Rede Neural Convolucional

Aula 18 - 2 de maio de 2024

Pontos Principais

- Redes Neurais Artificiais
- Imagem digital
- Filtros para detecção de bordas
- Motivação
- Introdução à Rede Neural Convolucional (CNN- Convolutional neural network)
- Exemplos de CNNs
- Exemplo de aplicação: Controle de Veículo Autônomo Utilizando Inteligência Artificial
- Implementação de uma CNN utilizando Keras

Redes Neurais Artificiais

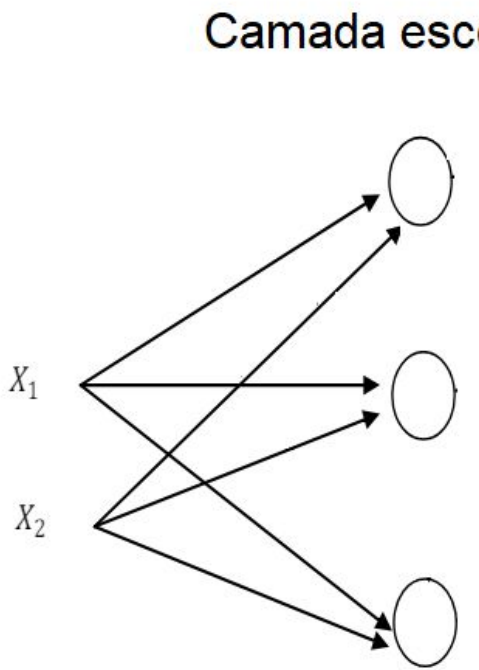
Redes Neurais Artificiais

Entrada

X_1

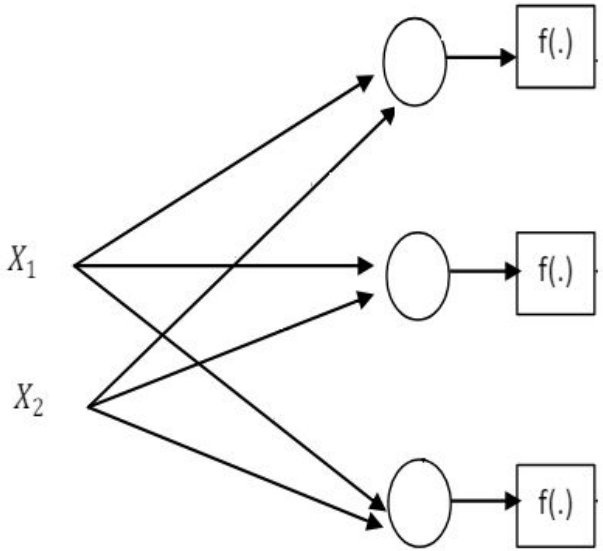
X_2

Redes Neurais Artificiais

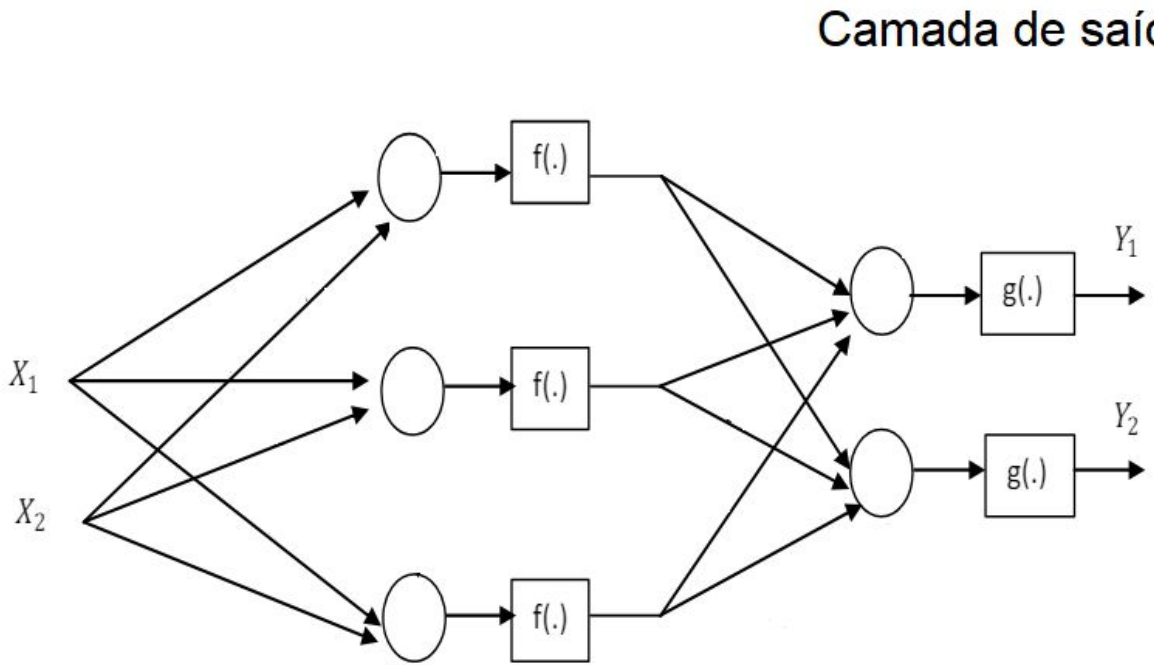


Redes Neurais Artificiais

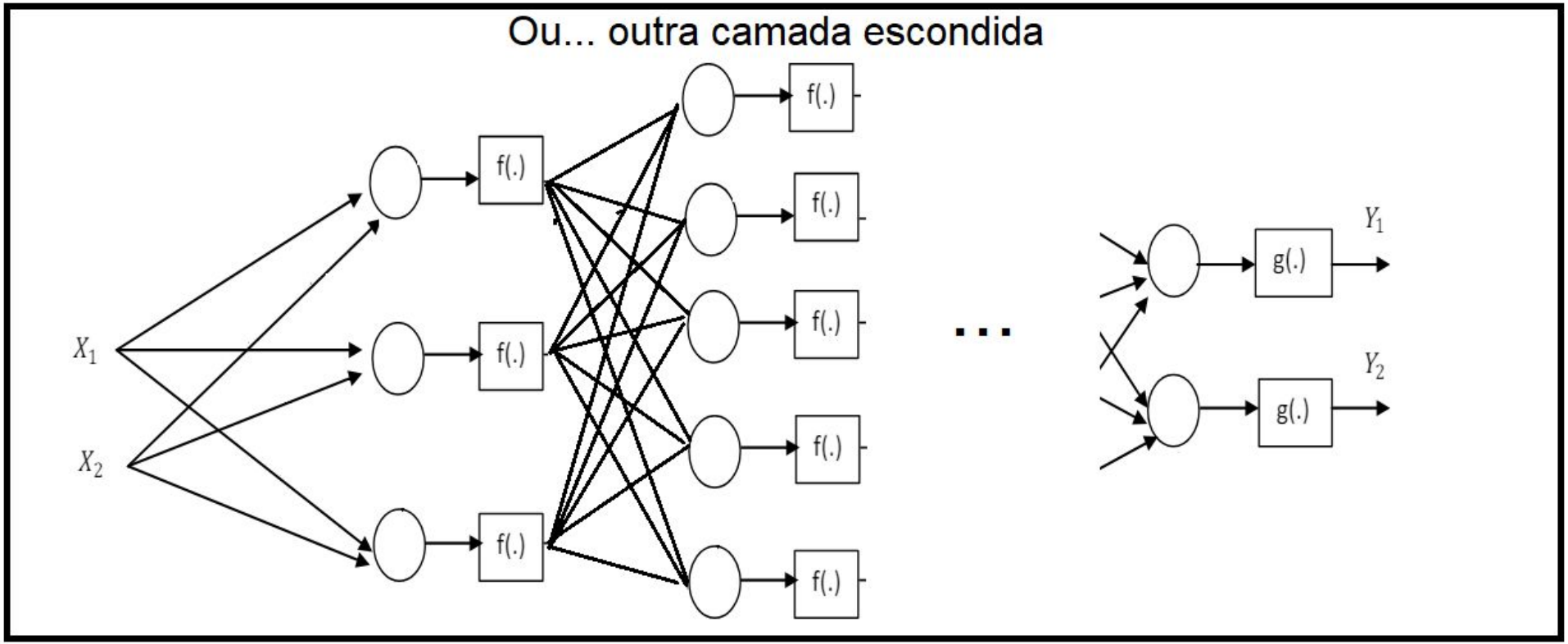
Função de ativação



Redes Neurais Artificiais



Redes Neurais Artificiais



Camada densa ou totalmente conectada

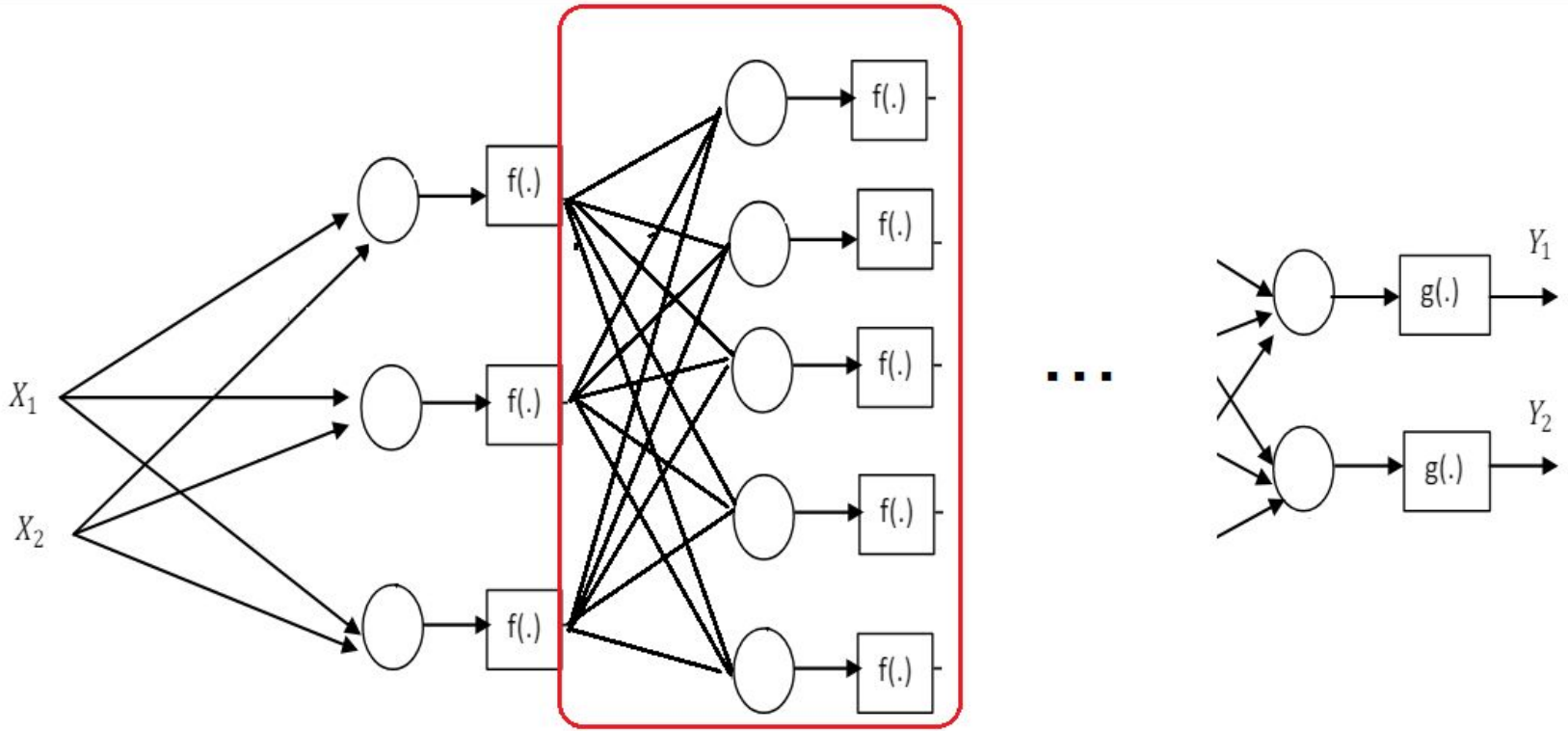


Imagem digital

Representação matricial de uma imagem digital

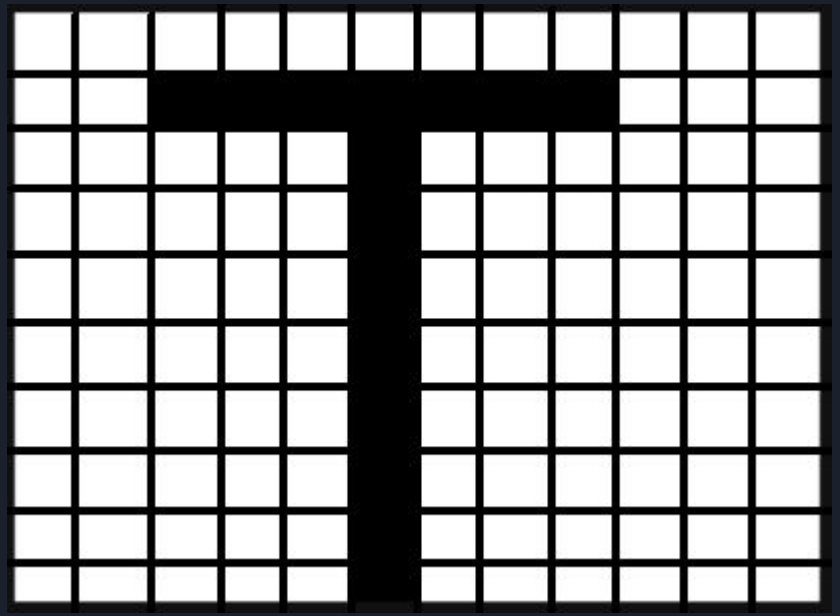
- Os valores da matriz representam a intensidade luminosa
- Normalmente 8 bits (0 a 255)
- Em escala de cinza 0 corresponde ao preto e 255 ao branco
- Sistema RGB - 3 canais de cor (Vermelho, verde e azul)

Representação matricial de uma imagem digital

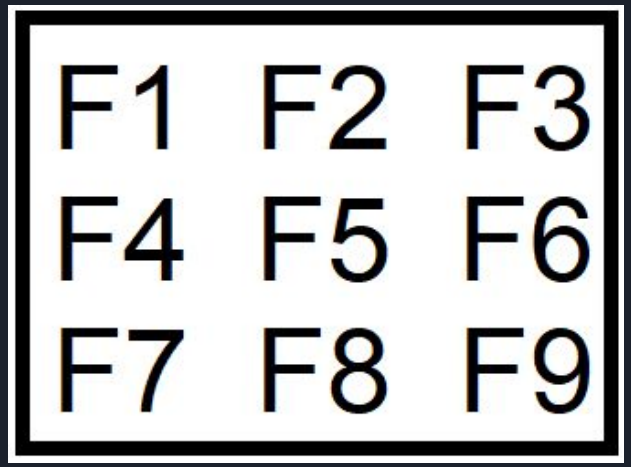
```
array([[255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255],
       [255, 255,   0,   0,   0,   0,   0,   0,   0, 255, 255, 255],
       [255, 255, 255, 255, 255,   0, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255,   0, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255,   0, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255,   0, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255,   0, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255,   0, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255,   0, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255,   0, 255, 255, 255, 255, 255, 255]])
```

Filtro de detecção de borda

Filtro de detecção de borda

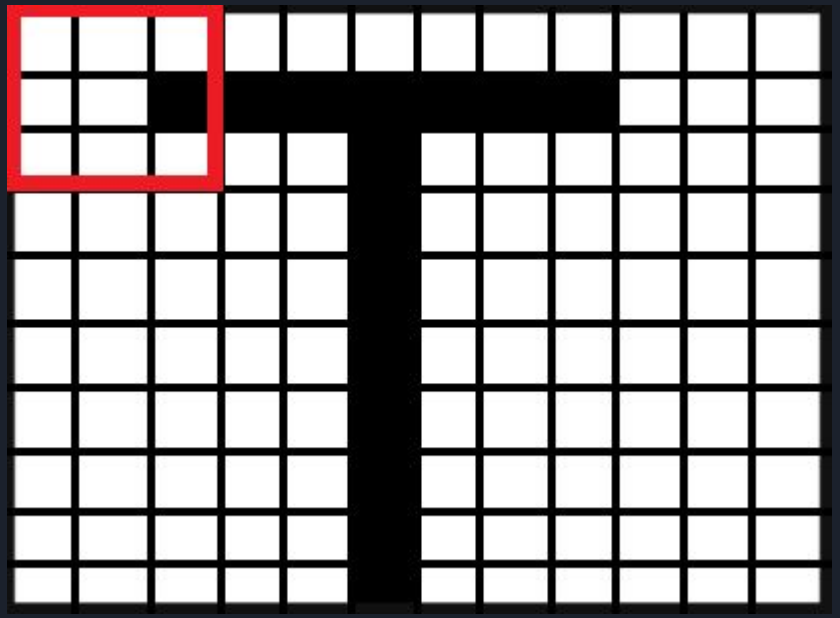


Filtro de detecção de borda



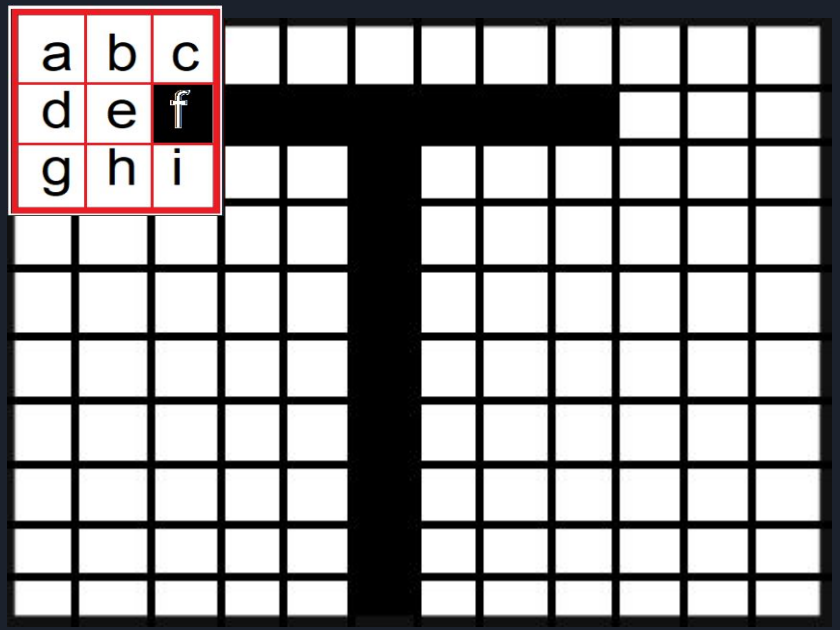
Filtro de detecção de borda

F1	F2	F3
F4	F5	F6
F7	F8	F9

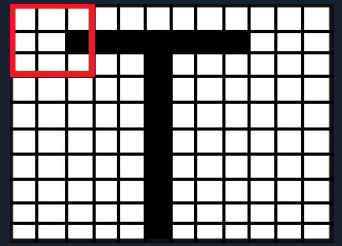


Filtro de detecção de borda

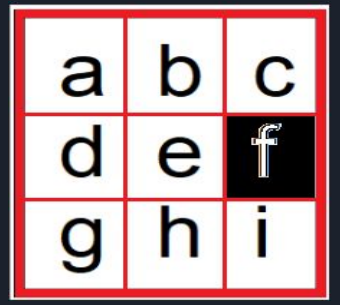
F1	F2	F3
F4	F5	F6
F7	F8	F9

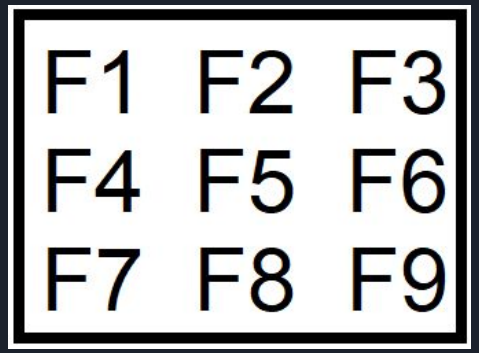


Filtro de detecção de borda



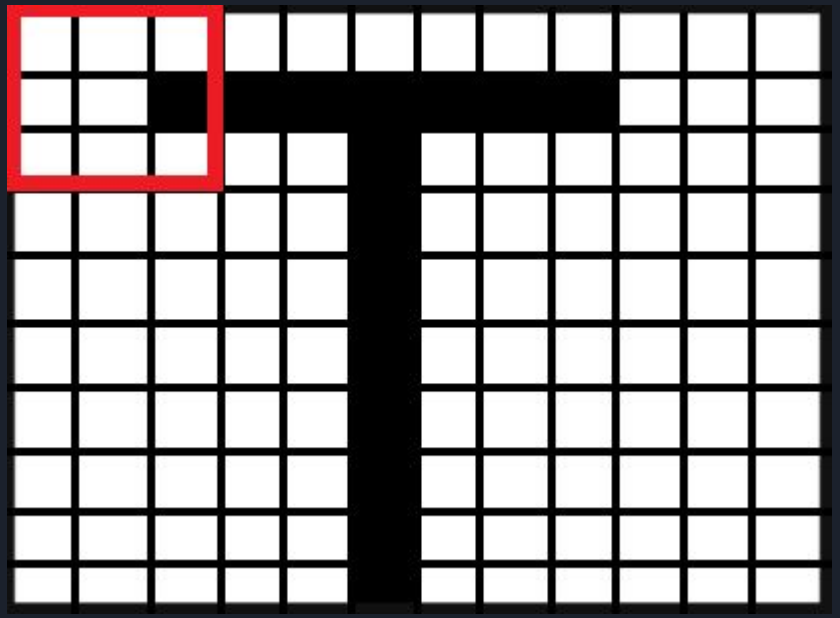
Resultado = F1.a + F2.b + F3.c + F4.d + F5.e + F6.f + F7.g + F8.h + F9.i



