Universidad Mayor Real y Pontificia de
San Francisco Xavier de Chuquisaca
Facultad de Tecnología

Informe Laboratorio 10 - Análisis de Modelos de Face Detection y Reconocimiento Facial

Universitario: Crespo Rejas Jhamil

Carrera: Ingeniería en Ciencias de la Computación

Materia: Inteligencia Artificial II

Sigla: SIS421

Docente: Ing. Carlos Pacheco

Sucre – Bolivia

Análisis de Modelos de Face Detection y Reconocimiento Facial

1. Introducción

El reconocimiento facial se ha convertido en una tecnología clave dentro del campo de la visión por computadora, con aplicaciones que abarcan desde seguridad y autenticación biométrica hasta análisis de emociones y marketing personalizado. Según Deng et al. (2018), la efectividad de estas tecnologías depende en gran medida de la calidad de los modelos utilizados tanto para la detección de rostros como para el reconocimiento facial. Por lo tanto, una combinación de detección precisa y reconocimiento robusto es esencial para aplicaciones prácticas.

Este informe aborda una evaluación comparativa de los modelos más destacados para **Face Detection** y **Reconocimiento Facial**, analizando sus características, fortalezas y limitaciones. Además, se identifica el modelo más adecuado para cada tarea, basándose en benchmarks reconocidos y casos de uso específicos.

2. Comparativa de Modelos de Face Detection

La detección facial, como etapa inicial del reconocimiento, requiere modelos que equilibren precisión, velocidad y eficiencia en el consumo de recursos. En la siguiente tabla se presentan las características clave de cinco modelos populares de detección facial: **SRN-CFD**, **YuNet**, **RetinaFace**, **MTCNN** y **Dlib (HOG)**.

Aspecto	SRN-CFD	YuNet	RetinaFace	MTCNN	Dlib (HOG)
Precisión	Muy alta	Media	Muy alta	Alta	Media
Velocidad	Media	Muy alta	Media	Media	Alta
Landmarks	Precisa	Básica	Precisa	Precisa	No
Consumo de Recursos	Alto	Muy bajo	Medio	Medio	Muy bajo

Casos ideales	Rostros pequeños/multitudes	Móviles, IoT	Vigilancia avanzada	Investigación	Prototipos rápidos

Referencias de Benchmarks:

- **SRN-CFD**: Demuestra alta precisión en condiciones de multitudes y rostros pequeños, evaluado en benchmarks como WIDER FACE, donde supera a otros modelos en entornos densos y complejos (Zhu et al., 2018).
- **YuNet**: Ideal para dispositivos con recursos limitados, muestra un excelente desempeño en velocidad con imágenes de resolución moderada. Según pruebas realizadas en OpenCV Zoo, YuNet sobresale en aplicaciones en tiempo real para IoT (OpenCV, 2022).
- **RetinaFace**: Reconocido por su capacidad para manejar poses extremas y condiciones de iluminación adversa. Según Deng et al. (2019), RetinaFace obtiene resultados sobresalientes en WIDER FACE, superando consistentemente a MTCNN en precisión.
- **MTCNN**: Su enfoque multitarea lo hace ideal para proyectos de investigación donde la alineación facial es crucial. Zhang et al. (2016) destacan su precisión para detección y alineación simultánea.
- **Dlib (HOG)**: Aunque menos preciso, es una opción eficiente y rápida para prototipos básicos, como lo señala King (2009) en la documentación oficial de Dlib.

Conclusión de Face Detection

RetinaFace: La Mejor Elección para Precisión y Robustez

RetinaFace se posiciona como el modelo líder en detección facial gracias a su alta precisión y robustez, particularmente en escenarios desafiantes como rostros parcialmente visibles, poses extremas y condiciones de iluminación adversa. Este modelo combina una arquitectura de detección avanzada con regresión precisa de landmarks faciales, lo que lo hace ideal para aplicaciones críticas donde el análisis facial detallado es fundamental. Deng et al. (2019) destacan su sobresaliente desempeño en benchmarks como WIDER FACE, donde consistentemente supera a otros modelos en precisión y manejo de variaciones faciales complejas.

- **Aplicaciones recomendadas**:
  - Vigilancia avanzada.
  - Sistemas de control de acceso biométrico.
  - Análisis forense y tareas donde la precisión es prioritaria.
- **Requisitos**:
  - Es preferible el uso de GPUs para aprovechar al máximo sus capacidades computacionales.

