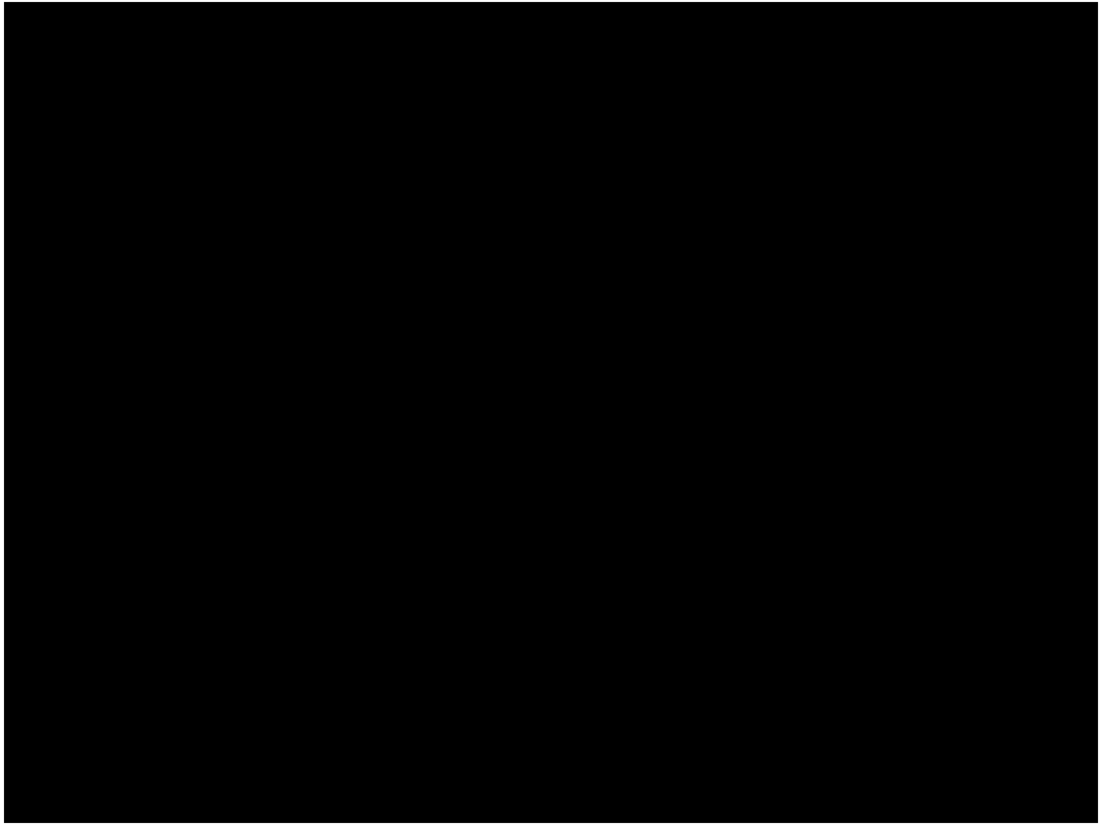


DeepTraffic: Deep Reinforcement Learning



Fonte: MIT 6.S094: Deep Learning for Self-Driving Cars

Carro autônomo no Brasil: um breve histórico



End to End Learning for Self-Driving Cars

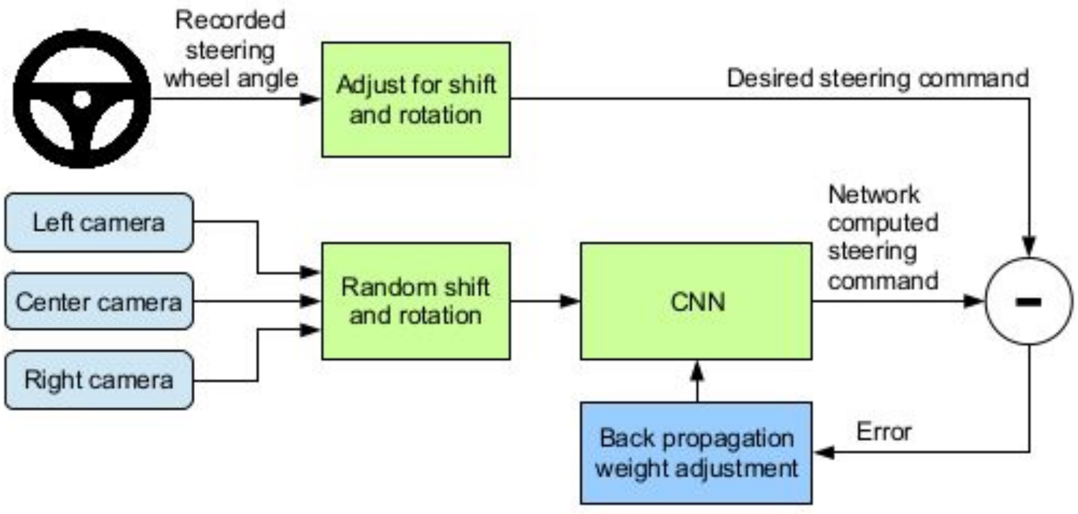


Figure 2: Training the neural network.

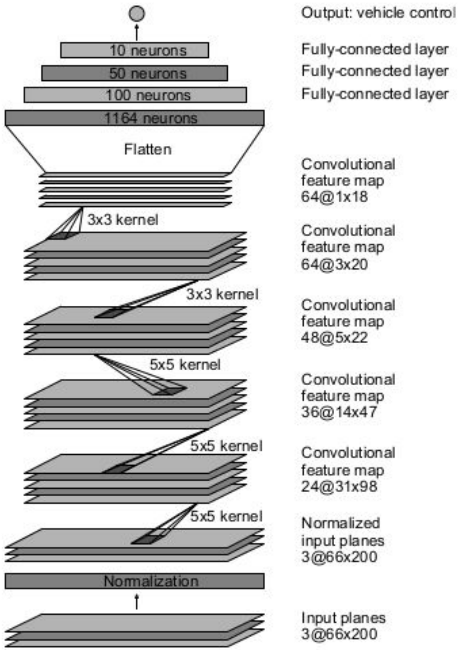
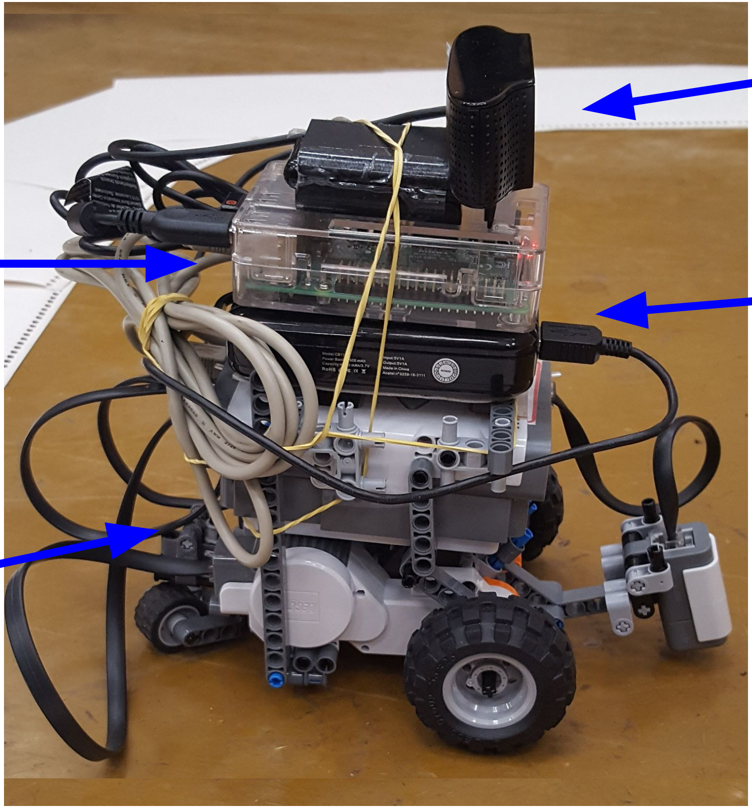


Figure 4: CNN architecture. The network has about 27 million connections and 250 thousand parameters.

Fonte: End to End Learning for Self-Driving Cars - Bojarski, M., et al.

