Detección e identificación de características en vehículos utilizando algoritmos de Machine Learning.

Instituto Balseiro - Universidad Nacional de Cuyo

Alumno: Enrique Nicanor Mariotti

Noviembre 2020

Instituto Balseiro

COMISIÓN NACIONAL DE ENERGÍA ATÓMICA

INVAP

Texto


Attention
Dog Guardians
Pick up after your dogs. Thank you.
Attention Dogs
Grrrrr, bark, woof. Good dog.
District of North Vancouver.
Bylaw 5981-11(i)


BEWARE OF SMARTPHONE ZOMBIES


DANGER
THE DOG HAS A GUN AND REFUSES TO TAKE HIS MEDICATION


CAUTION
THIS SIGN HAS
SHARP EDGES
DO NOT TOUCH THE EDGES OF THIS SIGN
ALSO, THE BRIDGE IS OUT AHEAD

Texto

Particularidades

- Permite inferir información sobre el contenido semántico de la escena.
- Particularmente difícil mediante técnicas de OCR, ideadas para la lectura de documentos escritos.
- Gran numero de aplicaciones y amplia disponibilidad.

Texto

Problemas asociados:

- Variables en lay-out
- Estilos y tipografías distintas
- Iluminación variable
- Oclusiones parciales o totales
- Orientación
- Ruido
- Background FP

ALPR

Automatic Licence Plate Recognition:

- Control automotor por parte de las fuerzas de la ley.
- Sondeo de información sobre lotes automotores vastos.
- Control de acceso a lugares privados.

ALPR

Image Adquisition

Licence Plate Segmentation

Character Segmentation

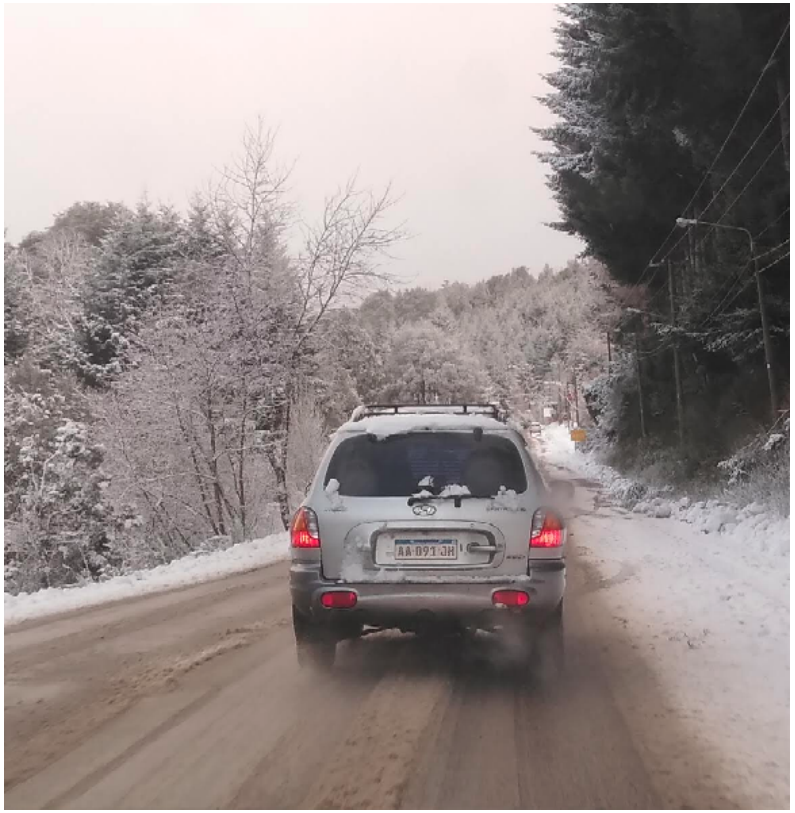
Character Recognition

ALPR

Problemas:

- Diferentes colores, fuentes y fondos
- Tamaño de caracteres no uniforme
- Factores externos:
  - Velocidad del vehículo
  - Angulo de la cámara
  - Trasfondo de la escena
  - Iluminación
  - Muchos otros...

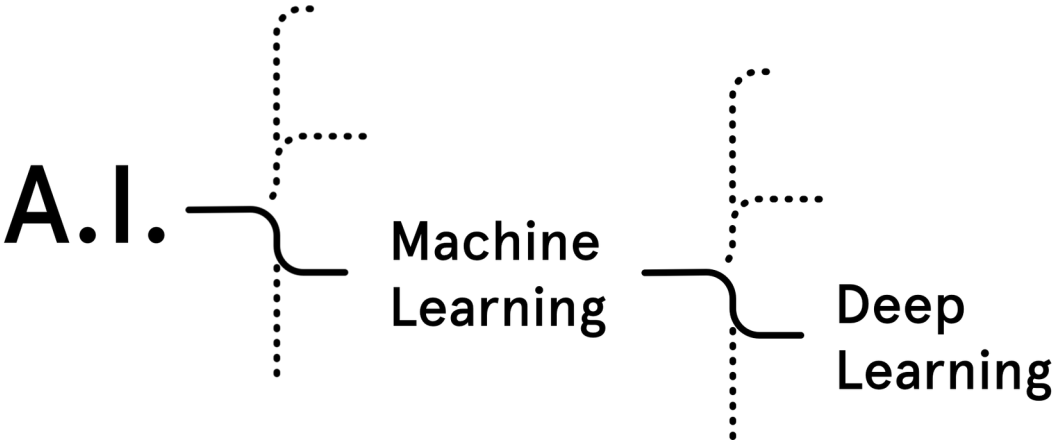
ALPR

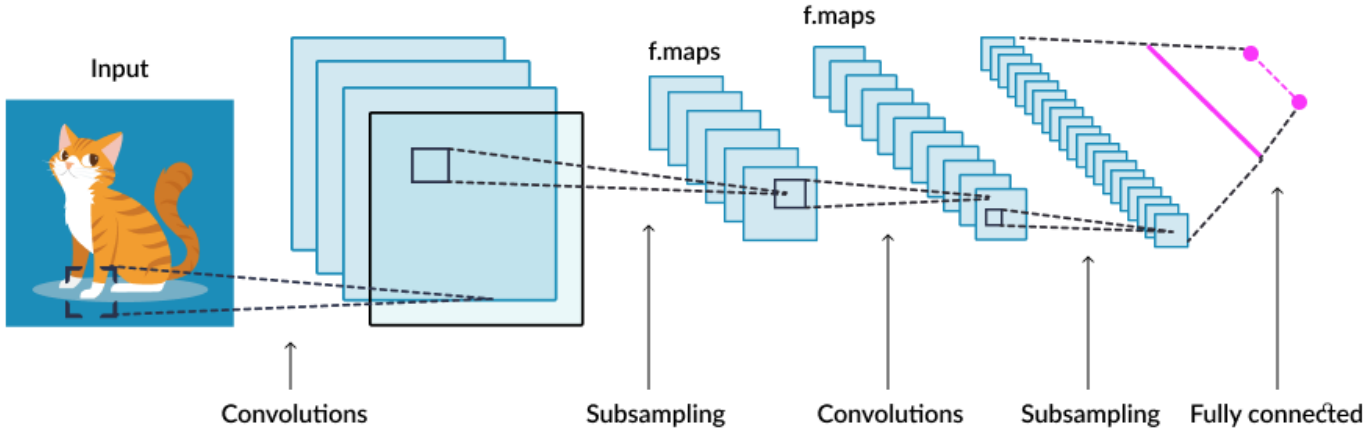


Objetivos

- Explorar una solución integral al problema **ALPR** usando **Deep Learning** (**DL**):
  - Patrones complejos de información.
  - Trabajar con datos no estructurados.
  - La interpretabilidad del resultado no es un factor preponderante.

Objetivos


A.I.
Machine Learning
Deep Learning


Input
f.maps
f.maps
Convolutions
Subsampling
Convolutions
Subsampling
Fully connected

Objetivos

Problemática:

- Disponibilidad de datos anotados (**ALPR**) y recursos físicos (muy) limitada.

¿Solución?:

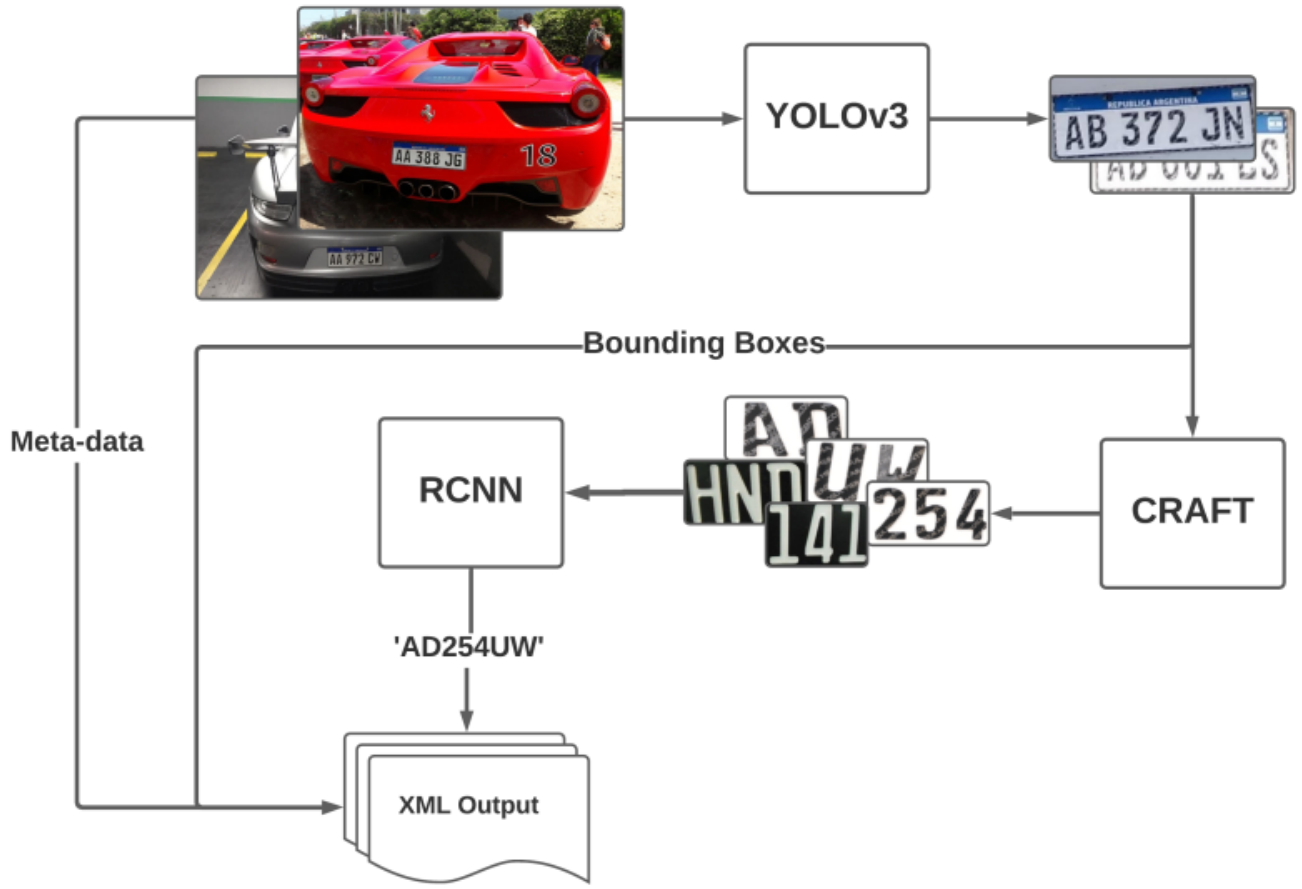
- Priorizar el uso de transfer learning en redes profundas.