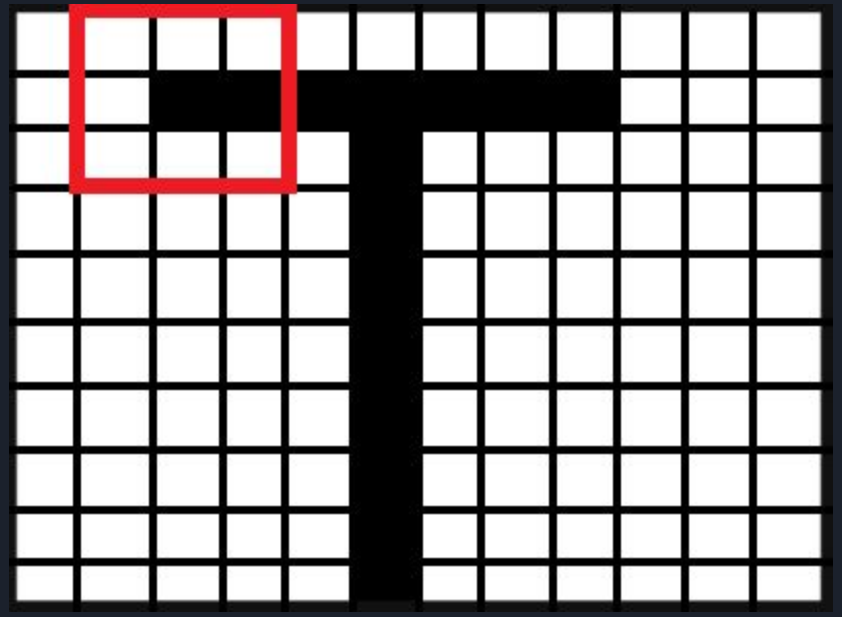


Filtro de detecção de borda

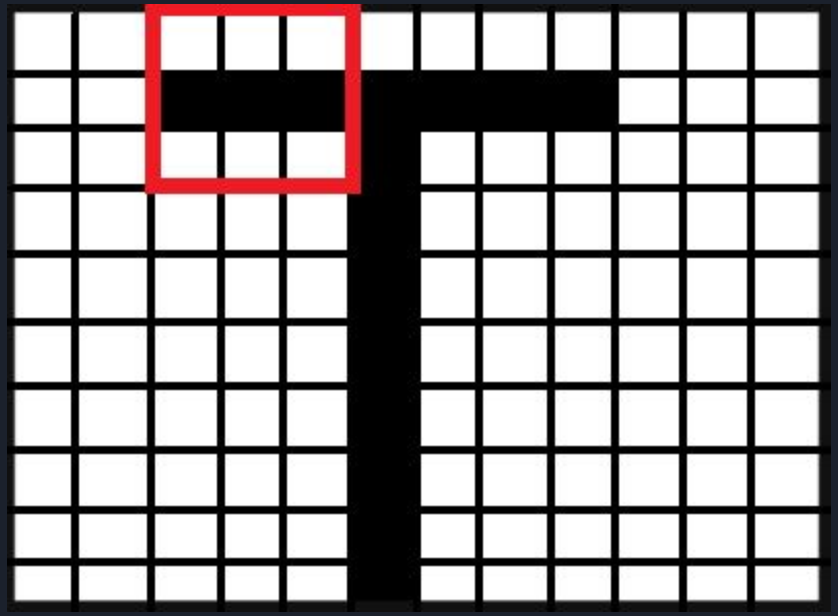
F1	F2	F3
F4	F5	F6
F7	F8	F9



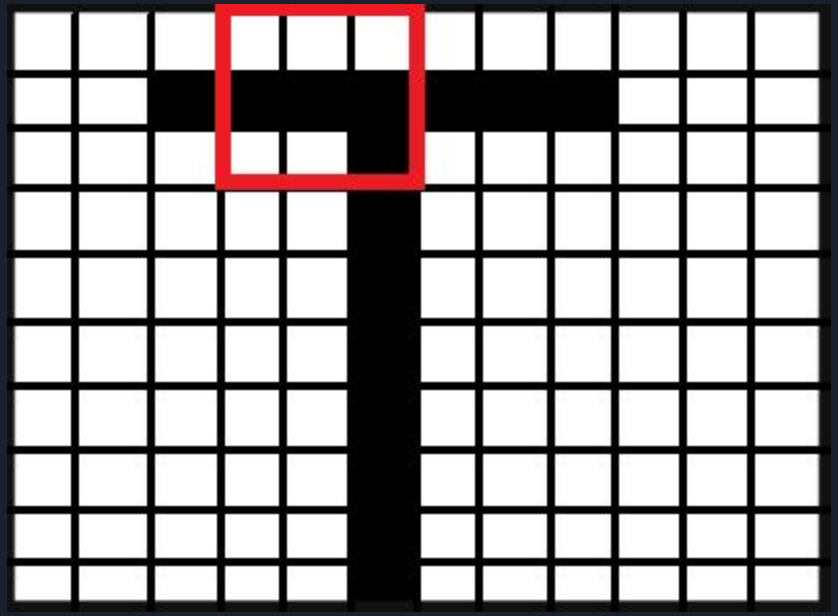
Filtro de detecção de borda



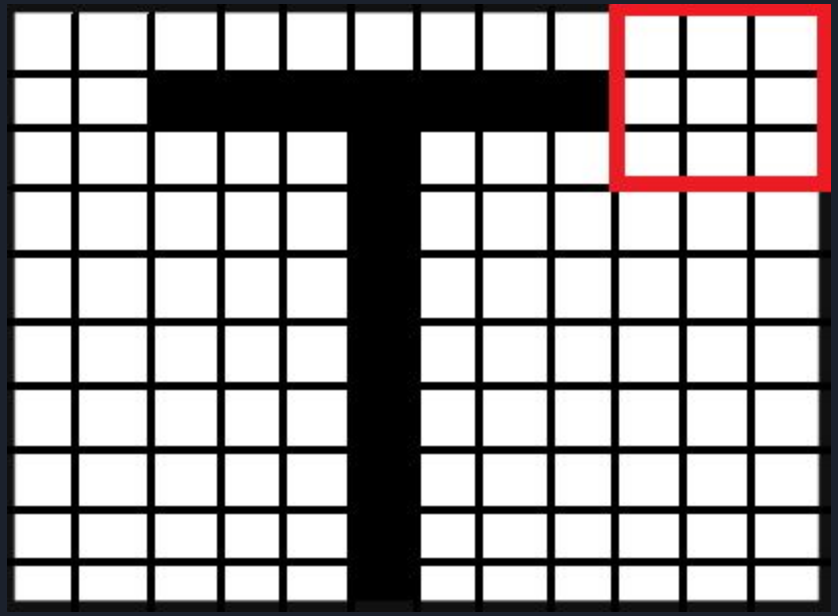
Filtro de detecção de borda



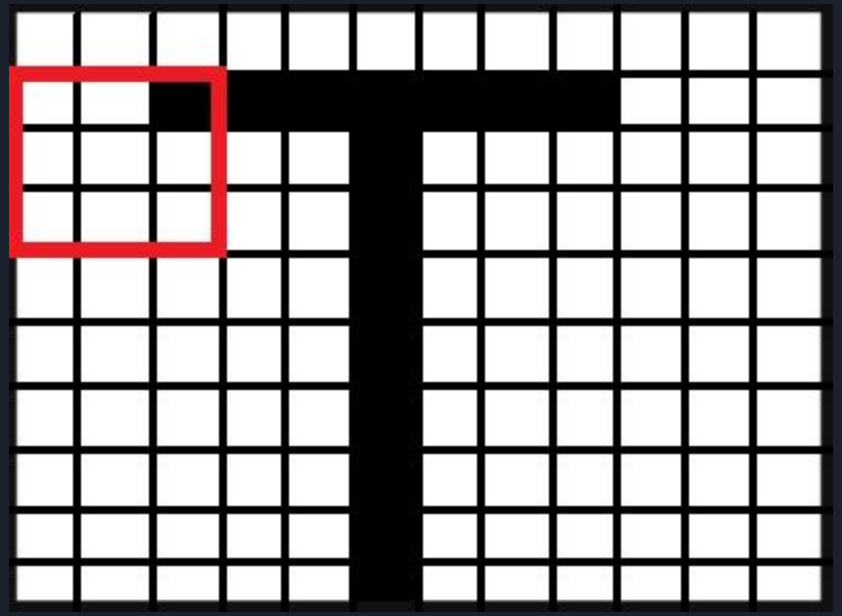
Filtro de detecção de borda



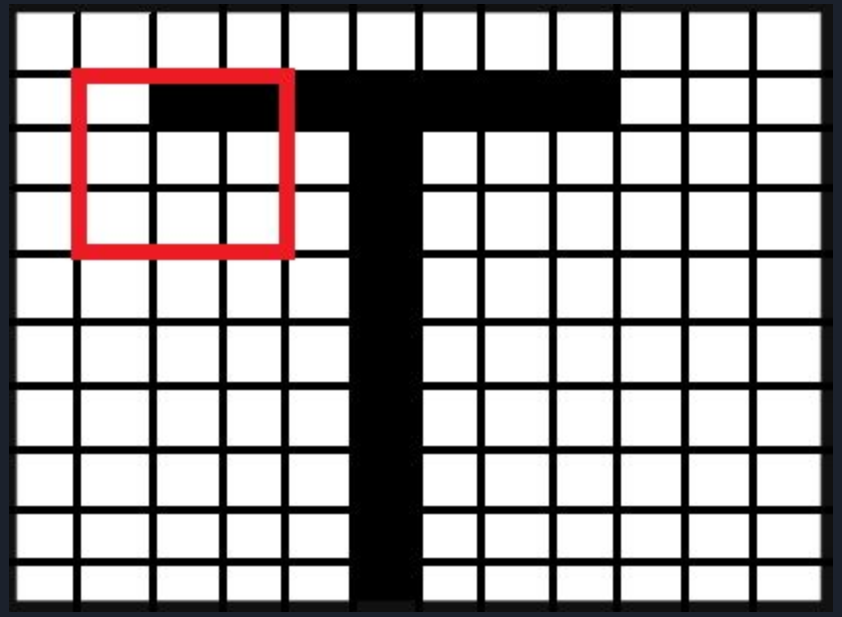
Filtro de detecção de borda



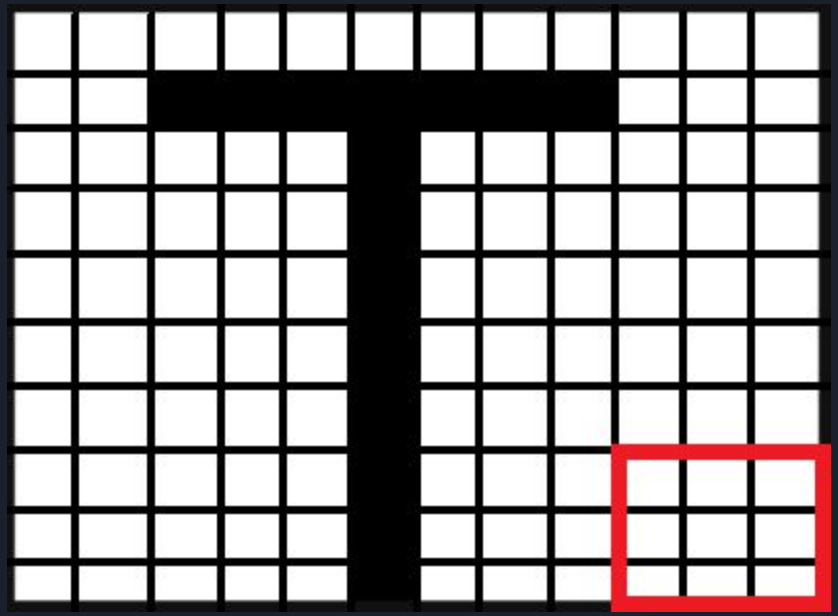
Filtro de detecção de borda



Filtro de detecção de borda

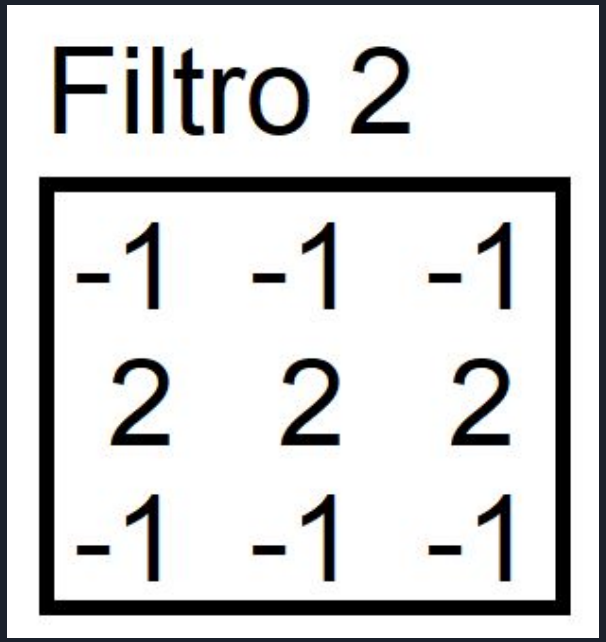


Filtro de detecção de borda

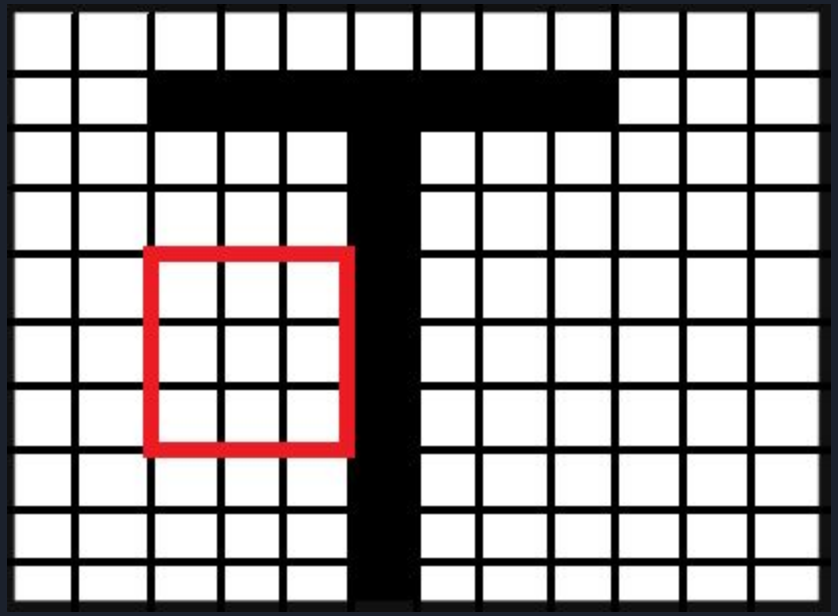


Exemplo

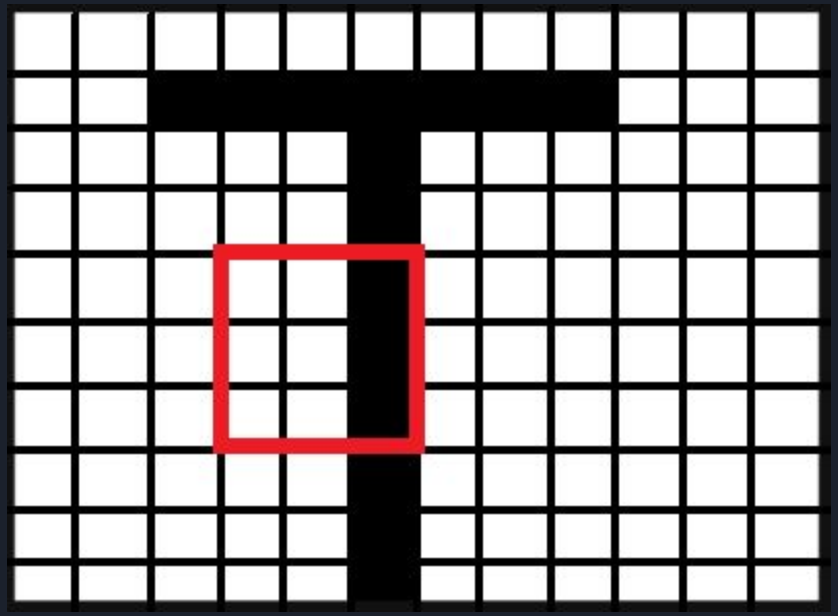
Filtro de detecção de borda



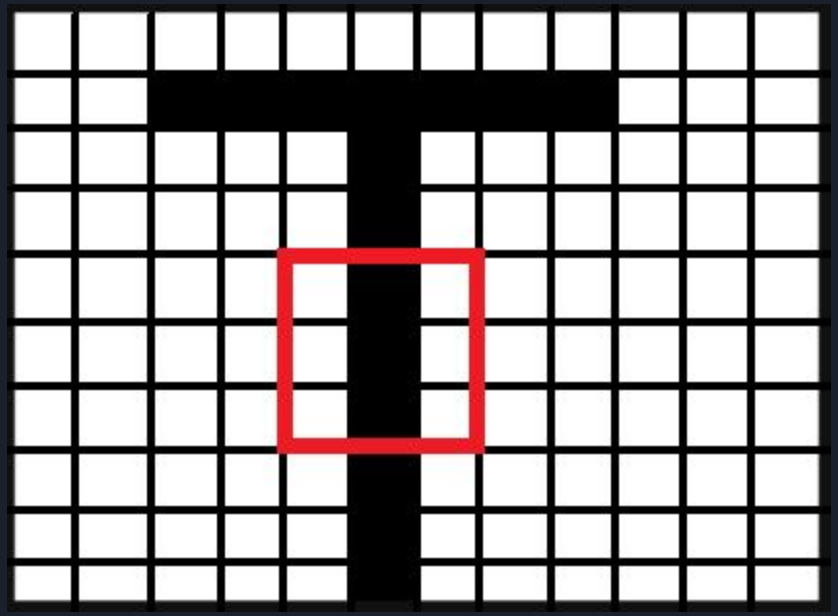
Filtro de detecção de borda



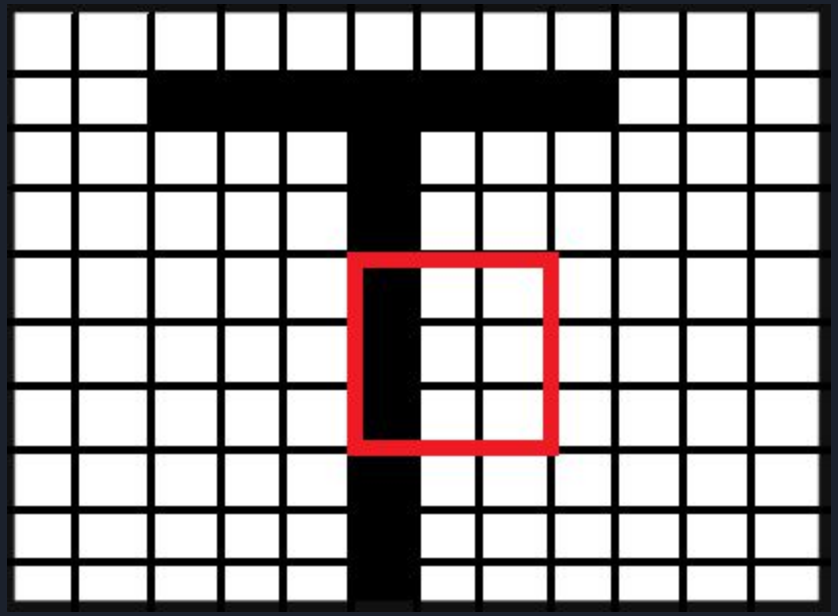
Filtro de detecção de borda



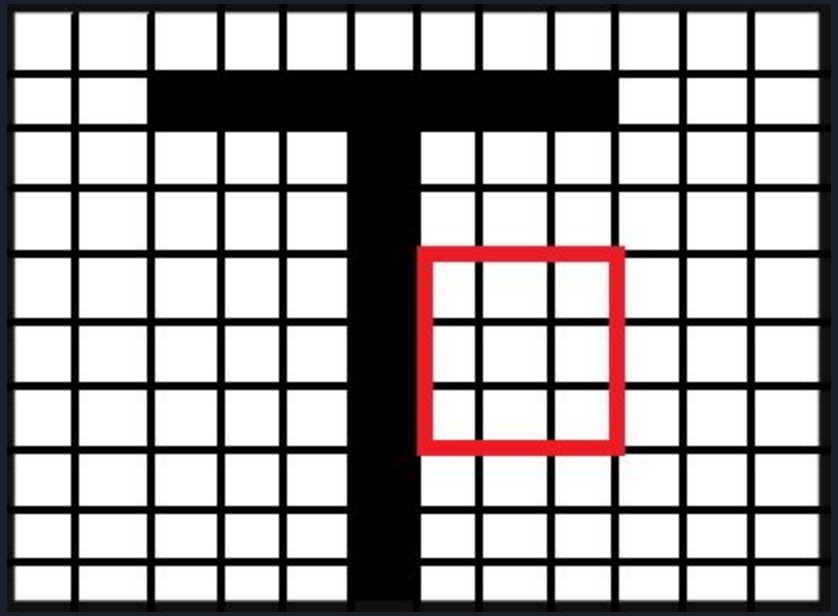
Filtro de detecção de borda



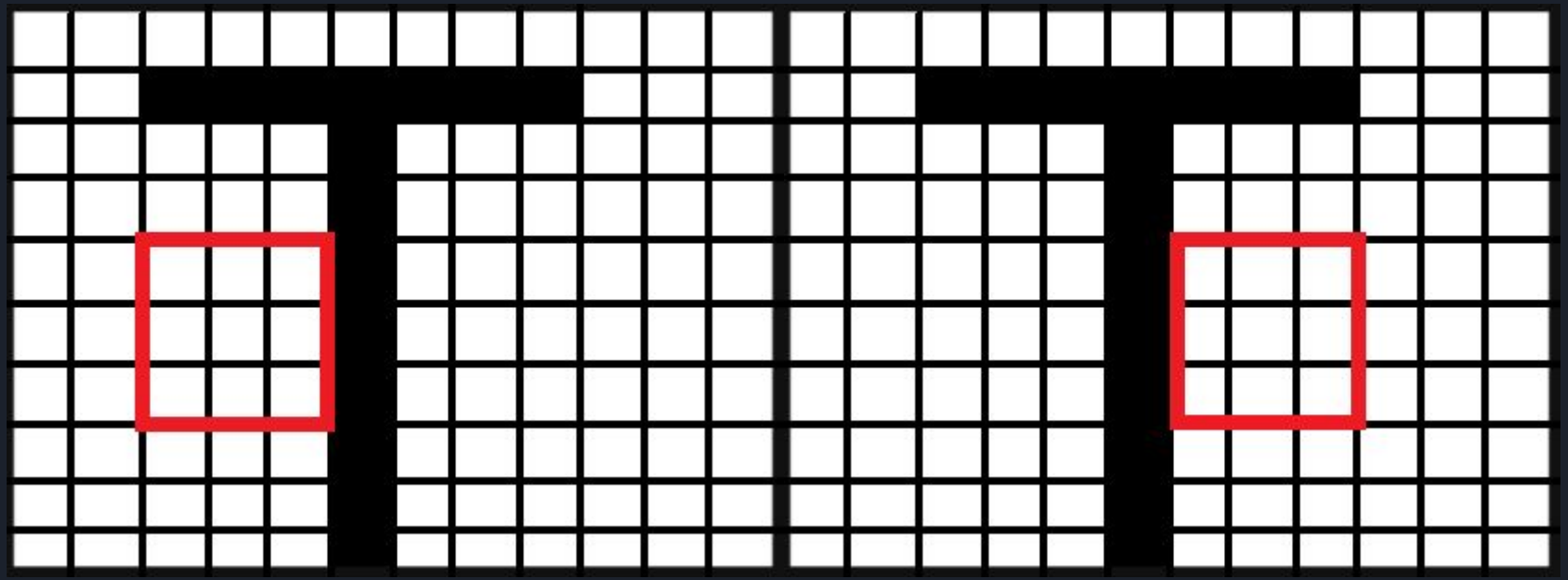
Filtro de detecção de borda



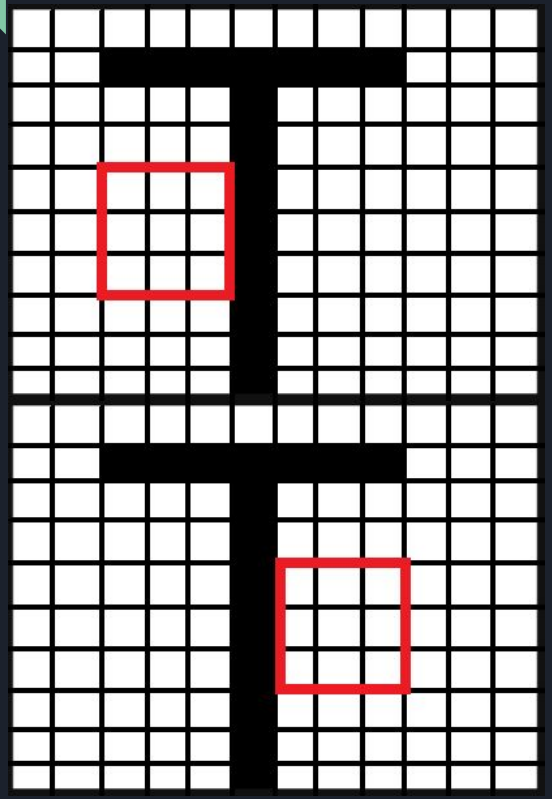
Filtro de detecção de borda



Filtro de detecção de borda



Filtro de detecção de borda



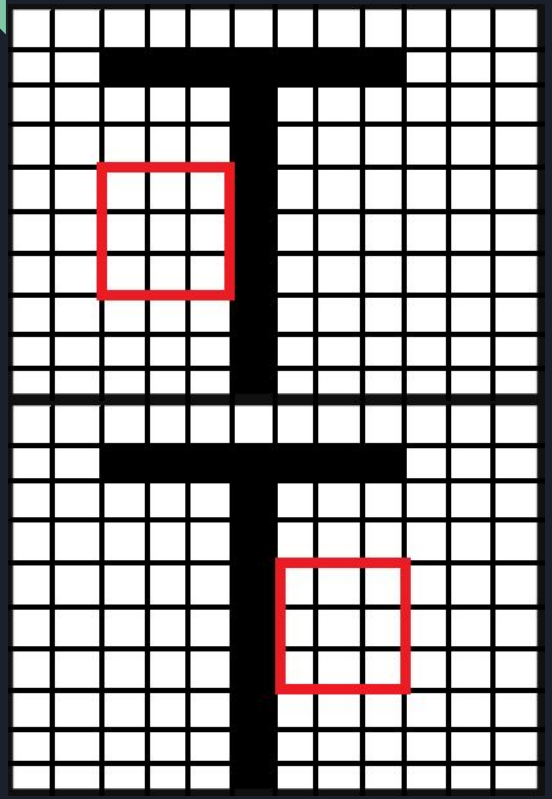
Filtro 1

-1	2	-1
-1	2	-1
-1	2	-1

Filtro 2

-1	-1	-1
2	2	2
-1	-1	-1

Filtro de detecção de borda



Filtro 1

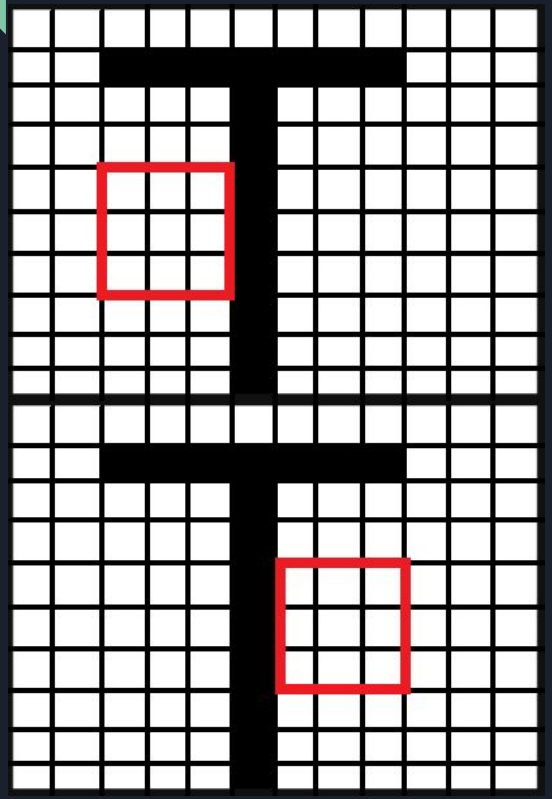