Recursos utilizados

Raspberry Pi



Webcam

Power bank

Lego NXT

custo total: $ 60

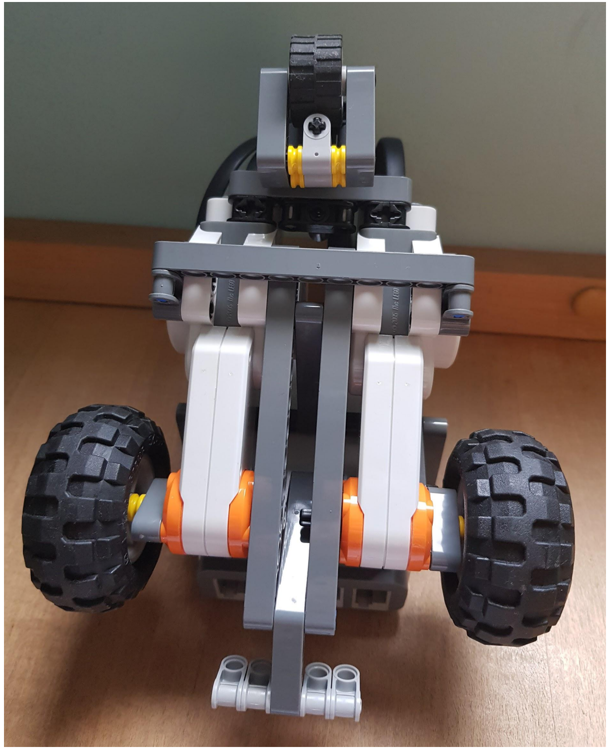
OpenCV

TensorFlow™

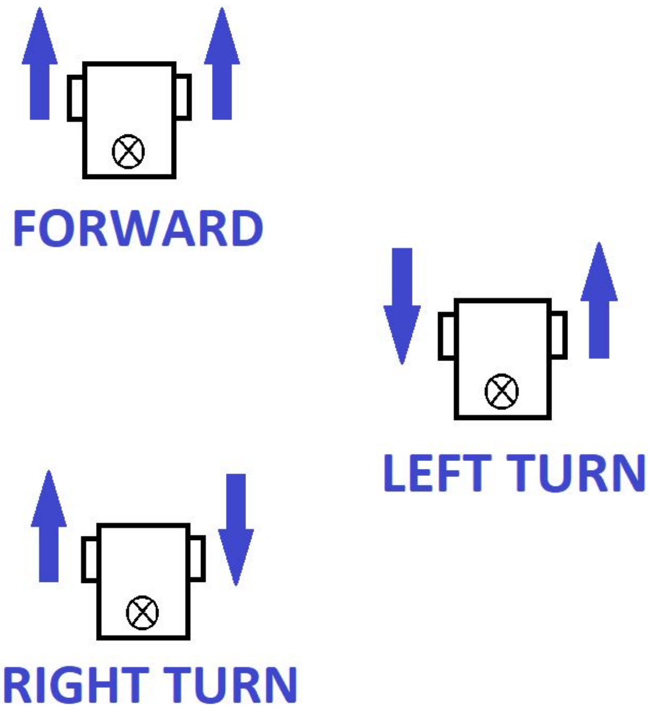
NXT-PYTHON

Robô com acionamento diferencial

- Sistema com duas rodas controladas por **atuadores independentes**
- Possui uma roda passiva (***castor wheel***) para maior estabilidade
- Benefícios:
  - simplicidade
  - permite girar no próprio eixo
- Movimentação baseada na **diferença de velocidade** entre os motores



Robô com acionamento diferencial

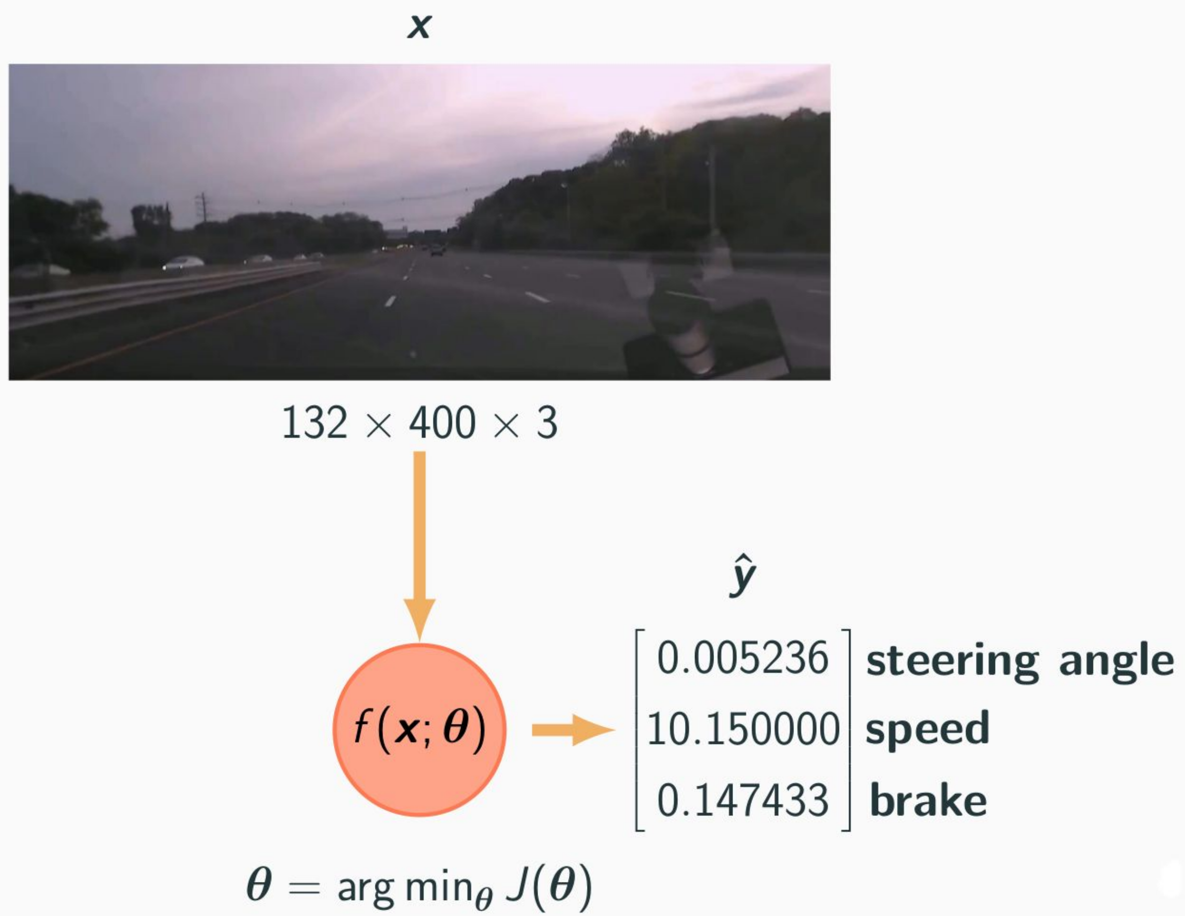



```python
self.leftMotor = nxt.Motor(self.brick, nxt.PORT_B)
self.rightMotor = nxt.Motor(self.brick, nxt.PORT_A)
self.both = nxt.SynchronizedMotors(self.leftMotor,
                                   self.rightMotor,
                                   turn_ratio)
def move_up(self):
    """
    Execute action of moving up
    """
    self.both.run(self.power_up)

def move_left(self):
    """
    Execute action of moving left for tacho_left degrees
    """
    self.rightMotor.weak_turn(self.power_left, self.tacho_left)
    self.leftMotor.weak_turn(- self.power_left, self.tacho_left)

def move_right(self):
    """
    Execute action of moving rigth for tacho_right degrees
    """
    self.rightMotor.weak_turn(- self.power_right, self.tacho_right)
    self.leftMotor.weak_turn(self.power_right, self.tacho_right)
```

Controle como regressão

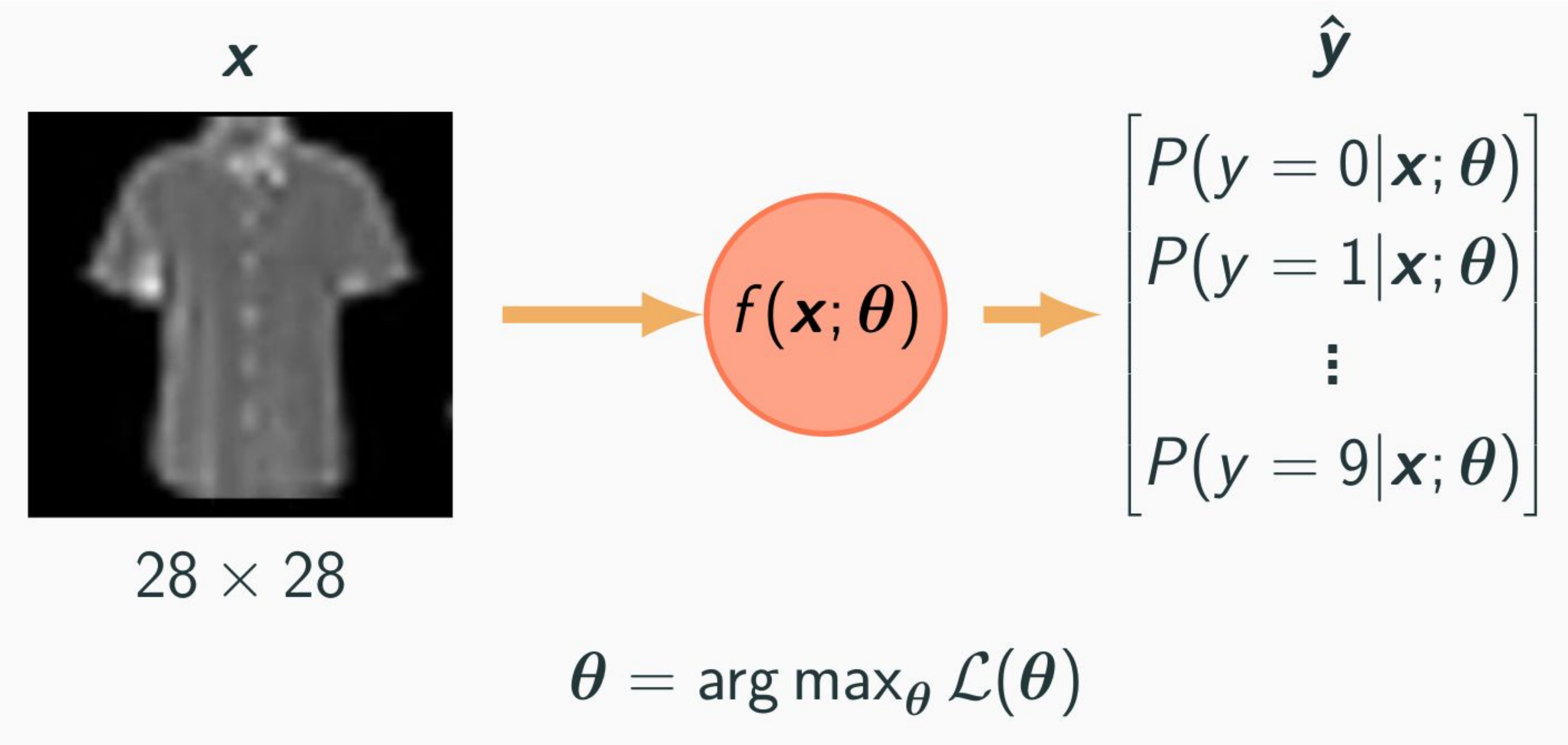


Exemplos:

Udacity’s Lincoln MKZ

DeepTesla

Classificação de imagens



Fonte: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms- Xiao, K., et al.

