COMPUTER VISION:
APLICACIONES Y TÉCNICAS UTILIZANDO RECONOCIMIENTO VISUAL

JOSEPH HOWSE

PRESIDENT AT NUMMIST MEDIA CORPORATION LIMITED

josephhowse@nummist.com

Nummist Media – https://nummist.com/

Saturdays.AI
Panamá

¿Quién es Joe?

- Desarrollador de aplicaciones, librerías y sistemas
- Autor de libros sobre la visión artificial (“computer vision”)
- Presidente de Nummist Media, una empresa de consultoría.

Yo entreno mi computadora y mis gatos me entrenan a mí.

Co-organizer
Member

Saturdays.AI
Panamá

¿Qué es Reconocimiento Visual?

- Es cualquier proceso de tomar datos visuales como entrada y generar una descripción “inteligente” como salida.
- Hoy hablamos de visión artificial.
- Los datos de entrada pueden ser los valores de píxeles en una imagen.

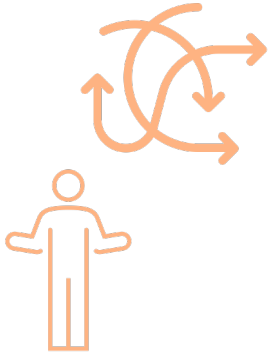
<u>Las salidas pueden ser:</u>

- Puntuaciones de clasificación
- p.ej. “persona (confianza del 99%) y uniforme (confianza del 80%).”
- Posiciones en la imagen
- Poses 3D en el espacio real
- etc.

Ejemplos de problemas

Clasificación de imágenes

- Queremos conocer las clases que describen la imagen completa.
- p.ej. Etiquetar fotos: “Si la gente busca esta foto, ¿qué palabras clave utilizará?”



Detección de objetos

- Queremos conocer la cantidad, las clases y las posiciones de los objetos en la imagen.
- p.ej. "¿Cuántas personas hay en el centro comercial y dónde están?”

Segmentación de instancias

- Además de la detección, queremos conocer qué píxeles forman parte de qué objeto.
- p.ej. “¿Hay algún equipaje desatendido y cuál es su forma?”

Estimación de pose 3D

- Además de la detección, queremos conocer qué píxeles forman parte de qué objeto.
- p.ej. “¿Hay algún equipaje desatendido y cuál es su forma?”

Ejemplos de soluciones

Código de barras de dos dimensiones

- QR, ArUco, etc.
- Son un sistema de identificación muy practico para objetos producidos en masa o objetos en condiciones controladas (laboratorio, fábrica, estudio de filmación, etc.).

Asociación de puntos característicos o descriptores (“keypoint matching”, “descriptor matching”)

- Haar, HOG, ORB, SIFT, SURF, etc.
- Mapas de contornos (para objetos no rígidos): PCA, ASM, AAM
- Calibración de cámara (para estimar el pose 3D)
- Estas técnicas son populares para la realidad aumentada.
- Más información y código de muestra: **https://github.com/JoeHowse/VisualizingTheInvisible/**

Redes neuronales profundas (“Deep neural networks”, DNNs)

- Son muy versátiles pero pueden ser opacan (una caja negra) y computacionalmente caras.
- Más información y código de muestra: **https://github.com/JoeHowse/Image-Classification-Demos/**

Problemas y sus soluciones más típicas

	Clasificación de imágenes	Detección de objetos	Segmentación de instancias	Estimación de pose 3D
Códigos de barras de dos dimensiones		✓		✓
Asociación de puntos característicos		✓		✓
Redes neuronales profundas	✓	✓	✓	

Caso de Uso

U-Count: una aplicación por mi Cliente, Market Beat

- Es un sistema para contar personas cerca de centros comerciales.
- Combina múltiples técnicas de visión artificial.
- Utiliza hardware de bajo costo.
- Desarrollado en El Salvador
- Se utiliza en Centroamérica, Panamá, Estados Unidos, etc.

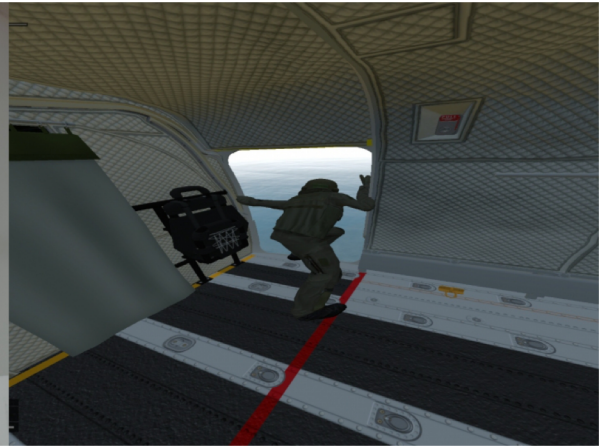
http://www.market-beat.com/ https://www.facebook.com/marketbeat/

Caso de Uso

Simulación del helicóptero CH-47: una aplicación por mi cliente, Bluedrop

- Es una simulación de entrenamiento de realidad virtual.
- Combina múltiples técnicas para estimar el pose 3D del cuerpo, cabeza y manos de cada de los usuarios.
- Desarrollado en Canadá

Realidad



Virtualidad

https://bisimulations.com/company/customer-showcase/vr-based-helicopter-crew-training/

Hablemos más sobre DNNs…

¿Porqué es un tema popular?