Alternativa Eficiente: YuNet

Para aplicaciones que priorizan la velocidad y la eficiencia en recursos, **YuNet** es una solución ideal. Este modelo, optimizado para dispositivos móviles y sistemas IoT, ofrece un

balance excepcional entre rendimiento y consumo. Según OpenCV (2022), su diseño liviano permite su implementación en entornos con recursos limitados, logrando un desempeño en tiempo real con imágenes de resolución moderada.

- **Aplicaciones recomendadas**:
    - Sistemas de reconocimiento facial en dispositivos móviles.
    - Cámaras de seguridad de bajo costo.
    - Aplicaciones ligeras como filtros faciales o desbloqueo facial.
- **Requisitos**:
    - Puede ejecutarse eficientemente en CPUs, lo que lo hace accesible para dispositivos económicos.

Comparación entre RetinaFace y YuNet

Aspecto	RetinaFace	YuNet
Precisión	Muy alta	Media
Velocidad	Media	Muy alta
Consumo de Recursos	Medio (recomendado con GPU)	Muy bajo (CPU suficiente)
Casos ideales	Vigilancia avanzada, análisis crítico	Aplicaciones móviles, sistemas IoT

Referencias de Benchmarks:

1. **RetinaFace**:
    - Según Deng et al. (2019), RetinaFace lidera en precisión en el benchmark WIDER FACE, gracias a su capacidad para detectar landmarks con alta exactitud incluso en condiciones adversas.
2. **YuNet**:
    - OpenCV (2022) subraya su velocidad excepcional en sistemas de bajo consumo, destacando su desempeño eficiente en tiempo real en dispositivos móviles e IoT.

Recomendación Final

1. **Si el objetivo principal es la precisión y la tarea implica entornos adversos o aplicaciones críticas**:
    - **RetinaFace** es la mejor elección debido a su capacidad para manejar variaciones complejas en rostros, convirtiéndose en el líder indiscutible en detección facial avanzada.
2. **Si la prioridad está en la eficiencia y la rapidez con recursos computacionales limitados**:
    - **YuNet** es una alternativa excelente, ideal para sistemas ligeros que requieren velocidad y bajo consumo sin comprometer significativamente el desempeño.

3. Comparativa de Modelos de Reconocimiento Facial

La selección del modelo de reconocimiento facial depende del caso de uso y las prioridades, como precisión, velocidad y consumo de recursos. En la siguiente tabla se presenta una comparación de los modelos más destacados, destacando sus fortalezas, limitaciones y aplicaciones ideales.

Tabla Comparativa de Modelos de Reconocimiento Facial

Modelo	Fortalezas	Limitaciones	Casos Ideales de Uso
DeepFace	Framework modular compatible con múltiples modelos (VGG-Face, ArcFace, FaceNet). Fácil de integrar y soporta análisis avanzados (emociones, género).	Velocidad moderada en comparación con modelos especializados.	Sistemas comerciales y aplicaciones multifuncionales.
FaceNet	Representaciones compactas (128 dimensiones). Excelente precisión para comparación de rostros y clustering.	Requiere mayor personalización para tareas específicas.	Bases de datos grandes y sistemas escalables.
ArcFace	Precisión sobresaliente gracias a su función de pérdida angular (*angular margin loss*). Robusto en escenarios críticos.	Alto consumo de recursos y configuración técnica compleja.	Autenticación en sistemas de alta seguridad.

Dlib (ResNet)	Rápido, ligero y fácil de implementar. No requiere GPU para procesamiento básico.	Precisión limitada frente a modelos avanzados.	Prototipos rápidos y tareas ligeras de reconocimiento.
SRN-CFD	Precisión excepcional en rostros compactos y multitudes. Maneja rostros pequeños en alta densidad.	Velocidad moderada y mayor consumo de recursos.	Escenarios densos como multitudes y vigilancia avanzada.

Referencias de Benchmarks y Evaluaciones:

1. **DeepFace**: Según Serengil (2020), su diseño modular lo hace adecuado para tareas avanzadas y flexibles, integrando modelos populares como ArcFace y FaceNet.
2. **FaceNet**: Schroff et al. (2015) destacan su efectividad en clustering y comparación de rostros, especialmente en bases de datos grandes.
3. **ArcFace**: Deng et al. (2018) destacan su capacidad para mejorar la separación entre clases mediante su función de pérdida angular margin loss.
4. **Dlib (ResNet)**: Según King (2009), es una opción eficiente para prototipos rápidos debido a su implementación ligera.
5. **SRN-CFD**: Zhu et al. (2018) subrayan su precisión en condiciones adversas, especialmente en rostros pequeños o en escenarios densos.

