Proyecto Final: Detección de personas en imágenes y clasificación de género desde una perspectiva cenital.

Ronald Caravaca M. y Carlos Brenes J.

Resumen - Por medio del aprendizaje de máquina, se busca aplicar un red neuronal convolucional para procesar imágenes en un subconjunto de datos restringidos desde una perspectiva cenital.

Palabras clave - Redes Regionales Convolucionales, capas ocultas, "pooling", totalmente conectadas, Detectron2 API.

I. Introducción

La predicción de atracción humana en eventos de citas rápidas es parte del campo de estudio del procesamiento de señales sociales.
En este proyecto de investigación se tiene como problema a resolver la identificación de personas y su respectivo género en imágenes tomadas desde un plano de referencia cenital, por medio de la aplicación de los conceptos y soluciones del Aprendizaje de máquina.
Se utilizará el API de código abierto (Detectron2) y una red convolucional regional (RCNN) como propuesta de solución.

II. Estado del Arte

II-A. Visión por Computadora

La aplicación de Visión por computadora es inherente a la vida diaria [1]. Las aplicaciones varían, desde la aplicación de **Realidad Virtual (VR)**, juegos de **Realidad Aumentada (AR)**, "Scanning" de documentos por cámaras en teléfonos inteligentes, Códigos **QR** y **Reconocimiento de Rostros** entre muchas más.

Con los avances recientes en aprendizaje de máquina y aprendizaje profundo, aplicaciones como **Clasificación**, **Detección de objetos**, **Seguimiento** son cada vez más exactas y esto ha logrado avanzar el desarrollo de sistemas autónomos mas complejos, tales como: drones, automóviles auto manejados, humanoides, etc.

Por medio del uso de Aprendizaje Profundo, imágenes (videos) pueden ser transformadas en detalles más complejos; por ejemplo, covertir imágenes en pinturas del estilo de Van Gogh.

¿Qué es Visión Por Computadora? Observando la imagen de la figura 1. Auque nunca se haya realizado esta actividad antes, se puede claramente distinguir que hay

Fig. 1: Imagen Original. [1]

personas esquiando en montañas nevadas en un día nuboso. Esta información que se puede percibir es muy compleja y se puede sub-dividir en inferencias más básicas para un sistema de visión por computadora.

La información más básica de la imagen son los objetos o cosas en la misma. En la imagen anterior, se observan árboles, montañas, nieve, el cielo, personas, etc.

Si se extrae esta información, esto se conoce como **Clasificación de Imágenes**, donde se etiqueta una imagen con un conjunto predefinido de categorías.

En este caso, las etiquetas son las cosas que se observan en la imagen. En una observación más amplia, se observa el escenario de la imagen. El mismo consiste de nieve, montañas y el cielo, como se muestra en la figura 2.

A pesar de que es difícil crear los límites exactos de donde la nieve, montañas, y el cielo están en la imagen, se pueden aproximar las regiones de la imagen para cada elemento. Esto se conoce como segmentación de una imagen, donde se dividen en regiones deacuerdo al espacio que ocupa el objeto. Haciendo la observación más concreta, se pueden identificar exactamente los límites de los objectos en la imagen, como se muestra en la figura 3.

En la imagen, se observa personas haciendo diferentes actividades y por lo tanto tienen diferentes formas; algunas están sentadas y otras esquiando. A pesar de tener muchas variaciones, se pueden detectar objetos y se pueden crear

Fig. 2: Observación más amplia de la imagen. [1]

Fig. 3: Segmentación de la imagen. [1]

cajas de límite alrededor de ellos (en la imagen se muestran algunas cajas de límite).

Mientras en la imagen, se muestran cajas de límite rectangular alrededor de algunos objetos, no se está categorizando qué objeto está en la caja. El próximo paso es describir que la caja contiene a una persona. La combinación de observación de detectar y categorizar la caja se conoce como **detección de objetos**.

Extendiendo la observación de personas y alrededores, se puede describir que hay diferentes personas en la imagen con diferentes alturas, hasta indicar que unas están más cerca y otras más lejos de la cámara. Esto se debe al entendimiento intuitivo de la formación de la imagen y de las relaciones de los objetos. Se sabe que un árbol es casi siempre más alto que una persona, a pesar de que los árboles en la imagen se observan más bajos que las personas que están más cerca de la cámara.

El extraer la información de la geometría en las imágenes es otra sub-área de la visión por computadora y se conoce como **reconstrucción de imágenes**.

En 1963, el científico de computadoras Larry Roberts, conocido como el padre de la Visión por Computadora, describió la posibilidad de extraer información geométrica 3D de bloques de vistas de perspectivas 2D en su investigación titulada "BLOCK WORLD" (Mundo Bloque). [2]

Este fue el primer avance en el mundo de la visión por computadora.

Mundialmente, muchos investigadores en aprendizaje de máquina e inteligencia artificial continuaron ese trabajo desde la perspectiva del "Mundo Bloque".