Controle como classificação de imagens

$\boldsymbol{x}$

$45 \times 80 \times 3$

$$f(\boldsymbol{x}; \boldsymbol{\theta})$$

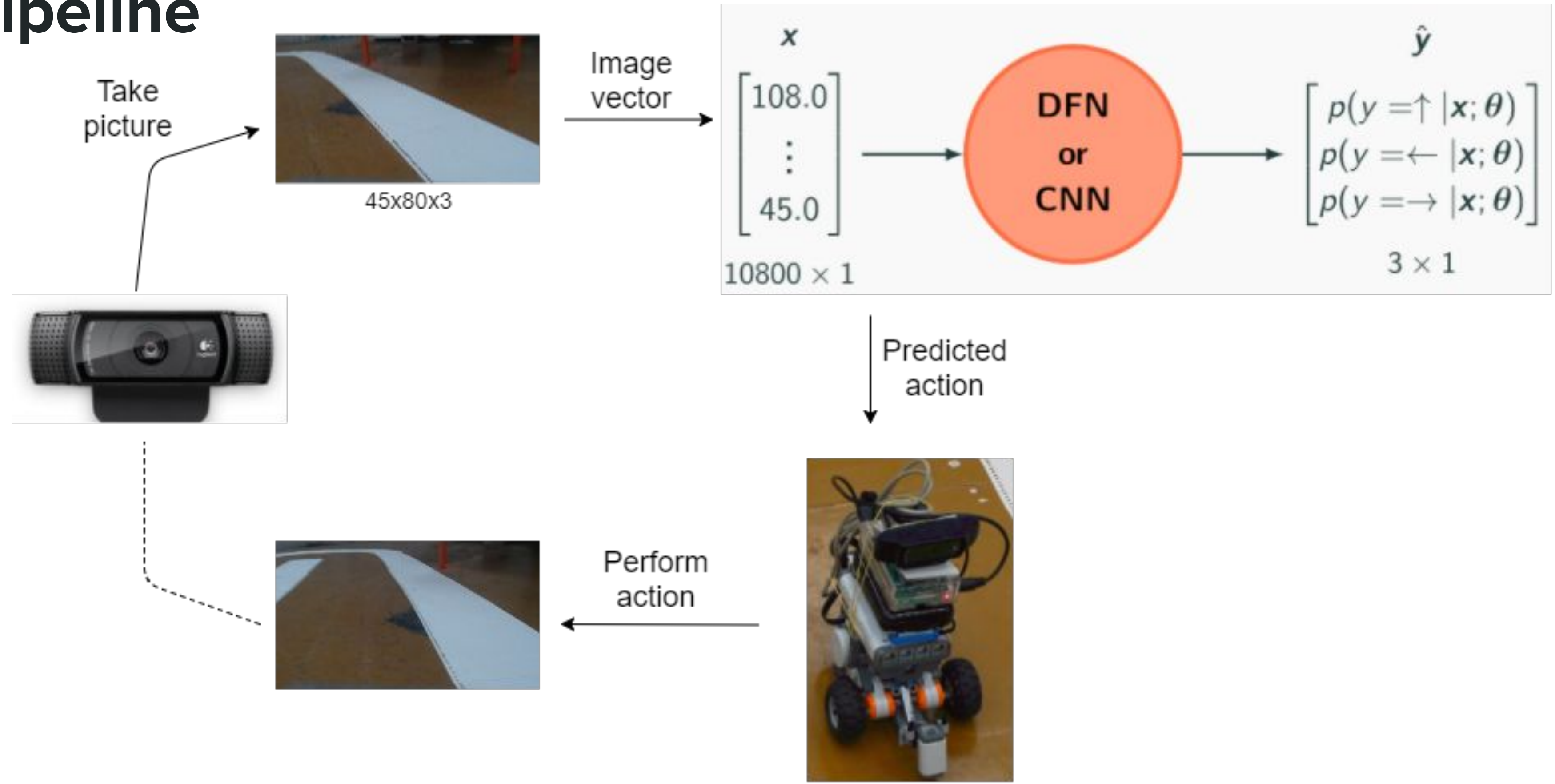
$$\hat{\boldsymbol{y}} = \begin{bmatrix} P(y = a_1 | \boldsymbol{x}; \boldsymbol{\theta}) \\ P(y = a_2 | \boldsymbol{x}; \boldsymbol{\theta}) \\ \vdots \\ P(y = a_n | \boldsymbol{x}; \boldsymbol{\theta}) \end{bmatrix}$$

$$\boldsymbol{\theta} = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$$

$$\pi(a | \boldsymbol{x}) = P(y = a | \boldsymbol{x}; \boldsymbol{\theta})$$

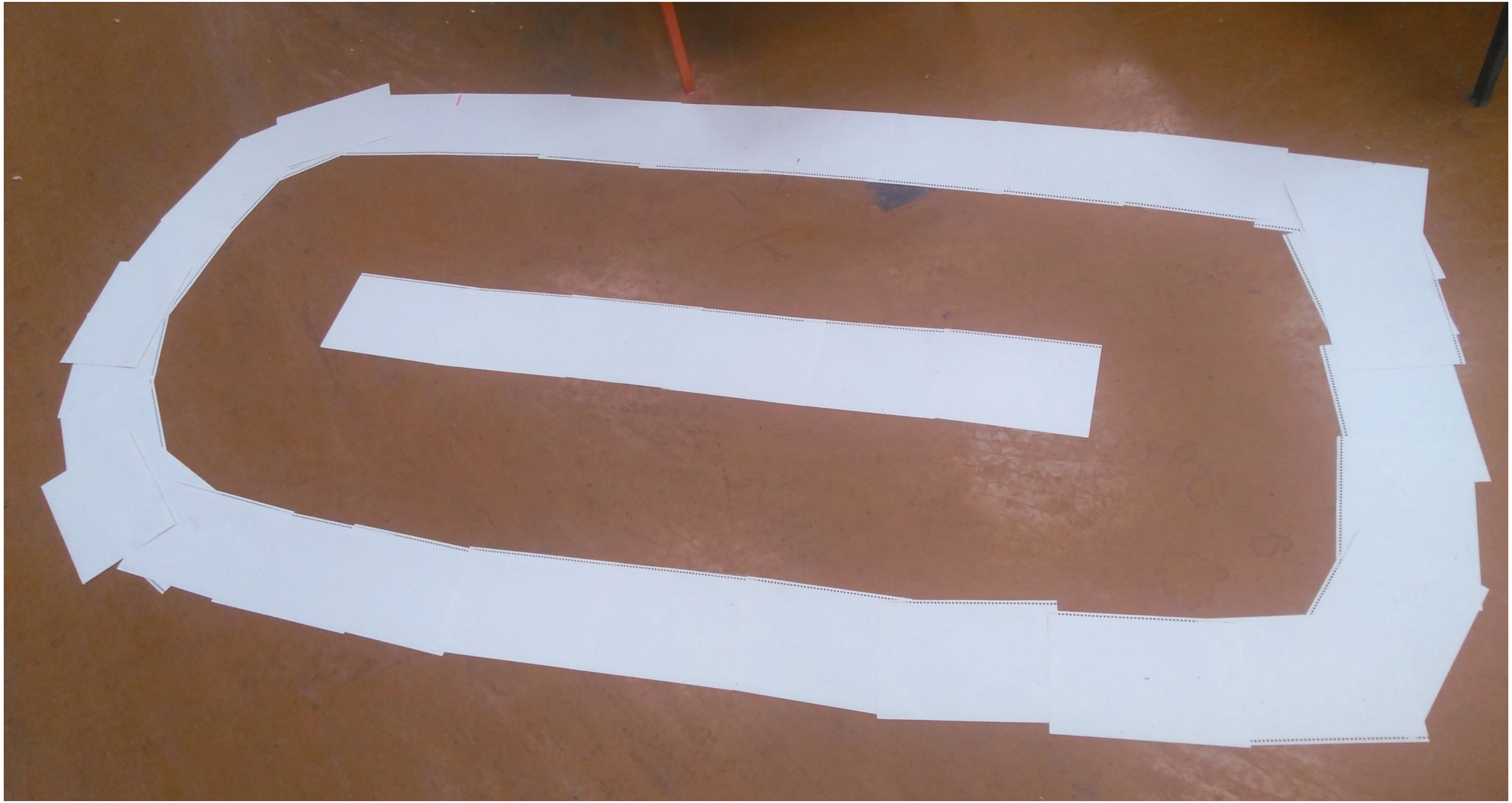
$$\mu(\boldsymbol{x}) = \arg\max_a P(y = a | \boldsymbol{x}; \boldsymbol{\theta})$$