Topología de DNNs

- Un DNN tiene **múltiples capas** ocultas (entre la capa de entrada y la capa de salida).
- Las capas ocultas representan características "**aprendidas**".
- Podemos **afinar ("fine-tune")** un modelo sin cambiar todas las capas ocultas.
- p.ej. Con **pocos datos** nuevos, podemos cambiar las clases de salida.
- Por eso, es útil evaluar **modelos pre-entrenados** incluso si sus clases no son pertinentes para nuestra aplicación.
- El modelo pre-entrenado puede servir como **"columna vertebral" ("backbone")** de un nuevo modelo con nuevas clases.
- Hoy vamos **experimentar** un poco con modelos pre-entrenados sin afinarlos.

Experimento con Gatos

Clasificación por IBM Watson Visual Recognition

- color gris (93.3%)
- gato (84.0%)
- gato doméstico (55.6%)
- gato callejero (50.0%)
- color marrón claro (67.5%)
- felpudo (62.9%)
- umbral de la puerta (52.5%)

Código de muestra:
https://github.com/JoeHowse/Image-Classification-Demos

Experimento con Gatos

Detección por MobileNet SSD en OpenCV

- dos gatos

Código de muestra:
https://github.com/JoeHowse/Image-Classification-Demos

Experimento con Gatos

Segmentación por Mask R-CNN en PyTorch

- cuatro gatos

Código de muestra:
https://github.com/JoeHowse/Image-Classification-Demos

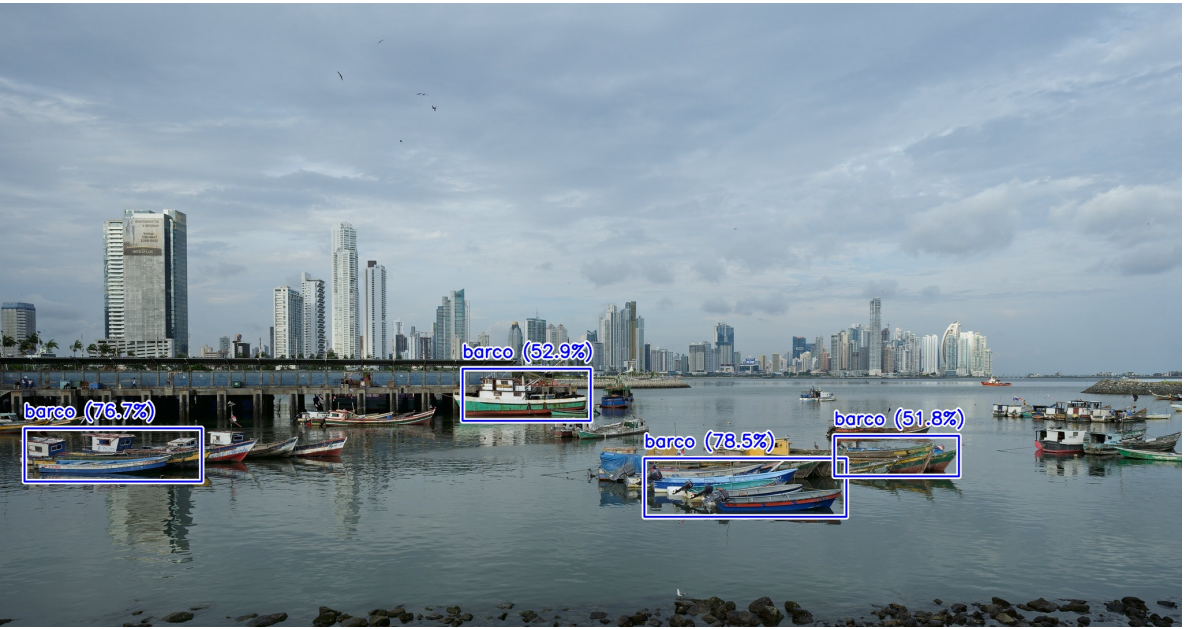
Experimento con la Bahía

Clasificación por IBM Watson Visual Recognition

- color azul (98.4%)
- puerto (80.2%)
- pista (79.9%)
- embarcaciones (79.1%)
- costa (78.8%)
- orilla de lago (56.3%)
- frente marítimo (50.0%)

