Avance de Proyecto Final

Ronald Caravaca
Carlos Brenes

6 de Agosto del 2020

1. 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.

2. Conjunto de Datos

El conjunto de datos a utilizar en el proyecto de investigación es de tipo de acceso restringido por razones de confidencialidad. Se conoce como el conjunto de datos *MatchNMingle* de [1]. Es un conjunto de datos multi-modal creado específicamente para contribuir con el análisis automático 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.

3. 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, Cascade R-CNN, cuadros delimitadores rotados, etc. [3]

4. Arquitectura [2]

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.

El modelo de alto nivel se muestra en la figura 1, 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).

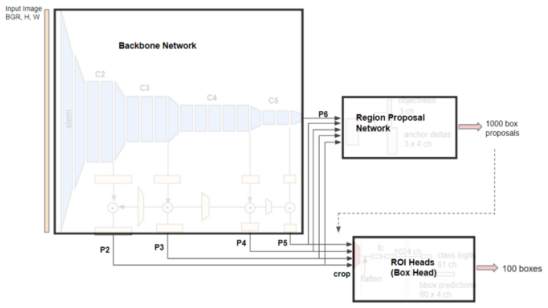

Figura 1: Arquitectura Base RCNN FPN

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.

4.1. 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 2.

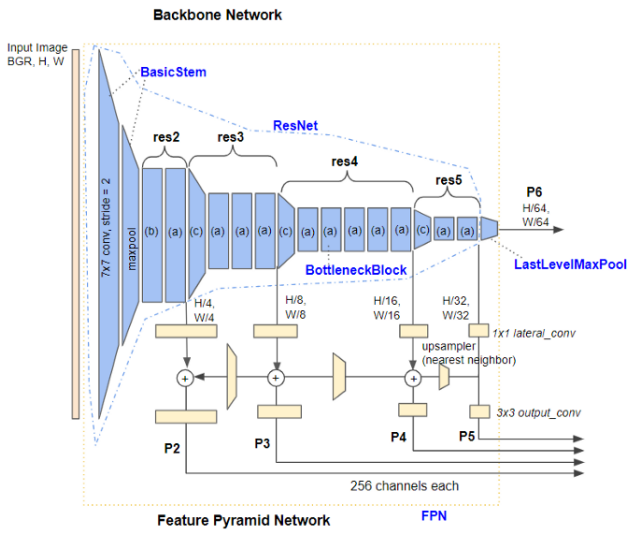

Figura 2: Arquitectura detallada de la red "backbone"Base-RCNN-FPN con ResNet

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.

4.1.1. ResNet

La 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) × 2
(res3 stage, 1/8 scale)
BottleneckBlock (c)(stride=2, with shortcut conv)
BottleneckBlock (a)(stride=1, w/o  shortcut conv) × 3
(res4 stage, 1/16 scale)
BottleneckBlock (c)(stride=2, with shortcut conv)
BottleneckBlock (a)(stride=1, w/o shortcut conv) × 5
(res5 stage, 1/32 scale)
BottleneckBlock (c)(stride=2, with shortcut conv)
BottleneckBlock (a)(stride=1, w/o  shortcut conv) × 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 3 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 3×3 que no cambia el número de canales.

4.2. Red de Propuesta de Región o Region Proposal Network

Esta es una red neuronal simple, la figura 4 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 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 detectar objetos que un detector de una sola escala no puede encontrar.