Pipeline

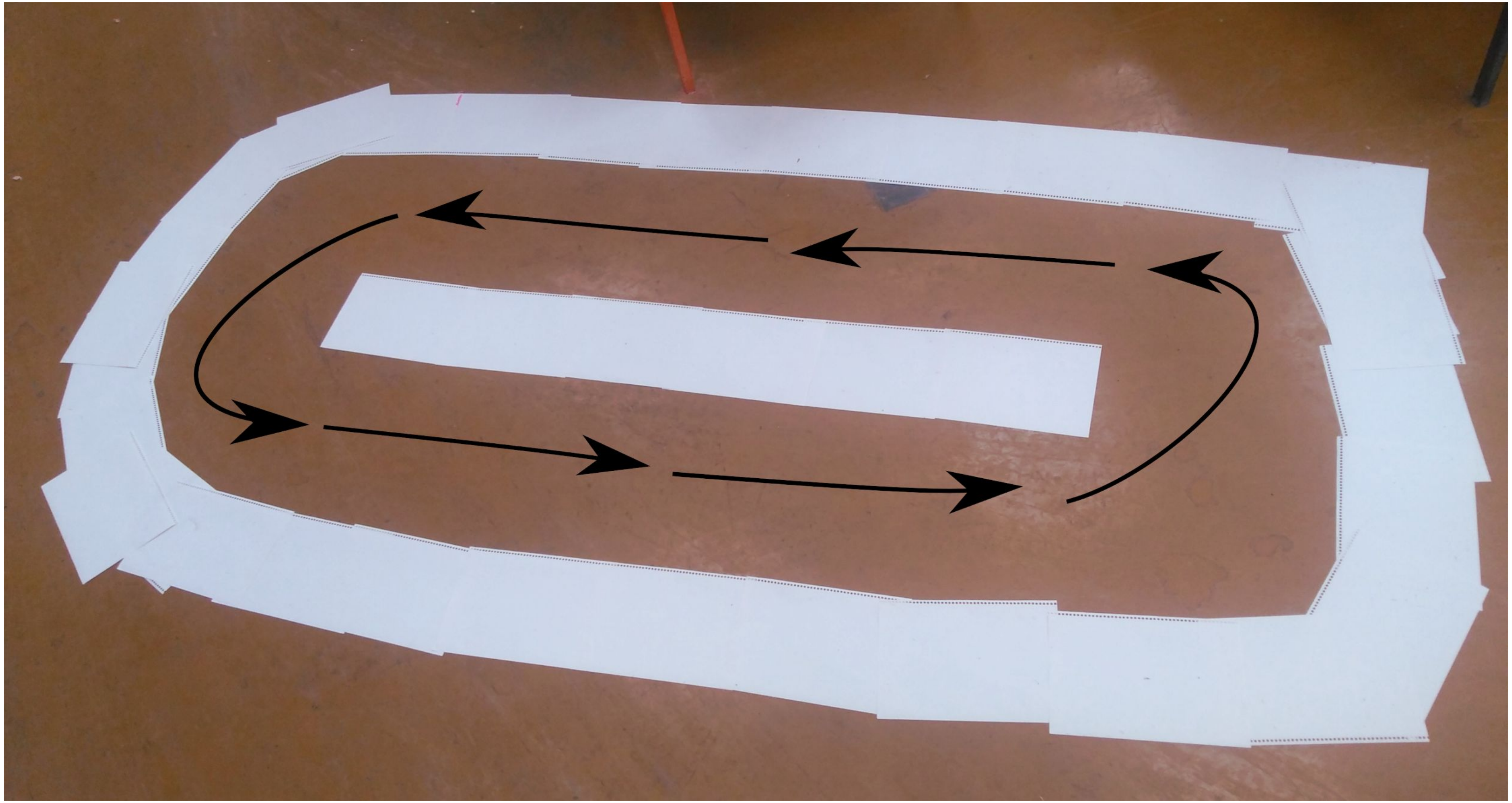


Take picture
45x80x3
Image vector
x
108.0
⋮
45.0
10800 × 1
DFN or CNN
ŷ
p(y =↑ |x; θ)
p(y =← |x; θ)
p(y =→ |x; θ)
3 × 1
Predicted action
Perform action

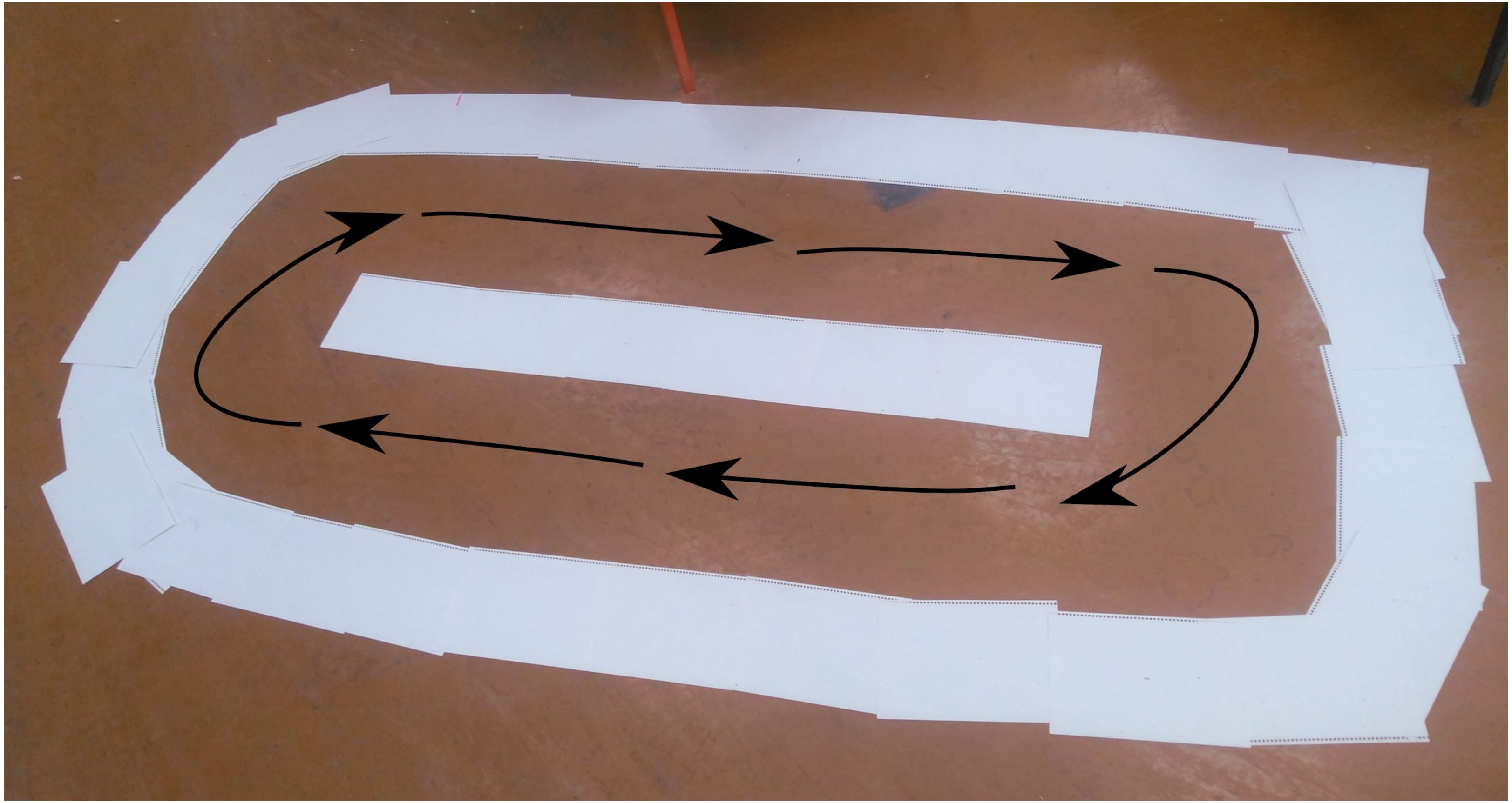
Coleta de dados

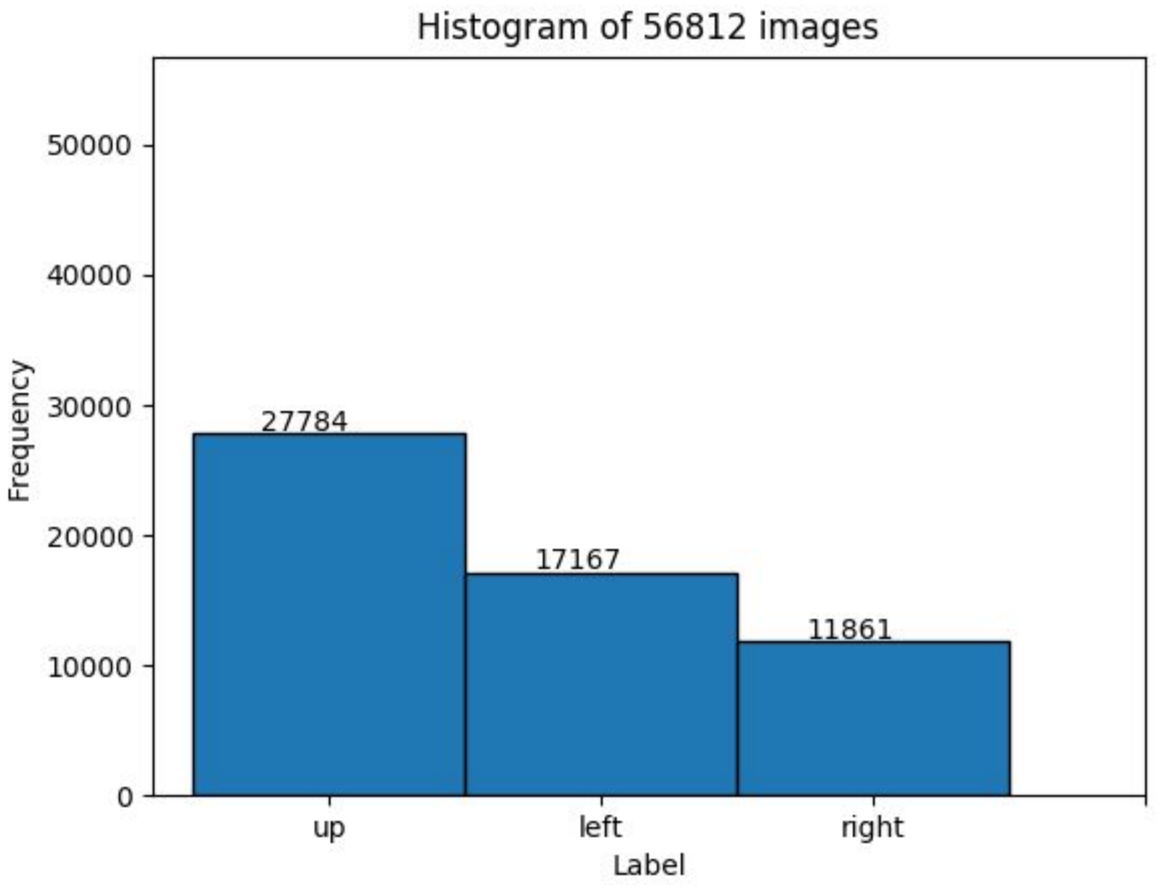


Coleta de dados



Coleta de dados




Histogram of 56812 images
50000
40000
30000
20000
10000
0
Frequency
27784
17167
11861
up
left
right
Label