Objetivos

Requerimientos adicionales:

- Secuencia de etapas modulares (**OOP**). Los módulos u objetos individuales pueden ser usados con otros propósitos.
- **Caso de estudio:** Patente única del Mercosur, para facilitar la comparación de los resultados.

Lay-Out


YOLOv3
Bounding Boxes
Meta-data
RCNN
CRAFT
'AD254UW'
XML Output

YOLO

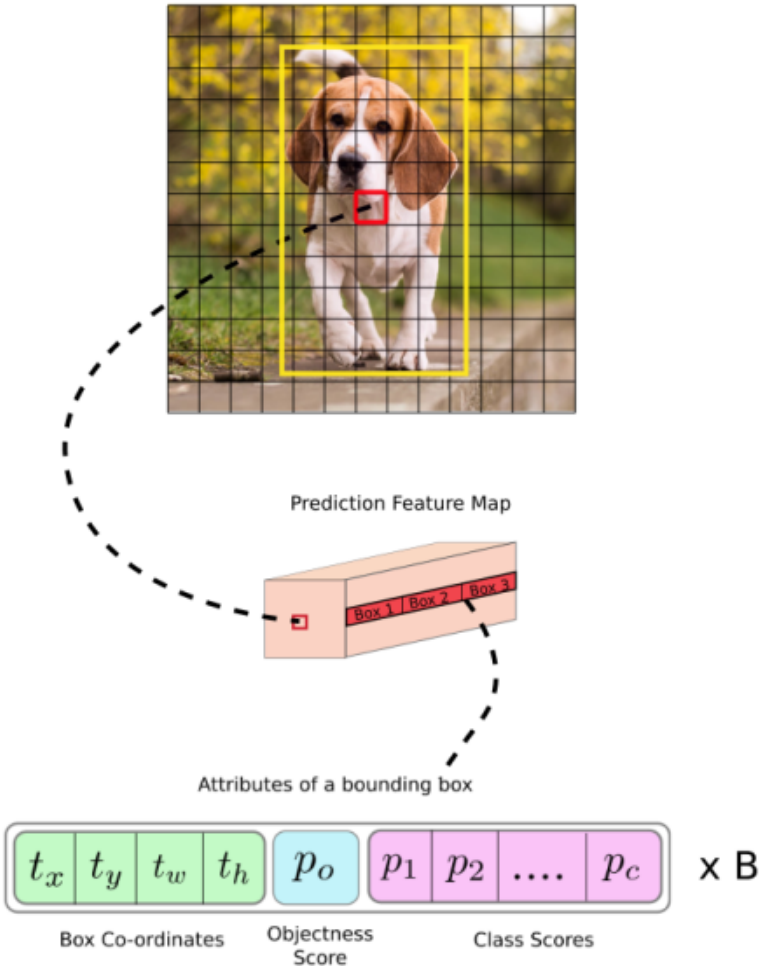
	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Tabla 2.1: **CNN** detrás del procesamiento de **YOLO** [1].

YOLO

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Tabla 2.1: CNN detrás del procesamiento de **YOLO** [1].



YOLO

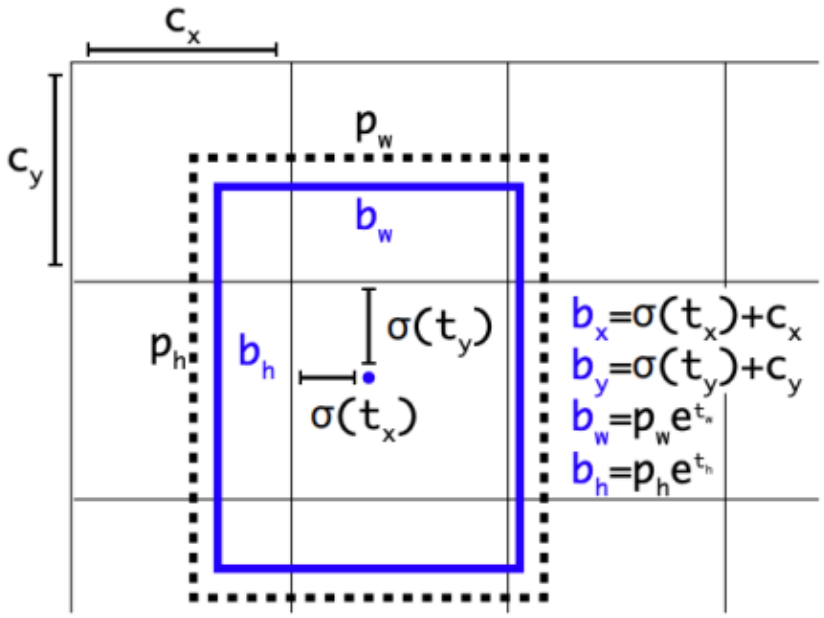
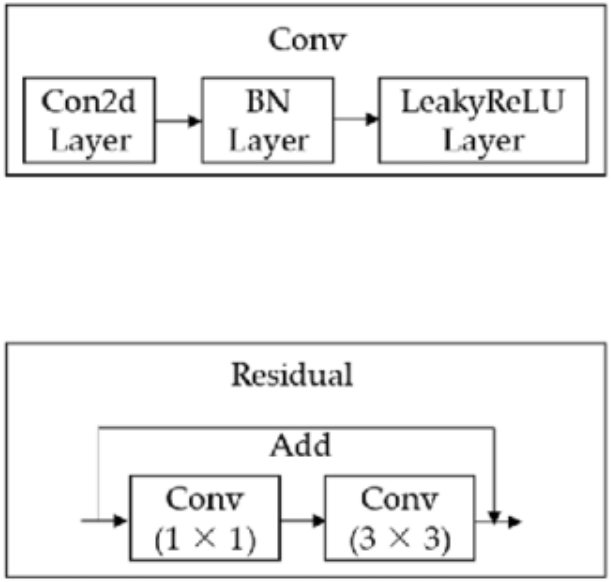


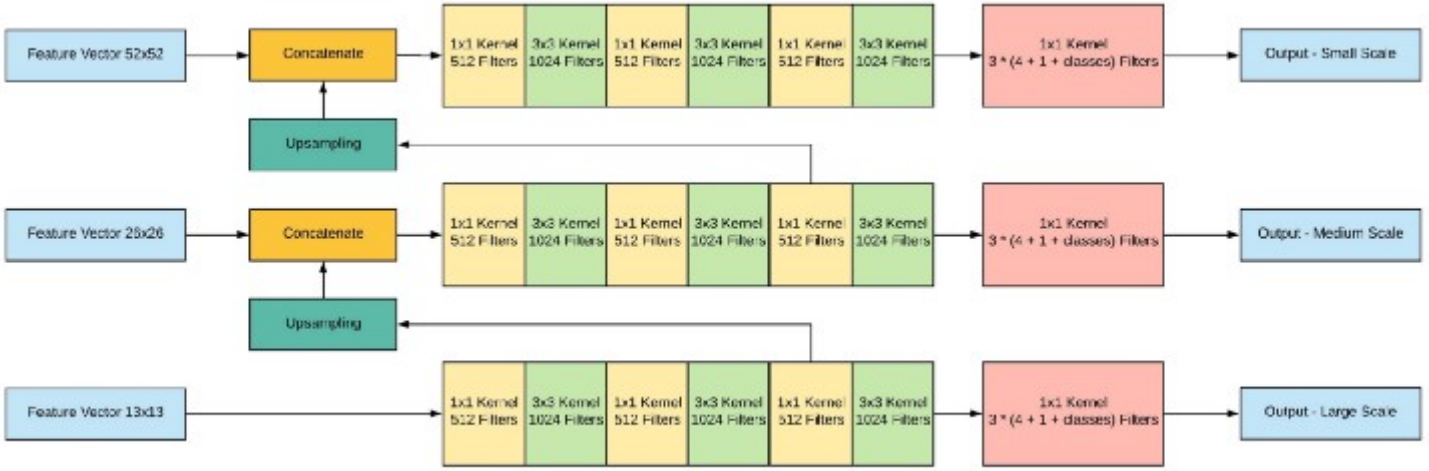
Figura 2.3: Predicción de la celda (c_x, c_y) en función de las dimensiones predefinidas de los *anchors* y de una de las predicciones de la red [1].