4.2.1. 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]]
```

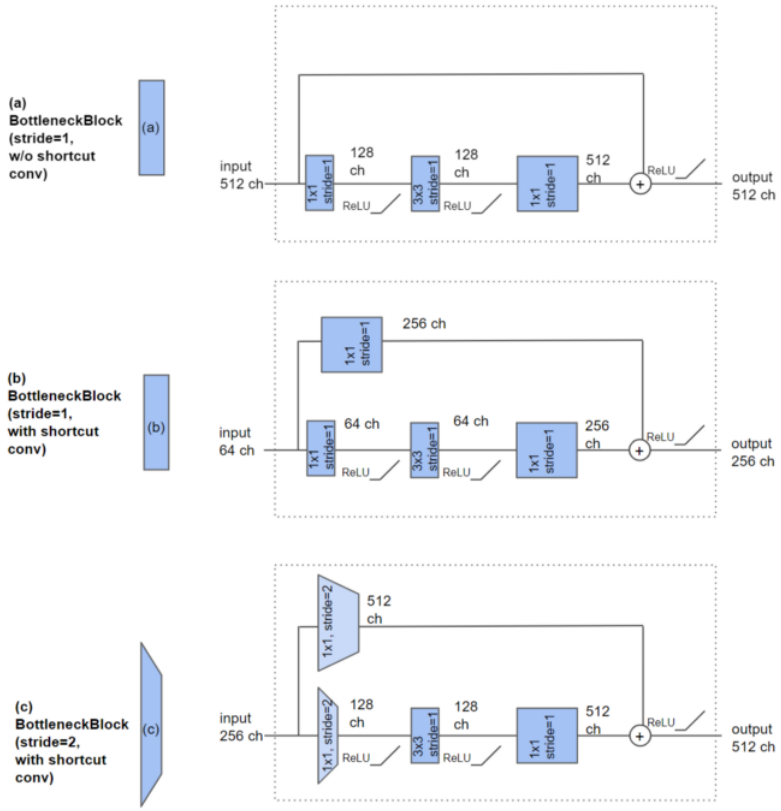

Figura 3: Tres tipos de cuellos de botella

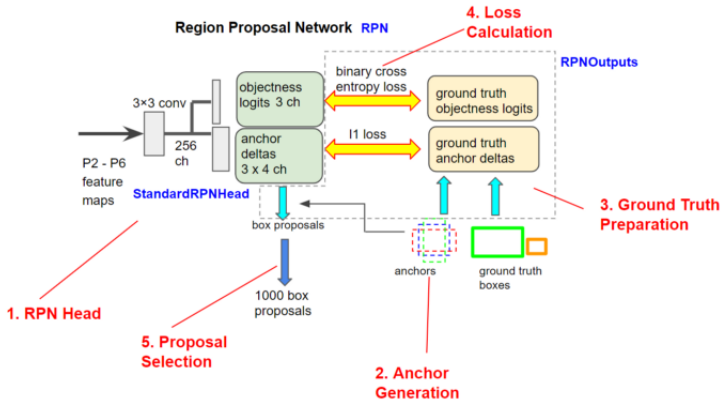

Figura 4: Esquema de **Region Proposal Network**

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 32×32. 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.

4.3. 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).

4.4. 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.

4.4.1. 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 parámetros de configuración.

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

5. Cronograma de Actividades

Se han definido las siguientes actividades para la realización del proyecto. Figura 5.

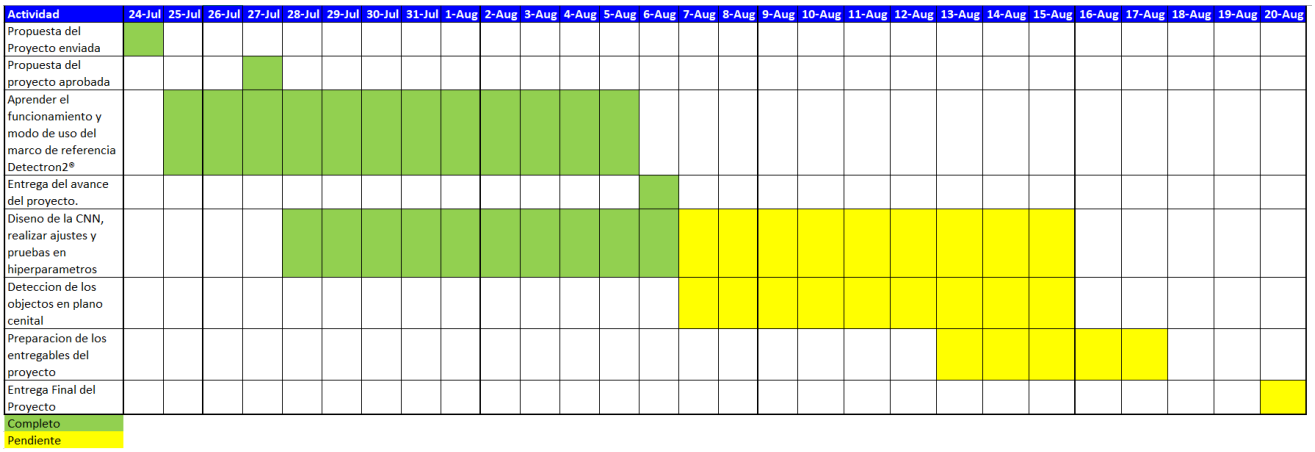

Figura 5: Cronograma tentativo de actividades.

6. Etapas pendientes

Definir y realizar el ajuste de los hiper-parámetros para el modelo de aprendizaje y su posterior entrenamiento. Entre los hiper-parámetros a ajustar se tienen: [3]

```
# ---------------------------------------------------------------------------- #
# FPN options
# ---------------------------------------------------------------------------- #
_C.MODEL.FPN = CN()
# Names of the input feature maps to be used by FPN
# They must have contiguous power of 2 strides
# e.g., ["res2", "res3", "res4", "res5"]
```