Manipulação de imagens

Original

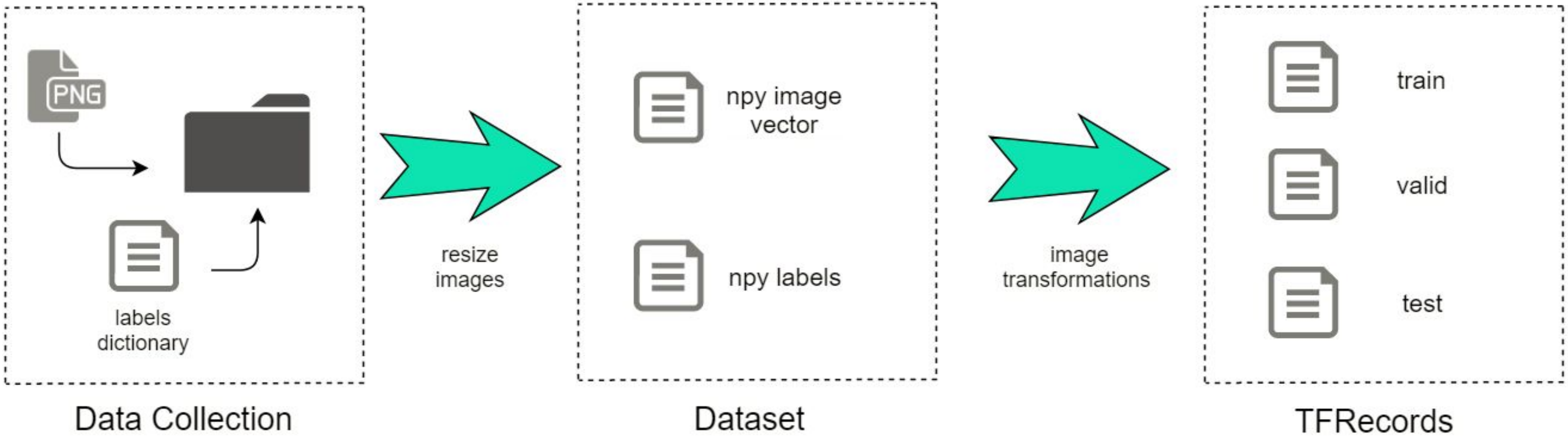
Binary

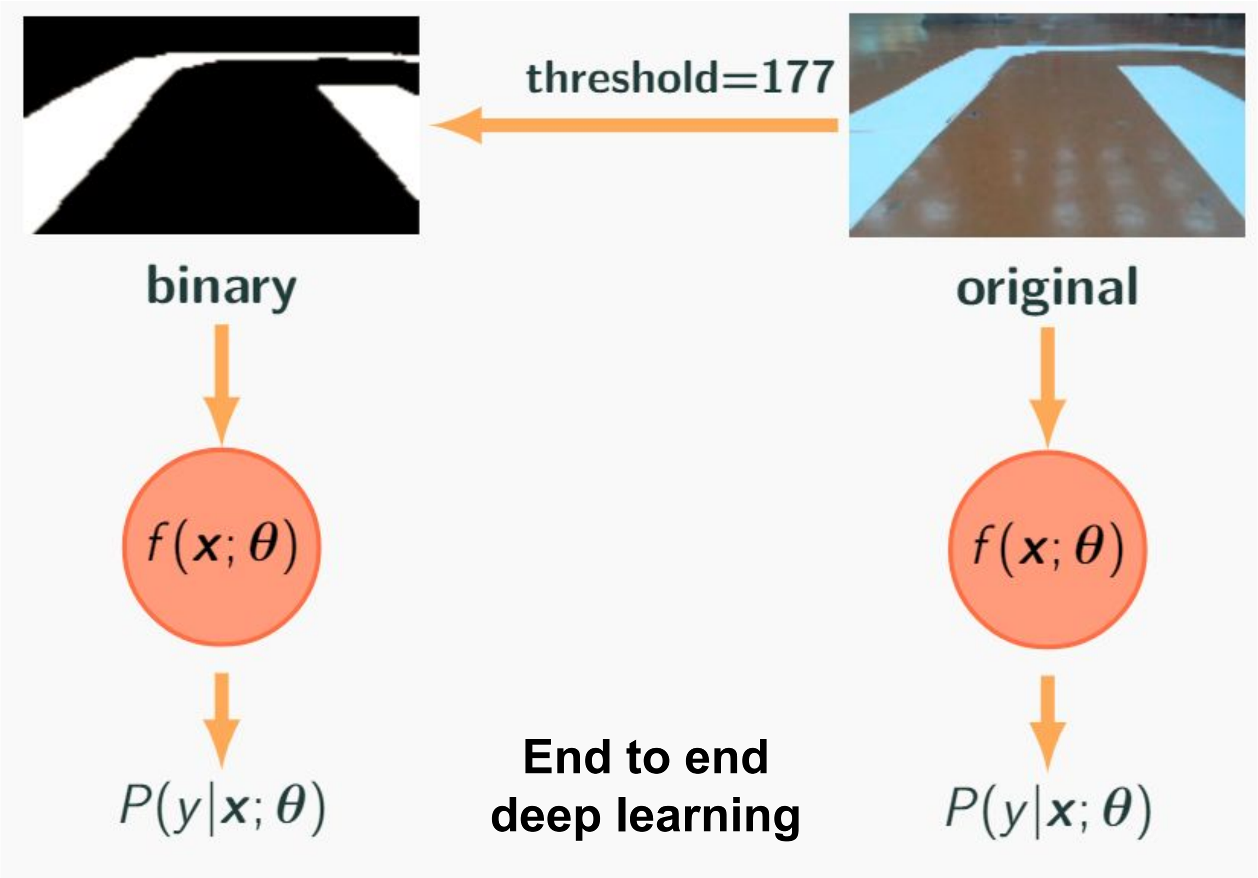
Random shadow

Grayscale

Gaussian Blur 5x5

Data pipeline





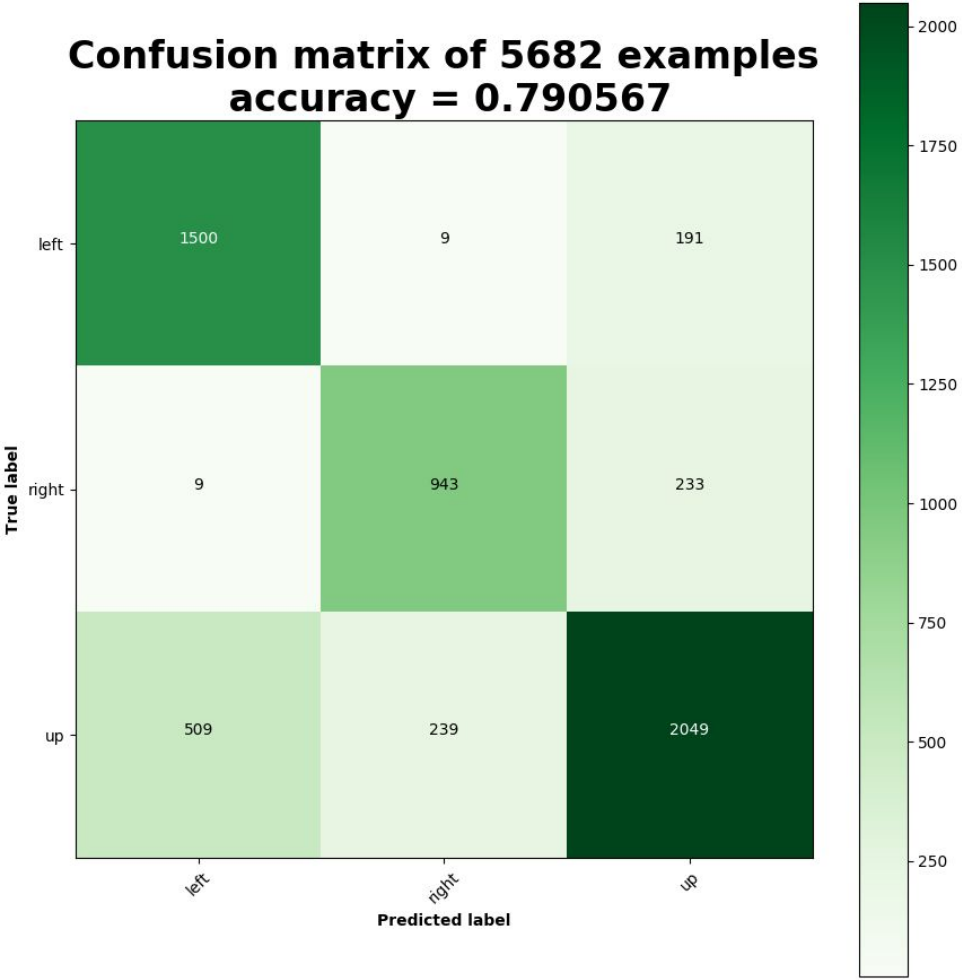
End to end deep learning

Resultados com DFN

architecture	preprocessing	hit rate $\uparrow$	hit rate $\leftarrow$	hit rate $\rightarrow$
[3]	none	0.80	0.76	0.68
[2350, 3]	none	0.80	0.70	0.80
[1333, 200, 3]	none	0.79	0.82	0.67
[3]	binarization	0.75	0.87	0.64
[233, 3]	binarization	0.72	0.85	0.82
[1628, 47, 3]	binarization	0.71	0.90	0.84