Código de muestra:
https://github.com/JoeHowse/Image-Classification-Demos

Experimento con la Bahía



Detección por MobileNet SSD en OpenCV

- cuatro barcos

Código de muestra:
https://github.com/JoeHowse/Image-Classification-Demos

Experimento con la Bahía

Segmentación por Mask R-CNN en PyTorch

- veinte barcos

Código de muestra:
https://github.com/JoeHowse/Image-Classification-Demos

Experimento con Personas

Clasificación por IBM Watson Visual Recognition

- color rojo alazarino (96.6%)
- acera (69.5%)
- persona (60.0%)
- calle (50.0%)
- carretera (50.0%)

Código de muestra:
https://github.com/JoeHowse/Image-Classification-Demos

Experimento con Personas

Clasificación por IBM Watson Visual Recognition

Rojo alazarino (“alizarin crimson”):

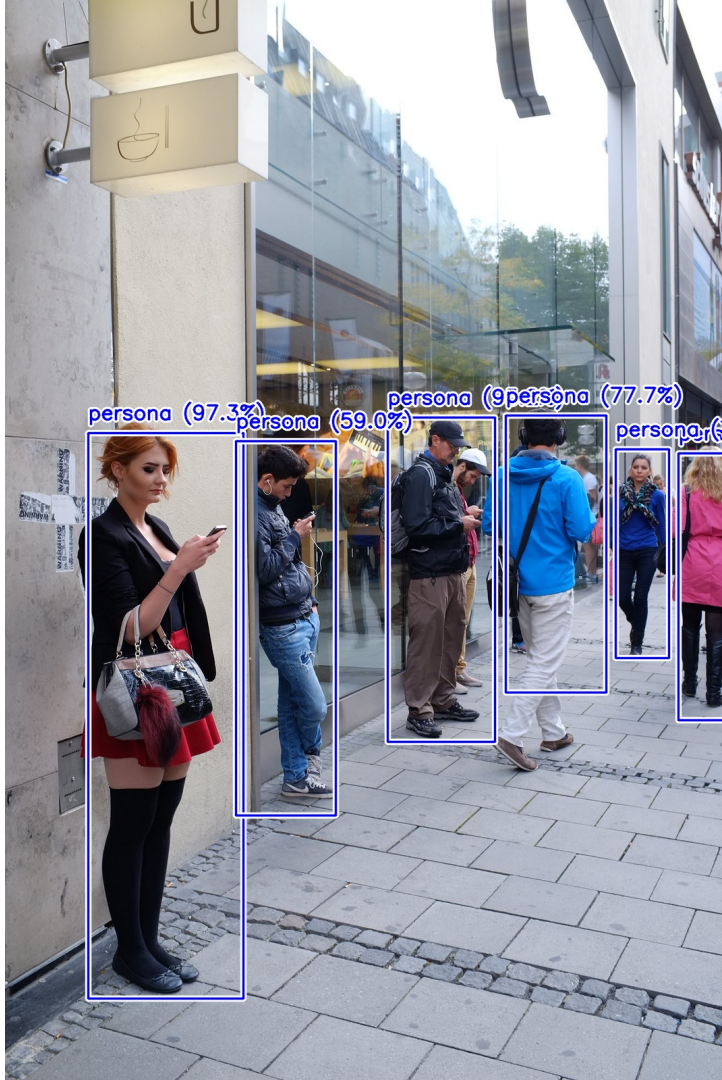
https://www.winsornewton.com/row/articles/colours/spotlight-on-alizarin-crimson/

Código de muestra:
https://github.com/JoeHowse/Image-Classification-Demos

Experimento con Personas

Detección por MobileNet SSD en OpenCV

- seis personas



Código de muestra:
https://github.com/JoeHowse/Image-Classification-Demos

Experimento con Personas

Segmentación por Mask R-CNN en PyTorch

- ocho personas
- tres bolsos
- un celular
- una mochila

Código de muestra:
https://github.com/JoeHowse/Image-Classification-Demos

Experimento con flores

Clasificación por IBM Watson Visual Recognition

- flor allamanda (96.3%)
- color verde (90.8%)
- color amarillo limón (90.4%)
- planta allamanda común (50.0%)

Código de muestra:
https://github.com/JoeHowse/Image-Classification-Demos

Ahora miremos el código…

¿Cómo se hace?

Código Python: Clasificación por Watson

```
import os
from ibm_watson import VisualRecognitionV3

threshold = 0.5

service = VisualRecognitionV3(
    '2018-03-19',
    url=os.environ['WATSON_VISUAL_RECOGNITION_URL'],
    iam_apikey=os.environ['WATSON_VISUAL_RECOGNITION_KEY'])

images_path = os.path.abspath('resources/images.zip')
with open(images_path, 'rb') as image_file:
    classify_result = service.classify(
        images_file=image_file,
        threshold='0.1',
        classifier_ids=['default']).get_result()

# Los resultados son un diccionario JSON.

for image in classify_result['images']:
    filename = image['image']
    classes = image['classifiers'][0]['clases']
    for c in classes:
        score = c['score']
        if score >= threshold:
            ...  # Hace algo con esta clase.
```

Importa la librería.