YOLO

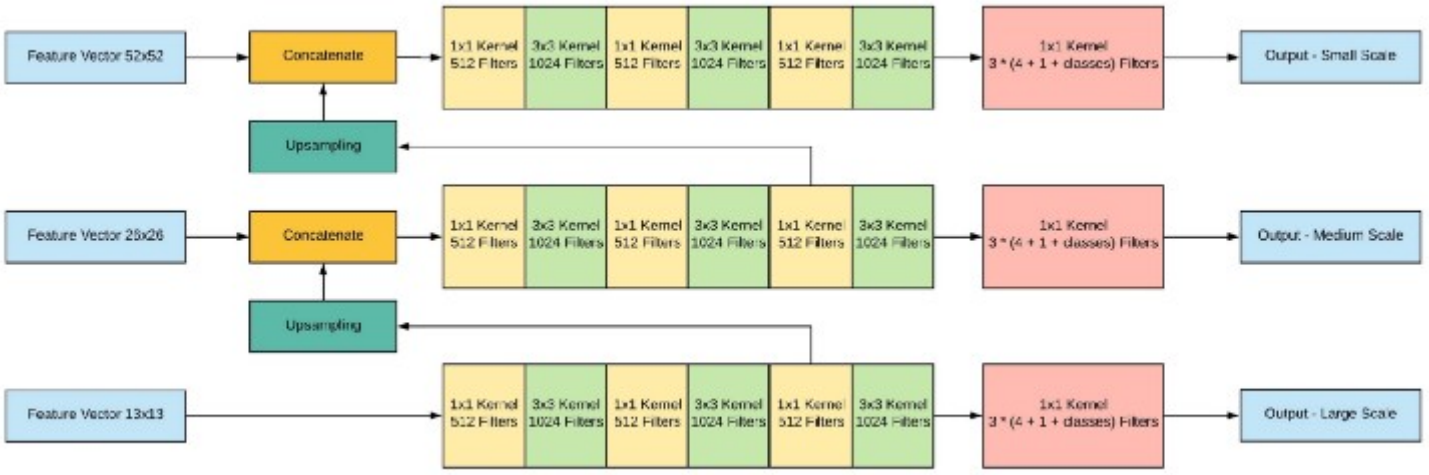
Layer	Filters size	Repeat	Output size
Image			416 × 416
Conv	32 3 × 3/1	1	416 × 416
Conv	64 3 × 3/2	1	208 × 208
Conv Conv Residual	32 1 × 1/1 64 3 × 3/1	Conv Conv Residual × 1	208 × 208 208 × 208 208 × 208
Conv	128 3 × 3/2	1	104 × 104
Conv Conv Residual	64 1 × 1/1 128 3 × 3/1	Conv Conv Residual × 2	104 × 104 104 × 104 104 × 104
Conv	256 3 × 3/2	1	52 × 52
Conv Conv Residual	128 1 × 1/1 256 3 × 3/1	Conv Conv Residual × 8	52 × 52 52 × 52 52 × 52
Conv	512 3 × 3/2	1	26 × 26
Conv Conv Residual	256 1 × 1/1 512 3 × 3/1	Conv Conv Residual × 8	26 × 26 26 × 26 26 × 26
Conv	1024 3 × 3/2	1	13 × 13
Conv Conv Residual	512 1 × 1/1 1024 3 × 3/1	Conv Conv Residual × 4	13 × 13 13 × 13 13 × 13



YOLO


Feature Vector 52x52
Concatenate
Upsampling
1x1 Kernel 512 Filters
3x3 Kernel 1024 Filters
1x1 Kernel 512 Filters
3x3 Kernel 1024 Filters
1x1 Kernel 512 Filters
3x3 Kernel 1024 Filters
1x1 Kernel 3 * (4 + 1 + classes) Filters
Output - Small Scale
Feature Vector 26x26
Concatenate
Upsampling
1x1 Kernel 512 Filters
3x3 Kernel 1024 Filters
1x1 Kernel 512 Filters
3x3 Kernel 1024 Filters
1x1 Kernel 512 Filters
3x3 Kernel 1024 Filters
1x1 Kernel 3 * (4 + 1 + classes) Filters
Output - Medium Scale
Feature Vector 13x13
1x1 Kernel 512 Filters
3x3 Kernel 1024 Filters
1x1 Kernel 512 Filters
3x3 Kernel 1024 Filters
1x1 Kernel 512 Filters
3x3 Kernel 1024 Filters
1x1 Kernel 3 * (4 + 1 + classes) Filters
Output - Large Scale

YOLO



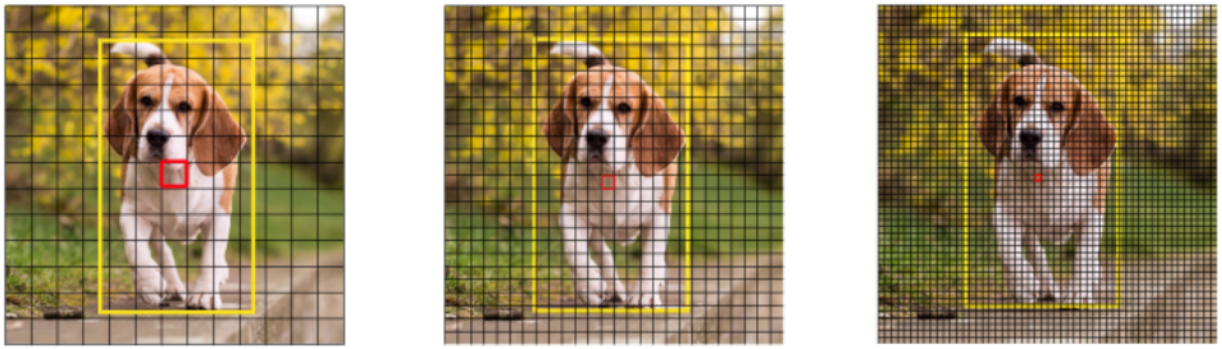
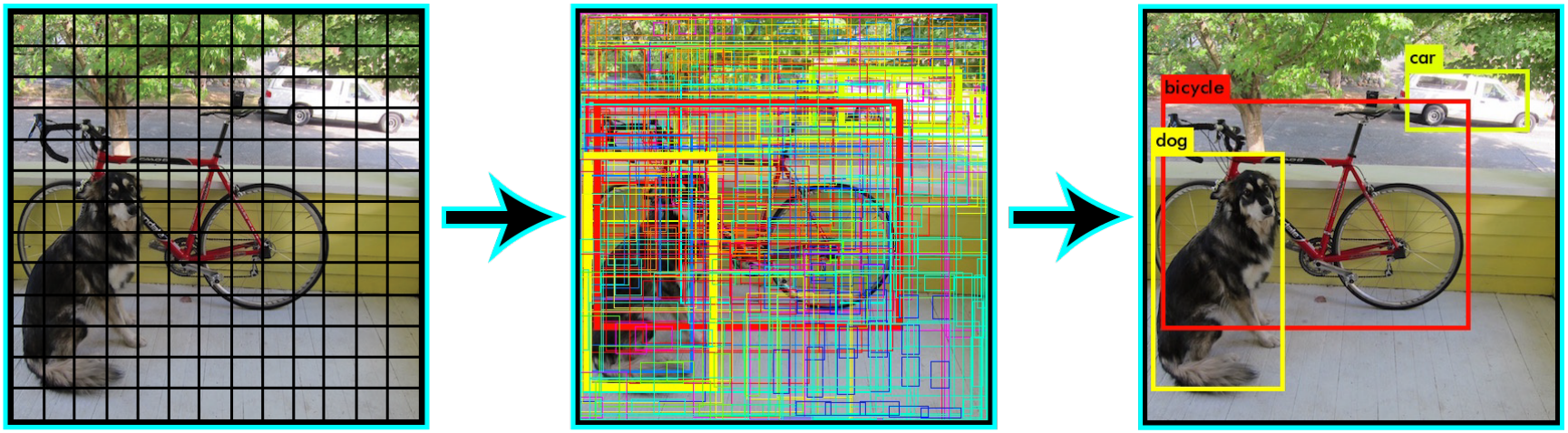


Figura 2.4: Salida multiescala de **YOLOv3** [28].

YOLO



YOLO

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \lambda_{\text{obj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{classes}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

YOLO

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \lambda_{\text{obj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

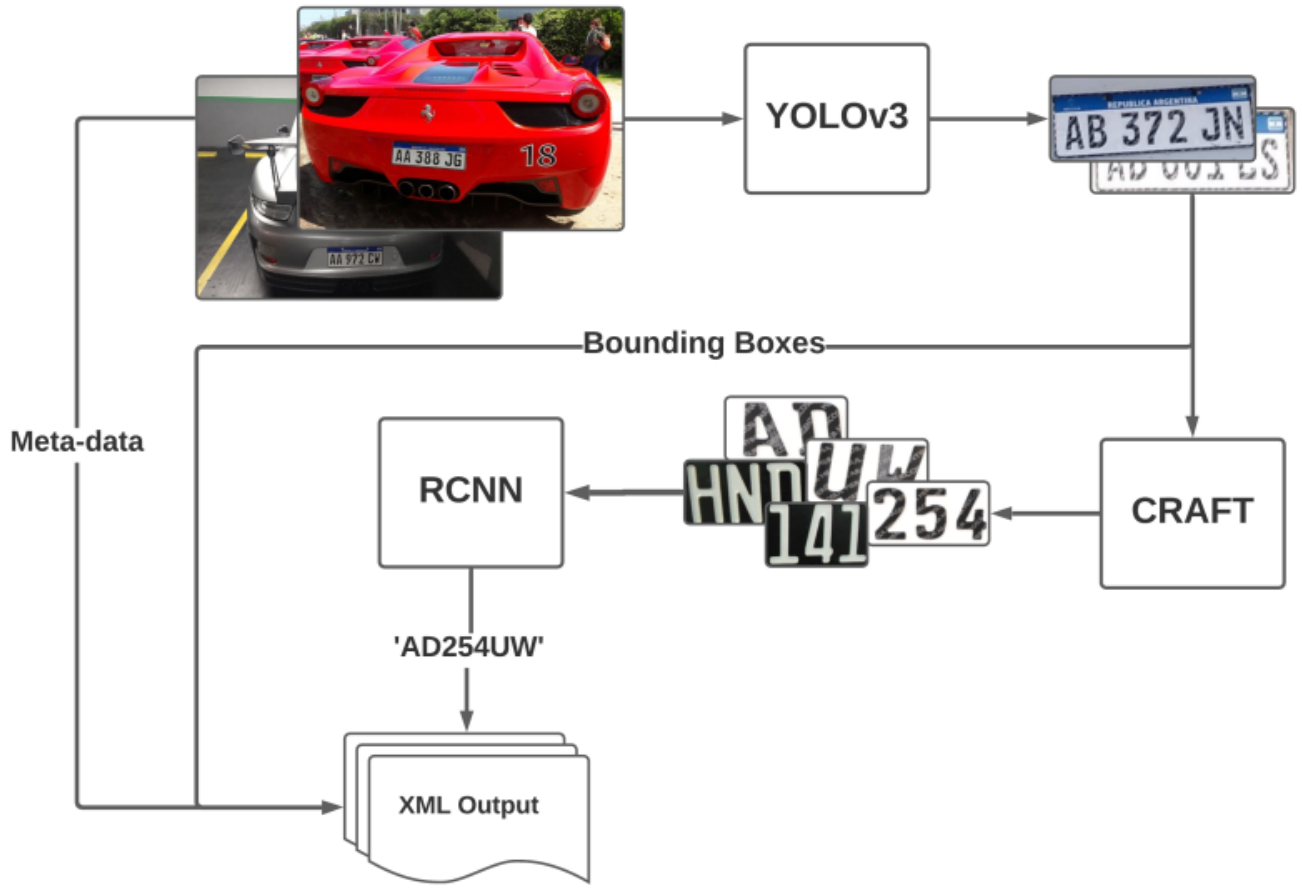
$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{classes}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

