Facultad De Ingeniería
Ingeniería Civil informática

Segmentación de Imágenes *NIR* de Iris Empleando Técnicas de Deep Learning

Tesis de pregrado
Para optar al título de Ingeniero Civil Informático

Autor:
Andrés Eduardo Valenzuela González

Profesor Guía:
PhD. Juan Eduardo Tapia Farias

Santiago, Chile
2019

Índice General

Índice de figuras

Índice de Tablas

CAPÍTULO 1

Introducción

1. Contexto y Motivación

En los tiempos actuales la seguridad informática es un tópico recurrente en sectores de salud, gubernamentales, bancarios, aeropuertos internacionales, marketing basado en internet, entre otros. Por ello la identificación correcta y fiable de una persona que requiere de estos servicios es crucial al momento de tratarse de seguridad [28]. Dentro de las técnicas de seguridad más utilizadas podemos encontrar el uso de contraseñas escritas, credenciales de identificación, firmas, entre otras, lo cual puede representar un eventual riesgo si las contraseñas se olvidan, las credenciales se extravían o cuando las firmas son falsificadas. Dado lo mencionado, la biometría representa una respuesta de carácter importante gracias al rendimiento y nivel de seguridad que provee [29, 30, 31].
Por definición, biometría se refiere al estudio de la identificación de personas basado en sus rasgos físicos o comportamientos biológicos [28]. Entre los rasgos biométricos se pueden incluir la escritura a mano, la manera de caminar o firmas escritas, mientras que los rasgos físicos comunes son el rostro [29], huellas dactilares [32], venas [33], retina [34], el iris [35, 36, 37] entre otros.

El iris es considerado como una modalidad biométrica muy interesante debido a su unicidad, estabilidad a través del tiempo y por su protección ante degradaciones externas puesto que es un órgano interno. Todos estos factores conducen a una precisión de identificación notablemente alta. Aún así, el rendimiento de un sistema de reconocimiento de iris es un problema crítico cuando la adquisición del iris está menos restringida frente a factores como el reflejo de luz natural sobre el iris u otras oclusiones. Tales problemas, cuando el iris se adquiere en situaciones más realistas, están en el centro de trabajos de investigación [38].

Luego, la tarea de segmentación es importante puesto que representa la parte fundamental de un sistema de reconocimiento de iris ya que una segmentación deficiente puede generar problemas más tarde al momento de realizar la comparación de plantillas de iris comprometiendo el desempeño final del sistema de reconocimiento de iris [39].

En la figura 1.1 se presenta la estructura externa del ojo conformada por al menos tres elementos notorios (destacados en negrita): **La pupila** (región negra localizada en el centro), rodeado por **el iris** (región texturizada), y este ultimo rodeado por **la esclera** (el blanco del ojo), cada uno con sus limites definidos. Además, el conjunto de la estructura ocular presentada es acompañada por pestañas y cejas, las cuales pueden interceptar los limites de cada región [36, capítulo 7.1]. Con base a estudios realizados [40, 41, 42, 43], el iris (figura 1.1) cumple con características relevantes para protagonizar un papel importante en un sistema de seguridad ya que es un órgano protegido externamente visible, con un patrón único y se mantiene estable a lo largo de la vida del individuo [40, 44, 28]. Conforme a lo antes dicho, los sistemas biométricos de

reconocimiento de iris son capaces de responder a la necesidad de seguridad [45] utilizando el iris por su unicidad, aleatoriedad[1] [46, 38, 39] y tolerancia a falsificaciones [47].

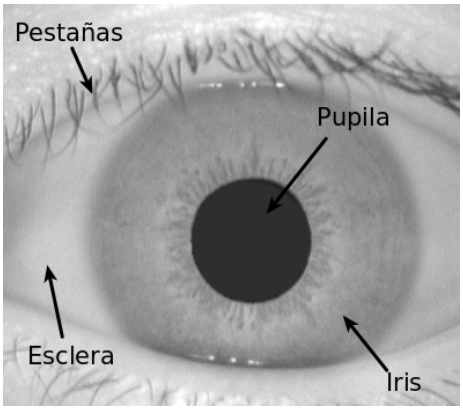


Figura 1.1: Imagen en espectro *NIR* de la estructura externa del ojo [1].

1.1. Sistema de Reconocimiento de Iris en espectro NIR

Para fines explicativos, el primer sistema funcional de reconocimiento de iris fue patentado [48] e implementado [49, 43] por John Daugman. Aun siendo acreditado por ser el más exitoso y conocido, diversos sistemas han sido desarrollados [50, 40, 51, 52, 53, 2, 39], los cuales en su mayoría comparten el mismo flujo de trabajo [2, 22] el cual comprende la adquisición de la imagen (1), la segmentación del iris (2), la normalización del iris (3) y la extracción de características con búsqueda de coincidencia para la identificación o verificación (4). Mencionado flujo es detallado a continuación:

1. **Adquisición de Imagen:**
   El rendimiento de un sistema biométrico de reconocimiento de iris típicamente depende de las condiciones de adquisición de la imagen debido a que su calidad afecta al nivel de características de los limites del iris [40, 54, 55, 56, 57]. En esta etapa la imagen del ojo suele ser adquirida por medio de cámaras en espectro visible [58, 59, 60, 22] o por medio de sensores infrarrojos [24, 54, 56, 61, 22]. Algunos de los estándares de adquisición de imágenes que debe cumplir un sistema biométrico de reconocimiento de iris son las ISO/IEC 19794-6:2005[2] e ISO/IEC 19794-6:2011[3].

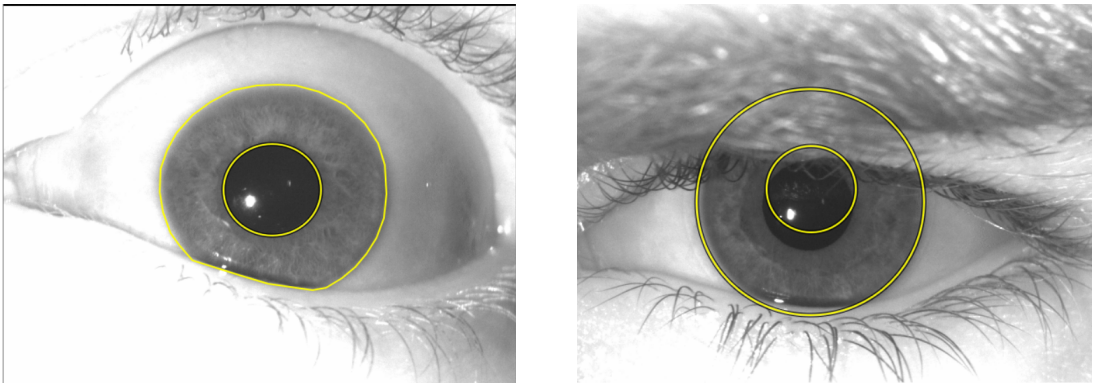
2. **Segmentación de Iris:**
   En este paso el sistema se encarga de localizar e identificar los limites del iris con el fin de separar esta estructura de las partes que no son iris (párpados, cejas, pestañas, pupila y esclera) para posteriormente realizar la extracción de características relevantes [2, 62, 39, 22]. Existen varios métodos de segmentación [63, 64, 65, 66, 67, 68], muchos como alternativa al método de segmentación propuesto por Daugman [43]. En lo que concierne a este paso, se presentan errores [69] tales como:

   - Error en la ubicación del centro de la pupila.
   - Error en la estimación del radio de la pupila (o equivalente en modelos de pupila no circulares).
   - Error en la ubicación del centro del iris.

[1] Por aleatoriedad se refiere a la complejidad combinacional respecto a diferentes personas, la cual se extiende por 249 grados de libertad y genera una entropía discriminante de alrededor de 3.2 bits por milímetro cuadrado.
[2] International Organization for Standardization: https://www.iso.org/standard/38750.html
[3] International Organization for Standardization: https://www.iso.org/standard/50868.html

- Error en la estimación del radio del iris (o equivalente en modelos de iris no circulares).
- Desviación de limites de la pupila o del iris (mal formaciones).
- Errores causados por la oclusión de cejas y/o pestañas.

Si bien la calidad del sistema de reconocimiento de iris depende en parte del paso 1, la segmentación de iris es crucial para el rendimiento final del sistema [70, 71, 22] puesto que una segmentación deficiente del iris puede generar errores en la sección de normalización y codificación [72, 70, 69, 2, 62, 73]. En la figura 1.2a se puede apreciar una correcta segmentación de iris, mientras que en la figura 1.2b se denota una segmentación no competente, donde se destaca la localización inexacta de la pupila y la inclusión de pestañas y cejas dentro de lo que correspondería al segmento del iris.



(a) Imagen bien segmentada [1].

(b) Imagen mal segmentada [1].

Figura 1.2: Diferencia visual entre una correcta y una incorrecta segmentación de iris.

3. **Normalización:**
   Una vez que la región del iris ha sido satisfactoriamente segmentada, la siguiente etapa consiste en redimensionar esta región de iris para disponer siempre de las mismas dimensiones con el fin de permitir posteriores comparaciones. A continuación es aplicado el proceso de normalización, el cual consiste en estirar la región de iris segmentada y así asegurar que dos imágenes de un mismo ojo bajo diferentes condiciones poseerán las mismas características distintivas en la misma ubicación espacial. Para esto es considerado el centro de la pupila como punto de referencia y vectores radiales pasan por sobre la región del iris tal como es mostrado en la figura 1.3. Luego, una cantidad de píxeles de datos son seleccionados a lo largo de cada linea radial y esto se define como la resolución radial. El número de lineas radiales que rodean la región del iris se define como la resolución angular. Dado que la pupila puede ser no concéntrica, es necesaria una formula de reasignación para volver a escalar los puntos según el ángulo alrededor del círculo. Esto esta dado por la formula

$$r' = \sqrt{\alpha}\beta \pm \sqrt{\alpha\beta^2 - \alpha - r_I^2} \tag{1.1}$$

Fórmula de reasignación de iris, con $\alpha = o_x^2 + o_y^2$ y $\beta = \cos(\pi - \arctan(\frac{o_y}{o_x} - \theta))$.

Donde el desplazamiento del centro de la pupila en relación con el centro del iris viene dado por o_x, o_y, r' es la distancia entre el borde de la pupila y el borde del iris en un ángulo θ alrededor de la región y r_I es el radio del iris. La fórmula de reasignación primero da el radio de la región del

iris en forma de "dona"[4] como una función del ángulo θ. Finalmente se elige un número constante de puntos a lo largo de cada línea radial, de modo que se toma un número constante de puntos de datos radiales, independientemente de qué tan estrecho o ancho sea el radio en un ángulo particular. El patrón normalizado se creó para encontrar las coordenadas cartesianas de los puntos de datos desde la posición radial y angular en el patrón normalizado. Desde la región del iris con forma de "dona", la normalización produce una matriz 2D con dimensiones horizontales de resolución angular y dimensiones verticales de resolución radial [2, 43].

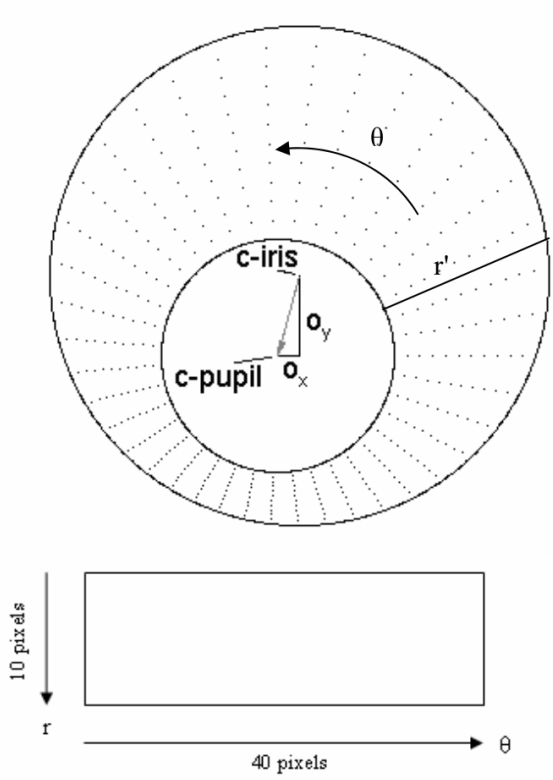


Figura 1.3: Esquema del proceso de normalización con resolución radial de 10 píxeles y resolución angular de 40 píxeles. El desplazamiento de la pupila en relación con el centro del iris se exagera para fines ilustrativos [2].

4. **Codificación de Características y Búsqueda de Coincidencia Para Autenticación:**
   A fin de proporcionar el reconocimiento preciso de un individuo, solo la información discriminante que se presenta en los patrones del iris normalizado debe ser extraída. Esta información es codificada [74, 75, 76, 77, 51, 52, 78] generando una plantilla única (Ver figura 1.4) para la región de iris con la cual se podrán llevar a cabo comparaciones[5] con plantillas que el sistema tenga almacenadas (Ver figura 1.5). Estas comparaciones son realizadas con base a una o más métricas de coincidencia [79, 80, 50] que disponga el sistema, las cuales proveerán una medida de similitud entre dos plantillas de iris con la cual discriminar con alta confianza si estas plantillas corresponden o no al mismo iris [2]. La plantilla del iris será comparada o almacenada dependiendo de la acción solicitada, es decir, si se requiere registrar o autenticar al usuario.

[4] Se refiere así a la forma del iris dado que al no tener la pupila en el centro, el iris queda ahuecado formando una dona.

[5] La comparación entre dos plantillas del mismo iris se denomina comparación *intra-class*, mientras que una comparación entre dos plantillas de iris distintos es denominada comparación *inter-class* [2]

[00, 10, 01, 01, 00, 10, 10, 01, 11, 11, 00, 01, 01, 10, 11, 01, 10, 00, 01, 01, 01,
10, 01, 11, 00, 00, 00, 11, 00, 10, 01, 10, 01, 10, 11, 11, 00, 00, 00, 10, 01, 01, 11]

Figura 1.4: Plantilla de iris codificado.

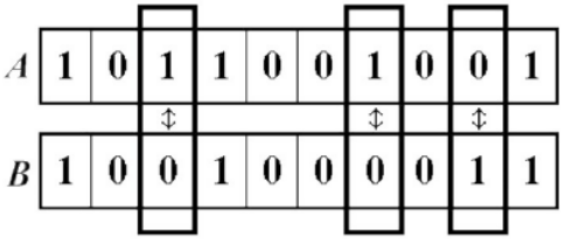


Figura 1.5: Ejemplo de comparación empleando distancia *Hamming*. La distancia *Hamming* es de 3 unidades.

En el flujo de trabajo presentado, el problema de calidad de adquisición de imagen (Paso 1) suele ser solapado mejorando la calidad de la cámara y/o cambiando el espectro visual en el cual se trabaja, mientras que la segmentación depende netamente del método utilizado, por lo tanto **esta etapa termina siendo el paso crítico del sistema** de reconocimiento de iris puesto que una segmentación incompetente puede inducir a errores en la generación de plantillas de iris, errores de autenticación u otros comportamientos no deseados. Bajo condiciones especificas, el iris y la pupila pueden no representar característica concéntrica[6], el iris puede estar ocluido por pestañas o parcialmente descubierto por los párpados. A estos y otros problemas se ven enfrentadas las técnicas basadas en *machine learning* y dependientes de una estructura ideal del ojo (denominadas *handcrafted features*[7] en [81]), reduciendo significativamente su rendimiento ante situaciones con cantidad de oclusiones significativas.
Por otro lado, técnicas de segmentación asociadas al campo de *Deep Learning* han demostrado superar el rendimiento de las técnicas antes mencionadas dadas las distintas capacidades de aprendizaje y tomas de decisiones inteligentes que poseen (Ver capítulo 2, estado del arte).
Ante la situación descrita, **la motivación** de la presente tesis es implementar un segmentador de iris preciso y robusto basado en *Deep Learning*, el cual disponga un rendimiento competente con el estado del arte y además pueda ser implementado en un sistema biométrico de reconocimiento de iris.

2. Marco de Trabajo

La presente tesis esta orientado al desarrollo de un modelo capaz de segmentar imágenes de ojos en espectro *NIR* de manera automática, basado en inteligencia artificial y empleando técnicas de visión computacional. Esta tesis apunta a generar conocimientos y beneficios a las industrias de seguridad, enfocado a mejorar el rendimiento de los sistemas biométricos de reconocimiento de iris.

3. Hipótesis

A partir de la investigación realizada en el estado del arte, se plantean las siguientes tres hipótesis:

[6] Presentar característica concéntrica significa que tanto el iris como la pupila comparten el mismo centro (coordenadas *(x, y)*).
[7] *Handcrafted features* se refiere a las propiedades derivadas de varios algoritmos usando la información presente en una imagen. Por ejemplo, dos características simples que pueden ser extraídas de las imágenes son los bordes y las esquinas.

1. Se estima que técnicas de segmentación asociadas al campo de *Deep Learning* permitirán extraer de manera automática y precisa información relevante del iris.
2. Se considera que es posible segmentar cada una de las zonas del ojo en espectro *NIR*.
3. Se espera obtener resultados competitivos con el estado del arte para cada una de las zonas segmentadas del ojo.

4. Objetivos

4.1. Objetivo General

El objetivo general de la presente tesis es desarrollar e implementar un segmentador de iris en espectro *NIR* por medio del uso de segmentadores basados en *Deep Learning*.
Del presente objetivo general derivan tres objetivos específicos, los cuales cada uno responderá una de las hipótesis planteadas con anterioridad.

4.2. Objetivos Específicos

1. Estudiar y analizar los distintos algoritmos de segmentación presentados en el estado del arte.
2. Diseñar una base de datos de imágenes segmentadas de iris en espectro *NIR*.
3. Entrenar modelos propuestos por el estado del arte y evaluar el desempeño de cada modelo utilizando la métrica *IoU*.

4.3. Alcance

La presente tesis pretende desarrollar un modelo de segmentación semántica de imágenes con tasas de segmentación competitivas con el estado del arte y que brinde conocimientos y beneficios técnicos tanto a las industrias de seguridad como a otros investigadores del área biométrica.

5. Metodología De Trabajo

Para resolver cada uno de los objetivos específicos planteados se seguirá la metodología mostrada en la figura 1.6:

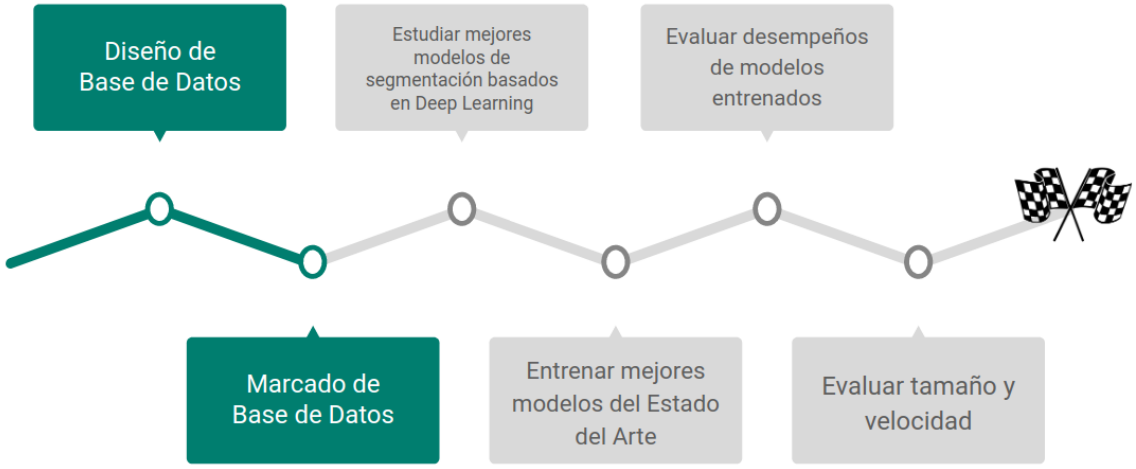


Figura 1.6: Diagrama de metodología propuesta.

La metodología en cuestión es detallada a continuación:

5.1. Metodología Objetivo Específico N°1

Con base al título de la presente tesis y palabras claves, será realizada una búsqueda del estado del arte en distintos portales de investigación tales como *IEEE*, *ResearchGate*, *Scholar Google* y *arXiv*, rebuscando escritos con hasta 3 años de antigüedad y sus algoritmos asociados al campo de *Deep Learning* capaces de realizar tareas de segmentación relacionadas con el iris con un desempeño por sobre el propuesto por el sistema de Daugman en [43]. Luego, estos trabajos serán organizados en una tabla comparativa (Ver tablas 2.11 y 2.12) para con esto obtener un acercamiento a los mejores métodos de segmentación.

5.2. Metodología Objetivo Específico N°2

La base de datos en [1] fue creada realizando capturas de imágenes de iris pertenecientes a distintas personas almacenándolas junto con su *metadata* (genero, raza, color de ojos, uso de lentes de contactos, etc) en carpetas cuyos nombres corresponden a un id único representando a cada persona, luego, dichas carpetas fueron organizadas según su fecha de captura en sub-carpetas con estas fechas por nombre de directorio. Para este labor fueron utilizadas las cámaras *LG2200* y *LG4000*, de las cuales con esta primera fueron realizadas capturas de iris por separado (ojo izquierdo, luego el derecho) mientras que con la segunda fue realizada una captura de la región orbitaria de los rostros como una sola imagen para luego ser separada en dos imágenes de ojos. Ejemplos de capturas de ambas cámaras se pueden apreciar en la figura 1.7:

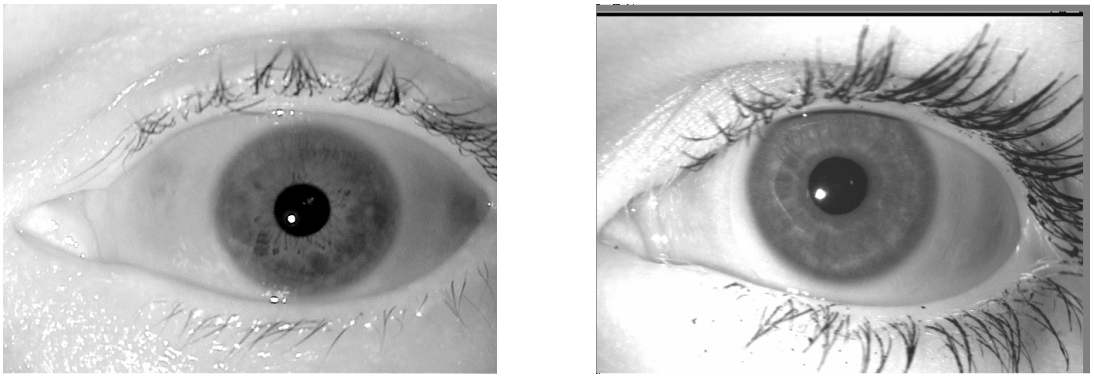


Figura 1.7: La imagen de la izquierda corresponde a una captura de la cámara LG2200 a un hombre, mientras que la imagen de la derecha corresponde a una captura de la cámara LG4000 a una mujer. Imágenes obtenida de [1].

Para la base de datos descrita serán marcadas manualmente las coordenadas de las regiones del iris, pupila, esclera izquierda, esclera derecha y maquillaje (solo en el caso de estar presente) utilizando el programa *VIA* versión 2.0.5 [3] tal como se muestra en la figura 1.8:

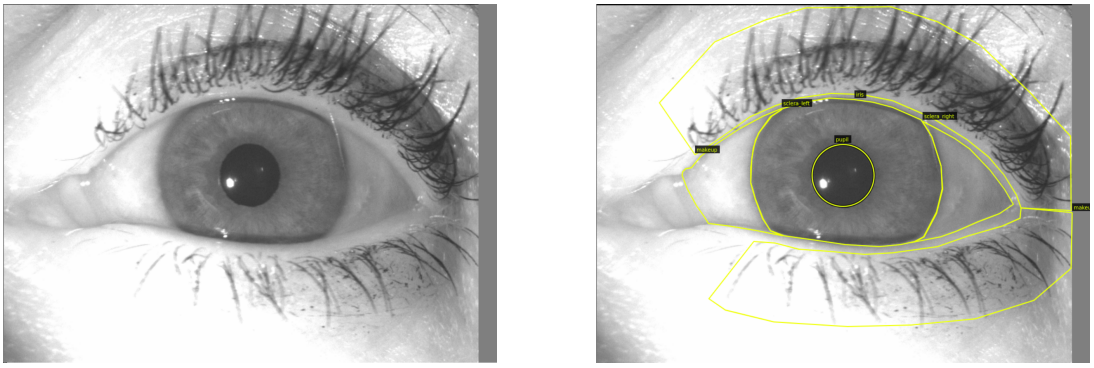


Figura 1.8: La imagen de la izquierda corresponde a una imagen sin marcar, mientras que la imagen de la derecha es su equivalente con marcas de la aplicación VIA [3]. Imagen obtenida de [1].

El uso de este programa consistirá en marcar una región de interés (de aquí en adelante *ROI*[8]) mediante la utilización de 6 distintas figuras (rectángulo, circulo, elipse, polígono, punto ó polilínea), encerrando cada una de las regiones previamente mencionadas formando un polígono cuyos puntos representarán coordenadas en la imagen, siendo para el caso de las pestañas la utilización de una polilínea. Conforme se progrese con el marcado de imágenes, cada 100 (50 ojos de mujeres y 50 ojos de hombres) será guardado el archivo de proyecto del programa *VIA* [3] como punto de control con el motivo de separar distintos conjuntos de entrenamiento para con esto registrar y evaluar el desarrollo y las mejoras de los modelos a entrenar con base a gráficos de interés.

Las coordenadas de cada *ROI* marcado serán almacenadas en una base de datos en formato *JSON* [82] para posteriormente ser recortadas y redimensionadas utilizando un programa desarrollado en lenguaje *Python* [83] encargado de leer las coordenadas de la base de datos y, de manera automática, proceder a recortar las imágenes que serán utilizadas para el entrenamiento de los modelos. Se muestra a continuación un ejemplo de la estructura de un archivo JSON con una imagen marcada:

[8]Por sus siglas en inglés *Region Of Interest*.