Resultados com CNN

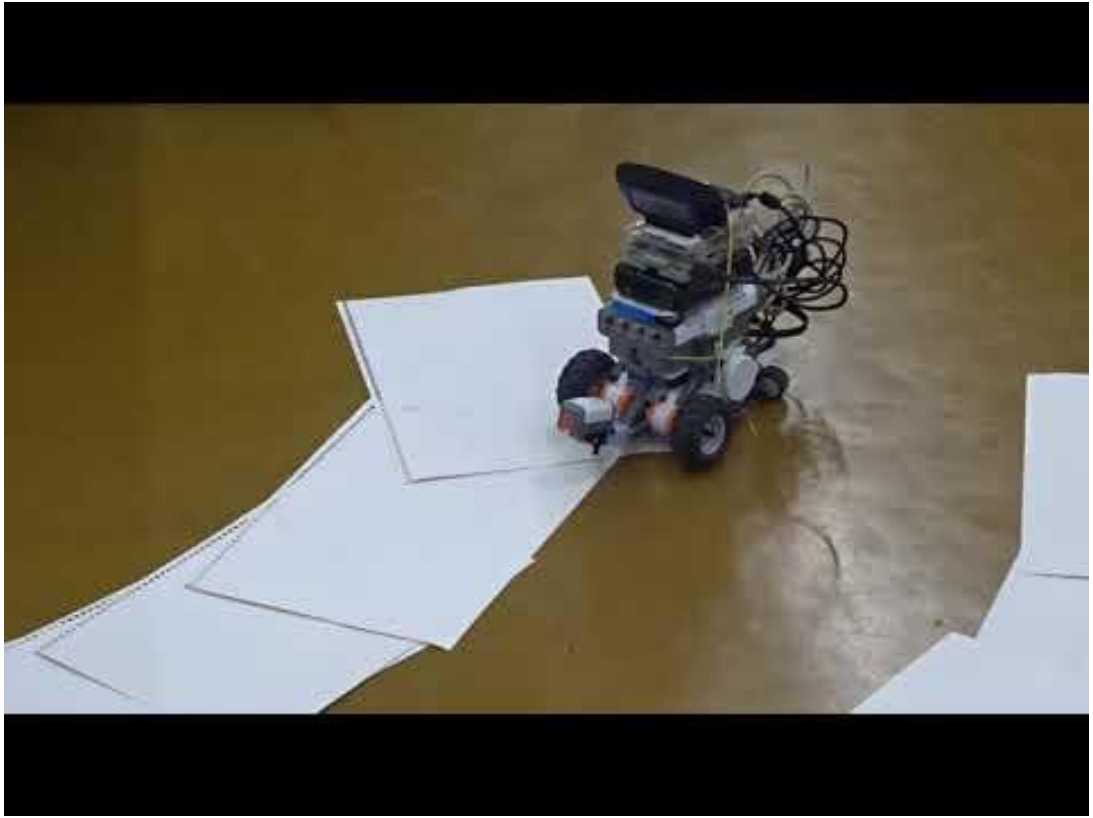
architecture	preprocessing	hit rate $\uparrow$	hit rate $\leftarrow$	hit rate $\rightarrow$
$[(24, 5), 731, 3]$	none	0.80	0.71	0.73
$[(32, 5), (64, 5), 3]$	none	0.82	0.80	0.66
$[(24, 5), (36, 5), (64, 5), 200, 3]$	none	0.76	0.84	0.72
$[(24, 5), 456, 3]$	binarization	0.79	0.86	0.67
$[(32, 5), (64, 5), 3]$	binarization	0.78	0.80	0.73
$[(24, 5), (36, 5), (64, 5), 200, 3]$	binarization	0.79	0.83	0.73



Confusion matrix of 5682 examples
accuracy = 0.790567
True label
Predicted label
left
right
up
1500
9
191
9
943
233
509
239
2049
2000
1750
1500
1250
1000
750
500
250