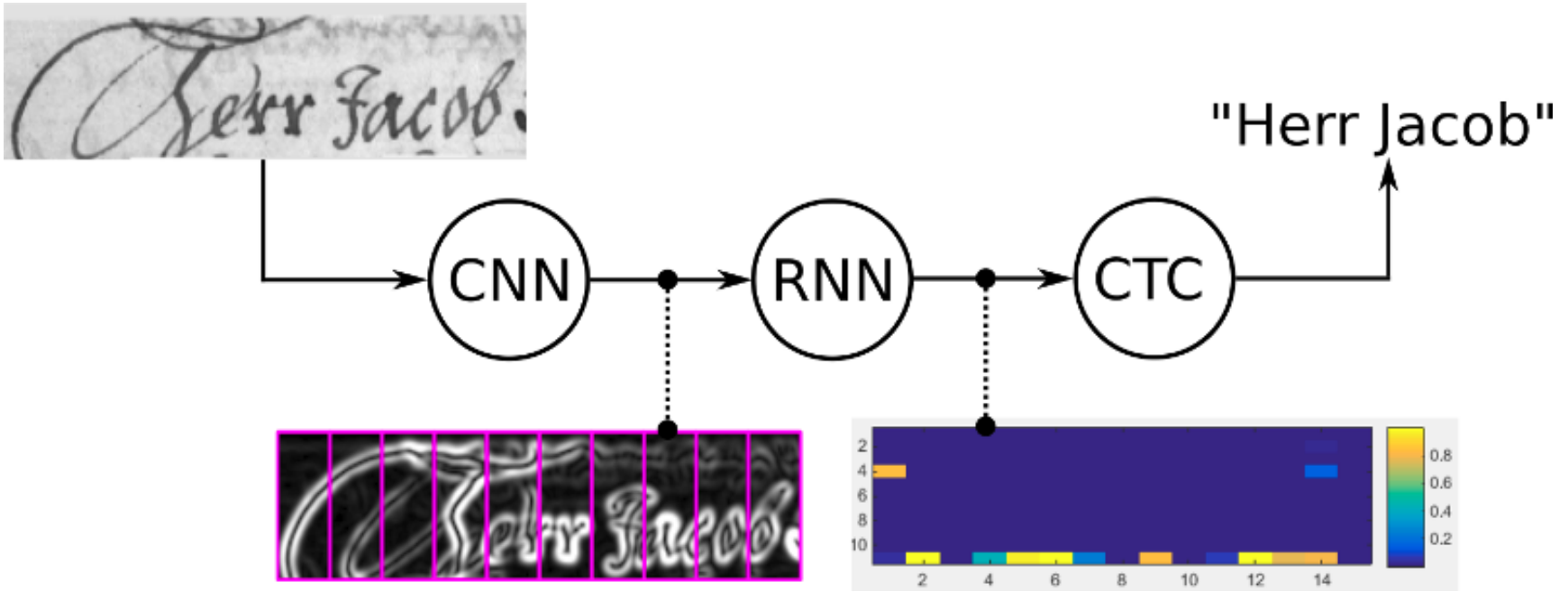
YOLO

- 100 epochs
- Darknet-53 freezada
- Stochastic Gradient Descent (SGD)
- Learning rate: $1x10^{-4}$
- Decay: $5x10^{-4}$
- Momentum: $9x10^{-1}$
- Batch de entrenamiento: 16 imágenes.
- Aumentación de datos no distorsiva.
- Patente de la Unión Europea (500 imágenes). Anotación con localización

Lay-Out


YOLOv3
Bounding Boxes
Meta-data
RCNN
CRAFT
'AD254UW'
XML Output

RNN

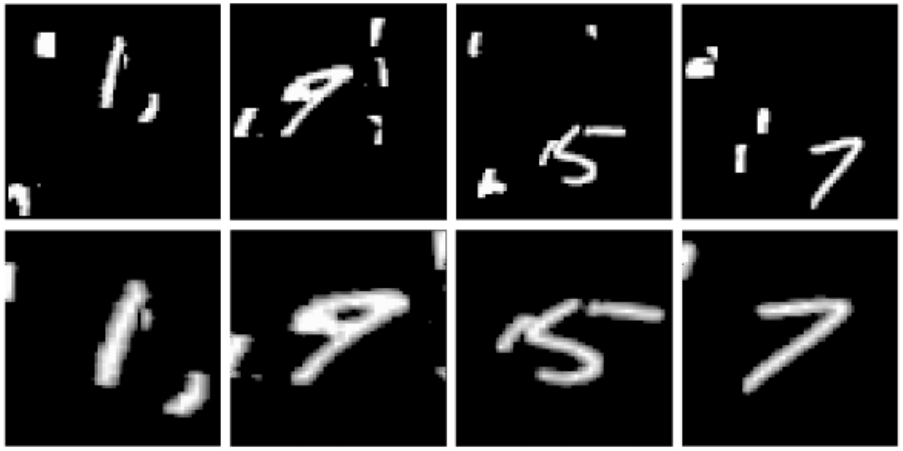


RNN

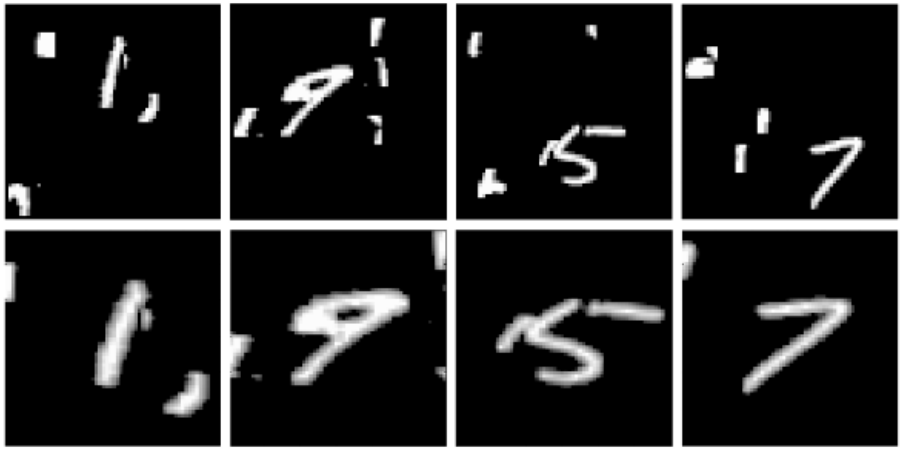
Bloque	Layer	Output Size	Features	Kernel Size	Padding	Activation
1	Input	200x31	-	-	-	-
	Conv2D	200x31x64	64	3x3	same	relu
	Conv2D	200x31x128	128	3x3	same	relu
	Conv2D	200x31x256	256	3x3	same	relu
	BatchNormalization	200x31x256	-	-	-	-
	MaxPool2D	100x15x256	-	2x2	-	-
2	Conv2D	100x15x256	256	3x3	same	relu
	Conv2D	100x15x512	512	3x3	same	relu
	BatchNormalization	100x15x512	-	-	-	-
	MaxPool2D	50x7x512	-	2x2	-	-
3	Conv2D	50x7x512	512	3x3	same	relu
	Conv2D	50x7x512	512	3x3	same	relu
	BatchNormalization	50x7x512	-	-	-	-

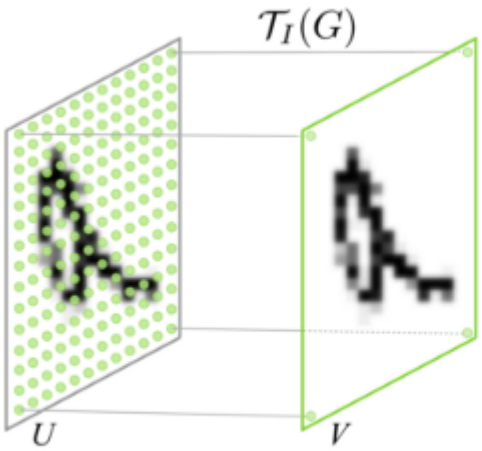
Tabla 3.1: Arquitectura empleada.