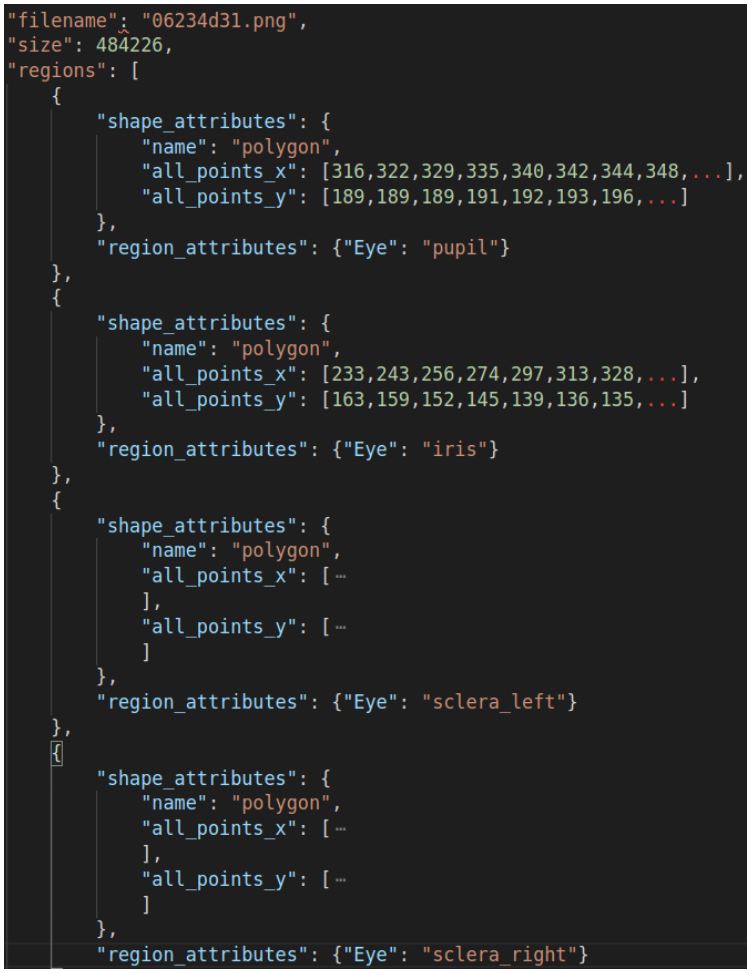


Figura 1.9: Ejemplo de marcas de imagen con programa VIA [3]. Las clases pupila, iris, esclera izquierda y esclera derecha tienen N cantidad de puntos en el eje X y N cantidad de puntos en el eje Y (pares de puntos x-y). Fue utilizada la figura polígono para marcar las cuatro clases mencionadas.

5.3. Metodología Objetivo Específico N°3

Con base a los trabajos estudiados en la sección 5.1 será entrenado un segmentador semántico utilizando el framework *Mask R-CNN* con dos modelos distintos de redes neuronales convolucionales: *ResNet50* [84] y *ResNet101* [84]. Mientras que para otra ronda de experimentos será entrenado un modelo *DenseNet* [16] desarrollado para segmentación semántica de iris. Estos modelos serán entrenados utilizando técnicas de *transfer learning* [85, 86] y *data augmentation* [87, 88], además de ser evaluados con base a las métricas *IoU* (Ver sección 4.8) con motivo de obtener un segmentador de iris con un desempeño competente con el estado del arte. Además, como propuesta de valor, se buscará segmentar el resto de las zonas del ojo, siendo estas además del iris, la pupila, la esclera (dividida en izquierda y derecha) y el maquillaje solo en caso de estar presente en la imagen del ojo. Mientras que por parte del modelo *DenseNet*, se propone como valor un modelo rápido en términos de ejecución y ligero (medido en MegaBytes) capaz de segmentar iris, pupila y esclera.

CAPÍTULO 2

Estado del Arte

En esta sección se presenta el estudio de trabajos del estado del arte con hasta 3 años de antigüedad. Para el análisis de estos fueron realizadas búsquedas en distintos portales de investigación tales como *IEEE*, *ResearchGate* , *Scholar Google* y *arXiv* utilizando palabras claves tales como *"iris"*, *"instance"*, *"semantic"*, *"segmentation"*, *"mask"*, *"near-infrared"*, *"nir spectrum"*, *"machine learning"*, *"computer vision"*, *"deep learning"* y *"convolutional neural network"*.
Los resultados de los estudios fueron analizados, construyendo con estos una tabla comparativa general (ver tablas 2.11 y 2.12) en donde se indica el autor, el método utilizado, la base de datos utilizada, la longitud de onda empleada para obtener las imágenes, el numero de imágenes utilizadas, la utilización de *data augmentation*, la cantidad de imágenes para los sets de entrenamiento y evaluación, las métricas de evaluación utilizadas y las métricas obtenidas. A continuación son expuestos los trabajos más relevantes, los cuales se estima utilizar como base para construir nuevos conocimientos con motivo de que la presente tesis signifique un aporte a las fuentes ya existentes:

En [21] los autores aplican las *Fully Convolutional Networks (FCNs)* y las *Generative Adversarial Networks (GANs)* (tecnicas basadas en *deep learning*) para la segmentación de iris en espectro visual y espectro *NIR*, obteniendo resultados prometedores en todas las bases de datos evaluadas. Las bases de datos utilizadas fueron *BioSec, CasiaI3, CasiaT4, IITD-1* para imágenes en espectro *NIR* y *NICE.I, CrEye-Iris y MICHE-I* para imágenes en espectro visual en dominios cooperativos y no cooperativos. Como enfoque propusieron localizar el iris y segmentar el mismo en dos tareas por separado.
En primer lugar, para la **detección y localización del iris** implementaron una red neuronal convolutiva la cual se encargo de realizar la detección de la región periocular[1] (*PRD* por sus siglas en ingles *Periocular Region Detection*) empleando la técnica *fine-tuning* con base al detector de objetos *YOLO* en su versión lite *Fast-YOLO* [89] (utilizaron un modelo más pequeño dado que solo necesitaban localizar una única clase, en este caso, el iris), utilizando los pesos[2] de una red preentrenada con la base de datos *ImageNet*.
La red *PRD* fue entrenada con imágenes sin ser preprocesadas y con las coordenadas de sus *ROIs* correspondientes. Luego, para la **segmentación de iris** fueron utilizadas las *FCN* y *GAN* dados sus buenos resultados en la segmentación de escleras en su trabajo anterior [90].
Como protocolo de evaluación utilizaron el protocolo del concurso *NICE.I* y además realizaron una comparación de píxel-a-píxel entre las imágenes marcadas manualmente versus las predicciones realizadas por los algoritmos (denotadas como mascaras), generando un error de segmentación promedio (denotado como *E*)

[1] Los párpados, con sus pestañas y sus glándulas, y el saco lagrimal son estructuras que forman parte de lo que se define como zona periocular.
[2] Disponible en https://plreddie.com/darknet/yolo

computado como la divergencia a nivel de píxel, denotado por la siguiente formula:

$$E = \frac{1}{(h \times w)} \sum_i \sum_j M_k(i,j) \oplus GT_k(i,j) \tag{2.1}$$

Donde *i* y *j* son las coordenadas de la mascara predicha *M* y las imágenes marcadas manualmente *GT*, *h* y *w* corresponden a la altura y anchura de la imagen, respectivamente. Bajos y altos valores para *E* representa buenos y malos resultados respectivamente. Además, se reporta la medida *F-1 score* el cual es la media armónica entre *Precision* y *Recall*.

Para la etapa de entrenamiento dividieron todos las las bases de datos de manera aleatoria en dos subsets, conteniendo uno el 80 % de las imágenes para entrenamiento y el restante para evaluación. De acuerdo a un análisis hecho, más iteraciones significaban un aumento en el rendimiento de los modelos, por lo cual 32.000 iteraciones fueron suficientes para todas las bases de datos.

Los resultados obtenidos para cada método se exponen en la siguiente tabla con sus métricas de evaluación correspondientes:

Tabla 2.1: Resumen de resultados de segmentación de iris empleando ambos protocolos de evaluación. Investigación realizada por Bezerra et al. [21]

Método	**F1 %**	**E %**
FCN_{NIR}	97,46 ± 00,74	00,44 ± 00,12
GAN_{NIR}	96,82 ± 02,83	00.74 ± 01,40
FCN_{VIS}	97.04 ± 01.21	00.96 ± 00.36
GAN_{VIS}	92.61 ± 05.86	03.02 ± 03.22

Las conclusiones realizadas por los investigadores son que los resultados de las imágenes en espectro visual (*VIS*) son ligeramente peores que las imágenes en espectro *NIR* debido a que estas primeras imágenes suelen tener más oclusiones (e.g., reflejos), además se destaca que la técnica *transfer learning* fue crucial para alcanzar las medidas obtenidas (además de ser un logro para la *FCN* debido a que esta fue entrenada con una menor cantidad de imágenes que la *GAN*). Por último, técnicas especificas como *data augmentation* pueden ser aplicadas para mejorar el rendimiento de la *GAN*.

En [22] los autores proponen la utilización de una red neuronal convolutiva (*CNN*) de tipo *Densely Connected Fully Convolutional Network* apodada *IrisDenseNet*. En los experimentos realizados, 5 bases de datos de iris en espectro visual (*VIS*) y *NIR* fueron usadas. Para las imágenes en entorno visual se usaron bases de datos de imágenes en ambientes no controlados[3] (*NICE-II*, imágenes seleccionadas de la base de datos *UBIRIS.v2* y *MICHE-I*), mientras que las imágenes en espectro *NIR* comprenden el ambiente controlado[4] (*CASIA v4.0 interval*, *CASIA v4.0 distance* y *IITD Delhi v1.0*).

El método propuesto (*IrisDenseNet*) comprende de dos partes principales: (1) *Densely Connected Encoder* y *SegNet decoder*. La elección de estos modelos fue dadas sus características, por ejemplo las *DenseNet*, por sus conexiones densas fortalecen la propagación de características, mientras que *SegNet* es prácticamente una red convolucional totalmente conectada creada para fines de segmentación semántica[5].

El entrenamiento de esta red se llevo a cabo con los tipos de imágenes mencionados anteriormente (*VIS* y *NIR*, en ambientes no controlados y controlados respectivamente), resaltando que las imágenes en ambiente

[3] Ambientes externos con luminosidad no controlada.
[4] Ambientes interiores con luminosidad controlada.
[5] Segmentacion semántica significa analizar imágenes y asociar cada píxel de estas a una clase especificada.

no controlado declaran ciertas dificultades tales como desenfoques por movimientos, oclusiones por cejas y pestañas, lentes, capturas fuera de ángulo, iluminaciones no uniformes y algunas capturas parciales de las imágenes dado rotaciones irregulares. La mitad de las imágenes de la base de datos *NICE-II* fueron usadas para entrenamiento, mientras que su mitad restante para evaluación basado en el método de *two-fold cross validation*[6]. Además, para mejorar la calidad de la segmentación final, se implemento la técnica de *data augmentation*. Por último, el entrenamiento de la red fue hecho desde cero con una base de datos experimental perteneciente a ellos.
La mención de la utilización de *data augmentation* fue debido a la cantidad limitada de imágenes a disposición, por lo cual cada set de entrenamiento fue aumentado 12 veces aplicando recortes, dimensiones con interpolación, volteo de imagen solo en dirección horizontal y traslación horizontal y vertical.
En la evaluación propuesta para *IrisDenseNet*, una imagen sin pre-procesar es proporcionada al modelo entrenado, esta imagen pasa por el *dense encoder* y luego por el *decoder*, obteniendo por salida una mascara binaria[7], la cual es usada para generar y evaluar los resultados de segmentación. El rendimiento de *IrisDenseNet* es evaluado utilizando el protocolo de *NICE-I* [91] (ver ecuación 2.1). Junto con esto, se obtiene el promedio general del error de segmentación (denotado como E_a, el cual es calculado ponderando el error de clasificación E (ver formula 2.1) de cada imagen en la base de datos:

$$E_a = \frac{1}{t} \sum_i E_i \tag{2.2}$$

Donde t representa el numero total de imágenes a evaluar mientras que E_i representa el error de segmentación de una imagen (formula 2.1).

El valor de E_a siempre se encuentra en el rango de [0, 1]. Si el valor es cercano a "0", indica un mínimo de error, mientras que E_a se acerca a "1", indica un error más alto. También se analiza la relación entre el aumento del error promedio evaluado (E_a) y el uso excesivo de *data augmentation* tal como se muestra en la siguiente tabla:

Tabla 2.2: Relación entre *Data Augmentation* y error promedio evaluado. Investigación realizada por Arsalan et al. [22]

Método	Ea
Excesivo *data augmentation*	0,00729
Data Augmentation cambiando el contraste y brillo de la imagen de iris	0,00761
Data Augmentation propuesto	0,00695

Los resultados obtenidos fueron medidos con base a la ecuación 2.2, el cual representa el error promedio de la base de datos. Los resultados se presentan gráficamente en [22] página 15, disponiendo de dos tipos de errores: falsos positivos (color verde) y falsos negativos (color rojo). Mientras, los casos verdaderos positivos son aquellos en que el píxel de iris fue correctamente clasificado como píxel de iris (representado en negro). Los errores falsos positivos son causados usualmente por píxeles de pestañas o píxeles próximos a reflejos de la pupila debido a que sus valores de píxeles son similares a los píxeles del iris, mientras que los errores falsos negativos son causados por los reflejos de lentes o un área más oscura en el iris. Entre otras métricas evaluadas, también se exponen *Precision*, *Recall* y *F1-score*. En la siguiente tabla se exponen las métricas obtenidas por Arsalan et al. [22]:

[6] La validación cruzada usualmente es utilizada para garantizar que los datos de entrenamiento y prueba sean independientes entre si.
[7] Por mascara binaria se entiende una imagen negra con píxeles blancos en donde se ubica la clase a predecir por el modelo entrenado.

Tabla 2.3: Error promedio ponderado (E_a, ver ecuación 2.2), *IrisDenseNet* [22]

Base de Datos	Ea
NICE-II	0,00695
MICHE-I	0,0020
CASIA v4.0 Distance	0,0034

Tabla 2.4: Medidas RPF obtenidas, *IrisDenseNet* [22], donde μ representa el promedio y σ representa la desviación estándar.

Base de Datos	Métricas RPF Obtenidas
CASIA v4.0 interval	R: (μ: 97,10) / (σ: 2,12) P: (μ: 98,10) / (σ: 1,07) F: (μ: 97,58) / (σ: 0,99)
IITD	R: (μ: 98,00) / (σ: 1,56) P: (μ: 97,16) / (σ: 1,40) F: (μ: 97,56) / (σ: 0,84)

Los comentarios realizados por los investigadores en [22] son que en los métodos convencionales se elimina[8] la información espacial durante capas continuas de convolución, por lo cual se estima imperante mantener estas características luego de las convoluciones. Un método para solucionar esto es aplicando *dense connection* (ver figura 4 en [22]). La calidad de las características se pueden apreciar en figuras 19 y 20 en [22].

En cuanto a la segmentación, los autores recalcan la comparación entre su método propuesto (*IrisDensenet*) versus *SegNet*. En primer lugar, los resultados de segmentacion de iris obtenidos por *IrisDenseNet* muestran un limite de iris más fino que los resultantes de *SegNet*, lo cual disminuye el ratio de error. En segundo lugar, *IrisDenseNet* es más robusto ante la detección del limite del iris que *SegNet*. Finalmente, el método propuesto es más robusto que *SegNet* ante áreas "fantasmas"[9].

Las conclusiones en este estudio destacan que para obtener resultados de excelencia en la segmentación de iris el *encoder* utilizado fue empoderado con una *dense connection* [16], es decir, todas las capas están directamente concatenadas (conectadas) con sus capas predecesoras, lo que le otorga a la red la capacidad de reusar características para un mejor rendimiento (en vez de perderlas debido a efectos de convoluciones y *max pooling*). El método propuesto proporciona una segmentación de extremo a extremo sin necesidad de alguna actividad de preprocesamiento de imagen. Entre las pruebas realizadas, tanto imágenes en espectro visual como en espectro *NIR* obtuvieron buen rendimiento.

En [24] los autores investigan las aplicaciones de las *CNNs*[10], problemas y soluciones respecto al reconocimiento de iris en dispositivos móviles debido a que estos instrumentos se han vuelto ubicuos y el típico teléfono celular es más que un teléfono; frecuentemente los utilizamos para revisar mensajería instantánea (emails), cuentas bancarias, accedemos a servicios web y recursos empresariales. Como resultado, varias vulnerabilidades nacen y tienen por objetivo los dispositivos móviles que garantizan acceso por medio de contraseñas/PINs, por lo cual buscan implementar autenticación por biométrica a estos artefactos móviles no sin antes enfrentarse a problemas tales como la adquisición de imagen en un entorno no controlado.

[8] I.e., Las imágenes que avanzan por las capas convolutivas van reduciendo sus dimensiones, teóricamente reduciendo características pequeñas a simples píxeles, de los cuales información relevante no se puede obtener

[9] Reflejos en el iris.

[10] Convolutional Neural Networks

Estiman además que la biométrica es más robusta frente a ataques de seguridad debido a su dificultosa capacidad de clonación, por lo cual en este estudio se proponen técnicas de segmentación semántica basadas en redes neuronales convolucionales.
Dada la reducción de nivel de computo de un dispositivo móvil, el autor propone un modelo ligero diseñado para segmentación semántica de iris.
Reportan resultados excelentes en trabajos anteriores respecto a *RefineNet* [23] y *iFCEDN* (por sus siglas en ingles *improved Fully Convolutional Encoder–Decoder Network*), por lo cual se estima el uso de ambos para el trabajo final de segmentación semántica de iris.
Ambos métodos generan una mascara binaria, la cual es utilizada para evaluar las segmentaciones realizadas con las imágenes marcadas manualmente de evaluación.
Para el entrenamiento de estos modelos se utilizaron las bases de datos *IIT Delhi Iris Database version 1.0 (iitd)*, *CASIA Iris Image Database version 4.0 (casia4i)*, *CASIA Iris Subject Ageing Version 1.0 Database (casiaA)*, *PROTECT Multimodal DB (protMI)*, con las cuales se empleo la técnica de *five-fold cross validation*, generando 5 sets distintos de entrenamiento y otros 5 de validación. Los parámetros de entrenamiento de *RefineNet* y *iFCEDN* son expuestos en las siguientes tablas respectivamente:

Tabla 2.5: Parámetros de entrenamiento de *RefineNet* [23]. Obtenidos de [24].

Parameter	Batch size	Epoch	Momentum	Learning Rate
Values	8	1.000	0,1	0,9

Tabla 2.6: Parámetros de entrenamiento de *iFCEDN*. Obtenidos de [25] según se indica en [24].

Parameter	Iterations	Momentum	Learning Rate
Values	10.000	0,9	0,001

La base de datos *protMI* fue construida y segmentada a mano por los investigadores de este estudio, pero a pesar de la buena calidad de las imágenes de iris al principio, las mismas demuestran algunos detalles no favorables cuando son adquiridas usando un dispositivo de mano, además de estar en un ambiente semi-controlado (dentro de un vehículo).
Las condiciones ambientales, el dispositivo móvil de captura y el hecho de que la adquisición de imagen fue realizada por usuarios no entrenados, condujeron a un incremento en la rotación de las imágenes capturadas. Este hecho fue el core de las evaluaciones realizadas en el estudio dado que las otras bases de datos no disponían de imágenes tan rotadas como las adquiridas. Imágenes extremadamente rotadas o mal adquiridas fueron eliminadas de la base de datos.
Para el entrenamiento y evaluación de los modelos propuestos fueron utilizadas las métricas de evaluación error tipo 1 (denotado por *E1*) y error tipo 2 (denotado por *E2*) de la competencia *NICE-I* [91], además de la medida *F1-score*. Los resultados revelaron que el modelo *RefineNet* supero al modelo *iFCEDN* en todas las bases de datos excepto en la creada en esta investigación (protMI), lo cual apunta (según la investigación) que el modelo de segmentación ideal dependerá del conjunto de datos objetivo a trabajar. Aun así, independiente del conjunto de características que la base de datos posea, la proposición de selección de un modelo ideal para una base de datos especifica es altamente cuestionable. A continuación se presentan los resultados obtenidos para cada modelo en la siguiente tabla:

Las conclusiones realizadas por los investigadores apuntan que la adquisición de imágenes por parte de usuarios no capacitados refleja más el efecto de entorno no controlado, lo cual añade más desperfectos

Tabla 2.7: Resultados de *RefineNet* [23] e *iFCEDN* obtenidos de [24]. Se marca con negrita la excepción producida por *iFCEDN* con la base de datos *protMI*.

Modelo	Base de Datos	E1	E2	F1-score
RefineNet	casia4i	0,009	0,011	0,984
	casiaA	0,005	0,012	0,972
	iitd	0,015	0,018	0,974
	protMI	**0,044**	**0,151**	**0,746**
iFCEDN	casia4i	0,021	0,028	0,962
	casiaA	0,007	0,020	0,966
	iitd	0,018	0,022	0,970
	protMI	**0,008**	**0,024**	**0,952**

(rotaciones, reflejos, desenfoque por movimientos, etc) a la base de datos de imágenes en general, lo cual puede representar una desventaja ante la segmentación semántica de iris en dispositivos móviles puesto que es una variable más por controlar. Aun así, este trabajo demuestra que una arquitectura de red neuronal basada en *Deep Learning* es capaz de obtener un rendimiento significativo incluso bajo imágenes con rotaciones excesivas.

En [26] los autores introducen el método de segmentación basado en *CNN*[11], cerrando la brecha entre las *CNNs* y métodos basados en geometría y teorías matemáticas[12]. Hofbauer considera nuevamente el uso de *RefineNet* [23] por sobre *iFCEDN* dados sus buenos resultados presentados en [24] (ver tabla 2.7). Para evaluar el rendimiento de segmentación, i.e., la generación de mascaras binarias, fue evaluada la *CNN* mencionada en todas las muestras de las bases de datos (*UBIRIS* [92], *IIT Delhi Iris Database version 1.0 (iitd)*, *CASIA Iris Image Database version 4.0 (casia4i)*, *ND-0405 Iris Image Dataset (ndi)*, *CASIA Iris Subject Agening Version 1.0 Database (casiaA)*, *PROTECT Multimodal DATABASE*) sin realizar cruce de datos gracias al uso del método de *five-fold cross validation*, generando así 5 sets de entrenamiento y otros 5 sets de evaluación. En conjunto con esto, la arquitectura del modelo de *RefineNet* [23] se mantuvo sin modificar, tal como la proporcionaron los autores[13], además de utilizar los parámetros dispuestos en la tabla 2.5. Luego, se realiza un proceso de parametrización de mascaras, el cual involucra los siguientes pasos: (1) preprocesar las imágenes aplicando difuminación media para suavizar los bordes, (2) generación de segmentación candidata usando la transformada de Hough [93] y (3) seleccionar los mejores candidatos como parametrización final. La implementación de esta parametrización (nombrada de aquí en adelante como *CNNHT*) da como resultado imágenes con un iris más focalizado[14], lo que permite obtener menos píxeles *non-iris* y segmentar mayor cantidad de píxeles *iris*. En la investigación destacan que es importante comenzar con las coordenadas del limite de la pupila (sin importar la fuente de luz usada, *NIR* o luz visible) debido a dos razones: (1) la forma circular esta más intacta para la pupila per se, lo cual hace más fácil encontrar un punto de partida más estable y (2) la *CNN* realiza un muy buen trabajo generalizando los píxeles de las mascaras independiente del tipo de onda de luz que se dispone de iluminación. La segunda opción fue usar la transformada de Hough circular en vez de elíptica, esto debido a que las curvas elípticas tienden a sobre ajustarse debido a las oclusiones (pestañas o cejas), lo cual dificultaría la segmentación del iris por parte de la red neuronal.

[11] Convolutional Neural Network
[12] Mencionado como *rubbersheet-transformation*.
[13] https://github.com/guosheng/refinenet
[14] Denotado como circulo candidato.

En la sección de evaluación fue usado el set de herramientas de *USIT* [94] dado a que es modular y permite cambiar el método de segmentación (entre otras partes) sin corromper otras partes del sistema. Por lo tanto, para la comparación de segmentación fue usado *CAHT* [43] (método propuesto en la investigación de Daugman), además de utilizar *WAHET* [66]. Además, fue empleado un método basado en un modelo de variación total (*TVMIRIS*) [68], el cual produce parametrizaciones. Los parámetros utilizados fueron los sugeridos en la base de datos *UBIRIS* [92], obteniendo los siguientes resultados:

Tabla 2.8: Rendimiento del reconocimiento biométrico con segmentación de mascaras parametrizadas, para el EER y FNMR@FMR=0,01 %. Sea EER: Equal Error Rate, FNMR: False Non Match Rate, FMR: False Match Rate del 0,01 %, denotado como $OP_{0,01}$ en la tabla, SE: Segmentation Error, ME: Masking Error. Todos los valores de la tabla son porcentajes. Tabla obtenida y reducida de [26].

Testset	*Seg.*	*EER*	$OP_{0,01}$	*SE*	*ME*
casiaA	*CNNHT*	2.685	9.451	0	0
	Groundtruth	1.310	5.761	0	0
casia4i	*CNNHT*	1.085	2.699	0	0
	Groundtruth	0.244	0.414	0	0
iitd	*CNNHT*	1.103	37.183	0	0
	Groundtruth	0.357	25.253	0	0
protI	*CNNHT*	8.220	31.829	0	0
	Groundtruth	7.899	29.565	0	0
ndi	*CNNHT*	23.492	37.218	0	0
	Groundtruth	23.906	33.557	0	0

Los errores producidos por el sistema son los errores de segmentación (*SE*)[15] y los errores de mascara (*ME*)[16].

Una primera conclusión que se puede realizar observando la tabla es que no se produjo ningún error de mascara ni de segmentación en ninguna base de datos. Dado que la información de textura se pierde en *CNNHT*, no se puede utilizar esta durante una segmentación.