Simulação

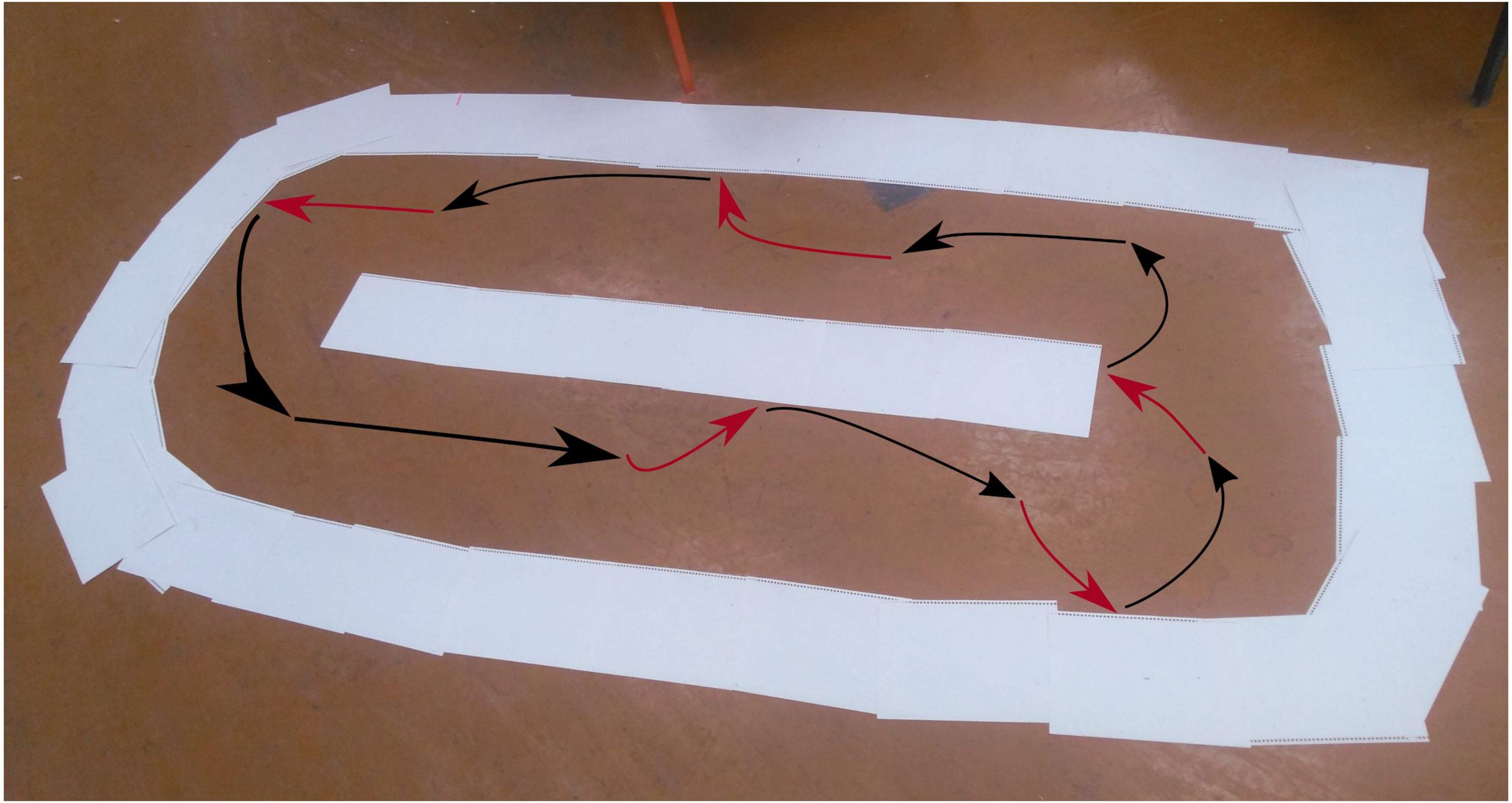
Primeira Tentativa



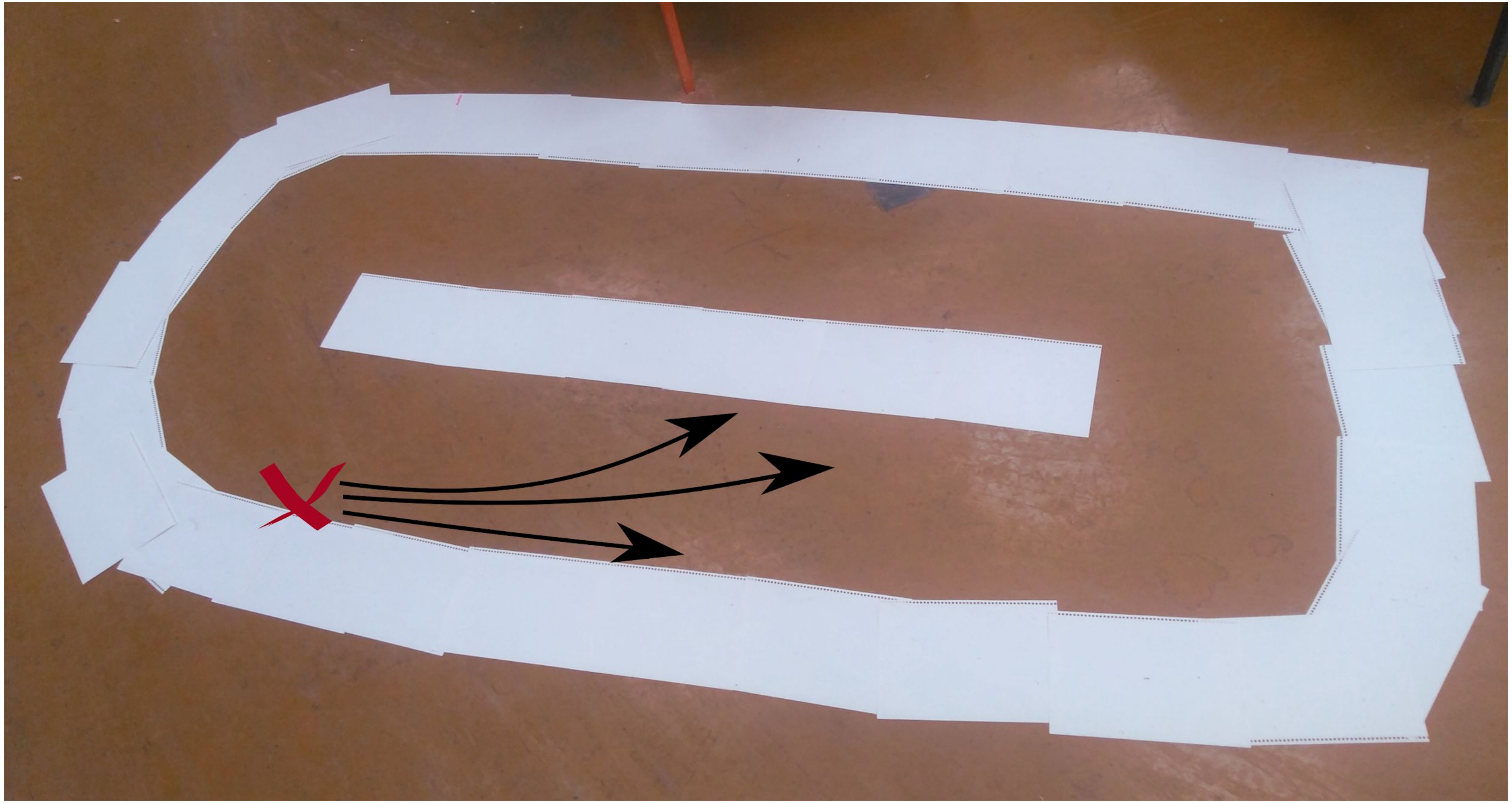
PROBLEMA: dados pouco informativos

Comando “ir para frente” estava fortemente associado ao robô estar centralizado na pista