Código Python: Clasificación por Watson

```
import os
from ibm_watson import VisualRecognitionV3

threshold = 0.5

service = VisualRecognitionV3(
    '2018-03-19',
    url=os.environ['WATSON_VISUAL_RECOGNITION_URL'],
    iam_apikey=os.environ['WATSON_VISUAL_RECOGNITION_KEY'])

images_path = os.path.abspath('resources/images.zip')
with open(images_path, 'rb') as image_file:
    classify_result = service.classify(
        images_file=image_file,
        threshold='0.1',
        classifier_ids=['default']).get_result()

# Los resultados son un diccionario JSON.

for image in classify_result['images']:
    filename = image['image']
    classes = image['classifiers'][0]['clases']
    for c in classes:
        score = c['score']
        if score >= threshold:
            ...  # Hace algo con esta clase.
```

Define el servicio que vamos usar.

Código Python: Clasificación por Watson

```
import os
from ibm_watson import VisualRecognitionV3

threshold = 0.5

service = VisualRecognitionV3(
    '2018-03-19',
    url=os.environ['WATSON_VISUAL_RECOGNITION_URL'],
    iam_apikey=os.environ['WATSON_VISUAL_RECOGNITION_KEY'])

images_path = os.path.abspath('resources/images.zip')
with open(images_path, 'rb') as image_file:
    classify_result = service.classify(
        images_file=image_file,
        threshold='0.1',
        classifier_ids=['default']).get_result()

# Los resultados son un diccionario JSON.

for image in classify_result['images']:
    filename = image['image']
    classes = image['classifiers'][0]['clases']
    for c in classes:
        score = c['score']
        if score >= threshold:
            ...  # Hace algo con esta clase.
```

Envía imágenes en un archivo ZIP a Watson y obtiene resultados de clasificación.

Código Python: Clasificación por Watson

```
import os
from ibm_watson import VisualRecognitionV3

threshold = 0.5

service = VisualRecognitionV3(
    '2018-03-19',
    url=os.environ['WATSON_VISUAL_RECOGNITION_URL'],
    iam_apikey=os.environ['WATSON_VISUAL_RECOGNITION_KEY'])

images_path = os.path.abspath('resources/images.zip')
with open(images_path, 'rb') as image_file:
    classify_result = service.classify(
        images_file=image_file,
        threshold='0.1',
        classifier_ids=['default']).get_result()

# Los resultados son un diccionario JSON.

for image in classify_result['images']:
    filename = image['image']
    classes = image['classifiers'][0]['clases']
    for c in classes:
        score = c['score']
        if score >= threshold:
            ...  # Hace algo con esta clase.
```

Analiza los resultados en un bucle.

Código Python: Detección por MobileNet SSD en OpenCV

Importa la librería.

```
import cv2

model = cv2.dnn.readNetFromCaffe('objects_data/MobileNetSSD_deploy.prototxt’,
    'objects_data/MobileNetSSD_deploy.caffemodel')
blob_height = 300
color_scale = 1.0/127.5
average_color = (127.5, 127.5, 127.5)
confidence_threshold = 0.5

labels = ['avión', 'bicicleta', 'ave', 'barco', ...]  # Nombres de las clases en el modelo

input_filename = 'resources/image.jpg'
image = cv2.imread(input_filename)

h, w = image.shape[:2]
aspect_ratio = w/h

# Detecta objetos.

blob_width = int(blob_height * aspect_ratio)
blob_size = (blob_width, blob_height)

blob = cv2.dnn.blobFromImage(image, scalefactor=color_scale, size=blob_size, mean=average_color)

model.setInput(blob)
results = model.forward()

for object in results[0, 0]:
    confidence = object[2]
    if confidence >= confidence_threshold:

        x0, y0, x1, y1 = (object[3:7] * [w, h, w, h]).astype(int)  # Rectángulo delimitador del objeto

        id = int(object[1])  # Clase del objeto
        label = labels[id - 1]  # Nombre de la clase del objeto

        ...  # Hace algo con este objeto.
```

Código Python: Detección por MobileNet SSD en OpenCV

```
import cv2

model = cv2.dnn.readNetFromCaffe('objects_data/MobileNetSSD_deploy.prototxt',
    'objects_data/MobileNetSSD_deploy.caffemodel')
blob_height = 300
color_scale = 1.0/127.5
average_color = (127.5, 127.5, 127.5)
confidence_threshold = 0.5

labels = ['avión', 'bicicleta', 'ave', 'barco', ...]  # Nombres de las clases en el modelo

input_filename = 'resources/image.jpg'
image = cv2.imread(input_filename)

h, w = image.shape[:2]
aspect_ratio = w/h

# Detecta objetos.

blob_width = int(blob_height * aspect_ratio)
blob_size = (blob_width, blob_height)

blob = cv2.dnn.blobFromImage(image, scalefactor=color_scale, size=blob_size, mean=average_color)

model.setInput(blob)
results = model.forward()

for object in results[0, 0]:
    confidence = object[2]
    if confidence >= confidence_threshold:

        x0, y0, x1, y1 = (object[3:7] * [w, h, w, h]).astype(int)  # Rectángulo delimitador del objeto

        id = int(object[1])  # Clase del objeto
        label = labels[id - 1]  # Nombre de la clase del objeto

        ...  # Hace algo con este objeto.
```

Importa el modelo y define sus parámetros.