Las conclusiones generales realizadas por los investigadores sugieren que la parametrización de la segmentación de la red neuronal supera en rendimiento a los algoritmos propuestos por Daugman en [43] en un gran numero de casos, pero en dos casos específicos no se muestra mejoría alguna. En casos de que el sistema biométrico realice capturas de imágenes en alta definición, el método propuesto en esta investigación no muestra mejoras, esto debido a la baja cantidad de oclusiones. Por otro lado, en capturas de imágenes de segmentación dificultosa (alto nivel de oclusiones), los métodos tradicionales (*WAHET* y *CAHT*) usualmente fallan al momento de segmentar el iris.

En general, la segmentación basada en *CNNs* y el método propuesto en esta investigación (*CNNHT*) denotan un perfeccionamiento de la técnica de segmentación ante métodos tradicionales excepto ante imágenes con un nivel muy reducido de oclusiones (o ninguno).

En [27] los autores proponen un algoritmo de 3 pasos: (1) detección de ojo, (2) detección de pupila y (3) estimador de limite de pupila. Para detectar la ubicación de un ojo en una imagen utilizaron *Faster R-CNN* [95]. Luego, la pupila y su limite fueron encontrados usando un modelo de mezcla gaussiana (*GMM* por sus siglas en ingles *Gaussian Mixture Model*), *MIGREP* [96] y así la región del iris es precisamente localizada.

[15] Errores donde la segmentación falla al generar una parametrización usable.

[16] Errores donde la superposición de dos mascaras de una imagen ingresada y la prueba presentada por el sistema no deja píxel para la comparación. El fallo de los algoritmos *CAHT* y *WAHET* al generar una parametrización es considerada un error de mascara.

Para la detección del ojo *VGG-16* [97] fue reemplazada por una arquitectura propuesta para esta investigación (ver figura 1 en [27]). Luego de generar los *ROIs* correspondientes a los ojos, *GMM* es aplicado (y no otro modelo de *Faster R-CNN* debido al tiempo computacional que esto significa) como detector de pupila. Finalmente, la detección del limite de la pupila es estimado con base al algoritmo *MIGREP* [96].
Luego, para los experimentos realizados fue utilizada la base de datos *CASIA-IRIS-THOUSAN*, la cual destaca por poseer una gran cantidad de iris tras lentes ópticos y reflejos ocasionados por los lentes y/o el ojo. Tanto *Faster R-CNN* como el modelo *GMM* fueron entrenados con el 60 % de la base de datos, mientras que el 40 % restante fue utilizado para pruebas. Teniendo en cuenta que el modelo final estaba destinado a ser implementado en un dispositivo móvil, la resolución de las imágenes fue reducida en gran medida para pasar a través del modelo. Para el entrenamiento del detector de objetos, fue definido un *IoU* de 0,8[17] para con esto realizar detecciones efectivas en una condición estricta. Para medir el rendimiento del detector de objetos fueron utilizadas las métricas de *Precision* y *Recall*.
Para la segmentación de iris fue utilizado el mismo modelo en conjunto (*Faster R-CNN*, *GMM* y *MIGREP*), obteniendo los siguientes resultados:

Tabla 2.9: Rendimiento de algoritmo segmentador de iris. Tabla obtenida de [27].

Limite de Pupila	96,77 %
Limite de Iris	98,32 %
Ambos Limites	95,49 %

En donde cada porcentaje representa el valor de rendimiento del segmentador.
En conclusión, en esta investigación se presento un segmentador de iris y pupilas robusto y rápido utilizando la arquitectura de *Faster R-CNN* [95], se extrajeron características con *GMM*[18]. Los experimentos mostraron la efectividad y eficiencia del método propuesto, el cual obtuvo un 95,49 % de rendimiento de segmentación, tomando un tiempo de computo de 0,06 segundos. La principal ventaja que anuncian los investigadores por sobre los modelos segmentadores de iris del estado del arte es que el modelo propuesto es más pequeño, lo cual permite una segmentación rápida lo cual es crucial al momento de implementar sistemas en tiempo real.

El trabajo más actual propuesto por Kerrigan et al., En [57] los autores exploran tres métodos de segmentación de iris basados en *deep learning* y una metodología para usar mascaras segmentadoras irregulares en un modelo de reconocimiento de iris convencional basado en *Gabor-wavelet*. Para entrenar y validar los métodos empleados, usaron un amplio espectro de imágenes de iris adquiridos por distintos equipos (incluyendo imágenes de *CASIA-Iris-Interval-v4*, *BioSec*, *ND-Iris-0405, UBIRIS*, *Warsaw-BioBase-Post-Mortem-Iris v2.0* (imágenes de iris post-mortem) y *ND-TWINS-2009-2010*), distintos sensores y un publico diverso. Esta amplia variedad de datos teóricamente debería incrementar las capacidades de generalización de las técnicas de segmentación propuestas (*SegNet* [98], *Deep Residual Network* [84] y *CNN* incorporando convoluciones dilatadas [99]).
Para propósitos de este estudio, cuatro distintas arquitecturas convolucionales diseñadas para segmentación semántica fueron seleccionadas para ser reentrenadas y evaluadas. Dos de ellas emplean convoluciones dilatadas [99]. Se refiere a estas redes como el modulo *front-end* y el modulo de *contexto*. El modulo *front-end* es una red *VGG-16* [97] *pyramid-like*. El cambio más significante fue haber eliminado las últimas capas de *pooling* y las capas *strides*. Más tarde, para los experimentos encontraron que 100.000 iteraciones fueron suficientes para obtener un modelo satisfactorio. Finalmente, la cuarta red es *SegNet* [98], la cual contiene

[17] Lo cual se considera estricto según [27].
[18] Gaussian Mixture Model

una red *encoder-decoder*.

Las bases de datos adquiridas fueron divididas en dos subsets, uno de entrenamiento y el otro de pruebas.

Los datos de entrenamiento provienen de 3 bases de datos: 2.639 imágenes de la base de datos *CASIA-Iris-Interval-v4*, 898 imágenes de la base de datos de *ND-Iris-0405* y 800 imágenes están mezcladas de las bases de datos *Warsaw-BioBase-Post-Mortem-Iris v2.0 6* y *ND-TWINS-2009-2010*.

Los datos de prueba provienen de 3 bases de datos: 385 imágenes de la base de datos ND-Iris-0405 (estas imágenes son *person-disjoint*[19] con las imágenes usadas en el set de entrenamiento), 1.200 imágenes de la base de datos *BioSec baseline corpus* y 2.250 imágenes de *UBIRIS.v2*. La base de datos de *UBIRIS.v2* es la única utilizada en su totalidad para pruebas debido a que fue construida solo con imágenes en espectro visual. Durante las evaluaciones, solo el canal rojo de la base de datos *UBIRIS.v2* fue utilizado.

Las estrategias de evaluación propuestas consisten en dos metodologías: (1) solo fueron evaluadas los rendimientos de las predicciones de los segmentadores realizados por sus modelos, (2) los resultados de segmentación fueron inyectados a un *pipeline* de reconocimiento de iris, cuyo rendimiento de reconocimiento es entonces evaluado en comparación con *OSIRIS* [39].

Para evaluar la precisión de la segmentación fue utilizada la métrica *IoU*[20]. Para cada base de datos de prueba y para cada modelo empleado para generar resultados de segmentación se reporta el valor ponderado de la métrica *IoU*, junto con su (σ) desviación estándar.

Adicionalmente se explica que se empleo el uso de la técnica *data augmentation*, con la cual cada imagen del set de entrenamiento fue aumentado 5 veces, obteniendo un total de 26.022 imágenes para entrenamiento (fue aplicado desenfoque gaussiano en un radio de 2, 3 y 4, mientras que para los restantes 2 fueron aplicadas mejoras de orillas).

Para la etapa de entrenamiento se utilizo el 80 % de las imágenes de entrenamiento para entrenar, mientras que el 20 % restante se aparto para validaciones. Se usaron imágenes con dimensiones de 640 $\times$ 480 para entrenar el modulo *front-end* y el modulo de contexto para para los modelos de convoluciones dilatadas, mientras que imágenes de 320 $\times$ 240 fueron utilizadas para entrenar *SegNet* y *DRN*[21]. Cada red fue inicializada con sus pesos pre-entrenados. Inicialmente estas redes fueron entrenadas con bases de datos non-iris tales como *ImageNet* [100], la cual contiene imágenes naturales de varios objetos. Además, se entreno el modulo *front-end* de contexto por 100.000 iteraciones (al rededor de 5 épocas según indican en [57]), con un *batch size* de 1 debido a poco espacio en memoria.

Luego, para las evaluaciones una predicción binaria fue obtenida de cada una de las 3 redes y además de *OSIRIS* [39] (algoritmo por defecto) por cada imagen en el set de evaluación, entonces la métrica *IoU* fue calculada para evaluar la coherencia entre lo predicho y las mascaras *ground truth*.

Finalmente, para los resultados de segmentación se promedia el *IoU* de cada imagen (para todas las imágenes), obteniendo así el peor rendimiento para *OSIRIS*, mientras que las 3 redes restantes obtuvieron resultados sobresalientes al sistema mencionado, tal como se muestra en la siguiente tabla:

Tabla 2.10: Mejores resultados (media y desviación estándar) de evaluación con distintas bases de datos. OSIRIS obtuvo el peor rendimiento en todos los datasets.

Database	*OSIRIS (%)*	*DRN (%)*	*Context-100k (%)*	*SegNet (%)*
BioSec	84.81 $\pm$ 06.73	**87.29 $\pm$ 05.79**	85.92 $\pm$ 05.91	85.93 $\pm$ 06.79
ND-Iris-0405	86.28 $\pm$ 06.5	89.61 $\pm$ 05.08	89.45 $\pm$ 03.85	**89.75 $\pm$ 04.95**
UBIRIS	42.69 $\pm$ 36.79	44.4 $\pm$ 29.75	**56.28 $\pm$ 26.93**	48.27 $\pm$ 26.42

[19] Person-disjoint indica que ninguna persona esta repetida en otro set (de entrenamiento o de pruebas)

[20] La cual denota la proporción de la intersección del *ground truth* y la predicción de la unión de los dos.

[21] Una *DRN* es una *ResNet* modificada, según [57].

Los autores de la investigación concluyen que fueron entrenados 3 tipos de redes convolucionales distintas en un amplio espectro de imágenes en espectro visual y en espectro *NIR* con éxito. Así como se muestra en la tabla 2.10, los modelos entrenados superaron al método convencional propuesto en *OSIRIS* [39]. Habiendo logrado esto, las implementaciones y los pesos de los modelos empleados en esta investigación quedan a libre disposición para ser descargados e implementados en donde se estime conveniente.

A continuación se presenta una tabla comparativa resumiendo los trabajos mencionados con anterioridad:

Tabla 2.11: Resumen del estado del arte (1/2).

Author	Method	Database Used	Metrics Used	Metrics Obtained (%)
Bezerra et al. [21]	Fully Convolutional Network (FCN) Generative Adversarial Network (GAN)	BioSec CasiaI3 CasiaT4 IITD-1 NICE.I CrEye-Iris MICHE	Average Segmentation Error (E) F-Measure (F1)	FCN (best): - F1 (%): 97,46 $\pm$ 00,74 - E (%): 0,44 $\pm$ 0,12 GAN (best): - F1 (%): 96,82 $\pm$ 2,83 - E (%): 0,74 $\pm$ 1,40
Arsalan et al. [22]	Densely Connected Fully Convolutional Networks	NICE-II (from UBIRIS.v2) MICHE-I CASIA v4.0 interval CASIA v4.0 distance IITD Delhi v1.0 iris dataset	Overall Average Segmentation Error (Ea), RPF-Measure	dataset / Ea NICE-II / 0,00695 MICHE-I / 0,0020 CASIA v4.0 D / 0,0034 RPF-Measure: - CASIA v4.0 interval R: (μ: 97,10) / (σ: 2,12) P: (μ: 98,10) / (σ: 1,07) F: (μ: 97,58) / (σ: 0,99) - IITD R: (μ: 98,00) / (σ: 1,56) P: (μ: 97,16) / (σ: 1,40) F: (μ: 97,56) / (σ: 0,84)
Hofbauer et al. [24]	RefineNet Fully Convolutional Encoder–Decoder Network (iFCEDN)	IIT Delhi Iris Database Version 1.0 (iitd) CASIA Iris Image Database Version 4.0 (casia4i) CASIA Iris Subject Ageing Version 1.0 Database (casiaA) PROTECT Multimodal DB (protMI)	NICE.I Evaluation: type 1 and type 2 errors (E1 & E2) F1-measure (F1)	RefineNet (Testset / E1 / E2 / F1): - casia4i / 0,009 / 0,011 / 0,984 iFCEDN (Testset / E1 / E2 / F1): - casiaA / 0,007 / 0,020 / 0,966
Hofbauer et al. [26]	RefineNet	UBIRIS IIT Delhi Iris Database version 1.0 (iitd) CASIA Iris Image Database version 4.0 (casia4i) ND-0405 Iris Image Dataset (ndi) CASIA Iris Subject Ageing Version 1.0 Database (casiaA) PROTECT Multimodal DATABASE	Equal Error Rate (EER) False Non Match Rate (FNMR) at a False Match Rate (FMR) of 0,01 as $OP_{0,01}$ Segmentation Errors (SE) Masking Errors (ME)	Testset / EER / $OP_{0,01}$ / SE / ME Casia4i / 0,320 / 0,619 / 0 / 0
Li et al. [27]	Faster R-CNN Gaussian Mixture Model (GMM)	CASIA-Iris-Thousand Database	IoU (threshold = 0,8), Accuracy of segmentation (q value)	q: 95,49 %
Kerrigan et al. [57]	Deep Residual Network (DRN) CNN incorporating dilated Convolutions (Context-100k) SegNet	CASIA-Iris-Interval-v4 ND-Iris-0405 Warsaw-BioBase-PostMortem-Iris v2.0 ND-TWINS-2009-2010 BioSec baseline corpus database UBIRIS v2	Intersection Over Union (IoU)	IoU (best): - BioSec: DRN (87,29 $\pm$ 5,79) - ND-Iris-0450: SegNet (89,75 $\pm$ 4,95) - UBIRIS: Context-100k (56,28 $\pm$ 26,93)

Tabla 2.12: Resumen del estado del arte (2/2). Información adicional.
Sea **NoIU**: *Number of Images Used*.

Author	Wavelength	NoIU	Data Augmentation	Split Train - Test
Bezerra et al. [21]	NIR RGB	9.244	No	80/20 %
Arsalan et al. [22]	NIR RGB	NICE-II: 1.000 MICHE-I: 1.552 CASIA v4.0: 2.567 IITD: 2.240	Yes (Training Data Augmented 12 times)	NICE-II: 500 (500) MICHE-I: 786 (766) CASIA v4.0: 2.167 (400) IITD: Not Specified
Hofbauer et al. [24]	NIR	protMI: 1.008 Remainder: Not specified	No	five-fold cross validation
Hofbauer et al. [26]	NIR RGB	Not specified	No	five-fold cross validation
Li et al. [27]	Not Specified	20.000 (1.000 subjects)	No	train: 12.000 test: 8.000
Kerrigan et al. [57]	train: NIR test: Red channel from RGB	train: 26.022 (4.337) test: 3.835	Five-fold	80/20 %

CAPÍTULO 3

Marco Teórico

1. Espacios de Colores

Los espacios de colores son modelos abstractos que describen la forma en que los colores pueden ser representados de manera matemática, usualmente en tupla de tres componentes primarios, donde los colores individuales pueden ser formados por los valores correspondientes de estos. De esta manera, al imitar los mecanismos de percepción del color del ojo humano, los dispositivos digitales gráficos, de visualización y de procesamiento de imágenes (como monitores o cámaras digitales) son capaces de reproducir las longitudes de onda pertenecientes al espectro visible [101].
Los tres componentes primarios que pueden ser mezclados para producir la mayor cantidad de colores son longitudes de onda particulares de rojo (R), verde (G) y azul (B). Por este motivo, la mayoría de los dispositivos de visualización a color se basan en tres fuentes de luz lo más cercanamente posible a estos colores [102]. Cabe destacar que para muchas aplicaciones es conveniente considerar un espacio de color formado por estos componentes, pero otros espacios de colores más adecuados pueden servir para facilitar ciertos cálculos tratamientos en el procesamiento de imágenes [102].

1.1. Espectro Visible

El espectro visible (o luz visible) corresponde al fragmento del espectro electromagnético que es visible por el sistema visual humano y es responsable del sentido de la vista ya que el color de los objetos depende de aquellas longitudes de onda que absorbe y aquellas que refleja, lo cual depende de la composición del material del objeto en cuestión. El rango de la radiación electromagnética que pertenece al espectro visible se encuentra entre las longitudes de onda que van desde los 380 ηm a los 750 ηm[1] aproximadamente, ubicándose entre la luz ultravioleta y la luz infrarroja [103].
Para fines explicativos, la luz no está compuesta por una sola longitud de onda, sino que es una mezcla continua de una parte o de todas las longitudes de onda pertenecientes al espectro visible. El color percibido es entonces una función de la distribución espectral de la amplitud por la longitud de onda. La forma de la distribución determina lo que se percibe como la cromaticidad (color, sin tener en cuenta la intensidad) de la luz. Para una cromaticidad determinada, el nivel de potencia absoluta (o la altura de la distribución) determina lo que se percibe como la intensidad.

En la figura 3.1 se puede apreciar una representación lineal del espectro visible donde los colores están

[1] Nanómetros.

representados por sus iniciales en ingles, siendo V[2] para el color violeta, B[3] el azul, G[4] el verde, Y[5] el amarillo, O[6] el naranja y R[7] el rojo, es decir, la longitud de onda determina el tono percibido de la luz.
Las esquinas de color negro representan longitudes de ondas no visibles por el ojo humano.

Figura 3.1: Representación del espectro visible [4].

1.2. Espectro NIR

El espectro *NIR* (por sus siglas en ingles *Near-Infrared*) forma parte del espectro electromagnético comprendiendo un rango de longitud de onda desde los 750 ηm hasta los 2.500 ηm aproximadamente [104] tal como se muestra en la figura 3.2:

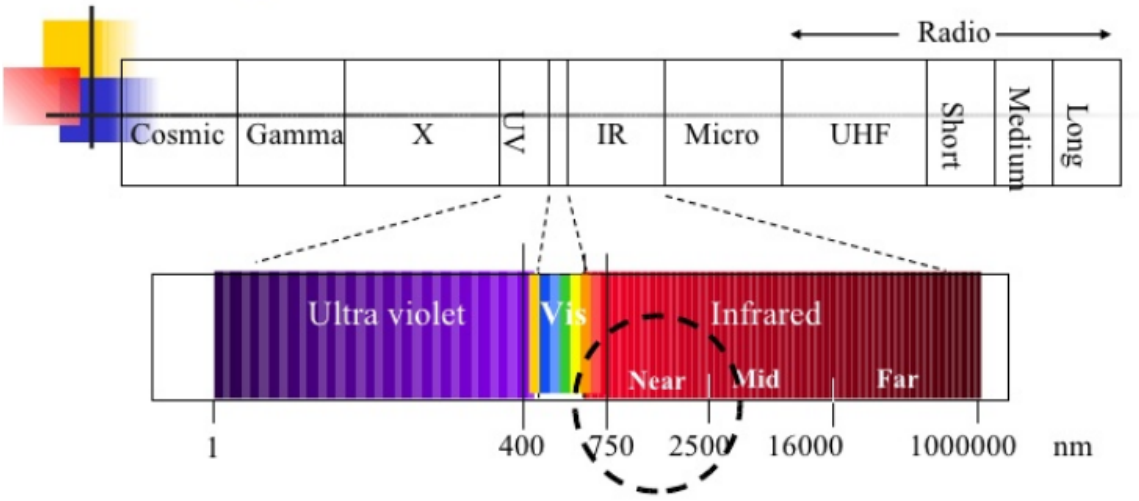


Figura 3.2: Representación gráfica del espectro *NIR*.

Esto sugiere que el espectro *NIR* se escapa del espectro visual debido a su longitud de onda, lo que en otras palabras significa que es imperceptible por el ojo humano.
Este tipo de luz infrarroja es ampliamente utilizado industrial, comercial y académicamente[8] y, para efectos de esta tesis, para el estudio de la segmentación del iris así como fue presentado en el estado del arte (ver capítulo 2).
Una diferencia destacable entre la captura de imagen en espectro visual y la captura en espectro *NIR* es que

[2] *Violet* para violeta
[3] *Blue* para azul.
[4] *Green* para verde.
[5] *Yellow* para amarillo.
[6] *Orange* para naranja.
[7] *Red* para rojo.
[8] En las ultimas décadas se ha presenciado un incremento en el uso de este espectro de luz debido a los avances y abaratamientos tecnológicos [104].

este último no consigue obtener los detalles del iris para colores claros (como lo son el verde y el azul), mientras que para colores oscuros (e.g. Café) consigue obtener una mejor calidad de detalles del iris que el espectro visual [61]. Además, cabe destacar que las capturas de imágenes en espectro visual suelen registrar los reflejos de la luz ambiental en la estructura ocular mientras que las capturas en espectro *NIR* carecen de este fenómeno debido a la longitud de onda en la cual se trabaja [61].

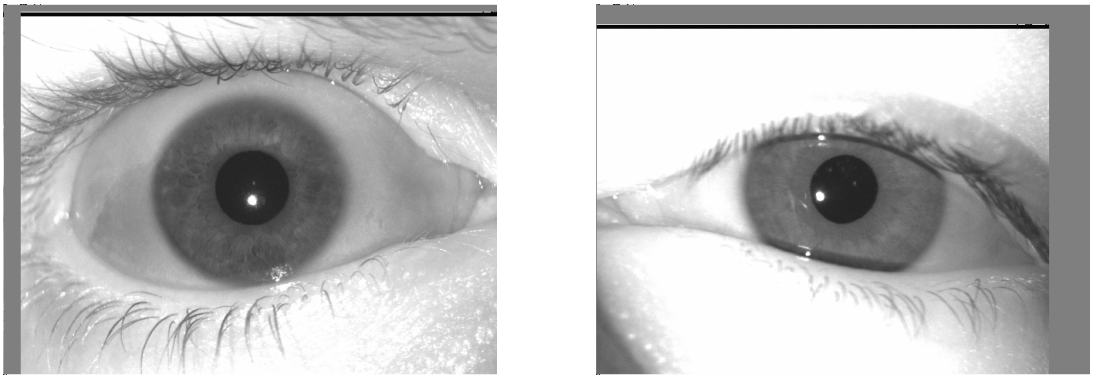


Figura 3.3: Ejemplos de imágenes capturadas bajo espectro NIR. Imágenes obtenidas de [1].

2. Inteligencia Artificial (IA)

La inteligencia artificial (de aquí en adelante *IA*) es la inteligencia llevada a cabo por maquinas, por ello se define como una maquina inteligente ideal a la que es un agente que percibe su entorno y es capaz de tomar decisiones de tal manera que le garantice el éxito en la realización de sus tareas y/u objetivos.
Se define *IA* como "*(...) una forma de hacer que las maquinas piensen y se comporten de manera inteligente*" [105]. Para fines descriptivos, la *IA* es una ciencia en donde se buscan teorías y metodologías que ayuden a las maquinas a comprender el mundo y actuar de acuerdo a las situaciones de la misma manera que los humanos lo hacen. Para ello, este campo se relaciona con el estudio del funcionamiento del cerebro humano[9] ya que simulando su actividad y estructura (redes neuronales) se puede realizar una mímica del pensamiento, la toma de decisiones y la habilidad de aprender a realizar exitosamente una tarea en particular. Por consiguiente, la inteligencia artificial es ampliamente usada hoy[10] para distintos tipos de actividades surgiendo así distintas sub-especialidades tal como se muestra en la figura 3.4:

[9] En primeras instancias el estudio fue centrado en el cerebro animal.
[10] Esto debido a la velocidad y facilidad con la que se genera información nueva, la cual necesita ser clasificada de manera rápida y eficaz.

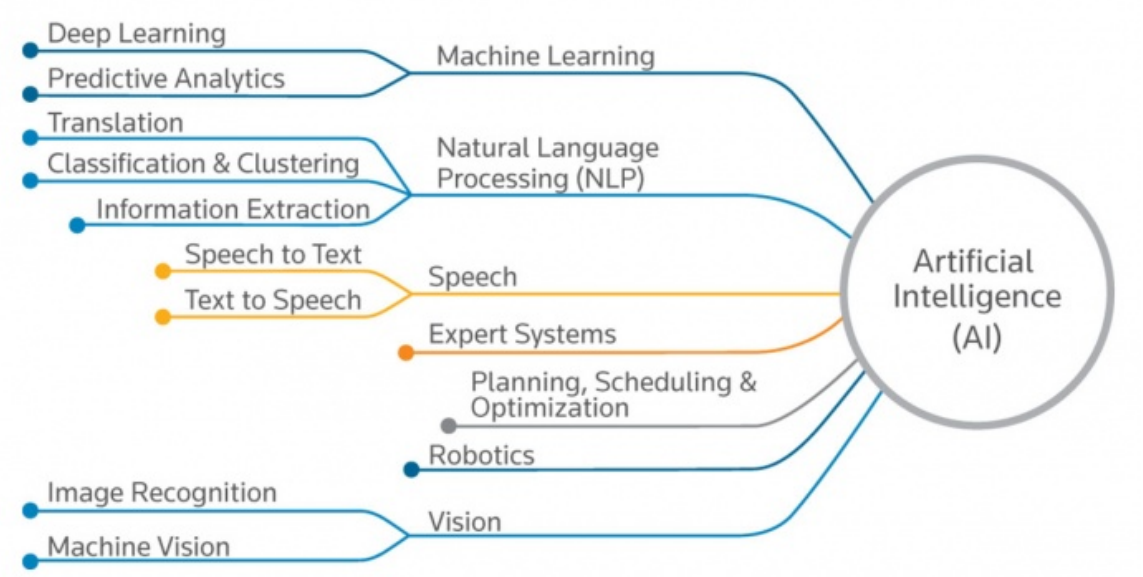


Figura 3.4: Sub-especialidades del campo de inteligencia artificial [5].