Los seres humanos pueden reconocer bloques sin importar la orientación o cambios de iluminación que ocurran. Roberts indicó que es importante entender formas simples semejantes a bordes en imágenes.

El extrajo estas formas semejantes a bordes de los bloques para poder hacer que la computadora "entendiera" que esos bloques son los mismos sin importar la orientación como se muestra en la figura 4

La visión inicia con una simple estructura. Este es el inicio de la visión por computadora como un modelo de ingeniería.

David Mark, un científico de visión por computadora del MIT, dió el siguiente concepto importante y es que la visión es jerárquica. En su libro titulado "VISION", definió que la imagen consiste de varias capas.

Estos dos principios formaron la base de la arquitectura de aprendizaje profundo. Sin embargo, no indicaban el tipo de modelo matemático a usar.

En la década de 1970, el primer algoritmo de reconocimiento visual, conocido como el modelo generalizado del cilindro se originó del AI Lab en la Universidad de Standford.

La idea es que el mundo está compuesto de formas simples y que cualquier objeto real es una combinación de estas formas simples.

Al mismo tiempo, otro modelo, conocido como el modelo de estructura pictórica, fue publicado por SRI Inc.

El concepto es el mismo que el modelo generalizado del cilindro, pero las partes están conectadas por "resortes"; por lo que introduce el concepto de variabilidad.

El primer algoritmo de reconocimiento visual se utilizó en una cámara digital de Fujifilm en el 2006.

*II-B. **R-CNNs.***

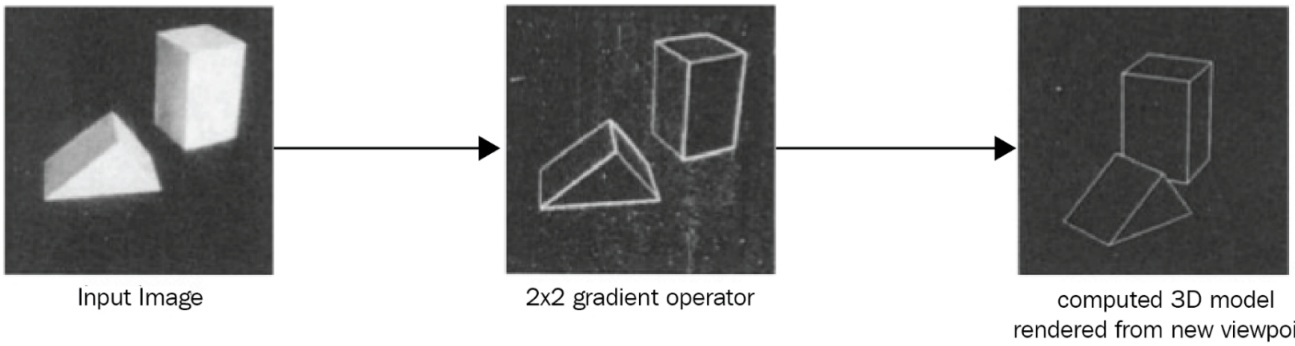

Fig. 4: a)Imagen de entrada. b)2x2 Operador Gradiente. c)Modelo 3D computarizado generado de una perspectiva diferente [2]

R-CNN o Regiones con características CNN, utiliza CNN para generar características que son clasificadas usando una técnica que no es CNN (por ejemplo: máquinas de soporte vectorial - SVM).

R-CNN utilizan el método de desplazar una ventana (tomando un paso de L x W) para generar cerca de 2,000 regiones de interés, para después convertirlas en características para clasificación usando CNN como se muestra en la figura 5.

R-CNN usa una técnica muy popular conocida como Deformación Afín de la imagen para calcular una entrada de tamaño fijo a la CNN a partir de cada región propuesta, sin importar la forma de la región.

Un reto importante es que las regiones candidatas generadas, no son exactas, o no tienen límites muy ajustados alrededor del objeto. Para resolver esto y aumentar la exactitud de las cajas de límite se ejecuta una función de regresión (llamada el regresor de las cajas de límite) para identificar los límites de separación.

III. Entrenamiento y Pérdida

Entrenar un modelo significa aprender o determinar valores correctos para todas las ponderaciones y las ordenadas al origen de las muestras etiquetadas. [3]

En un aprendizaje supervisado, el algoritmo de aprendizaje automático construye un modelo al examinar varios ejemplos e intenta encontrar un modelo que minimice la pérdida. Este proceso se denomina **Minimización del Riesgo Empírico**.

La pérdida es un "castigo" por una predicción incorrecta. En otras palabras, indica cuantitativamente qué tan incorrecta fue la predicción del modelo en una sola muestra.

Si la predicción del modelo es perfecta, la pérdida es cero; de lo contrario, la pérdida es mayor. El objetivo de entrenar un modelo es encontrar un conjunto de ponderaciones y ordenadas al origen que, en promedio, tengan pérdidas bajas en todos los ejemplos.