Conclusión de Reconocimiento Facial

DeepFace: La Opción Más Versátil

DeepFace se destaca como el modelo más completo debido a su compatibilidad con múltiples algoritmos y su capacidad para realizar tareas avanzadas como análisis emocional, de género y edad. Serengil (2020) destaca su integración modular, lo que lo convierte en una herramienta ideal para aplicaciones comerciales y multifuncionales.

- **Casos ideales**:
    - Sistemas comerciales.
    - Aplicaciones que requieren flexibilidad y precisión consistente.

Modelo Especializado: SRN-CFD

SRN-CFD es particularmente efectivo en escenarios densos, como multitudes o rostros pequeños, gracias a su enfoque de refinamiento selectivo. Zhu et al. (2018) lo posicionan como un modelo valioso para tareas de vigilancia avanzada y análisis en tiempo real.

- **Casos ideales**:
    - Vigilancia en multitudes.
    - Entornos con rostros pequeños o condiciones adversas.

Recomendación Final

1. **Para aplicaciones generales y multifuncionales**: **DeepFace** es la mejor elección por su versatilidad y soporte para tareas avanzadas.
2. **Para entornos desafiantes y específicos**: **SRN-CFD** ofrece precisión excepcional en escenarios densos, aunque con un mayor costo computacional.

Arquitectura de RetinaFace

Backbone: ResNet con Feature Pyramid Network (FPN)

1. **Uso de ResNet**:
    - RetinaFace utiliza una red **ResNet** como backbone (normalmente ResNet-50 o ResNet-101) para extraer características profundas de las imágenes de entrada. ResNet se ha convertido en un estándar en redes neuronales profundas gracias a su diseño con bloques residuales, que permite un entrenamiento más estable y eficiente (He et al., 2016).
2. **Feature Pyramid Network (FPN)**:
    - La **FPN** combina características de múltiples niveles de resolución extraídas de diferentes capas de ResNet (C2, C3, C4, C5). Según Lin et al. (2017), esto mejora significativamente la capacidad del modelo para manejar variaciones de escala, un desafío común en la detección de rostros pequeños o lejanos.

   Ejemplo: Para rostros pequeños, las características de P2 (160x160x256) proporcionan detalles de alta resolución, mientras que P5 (40x40x2048) captura características globales más abstractas.

Módulo de Contexto

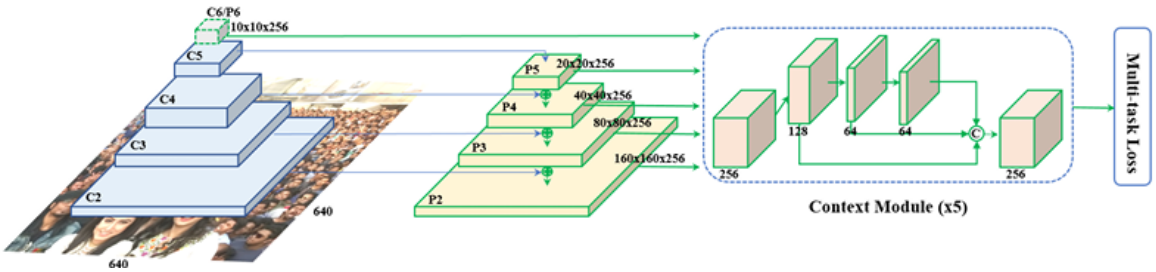
El Context Module es un componente clave diseñado para mejorar la detección facial en condiciones adversas.

1. **Función**:
    - Captura el contexto alrededor de cada región candidata mediante convoluciones adicionales, lo que permite al modelo diferenciar mejor los rostros de su entorno (Deng et al., 2019).
2. **Impacto**:
    - Según Deng et al. (2019), este módulo es especialmente útil en escenarios como multitudes o condiciones de iluminación variable, donde el contexto es crucial para una detección precisa.