2.1. Machine Learning

Machine Learning (o aprendizaje automático, de aquí en adelante *ML*) es una rama de la inteligencia artificial (ver figura 3.4) que estudia la construcción de algoritmos que puedan aprender características relevantes de un conjunto de datos[11] para luego tomar decisiones, realizar predicciones o identificar patrones[12] sobre nuevos conjuntos de datos sin haber sido explícitamente programados para ello [106, 107]. Esto se logra a través de la creación de un modelo analítico[13], el cual es realizado de manera automática durante la fase de entrenamiento a través del análisis efectuado sobre un *dataset* que contiene datos de entrenamiento, a partir de los cuales el algoritmo puede realizar una generalización y modelar una función de pérdida que exprese la disimilitud entre las predicciones del modelo entrenado y las instancias reales de muestras no vistas con anterioridad[14]. Debido a esto, es importante contar con un *dataset* representativo, con el fin de minimizar la función de pérdida en datos no vistos [108].

Dentro de lo que se comprende por *ML* podemos ubicar a los dos tipos de aprendizaje más utilizados y más estudiados[15], los cuales corresponden a [105]:

Aprendizaje Supervisado: Se refiere al proceso de construir un modelo de *Machine Learning* **basado en datos catalogados**. Por ejemplo, se busca construir un sistema que clasifique los distintos tipos de vinos. Para ello es necesario crear una base de datos de vinos con todos los detalles y características necesarias y catalogarlos según corresponda (vino tinto, blanco o rosado). En el proceso de entrenamiento esto indicará qué parámetros corresponden a cada etiqueta, luego, el algoritmo generalizará a partir de lo aprendido y distinguirá entre cada vino. Algunos ejemplos de algoritmos de aprendizaje supervisado son:

- Regresión Lineal

[11] Estos conjuntos de datos pueden estar catalogados o no catalogados dependiendo de la necesidad y del tipo de algoritmo que será empleado.

[12] Serie de variables constantes, identificables dentro de un conjunto mayor de datos.

[13] Los modelos analíticos son derivados de formulaciones matemáticas, lo cual es básicamente una secuencia de pasos definidos para llegar a una ecuación final [105].

[14] Este segmento de datos no vistos es considerado como set de pruebas.

[15] Existen 4: Supervisado, No Supervisado, *Self-Supervised* y Aprendizaje Reforzado [7].

- Regresión Logística
- Clasificación Bayesiana
- Maquinas de Soporte Vectorial (*SVM*)
- Análisis Discriminante Lineal
- Árboles de Decisión
- Redes Neuronales

Aprendizaje No Supervisado: Se refiere al proceso de construir un modelo de *Machine Learning* que **no dependa de que los datos estén etiquetados**. Estos tipos de algoritmos son lo contrario a los tipos de algoritmos discutidos anteriormente puesto que solo son necesarios los atributos y características que definen a los datos. Por ejemplo, se busca construir un sistema donde se requiera separar los datos en múltiples grupos. Lo complicado del trabajo solicitado es que el criterio de separación no es conocido, por lo tanto un algoritmo de aprendizaje no supervisado debe separar el conjunto de datos en varios grupos de la mejor manera posible. Algunos ejemplos de algoritmos de aprendizaje no supervisado son:

- Agrupamiento Jerárquico
- *K-Means*
- *DBSCAN*
- *Local Outlier Factor*

2.1.1. Entrenamiento

La gran mayoría de los algoritmos pertenecientes al campo de *Machine Learning* son entrenados de manera similar. En mencionado proceso es utilizada una base de datos, la cual es sub-dividida en 3 sets distintos [7, 105, 107]:

Set de Entrenamiento Este set (también llamado *train set*) comprende casi la totalidad de los datos (frecuentemente el 80 %). Su existencia es indispensable debido a que es empleado para ajustar los parámetros internos del algoritmo a entrenar.

Set de Pruebas Este set (también llamado *test set*) comprende una parte reducida de la totalidad de los datos (frecuentemente el 20 % restante). Este set de datos es empleado para comprobar **que tan correcto** responde el algoritmo respecto a un dato antes no visto mientras es entrenado, de esta manera es otorgada una métrica de confiabilidad. Cabe destacar que los datos dentro de este conjunto de datos no deben estar repetidos dentro del set de entrenamiento. A esto usualmente se le denomina como *disjoin-set*.

Set de Validación Este set (también llamado *validation set*) comparte ciertas características con el set de pruebas ya que también corresponde a una porción de datos no incluida ni repetida en el set de entrenamiento (hasta que se necesite), pero su función radica en ser empleado para ajustar los hiper-parámetros[16] de los algoritmos.

[16] Parámetros que son seteados antes del proceso de entrenamiento.

Una división ideal es en donde ningún dato se encuentra repetido en ningún set. Esto se puede lograr dividiendo la base de datos de manera manual o automáticamente según lo requerido.
Para el caso de una división automática, existen algunas técnicas tales como:

Train/Test Split La base de datos es sub-dividida en dos únicos sets. El set de entrenamiento consiste en la mayoría de los datos (se sugiere el 80 %) mientras que el remanente es considerado como el set de pruebas.

Cross Validation La base de datos es sub-dividida en sets de entrenamiento y prueba. Luego, para el set de evaluación es considerado el 20 % del actual set de entrenamiento.

K-folds Cross Validation La base de datos es dividida en *K* partes iguales. Luego, para el entrenamiento es usado *K-1* partes mientras que el 1 restante es considerado como el set de pruebas.

Estos tipos de divisiones son ideales para evitar problemas post entrenamiento tales como el ***overfitting*** y el ***underfitting***. Se habla de *underfitting* cuando el modelo generado por el algoritmo de aprendizaje no es capaz de obtener un valor de error reducido para el set de entrenamiento, en otras palabras, el modelo "no fue capaz de aprender lo suficiente". *Overfitting* ocurre cuando la brecha entre los valores de error de entrenamiento y los valores de error de prueba son excesivos.
Otra manera de solventar estos tipos de errores son implementando algoritmos mas robustos y aumentando o reduciendo sus capacidades (arquitecturalmente hablando) [7, 105, 107].

2.1.2. Data Augmentation

Para el correcto entrenamiento de un modelo de *Machine Learning* es necesaria una cantidad importante de datos para que estos algoritmos puedan generalizar bien frente a datos no antes vistos. Para solventar en parte este problema, existe una técnica para incrementar la cuantía de datos del set de entrenamiento llamada ***Data Augmentation***, la cual permite incrementar artificialmente el tamaño del set de entrenamiento generando variantes realistas de cada instancia dentro del set de datos. Esto en parte reduce el *overfitting*, haciendo de esto una técnica regularizadora. Las instancias generadas deben ser lo mas realistas posible, idealmente lo suficiente como para que un humano no pueda reconocer si un set de datos fue aumentado o no [109]. Algunos ejemplos de *Data Augmentation* radican en rotar la imagen cierta cantidad de grados, opacarla, aumentar su brillo, entre otros efectos visuales. Un ejemplo de *Data Augmentation* es mostrado en la figura 3.5:

Figura 3.5: En la presente figura, el *Data Augmentation* es realizado rotando la imagen y siendo redimensionada, generando 2 imágenes extra de la original.

2.1.3. Redes Neuronales Artificiales

Dentro de lo que se comprende por *Machine Learning* podemos encontrar un tipo de algoritmo llamado redes neuronales artificiales (*ANN*, por sus siglas en inglés *Artificial Neural Network*), las cuales corresponden

a un modelo designado para simular el proceso de aprendizaje humano [105]. La mayoría de las arquitecturas de redes neuronales son del tipo de *Aprendizaje Supervisado*, lo cual indica que aprenden de datos previamente etiquetados por humanos.
Para fines explicativos, la neurona corresponde a la unidad básica de una red neuronal. Mencionada unidad comprende las entradas de las variables (x_i), los pesos designados para cada input de la neurona (w_i), el *bias* (b - constante otorgada para ajustar el modelo a los datos), la suma ponderada de la multiplicación de todos los pesos y todos los *inputs* ($\sum_i w_i x_i + b$), la función de activación (f) y el resultado de salida (*output*) tal como se muestra en la figura 3.7.

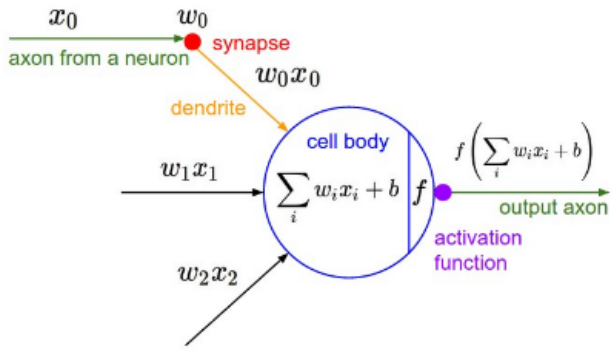


Figura 3.6: Unidad básica de una red neuronal. Imagen obtenida de [6].

El conjunto jerárquico de neuronas conforman una red neuronal, en otras palabras, cada conjunto de neuronas (llamado capa) es precedido por otro conjunto de neuronas. Junto a lo antes dicho, se destaca que existen 3 tipos de capas, siendo estas la capa de entrada *(input layer)*, las capas ocultas (*hidden layers*) y la capa de salida (*output layer*). Cabe señalar que una red neuronal simple solo dispone como máximo dos capas ocultas y, además, todas sus neuronas están completamente conectadas tal como se muestra en la figura 3.7.

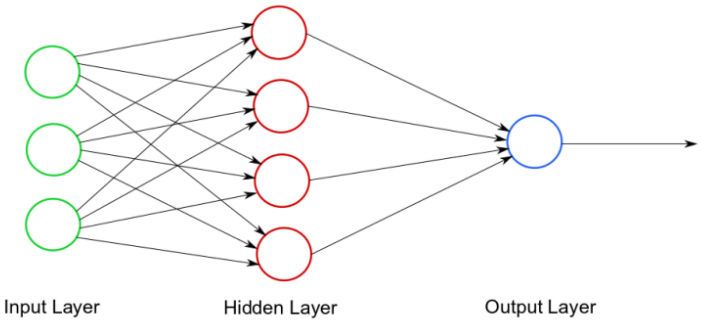


Figura 3.7: Red neuronal artificial.

Este tipo de algoritmo es utilizado para regresiones (lineales y no lineales), clasificaciones, procesamiento de imágenes, reconocimiento de caracteres, entre otras aplicaciones.

2.2. Deep Learning

Deep Learning (o aprendizaje profundo, de aquí en adelante *DL*) son técnicas que pertenecen a la familia de metodologías de *Machine Learning* y consisten en cascadas de múltiples capas de procesamiento no

lineales, las cuales extraen características y las transforman para análisis de patrones o clasificación.
Los algoritmos de *Deep Learning* aprenden múltiples niveles de representación (correspondientes a niveles de abstracción) para modelar relaciones complejas entre los datos, donde cada nivel usa como entrada el resultado de la capa anterior, creando efectivamente una cascada jerárquica [7, 105, 107, 110]. Un ejemplo cercano a lo antes dicho se puede apreciar en la figura 3.8.

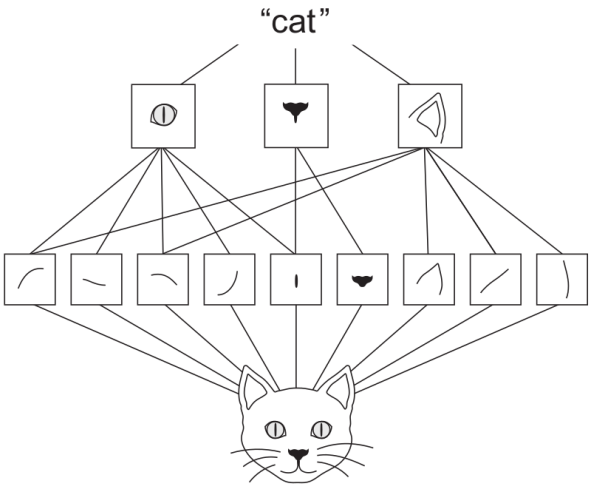


Figura 3.8: Representación gráfica de niveles de abstracción donde la relación entre cada nivel converge a la etiqueta “gato”. Imagen obtenida de [7].

La mayoría de los modelos de *Deep Learning* consisten de redes neuronales artificiales (Ver sección 2.1.3), en donde cada capa de abstracción representa un conjunto de nodos que, en términos generales, simbolizan las neuronas de un cerebro animal [111].

2.2.1. Convolutional Neural Networks

Las redes convolucionales neuronales (por sus siglas en inglés, *Convolutional Neural Networks - CNN*) son arquitecturas de *Deep Learning*, generalmente aplicadas para el análisis visual de imágenes[17]. Son entrenadas con miles o incluso millones de imágenes con el fin de que el modelo resultante sea capaz de generalizar la información y realizar predicciones sobre datos no antes vistos.
La arquitectura de una *CNN* posee algunas diferencias respecto a otras redes neuronales artificiales ya que sus capas son tridimensionales y las neuronas en cada capa solo se conectan con una región de la capa siguiente, mientras que el resultado de la capa final es un vector unidimensional que contiene las tasas de predicción. Las tres dimensiones mencionadas anteriormente corresponden al alto y ancho de las imágenes de input más la profundidad, que corresponde a los canales o componentes de su espacio de color (Ver sección 1), el output es también tridimensional. La figura 3.9 ejemplifica la arquitectura de una *CNN* [105].

[17] Formalmente llamado visión computacional.

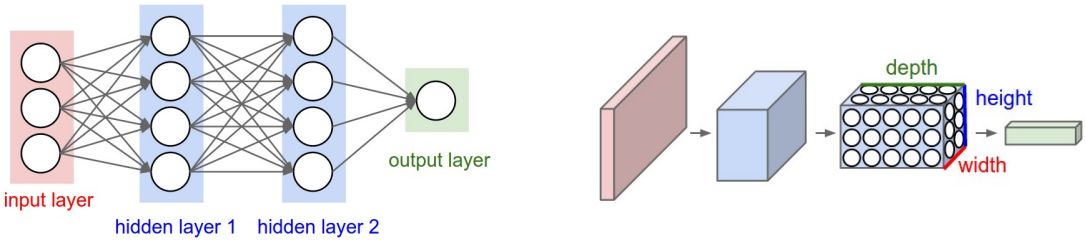


Figura 3.9: Ejemplo de *CNN*. En la imagen de la derecha se aprecian las neuronas tridimensionales [6].

Para fines explicativos, la **convolución** es un efecto de filtro de propósito general para imágenes. Formalmente expresado, es una matriz (llamada kernel o filtro) aplicada a una imagen y una operación matemática compuesta de números. Esta operación (la convolución) consiste en determinar el valor de un píxel central sumando los valores ponderados de todos sus vecinos.
El kernel (o filtro) usualmente es una matriz pequeña compuesta de números (representando un patrón), los cuales son empleados para realizar la convolución por sobre la imagen. Distintos kernels de distintos tamaños con distintos valores producen distintos resultados al ser aplicada la convolución [7, 8, 105, 107].
El proceso de convolución será detallado paso a paso aplicando el kernel mostrado en la figura 3.10:

0	-1	0
-1	5	-1
0	-1	0

Figura 3.10: Kernel simétrico de 3×3. Imagen obtenida de [8].

El proceso de convolución consiste en voltear el kernel horizontal y verticalmente para luego disponer el primer elemento de esta matriz por sobre el primer píxel de la imagen tal como se muestra en la figura 3.11:

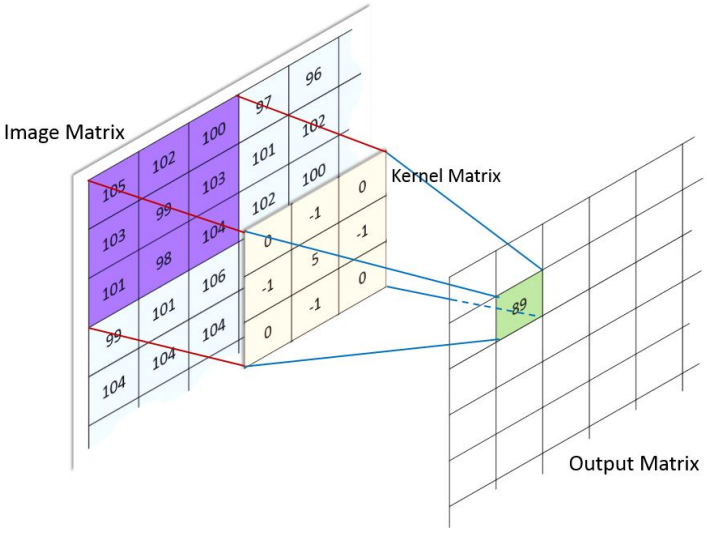


Figura 3.11: Proceso de convolución (1/2). Dado que el kernel es simétrico, al momento de voltearlo queda igual al kernel original. Imagen obtenida de [8].

A continuación, los valores del kernel volteado son multiplicados con cada valor de píxel correspondiente de la imagen. Finalmente, todos los productos son sumados y el resultado es puesto en la ubicación central del kernel pero en una nueva matriz tal como se muestra en la figura 3.12:

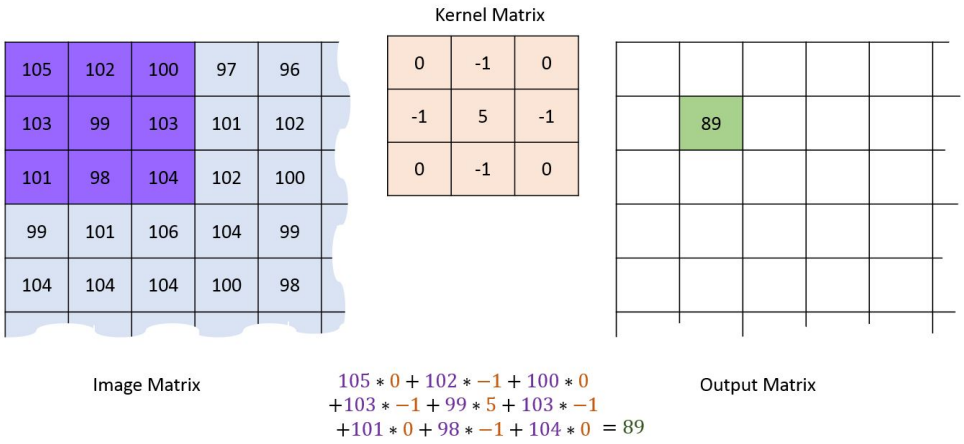


Figura 3.12: Proceso de convolución (2/2). Imagen obtenida de [8].

Este proceso se repite avanzando píxel por píxel hasta llegar al último de la imagen. En algunas situaciones se requiere avanzar cada cierta cantidad de píxeles para agilizar el proceso de convolución, por lo cual el hiper-parámetro **strides** es seteado, por ejemplo, un $stride = 2$ indica que el kernel será movido cada 2 píxeles.

Al momento de ser aplicada la operación de convolución se anteponen dos situaciones: (1) **el primer valor** del kernel corresponderá con el primer píxel de la imagen (tal como es mostrado en las figuras 3.11 y 3.12) o (2) **el centro** del kernel corresponderá con la ubicación del primer píxel de la imagen, quedando valores del kernel fuera de los limites de la imagen. Para el caso (1) la nueva matriz resultante será mas pequeña que la imagen original resultando en perdida de información, mientras que para el caso (2) la nueva matriz corresponderá al tamaño de la imagen original. Para resolver el problema de los valores del kernel fuera de los limites de la imagen es aplicado un **padding**[18], lo cual consiste en rellenar el borde de la imagen con valores correspondientes al negro[19], al blanco[20] o a valores del píxel mas cercano al borde para que así el kernel vuelva a estar dentro de los límites de la imagen y se pueda realizar la convolución sin reducir las dimensiones de la matriz de salida. Esta acción se puede apreciar en la figura 3.13 donde es aplicado un padding de ceros:

[18] Relleno en español.
[19] Valor de píxel: 0
[20] Valor de píxel: 255

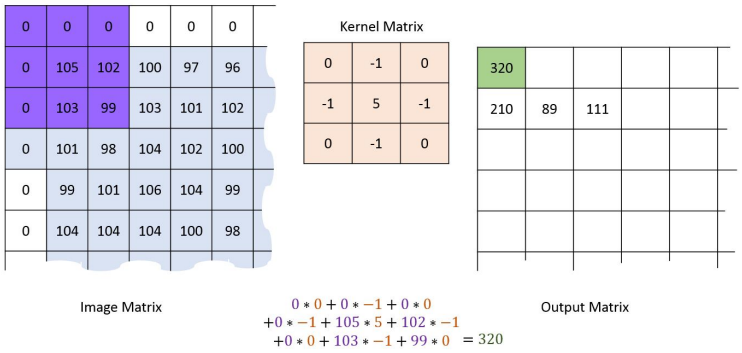


Figura 3.13: Padding de ceros. Imagen obtenida de [8].

Finalmente, la matriz de salida corresponderá a una nueva imagen y, dependiendo del kernel, esta imagen dispondrá de un efecto tal como difuminación, refinado, detección de bordes, detección de esquinas, entre otros.
Luego, el diseño de una *CNN* consiste en el uso de distintas capas tales como [7, 105]:

Input Layer Capa que recibe la imagen de entrada. Esta imagen es representada por los valores numéricos de sus canales de colores (Ver sección 1).

Convolutional Layer Capa compuesta por filtros convolucionales aprendibles. Esta capa es la encargada de realizar la operación de convolución por sobre la imagen de entrada con cada filtro disponible, por lo cual esta operación representa un gran costo computacional dependiendo de la profundidad de la red, la cantidad de filtros por aprender y las dimensiones de las imágenes a trabajar, es decir, entre mayor sea la profundidad de la red, entre más filtros y más grandes las imágenes, más tiempo será empleado en el proceso de entrenamiento y ejecución de la red neuronal.

Activation Layer Capa que aplica una función de activación de salida de cada neurona de una capa en específico. La funcionalidad principal de la función de activación es agregar no linealidad a la red y además permitir la activación de las neuronas, lo cual se puede interpretar como "que tan relevante es la información que la neurona esta recibiendo". Dependiendo de su relevancia, la información será transferida a la siguiente capa de neuronas.

Pooling Layer Capa encargada de reducir las dimensiones de la imagen de entrada conforme se avanza en la red. Esto ayuda a mantener solo las partes (características) prominentes de la imagen. El tipo de capa *MaxPooling* es utilizado con frecuencia, en la cual es elegido el valor máximo en una sección de $N \times N$ determinada dentro de la imagen.

Fully Connected Layer También llamada *Dense Layer*. Capa encargada de computar las puntuaciones de la capa antecedente a esta, resultando un vector de salida $1 \times 1 \times L$, donde L es el numero de clases[21].

Dropout Layer Capa encargada de desactivar de manera aleatoria distintas neuronas según se especifique en los hiper-parámetros[22] especificados, esto con la finalidad de reducir el *overfitting*.

Batch Normalization Capa encargada de normalizar las activaciones de la capa anterior aplicando una transformación que mantiene la activación media cercana a cero y la activación de la desviación

[21] Estas clases son determinadas en el *dataset* de entrenamiento. Un ejemplo de clases pueden ser los números del 0 al 9 de la *base de datos MNIST*. Esta base de datos se puede ubicar en `http://yann.lecun.com/exdb/mnist/`.
[22] Parámetros que son seteados antes del proceso de entrenamiento.

estándar cercana a 1. Corresponde además de un método regularizador para evitar problemas como el *overfitting* y el *underfitting*.

Custom Layer Existe también la posibilidad de crear una capa personalizada combinando las capas mencionadas con anterioridad o aplicando otro tipo de operaciones.

2.2.2. Mapas de Características

Los mapas de características son el resultado de haber aplicado una operación (sea esta una convolución, producto matricial, producto punto, suma, entre otras) en una imagen. En estos mapas se puede apreciar visualmente qué características son las más relevantes para una red neuronal, las cuales son destacadas tal como se muestra en la siguiente figura:

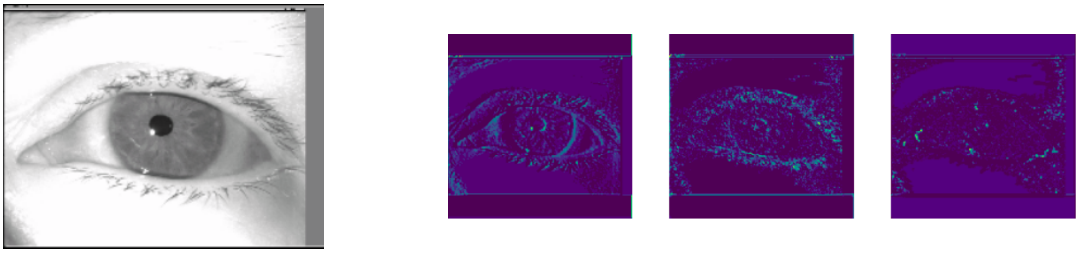


Figura 3.14: Ejemplo de mapas de características. A la izquierda la imagen normal y a la derecha 3 distintas imágenes representando cada una un mapa de características distinto. Se puede ver como parte del limite del iris y de la pupila son destacados en el primer mapa de características.

2.3. Visión Computacional

La visión computacional es un campo interdisciplinario relacionado con la inteligencia artificial que se dedica al estudio del desarrollo de modelos computacionales y sistemas inteligentes autónomos que puedan desempeñar tareas del sistema visual humano [112] a través de la extracción y análisis de características de información visual digital (ya sean imágenes o vídeos) con técnicas basadas en *Machine Learning* y *Deep Learning*.

2.4. Clasificación

La tarea de clasificación de imágenes corresponde en analizar, reconocer y categorizar de forma automática una imagen completa de acuerdo a alguna clase en particular. Una red neuronal retorna la etiqueta de la clase correspondiente y la probabilidad de que la imagen pertenezca a esa clase.

2.5. Detección de Objetos

En este tipo de tarea primero se detectan regiones de interés para luego ser clasificadas de forma individual de acuerdo a su clase. La detección de objetos permite separar cada clase por instancia del objeto. En la figura 3.15 se muestra un ejemplo de lo antes mencionado.