-1	2	-1
-1	2	-1
-1	2	-1

Filtro 2

-1	-1	-1
2	2	2
-1	-1	-1

Resultado = (- 1) * 255 + 2 * 255 + (- 1) * 255 +
(- 1) * 255 + 2 * 255 + (- 1) * 255 +
(- 1) * 255 + 2 * 255 + (- 1) * 255

Filtro de detecção de borda



Filtro 1

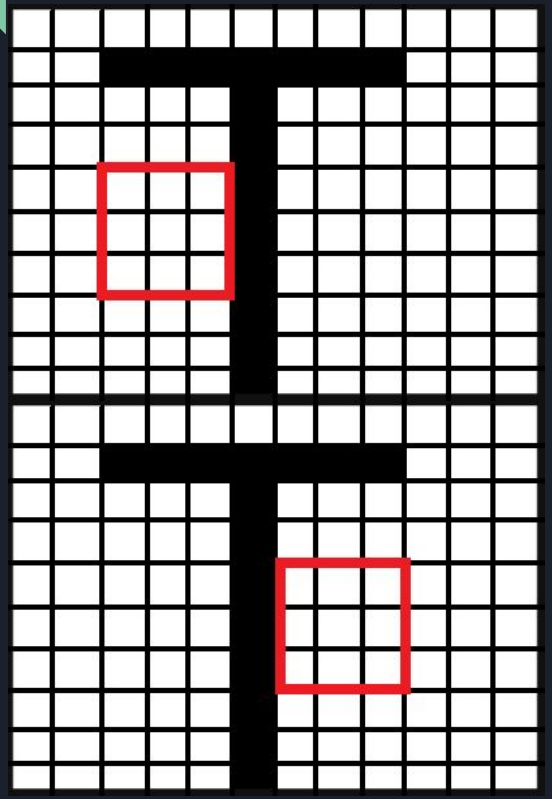
-1	2	-1
-1	2	-1
-1	2	-1

Filtro 2

-1	-1	-1
2	2	2
-1	-1	-1

Resultado = 3*(2)*255 + 6*(-1)*255

Filtro de detecção de borda



Filtro 1

-1	2	-1
-1	2	-1
-1	2	-1

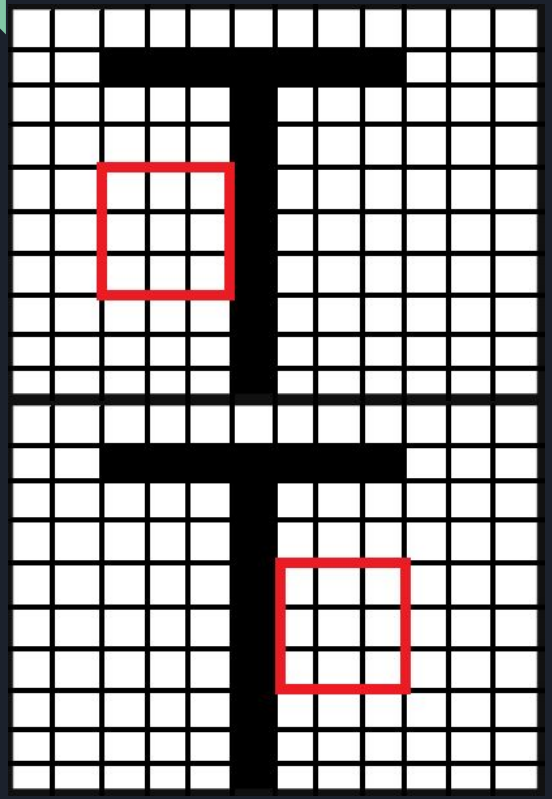
Filtro 2

-1	-1	-1
2	2	2
-1	-1	-1

Resultado = 3*(2)*255 + 6*(-1)*255

= 6*255 - 6*255

Filtro de detecção de borda



Filtro 1

-1	2	-1
-1	2	-1
-1	2	-1

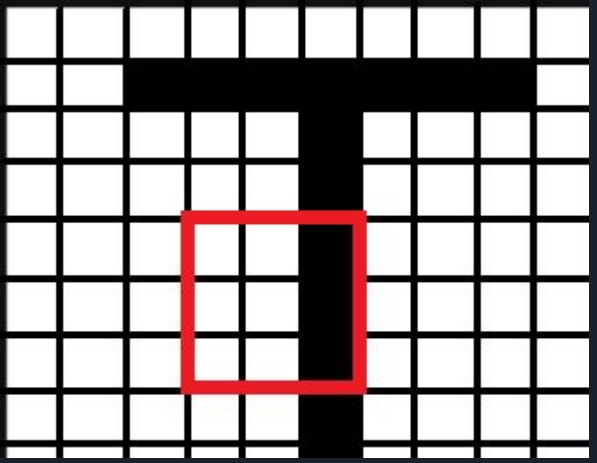
Filtro 2

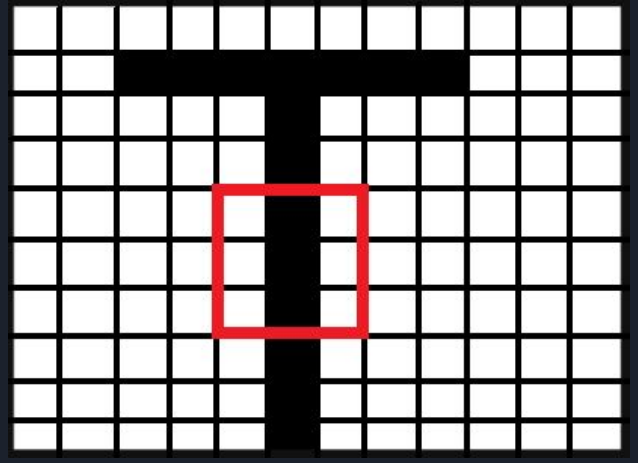
-1	-1	-1
2	2	2
-1	-1	-1

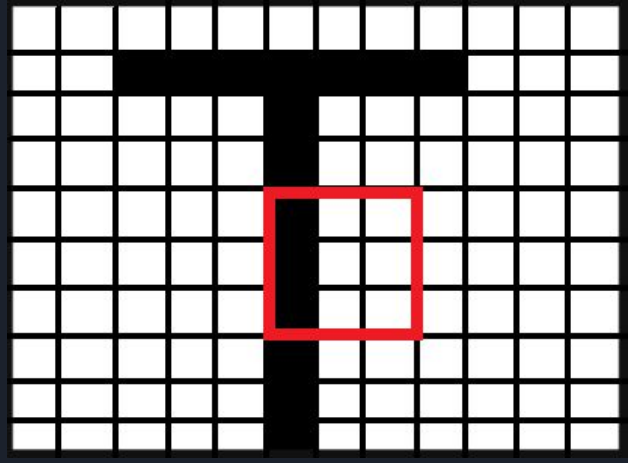
Resultado = 3*(2)*255 + 6*(-1)*255

= 6*255 - 6*255

= 0







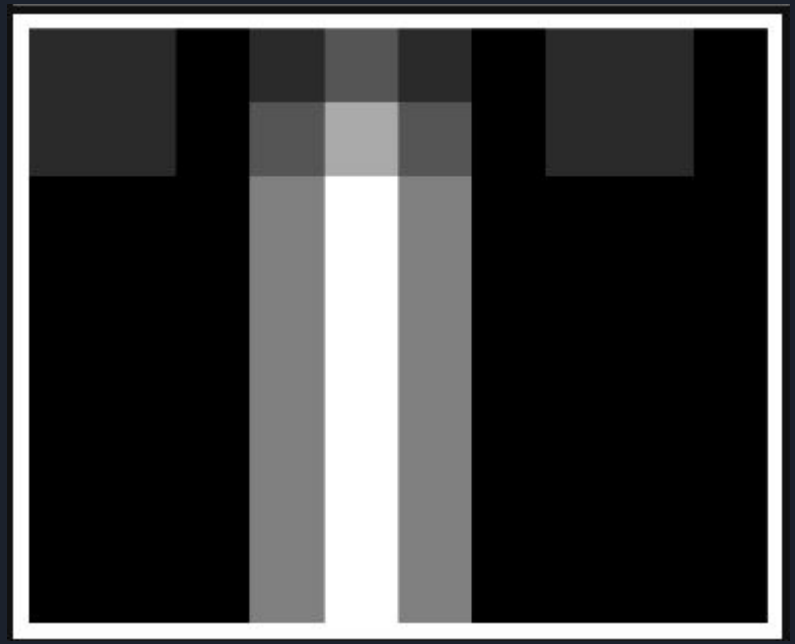
Filtro 1

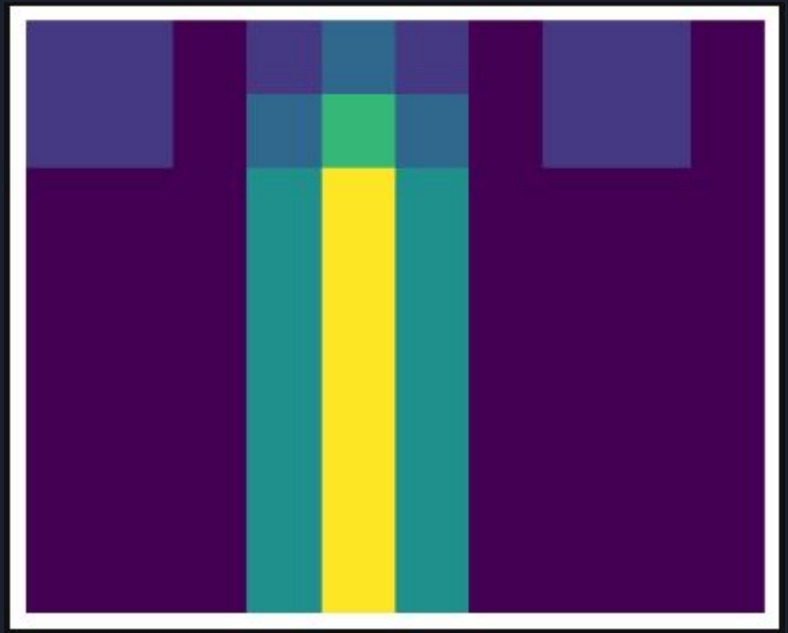
-1	2	-1
-1	2	-1
-1	2	-1

Filtro 2

-1	-1	-1
2	2	2
-1	-1	-1

Filtro de detecção de borda - Filtro 1





Filtro de detecção de borda - Filtro 2

Motivação

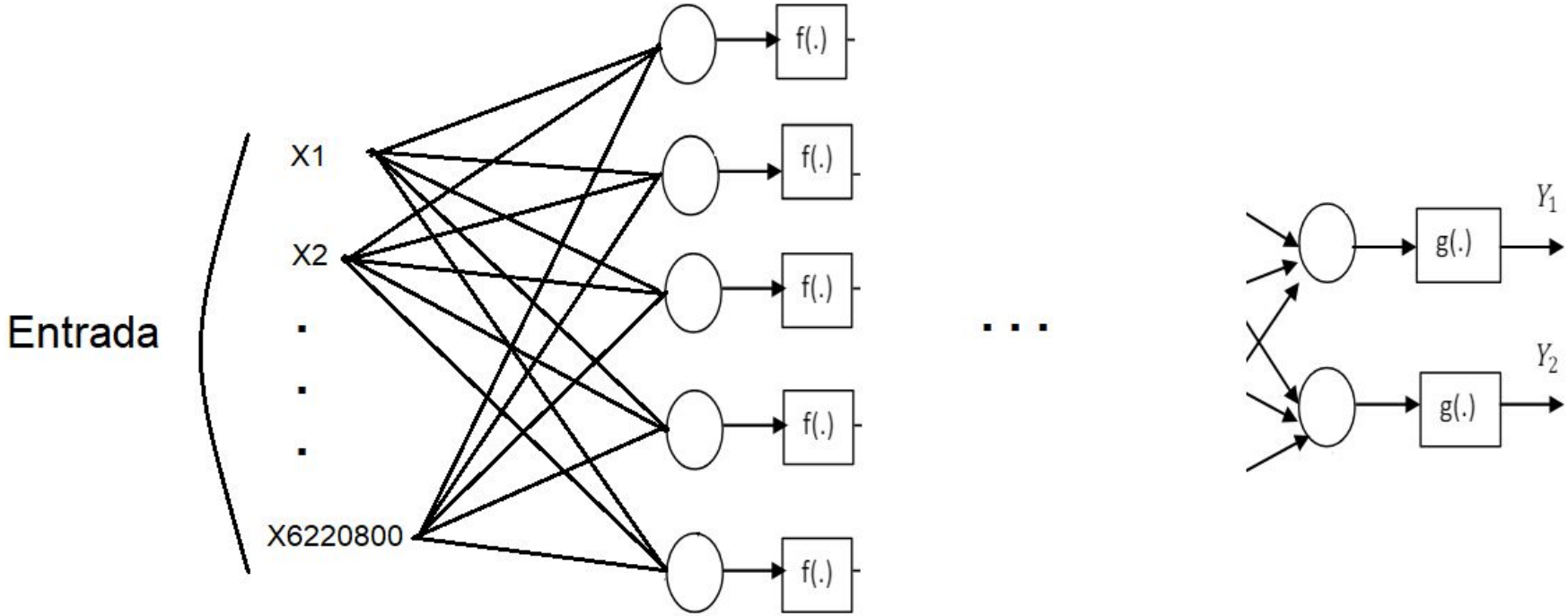
Motivação

- Alta dimensionalidade

Motivação

- Alta dimensionalidade
  - Ex.: Uma imagem full HD possui 1920 x 1080 pixels. Então uma imagem colorida nesta resolução possui 1920 x 1080 x 3 valores = 6220800