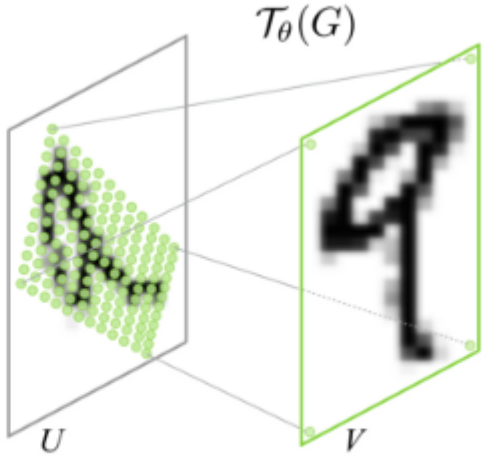
RNN



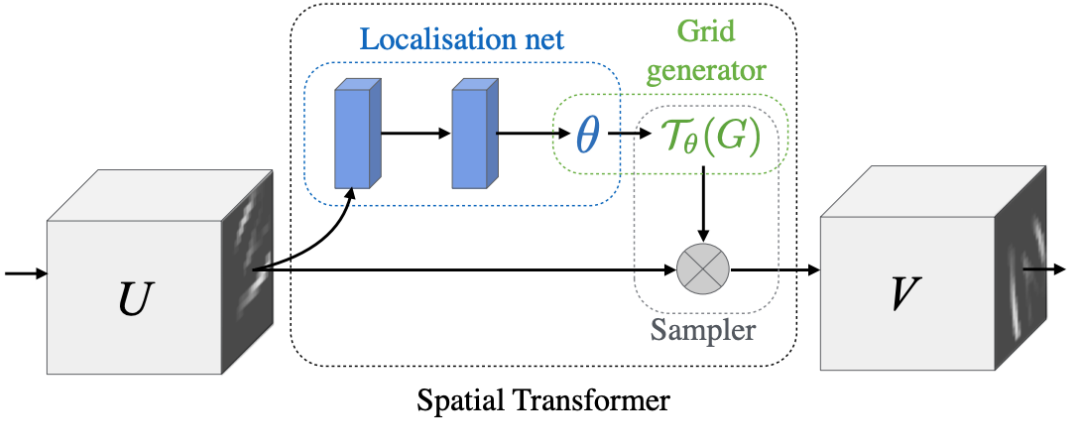
RNN



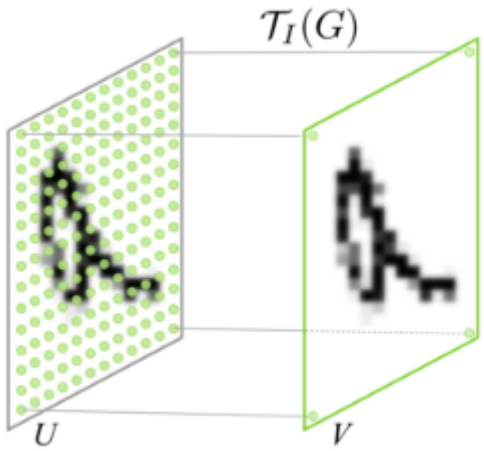


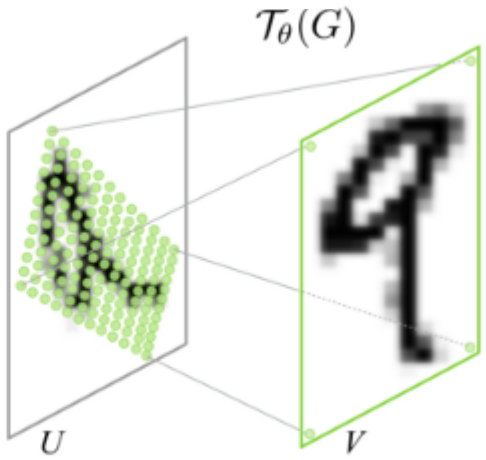


RNN



Spatial Transformer





RNN

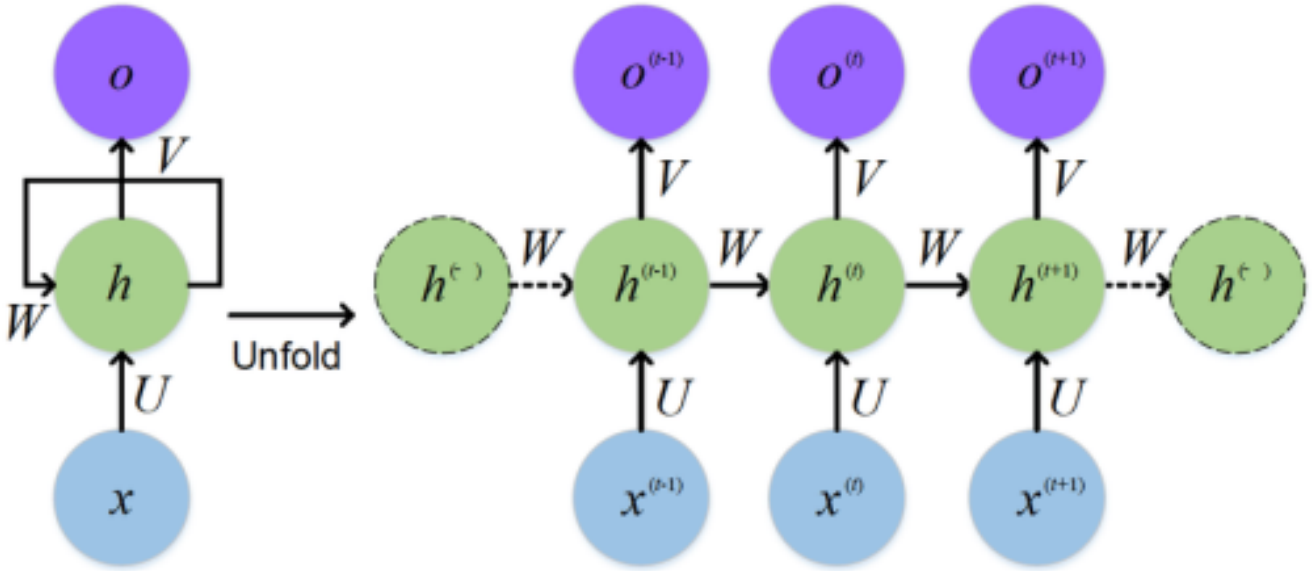
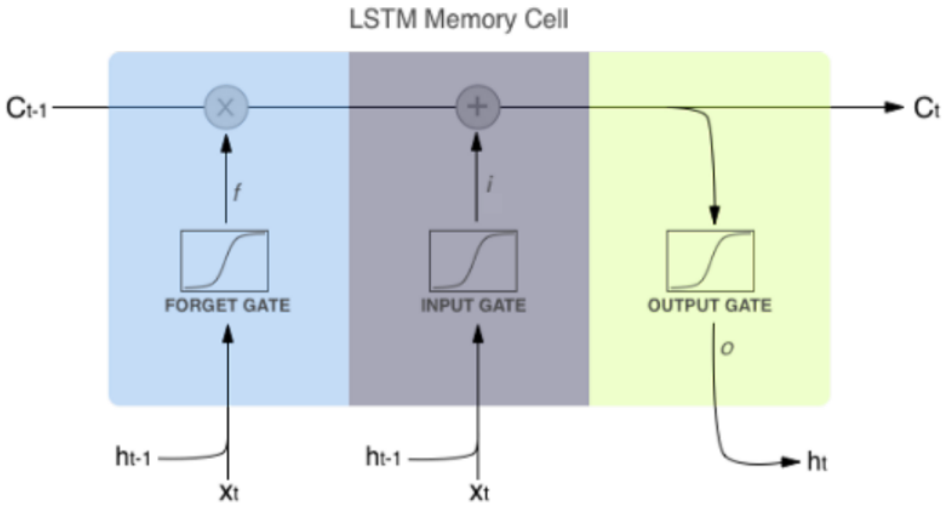
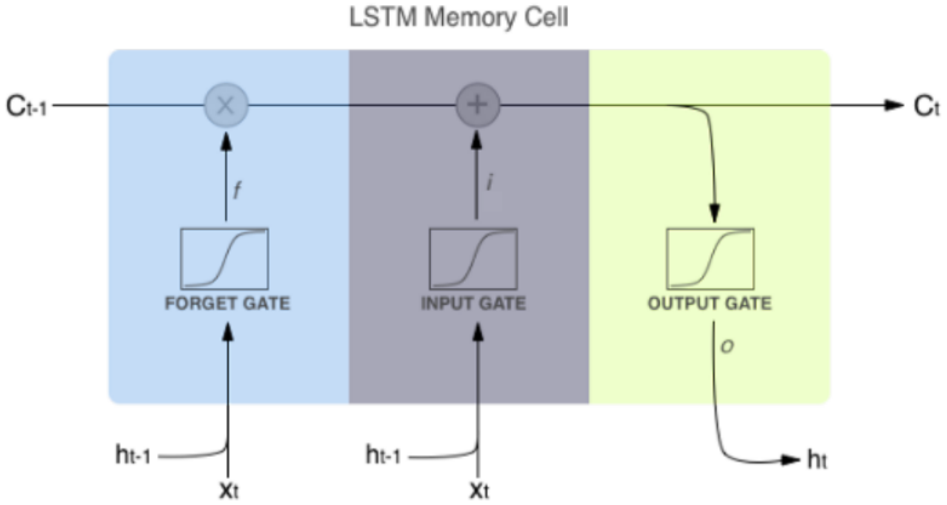


Figura 3.4: Desglose del funcionamiento de una **RNN** [7].