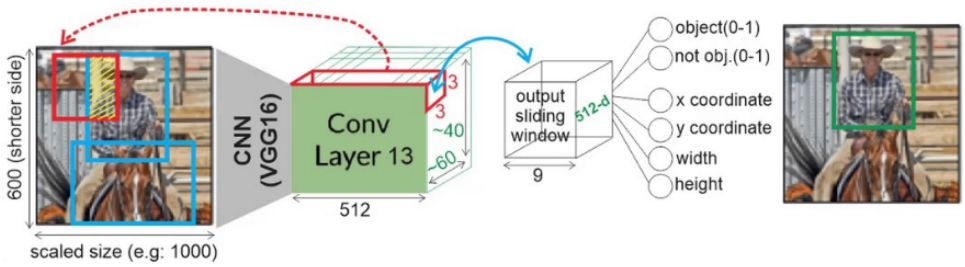


Figura 3.15: Detección de objetos con *RPN (Region Proposal Network)* [9].

2.6. Segmentación de Imágenes

La segmentación de imágenes es la tarea de dividir una imagen en distintas regiones, categorías o segmentos, los cuales corresponden a distintos objetos. Por consiguiente, cada píxel de la imagen es agrupado con aquellos que tienen atributos y características similares, siendo finalmente asignados a una categoría en especifico.

La tarea de segmentación es a menudo considerada como el paso crítico en el análisis de imágenes puesto que, si es correctamente realizado, todas las demás etapas del análisis de imágenes es simplificado.

Algunas aplicaciones de la segmentación de imágenes comprenden, por ejemplo, la detección y segmentación de células cancerígenas [113], *self-driving cars* [114], segmentación morfológica de imágenes satelitales [115], entre otros trabajos. Dado lo antes expuesto, existen distintos métodos de segmentación de imágenes y, para esta tesis, solo serán detalladas las siguientes:

2.6.1. Segmentación Semántica

A diferencia de la clasificación de imágenes en forma general, en la tarea de segmentación semántica se clasifica cada píxel de la imagen de acuerdo a su clase correspondiente, sin separar las distintas instancias de los objetos que pertenecen a una misma clase.

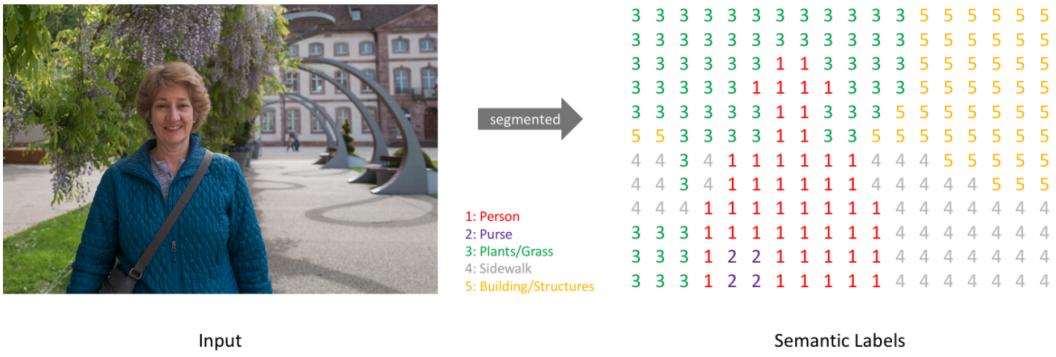


Figura 3.16: Ejemplo de segmentación semántica [10].

2.6.2. Segmentación de Instancia

Similar a la tarea de detección de objetos, en la tarea de segmentación de instancia se identifican y clasifican los objetos de la imagen por cada instancia de estos. Además, se clasifica cada píxel de cada instancia. Es efectivamente una combinación entre la detección de objetos y la segmentación semántica.

En la figura 3.17 se muestra un ejemplo de las cuatro tareas mencionadas anteriormente:

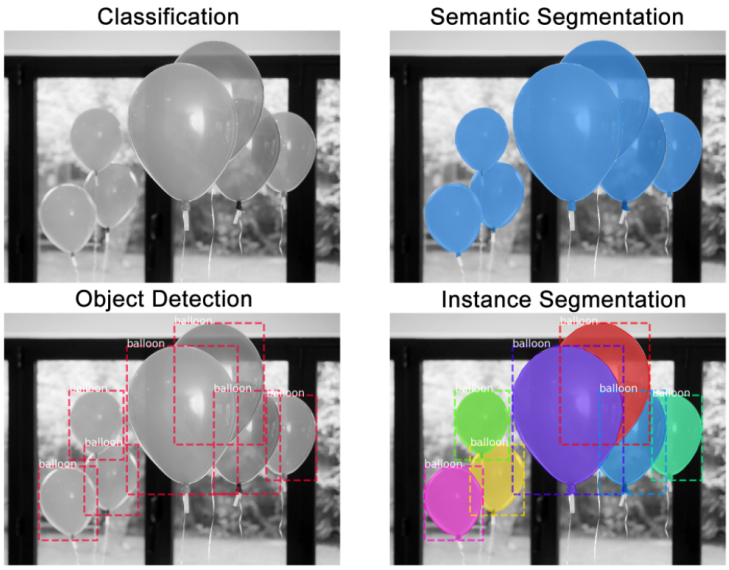


Figura 3.17: Diferencias entre distintas tareas de visión computacional [11].

2.7. Mask R-CNN

Mask R-CNN [12] es un *framework*[23] que extiende *Faster R-CNN* [95] para realizar segmentación de instancia agregando una rama paralela dedicada a la detección de objetos con el fin de predecir la máscara que segmenta cada instancia de objeto (a nivel de píxel) del resto de la imagen. Este *framework* es robusto, flexible y puede ser generalizado a otras tareas, además de demostrar un desempeño superior que sus predecesores. La rama en paralelo para predecir máscaras de segmentación corresponde a una *FCN* [116] de bajo costo aplicada a cada *RoI* (figura 3.18). Las máscaras generadas por *Mask R-CNN* son máscaras binarias, es decir, una matriz donde los elementos con el valor de 1 indican los píxeles pertenecientes al objeto y los elementos con el valor de 0 indican píxeles no pertenecientes al objeto.

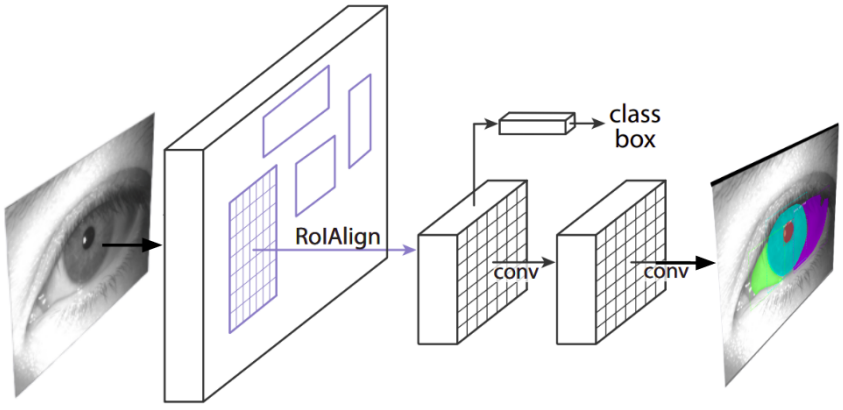


Figura 3.18: Modelo de arquitectura de *Mask-RCNN* [12].

[23] Marco de trabajo.

2.7.1. Region Proposal Networks

Redes basadas en proposición de regiones (o por sus siglas en inglés *Region Proposal Networks - RPN*) constan de dos pasos: (1) Examinar el área en forma general y luego (2) dar enfoque individual a cada una de las regiones de interés encontradas. Este proceso es similar a la actividad atencional del cerebro humano. Las regiones de interés son identificadas empleando una ventana deslizante que detecta objetos relevantes, los cuales son clasificados con posterioridad de acuerdo a su clase [9, 95, 117]. *Mask R-CNN* [12] hace uso de una *RPN* sobre los mapas de características extraídos por el *backbone*, al igual que su antecesor *Faster R-CNN* [95].

2.7.2. RoIAlign

Una mejora introducida al *framework Mask R-CNN* es *RoIAlign*, el cual reemplaza a *RoIPool* (proveniente de *Faster R-CNN* [95]). Con esta mejora es evitada la cuantificación de los mapas de características extraídos, aplicando en su lugar una interpolación bilineal, lo que mitiga la desalineación entre los píxeles de cada *RoI* y el mapa de características de salida. Este problema causado por *RoIPool* no impacta la clasificación de cada *RoI* debido a la robustez de *Fast R-CNN*, pero constituye un efecto negativo de gran importancia en la predicción de máscaras que compone la tarea de segmentación semántica introducida en *Mask R-CNN*, motivo por el cual los autores del *framework* proponen finalmente *RoIAlign*. En la figura 3.19 es mostrada la diferencia en un mapa de características entre *RoIPool* (parte izquierda) y *RoIAlign* (parte derecha) [13].

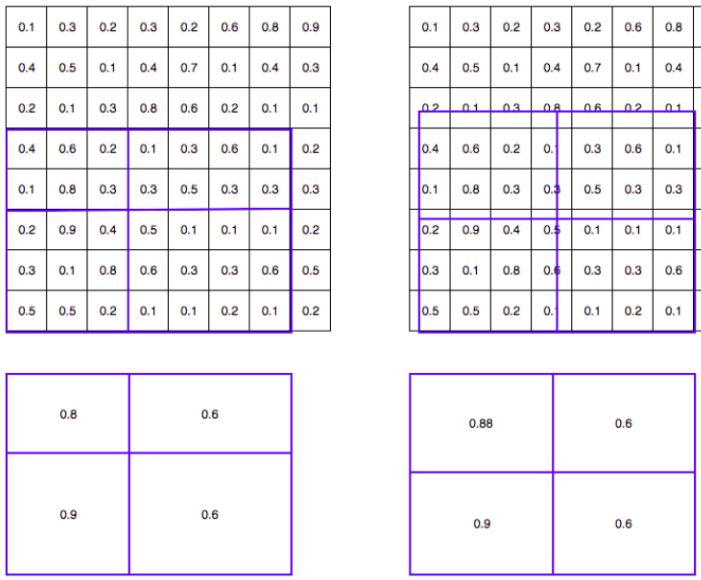


Figura 3.19: RoIPool vs RoIAlign [13].

2.7.3. Arquitectura

Mask R-CNN, al igual que sus predecesores emplea una *CNN* previamente entrenada como extractor de características conocido como *backbone*. En el caso de este *framework* los *backbones* evaluados son *ResNet* [84] y *ResNeXt* [118]. Además, se explora con buenos resultados el uso de una *Feature Pyramid Network (FPN)* [14], la cual mejora la calidad de las detecciones de objetos al utilizar una pirámide de múltiples mapas de características en varias escalas. La pirámide esta compuesta de una vía *bottom-up* en la cual se extraen los mapas de características, reduciendo la resolución y aumentando el valor semántico por cada capa, y

una vía *top-down* en la cual se construyen capas de mayor resolución a partir de la capa con mayor valor semántico. Además, se crean conexiones laterales entre las capas construidas y sus mapas de características correspondientes, con el fin de mejorar la precisión del detector respecto a la locación de los objetos en la imagen [15]. El uso de una *FPN* mejora de manera importante la detección de objetos de menor tamaño, con un impacto menor en el costo en tiempo de ejecución [14]. En la figura 3.20 se observa la arquitectura general de una *FPN*.

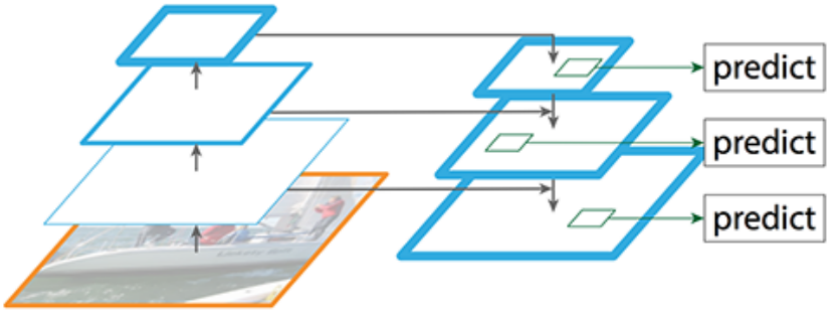


Figura 3.20: Arquitectura de *Feature Pyramid Network* [14]. Vía *bottom-up* a la izquierda y *top-down* a la derecha con conexiones laterales al centro.

Para el caso de *Mask R-CNN*, los *backbones* mencionados anteriormente se utilizan para construir la vía *bottom-up* de la *FPN* (figura 3.21):

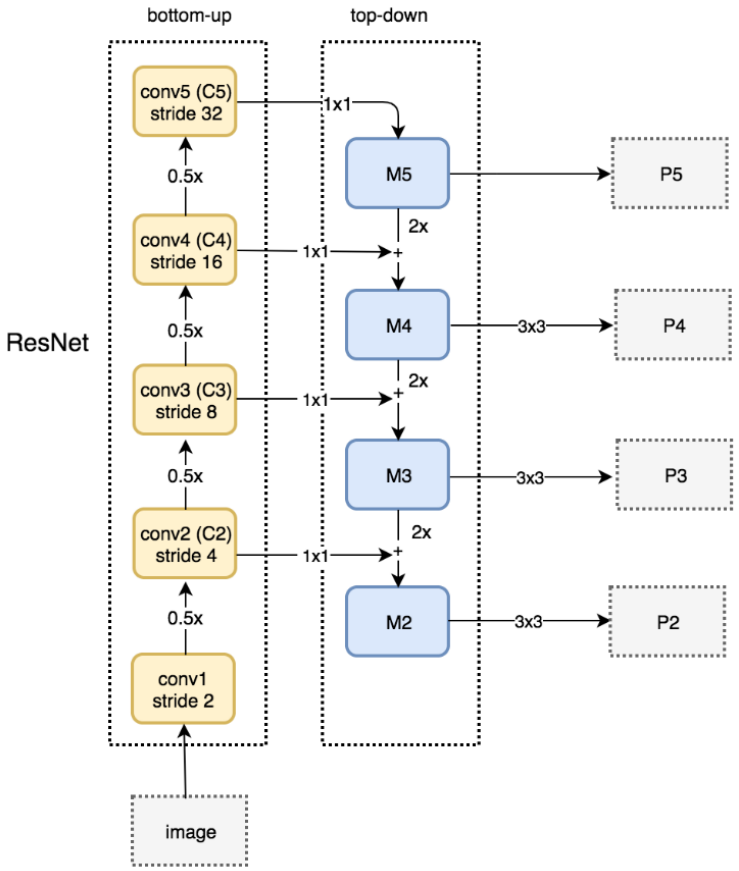


Figura 3.21: *FPN* con *ResNet* [15]

Las regiones propuestas por la *RPN* y los mapas de características pasan por el módulo de *RoIAlign*, cuyo output es luego es ingresado a las capas *Fully Connected* (FC) para efectos de clasificación de objetos (utilizando *softmax*) y predicción de *bounding boxes (bounding box regressor)*. Además, el output de *RoIAlign* sirve como input para la rama de predicción de máscaras, la cual esta constituida de dos capas convolucionales (Conv). La predicción de *bounding boxes* y clasificación de objetos, y la predicción de máscaras ocurren en paralelo [85]. En la figura 3.22 se puede observar la arquitectura de *Mask R-CNN*.

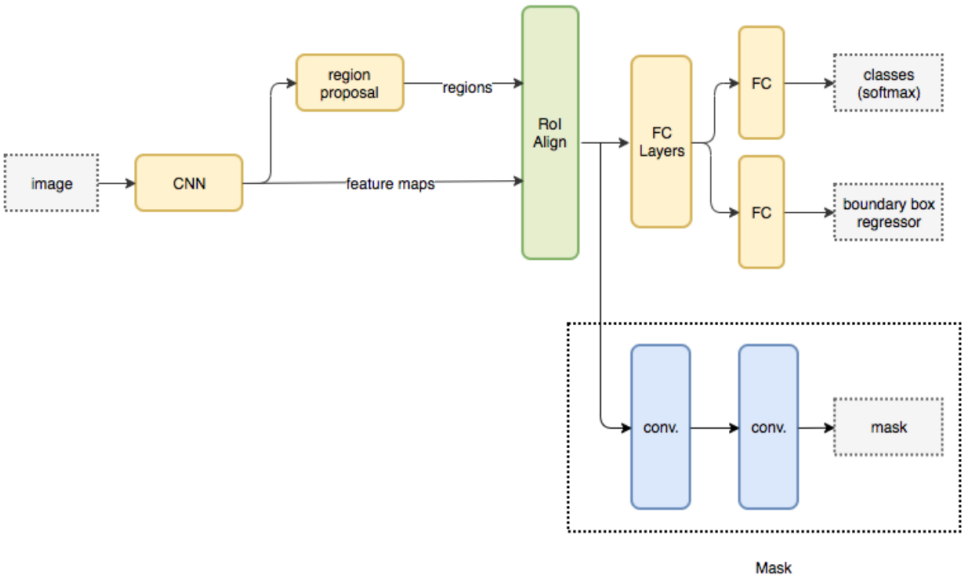


Figura 3.22: Arquitectura de *Mask R-CNN* [13].

2.8. The One Hundred Layers Tiramisu: Fully Convolutional DenseNets

Las *Densely Connected Convolutional Networks (DenseNets)* [17, 119] son construidas a partir de bloques densos (dense blocks) y operaciones de agrupación (pooling operations), donde cada bloque denso es una concatenación iterativa de un mapa de características previo. Las capas de agrupación (average[24], sum[25] y mean[26] pooling) son usadas principalmente para reducir el volumen espacial y para reducir las dimensiones de los mapas de características de las capas anteriores. La operación *mean-pooling* es la mas usada por dos principales razones: (1) eliminando valores máximos y mínimos reduce el trabajo computacional para las capas superiores y (2) provee un tipo de invariación de traducción. Por ejemplo, uno puede imaginar una capa de *mean-pooling* en cascada con una capa convolucional. Existen 8 direcciones en la que uno puede traducir el ingreso de una imagen desde una sola referencia de píxel. Si la operación de *mean-pooling* es hecha a través de una región de $3x3$, 3 de esas 8 posibles configuraciones producirán el mismo resultado en la capa convolucional.

2.8.1. Dense Block (DB)

Una capa densa (dense layer - dl) está compuesta por una capa de Batch Normalization (Ver sección 2.2.1 del presente capítulo) seguida de una capa de activación con función ReLU, una capa convolucional con un kernel de 3×3 (sin perdida de resolución) y una capa de dropout con un ratio de 0.5 tal como muestra la tabla 3.1:

[24] Promedio
[25] Suma
[26] Media

Layer
Batch Normalization
ReLU
3 x 3 Convolution
Dropout p = 0.5

Tabla 3.1: Bloque de construcción: Capa densa [16].

Luego, un bloque denso (DB) está compuesto por una cantidad determinada de estas capas, cada una sucedida por una concatenación. Cada una de estas concatenaciones está conectada a su concatenación sucesora y finalmente cada capa está conectada a la concatenación final tal como se muestra en la figura 3.23

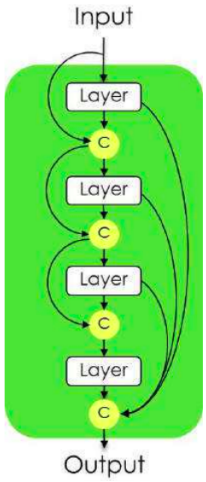


Figura 3.23: Diagrama de un bloque denso de 4 capas. Imagen extraída de [16].

2.8.2. Transition Down (TD)

Una capa de transición hacia abajo (TD) está compuesta por una capa de Batch Normalization, seguida por una activación ReLU, una capa convolutiva con un kernel de dimensiones 1×1, una capa Dropout con un ratio de 0.5 y una capa MaxPooling de tamaño 2×2 tal como muestra la tabla 3.2.

Transition Down (TD)
Batch Normalization
ReLU
1 x 1 Convolution
Dropout p = 0.5
2 x 2 MaxPooling

Tabla 3.2: Bloque de construcción: Capa de transición hacia abajo [16].

2.8.3. Transition UP (TU)

Una capa de transición hacia arriba (TU) está compuesta por una convolución transpuesta [120] con un *stride* de 2 (para compensar las operaciones de MaxPooling de las capas TD) tal como se muestra en la tabla 3.3.

Transition Up (TU)
3 x 3 Transposed Convolution strides = 2

Tabla 3.3: Bloque de construcción: Capa de transición hacia arriba [16].

2.8.4. Arquitectura

Su arquitectura puede ser vista como una extensión de las *ResNets* [84], las cuales realizan una suma iterativa de los mapas de características anteriores. Aun así, las *DenseNets* son mas eficientes en término de número de parámetros; realizan una supervisión profunda gracias a las conexiones atajos (*skip-connections*) a todos los mapas de características disponibles en la arquitectura. Todas las capas pueden ser accedidas fácilmente desde sus capas anteriores, tornando sencilla la reutilización de información desde un mapa de características previamente computado.

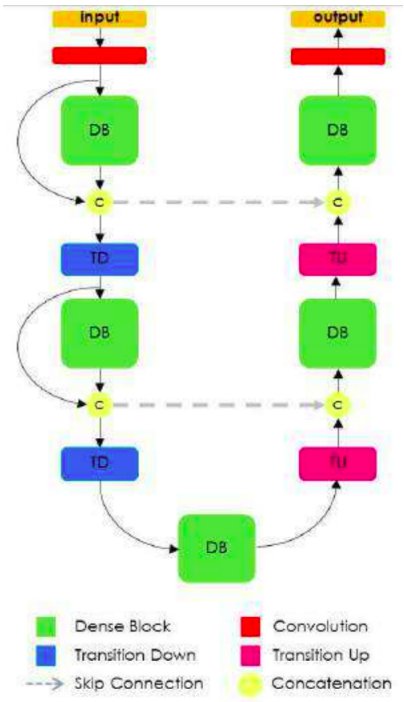


Figura 3.24: Diagrama de arquitectura de *FC-DenseNet* [17]. El diagrama esta compuesto por bloques densos, convoluciones, dos capas de transición hacia abajo (TD) y dos capas de transición hacia arriba (TU). Un circulo amarillo representa una concatenación mientras que las flechas punteadas representan las conexiones de atajo (skip connection).

3. Base de Datos OpenEDS

La base de datos OpenEDS [18] proporcionada por Facebook está conformada por imágenes de ojos usando un dispositivo de realidad aumentada (*virtual-reality Human Machine Device - HMD*) con dos cámaras sincronizadas mirando a los ojos con un ratio de frames de 200 Hz con iluminación controlada. Esta base de datos fue creada a partir de 152 participantes, de los cuales fueron capturadas 12,759 imágenes con sus máscaras de segmentación respectivas para cada clase (iris - pupila - esclera) así como se muestra en la figura 3.25:

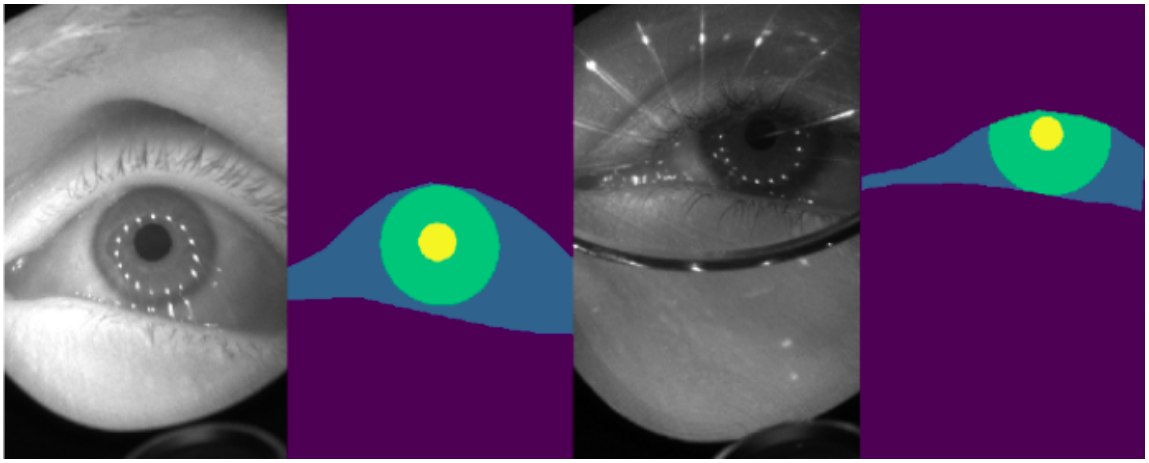


Figura 3.25: Ejemplo de imagen de base de datos OpenEDS. El color amarillo representa la pupila, el color verde el iris, el azul petróleo la esclera y el color morado el resto de la imagen. Obtenida de [18].

Esta base de datos fue dividida por Facebook en tres subsets: *Train*, *Test* y *Validation*. Mencionados sets disponen de 8,916 1,440 y 2,403 imágenes respectivamente con una resolución de 640 píxeles de altura y 400 píxeles de anchura. Dado que esta base de datos fue entregada con las marcas correspondientes de cada clase para cada imagen, no es necesario realizar un nuevo marcado manual.

4. Métricas de desempeño

4.1. Matriz de confusión

La matriz de confusión es un tipo de tabla de contingencia, típicamente utilizada en aprendizaje supervisado, que permite la visualización del rendimiento de un algoritmo. La matriz posee dos dimensiones: "real", también llamado "ground truth" (representada en las filas), y "predicho" (representada en las columnas). Las matrices de confusión facilitan distinguir si el modelo evaluado está erróneamente prediciendo una clase como otra. En la figura 3.26 se ejemplifica una matriz de confusión de dos clases.

Predicted Value (predicted by the test)		Actual Value (as confirmed by experiment)	
		positives	negatives
	positives	**TP** True Positive	**FP** False Positive
	negatives	**FN** False Negative	**TN** True Negative

Figura 3.26: Matriz de Confusión

Donde:

- **Verdaderos positivos (true positive o TP)** es el número de elementos positivos clasificados de manera correcta, por ejemplo: perros clasificados como perros.

- **Falsos positivos (false positive o FP)** es el número de elementos positivos clasificados de manera incorrecta, por ejemplo: perros clasificados como gatos.
- **Falsos negativos (false negative o FN)** es el número de elementos negativos clasificados de manera incorrecta, por ejemplo: gatos clasificados como perros.
- **Verdaderos negativos (true negative o TN)** es el número de elementos negativos clasificados de manera correcta, por ejemplo: gatos clasificados como gatos.