Por ejemplo, la Figura 6 muestra un modelo al lado izquierdo con una pérdida alta, y al lado derecho un modelo con pérdida baja.

La flecha roja representa la pérdida. La línea azul representa las predicciones. La línea azul en la figura de la derecha es un modelo de predicción mucho más acertado que la línea azul en la figura de la izquierda.

Pérdida L2 o Pérdida al Cuadrado: Los modelos de regresión lineal utilizan esta función de pérdida.

$$MSE = (Observacion - prediccion(x))^2 \ = (y - y')^2 \quad (1)$$

El **Error cuadrático medio (MSE)** es el promedio de la pérdida al cuadrado de cada muestra. Para calcular el MSE, se suman todas las pérdidas al cuadrado de las muestras individuales y, luego, se dividen por la cantidad de muestras:

$$MSE = \frac{1}{N} \sum_{(x,y)\in D} (y - prediccion(x))^2 \quad (2)$$

donde:

- (x,y) es una muestra.
  - x es el conjunto de atributos que el modelo utiliza para realizar las predicciones.
  - y es la etiqueta de la muestra.
- $prediccion(x)$ es un atributo de las ponderaciones y las ordenadas al origen en combinación con el conjunto de atributos x.
- D es el conjunto de datos que contiene muchas muestras etiquetadas, que son los pares (x,y).
- N es la cantidad de muestras en D.

IV. Conjunto de Datos

El conjunto de datos que se utilizó en el proyecto de investigación es de tipo de acceso restringido por razones de confidencialidad. Se conoce como el conjunto de datos *MatchNMingle* de [4].

Es un conjunto de datos multi-modal creado específicamente para contribuir con el análisis automático

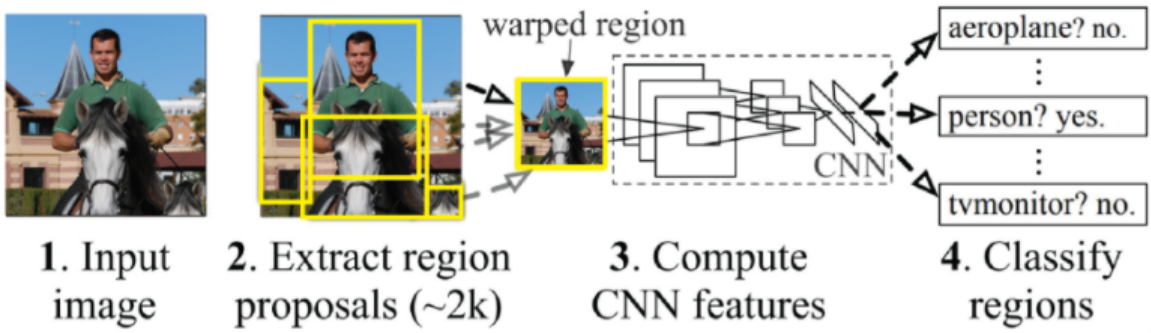

Fig. 5: R-CNN en funcionamiento. [2]

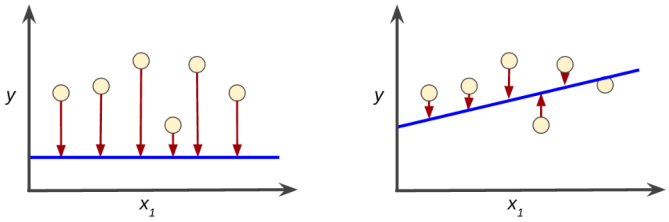

Fig. 6: Pérdida alta en el modelo de la izquierda; pérdida baja en el modelo de la derecha. [3]

de señales e interacciones sociales.

Consiste en 12 horas de grabación ininterrumpida de conversaciones de 92 personas. Los datos fueron tomados en un escenario real, durante 3 días de eventos de *citas rápidas* seguido de una actividad tipo coctel.

Los datos de video se colectaron por medio de cámaras con resolución de 1920x1080 (16:9), a una frecuencia de muestreo de 30 cuadros por segundo (fps) y un campo de visión amplio.

Las cámaras se ubicaron en la parte superior del lugar para evitar artefactos de oclusión por parte de los participantes.

Para el proyecto, se extrajeron imágenes de los videos debido a limitaciones en el recurso de procesamiento computacional disponible.

V. Detectron2

Detectron2 es un sistema de software de Facebook AI Research que implementa algoritmos de última generación para la detección de objetos. Utiliza PyTorch como marco de referencia principal.

Incluye funciones como segmentación panóptica, Cascada R-CNN, cuadros delimitadores rotados, etc. [5]

Detectron 2 permite entrenar un modelo de detección de objetos usando un conjunto de datos generados por el usuario. [6].

Todos los modelos que se encuentran en el modelo **zoo** de la biblioteca de Detectron 2 están pre-entrenados en conjuntos de datos **COCO**. Es necesario ajustar el conjunto de datos del propietario al modelo pre-entrenado.