Código Python: Detección por MobileNet SSD en OpenCV

```
import cv2

model = cv2.dnn.readNetFromCaffe('objects_data/MobileNetSSD_deploy.prototxt',
    'objects_data/MobileNetSSD_deploy.caffemodel')
blob_height = 300
color_scale = 1.0/127.5
average_color = (127.5, 127.5, 127.5)
confidence_threshold = 0.5

labels = ['avión', 'bicicleta', 'ave', 'barco', ...]  # Nombres de las clases en el modelo

input_filename = 'resources/image.jpg'
image = cv2.imread(input_filename)

h, w = image.shape[:2]
aspect_ratio = w/h

# Detecta objetos.

blob_width = int(blob_height * aspect_ratio)
blob_size = (blob_width, blob_height)

blob = cv2.dnn.blobFromImage(image, scalefactor=color_scale, size=blob_size, mean=average_color)

model.setInput(blob)
results = model.forward()

for object in results[0, 0]:
    confidence = object[2]
    if confidence >= confidence_threshold:

        x0, y0, x1, y1 = (object[3:7] * [w, h, w, h]).astype(int)  # Rectángulo delimitador del objeto

        id = int(object[1])  # Clase del objeto
        label = labels[id - 1]  # Nombre de la clase del objeto

        ...  # Hace algo con este objeto.
```

Define los nombres de las clases.

Código Python: Detección por MobileNet SSD en OpenCV

```
import cv2

model = cv2.dnn.readNetFromCaffe('objects_data/MobileNetSSD_deploy.prototxt',
    'objects_data/MobileNetSSD_deploy.caffemodel')
blob_height = 300
color_scale = 1.0/127.5
average_color = (127.5, 127.5, 127.5)
confidence_threshold = 0.5

labels = ['avión', 'bicicleta', 'ave', 'barco', ...]  # Nombres de las clases en el modelo

input_filename = 'resources/image.jpg'
image = cv2.imread(input_filename)

h, w = image.shape[:2]
aspect_ratio = w/h

# Detecta objetos.

blob_width = int(blob_height * aspect_ratio)
blob_size = (blob_width, blob_height)

blob = cv2.dnn.blobFromImage(image, scalefactor=color_scale, size=blob_size, mean=average_color)

model.setInput(blob)
results = model.forward()

for object in results[0, 0]:
    confidence = object[2]
    if confidence >= confidence_threshold:

        x0, y0, x1, y1 = (object[3:7] * [w, h, w, h]).astype(int)  # Rectángulo delimitador del objeto

        id = int(object[1])  # Clase del objeto
        label = labels[id - 1]  # Nombre de la clase del objeto

        ...  # Hace algo con este objeto.
```

Carga una imagen.

Código Python: Detección por MobileNet SSD en OpenCV

```
import cv2

model = cv2.dnn.readNetFromCaffe('objects_data/MobileNetSSD_deploy.prototxt',
    'objects_data/MobileNetSSD_deploy.caffemodel')
blob_height = 300
color_scale = 1.0/127.5
average_color = (127.5, 127.5, 127.5)
confidence_threshold = 0.5

labels = ['avión', 'bicicleta', 'ave', 'barco', ...]  # Nombres de las clases en el modelo

input_filename = 'resources/image.jpg'
image = cv2.imread(input_filename)

h, w = image.shape[:2]
aspect_ratio = w/h

# Detecta objetos.

blob_width = int(blob_height * aspect_ratio)
blob_size = (blob_width, blob_height)

blob = cv2.dnn.blobFromImage(image, scalefactor=color_scale, size=blob_size, mean=average_color)

model.setInput(blob)
results = model.forward()

for object in results[0, 0]:
    confidence = object[2]
    if confidence >= confidence_threshold:

        x0, y0, x1, y1 = (object[3:7] * [w, h, w, h]).astype(int)  # Rectángulo delimitador del objeto

        id = int(object[1])  # Clase del objeto
        label = labels[id - 1]  # Nombre de la clase del objeto

        ...  # Hace algo con este objeto.
```

Convierte la imagen en un tensor.