4.2. Precisión

$$Precisión = P = \frac{TP}{TP + FP} \tag{3.1}$$

El cálculo de la *precisión* nos permite determinar la proporción de elementos positivos clasificados correctamente, o en otras palabras, cuantos de los elementos predichos son relevantes, al dividir el número de verdaderos positivos por el número total de elementos positivos.

4.3. Recall

$$TPR = \frac{TP}{TP + FN} \tag{3.2}$$

El *recall (o TPR - True Positive Rate)* o representa la fracción de elementos relevantes que son clasificados con éxito, o cuantos elementos relevantes son predichos del total de elementos reales. Es importante notar que es trivial lograr un recall del 100 % al devolver todos los elementos en respuesta a cualquier clasificación, por lo tanto, el recall por sí solo no es suficiente para medir el desempeño del algoritmo, sino que también es necesario calcular la precisión.

La figura 3.27 ejemplifica gráficamente la precisión y el recall para un problema de dos clases.

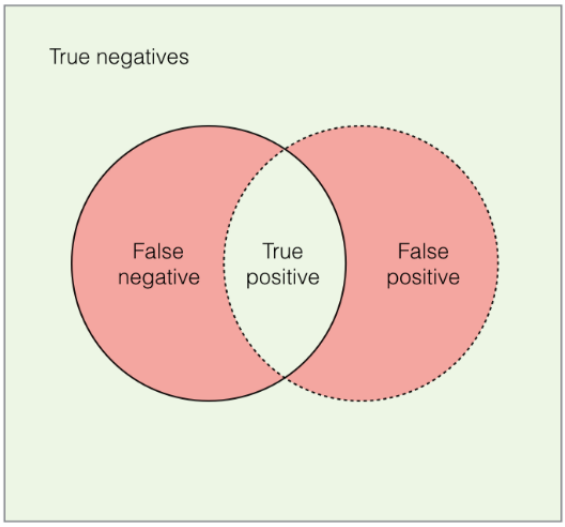


Figura 3.27: Diagrama de precisión y recall.

4.4. F-Score

$$F_\beta = \frac{(1+\beta^2) \times P \times TPR}{\beta^2 \times P + TPR} \tag{3.3}$$

$$F_1 = \frac{2 \times P \times TPR}{P + TPR} \tag{3.4}$$

El *F-Score* es la media armónica entre la precisión y el recall, y su propósito es evaluar la efectividad del modelo clasificador tomando en cuenta tanto los falsos positivos como los falsos negativos. El *F-Score* alcanza su mejor valor en 1 y su peor valor en 0. β indica a cual métrica se le da más importancia: cuando $\beta > 1$ se le da más peso al recall, mientras que si $\beta < 1$ se le da más peso a la precisión. Si $\beta = 1$, las métricas son ponderadas uniformemente, es decir, el *F-Score* seria balanceado.

4.5. True Negative Rate

$$TNR = \frac{TN}{TN + FP} \tag{3.5}$$

El *TNR (True Negative Rate)* corresponde a la proporción de negativos que se clasifican correctamente como tales.

4.6. False Positive Rate

$$FPR = \frac{FP}{FP + TN} \tag{3.6}$$

El *FPR (False Positive Rate)* se refiere a la relación entre el número de elementos negativos clasificados erróneamente como positivos (falsos positivos) y el número total de elementos negativos reales.

4.7. Accuracy

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{3.7}$$

Accuracy es la proporción de resultados verdaderos (tanto positivos verdaderos como negativos verdaderos) entre el número total de casos examinados.

4.8. Intersection Over Union

Intersection over Union o *IoU* mide el porcentaje de superposición de píxeles entre la región de interés predicha por la red neuronal y el ground truth. En otras palabras, el número de píxeles en común entre las regiones de interés divido en el número total de píxeles.

$$IoU = \frac{Predicción \cap Ground\ Truth}{Predicción \cup Ground\ Truth} \tag{3.8}$$

Esta métrica es utilizada en la tarea de detección de objetos, ya que la confianza de clasificación no es suficiente para medir el rendimiento del modelo y se vuelve necesario determinar con que capacidad la red evaluada logra localizar los objetos en la imagen utilizando bounding boxes. Los valores más cercanos a 1 indican una excelente capacidad para detectar las coordenadas del objeto y vice-versa. En el caso de tareas de segmentación, el IoU se mide utilizando máscaras. Las figuras 3.28 y 3.29 muestran ejemplos de la métrica de IoU [19].

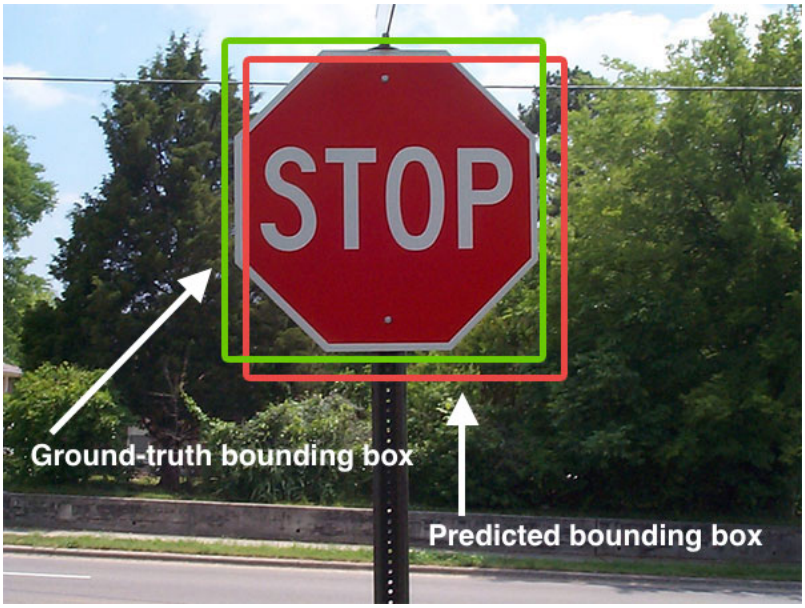


Figura 3.28: Ejemplo de métrica *IoU* en imagen [19].

Figura 3.29: Ejemplo de métrica IoU para varios *bounding boxes* [19].

4.9. Mean Intersection Over Union

mIoU (por sus siglas en inglés *Mean Intersection Over Union*) entrega la media ponderada del porcentaje de la superposición de píxeles entre cada región de interés predicha por la red neuronal y el ground truth de cada región respectivamente. En otras palabras, suma todos los IoU calculados de cada región y los divide por la cantidad de regiones totales, tal como se muestra en la siguiente formula:

$$mIoU = \frac{1}{m} \sum \frac{Predicción \cap Ground\ Truth}{Predicción \cup Ground\ Truth} \tag{3.9}$$

Donde m = número total de regiones del ground truth. Si una región no dispone de una región de interés predicha por la red neuronal, su valor de IoU es 0.0.

4.10. Average Precision

Con el fin de cuantificar las detecciones correctas del modelo, se vuelve necesario definir que constituye una detección positiva en primer lugar, para lo cual se define un umbral de IoU, por ejemplo IoU ≥ 0.5. Luego, se procede a calcular la precisión (de los objetos detectados, cuantos poseen un ground truth correspondiente) y el recall (de los objetos en el ground truth, cuantos fueron detectados de forma positiva). A partir de esto es posible graficar los mayores valores de precisión contra cada valor de recall, formando la curva Precision–Recall, y a partir de esto, calcular el área bajo dicha curva para obtener el Average Precision (AP). La figura 3.30 muestra un ejemplo de una curva Precision–Recall [20].
Habiendo obtenido el *Average Precision*, es posible determinar el *mean Average Precision* (*mAP*), que corresponde al promedio simple del *AP* de cada clase.

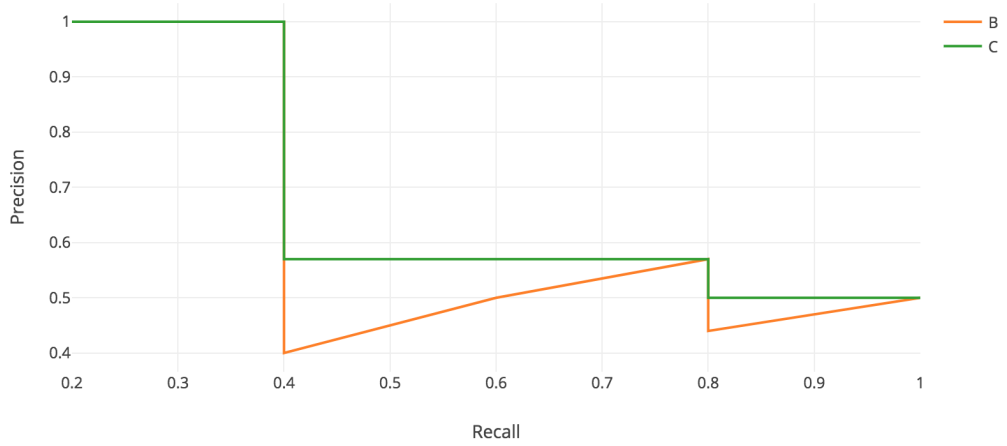


Figura 3.30: Ejemplo de curva Precision–Recall [20]. La curva verde representa los mayores valores de Precision por nivel de Recall.

4.11. EvalAI Performance Metric

Esta métrica sigue las reglas de la competencia "*The Eye Tracking Semantic Segmentation Challenge*"[27] realizada por Facebook, donde los modelos fueron evaluados por un programa en la página web de EvalAI

[27] https://research.fb.com/programs/openeds-challenge

usando la métrica de desempeño $(0 < M \leq 100)$[28] definida a continuación:

$$M = 50(P + min(\frac{1}{S})) \tag{3.10}$$

donde $0 < P \leq 1$ mide el desempeño del modelo definido como el IoU (Ver métrica 4.8) obtenido de un set de pruebas y $S > 0$ mide la complejidad del modelo (definido como el número de parámetros del modelo) medido en MegaBytes. Esta métrica considera el trade-off entre el desempeño de un modelo y la complejidad del mismo. Valores cercanos a 1 indican una buena puntuación, mientras que valores cercanos a 0 representan una mala puntuación.

[28] https://evalai.cloudcv.org/web/challenges/challenge-page/353/evaluation

CAPÍTULO 4

Experimentos y Análisis de Resultados

Para el presente capítulo fueron realizados tres grandes experimentos, donde en el primero de estos fue evaluado el framework Mask R-CNN con los pesos de otros proyectos y así demostrar que su arquitectura debía ser reentrenada para responder así a la problemática expuesta en esta tesis. Para el segundo experimento fueron realizadas 24 pruebas en total utilizando el framework Mask R-CNN con los backbones *ResNet50* y *ResNet101* con el motivo de obtener una métrica de *IoU* competente con el estado del arte. Finalmente, para el tercer experimento fue utilizada una arquitectura basada en DenseNet para la competencia "*The Eye Tracking Semantic Segmentation Challenge*"[1], en la cual se obtuvo el octavo lugar[2], quedando por debajo del primer lugar por 5 milésimas de diferencia tal como se muestra en la figura 4.1:

Rank	Participant team	mIoU	ModelComplexity	Total	Last submission at
1	RIT-MVRL (FINAL)	0.95276	248,900.00000	0.97638	2 months ago
2	tetelias	0.95189	242,664.00000	0.97595	2 months ago
3	Couger AI (Eye Net 3)	0.95112	258,021.00000	0.97556	2 months ago
4	CVLab (CLv2)	0.94971	261,860.00000	0.97485	2 months ago
5	TH2L	0.94911	104,728.00000	0.97455	2 months ago
6	tomcarrot	0.94781	259,244.00000	0.97390	2 months ago
7	manifold	0.94299	211,094.00000	0.97149	4 months ago
8	UNAB_01 (60_epochs_202k_67)	0.94293	202,084.00000	0.97147	2 months ago
9	perception	0.94232	228,834.00000	0.97116	3 months ago
10	PAU	0.94175	205,520.00000	0.97088	4 months ago

Figura 4.1: Tabla de clasificación de la competencia *The Eye Tracking Semantic Segmentation Challenge* realizada por Facebook, donde el equipo UNAB_01 corresponde al equipo conformado por el profesor guía y el autor de esta tesis. La presente tabla fue obtenida de *Leaderboard-EvalAI*.

[1] https://research.fb.com/programs/openeds-challenge
[2] https://evalai.cloudcv.org/web/challenges/challenge-page/353/leaderboard/1002

1. Experimento con Mask R-CNN

La base de datos utilizada para los experimentos de segmentación de imágenes con el framework Mask R-CNN fue elaborada de acuerdo a lo mencionado en la subsección 5.1 de la metodología. Una cantidad total de 130 imágenes fueron marcadas manualmente, lo que corresponde a:

- 130 objetos de la clase pupila
- 130 objetos de la clase iris
- 130 objetos de la clase esclera izquierda
- 130 objetos de la clase esclera derecha
- 56 objetos de la clase maquillaje

Para efecto de los experimentos de esta sección, fue creado un programa en Python el cuál recopiló todas las imágenes y sus respectivos archivos JSON de la primera base de datos y las distribuyó aleatoriamente en sets de *train* y *val* de acuerdo a un número deseado y especificado (por ejemplo, de 100 imágenes, 80 imágenes son designadas para entrenamiento y el restante 20 para validación), obteniendo así una división de 80 imágenes para el set de entrenamiento y 50 imágenes para el set de validación. Para la clase maquillaje, fue constatado que el set de entrenamiento tiene 32 objetos de la clase mencionada, mientras que el set de validación tiene 24. Para el resto de clases, la división es la siguiente: 80 elementos de cada clase para el set de entrenamiento y 50 elementos de cada clase para el set de validación.
Para el análisis visual de los resultados de los experimentos, fueron seleccionadas dos imágenes de control del set de validación con la finalidad de realizar la misma comparación entre los experimentos. Siendo así, se muestra en la figura 4.2 las imágenes mencionadas:

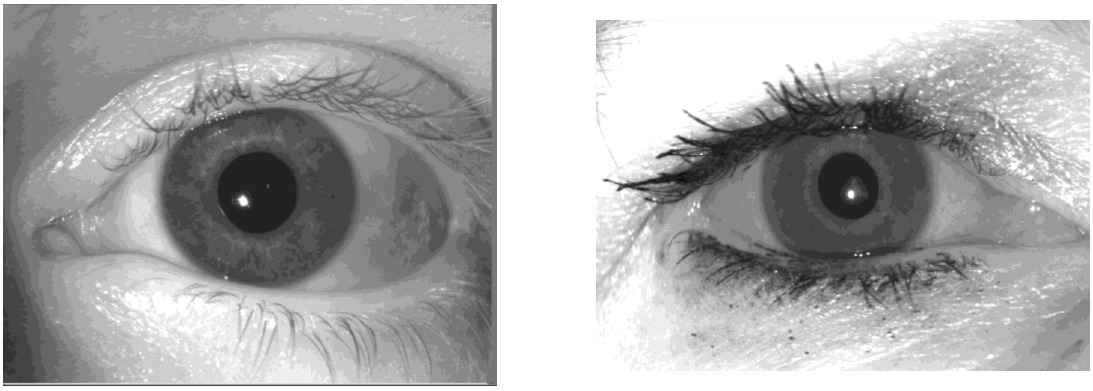


Figura 4.2: La imagen de la izquierda corresponde al ojo de un hombre, mientras que la imagen de la derecha corresponde a un ojo maquillado de una mujer.

La implementación de Mask R-CNN utilizada corresponde a una versión modificada de la implementación creada para Python 3, Keras y Tensorflow por Matterport (release 2.1) [121]. Las modificaciones realizadas consisten en ajustes de lectura de la base de datos desarrollada (Ver metodología, sección 5.2), además de métodos de visualización de resultados y calculo de métricas de mIoU (Ver métricas, sección 4.9). La GPU utilizada para el entrenamiento de los modelos de Mask R-CNN es una NVIDIA Quadro M6000 de 24GB [122].
Para el calculo de la métrica mIoU fue considerado un umbral de $IoU \geq 0{,}5$ (métrica estándar) [123]

Adicionalmente, fueron utilizadas las técnicas de transfer learning [85] (con los pesos de Imagenet [100] y COCO[3]) y Data Augmentation incrementando así con este último la cantidad de imágenes totales usando la herramienta *imgaug* [124]. Los aumentadores utilizados y sus parámetros respectivos fueron:

1. Volteado horizontal (Fliplr con $p = 1{,}0$)
2. Volteado vertical (Flipud con $p = 1{,}0$)
3. Rotación entre -45 y 45 grados sobre el centro (Affine con $rotate = (-45, 45)$)
4. Aumento de brillo en 15 % (en escala de 0 a 100) (Add con $value = 15$)
5. Decremento de brillo en 15 % (en escala de 0 a 100) (Add con $value = 15$)
6. Ruido Gaussiano aditivo (AdditiveGaussianNoise con $loc = 12{,}75$)
7. Ruido Laplaciano aditivo (AdditiveLaplaceNoise con $loc = 12{,}75$)
8. Ruido Poisson aditivo (AdditivePoissonNoise con $lam = 16{,}0$)
9. Desenfoque Gaussiana (GaussianBlur con $sigma = 0{,}5$)
10. Desenfoque promedio (AverageBlur con $k = 2$)
11. Desenfoque de movimiento (MotionBlur con $k = 3$)
12. Contraste Gamma (GammaContrast con $gamma = 0{,}5$)
13. Remarcado de bordes (EdgeDetect con $alpha = 0{,}1$)

Todos estos aumentadores fueron programados para tener una probabilidad de ocurrencia del 75 %, es decir que de cada 100 imágenes, 75 serán.
Algunas consideraciones a tener en cuenta son:

- Los aumentadores 1 y 2 son seleccionados ambos, uno por si solo o ninguno con probabilidad del 50 %.
- El aumentador 3 es seleccionado el 50 % de las veces.
- Es seleccionado solo uno de los aumentadores 4 y 5.
- Solo es seleccionado un rango de 0 a 3 de los segmentadores 6, 7, 8, 9, 10, 11, 12 y 13.
- Luego de obtener una lista de segmentadores, estos se aplican de manera aleatoria sobre la imagen a aumentar.

Algunos ejemplos del Data Augmentation antes descrito se pueden ver en la figura 4.3:

[3] https://github.com/matterport/Mask_RCNN/releases/tag/v2.1

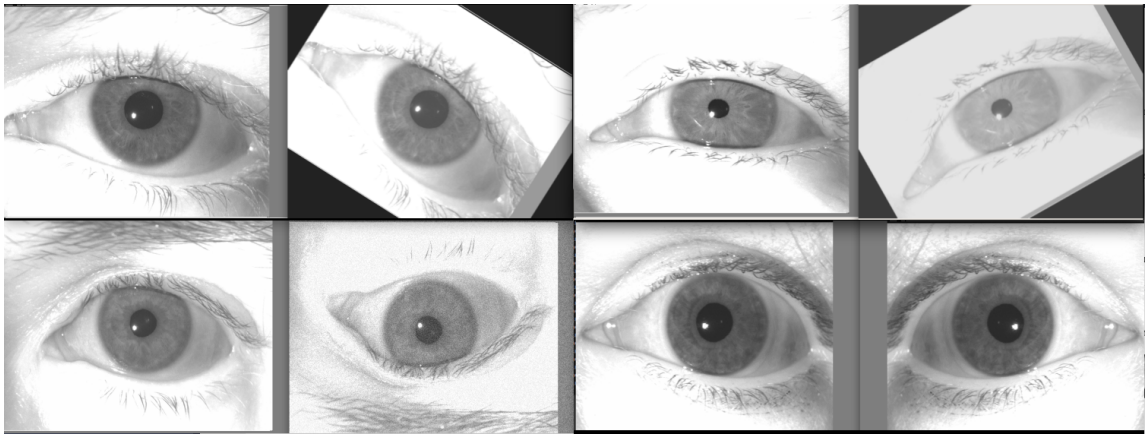


Figura 4.3: Ejemplo de Data Augmentation para cuatro imágenes distintas. Imágenes obtenidas de [1]

Los resultados de segmentación finales por prueba fueron obtenidos con el modelo con más alto mIoU.

1.1. Pre-trained Weights

En este apartado fueron realizadas pruebas de segmentación con los pesos de COCO y de un proyecto personal de Matterport donde globos eran segmentados [121], para demostrar que estas redes pre-entrenadas no eran capaces de segmentar ojos y que era necesario un nuevo entrenamiento. Se muestra así en la figura 4.4 los resultados de segmentación para los pesos de COCO:

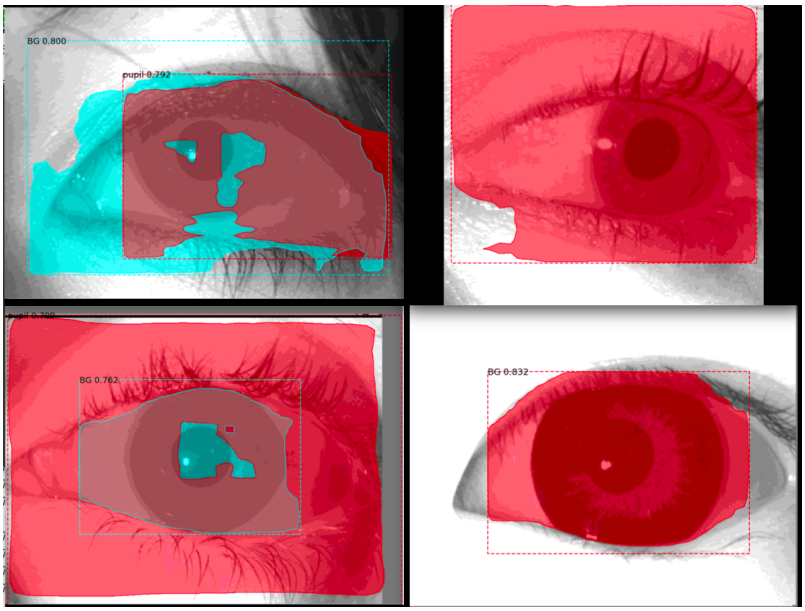


Figura 4.4: Resultados de segmentación con pesos de COCO - Red preentrenada.

Luego, resultados de segmentación para los pesos del proyecto de Matterport:

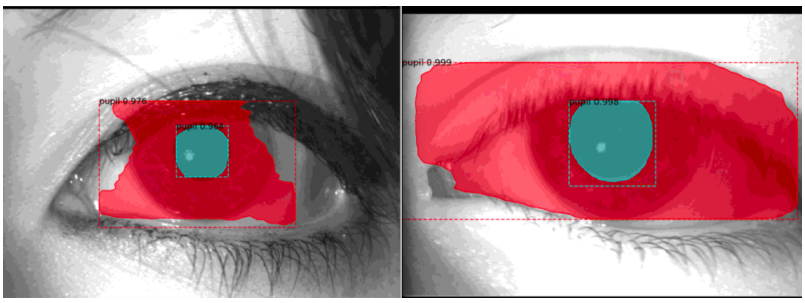


Figura 4.5: Resultados de segmentación con pesos de proyecto de Matterport (Balloons) - Red preentrenada.

1.2. ResNet50

Para este apartado, fueron realizadas 5 pruebas con el backbone ResNet50. En cada prueba fue entrenado con un total de 80 imágenes y validado con 50 imágenes.

1.2.1. Prueba 01

La primera prueba fue realizada para obtener un acercamiento al framework Mask R-CNN y así ajustar los hiper-parámetros para posteriores pruebas.
Los hiper-parámetros utilizados para esta prueba fueron:

Parameter	Value
Weights	COCO
Images per GPU	2
Image Min Dimension	800
Image Max Dimension	1024
Learning Rate	0.001
Weight Decay	0.0001
Detection Min Confidence	0.9
Epochs	50
Batch Size	2
Steps	40
Layers	heads
Data Augmentation	No

Tabla 4.1: Los steps corresponden al calculo: $steps = TI/(BatchSize)$, donde TI corresponde a las 80 imágenes de entrenamiento usadas.

Obteniendo los siguientes resultados de mIoU por época:

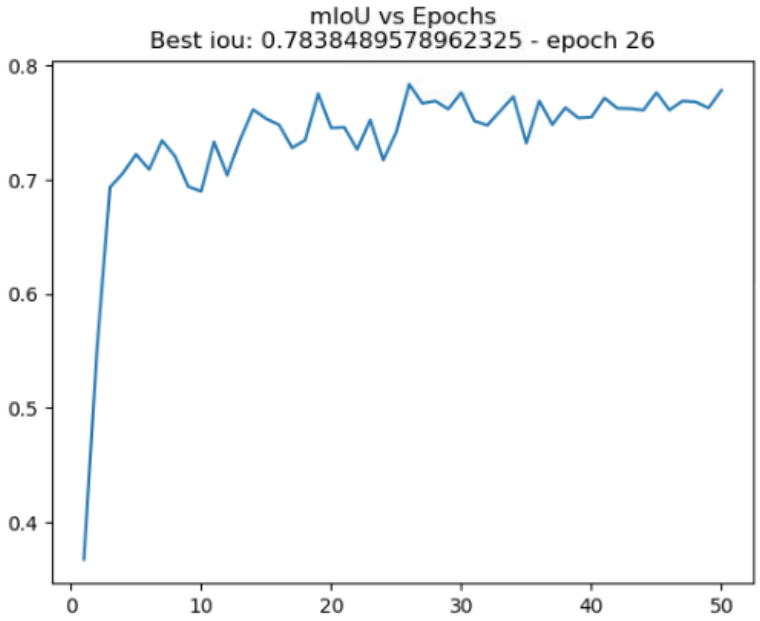


Figura 4.6: Resultados de mIoU por época de la prueba 01 con ResNet50 por backbone. El modelo con mejor rendimiento fue el de la época 26 con un mIoU de 0.78384 para todo el set de validación. El eje-X corresponde a las épocas, mientras que el eje-Y corresponde al mIoU.