Alta dimensionalidade



Motivação

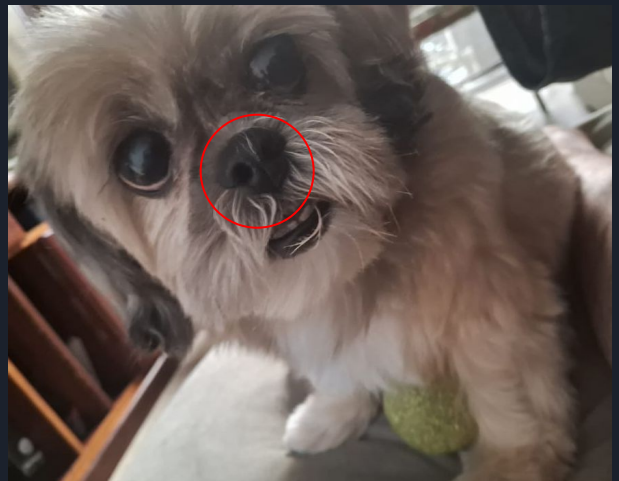
- Alta dimensionalidade
  - Ex.: Uma imagem full HD possui 1920 x 1080 pixels. Então uma imagem colorida nesta resolução possui 1920 x 1080 x 3 valores = 6220800 valores
- Alguns padrões estão limitados à pequenas regiões da imagem
- O mesmo padrão pode ocorrer em diferentes locais da imagem

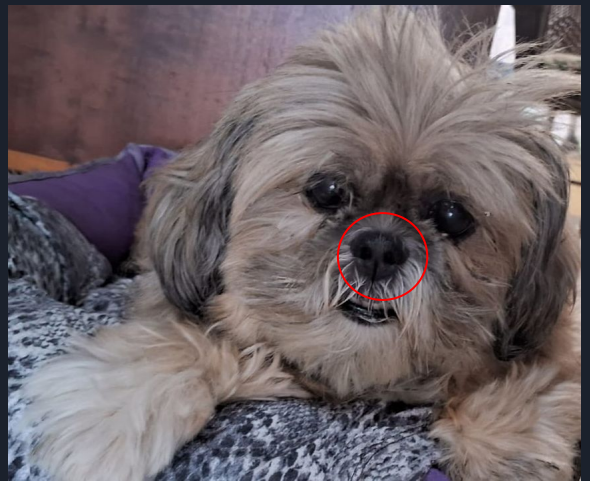
Motivação

- Alta dimensionalidade
  - Ex.: Uma imagem full HD possui 1920 x 1080 pixels. Então uma imagem colorida nesta resolução possui 1920 x 1080 x 3 valores = 6220800 valores
- Alguns padrões estão limitados à pequenas regiões da imagem
- O mesmo padrão pode ocorrer em diferentes locais da imagem

Motivação

- Alta dimensionalidade
  - Ex.: Uma imagem full HD possui 1920 x 1080 pixels. Então uma imagem colorida nesta resolução possui 1920 x 1080 x 3 valores = 6220800 valores
- Alguns padrões estão limitados à pequenas regiões da imagem
- O mesmo padrão pode ocorrer em diferentes locais da imagem





Introdução à Rede Neural Convolucional (CNN- Convolutional neural network)

Introdução à Rede Neural Convolucional (CNN- Convolutional neural network)

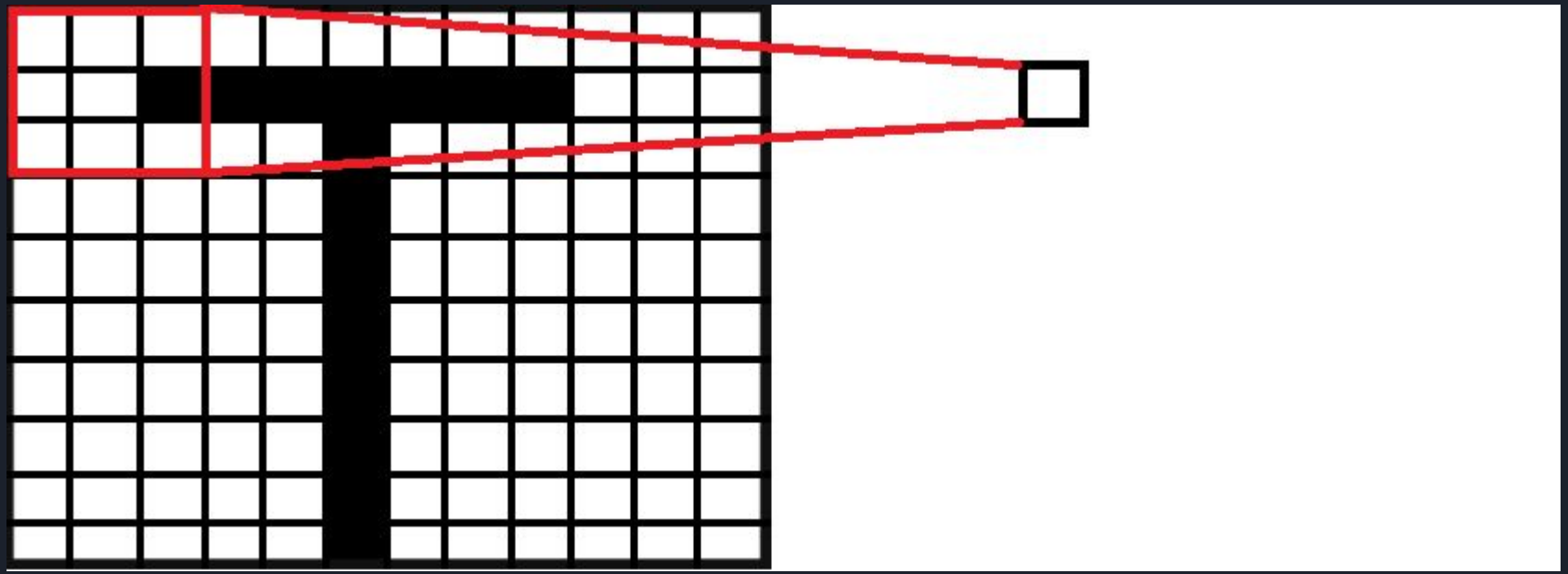
- As CNNs utilizam a operação matemática de convolução
- Possuem uma ou várias camadas que utilizam o operador linear de convolução
- As camadas convolucionais extraem padrões locais
- Facilidade para trabalhar com dados de alta dimensionalidade
- Diferentes tipos de camadas podem ser utilizadas na construção de uma CNN

Camada de convolução

- Quantidade de filtros
- Tamanho do kernel
- Stride
- Padding
- Função de ativação
- bias
- ...

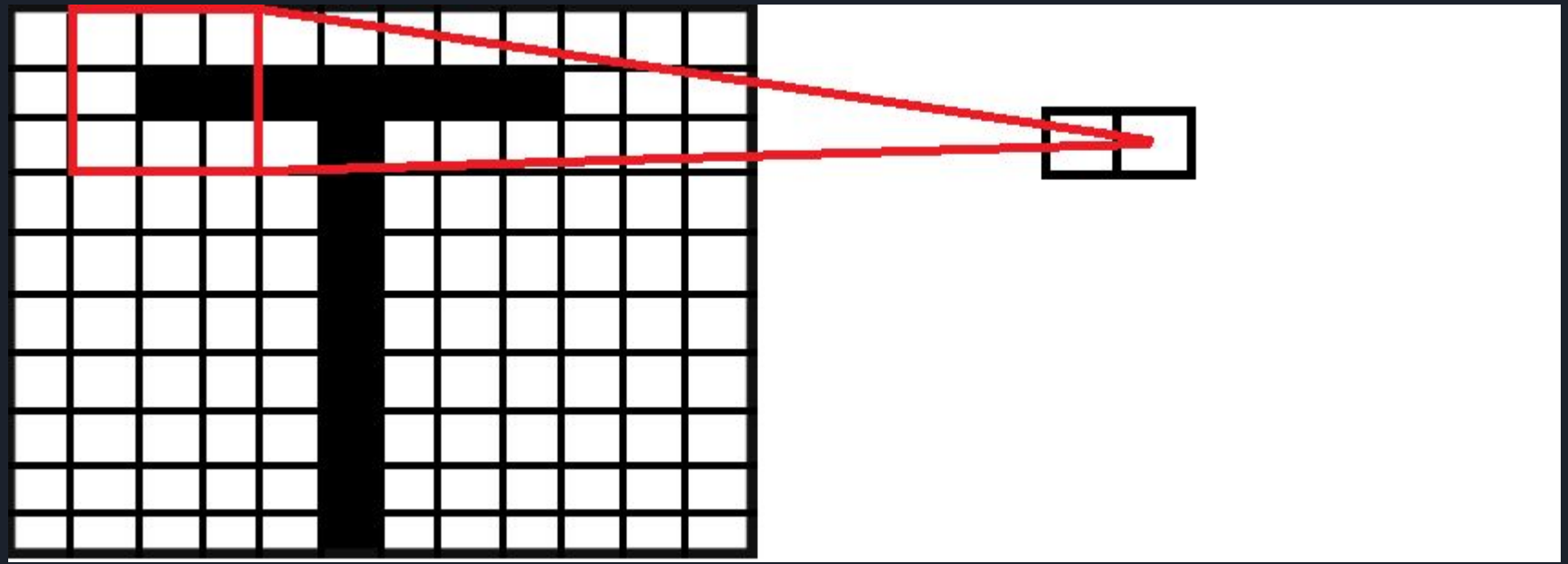
Camada de convolução

- Ex.: 1 filtro de kernel 3x3



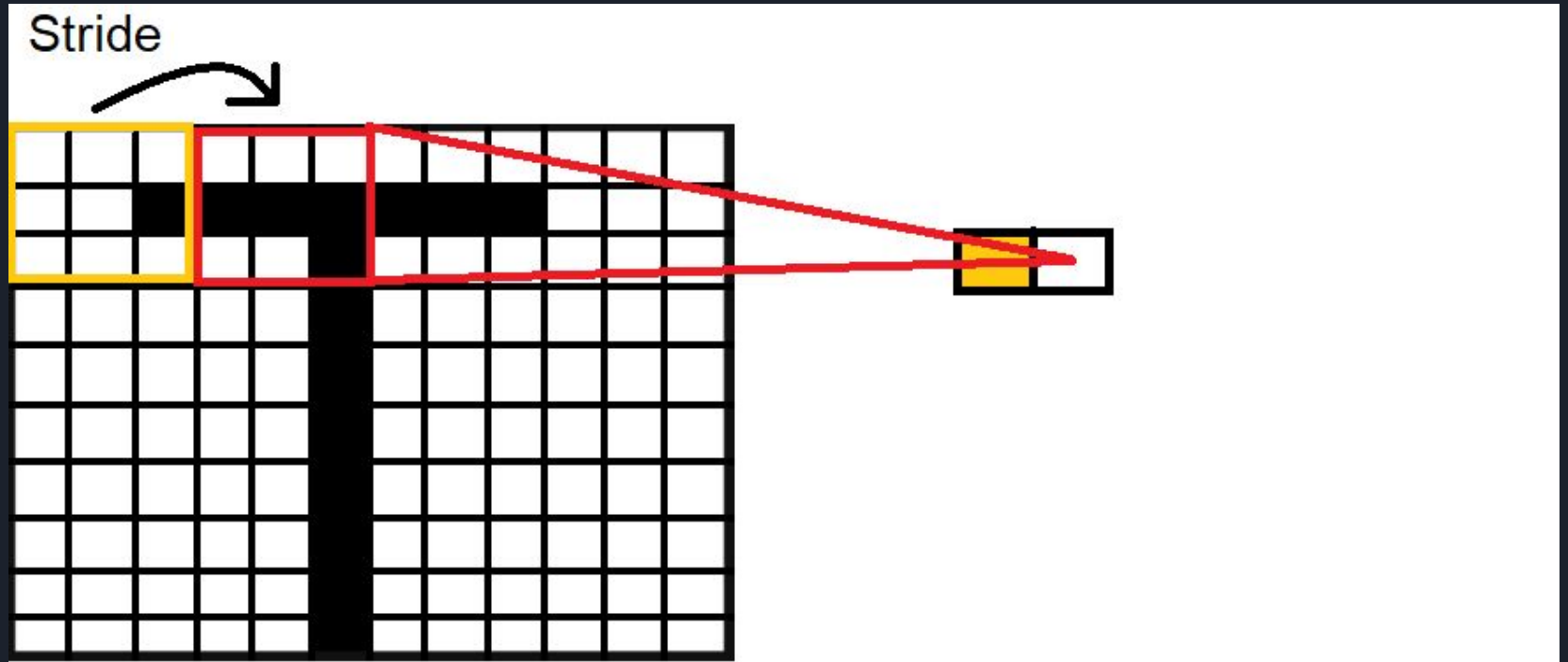
Camada de convolução

- Ex.: 1 filtro de kernel 3x3



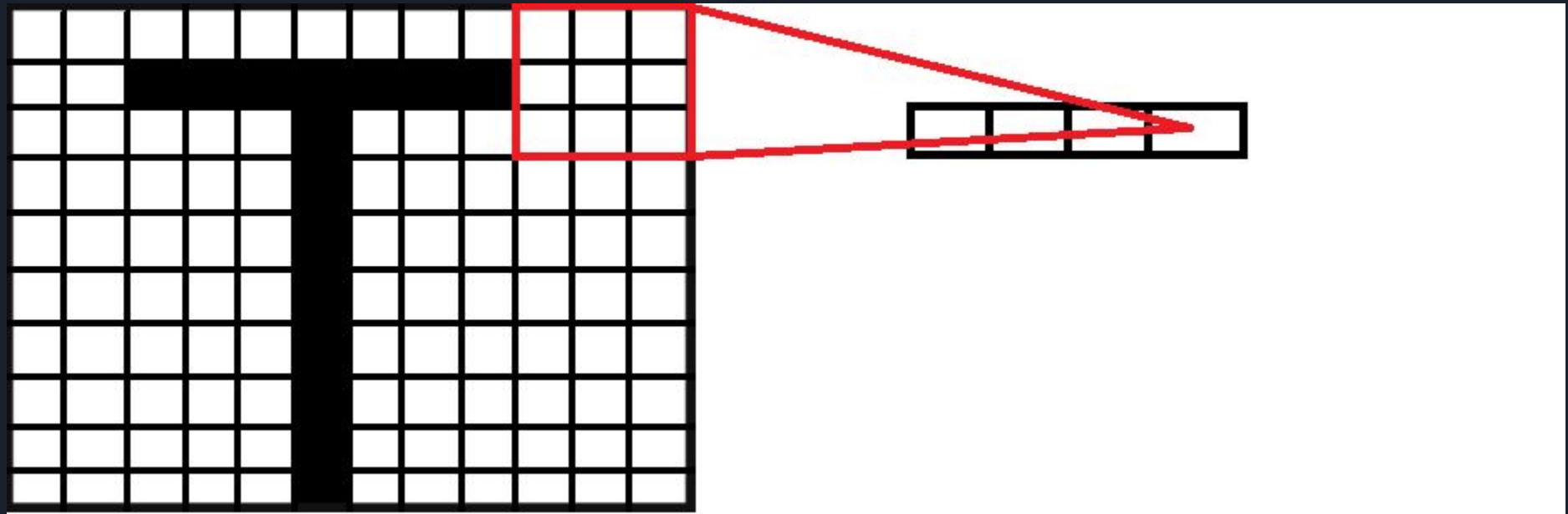
Camada de convolução

- Ex.: Com stride



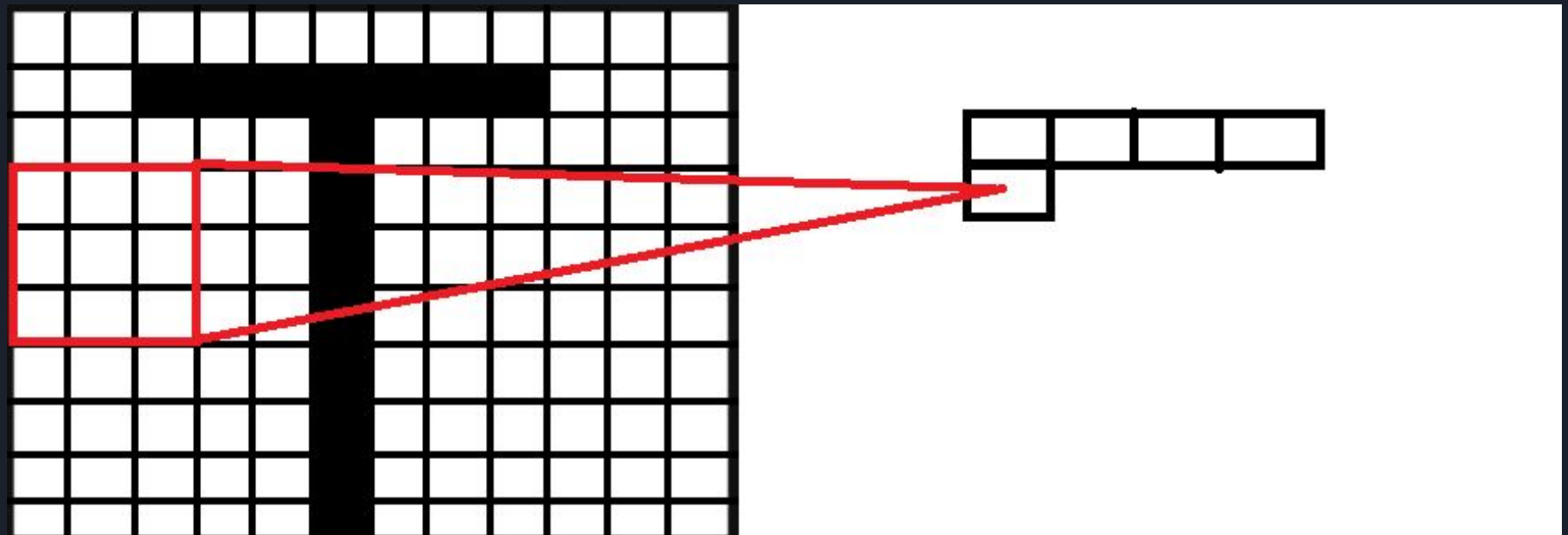
Camada de convolução

Continuando o exemplo...



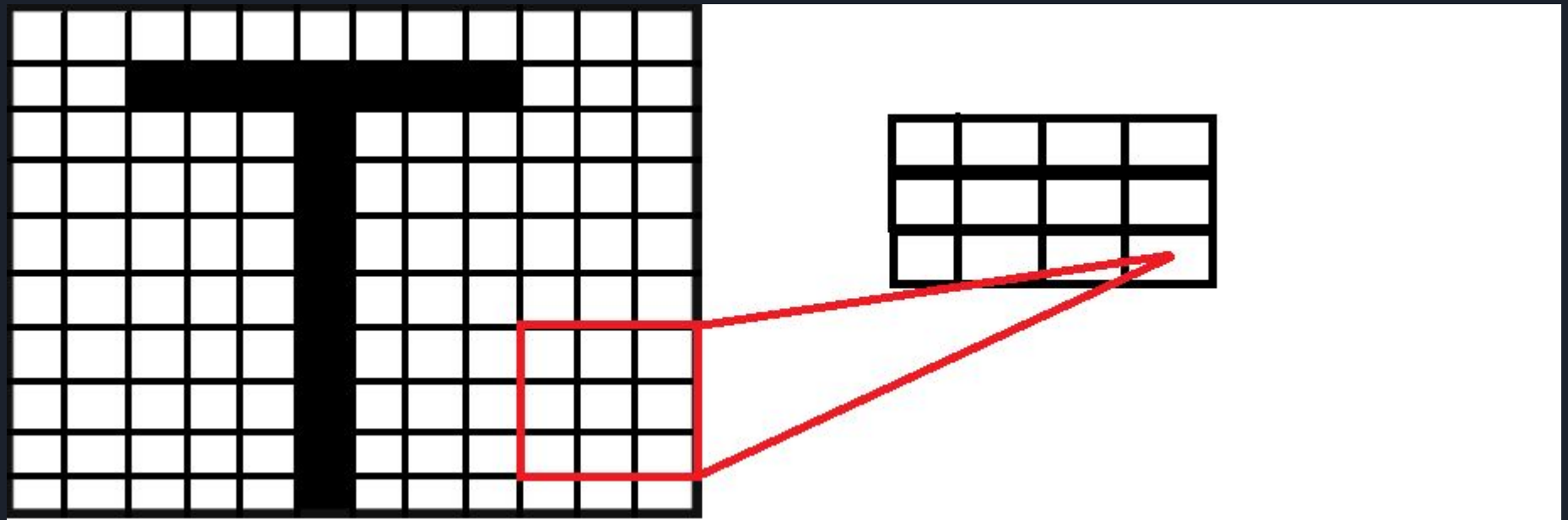
Camada de convolução

Continuando o exemplo...