Multitask Loss

1. **Optimización Simultánea**:
    - RetinaFace combina múltiples pérdidas en un solo objetivo de entrenamiento, conocido como *multitask loss*. Este enfoque mejora tanto la detección como la localización precisa de landmarks. Según el marco general de aprendizaje multitarea en redes neuronales, esto permite al modelo compartir representaciones aprendidas entre tareas relacionadas, lo que mejora la eficiencia y precisión general (Caruana, 1997).
2. **Componentes de la Pérdida**:
    - **Clasificación**: Determina si una región contiene un rostro.
    - **Regresión de Bounding Box**: Ajusta la ubicación exacta del rostro.
    - **Regresión de Landmarks**: Predice posiciones clave como ojos, nariz y boca (Deng et al., 2019).

Explicación del Funcionamiento del Modelo



La imagen y las etapas de procesamiento descritas provienen directamente del artículo original de Deng et al. (2019).

1. **Extracción de Características**:
    - La imagen de entrada pasa a través de una **ResNet** con **FPN**, donde cada nivel de resolución extrae diferentes niveles de detalle.
    - Ejemplo: Las características de **P2** tienen una resolución de 160x160 con 256 canales (Deng et al., 2019).
2. **Fusión Contextual**:
    - Las características de cada nivel (P2, P3, P4, P5) se refinan mediante el Context Module, lo que permite al modelo procesar información contextual adicional.
3. **Predicción con Multitask Loss**:
    - Clasificación, bounding box y landmarks son predichos simultáneamente, como se detalla en Deng et al. (2019).

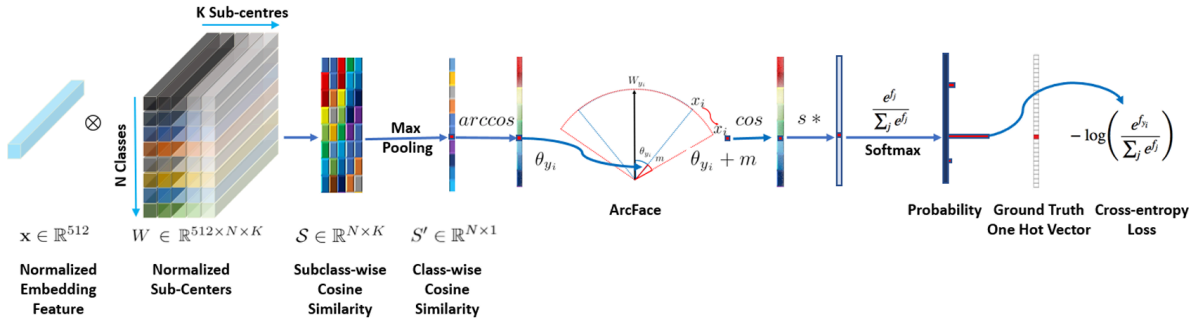
Arquitectura de ArcFace

ArcFace se fundamenta en el diseño de una pérdida angular con margen (*Angular Margin Loss*) que mejora significativamente la discriminación entre clases en tareas de reconocimiento facial. Este enfoque innovador combina normalización de características, separación angular y funciones avanzadas de pérdida para maximizar el rendimiento.

Componentes Principales de ArcFace

1. **Normalización de Características y Pesos**
    - Las características de entrada (representaciones embebidas del rostro) son generadas por una red backbone como ResNet y se normalizan para mantener una magnitud fija. Esto asegura que las comparaciones se realicen únicamente en términos de direcciones y no de magnitudes (Deng et al., 2018).
    - Además, los pesos de la última capa completamente conectada se normalizan, representando los centros de clase en un espacio de características. Según Ioffe & Szegedy (2015), la normalización es crucial para estabilizar el aprendizaje y evitar problemas causados por valores extremos.
2. **Función de Pérdida Angular Margin Loss**
    - ArcFace introduce un margen angular (*m*) adicional que separa las clases en el espacio angular. Este margen mejora la discriminación al forzar al modelo a maximizar la distancia entre clases cercanas (Deng et al., 2018).
    - El concepto de márgenes en funciones de pérdida se inspiró en trabajos previos, como el de Liu et al. (2017), quienes exploraron la pérdida angular como precursor del diseño de ArcFace.
3. **Proyección al Espacio Angular**
    - La similitud de coseno entre las características normalizadas y los centros de clase se convierte en ángulos mediante la función arccos. Esto permite al modelo optimizar directamente en el espacio angular, donde las separaciones son más interpretables y efectivas (Deng et al., 2018).