La figura 4.7 muestra los resultados de segmentación con el mejor modelo para la presente prueba:

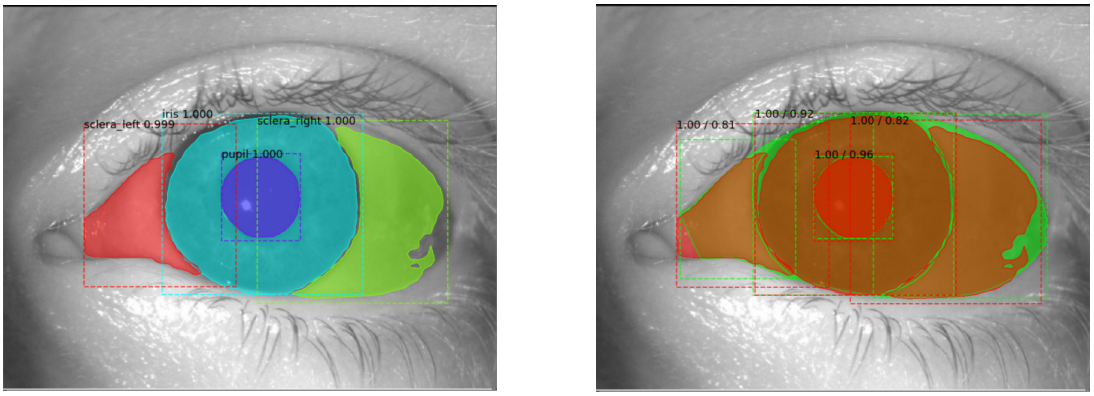


Figura 4.7: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.2:

Class	IoU
Iris	0.92201
Pupil	0.96381
Sclera right	0.82436
Sclera left	0.80883
mIoU	0.87975

Tabla 4.2: IoU por clase y mIoU promedio de cada clase.

La figura 4.8 muestra los resultados de segmentación con el mejor modelo para una imagen con presencia de maquillaje:

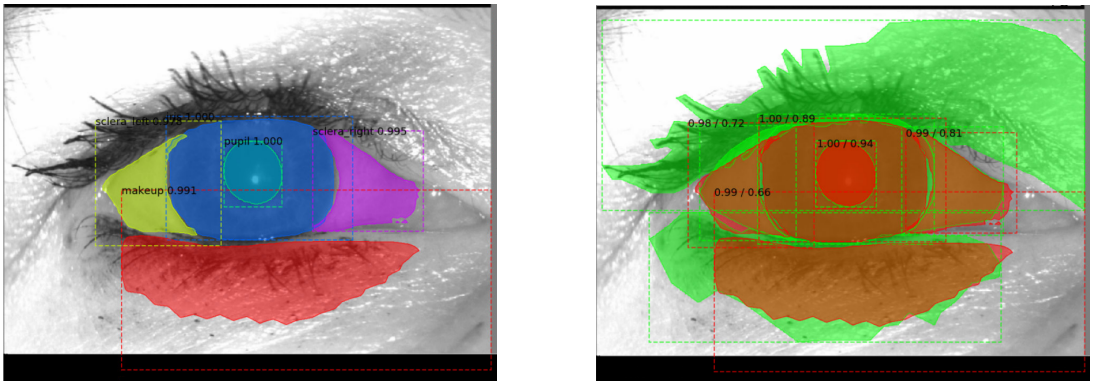


Figura 4.8: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.3:

Class	IoU
Iris	0.88878
Pupil	0.93788
Sclera right	0.80983
Sclera left	0.71568
MakeUp	0.66490
MakeUp	0.00000
mIoU	0.66951

Tabla 4.3: IoU por clase y mIoU promedio de cada clase. Se reporta dos veces la clase maquillaje puesto que existen dos áreas del ojo que presentan esta clase.

1.2.2. Prueba 02

Para la segunda prueba se realiza fine tunning con los pesos de IMAGENET, esto con el objetivo de analizar cuanto más mejora el modelo respecto al entrenamiento anterior. Además, se incrementan los steps al doble. Esto indica que el modelo será entrenado con una misma imagen dos veces por época. Los hiper-parámetros utilizados para esta prueba fueron:

Parameter	Value
Weights	IMAGENET
Images per GPU	2
Image Min Dimension	800
Image Max Dimension	1024
Learning Rate	0.001
Weight Decay	0.0001
Detection Min Confidence	0.9
Epochs	50
Batch Size	2
Steps	80
Layers	heads
Data Augmentation	No

Tabla 4.4: Los steps corresponden al calculo: $steps = [TI/(BatchSize)] * 2$, donde TI corresponde a las 80 imágenes de entrenamiento usadas.

Obteniendo los siguientes resultados de mIoU por época:

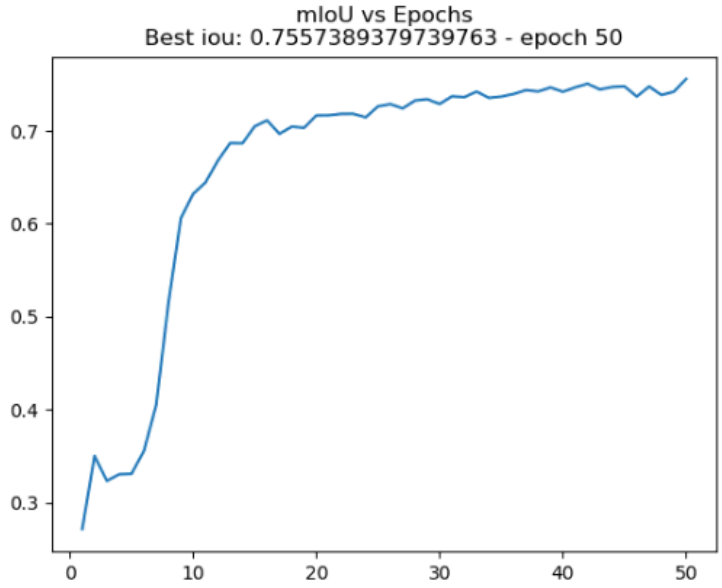


Figura 4.9: Resultados de mIoU por época de la prueba 02 con ResNet50 por backbone. El modelo con mejor rendimiento fue el de la época 50 con un mIoU de 0.75573 para todo el set de validación. El eje-X corresponde a las épocas, mientras que el eje-Y corresponde al mIoU.

La figura 4.10 muestra los resultados de segmentación con el mejor modelo para la presente prueba:

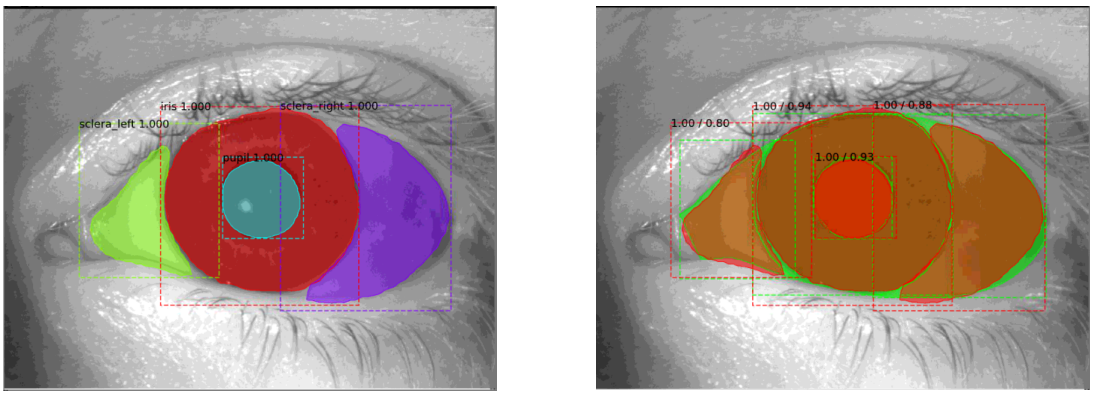


Figura 4.10: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.5:

Class	IoU
Iris	0.93701
Pupil	0.92902
Sclera right	0.87769
Sclera left	0.80161
mIoU	0.88633

Tabla 4.5: IoU por clase y mIoU promedio de cada clase.

La figura 4.11 muestra los resultados de segmentación con el mejor modelo para una imagen con presencia de maquillaje:

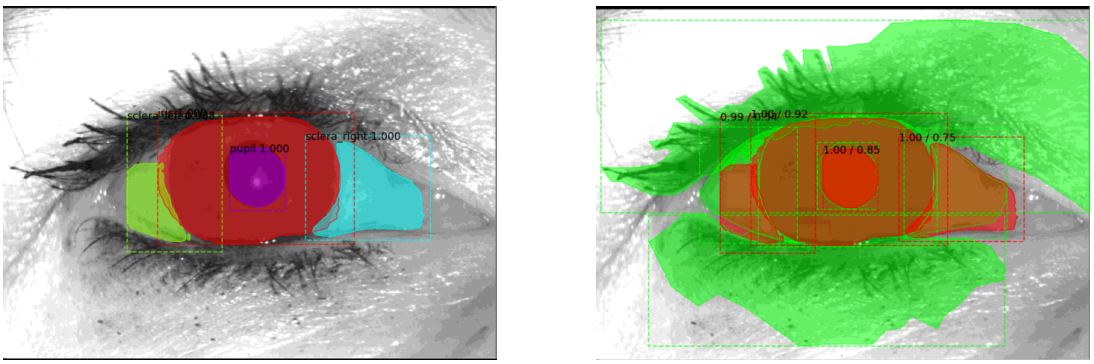


Figura 4.11: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.6:

Class	IoU
Iris	0.92487
Pupil	0.850104
Sclera right	0.74789
Sclera left	0.54018
MakeUp	0.00000
MakeUp	0.00000
mIoU	0.51051

Tabla 4.6: IoU por clase y mIoU promedio de cada clase. Se reporta dos veces la clase maquillaje puesto que existen dos áreas del ojo que presentan esta clase.

1.2.3. Prueba 03

Los resultados de la prueba anterior muestran que los pesos de COCO proveen un mejor resultado que los pesos de IMAGENET. Dado lo mencionado, se procede a realizar un reentrenamiento a la arquitectura completa de ResNet50. Los hiper-parámetros utilizados para esta prueba fueron:

Parameter	**Value**
Weights	COCO
Images per GPU	2
Image Min Dimension	800
Image Max Dimension	1024
Learning Rate	0.001
Weight Decay	0.0001
Detection Min Confidence	0.9
Epochs	50
Batch Size	2
Steps	80
Layers	all
Data Augmentation	No

Tabla 4.7: Los steps corresponden al calculo: $steps = [TI/(BatchSize)] * 2$, donde TI corresponde a las 80 imágenes de entrenamiento usadas.

Obteniendo los siguientes resultados de mIoU por época:

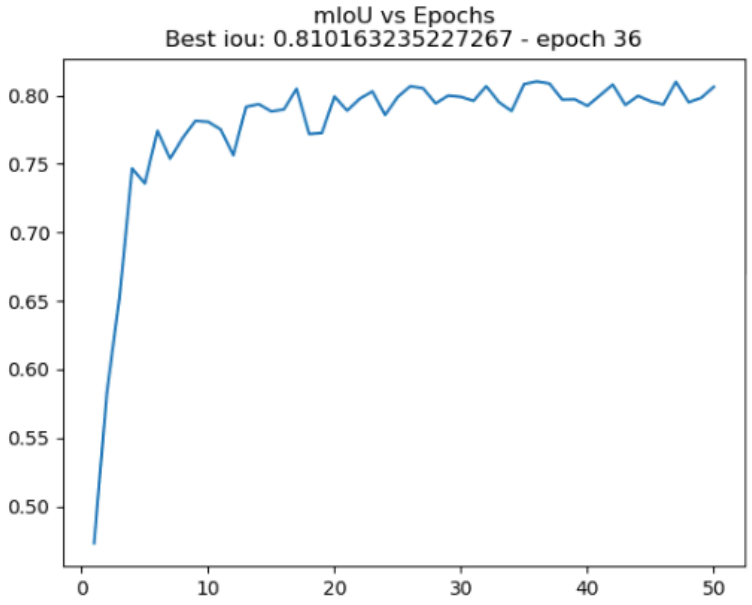


Figura 4.12: Resultados de mIoU por época de la prueba 03 con ResNet50 por backbone. El modelo con mejor rendimiento fue el de la época 36 con un mIoU de 0.81016 para todo el set de validación. El eje-X corresponde a las épocas, mientras que el eje-Y corresponde al mIoU.

La figura 4.13 muestra los resultados de segmentación con el mejor modelo para la presente prueba:

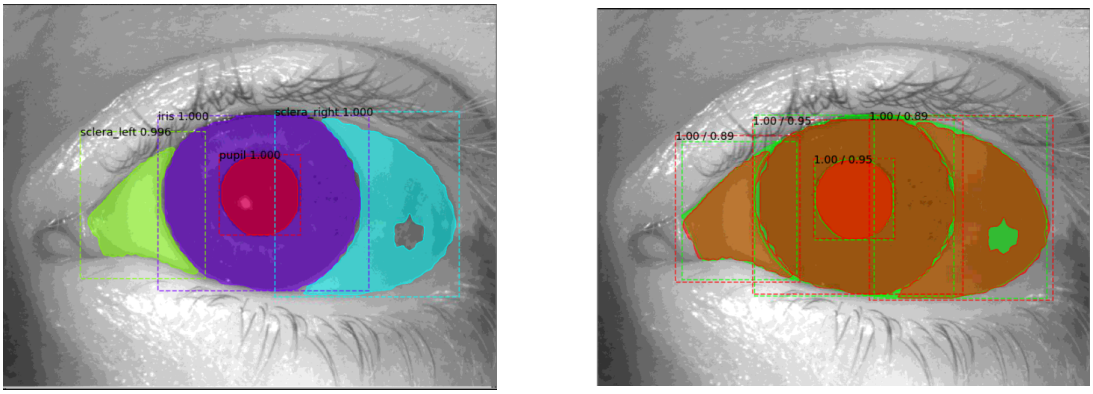


Figura 4.13: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.8:

Class	IoU
Iris	0.94500
Pupil	0.94897
Sclera right	0.88571
Sclera left	0.89093
mIoU	0.91765

Tabla 4.8: IoU por clase y mIoU promedio de cada clase.

La figura 4.14 muestra los resultados de segmentación con el mejor modelo para una imagen con presencia de maquillaje:

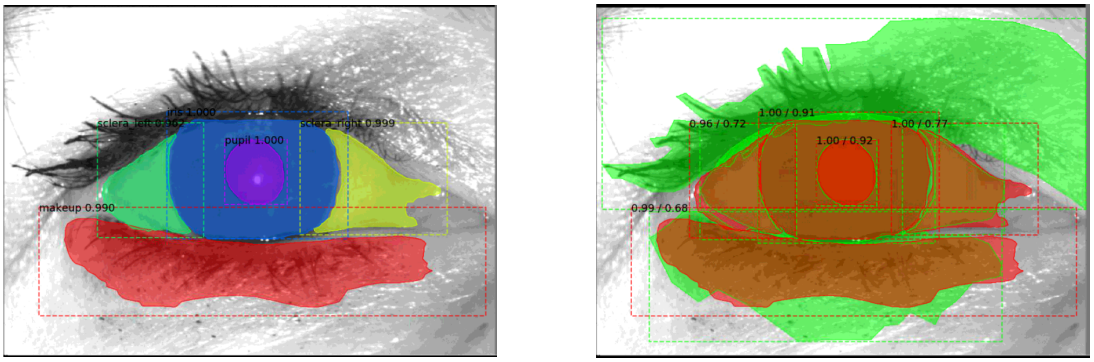


Figura 4.14: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.9:

Class	IoU
Iris	0.90784
Pupil	0.92309
Sclera right	0.76696
Sclera left	0.72057
MakeUp	0.67836
MakeUp	0.00000
mIoU	0.66614

Tabla 4.9: IoU por clase y mIoU promedio de cada clase. Se reporta dos veces la clase maquillaje puesto que existen dos áreas del ojo que presentan esta clase.

1.2.4. Prueba 04

La prueba anterior resulto en un aumento para la métrica mIoU del modelo. Se estima necesario reentrenar toda la red ResNet50 pero ahora con los pesos de IMAGENET. Los hiper-parámetros utilizados para esta prueba fueron:

Parameter	Value
Weights	IMAGENET
Images per GPU	2
Image Min Dimension	800
Image Max Dimension	1024
Learning Rate	0.001
Weight Decay	0.0001
Detection Min Confidence	0.9
Epochs	50
Batch Size	2
Steps	80
Layers	all
Data Augmentation	No

Tabla 4.10: Los steps corresponden al calculo: $steps = [TI/(BatchSize)] * 2$, donde TI corresponde a las 80 imágenes de entrenamiento usadas.

Obteniendo los siguientes resultados de mIoU por época:

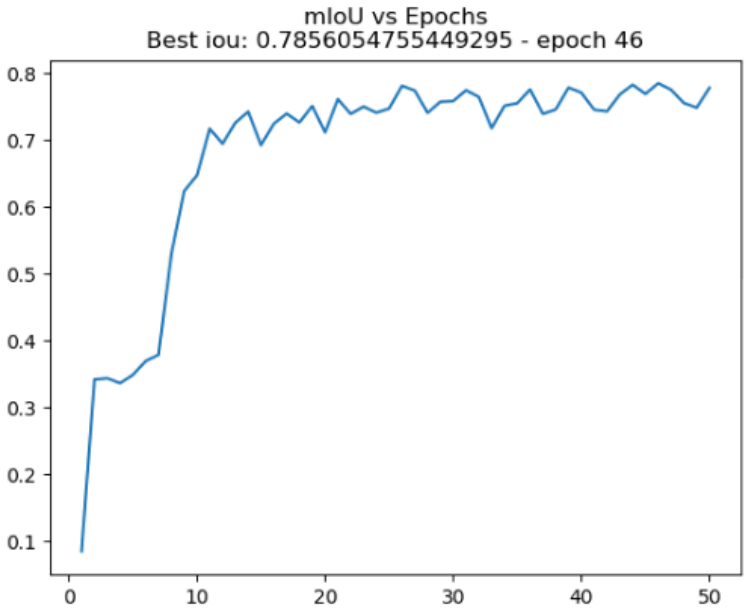


Figura 4.15: Resultados de mIoU por época de la prueba 04 con ResNet50 por backbone. El modelo con mejor rendimiento fue el de la época 46 con un mIoU de 0.78560 para todo el set de validación. El eje-X corresponde a las épocas, mientras que el eje-Y corresponde al mIoU.

La figura 4.16 muestra los resultados de segmentación con el mejor modelo para la presente prueba:

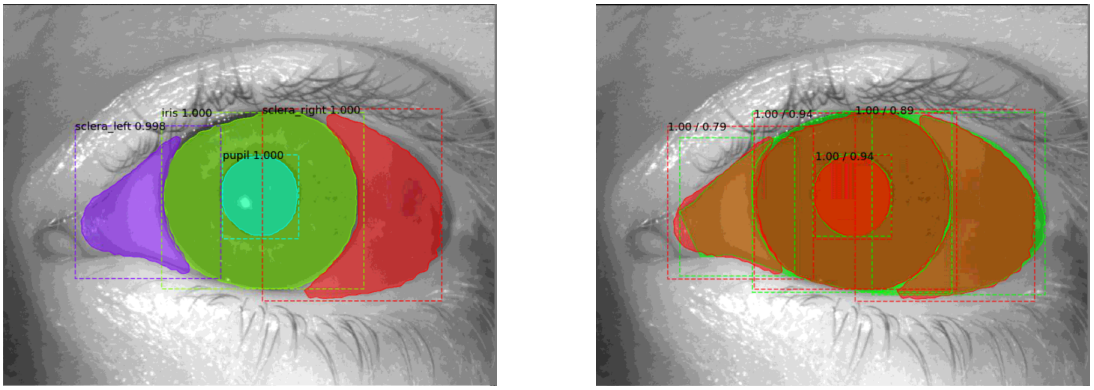


Figura 4.16: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.11:

Class	IoU
Iris	0.94308
Pupil	0.94173
Sclera right	0.88982
Sclera left	0.78790
mIoU	0.89063

Tabla 4.11: IoU por clase y mIoU promedio de cada clase.

La figura 4.17 muestra los resultados de segmentación con el mejor modelo para una imagen con presencia de maquillaje:

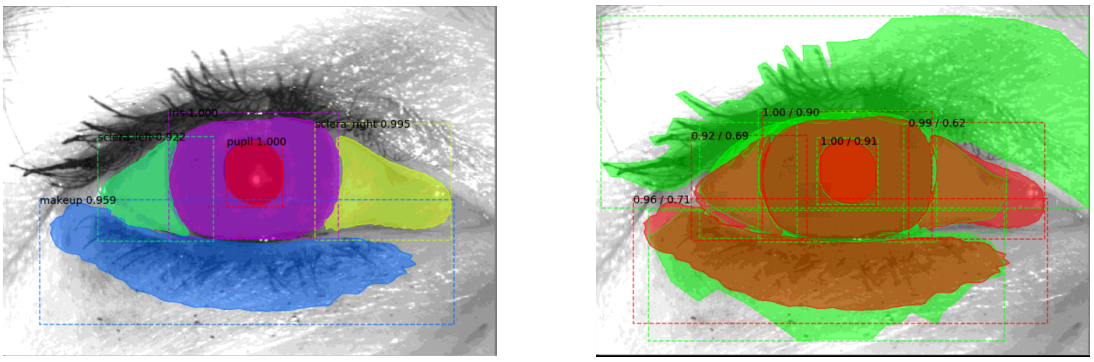


Figura 4.17: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.12:

Class	IoU
Iris	0.89931
Pupil	0.91454
Sclera right	0.61982
Sclera left	0.69257
MakeUp	0.70579
MakeUp	0.00000
mIoU	0.63867

Tabla 4.12: IoU por clase y mIoU promedio de cada clase. Se reporta dos veces la clase maquillaje puesto que existen dos áreas del ojo que presentan esta clase.

1.2.5. Prueba 05

El reentrenamiento de todas las capas de ResNet50 incrementa el mIoU usando tanto los pesos de COCO como de IMAGENET, pero este ultimo presenta el menor rendimiento. Ante esta situación, se realiza un reentrenamiento a todas las capas de ResNet50 usando los pesos de COCO, incrementando las épocas, los steps y empleando la técnica data augmentation descrita al principio de este capítulo. Los hiper-parámetros utilizados para esta última prueba fueron:

Parameter	**Value**
Weights	COCO
Images per GPU	2
Image Min Dimension	800
Image Max Dimension	1024
Learning Rate	0.001
Weight Decay	0.001
Detection Min Confidence	0.9
Epochs	200
Batch Size	2
Steps	800
Layers	all
Data Augmentation	Yes

Tabla 4.13: Los steps corresponden al calculo: $steps = [TI/(BatchSize)] * 20$, donde TI corresponde a las 80 imágenes de entrenamiento usadas.

Obteniendo los siguientes resultados de mIoU por época:

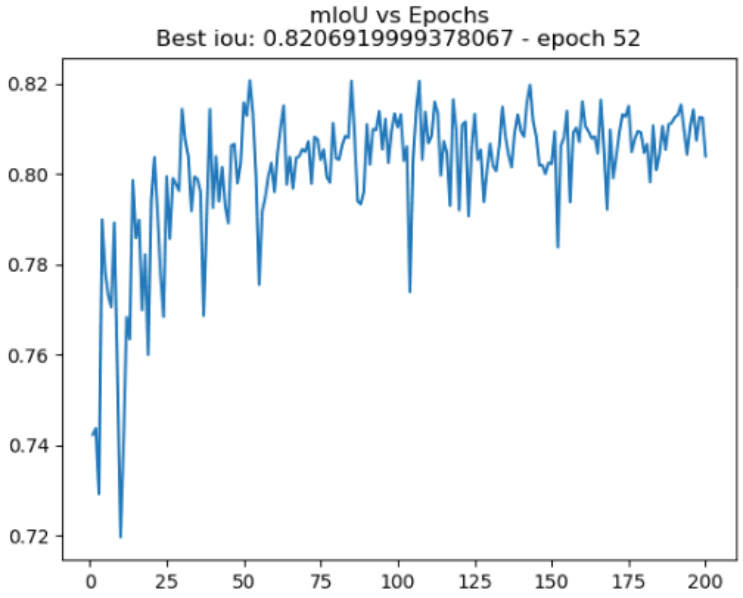


Figura 4.18: Resultados de mIoU por época de la prueba 05 con ResNet50 por backbone. El modelo con mejor rendimiento fue el de la época 52 con un mIoU de 0.82069 para todo el set de validación. El eje-X corresponde a las épocas, mientras que el eje-Y corresponde al mIoU.

La figura 4.19 muestra los resultados de segmentación con el mejor modelo para la presente prueba:

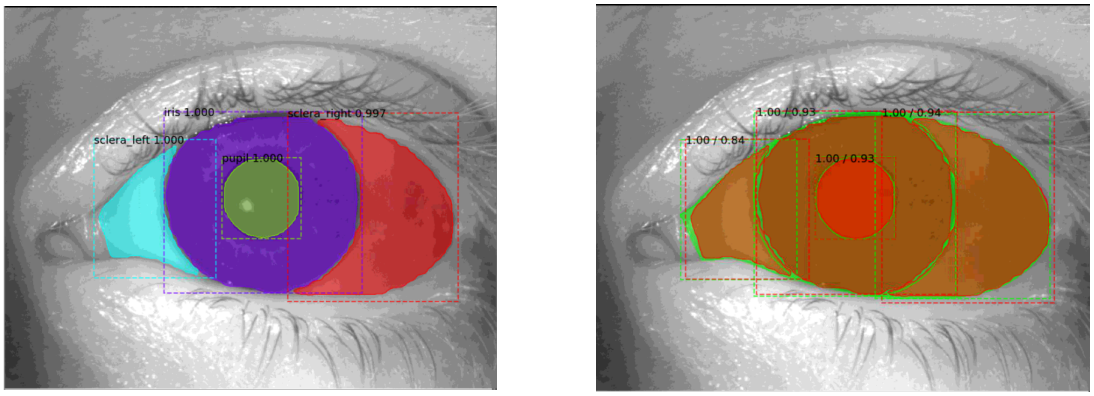


Figura 4.19: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.14:

Class	IoU
Iris	0.92944
Pupil	0.92966
Sclera right	0.94309
Sclera left	0.84184
mIoU	0.91101

Tabla 4.14: IoU por clase y mIoU promedio de cada clase.