Código Python: Detección por MobileNet SSD en OpenCV

```
import cv2

model = cv2.dnn.readNetFromCaffe('objects_data/MobileNetSSD_deploy.prototxt’,
    'objects_data/MobileNetSSD_deploy.caffemodel')
blob_height = 300
color_scale = 1.0/127.5
average_color = (127.5, 127.5, 127.5)
confidence_threshold = 0.5

labels = ['avión', 'bicicleta', 'ave', 'barco', ...]  # Nombres de las clases en el modelo

input_filename = 'resources/image.jpg'
image = cv2.imread(input_filename)

h, w = image.shape[:2]
aspect_ratio = w/h

# Detecta objetos.

blob_width = int(blob_height * aspect_ratio)
blob_size = (blob_width, blob_height)

blob = cv2.dnn.blobFromImage(image, scalefactor=color_scale, size=blob_size, mean=average_color)

model.setInput(blob)
results = model.forward()

for object in results[0, 0]:
    confidence = object[2]
    if confidence >= confidence_threshold:

        x0, y0, x1, y1 = (object[3:7] * [w, h, w, h]).astype(int)  # Rectángulo delimitador del objeto

        id = int(object[1])  # Clase del objeto
        label = labels[id - 1]  # Nombre de la clase del objeto

        ...  # Hace algo con este objeto.
```

Detecta objetos en la imagen.

Código Python: Detección por MobileNet SSD en OpenCV

```
import cv2

model = cv2.dnn.readNetFromCaffe('objects_data/MobileNetSSD_deploy.prototxt’,
    'objects_data/MobileNetSSD_deploy.caffemodel')
blob_height = 300
color_scale = 1.0/127.5
average_color = (127.5, 127.5, 127.5)
confidence_threshold = 0.5

labels = ['avión', 'bicicleta', 'ave', 'barco', ...]  # Nombres de las clases en el modelo

input_filename = 'resources/image.jpg'
image = cv2.imread(input_filename)

h, w = image.shape[:2]
aspect_ratio = w/h

# Detecta objetos.

blob_width = int(blob_height * aspect_ratio)
blob_size = (blob_width, blob_height)

blob = cv2.dnn.blobFromImage(image, scalefactor=color_scale, size=blob_size, mean=average_color)

model.setInput(blob)
results = model.forward()

for object in results[0, 0]:
    confidence = object[2]
    if confidence >= confidence_threshold:

        x0, y0, x1, y1 = (object[3:7] * [w, h, w, h]).astype(int)  # Rectángulo delimitador del objeto

        id = int(object[1])  # Clase del objeto
        label = labels[id - 1]  # Nombre de la clase del objeto

        ...  # Hace algo con este objeto.
```

Analiza los resultados en un bucle.

Código Python: Segmentación por

Mask R-CNN en PyTorch

```
import numpy
import PIL
import torchvision


USE_CUDA = True
THRESHOLD = 0.75

model = torchvision.models.detection.maskrcnn_resnet50_fpn(pretrained=True)
if USE_CUDA: model.cuda()
model.eval()

LABELS = ['__background__', 'persona', 'bicicleta', 'coche', ...] # Nombres de las clases en el modelo

input_filename = 'resources/image.jpg'
transform = torchvision.transforms.Compose([torchvision.transforms.ToTensor()])
tensor = transform(PIL.Image.open(input_filename))
if USE_CUDA: tensor = tensor.cuda()

# Segmenta instancias.
predictions = model([tensor])[0]

# Obtena resultados como listas.
scores = list(predictions['scores'].detach().cpu().numpy())
scores = [score for score in scores if score > THRESHOLD]
masks = (predictions['masks'] > 0.5).squeeze().detach().cpu().numpy()
labels = [LABELS[i] for i in list(predictions['labels'].detach().cpu().numpy())]
boxes = [[(int(box[0]), int(box[1])), (int(box[2]), int(box[3]))]
         for box in list(predictions['boxes'].detach().cpu().numpy())]

for i, (score, mask, label, box) in enumerate(zip(scores, masks, labels, boxes)):
    if score >= THRESHOLD:
        ...  # Hace algo con esta instancia.
```

Importa la librería.

Código Python: Segmentación por

Mask R-CNN en PyTorch

```
import numpy
import PIL
import torchvision


USE_CUDA = True
THRESHOLD = 0.75

model = torchvision.models.detection.maskrcnn_resnet50_fpn(pretrained=True)
if USE_CUDA: model.cuda()
model.eval()

LABELS = ['__background__', 'persona', 'bicicleta', 'coche', ...] # Nombres de las clases en el modelo

input_filename = 'resources/image.jpg'
transform = torchvision.transforms.Compose([torchvision.transforms.ToTensor()])
tensor = transform(PIL.Image.open(input_filename))
if USE_CUDA: tensor = tensor.cuda()

# Segmenta instancias.
predictions = model([tensor])[0]

# Obtena resultados como listas.
scores = list(predictions['scores'].detach().cpu().numpy())
scores = [score for score in scores if score > THRESHOLD]
masks = (predictions['masks'] > 0.5).squeeze().detach().cpu().numpy()
labels = [LABELS[i] for i in list(predictions['labels'].detach().cpu().numpy())]
boxes = [[(int(box[0]), int(box[1])), (int(box[2]), int(box[3]))]
         for box in list(predictions['boxes'].detach().cpu().numpy())]

for i, (score, mask, label, box) in enumerate(zip(scores, masks, labels, boxes)):
    if score >= THRESHOLD:
        ...  # Hace algo con esta instancia.
```

Carga el modelo y habilita las optimizaciones CUDA.