Explicación del Funcionamiento



El flujo de datos del modelo se basa en el paper original de Deng et al. (2018):

1. **Entrada Normalizada**

$$\mathbf{x} \in \mathbb{R}^{512}$$

   - Las características de entrada (512 dimensiones) se normalizan para garantizar consistencia en las comparaciones. Esto evita que las magnitudes influyan en las predicciones, asegurando que solo las direcciones sean relevantes.
2. **Centros de Subclase Normalizados**

$$W \in \mathbb{R}^{512 \times N \times K}$$

   - Los pesos normalizados representan los centros de cada clase, subdivididos en subclases (*K subclases*) para capturar variaciones intra-clase, como iluminación o expresiones.
3. **Similitud de Coseno por Subclases**

$$\mathcal{S} \in \mathbb{R}^{N \times K}$$

   - La similitud de coseno entre las características y los sub-centros se calcula y se optimiza mediante un proceso de *max pooling*, seleccionando la máxima similitud por clase (Deng et al., 2018).
4. **Margen Angular (ArcFace Loss)**

$$S' \in \mathbb{R}^{N \times 1}$$

   - Se aplica un margen angular (*m*) al ángulo correspondiente a la clase correcta, aumentando la separación entre clases y mejorando la discriminación (Liu et al., 2017; Deng et al., 2018).
5. **Softmax y Pérdida**

   - El vector ajustado se pasa por una capa **softmax** para generar probabilidades claras, y se calcula la pérdida de entropía cruzada usando el vector de verdad (Deng et al., 2018).

Importancia de Cada Componente

1. **Margen Angular**

   - Introducir márgenes angulares mejora la capacidad del modelo para distinguir rostros similares. Según Deng et al. (2018), esto es crucial para aplicaciones como autenticación biométrica.

2. **Normalización**
    - Estabiliza el proceso de aprendizaje y garantiza que las comparaciones sean consistentes entre características y pesos (Ioffe & Szegedy, 2015).
3. **Softmax Final**
    - Facilita la clasificación al convertir los resultados en probabilidades interpretable (Deng et al., 2018).

Implementación de DeepFace con RetinaFace y ArcFace

En esta sección se describe cómo implementar **DeepFace** utilizando **RetinaFace** para la detección facial y **ArcFace** para el reconocimiento facial. Se abarca el flujo completo, desde la entrada de imágenes hasta la comparación de rostros, destacando las ventajas de esta herramienta y proporcionando ejemplos prácticos.

Requisitos Previos

Instalación de DeepFace y Dependencias Para comenzar, es necesario instalar **DeepFace** y las bibliotecas requeridas. Los comandos son los siguientes:

```
pip install deepface

pip install retina-face

pip install tensorflow keras opencv-python
```

- **DeepFace**: Proporciona una API simple y modular para la detección y el reconocimiento facial.
- **RetinaFace**: Backend avanzado para detección de rostros, optimizado para precisión en condiciones adversas.
- **TensorFlow y Keras**: Frameworks necesarios para ejecutar los modelos preentrenados utilizados por DeepFace.
- **OpenCV**: Para el procesamiento de imágenes.

Según Serengil (2020), DeepFace encapsula múltiples modelos de detección y reconocimiento facial, permitiendo trabajar con arquitecturas complejas mediante comandos simples.

Potencialidades del Código

1. **Simplicidad de Uso**
    - DeepFace abstrae la complejidad de los modelos avanzados, permitiendo implementar detección y reconocimiento facial en pocas líneas de código.

- Es compatible con modelos preentrenados, eliminando la necesidad de configuraciones iniciales complejas (Serengil, 2020).

2. **Flexibilidad**

   - Soporta varios backends de detección facial, incluidos **RetinaFace** y **MTCNN**, así como múltiples modelos de reconocimiento facial como **ArcFace**, **FaceNet** y **VGG-Face**.
   - Permite ampliar el análisis a emociones, edad y género, lo que lo hace versátil para diferentes aplicaciones (Deng et al., 2018).

3. **Eficiencia en Recursos**

   - DeepFace puede ejecutarse en CPUs, lo que lo hace accesible para dispositivos con recursos limitados.
   - Sin embargo, para grandes volúmenes de datos o tareas intensivas, se recomienda utilizar una GPU compatible con CUDA para mejorar el rendimiento (Ioffe & Szegedy, 2015).