RNN


LSTM Memory Cell
Ct-1
Ct
f
i
o
FORGET GATE
INPUT GATE
OUTPUT GATE
ht-1
ht-1
xt
xt
ht

RNN

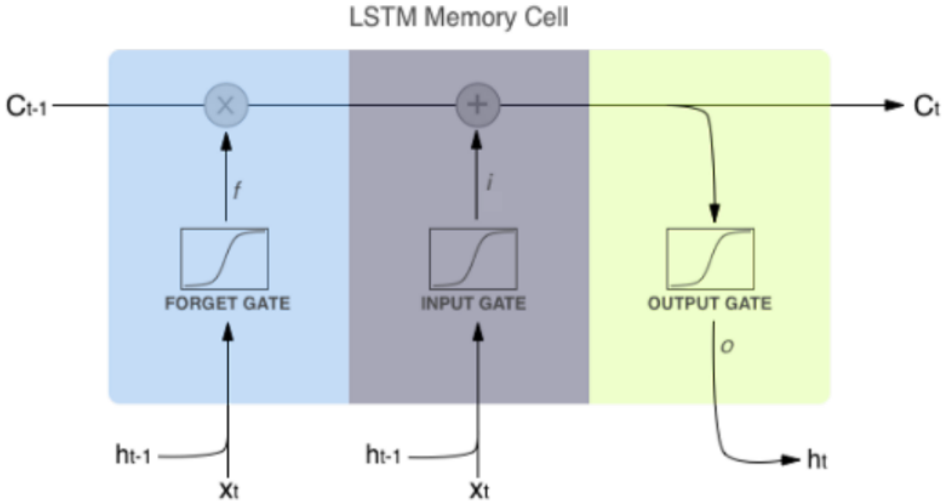


$$i_t = \sigma\left(w_i\left[h_{t-1}, x_t\right] + b_i\right)$$

$$f_t = \sigma\left(w_f\left[h_{t-1}, x_t\right] + b_f\right)$$

$$o_t = \sigma\left(w_o\left[h_{t-1}, x_t\right] + b_o\right)$$

RNN



$$i_t = \sigma\left(w_i\left[h_{t-1}, x_t\right] + b_i\right)$$

$$f_t = \sigma\left(w_f\left[h_{t-1}, x_t\right] + b_f\right)$$

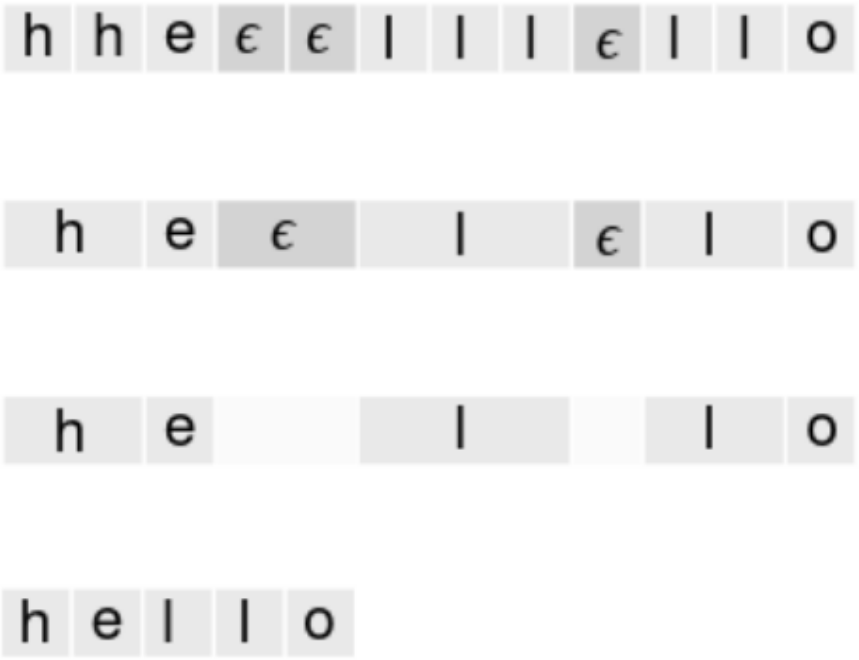
$$o_t = \sigma\left(w_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$\tilde{c}_t = \tanh\left(w_c\left[h_{t-1}, x_t\right] + b_c\right)$$

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t$$

$$h_t = o_t * \tanh\left(c_t\right)$$

RNN



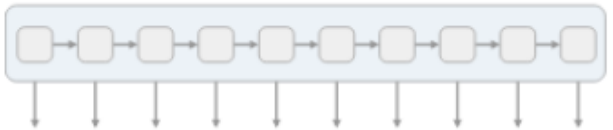
RNN

h h e ϵ ϵ l l l ϵ l l o

h e ϵ l ϵ l o

h e l l o

h e l l o



h	h	h	h	h	h	h	h	h	h
e	e	e	e	e	e	e	e	e	e
l	l	l	l	l	l	l	l	l	l
o	o	o	o	o	o	o	o	o	o
ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ

h	e	ϵ	l	l	ϵ	l	l	o	o
h	h	e	l	l	ϵ	ϵ	l	ϵ	o
ϵ	e	ϵ	l	l	ϵ	ϵ	l	o	o

h	e	l	l	o
e	l	l	o	
h	e	l	o	

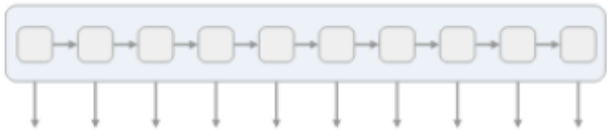
RNN

h	h	e	ϵ	ϵ	l	l	l	ϵ	l	l	o

h	e	ϵ	l	ϵ	l	o

h	e	l	l	o

$$p(\mathbf{Y}|\mathbf{X}) = \sum_{A \in \mathcal{A}_{X,Y}} \prod_{t=1}^{T} p(\alpha_t|\mathbf{X})$$



h	h	h	h	h	h	h	h	h	h
e	e	e	e	e	e	e	e	e	e
l	l	l	l	l	l	l	l	l	l
o	o	o	o	o	o	o	o	o	o
ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ

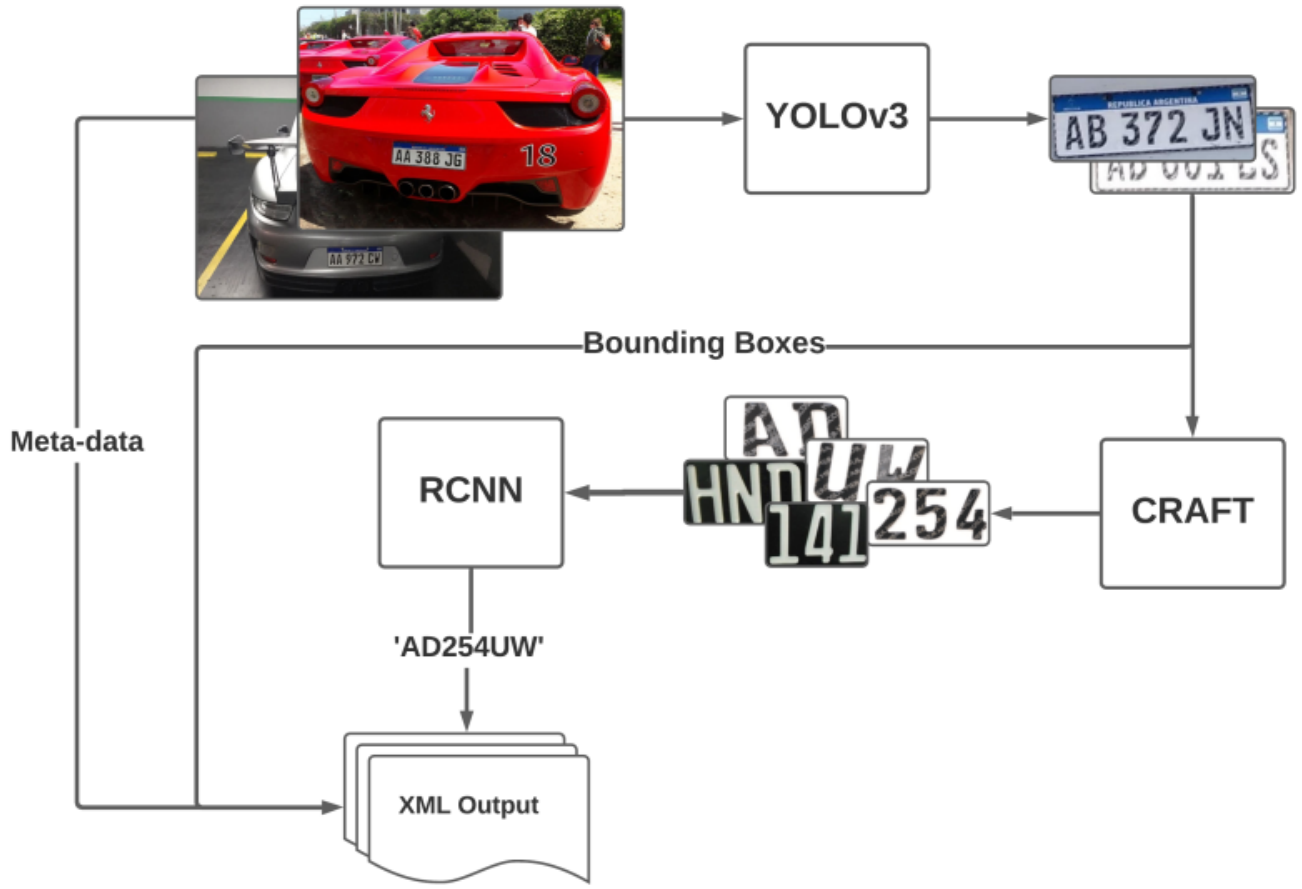
h	e	ϵ	l	l	ϵ	l	l	o	o
h	h	e	l	l	ϵ	ϵ	l	ϵ	o
ϵ	e	ϵ	l	l	ϵ	ϵ	l	o	o

h	e	l	l	o
e	l	l	o	
h	e	l	o	

RNN

- 50 epochs
- Adam
- Learning rate: $1x10^{-3}$
- $\beta_1 = \beta_2 = 9x10^{-1}$
- $\epsilon = 1x10^{-8}$
- Decay: $5x10^{-3}$
- Batch de entrenamiento: 128 imágenes.
- Sin aumentación de datos.
- Imágenes sintéticas ($1{,}5x10^{5}$). **Padding**.