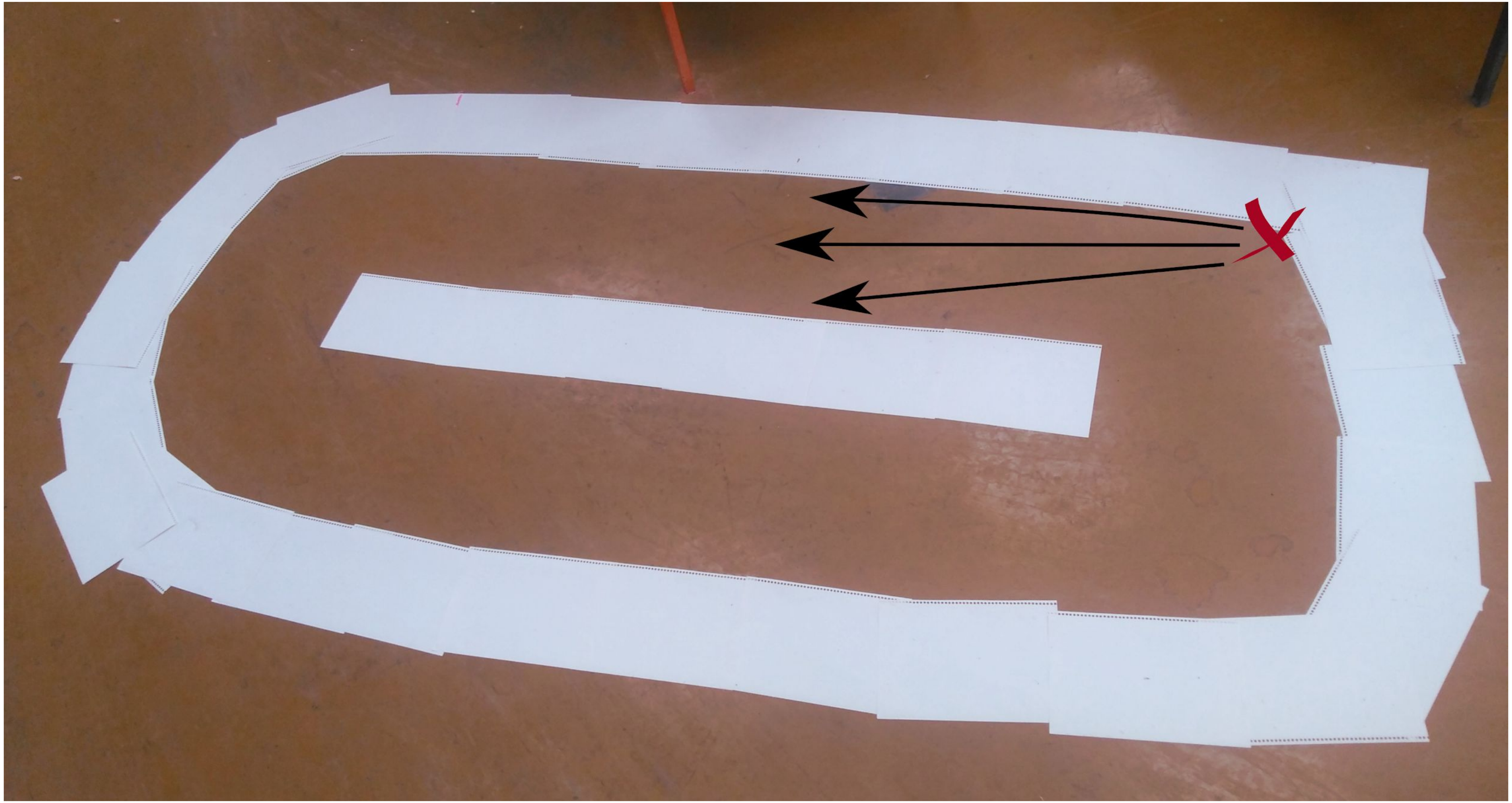
Nova coleta de dados



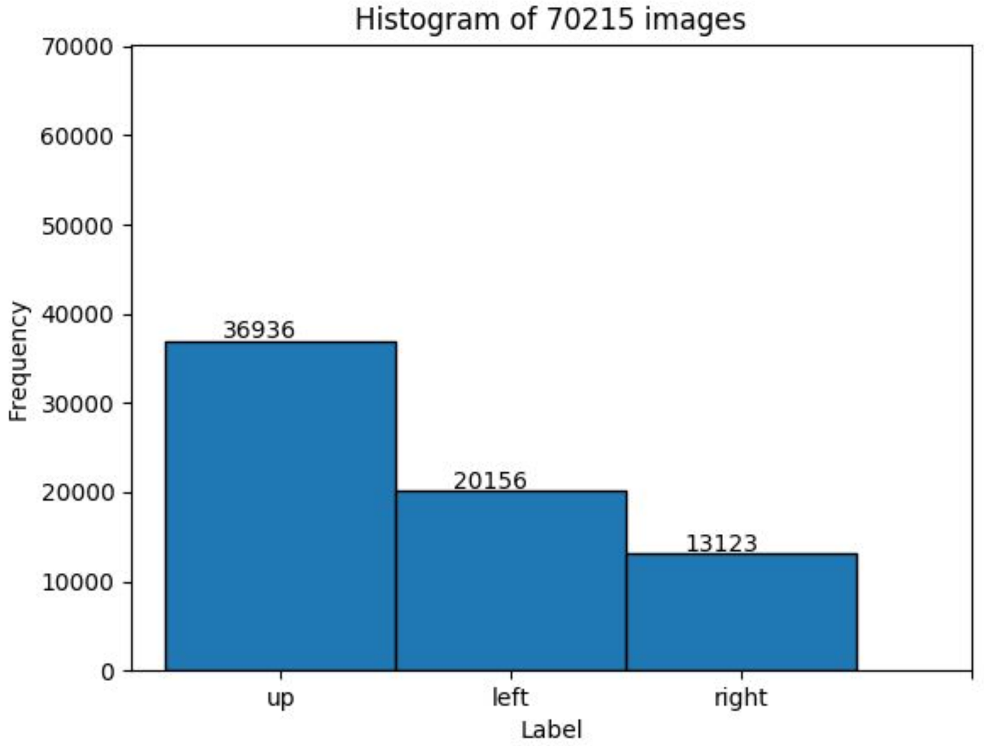
Nova coleta de dados



Nova coleta de dados



Nova coleta de dados: + 13403 imagens

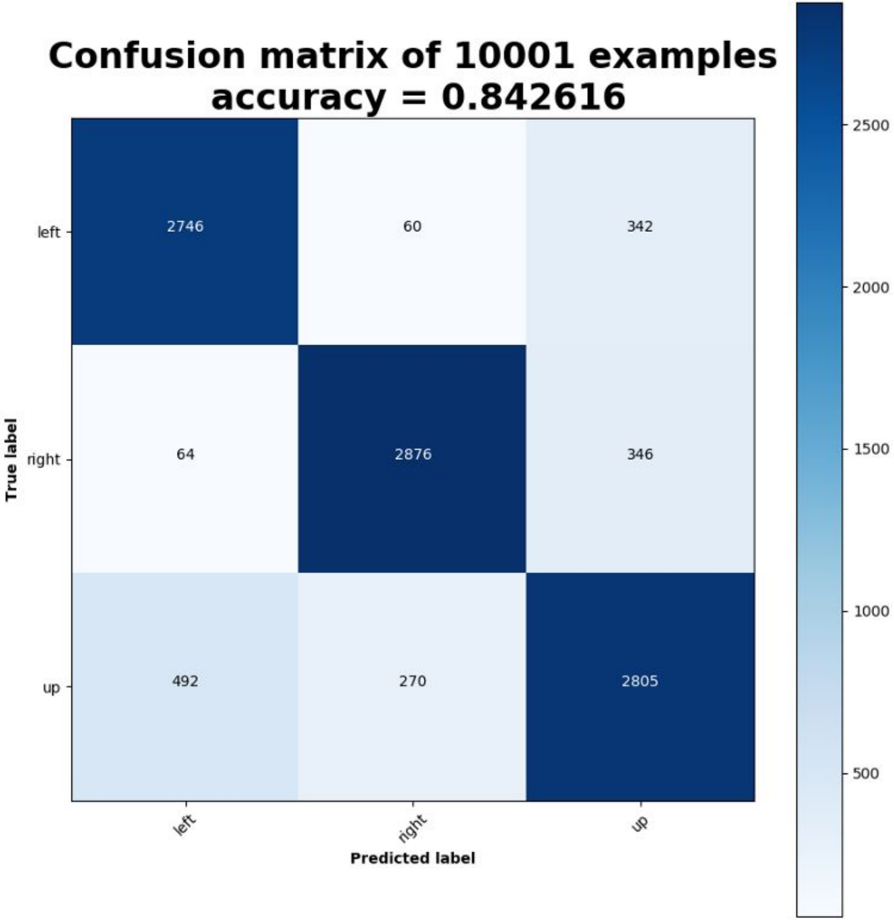


PROBLEMA: predição x controle

captura e processamento de imagem ~ 9 ms

command	time
$\uparrow$	0.06 s
$\leftarrow$	0.29 s
$\rightarrow$	0.29 s

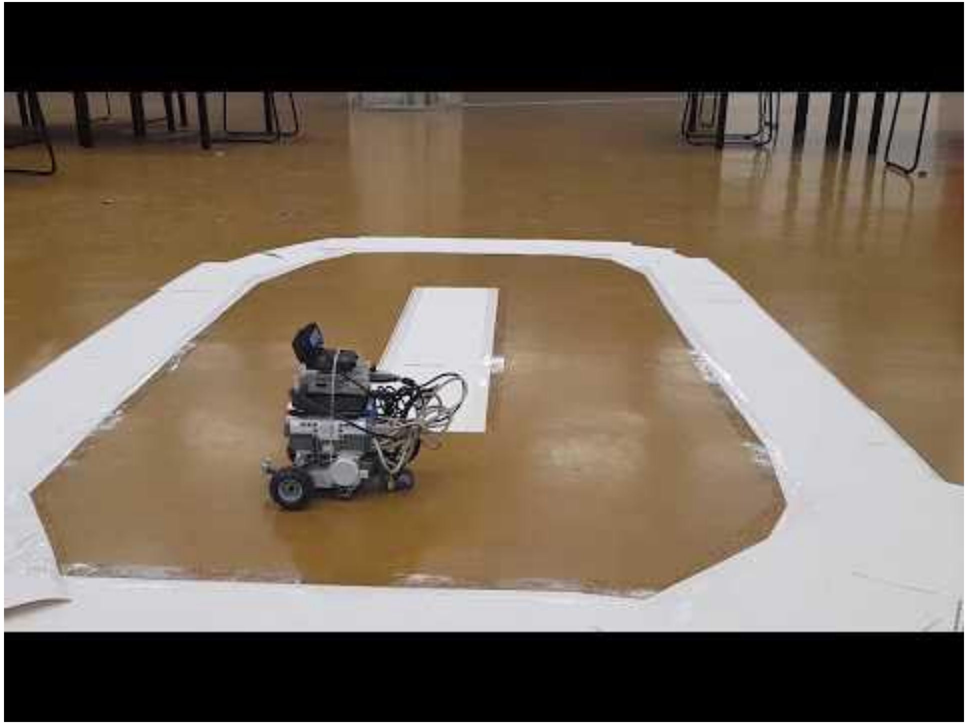
architecture	preprocessing	inference time
$[3]$	none	0.43 s
$[1333, 200, 3]$	none	1.35 s
$[(24, 5), 731, 3]$	none	1.41 s
$[3]$	binarization	0.42 s
$[233, 3]$	binarization	0.59 s
$[(32, 5), (64, 5), 3]$	binarization	0.89 s
$[(24, 5), (36, 5), (64, 5), 200, 3]$	binarization	1.24 s



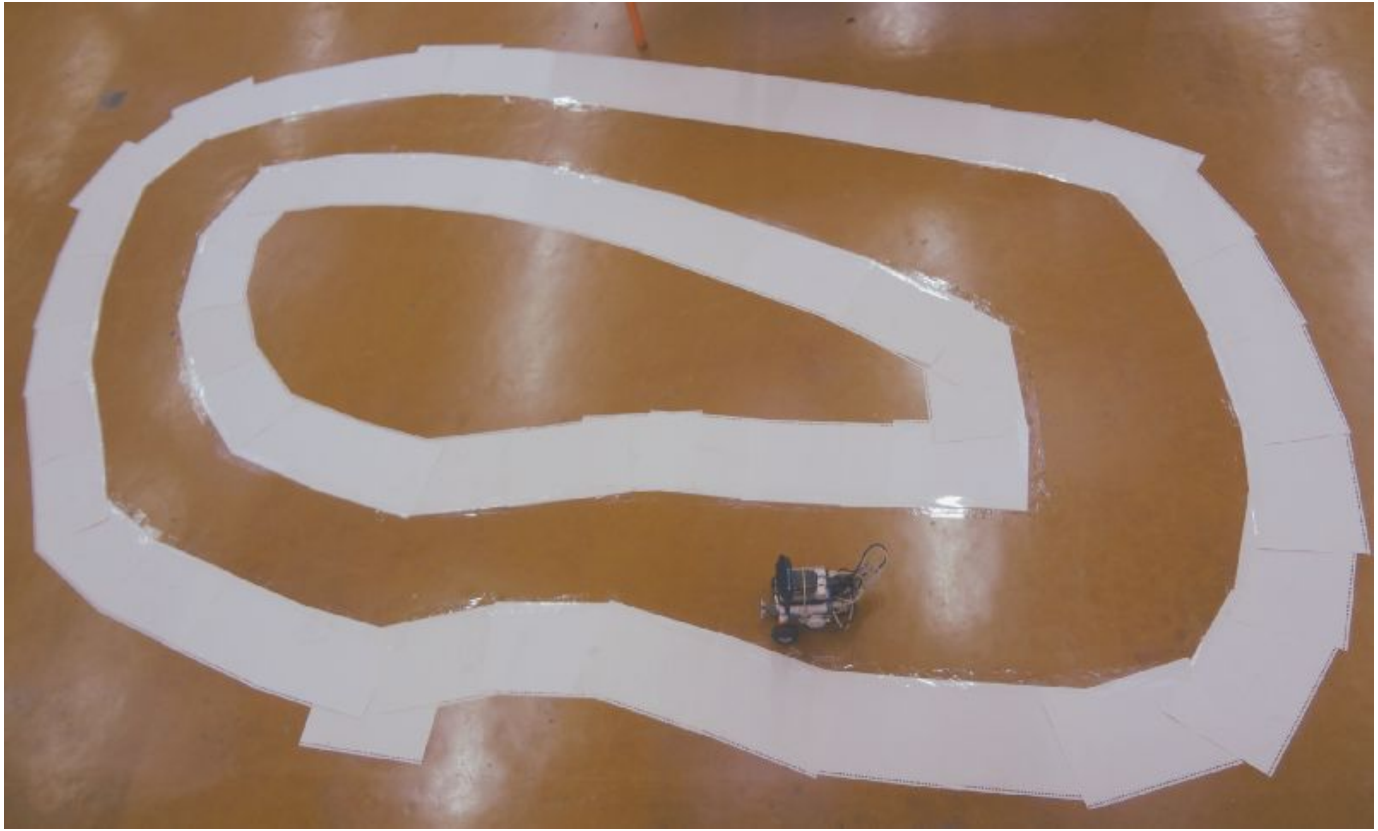
Melhor modelo: $[(36, 5), 3]$

- imagens RGB
- uma camada de convolução: 36 filtros 5x5
- uma camada de pooling 2x2
- sem camadas escondidas
- tempo de inferência: 0.69 s

Modo autônomo - pista de treino



Pista de teste



Modo autônomo - pista de teste

Próximos passos

- **Experimentar novos modelos de direção, e.g. modelo Ackerman**
- **Integrar o robô com outros sensores:**
  - GPS e Radar
- **Detecção de obstáculos**
- **Path planning**
  - Planejar trajetórias do ponto A ao ponto B

Conclusão

- Machine learning em **aplicações reais** é mais complicada do que em ambientes simulados
- **Começar** com deep learning para carro autônomo é **fácil**
- É possível implementar técnicas avançadas com hardware de **baixo custo**
- Variabilidade na coleta de dados é **essencial**
- O protótipo apresentado pode ser uma solução viável a disciplinas de **inteligência artificial** que desejem ir além de *toy problems*

Referências

- Artigo no Medium (@project_m)
    - Self drives me crazy: from 0 to autonomous car in 150 hours
- Código no GitHub
    - Self-Driving Pi Car
- Dataset
    - Self-Driving Data
- Curso do MIT:
    - Deep Learning for Self-Driving Cars

Agradecimentos

LIAMF

VISION

COMPUTER VISION RESEARCH GROUP

Obrigado