4. **Aplicaciones Versátiles**

   - La flexibilidad y modularidad de DeepFace lo hacen útil en una variedad de contextos, como:
     - Sistemas de autenticación biométrica y control de acceso.
     - Análisis demográfico y emocional para marketing.
     - Estudios académicos y experimentación con reconocimiento facial.

Ejemplo Práctico

A continuación, se presentan dos ejemplos clave que demuestran la simplicidad y potencia de DeepFace.

Comparar Dos Rostros

Este flujo utiliza **RetinaFace** como backend de detección facial y **ArcFace** para generar representaciones embebidas de los rostros.

```python
from deepface import DeepFace

# Comparar imágenes con ArcFace y RetinaFace
img1 = r"C:\Users\Jhamil\Desktop\IA2\Lab10\img_align_celeba\img_align_celeba\000023.jpg"
img2 = r"C:\Users\Jhamil\Desktop\IA2\Lab10\img_align_celeba\img_align_celeba\000015.jpg"

resultado = DeepFace.verify(
    img1_path=img1,
    img2_path=img2,
    model_name="ArcFace",
    detector_backend="retinaface"
)

print("¿Son la misma persona?:", resultado["verified"])
print("Distancia de similitud:", resultado["distance"])
```

Analizar Atributos Faciales

DeepFace también permite extraer información avanzada, como edad, género, emociones y raza.

```
# Analizar atributos faciales (edad, género, emoción, raza)
analisis = DeepFace.analyze(
    img_path=img1,
    actions=["age", "gender", "emotion", "race"]
)

print("Edad estimada:", analisis[0]["age"])
print("Género:", analisis[0]["gender"])
print("Emoción dominante:", analisis[0]["emotion"])
print("Raza predominante:", analisis[0]["dominant_race"])
```

Bibliografía

1. Caruana, R. (1997). Multitask Learning. *Machine Learning, 28*(1), 41–75. Recuperado de https://doi.org/10.1023/A:1007379606734

2. Deng, J., Guo, J., Xue, N., & Zafeiriou, S. (2018). ArcFace: Additive Angular Margin Loss for Deep Face Recognition. *arXiv preprint arXiv:1801.07698*. Recuperado de https://arxiv.org/abs/1801.07698

3. Deng, J., Guo, J., Zhou, Y., Yu, J., Kotsia, I., & Zafeiriou, S. (2019). RetinaFace: Single-stage Dense Face Localisation in the Wild. *arXiv preprint arXiv:1905.00641*. Recuperado de https://arxiv.org/abs/1905.00641

4. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Recuperado de https://arxiv.org/abs/1512.03385

5. Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *Proceedings of the International Conference on Machine Learning (ICML)*. Recuperado de https://arxiv.org/abs/1502.03167

6. King, D. (2009). Dlib: A Toolkit for Machine Learning and Face Detection. *Documentación Oficial*. Recuperado de http://dlib.net/

7. Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature Pyramid Networks for Object Detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Recuperado de

https://arxiv.org/abs/1612.03144

8. Liu, W., Wen, Y., Yu, Z., & Yang, M. (2017). SphereFace: Deep Hypersphere Embedding for Face Recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Recuperado de https://arxiv.org/abs/1704.08063

9. Medium. (2021). What is the Best Face Detector? Recuperado de https://medium.com/pythons-gurus/what-is-the-best-face-detector-ab650d8c1225

10. OpenCV. (2022). YuNet: Lightweight Face Detector. *GitHub Repository*. Recuperado de https://github.com/opencv/opencv_zoo/tree/main/models/face_detection_yunet

11. PyPI. (2023). DeepFace. Recuperado de https://pypi.org/project/deepface/

12. Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A Unified Embedding for Face Recognition and Clustering. *arXiv preprint arXiv:1503.03832*. Recuperado de https://arxiv.org/abs/1503.03832

13. Serengil, S. (2020). DeepFace: A Lightweight Framework for Deep Learning Face Recognition. *GitHub Repository*. Recuperado de https://github.com/serengil/deepface

14. Zhu, X., Pang, R., Deng, C., & Zhao, D. (2018). Selective Refinement Network for Compact Face Detection. *arXiv preprint arXiv:1812.10652*. Recuperado de https://arxiv.org/abs/1812.10652