Código Python: Segmentación por

Mask R-CNN en PyTorch

```
import numpy
import PIL
import torchvision


USE_CUDA = True
THRESHOLD = 0.75

model = torchvision.models.detection.maskrcnn_resnet50_fpn(pretrained=True)
if USE_CUDA: model.cuda()
model.eval()

LABELS = ['__background__', 'persona', 'bicicleta', 'coche', ...] # Nombres de las clases en el modelo

input_filename = 'resources/image.jpg'
transform = torchvision.transforms.Compose([torchvision.transforms.ToTensor()])
tensor = transform(PIL.Image.open(input_filename))
if USE_CUDA: tensor = tensor.cuda()

# Segmenta instancias.
predictions = model([tensor])[0]

# Obtena resultados como listas.
scores = list(predictions['scores'].detach().cpu().numpy())
scores = [score for score in scores if score > THRESHOLD]
masks = (predictions['masks'] > 0.5).squeeze().detach().cpu().numpy()
labels = [LABELS[i] for i in list(predictions['labels'].detach().cpu().numpy())]
boxes = [[(int(box[0]), int(box[1])), (int(box[2]), int(box[3]))]
         for box in list(predictions['boxes'].detach().cpu().numpy())]

for i, (score, mask, label, box) in enumerate(zip(scores, masks, labels, boxes)):
    if score >= THRESHOLD:
        ...  # Hace algo con esta instancia.
```

Define los nombres de las clases.

Código Python: Segmentación por

Mask R-CNN en PyTorch

```
import numpy
import PIL
import torchvision


USE_CUDA = True
THRESHOLD = 0.75

model = torchvision.models.detection.maskrcnn_resnet50_fpn(pretrained=True)
if USE_CUDA: model.cuda()
model.eval()

LABELS = ['__background__', 'persona', 'bicicleta', 'coche', ...] # Nombres de las clases en el modelo

input_filename = 'resources/image.jpg'
transform = torchvision.transforms.Compose([torchvision.transforms.ToTensor()])
tensor = transform(PIL.Image.open(input_filename))
if USE_CUDA: tensor = tensor.cuda()

# Segmenta instancias.
predictions = model([tensor])[0]

# Obtena resultados como listas.
scores = list(predictions['scores'].detach().cpu().numpy())
scores = [score for score in scores if score > THRESHOLD]
masks = (predictions['masks'] > 0.5).squeeze().detach().cpu().numpy()
labels = [LABELS[i] for i in list(predictions['labels'].detach().cpu().numpy())]
boxes = [[(int(box[0]), int(box[1])), (int(box[2]), int(box[3]))]
         for box in list(predictions['boxes'].detach().cpu().numpy())]

for i, (score, mask, label, box) in enumerate(zip(scores, masks, labels, boxes)):
    if score >= THRESHOLD:
        ...  # Hace algo con esta instancia.
```

Carga una imagen y la convierte en un tensor.

Código Python: Segmentación por

Mask R-CNN en PyTorch

```
import numpy
import PIL
import torchvision


USE_CUDA = True
THRESHOLD = 0.75

model = torchvision.models.detection.maskrcnn_resnet50_fpn(pretrained=True)
if USE_CUDA: model.cuda()
model.eval()

LABELS = ['__background__', 'persona', 'bicicleta', 'coche', ...] # Nombres de las clases en el modelo

input_filename = 'resources/image.jpg'
transform = torchvision.transforms.Compose([torchvision.transforms.ToTensor()])
tensor = transform(PIL.Image.open(input_filename))
if USE_CUDA: tensor = tensor.cuda()

# Segmenta instancias.
predictions = model([tensor])[0]

# Obtena resultados como listas.
scores = list(predictions['scores'].detach().cpu().numpy())
scores = [score for score in scores if score > THRESHOLD]
masks = (predictions['masks'] > 0.5).squeeze().detach().cpu().numpy()
labels = [LABELS[i] for i in list(predictions['labels'].detach().cpu().numpy())]
boxes = [[(int(box[0]), int(box[1])), (int(box[2]), int(box[3]))]
         for box in list(predictions['boxes'].detach().cpu().numpy())]

for i, (score, mask, label, box) in enumerate(zip(scores, masks, labels, boxes)):
    if score >= THRESHOLD:
        ...  # Hace algo con esta instancia.
```

Detecta objetos y segmenta instancias en la imagen.