Detectron 2 es una versión mejorada del primer Detectron liberado en el año 2018. La primera versión fue escrita en **Caffe2**, un marco de referencia de aprendizaje profundo desarrollado por Facebook. Ambos, Caffe2 y Detectron son obsoletos. Caffe2 es ahora parte de PyTorch y el sucesor, Detectron 2 está completamente escrito en PyTorch.

Detectron 2 tiene como objetivo avanzar el aprendizaje de máquina al ofrecer un entrenamiento más rápido y resuelve problemas en el paso de investigación a producción para las empresas.
Hay varios tipos de modelos de detección disponibles en Detectron 2 como se muestra en la figura 7.

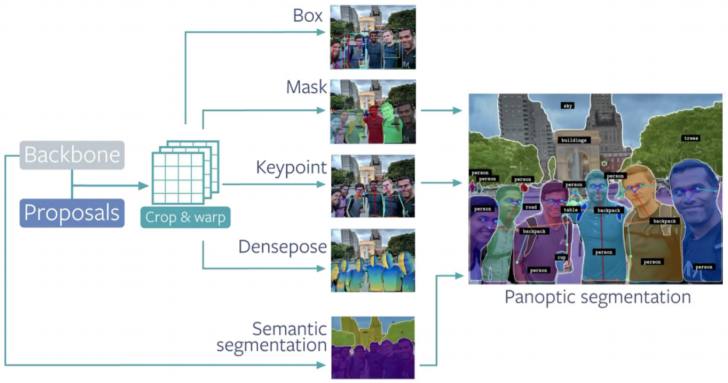

Fig. 7: Opciones de detección de objetos disponibles en Detectron 2. [7]

Detección de Instancia se refiere a la clasificación y localización de un objeto con una caja de límite (“Bounding box”)

VI. ARQUITECTURA [8]

La arquitectura del modelo de red neuronal a utilizar se conoce como *Base R-CNN-FPN (Base Regional Convolution Neural Network Feature Pyramid Network)* o también *Faster R-CNN*, que es un detector de objectos estándar con una alta precisión en la tarea y muy utilizado en problemas de segmentación.

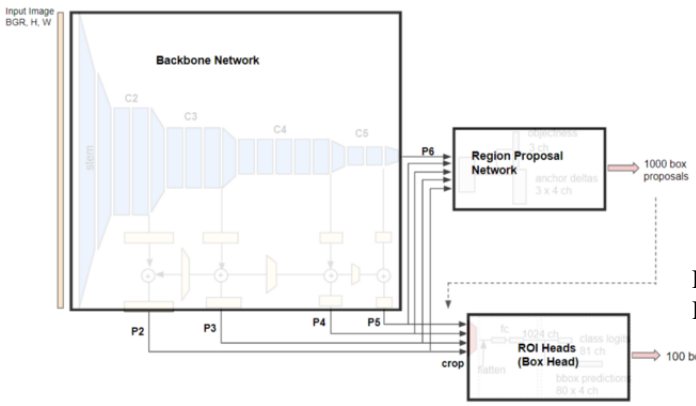

Fig. 8: Arquitectura Base RCNN FPN

El modelo de alto nivel se muestra en la figura 8, donde se observan tres bloques principales, estos son:

1. **Backbone:** extrae las características (features) de la imagen de entrada a diferentes escalas. Las características de salida se denominan P2 (escala 1/4), P3 (escala 1/8), P4 (escala 1/16), P5 (escala 1/32) y P6 (escala 1/64).
2. **Red de región propuesta o "Region Proposal Network(RPN)":** detecta regiones de objetos a partir de las características a múltiples escalas. Se obtienen 1000 propuestas de cajas (por defecto) con puntajes de confianza diferentes.
3. **Región de Interés (ROI Head):** Recorta y deforma las características utilizando las cajas propuestas por la RPN en características de tamaño fijo y haciendo un ajuste fino se obtienen las localizaciones de las cajas y la clasificación de estas a través de las *Capas totalmente conectadas o "Fully connected layers"*. Por último, utilizando la *Supresión No Máxima o "Non-maximum supression"*, las cajas son filtradas.

VI-A. Backbone (FPN)

El modelo principal o "backbone" está construido como una red de Pirámide de Características *Feature Pyramid Network* que contiene varias etapas de redes neuronales **ResNet** conectadas en una especie de pirámide, como se muestra en la figura 9.