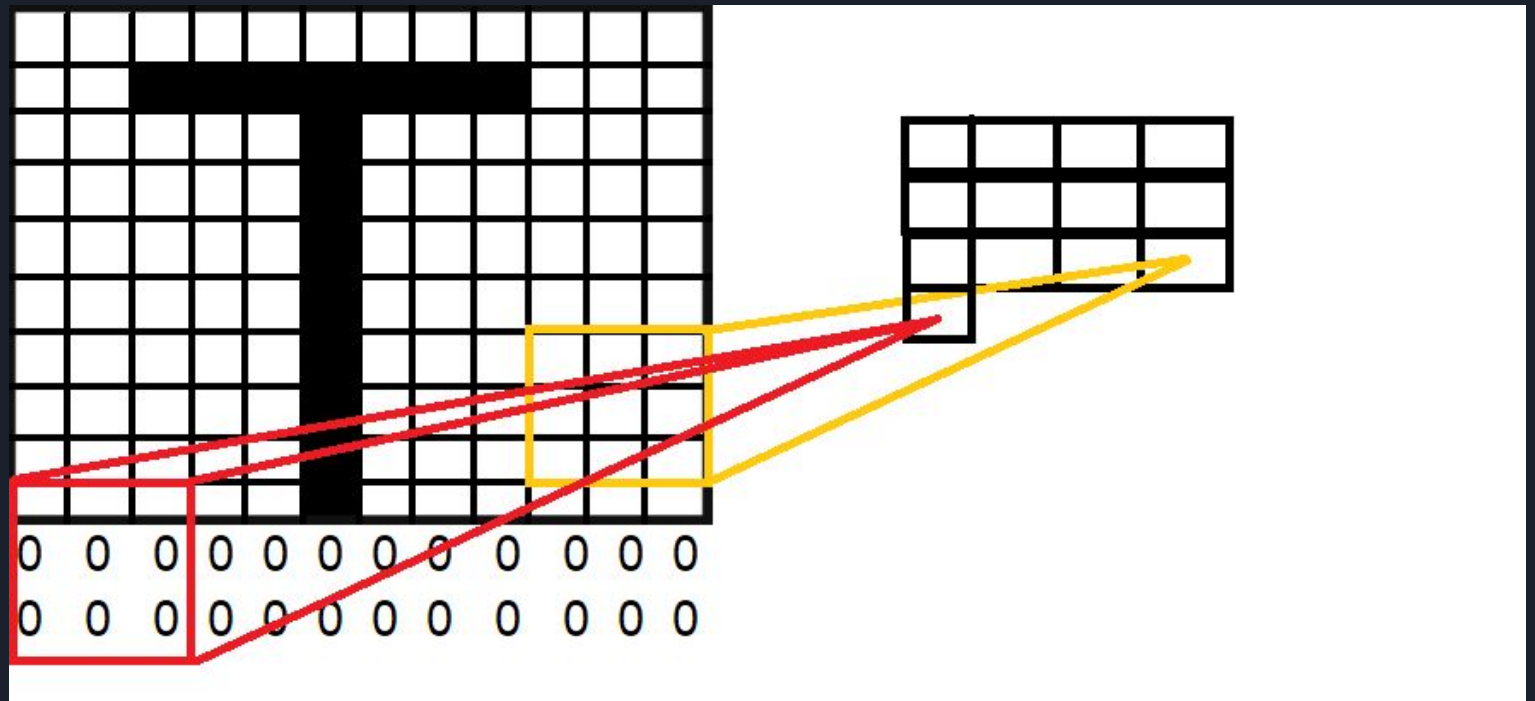
Camada de convolução

Continuando o exemplo...



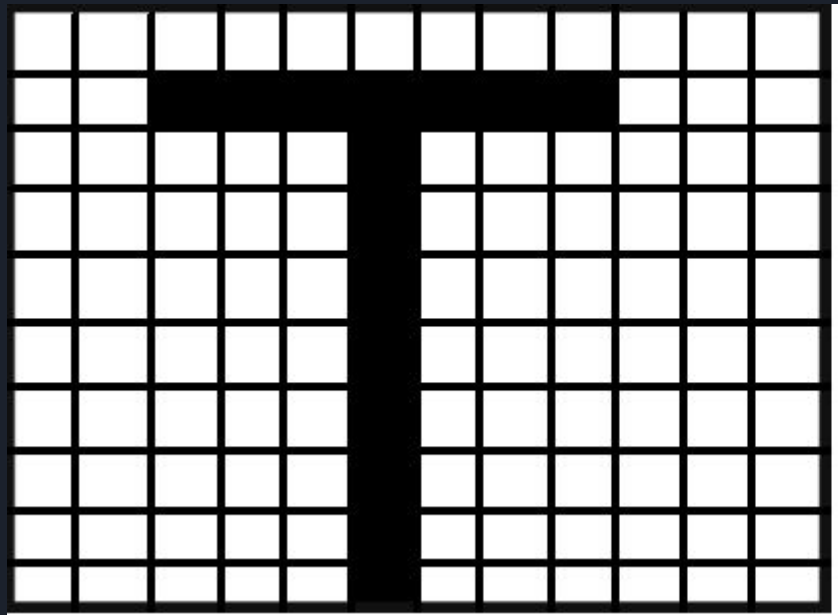
Camada de convolução

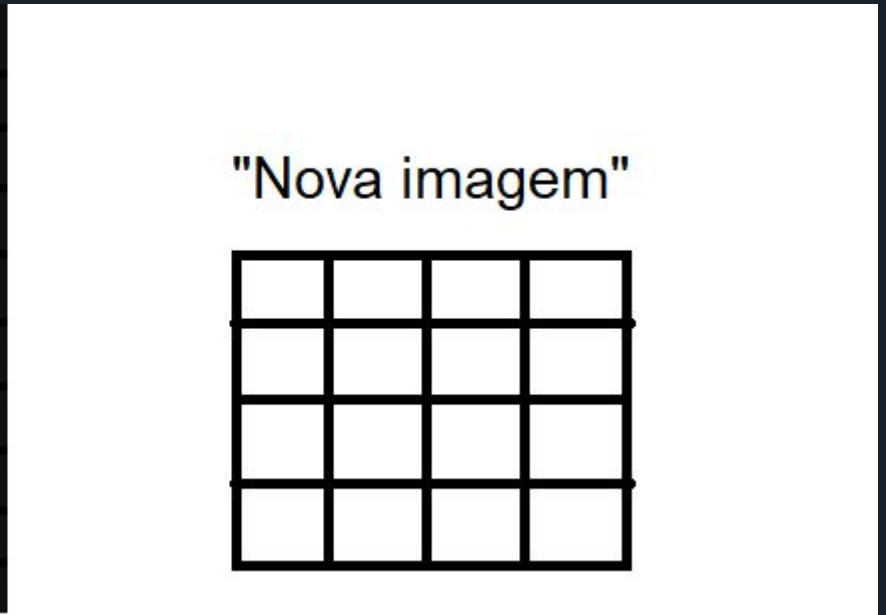
Com padding



Camada de convolução

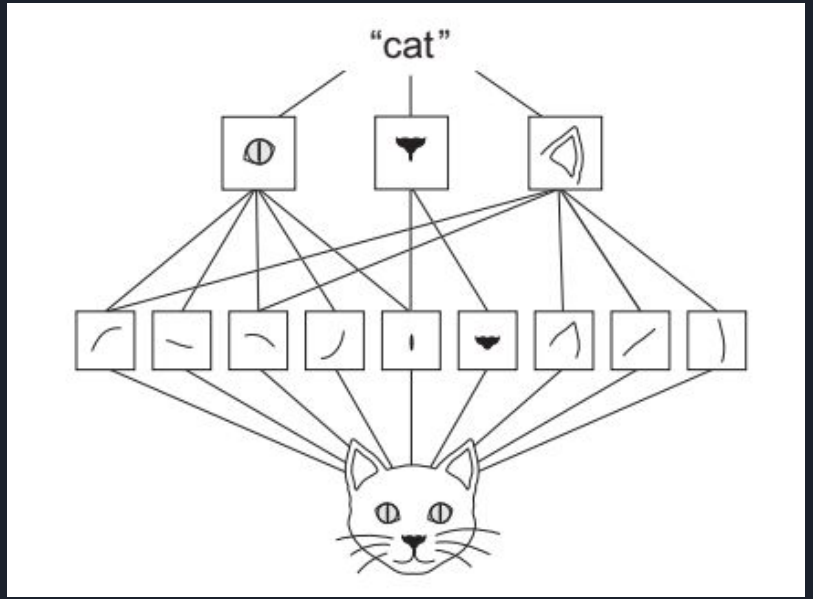
Novas camadas podem ser adicionadas à “imagem” obtida.





Camada de convolução

- Compartilhamento de pesos.
- As primeiras camadas extraem informações mais simples das imagens. Ex.: Bordas de diferentes formas e ângulos.
- Características mais complexas são extraídas a cada nova camada adicionada, aumentando o nível de abstração. Ex.: Identificação de olhos, nariz, boca, texturas…



Fonte: CHOLLET, F.Deep Learning with Python. 1st. ed. Greenwich, CT, USA: ManningPublications Co., 2017.

Camada de pooling

- Diminui o tamanho espacial das imagens resultantes.
- Não possui treinamento associado ao pooling (Nenhum peso é modificado)
- Reduz a quantidade de parâmetros
- Ajuda a controlar o sobre ajuste
- Diferentes funções podem ser utilizadas:
  - Pooling máximo
  - Pooling médio
  - Pooling norma L2
  - ...

Camada Flatten

Ordena toda a informação obtida em um vetor unidimensional

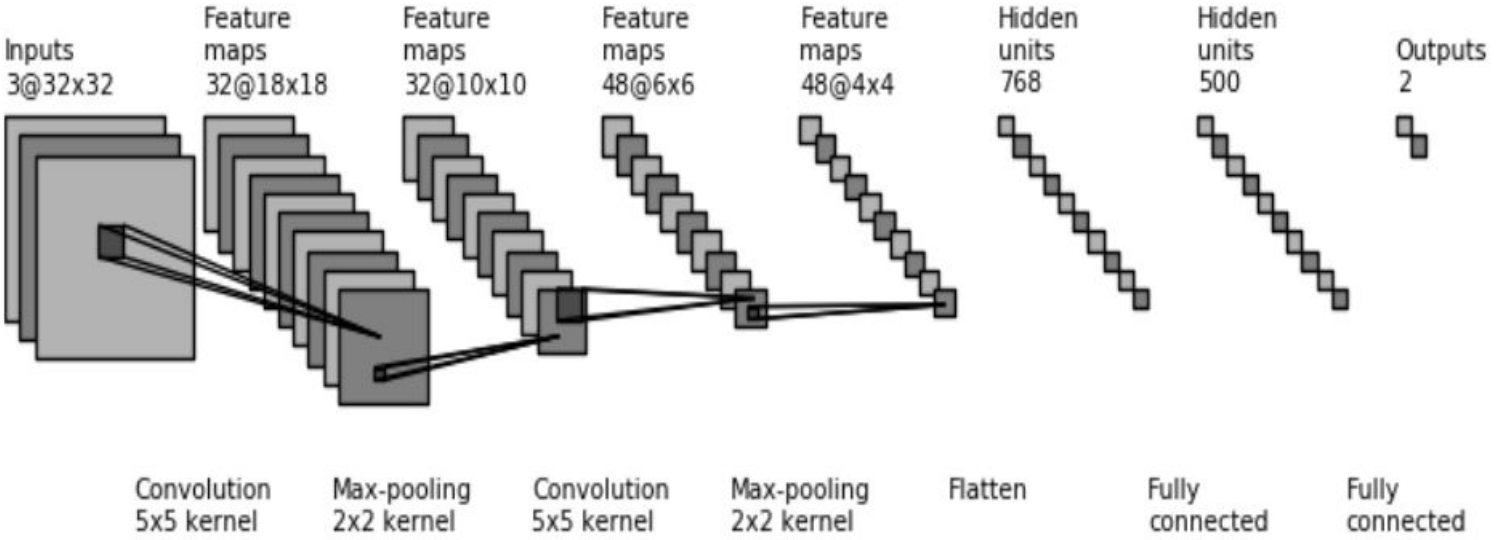
Dropout

Técnica de regularização que desativa aleatoriamente um conjunto de neurônios. (Zera a saída do neurônio)

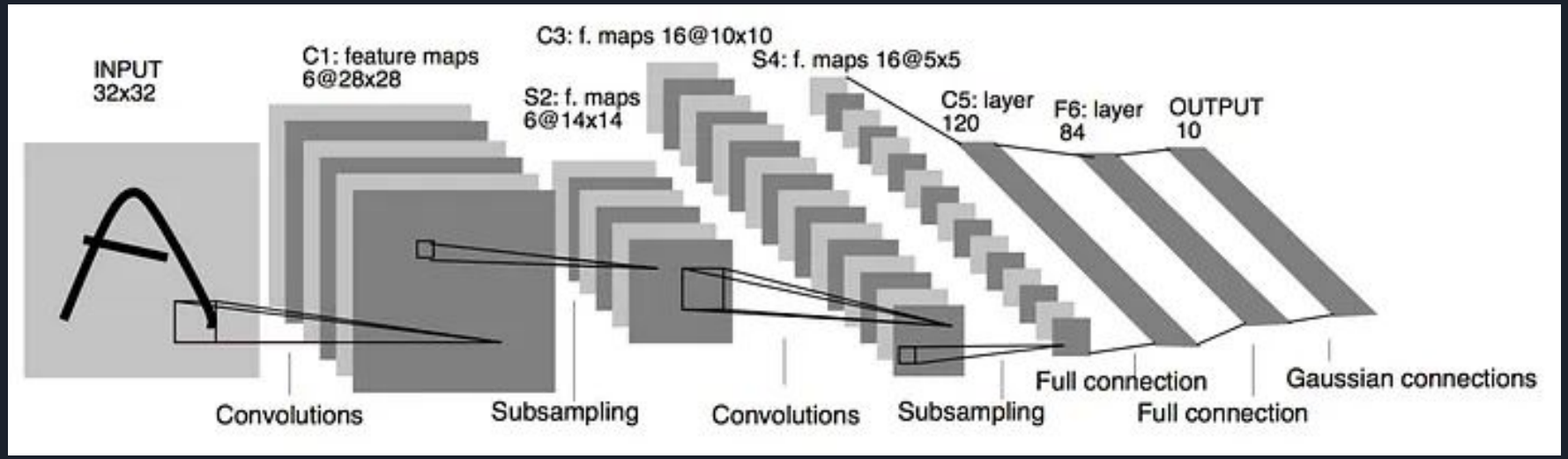
- Melhor robustez do modelo
- Previne sobre ajuste
- Melhora a capacidade de generalização do aprendizado
- Também pode ser aplicado aos dados de entrada

Exemplos de CNNs

Exemplo


Inputs
3@32x32
Feature
maps
32@18x18
Feature
maps
32@10x10
Feature
maps
48@6x6
Feature
maps
48@4x4
Hidden
units
768
Hidden
units
500
Outputs
2
Convolution
5x5 kernel
Max-pooling
2x2 kernel
Convolution
5x5 kernel
Max-pooling
2x2 kernel
Flatten
Fully
connected
Fully
connected

Exemplo



LeNet5 Network architecture. Extraído de https://medium.com/neuronio/understanding-convnets-cnn-712f2afe4dd3

Controle de Veículo Autônomo Utilizando Inteligência Artificial

Contexto: Controle de Veículo Autônomo

- Grandes investimentos das empresas no desenvolvimento da tecnologia.
- Redução de acidentes por erros humanos.
- Maior conforto aos motoristas.
- Redução do estresse.
- Aumento do poder computacional

Controle de Veículo Autônomo

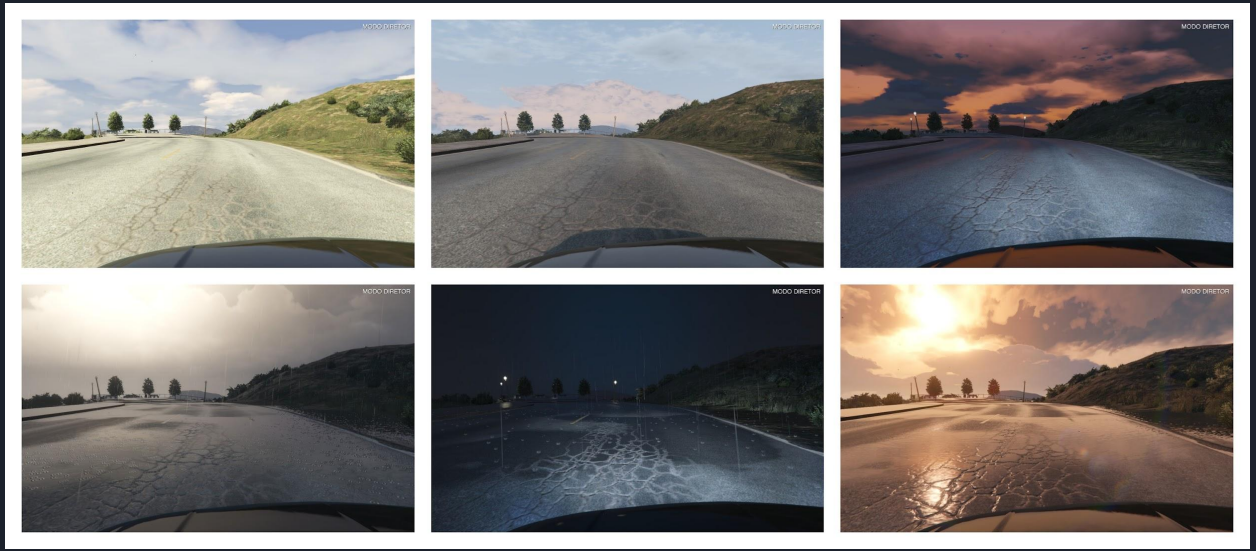
- Ambiente de Simulação.
- Redes Neurais.
- Visão Computacional.
- Aprendizado end-to-end.

Objetivo

- Controlar veículo em ambiente de simulação utilizando câmera como sensor.
- Pista de aproximadamente 1km sem trânsito ou pedestres.
- Não colidir com nenhum elemento da pista.
- Completar o trajeto sem intervenção externa.

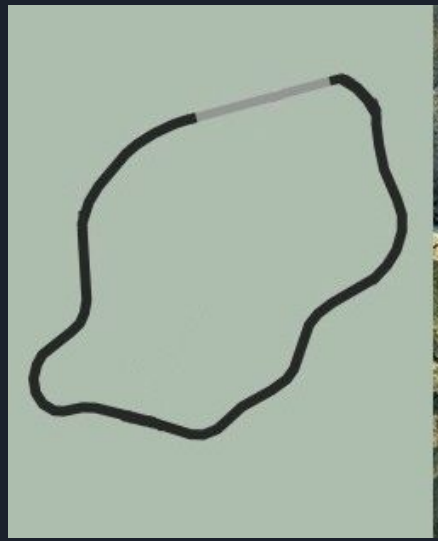
Ambiente de Simulação

- Jogo comercial Grand Theft Auto V.
- Gráficos realistas.
- Diversidade de elementos visuais.
- Possibilidade de controlar climas e horas do dia.

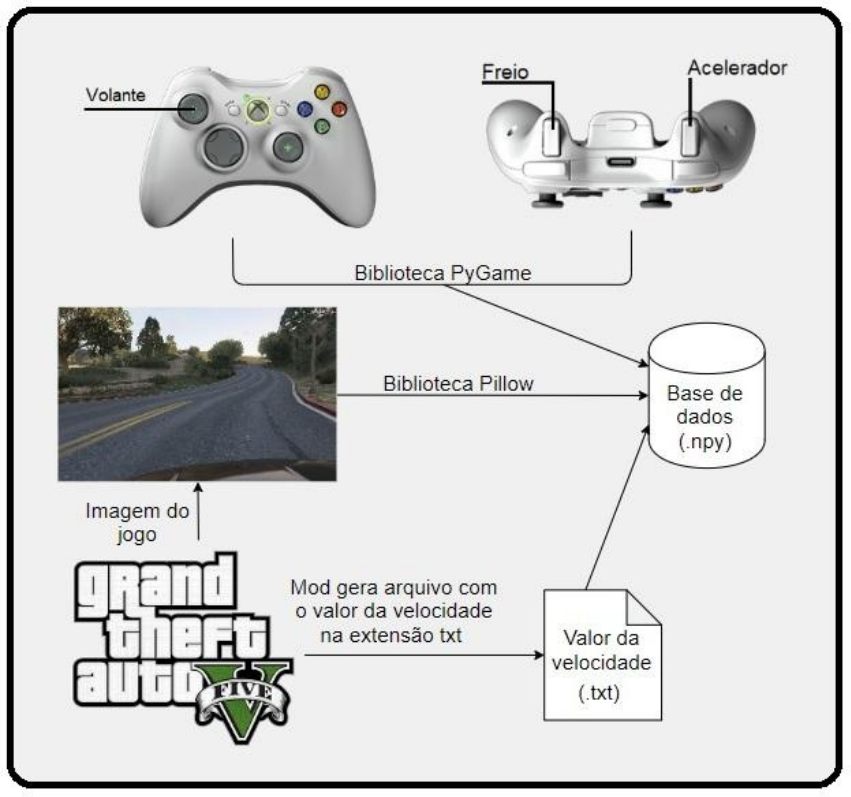


Pista

- Pista com aclives, declives.
- Curvas.
- Trecho com túnel.