Código Python: Segmentación por

Mask R-CNN en PyTorch

```
import numpy
import PIL
import torchvision


USE_CUDA = True
THRESHOLD = 0.75

model = torchvision.models.detection.maskrcnn_resnet50_fpn(pretrained=True)
if USE_CUDA: model.cuda()
model.eval()

LABELS = ['__background__', 'persona', 'bicicleta', 'coche', ...] # Nombres de las clases en el modelo

input_filename = 'resources/image.jpg'
transform = torchvision.transforms.Compose([torchvision.transforms.ToTensor()])
tensor = transform(PIL.Image.open(input_filename))
if USE_CUDA: tensor = tensor.cuda()

# Segmenta instancias.
predictions = model([tensor])[0]

# Obtena resultados como listas.
scores = list(predictions['scores'].detach().cpu().numpy())
scores = [score for score in scores if score > THRESHOLD]
masks = (predictions['masks'] > 0.5).squeeze().detach().cpu().numpy()
labels = [LABELS[i] for i in list(predictions['labels'].detach().cpu().numpy())]
boxes = [[(int(box[0]), int(box[1])), (int(box[2]), int(box[3]))]
         for box in list(predictions['boxes'].detach().cpu().numpy())]

for i, (score, mask, label, box) in enumerate(zip(scores, masks, labels, boxes)):
    if score >= THRESHOLD:
        ...  # Hace algo con esta instancia.
```

Convierte los resultados en listas y los analiza en un bucle.

Comparación de rendimiento

	Especificaciones del sistema	Tiempo para procesar una sola imagen con resolución de ~2000x1000	Uso de RAM
IBM Watson Visual Recognition	Conexión wifi con velocidad de 110Mbps	~1.253s => ~0.8 FPS	
MobileNet SSD	Ryzen 5950X	~0.012s => ~83.3 FPS	173MB
Mask R-CNN	Ryzen 5950X	~1.017s => ~1.0 FPS	4507MB
Mask R-CNN con optimizaciones CUDA	Ryzen 5950X, GeForce RTX 3080	~0.073s => ~13.7 FPS	3640MB

¿Y Jetson?

- NVIDIA promueve sus sistemas Jetson como plataforma de bajo consumo para DNNs.
- Jetson, como GeForce, ofrece optimizaciones CUDA.
- Jetson es eficiente pero en términos absolutos es mucho menos poderoso que GeForce.
- p.ej. ¿Puede Jetson ejecutar segmentación de instancias con Mask R-CNN en tiempo real? No.

	Rendimiento	Consumo	Precio
GeForce RTX 3080	238 TOPS	320W	US$699
Jetson AGX Xavier	32 TOPS	30W	US$699

TOPS = “Tensor Operations per Second”, Operaciones de Tensor por Segundo, una medida del rendimiento para IA

Recomendaciones en mi experiencia…

Consideraciones

No solamente es cuestión de procesar datos…

Los sistemas prácticos hacen muchas operaciones de entrada y salida para…

- Capturar datos de cámaras y otros sensores
- Grabar datos en el disco
- Enviar y recibir datos a través de la red
- Proporcionar una interfaz de usuario
- Los buses PCIe, USB, etc. pueden ser cuellos de botella.
- En todo tipo de trabajos de optimización, es importante identificar y solucionar los verdaderos cuellos de botella.
- p.ej. Si los buses son cuellos de botella, un coprocesador como Google Coral no ayuda, pero la compresión de datos puede ayudar.

Consideraciones

Cuál es un buen sistema operativo para este tipo de trabajo?

- A mi me gusta Manjaro (un derivado de Arch Linux).
- Manjaro (como Arch) tiene paquetes bien actualizados y bien optimizados, e incluso optimizaciones para CUDA.

- Por lo general, los desarrolladoras de DNNs usan Linux o Windows.
- CUDA no es compatible con Mac ni BSD.

Qué hemos aprendido?…

El Color Rojo Alzarino

Rojo alazarino (“alizarin crimson”):

https://www.winsornewton.com/row/articles/colours/spotlight-on-alizarin-crimson/

¿Y además?

- Los redes neuronales profundas (DNNs) pueden solucionar problemas de clasificación, detección y segmentación.

La elección del modelo es en parte una cuestión de recursos.

- ¿Hay una conexión a la nube?
- ¿Usamos un móvil, un Raspberry Pi, un Jetson, una estación de trabajo, etc.?
- ¿Se necesita procesar video en tiempo real?

Los DNNs no siempre son las mejores soluciones.

- p.ej. Para estimar el pose 3D, es mejor usar “keypoint matching” con calibración de cámara.

Saturdays.AI
Panamá

Preguntas?

Podrá ponerse en contacto a través del correo panama@saturdays.ai.

También nos puedes contactar a través de nuestras redes

https://www.saturdays.ai/projects_saturdays.html