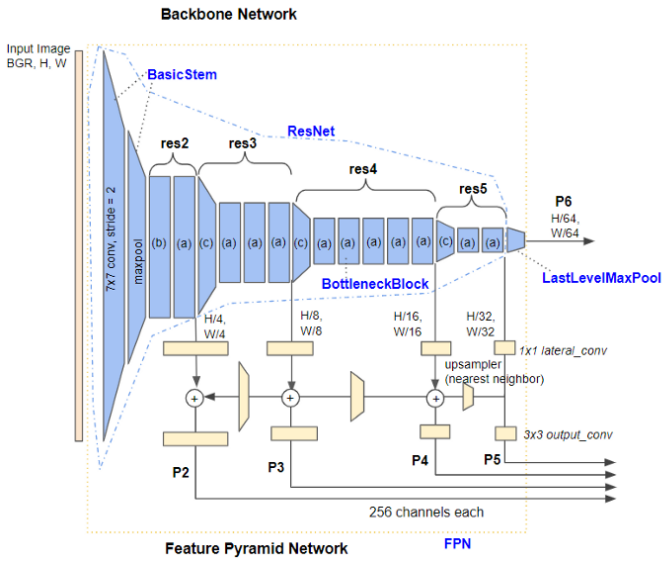

Fig. 9: Arquitectura detallada de la red "backbone"Base-RCNN-FPN con ResNet

Contiene también capas convolucionales laterales y de salida, sobre-muestreo *(up-samplers)* y una última capa de "pooling"*(last-level maxpool layer)*.

Recibe como entrada un tensor de la forma **input (torch.Tensor): (B, 3, H, W) image**, donde B es el tamaño del "batch", H y W representan el tamaño de la imagen. La imagen debe estar en formato BGR.

La salida será un tensor de características de la forma **output (dict of torch.Tensor): (B, C, H / S, W / S) feature**, donde C representa el tamaño del canal (por defecto es 256) y el "stride", y S las escalas para P2, P3, ... , P6.

*VI-A.1. **ResNet:*** Las redes neuronales residuales o **ResNets**, se inspiran en construcciones conocidas de células piramidales en la corteza cerebral, hacen esto mediante la utilización de conexiones de salto o atajos para saltar sobre algunas capas.

En este caso, consta de un bloque principal y etapas que contienen varios bloques de cuello de botella. Se utiliza la arquitectura **ResNet50** que tiene la siguiente estructura:

```
BasicStem
(res2 stage, 1/4 scale)
BottleneckBlock (b)(stride=1, with
shortcut conv)
BottleneckBlock (a)(stride=1, w/0
shortcut conv) x 2
(res3 stage, 1/8 scale)
BottleneckBlock (c)(stride=2,
with shortcut conv)
```

```
BottleneckBlock (a)(stride=1,
w/o  shortcut conv) x 3
(res4 stage, 1/16 scale)
BottleneckBlock (c)(stride=2,
with shortcut conv)
BottleneckBlock (a)(stride=1,
w/o shortcut conv) x 5
(res5 stage, 1/32 scale)
BottleneckBlock (c)(stride=2,
with shortcut conv)
BottleneckBlock (a)(stride=1,
w/o  shortcut conv) x 2
```

1. **BasicStem:** El bloque "stem" de las ResNet es bastante simple. Reduce la resolución de la imagen de entrada dos veces mediante una convolución con un "kernel"de 7×7 con "stride"de 2, un "pooling" con "stride" de 2 y una función de activación ReLU. La salida es un tensor de características cuyo tamaño es B, 64, H/4, W/4.

```
conv1 (kernel size = 7,
stride = 2)
batchnorm layer
ReLU
maxpool layer (kernel size = 3,
stride = 2)
```

2. **Bottleneck Block:** Tiene tres capas de convolución cuyos tamaños de "kernel" son 1×1, 3×3, 1×1 respectivamente. La figura 10 muestra los tres tipos de cuellos de botella y su diseño interno.
3. **Capas convolucionales laterales:** Las capas convolucionales laterales toman las características de las etapas res2-res5 con diferentes números de canal y devuelven características de 256 canales.

4. **Capas convolucionales de salida:** La capa convolucional de salida contiene una convolución con un "kernel" de 3 x 3 que no cambia el número de canales.

VI-B. Red de Propuesta de Región o Region Proposal Network

Esta es una red neuronal simple, la figura 11 muestra un esquema de la red. Se llama **RPN Head** y consta de tres capas convolucionales definidas como la clase StandardRPNHead. Los cinco niveles de características (P2 a P6) alimentan a la red uno por uno.

Esta red puede detectar objetos pequeños en P2 y P3 y los más grandes en P4 a P6. Este es el objetivo que busca la red piramidal. La red de múltiples escalas puede