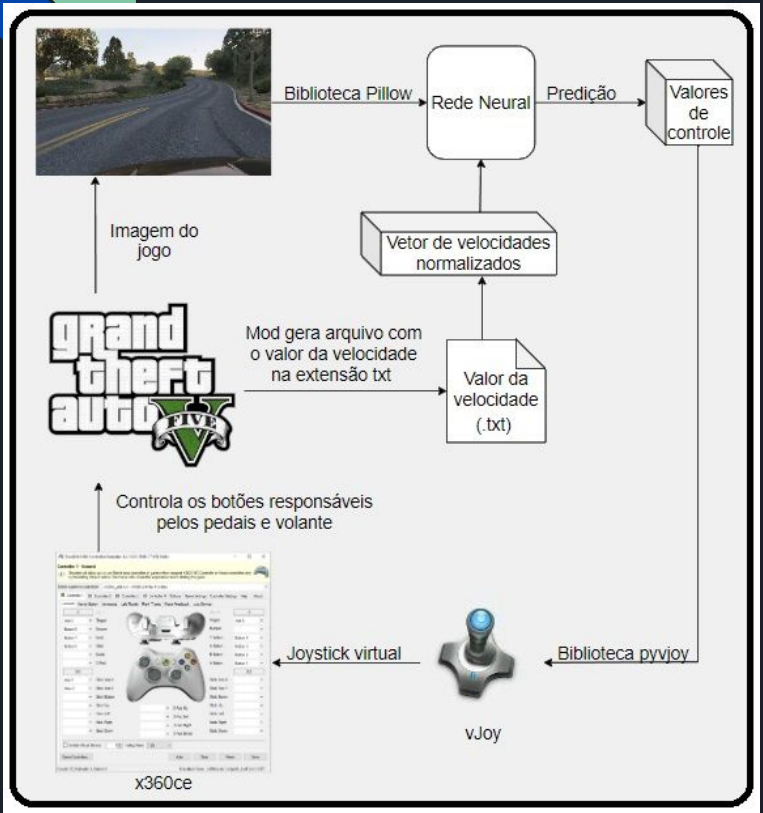


Interface de Coleta de Dados



- Dados salvos em vetores numpy:
  - Imagem (240x150x3).
  - Velocidade.
  - Pedal do acelerador.
  - Pedal do freio.
  - Ângulo do volante.

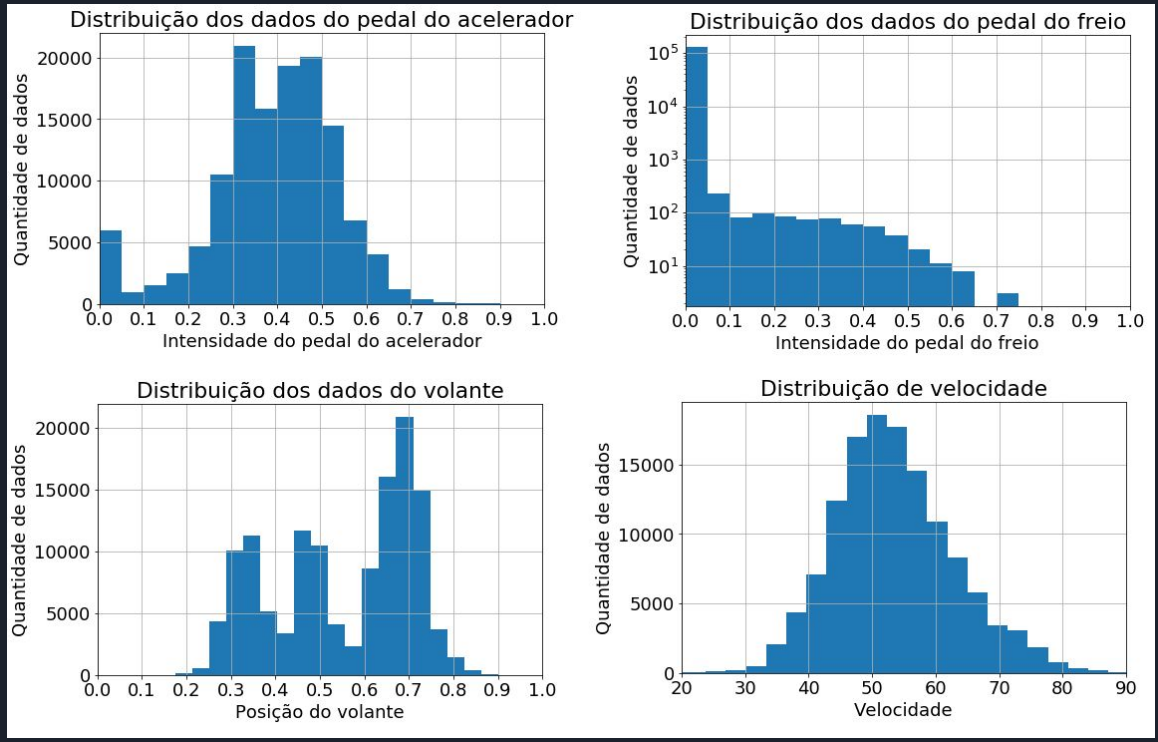
Interface de Controle



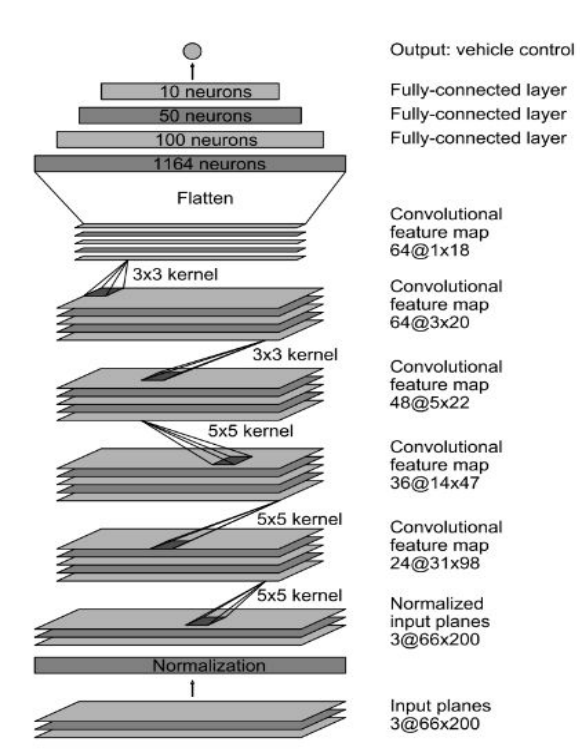
- Controle do veículo
  - Taxa de atuação de 10Hz.
  - Aquisição da imagem e velocidade.
  - Ordenar e normaliza valores de velocidades.
  - Rede neural calcula os valores de direção.
  - Atuação no jogo.

Base de Dados

- 130 mil exemplos coletados separados em três conjuntos:
  - Treino com 78386 dados
  - Validação com 26129 dados
  - Teste com 25000 dados



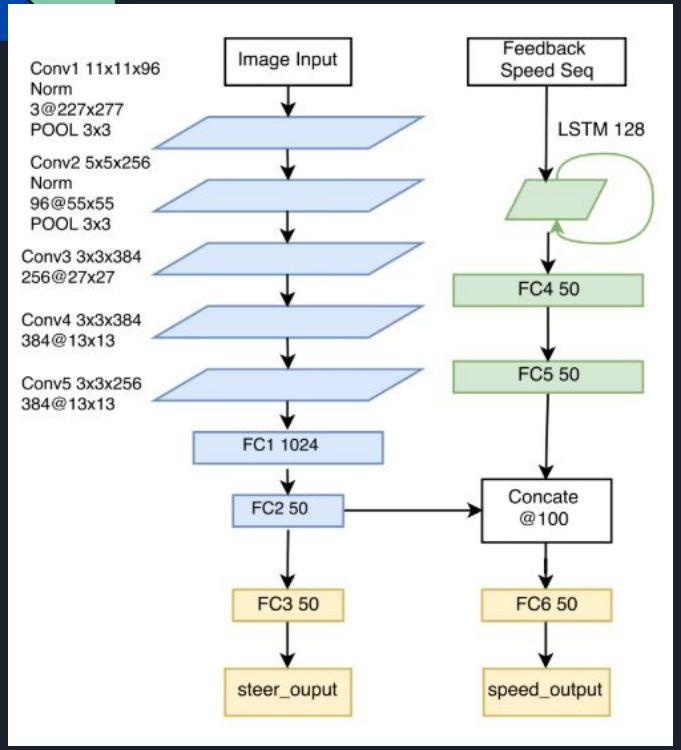
DAVE -2: NVIDIA



- 5 camadas convolucionais e 3 densamente conectadas.
- Entrada: Imagem de câmera.
- Saída: Ângulo de direção.
- Autonomia de 98%.
- Aprendizado end-to-end.

Fonte: BOJARSKI, M. et al. End to end learning for self-driving cars, 2016

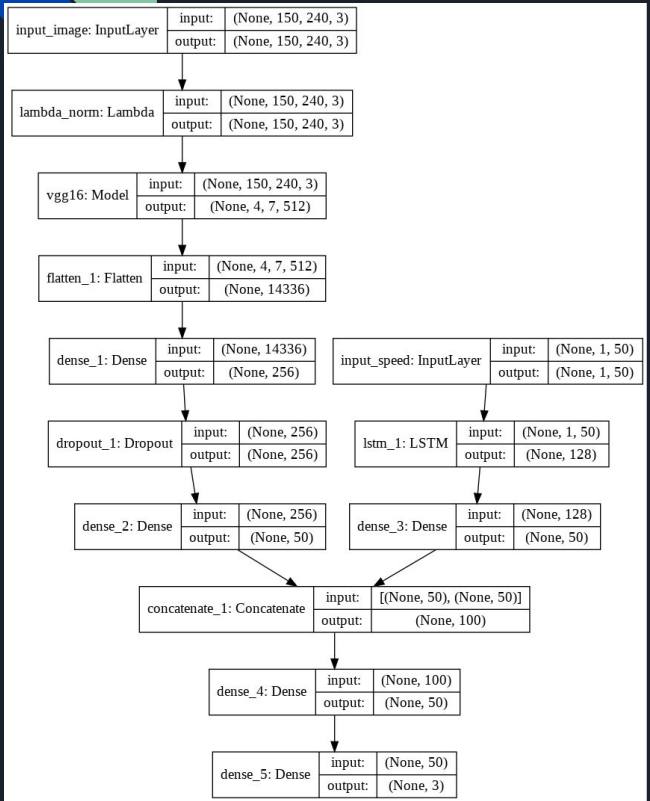
Multi-Modal Multi-Task Network



- Rede neural recorrente com unidade LSTM paralela.
- Concatenação convolucionais + recorrentes.
- Duas entradas: imagens + feedback de velocidades.
- Duas saídas: velocidade e ângulo de direção.
- Melhor controle de direção.

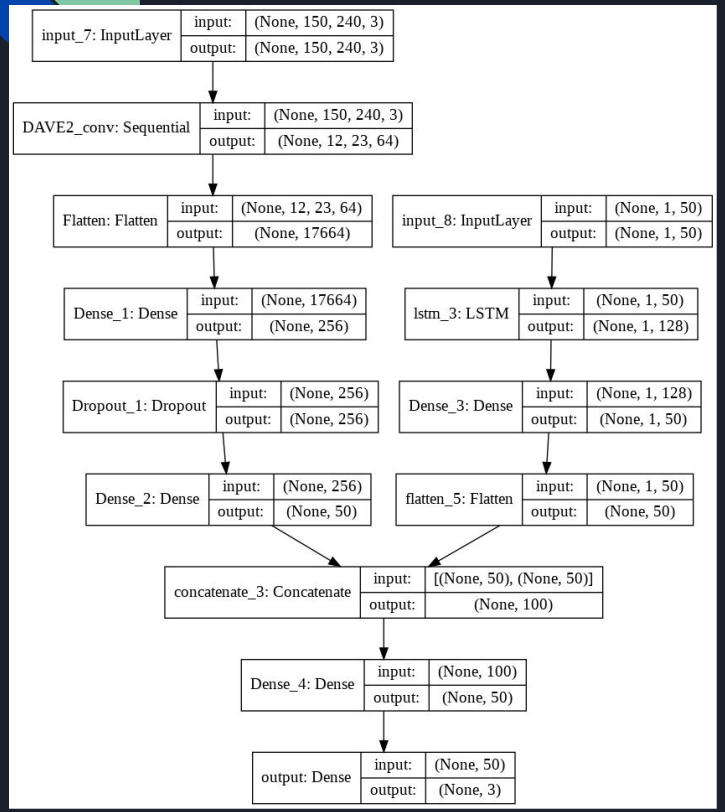
Fonte: YANG, Z. et al. End-to-end multi-modal multi-task vehicle control for self-driving cars with visual perception, 2018.

Redes Neurais Elaboradas



- Redes pré-treinadas: VGG16, MobileNetV2 e Xception.
- Rede LSTM paralela.
- Camada de Dropout para reduzir overfit.
- Concatenação rede recorrente + rede pré-treinada.
- Duas entradas: Imagem + Vetor de velocidades.
- 3 saídas: Ângulo de direção + acelerador + freio.

Redes Neurais Elaboradas



- Modelo baseado na DAVE-2.
- Camada LSTM paralela.

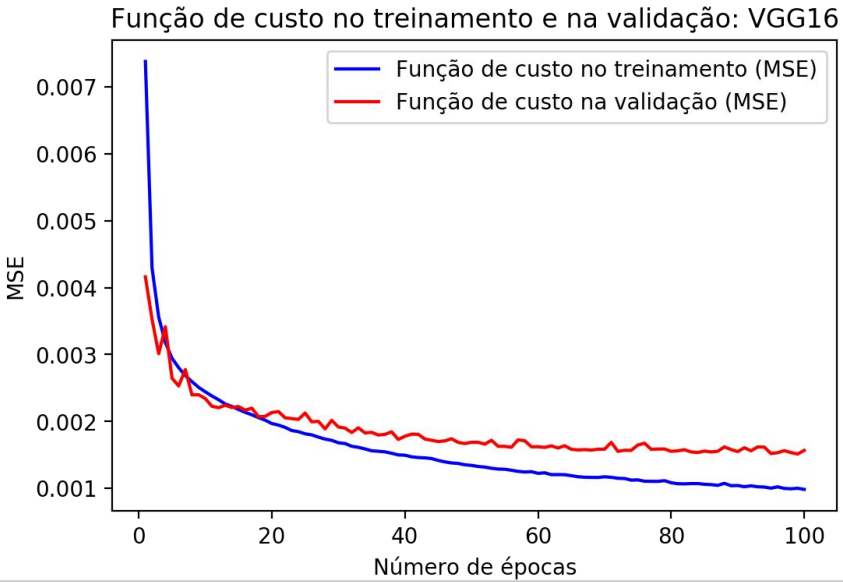
Treinamento

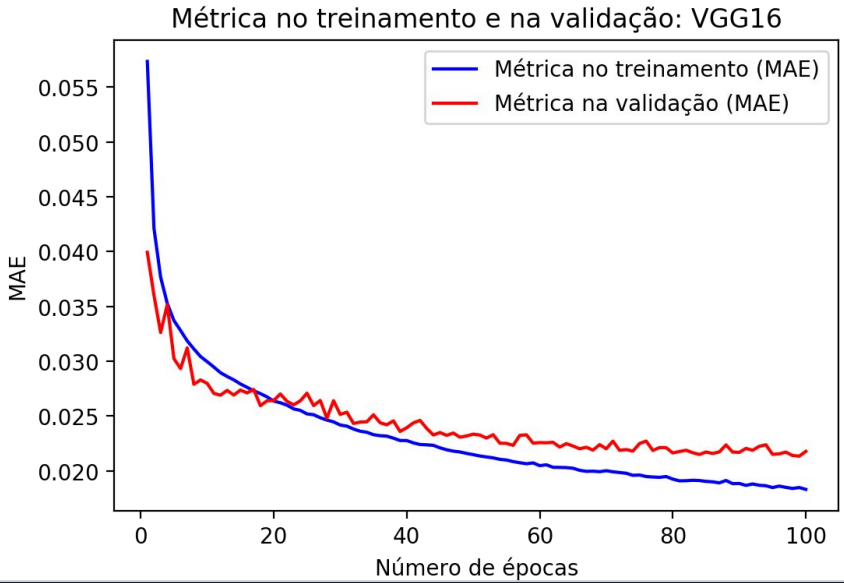
- Testes com diversos hiperparâmetros.
- Redes pré-treinadas são treinadas mais rapidamente.
- Função de custo: Mean Squared Error.
- Métrica: Mean Absolute Error.

Hiperparâmetros	
Tamanho da Batelada	128
Número de épocas	100
Rate (Dropout)	0,3 a 0.5
Otimizador	Adam
Taxa de aprendizado	1e-4

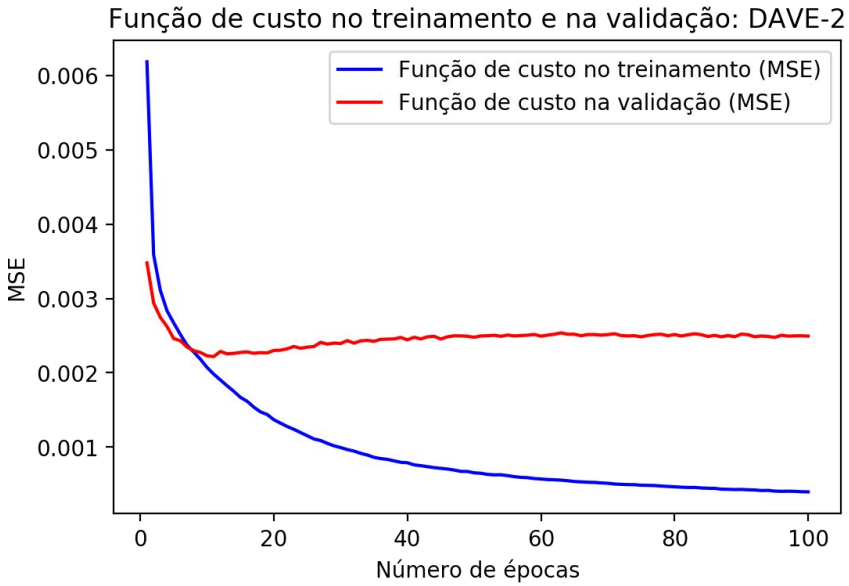
Rede	**Tempo de Treinamento**
VGG16	9s/época
Xception	9s/época
MobileNetV2	7s/época
DAVE-2	51s/época

Gráficos de Treinamento: VGG16





Gráficos de Treinamento: DAVE-2



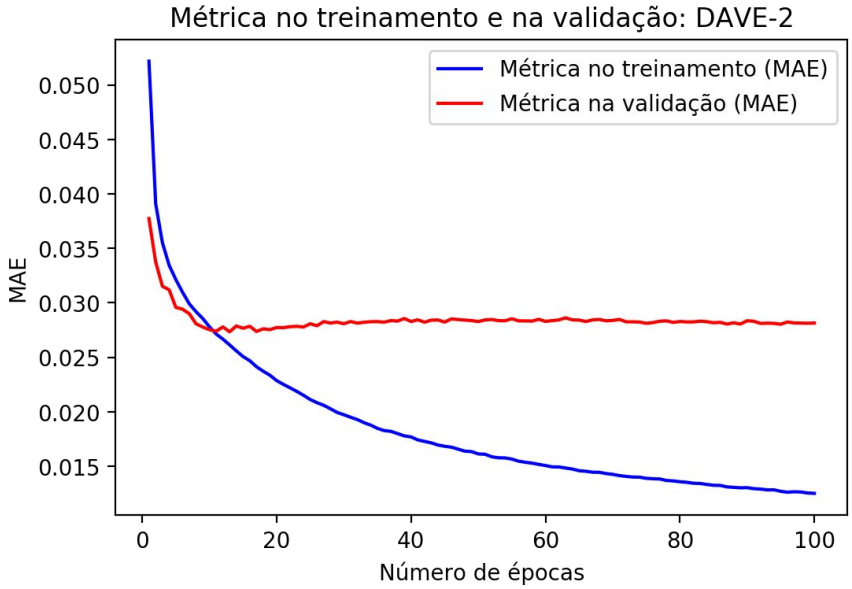
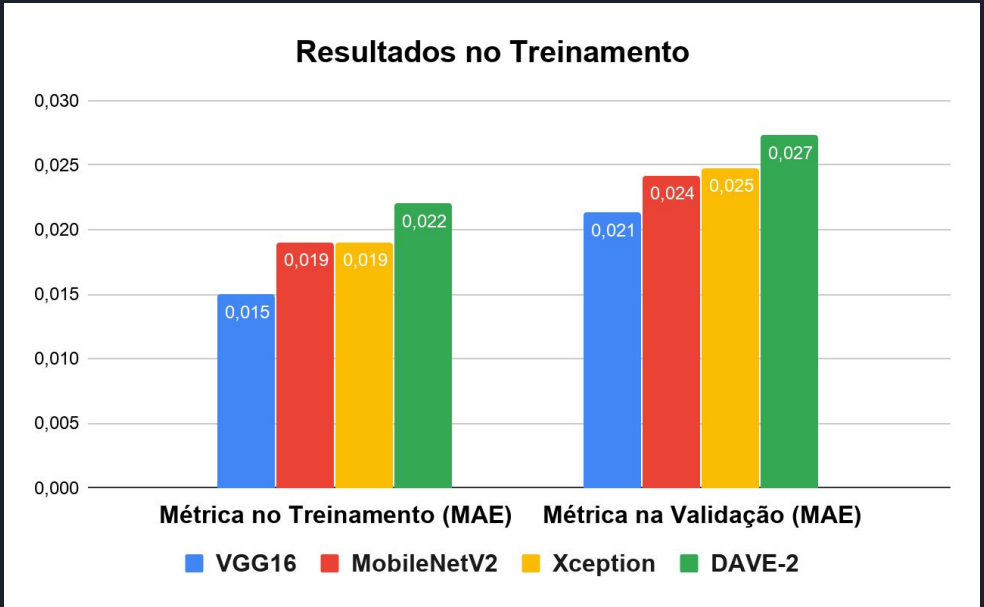


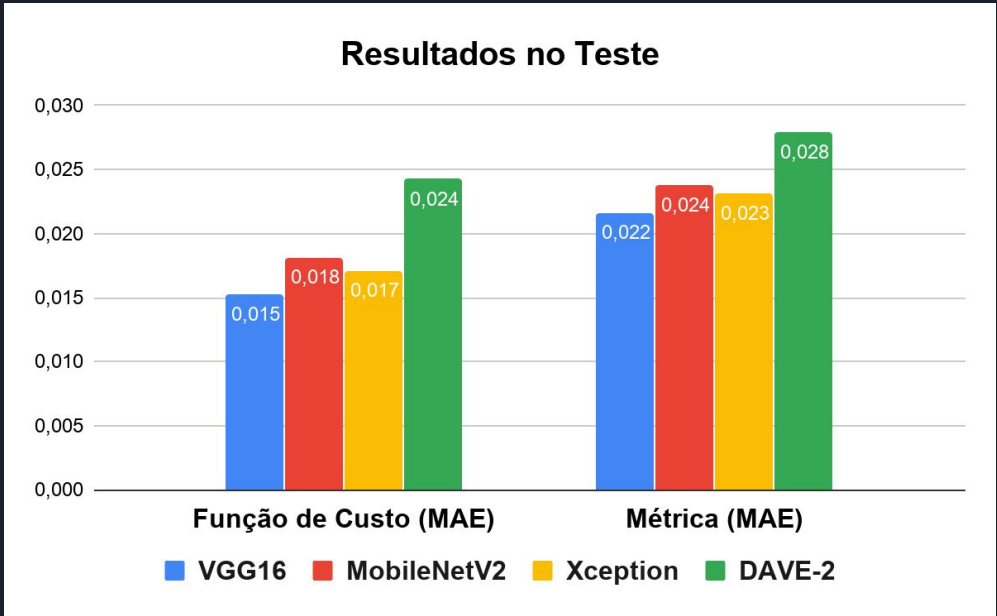
Tabela com Resultados do Treinamento

- VGG16 obteve valores mais baixos.
- DAVE-2 obteve valores mais altos.