Lay-Out


YOLOv3
Bounding Boxes
Meta-data
RCNN
CRAFT
'AD254UW'
XML Output

CRAFT

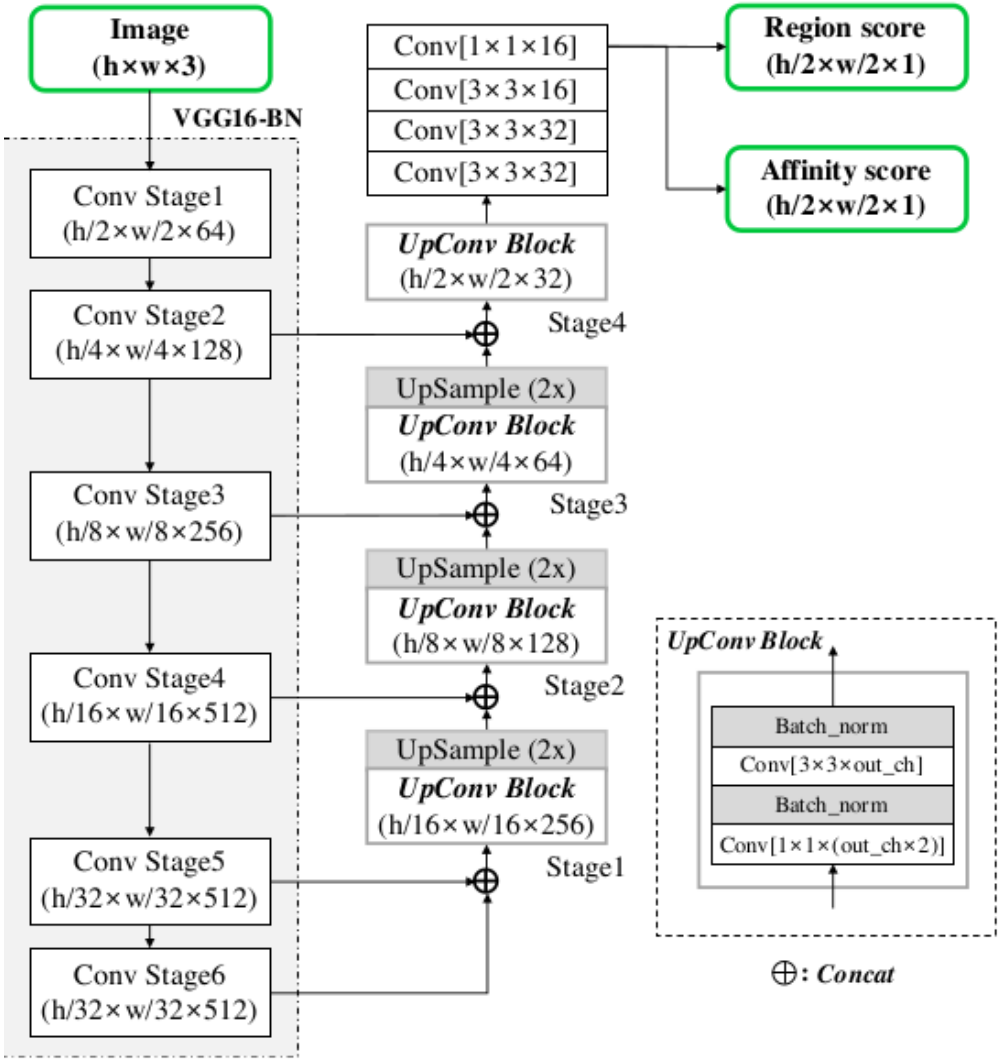


Image
(h×w×3)
VGG16-BN
Conv Stage1
(h/2×w/2×64)
Conv Stage2
(h/4×w/4×128)
Conv Stage3
(h/8×w/8×256)
Conv Stage4
(h/16×w/16×512)
Conv Stage5
(h/32×w/32×512)
Conv Stage6
(h/32×w/32×512)
Conv[1×1×16]
Conv[3×3×16]
Conv[3×3×32]
Conv[3×3×32]
Region score
(h/2×w/2×1)
Affinity score
(h/2×w/2×1)
UpConv Block
(h/2×w/2×32)
Stage4
UpSample (2x)
UpConv Block
(h/4×w/4×64)
Stage3
UpSample (2x)
UpConv Block
(h/8×w/8×128)
Stage2
UpSample (2x)
UpConv Block
(h/16×w/16×256)
Stage1
UpConv Block
Batch_norm
Conv[3×3×out_ch]
Batch_norm
Conv[1×1×(out_ch×2)]
⊕: Concat

CRAFT

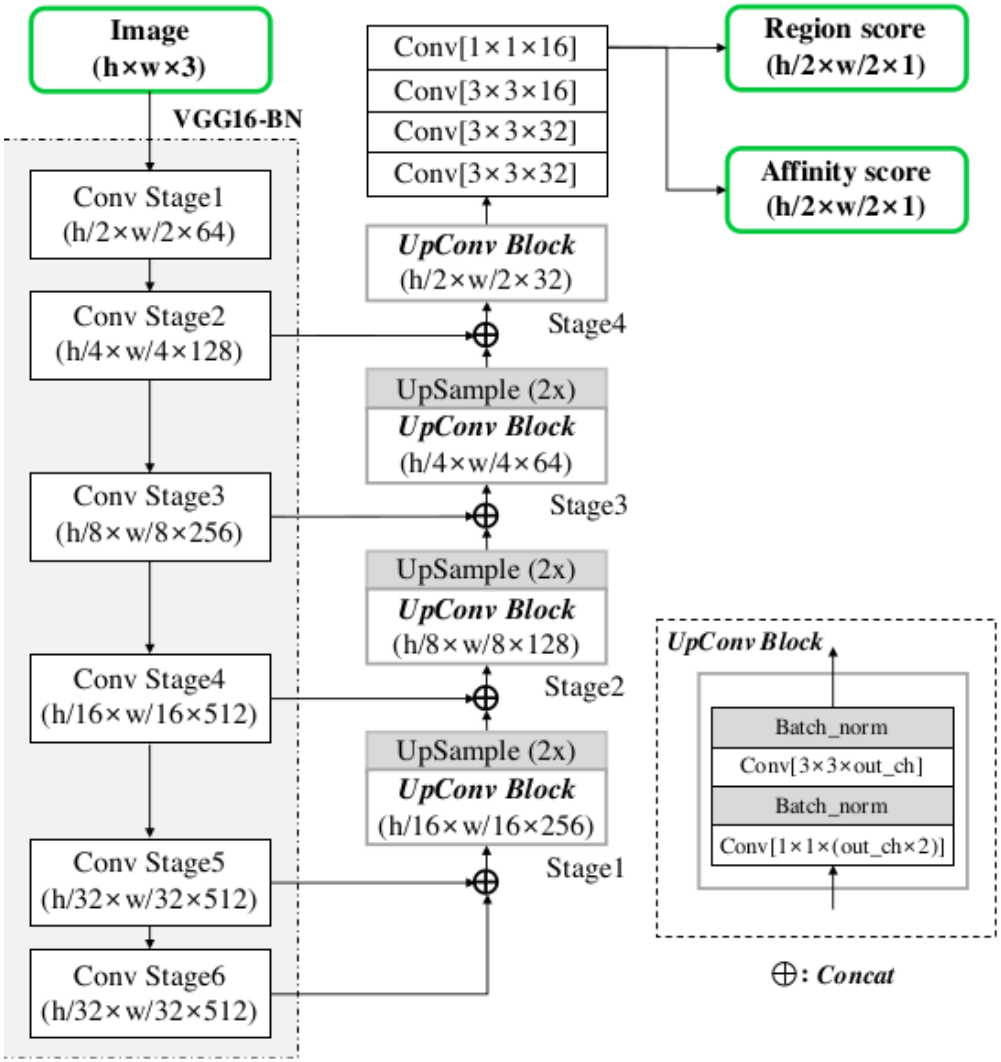
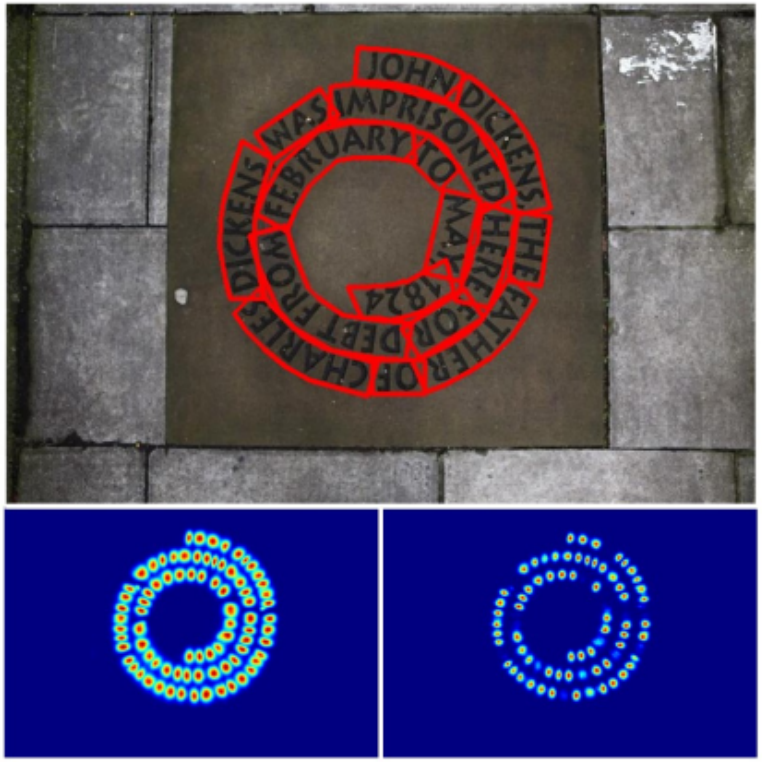


Image
(h×w×3)
VGG16-BN
Conv Stage1
(h/2×w/2×64)
Conv Stage2
(h/4×w/4×128)
Conv Stage3
(h/8×w/8×256)
Conv Stage4
(h/16×w/16×512)
Conv Stage5
(h/32×w/32×512)
Conv Stage6
(h/32×w/32×512)
Conv[1×1×16]
Conv[3×3×16]
Conv[3×3×32]
Conv[3×3×32]
Region score
(h/2×w/2×1)
Affinity score
(h/2×w/2×1)
UpConv Block
(h/2×w/2×32)
Stage4
UpSample (2x)
UpConv Block
(h/4×w/4×64)
Stage3
UpSample (2x)
UpConv Block
(h/8×w/8×128)
Stage2
UpSample (2x)
UpConv Block
(h/16×w/16×256)
Stage1
UpConv Block
Batch_norm
Conv[3×3×out_ch]
Batch_norm
Conv[1×1×(out_ch×2)]
⊕: Concat


JOHN DICKENS WAS IMPRISONED HERE FOR A FEBRUARY TO MAY 1824 DEBT FROM CHARLES DICKENS

CRAFT


Ford
REPUBLICA ARGENTINA
AA 021 GF


GOOD
HOME MADE
FOOD
CASK CONDITIONED
ALES
GARDEN

CRAFT

Tools

NumPy

matplotlib

Keras

Tools

```
enmariotti@raxamd:~/CV$ tree . -d
.
├── binder
├── data
│   ├── footage
│   │   ├── aerial_footage
│   │   ├── street_footage
│   │   └── text_footage
│   └── images
│       ├── test
│       └── train
│           ├── 00negative
│           ├── cars
│           └── text
├── dev
│   ├── cam_examples
│   │   └── tests
│   └── yolo_examples
├── doc
├── models
│   └── __pycache__
├── models-checkpoints
│   └── cam_model
│       ├── resnet50_car-text_non-freeze
│       └── resnet50_text_non-freeze
└── src
    └── __pycache__

25 directories
```

Hardware

CPU: 2 x Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz 8 cores por procesador, 16 subprocesos.