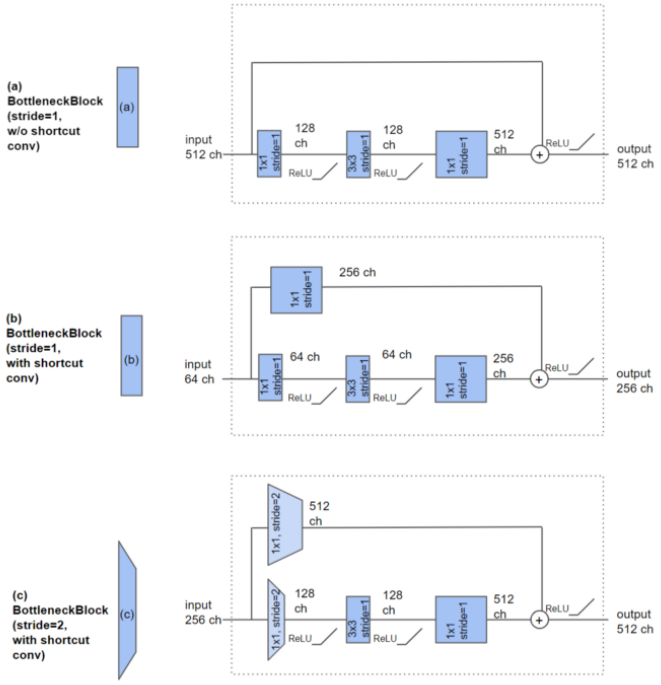

Fig. 10: Tres tipos de cuellos de botella

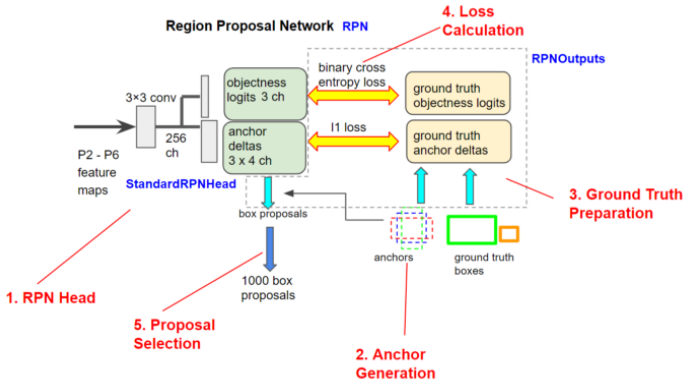

Fig. 11: Esquema de **Region Proposal Network**

detectar objetos que un detector de una sola escala no puede encontrar.

*VI-B.1. **Generación Ancla - Anchor generation**:* Existen unas cajas de referencia llamadas **anchors** que relacionan las características con los datos entrenamiento, es decir el *"ground truth"*. Estos "anchor" se definen como:

```
MODEL.ANCHOR_GENERATOR.SIZES= \
[[32], [64], [128], [256], [512]]
MODEL.ANCHOR_GENERATOR.ASPECT_RATIOS= \
[[0.5, 1.0, 2.0]]
```

Los cinco elementos de la lista ANCHOR_GENERATOR.SIZES corresponden a cinco niveles de características (P2 a P6).

Las *razones de aspecto "aspect ratios"* definen las formas de los "anchors". Para el ejemplo anterior, hay tres formas: 0.5, 1.0 y 2.0.

Los tres "anchors" de las características P2 tienen relaciones de aspecto de 1:2, 1:1 y 2:1 y las mismas áreas de 32 x 32.

En el nivel P3, los "anchors" son dos veces más grandes que los anclajes P2.

Estos anclajes se denominan células ancla "cell anchors" en Detectron2.

*VI-C. **Cálculo de pérdidas (Loss Calculation)***

Se aplican dos funciones de pérdida a los mapas de predicción y el *ground truth*: Pérdida de localización "localization loss"(loss_rpn_loc) y Pérdida de Objeto "objectness loss"(loss_rpn_cls).

*VI-D. **ROI Head***

En el **ROI Head**, se toman:

1. Las características de FPN: Se usan las dimensiones de los tensores de las características P2-P5, P6 no se usa.
2. Salida de RPN: Se utilizan las cajas propuestas por RPN para recortar las características.
3. *Ground truth*: Se obtienen del conjunto de datos.

*VI-D.1. **Muestreo de cajas propuestas**:* En RPN, se obtienen 1000 cajas propuestas de los cinco niveles de características (P2 a P6). Estas cajas se utilizan para recortar las regiones de interés (ROI) de las características, que se envían al "Box Head".

Durante el entrenamiento, las cajas propuestas de primer plano y de fondo se vuelven a muestrear en primer lugar para equilibrar el entrenamiento.

Para equilibrar los cuadros en primer plano y de fondo se hace lo siguiente: Sea N el número objetivo de cuadros (primer plano + fondo) y F el número objetivo de cuadros en primer plano. N y F/N se definen mediante los siguientes hiper-parámetros de configuración.

```
N:MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE
(typically 512)
F/N:MODEL.ROI_HEADS.POSITIVE_FRACTION
(typically 0.25)
```

VII. Selección de hiper-parámetros

Los hiper-parámetros seleccionados para el entrenamiento se muestran en la Tabla 1.

VIII. Resultados Obtenidos y Análisis

Aplicando el marco de referencia Detectron 2 se generó un conjunto de entrenamiento de 260 imágenes etiquetadas como se muestra en las figuras 12 13 14

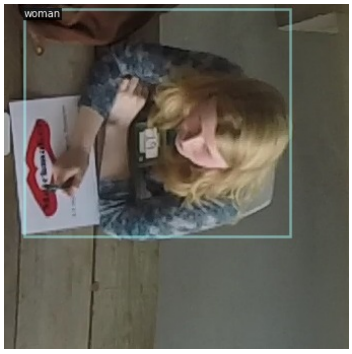

Fig. 12: **Muestra de Imagen en el Conjunto de Entrenamiento**

Fig. 13: **Muestra de Imagen en el Conjunto de Entrenamiento**

Se ajustaron los hiper-parámetros del modelo y se realizó el entrenamiento de la red R-CNN por medio de cincuenta mil iteraciones.