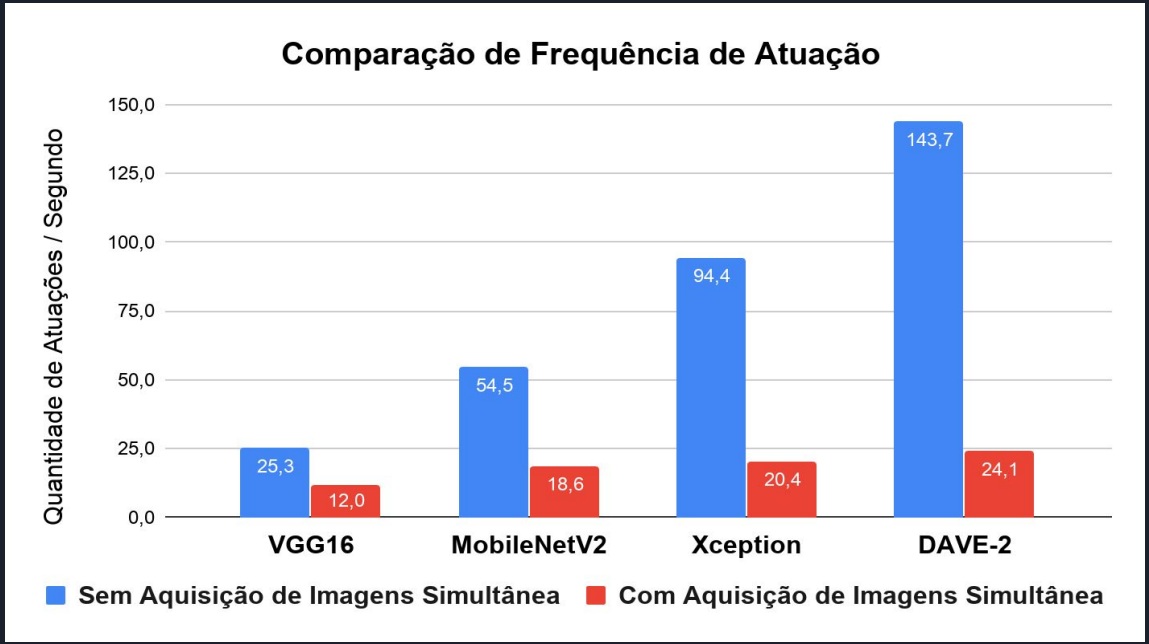
Resultados na Base de Teste

- Valores parecidos com a base de validação.
- Bons resultados.



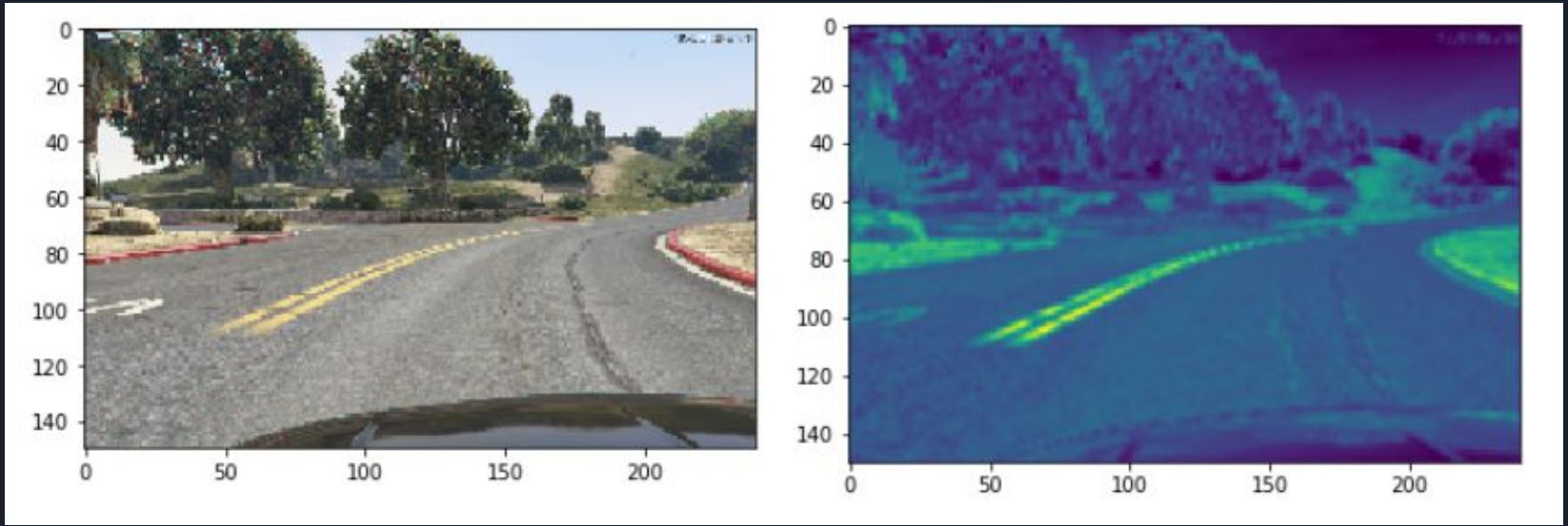
Frequência de cálculo

- Aquisição máxima de fotos por segundo: 29.3 FPS
- Todas as redes atendem a especificação de 10Hz

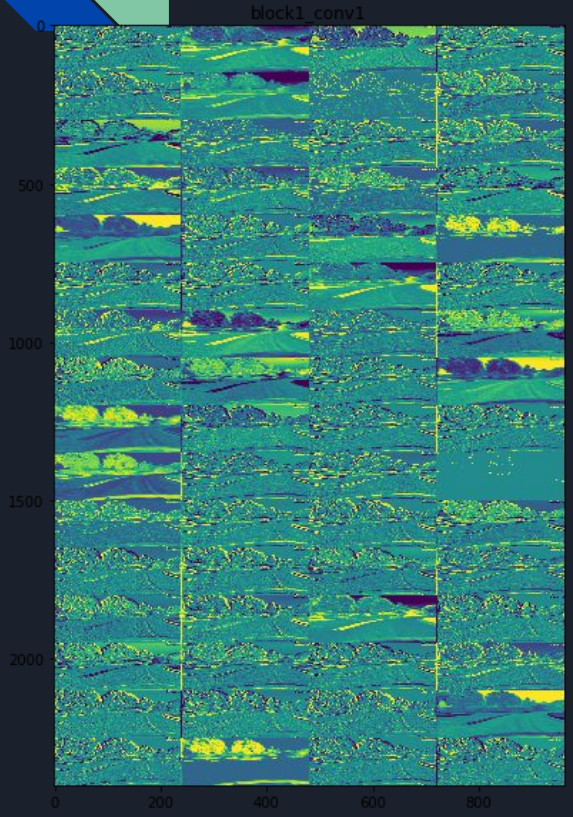


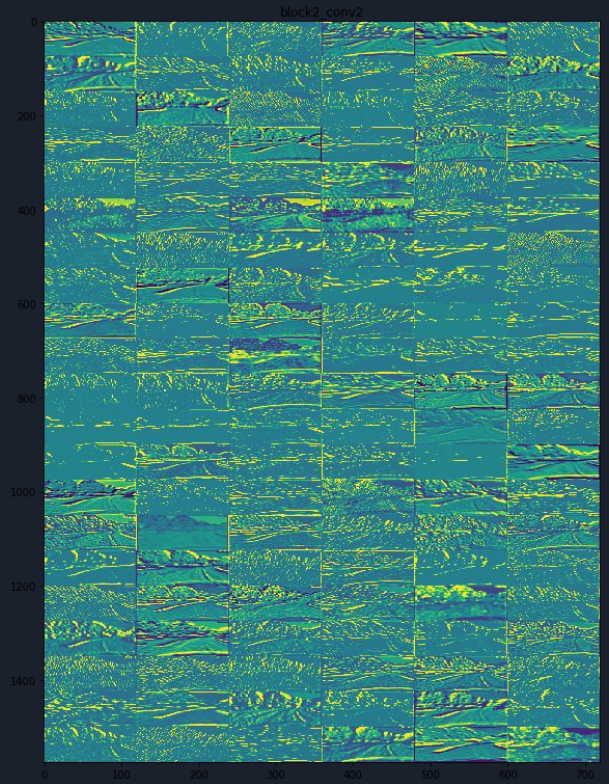
Mapa de Ativações

- Visualização dos filtros convolucionais.
- Destaque de faixas e contornos.



Mapa de Ativações

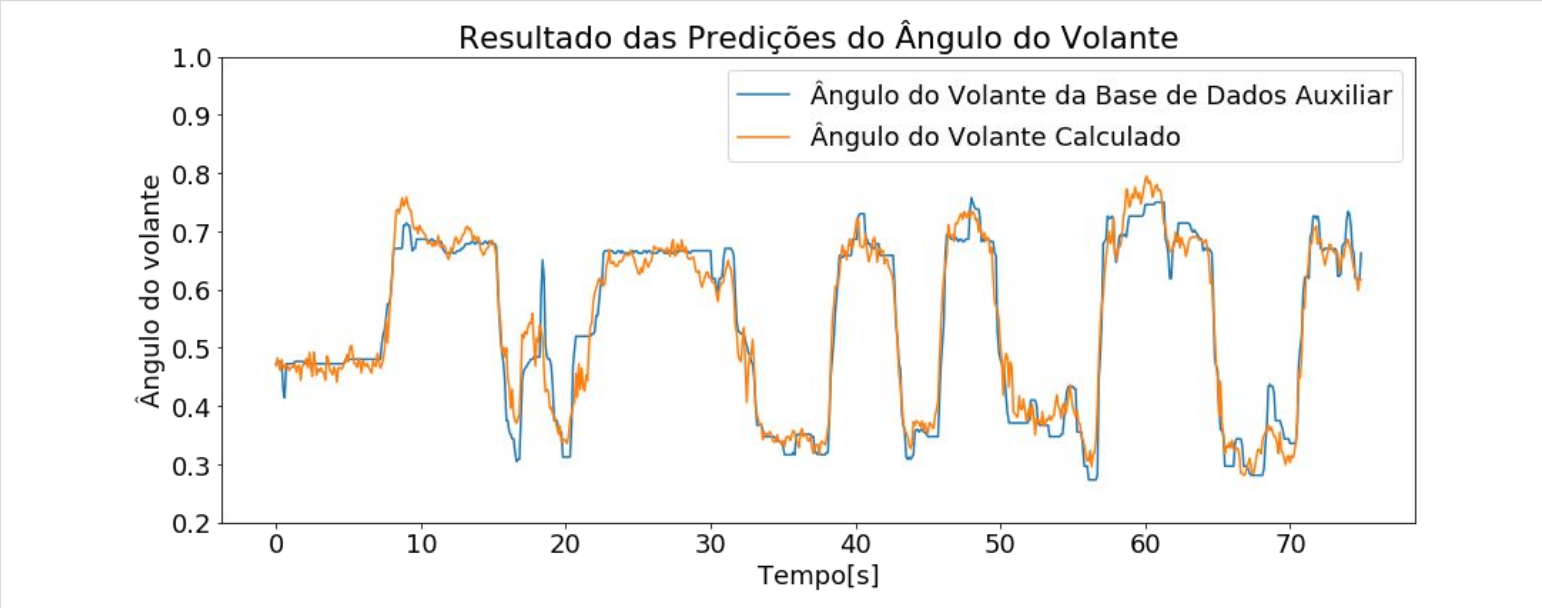




- 1ª e 4ª camadas convolucionais da VGG16.

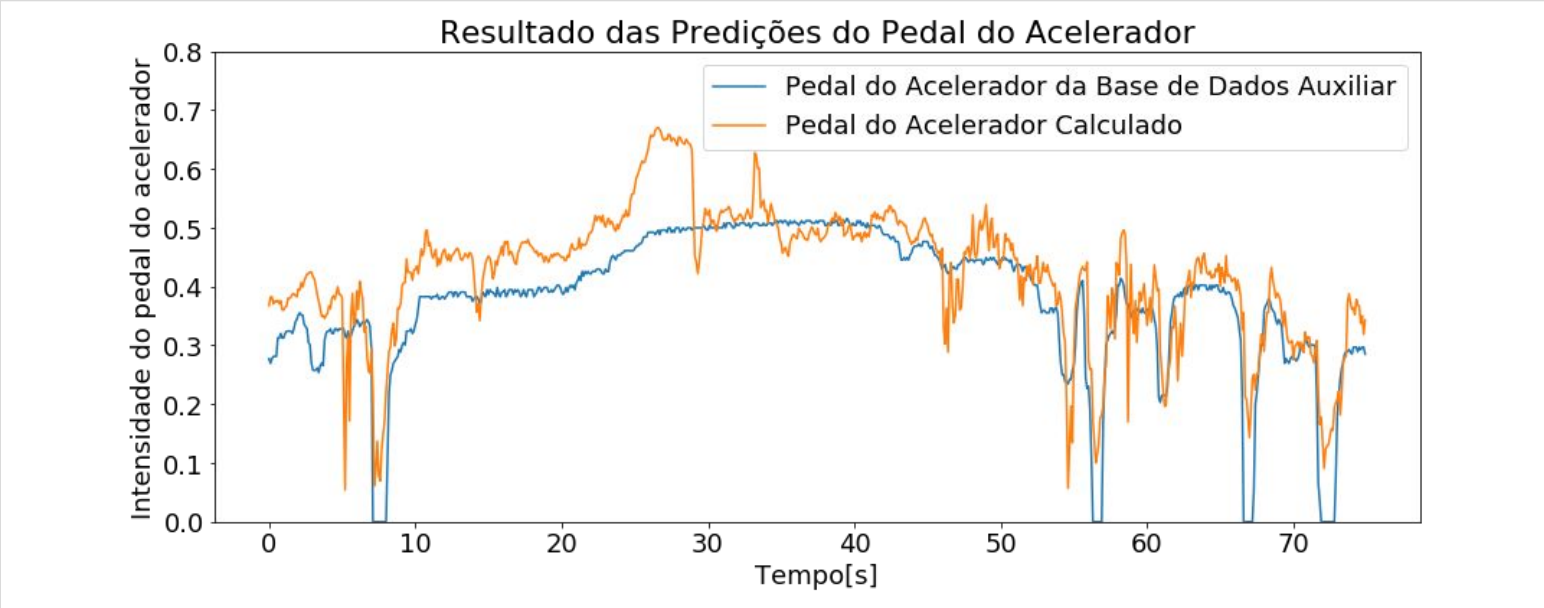
Comparação: Calculado x Real

- Trecho obtido separadamente da base de treinamento e validação.
- Valores calculados condizentes com os reais.



Comparação: Calculado x Real

- Trecho obtido separadamente da base de treinamento e validação.
- Valores calculados condizentes com os reais.



Video

Fonte: https://www.youtube.com/watch?v=8m_QSb9NPIM

Implementação

Implementação

```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from keras.optimizers import SGD

model = Sequential()

# Camadas convolucionais
model.add(Conv2D(8, (3,3), activation='relu', input_shape=(10,12,1)))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(32, (3,3), activation='relu'))
model.add(MaxPooling2D((2,2)))

# Camadas densas
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(26, activation='softmax'))

# Compilar o modelo
sgd = SGD(learning_rate=0.1)
model.compile(optimizer=sgd, loss='categorical_crossentropy', metrics=['accuracy'])

model.summary()
```

Implementação

```
model.summary()
```

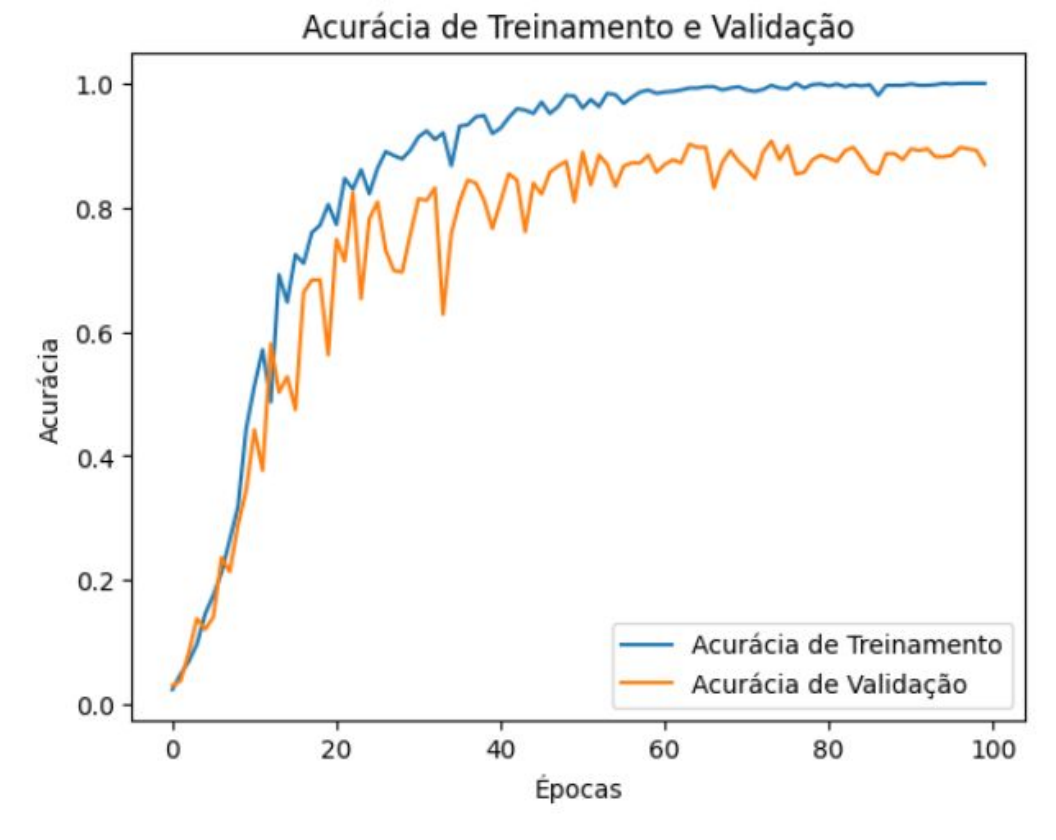
```
Model: "sequential_29"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_58 (Conv2D)          (None, 8, 10, 8)          80

 max_pooling2d_48 (MaxPoolin  (None, 4, 5, 8)          0
 g2D)

 conv2d_59 (Conv2D)          (None, 2, 3, 32)          2336

 max_pooling2d_49 (MaxPoolin  (None, 1, 1, 32)         0
 g2D)

 flatten_29 (Flatten)        (None, 32)                0

 dense_59 (Dense)            (None, 64)                2112

 dense_60 (Dense)            (None, 26)                1690

=================================================================
Total params: 6,218
Trainable params: 6,218
Non-trainable params: 0
_________________________________________________________________
```