```
_C.MODEL.FPN.IN_FEATURES = []
_C.MODEL.FPN.OUT_CHANNELS = 256

# Options: "" (no norm), "GN"
_C.MODEL.FPN.NORM = ""

# Types for fusing the FPN top-down and lateral features. Can be either "sum" or "avg"
_C.MODEL.FPN.FUSE_TYPE = "sum"

# ---------------------------------------------------------------------------- #
# Proposal generator options
# ---------------------------------------------------------------------------- #
_C.MODEL.PROPOSAL_GENERATOR = CN()
# Current proposal generators include "RPN", "RRPN" and "PrecomputedProposals"
_C.MODEL.PROPOSAL_GENERATOR.NAME = "RPN"
# Proposal height and width both need to be greater than MIN_SIZE
# (a the scale used during training or inference)
_C.MODEL.PROPOSAL_GENERATOR.MIN_SIZE = 0

# ---------------------------------------------------------------------------- #
# Anchor generator options
# ---------------------------------------------------------------------------- #
_C.MODEL.ANCHOR_GENERATOR = CN()
# The generator can be any name in the ANCHOR_GENERATOR registry
_C.MODEL.ANCHOR_GENERATOR.NAME = "DefaultAnchorGenerator"
# Anchor sizes (i.e. sqrt of area) in absolute pixels w.r.t. the network input.
# Format: list[list[float]]. SIZES[i] specifies the list of sizes
# to use for IN_FEATURES[i]; len(SIZES) == len(IN_FEATURES) must be true,
# or len(SIZES) == 1 is true and size list SIZES[0] is used for all
# IN_FEATURES.
_C.MODEL.ANCHOR_GENERATOR.SIZES = [[32, 64, 128, 256, 512]]
# Anchor aspect ratios. For each area given in `SIZES`, anchors with different aspect
# ratios are generated by an anchor generator.
# Format: list[list[float]]. ASPECT_RATIOS[i] specifies the list of aspect ratios (H/W)
# to use for IN_FEATURES[i]; len(ASPECT_RATIOS) == len(IN_FEATURES) must be true,
# or len(ASPECT_RATIOS) == 1 is true and aspect ratio list ASPECT_RATIOS[0] is used
# for all IN_FEATURES.
_C.MODEL.ANCHOR_GENERATOR.ASPECT_RATIOS = [[0.5, 1.0, 2.0]]
# Anchor angles.
# list[list[float]], the angle in degrees, for each input feature map.
# ANGLES[i] specifies the list of angles for IN_FEATURES[i].
_C.MODEL.ANCHOR_GENERATOR.ANGLES = [[-90, 0, 90]]
# Relative offset between the center of the first anchor and the top-left corner of the image
# Value has to be in [0, 1). Recommend to use 0.5, which means half stride.
# The value is not expected to affect model accuracy.
_C.MODEL.ANCHOR_GENERATOR.OFFSET = 0.0

# ---------------------------------------------------------------------------- #
# ROI HEADS options
# ---------------------------------------------------------------------------- #
_C.MODEL.ROI_HEADS = CN()
_C.MODEL.ROI_HEADS.NAME = "Res5ROIHeads"
# Number of foreground classes
_C.MODEL.ROI_HEADS.NUM_CLASSES = 80
# Names of the input feature maps to be used by ROI heads
# Currently all heads (box, mask, ...) use the same input feature map list
# e.g., ["p2", "p3", "p4", "p5"] is commonly used for FPN
_C.MODEL.ROI_HEADS.IN_FEATURES = ["res4"]
```

```
# IOU overlap ratios [IOU_THRESHOLD]
# Overlap threshold for an RoI to be considered background (if < IOU_THRESHOLD)
# Overlap threshold for an RoI to be considered foreground (if >= IOU_THRESHOLD)
_C.MODEL.ROI_HEADS.IOU_THRESHOLDS = [0.5]
_C.MODEL.ROI_HEADS.IOU_LABELS = [0, 1]
# RoI minibatch size *per image* (number of regions of interest [ROIs])
# Total number of RoIs per training minibatch =
#   ROI_HEADS.BATCH_SIZE_PER_IMAGE * SOLVER.IMS_PER_BATCH
# E.g., a common configuration is: 512 * 16 = 8192
_C.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 512
# Target fraction of RoI minibatch that is labeled foreground (i.e. class > 0)
_C.MODEL.ROI_HEADS.POSITIVE_FRACTION = 0.25
```

Mayores detalles en los hiper-parámetros se pueden encontrar en https://detectron2.readthedocs.io/modules/config.html?highlight=cfg.MODEL#detectron2.config.get_cfg

Referencias

[1] L. Cabrera-Quiros, A. Demetriou, E. Gedik, L.V.D. 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.

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

[3] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.