Se obtuvieron las inferencias o predicciones de clasificación de género a partir de un conjunto de prueba compuesto por 71 imágenes desconocidas para el modelo de predicción como se muestra en las figuras 15 16 17

Se realizaron ajustes del conjunto de hiper-parámetros del marco de referencia Detectron2 y varias sesiones de entrenamiento e inferencia del modelo.

Tipo	Hiper-parámetro	Valor	Función
INPUT	MIN_SIZE_TRAIN	340	Tamaño más pequeño de la imagen durante el entrenamiento
DATASETS	TRAIN	(”people_train”,)	Nombre del dataset para entrenamiento
DATALOADER	NUM_WORKERS	4	Numero de hilos para entrenamiento (GPU)
MODEL	BACKBONE.NAME	’build_resnet_fpn_backbone’	Arquitectura de red neuronal
MODEL	RESNETS.OUT_FEATURES	[’res2’, ’res3’, ’res4’, ’res5’]	Etapas de la ResNet
MODEL	ROI_BOX_HEAD.NAME	’FastRCNNConvFCHead’	Modelo de la etapa de ROI-HEAD
MODEL	ROI_BOX_HEAD.NUM_FC	2	Capas totalmente conectadas - Fully connected layers
MODEL	ROI_BOX_HEAD.POOLER_RESOLUTION	7	Pooling
MODEL	ROI_HEADS.IN_FEATURES	[’p2’, ’p3’, ’p4’, ’p5’]	Los nombres de los mapas de características de entrada que se utilizará en ROI-HEADS
MODEL	ROI_HEADS.NAME	’StandardROIHeads’	Formato de ROI-HEAD
MODEL	ROI_MASK_HEAD.NUM_CONV	4	Numero de convoluciones en ROI-HEAD
MODEL	RPN.IN_FEATURES	[’p2’, ’p3’, ’p4’, ’p5’, ’p6’]	Los nombres de los mapas de características de entrada que se utilizará en RPN
MODEL	RETINANET.NUM_CONVS	2	Numero de convoluciones en RETINANET
MODEL	BACKBONE.FREEZE_AT	3	Se desahibilitan las primeras etapas para que no sean entrenadas
MODEL	PIXEL_STD	[57.375, 57.120, 58.395]	Valores para normalizacion de pixeles
MODEL	ROI_HEADS.BATCH_SIZE_PER_IMAGE	32	Numero de regiones de interes
MODEL	ROI_HEADS.NUM_CLASSES	2	Cantidad de clases para el entrenamiento
MODEL	FPN.IN_FEATURES	[res2”, res3”, res4”, res5”]	Los nombres de los mapas de características de entrada que se utilizará en FPN
MODEL	ANCHOR_GENERATOR.SIZES	[[32], [64], [128], [256], [512]]	Definicion de las cajas de referencia
SOLVER	IMS_PER_BATCH	8	Número de imágenes por lote en todas las máquinas. Si tenemos 16 GPU e IMS_PER_BATCH = 32, cada GPU verá 2 imágenes por lote.
SOLVER	BASE_LR	0.00025	Leraning rate (Tasa de aprendizaje)
SOLVER	MAX_ITER	55100	Catidad de interaciones
SOLVER	GAMMA	0.1	A cierto numero de interaciones la tasa de aprendizaje puede disminuir por GAMMA
SOLVER	STEPS	(210000, 250000)	Numero de interaciones donde la tasa de aprendizaje disminuye por GAMMA

TABLE I: Hiper-parámetros del modelo

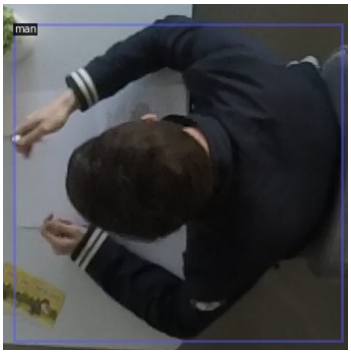

Fig. 14: **Muestra de Imagen en el Conjunto de Entrenamiento**

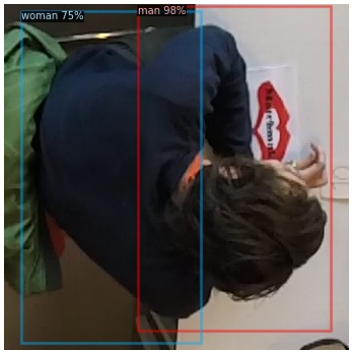

Fig. 15: **Muestra de Inferencia obtenida con el Conjunto de Prueba**

La Precisión Promedio (AP) que se obtuvo al final de los procesos de entrenamiento e inferencia fue de un 14 %. Figura 18