Implementação

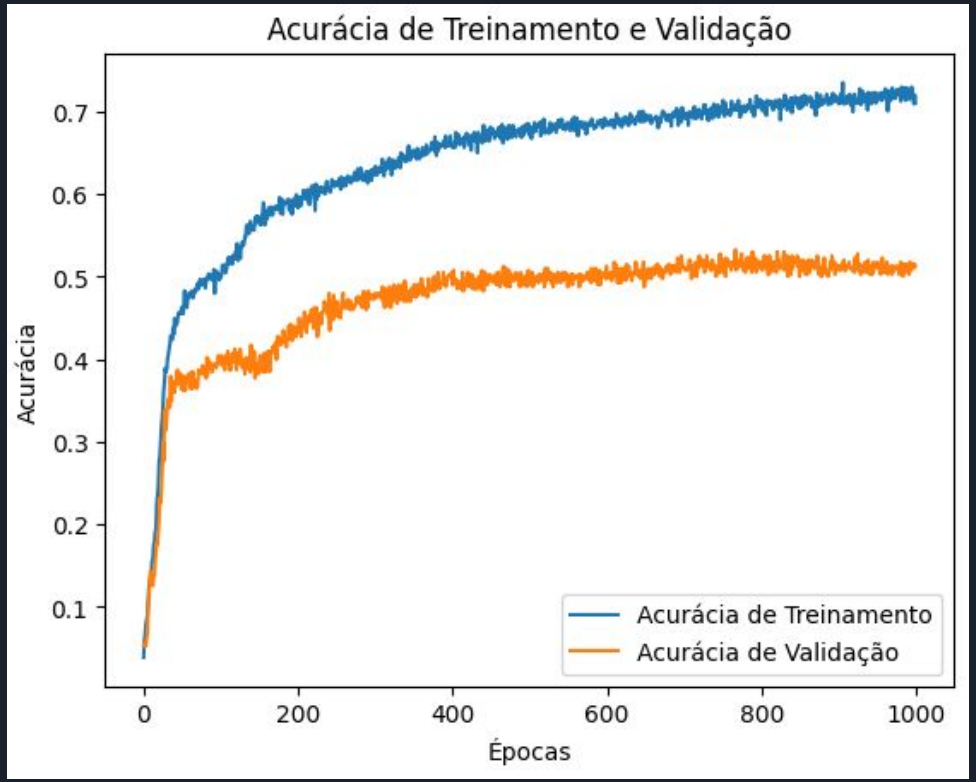
```
# Treinar o modelo
historico_treino = model.fit(X_treino, y_treino, epochs=100, verbose=1,
                             batch_size=128, validation_data=(X_validacao, y_validacao))
```

```
Epoch 1/100
8/8 [==============================] - 1s 33ms/step - loss: 3.2714 - accuracy: 0.0237 - val_loss: 3.2363 - val_accuracy: 0.0302
Epoch 2/100
8/8 [==============================] - 0s 9ms/step - loss: 3.2396 - accuracy: 0.0485 - val_loss: 3.2178 - val_accuracy: 0.0377
Epoch 3/100
8/8 [==============================] - 0s 9ms/step - loss: 3.2140 - accuracy: 0.0690 - val_loss: 3.1966 - val_accuracy: 0.0829
Epoch 4/100
8/8 [==============================] - 0s 9ms/step - loss: 3.1878 - accuracy: 0.0970 - val_loss: 3.1719 - val_accuracy: 0.1382
Epoch 5/100
8/8 [==============================] - 0s 9ms/step - loss: 3.1540 - accuracy: 0.1455 - val_loss: 3.1353 - val_accuracy: 0.1206
Epoch 6/100
8/8 [==============================] - 0s 9ms/step - loss: 3.1068 - accuracy: 0.1756 - val_loss: 3.0867 - val_accuracy: 0.1407
Epoch 7/100
8/8 [==============================] - 0s 9ms/step - loss: 3.0371 - accuracy: 0.2123 - val_loss: 3.0031 - val_accuracy: 0.2362
Epoch 8/100
8/8 [==============================] - 0s 9ms/step - loss: 2.9263 - accuracy: 0.2651 - val_loss: 2.8852 - val_accuracy: 0.2136
Epoch 9/100
8/8 [==============================] - 0s 9ms/step - loss: 2.7594 - accuracy: 0.3179 - val_loss: 2.6973 - val_accuracy: 0.2889
Epoch 10/100
8/8 [==============================] - 0s 9ms/step - loss: 2.5217 - accuracy: 0.4440 - val_loss: 2.4906 - val_accuracy: 0.3442
Epoch 11/100
8/8 [==============================] - 0s 9ms/step - loss: 2.1979 - accuracy: 0.5129 - val_loss: 2.1276 - val_accuracy: 0.4422
Epoch 12/100
```

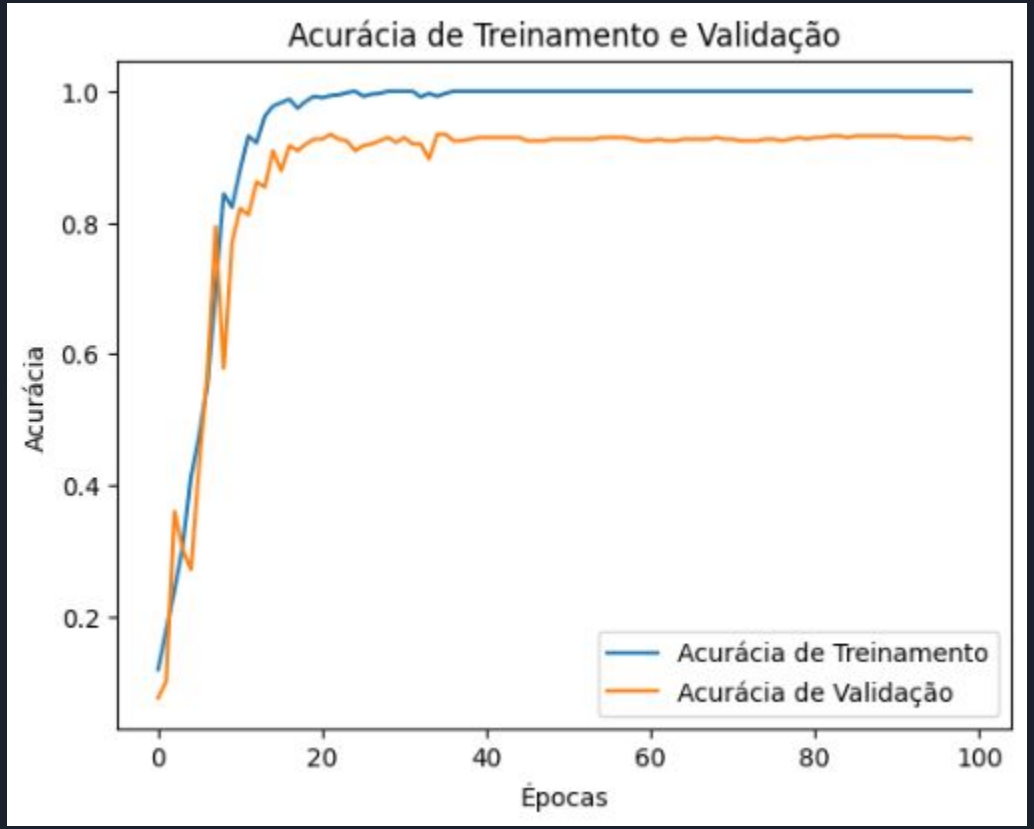
Implementação



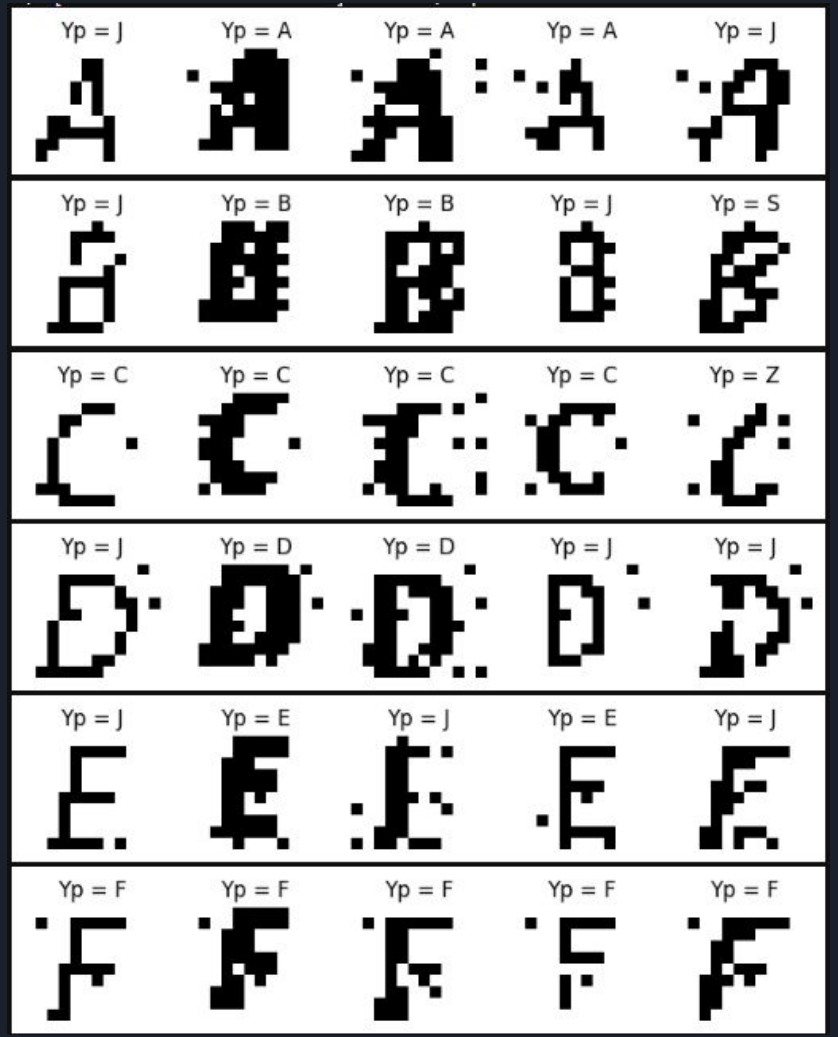
Implementação

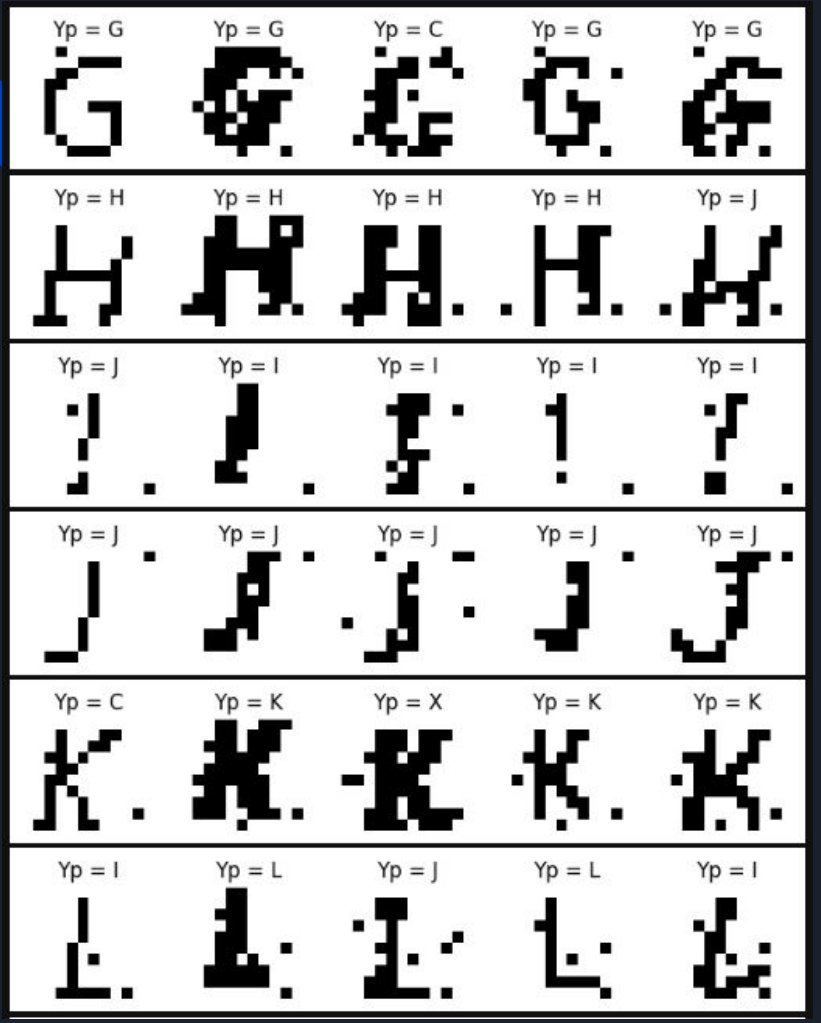


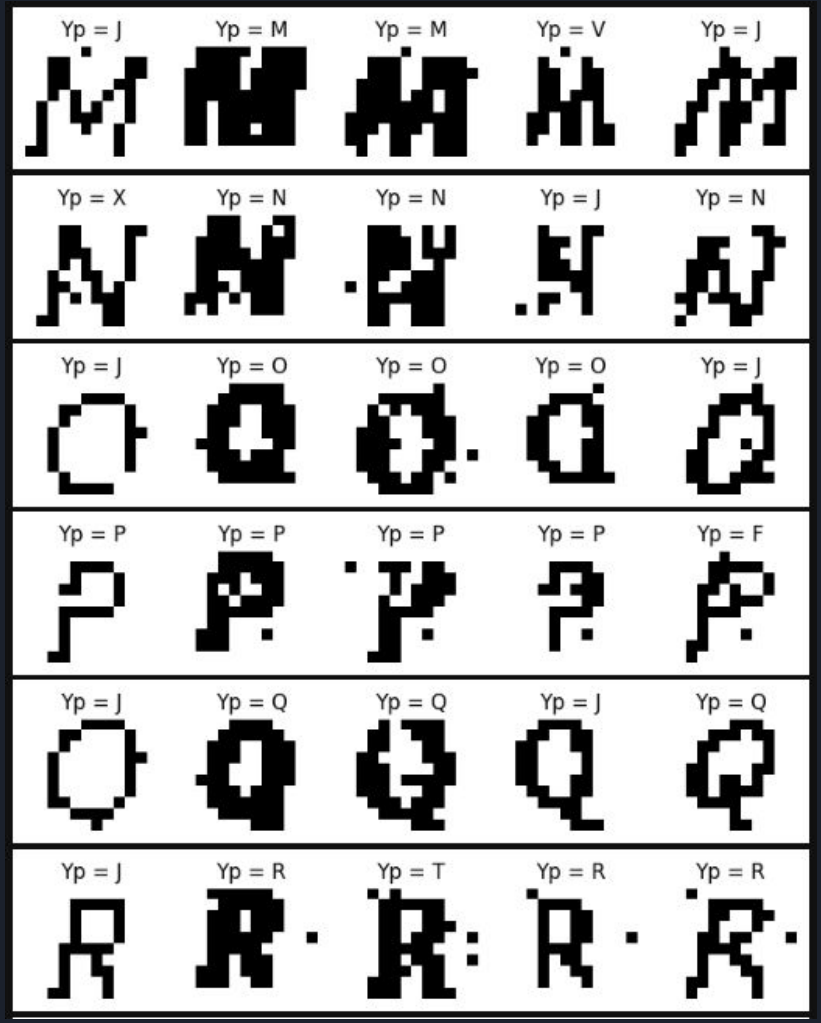
Implementação



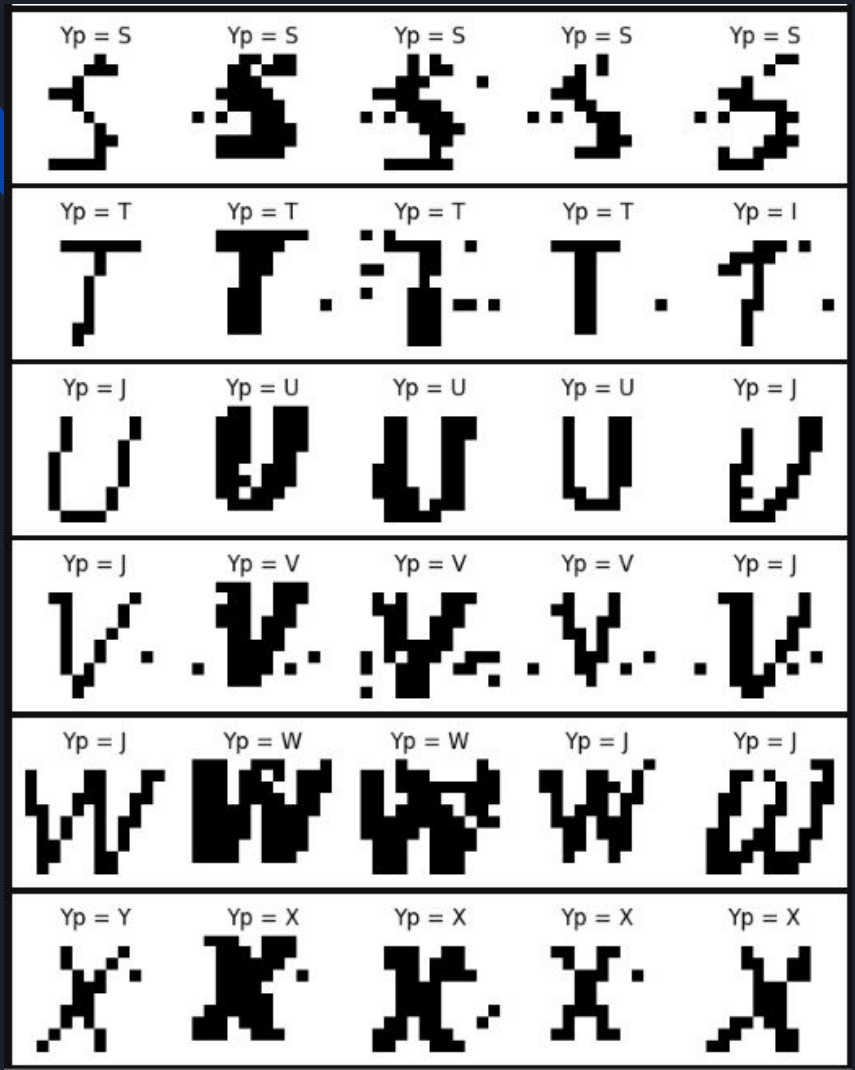
Dados de teste

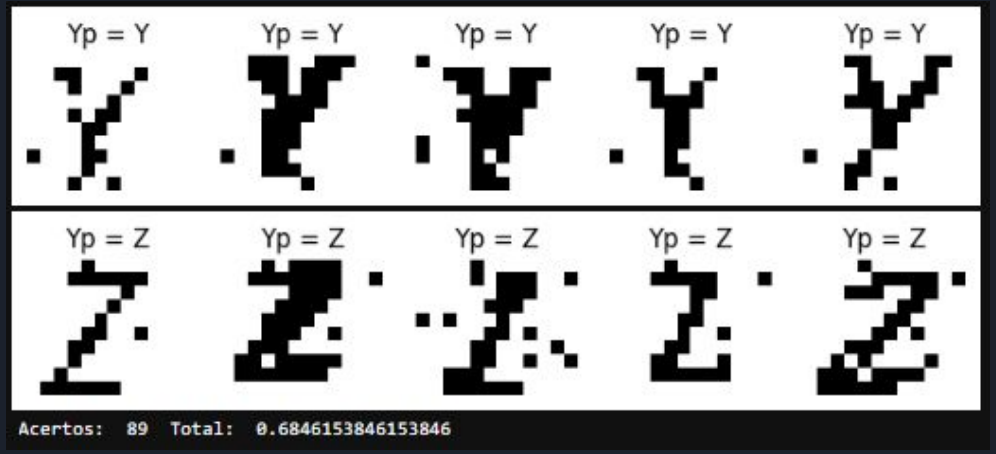



Yp = G
Yp = G
Yp = C
Yp = G
Yp = G
Yp = H
Yp = H
Yp = H
Yp = H
Yp = J
Yp = J
Yp = I
Yp = I
Yp = I
Yp = I
Yp = J
Yp = J
Yp = J
Yp = J
Yp = J
Yp = C
Yp = K
Yp = X
Yp = K
Yp = K
Yp = I
Yp = L
Yp = J
Yp = L
Yp = I


Yp = J
Yp = M
Yp = M
Yp = V
Yp = J
Yp = X
Yp = N
Yp = N
Yp = J
Yp = N
Yp = J
Yp = O
Yp = O
Yp = O
Yp = J
Yp = P
Yp = P
Yp = P
Yp = P
Yp = F
Yp = J
Yp = Q
Yp = Q
Yp = J
Yp = Q
Yp = J
Yp = R
Yp = T
Yp = R
Yp = R

Dados de teste





Fim