MEM: 32 GB DDR2

GPU: NVIDIA Tesla K-40 12 GB GDRR5

GPU: NVIDIA Quadro M4000 8GB GDDR5

INVAP

Dataset

Figura 4.1: Imagen original sin modificar.

Dataset

Figura 4.2: Imagen modificada con **Albumentations** [12].

Figura 4.3: Imagen modificada con **Albumentations** [12].

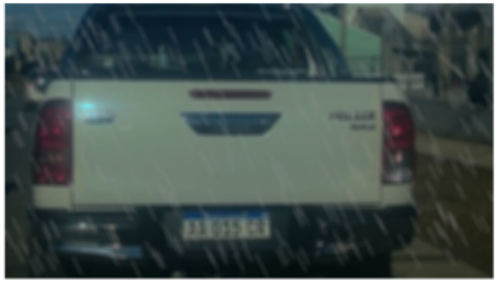


Figura 4.4: Imagen modificada con **Albumentations** [12].

Figura 4.1: Imagen original sin modificar.

Metricas

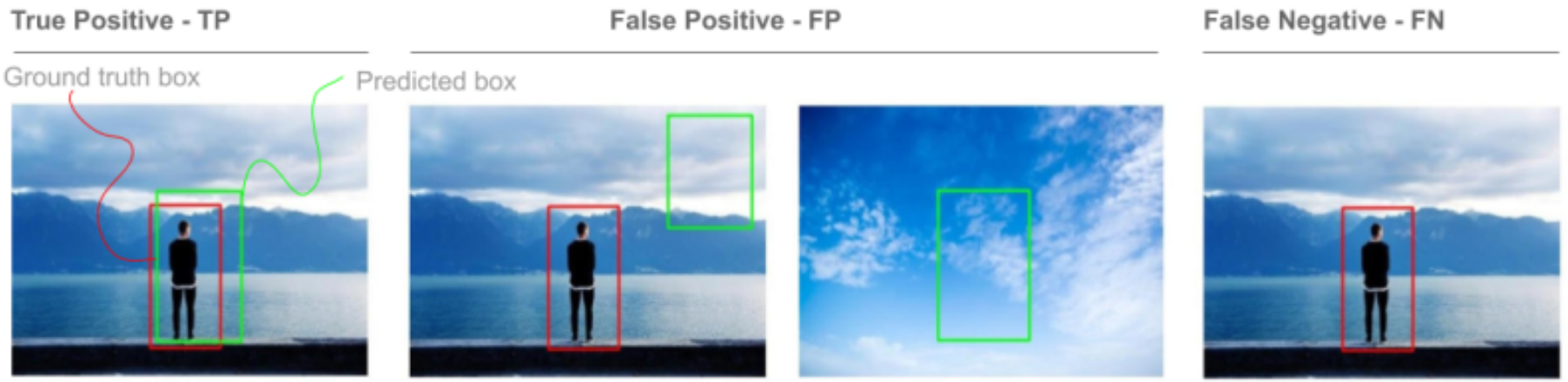


Figura 4.5: Ejemplo de clasificación en categorías no triviales [13].

Metricas

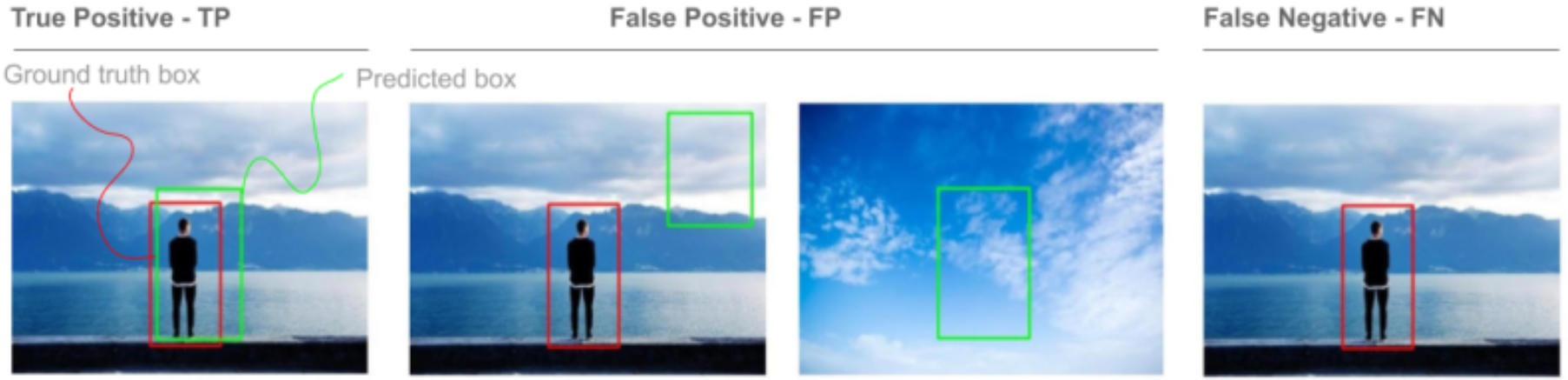


Figura 4.5: Ejemplo de clasificación en categorías no triviales [13].

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \qquad Precision = \frac{TP}{TP + FP} = \frac{true\ object\ detection}{all\ detected\ boxes}$$

$$Recall = \frac{TP}{TP + FN} = \frac{true\ object\ detection}{all\ ground\ truth\ boxes}$$

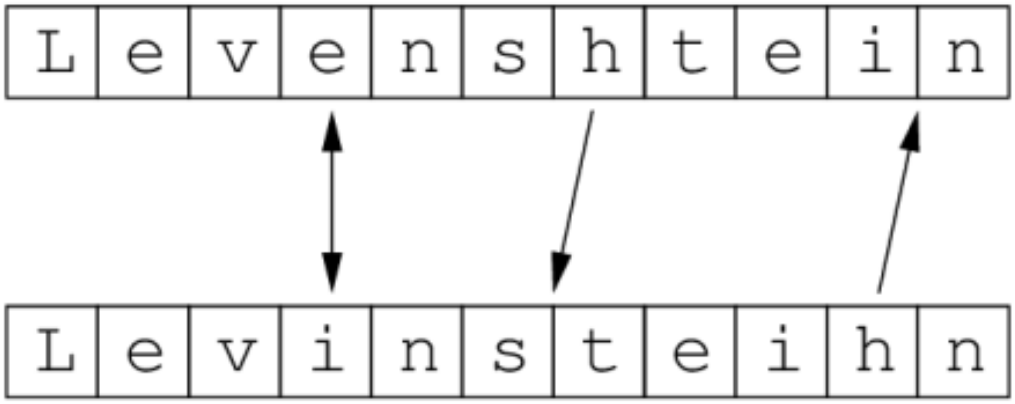
Metricas

Predictions: [['aa', '205', 'nx'], ['gdt', '465']]

Metricas

Text Distance Infographics

CATEGORY	APPROACH	⊕ PROS	⊖ CONS	ALGORITHM	AREAS
Edit based Similarities	compare two strings by counting the minimum number of operations required to transform one string into the other	+ Simplicity + Good for short strings	− Inefficient for longer strings − Computationally expensive − Doesn't take into account the semantic meaning	Hamming Levenshtein Damerau-Levenshtein Jaro-Winkler Strcmp95 Needleman-Wunsch Gotoh Smith-Waterman MLIPNS	Spelling correction Duplication search DNA analysis Correction systems for OCR Measuring the linguistic distance of the languages



Resultados

Dataset	Pipeline				OpenALPR			
	Accuracy	Precision	Recall	LS*	Accuracy	Precision	Recall	LS*
Normal	0.38	0.6	0.52	0.46	0.40	0.83	0.43	0.17
Soleado	0.55	0.78	0.65	0.34	0.44	0.79	0.50	0.21
Lluvioso	0.16	0.35	0.23	0.36	0.34	0.75	0.39	0.18
Ruidoso	0.42	0.48	0.76	0.49	0.61	0.90	0.65	0.16

Tabla 4.1: Comparación de resultados. (*) Promedio de la similitud normalizada de *Levenshtein*.

Resultados

Predictions: [['castle', 'stirling', 'lodgings', 'argylls']]

Predictions: [['mea', '959']]

Resultados

Resultados

(VIDEO & OUTPUT)

Conclusiones

- Resolvimos el problema **ALPR** con un sistema modular del tipo *detección-detección-reconocimiento* usando deep learning.
- Evaluamos nuestro sistema frente al estandar comercial, **OpenALPR**
- Dependiendo de la iluminación de la escena, uno u otro método entrega mejores resultados.
- Algunos resultados cualitativos atisban una transferencia buena a otras aplicaciones.

Mejoras

- Re-entrenar YOLOv3 con un dataset de patentes locales para mejorar su desempeño.
- Re-entrenar la RCNN con un pre-filtro convolucional a la entrada, sobre imágenes de sentencias de matriculas de origen natural anotadas.
- Implementar otros mecanismos de atención y entrenar.

¡GRACIAS!

....

¿PREGUNTAS?