Al efectuar la comparación de pérdidas entre las etapas de entrenamiento y pruebas, se confirma que el modelo presenta una condición de sobre-ajuste “Overfitting”. Figura 19

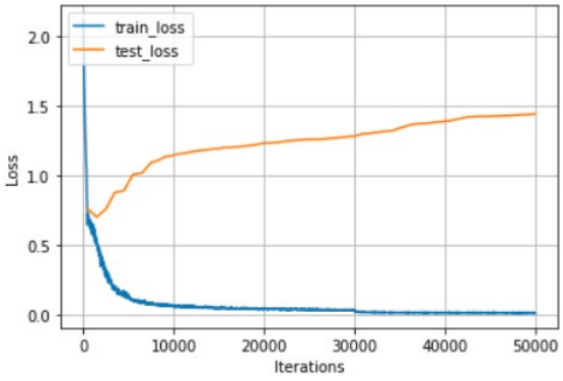

Fig. 19: **Comparación de Pérdidas durante la etapas de entrenamiento y prueba.**

Fig. 16: **Muestra de Inferencia obtenida con el Conjunto de Prueba**

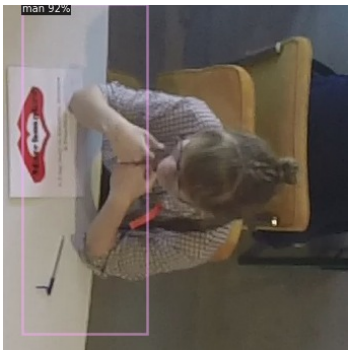

Fig. 17: **Muestra de Inferencia obtenida con el Conjunto de Prueba**

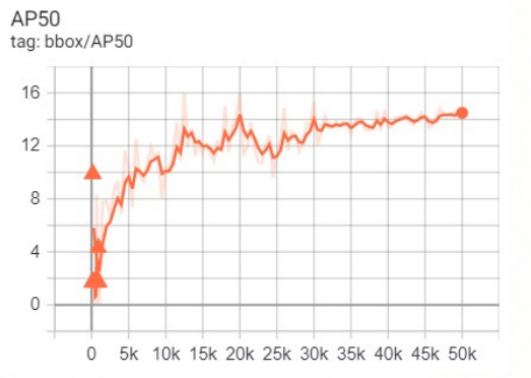

Fig. 18: **Precisión Promedio(AP) del modelo desarrollado**

IX. Conclusiones

1. Detectron 2 es una herramienta de detección de objetos pero tiene el inconveniente de tener escasa documentación para el ajuste del modelo.
2. El entrenamiento de la red R-CNN requiere un alto recurso computacional que obliga a tener una implementación en Google Colab.
3. Se sospecha que la segmentación de las imágenes requieren un pre-procesado de bordes para facilitar el entrenamiento de la red neuronal.
4. No se logra una buena precisión promedio en la inferencia de las imágenes causado por una condición de sobre-ajuste "overfitting"que no se logró solucionar con variaciones de hiper-parámetros.
5. El marco de referencia de Detectron 2 habilita al usuarios realizar funciones de segmentación panóptica, Cascada R-CNN, cuadros delimitadores rotados.
6. R-CNN utilizan el método de desplazar una ventana (tomando un paso de L x W) para generar cerca de 2,000 regiones de interés, para después convertirlas en características para clasificación usando CNN.

References

[1] A. Dadhich, *Practical Computer Vision: Extract Insightful Information from Images Using TensorFlow, Keras, and OpenCV*. Packt Publishing Ltd, 2018.

[2] M. Sewak, M. R. Karim, and P. Pujari, *Practical Convolutional Neural Networks: Implement Advanced Deep Learning Models Using Python*. Packt Publishing Ltd, 2018.

[3] Google, "Machine learning crash course," 2020, accesado el 2 de Agosto 2020. [Online]. Available: https://developers.google.com/machine-learning/crash-course

[4] L. Cabrera-Quiros, A. Demetriou, E. Gedik, L. Meij, and H. Hung, "The MatchNMingle dataset: a novel multi-sensor resource for the analysis of social interactions and group dynamics in-the-wild during free-standing conversations and speed dates," *IEEE Transactions on Affective Computing*, 2018.

[5] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.

[6] A.Yelisetty, "Object detection in 6 steps using detectron2," 2020, accesado el 2 de Agosto 2020. [Online]. Available: https://towardsdatascience.com/object-detection-in-6-steps-using-detectron2-705b92575578

[7] F. Research, "Detectron 2," 2020, accesado el 2 de Agosto 2020. [Online]. Available: https://research.fb.com/wp-content/uploads/2019/12/4.-detectron2.pdf

[8] H. Honda, "Digging into detectron 2 — by hiroto honda — medium," https://medium.com/@hirotoschwert/digging-into-detectron-2-47b2e794fabd, (Accessed on 08/06/2020).