La figura 4.20 muestra los resultados de segmentación con el mejor modelo para una imagen con presencia de maquillaje:

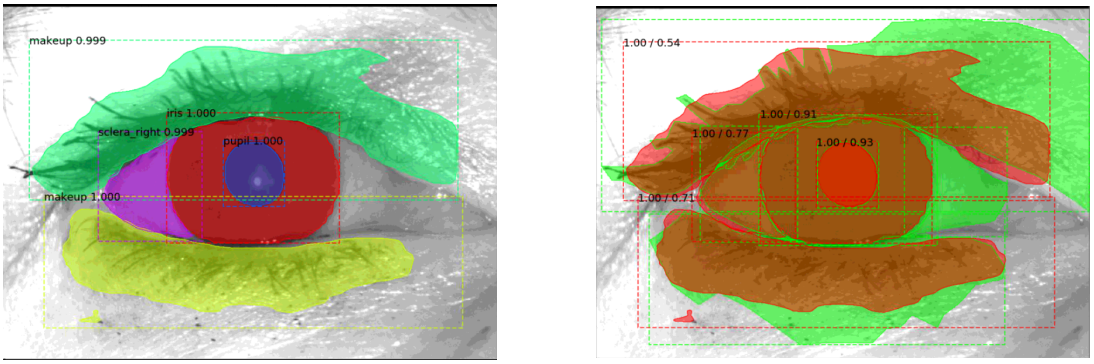


Figura 4.20: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.15:

Class	IoU
Iris	0.90684
Pupil	0.92891
Sclera right	0.76955
Sclera left	0.00000
MakeUp	0.71428
MakeUp	0.54001
mIoU	0.63867

Tabla 4.15: IoU por clase y mIoU promedio de cada clase. Se reporta dos veces la clase maquillaje puesto que existen dos áreas del ojo que presentan esta clase.

1.2.6. Resumen y análisis de pruebas ResNet50

Se presenta finalmente una tabla resumiendo las 5 pruebas anteriores:

Prueba	**Epocas**	**DA**	**Steps**	**Layers**	**Weights**	**mIoU**
01	50	No	40	heads	COCO	0.78384
02	50	No	80	heads	IMAGENET	0.75573
03	50	No	80	all	COCO	0.81016
04	50	No	80	all	IMAGENET	0.78560
05	200	Yes	800	all	COCO	0.82069

Tabla 4.16: Tabla de resumen de pruebas con ResNet50, donde la prueba 02 obtuvo el peor resultado y la prueba 05 obtuvo el mejor resultado (en términos de mIoU).

Adicionalmente, se reportan los mIoU y desviación estándar de la mejor prueba (05) separados por cada clase de la carpeta de validación en la tabla 4.17:

Class	mIoU	std
Iris	0.84483	0.09518
Pupil	0.91935	0.05493
Sclera left	0.83444	0.18248
Sclera right	0.80970	0.21543
MakeUp	0.57758	0.39763

Tabla 4.17: mIoU y desviación estándar de cada clase del set de validación (50 imágenes).

De las pruebas realizados con ResNet50 se puede rescatar que:

- Realizar transfer learning con los pesos de COCO ayuda a obtener buen desempeño de manera mas pronta.
- Reentrenar toda la red provee mejores resultados
- La cantidad de épocas es directamente proporcional con el rendimiento final del modelo.
- Sin aplicar data augmentation, incrementar la cantidad de steps de manera moderada (doble o triple) tiene un impacto levemente positivo sobre el rendimiento final del modelo.
- Aplicar data augmentation tiene un gran impacto positivo en el rendimiento final de la red.
- Todo lo anterior combinado (aumentar steps, epocas, aplicar data augmentation y reentrenar todo el backbone) resulta en un desempeño aun mayor que aplicando cada punto por si solo.
- Con ResNet50 por backbone, la red Mask R-CNN demora 0,1487 segundos en promedio realizar la predicción de máscaras de una imagen.

Dado los resultados mostrados en la tabla 4.16, se da inicio a pruebas con ResNet101 por backbone.

1.3. ResNet101

Para este apartado, fueron realizadas 18 pruebas con el backbone ResNet101. Cada prueba fue entrenada con un total de 80 imágenes y validado con 50 imágenes.

1.3.1. Prueba 01

Los hiper-parámetros utilizados para esta prueba fueron similares a los empleados para la mejor prueba con ResNet50:

Parameter	Value
Weights	COCO
Images per GPU	2
Image Min Dimension	800
Image Max Dimension	1024
Learning Rate	0.001
Weight Decay	0.0001
Detection Min Confidence	0.9
Epochs	100
Batch Size	2
Steps	80
Layers	heads
Data Augmentation	No

Tabla 4.18: Los steps corresponden al calculo: $steps = [TI/(BatchSize)] * 2$, donde TI corresponde a las 80 imágenes de entrenamiento usadas.

Obteniendo los siguientes resultados de mIoU por época:

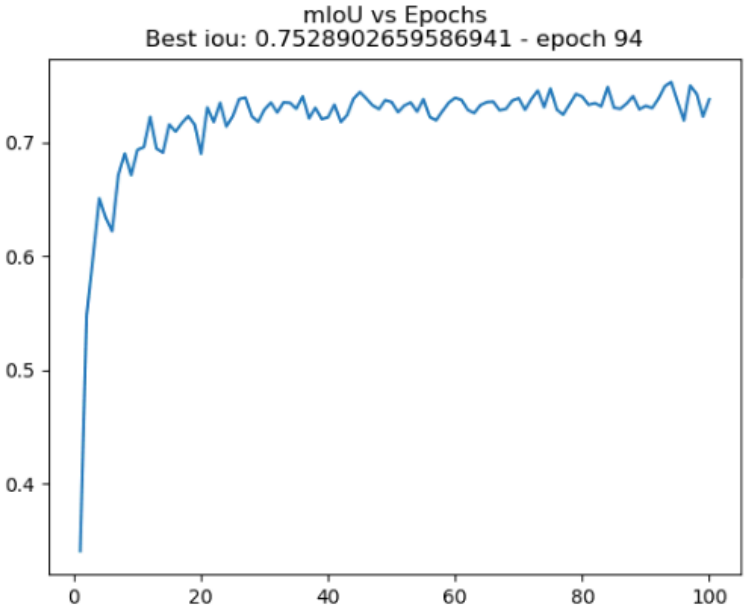


Figura 4.21: Resultados de mIoU por época de la prueba 01 con ResNet101 por backbone. El modelo con mejor rendimiento fue el de la época 94 con un mIoU de 0.75289 para todo el set de validación. El eje-X corresponde a las épocas, mientras que el eje-Y corresponde al mIoU.

La figura 4.22 muestra los resultados de segmentación con el mejor modelo para la presente prueba:

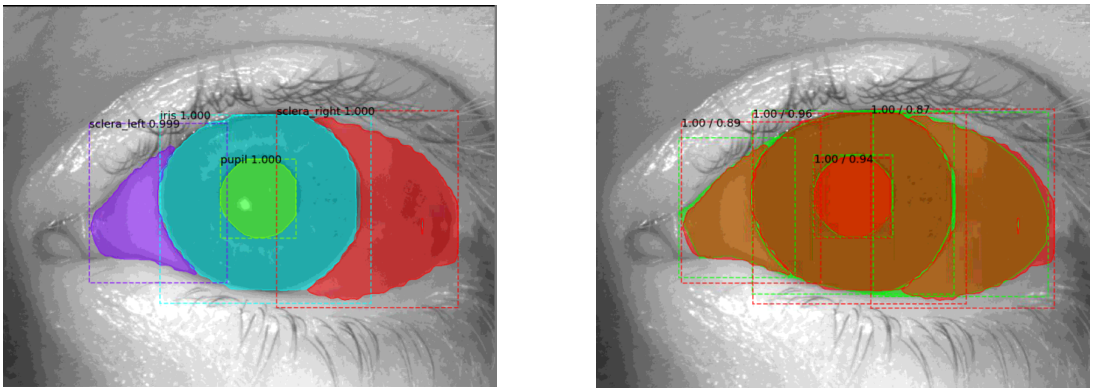


Figura 4.22: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.19:

Class	IoU
Iris	0.96487
Pupil	0.94346
Sclera right	0.87365
Sclera left	0.88705
mIoU	0.91726

Tabla 4.19: IoU por clase y mIoU promedio de cada clase.

La figura 4.23 muestra los resultados de segmentación con el mejor modelo para una imagen con presencia de maquillaje:

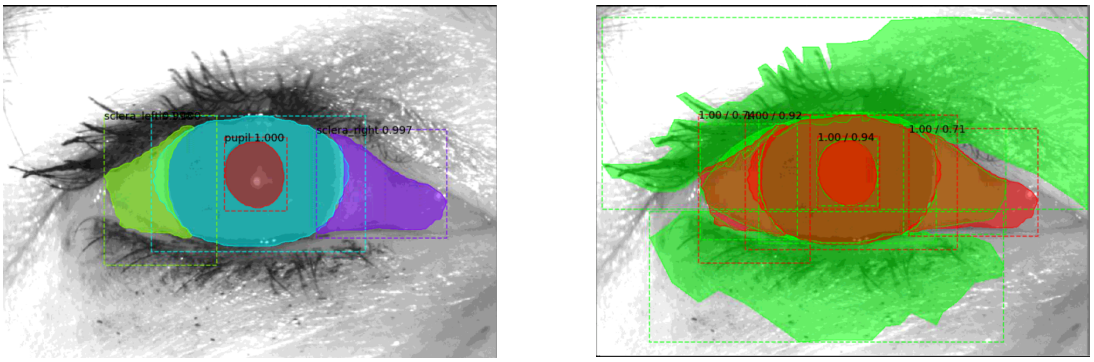


Figura 4.23: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.20:

Class	IoU
Iris	0.91553
Pupil	0.93878
Sclera right	0.71066
Sclera left	0.74048
MakeUp	0.00000
MakeUp	0.00000
mIoU	0.55091

Tabla 4.20: IoU por clase y mIoU promedio de cada clase. Se reporta dos veces la clase maquillaje puesto que existen dos áreas del ojo que presentan esta clase.

1.3.2. Prueba 02

Dada la prueba anterior, se busca incrementar el numero de épocas y disminuyendo el learning rate para buscar un mínimo óptimo, por lo tanto los hiper-parámetros utilizados para esta prueba fueron:

Parameter	**Value**
Weights	COCO
Images per GPU	2
Image Min Dimension	800
Image Max Dimension	1024
Learning Rate	0.0001
Weight Decay	0.0001
Detection Min Confidence	0.9
Epochs	100
Batch Size	2
Steps	80
Layers	heads
Data Augmentation	No

Tabla 4.21: Los steps corresponden al calculo: $steps = [TI/(BatchSize)] * 2$, donde TI corresponde a las 80 imágenes de entrenamiento usadas.

Obteniendo los siguientes resultados de mIoU por época:

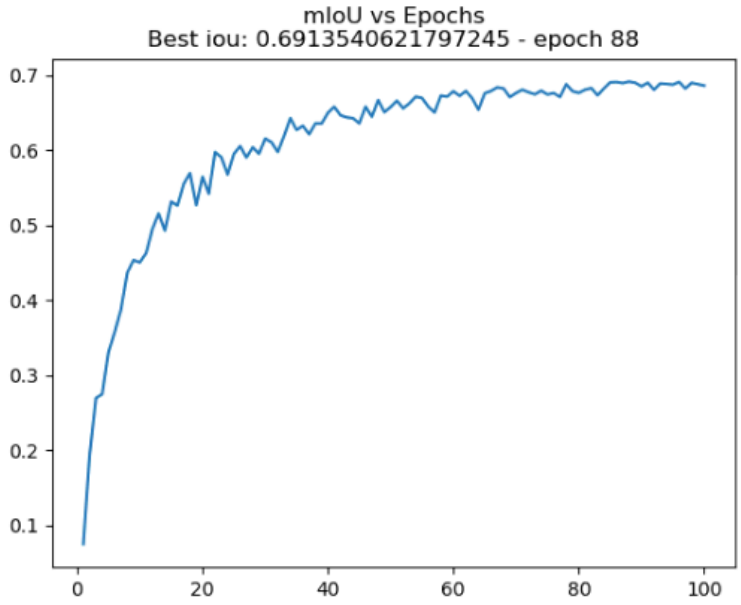


Figura 4.24: Resultados de mIoU por época de la prueba 02 con ResNet101 por backbone. El modelo con mejor rendimiento fue el de la época 88 con un mIoU de 0.69135 para todo el set de validación. El eje-X corresponde a las épocas, mientras que el eje-Y corresponde al mIoU.

La figura 4.25 muestra los resultados de segmentación con el mejor modelo para la presente prueba:

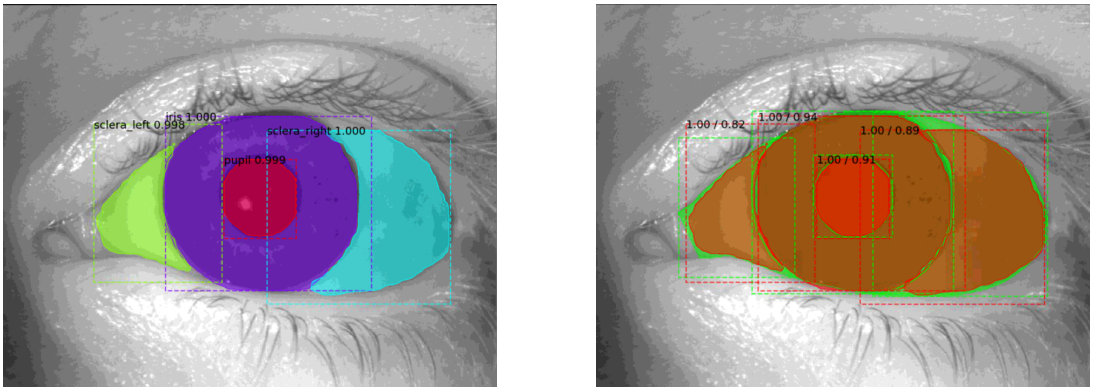


Figura 4.25: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.22:

Class	IoU
Iris	0.94271
Pupil	0.91028
Sclera right	0.89238
Sclera left	0.82434
mIoU	0.89243

Tabla 4.22: IoU por clase y mIoU promedio de cada clase.

La figura 4.26 muestra los resultados de segmentación con el mejor modelo para una imagen con presencia de maquillaje:

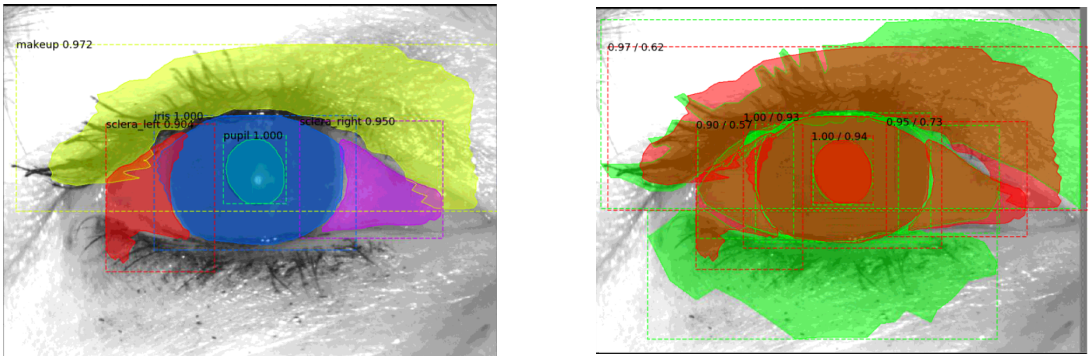


Figura 4.26: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.23:

Class	IoU
Iris	0.92622
Pupil	0.93794
Sclera right	0.72821
Sclera left	0.57362
MakeUp	0.61821
MakeUp	0.00000
mIoU	0.63070

Tabla 4.23: IoU por clase y mIoU promedio de cada clase. Se reporta dos veces la clase maquillaje puesto que existen dos áreas del ojo que presentan esta clase.

1.3.3. Prueba 03

La disminución del desempeño de la red fue debido al nuevo learning rate, por lo cual se estima que incrementando las épocas mejorará el desempeño. Luego, los hiper-parámetros utilizados para esta prueba fueron:

Parameter	**Value**
Weights	COCO
Images per GPU	2
Image Min Dimension	800
Image Max Dimension	1024
Learning Rate	0.0001
Weight Decay	0.0001
Detection Min Confidence	0.9
Epochs	150
Batch Size	2
Steps	80
Layers	heads
Data Augmentation	No

Tabla 4.24: Los steps corresponden al calculo: $steps = [TI/(BatchSize)] * 2$, donde TI corresponde a las 80 imágenes de entrenamiento usadas.

Obteniendo los siguientes resultados de mIoU por época:

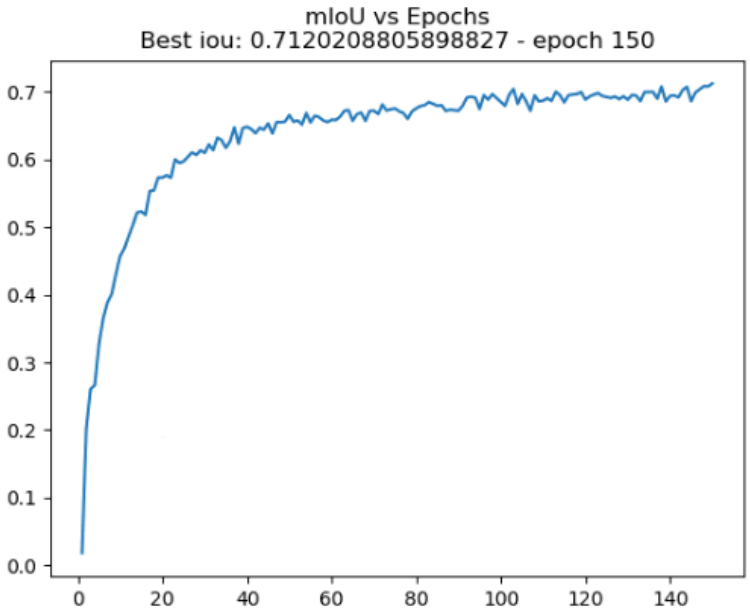


Figura 4.27: Resultados de mIoU por época de la prueba 03 con ResNet101 por backbone. El modelo con mejor rendimiento fue el de la época 150 con un mIoU de 0.71202 para todo el set de validación. El eje-X corresponde a las épocas, mientras que el eje-Y corresponde al mIoU.

La figura 4.28 muestra los resultados de segmentación con el mejor modelo para la presente prueba:

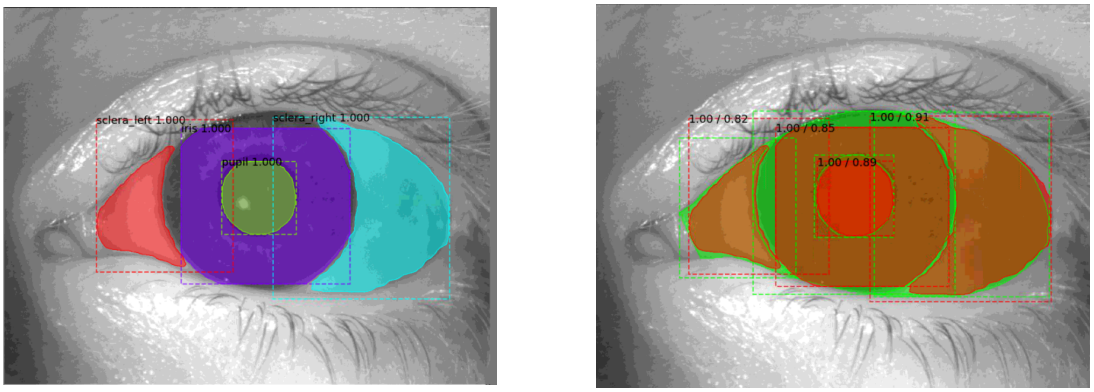


Figura 4.28: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.25:

Class	IoU
Iris	0.84806
Pupil	0.88532
Sclera right	0.90870
Sclera left	0.82446
mIoU	0.86663

Tabla 4.25: IoU por clase y mIoU promedio de cada clase.

La figura 4.29 muestra los resultados de segmentación con el mejor modelo para una imagen con presencia de maquillaje:

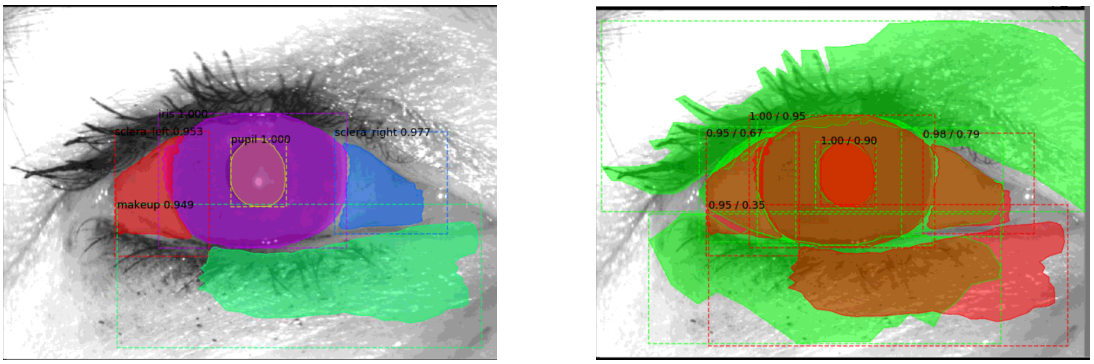


Figura 4.29: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.26:

Class	IoU
Iris	0.94970
Pupil	0.89726
Sclera right	0.78705
Sclera left	0.67281
MakeUp	0.35152
MakeUp	0.00000
mIoU	0.60972

Tabla 4.26: IoU por clase y mIoU promedio de cada clase. Se reporta dos veces la clase maquillaje puesto que existen dos áreas del ojo que presentan esta clase.

1.3.4. Prueba 04

Según aumentaron las épocas, se presencia un incremento del desempeño final del modelo, por lo cual para esta prueba fueron aun más las épocas para el modelo. Los hiper-parámetros utilizados para esta prueba fueron:

Parameter	Value
Weights	COCO
Images per GPU	2
Image Min Dimension	800
Image Max Dimension	1024
Learning Rate	0.0001
Weight Decay	0.0001
Detection Min Confidence	0.9
Epochs	200
Batch Size	2
Steps	80
Layers	heads
Data Augmentation	No

Tabla 4.27: Los steps corresponden al calculo: $steps = [TI/(BatchSize)] * 2$, donde TI corresponde a las 80 imágenes de entrenamiento usadas.

Obteniendo los siguientes resultados de mIoU por época:

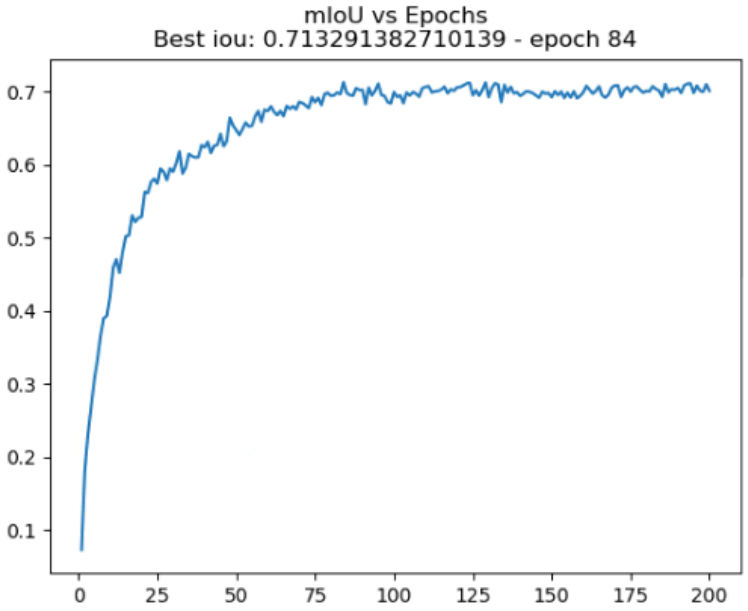


Figura 4.30: Resultados de mIoU por época de la prueba 04 con ResNet101 por backbone. El modelo con mejor rendimiento fue el de la época 84 con un mIoU de 0.71329 para todo el set de validación. El eje-X corresponde a las épocas, mientras que el eje-Y corresponde al mIoU.

La figura 4.31 muestra los resultados de segmentación con el mejor modelo para la presente prueba:

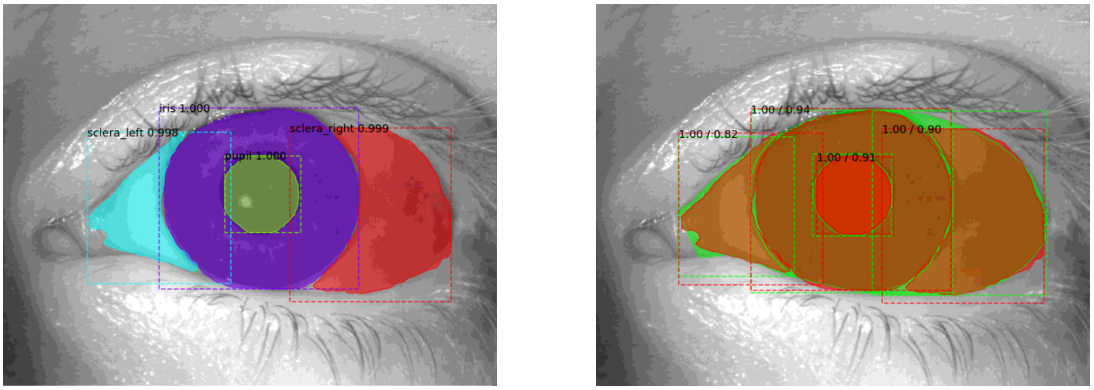


Figura 4.31: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.28:

Class	IoU
Iris	0.94488
Pupil	0.91147
Sclera right	0.89555
Sclera left	0.81988
mIoU	0.89294

Tabla 4.28: IoU por clase y mIoU promedio de cada clase.

La figura 4.32 muestra los resultados de segmentación con el mejor modelo para una imagen con presencia de maquillaje:

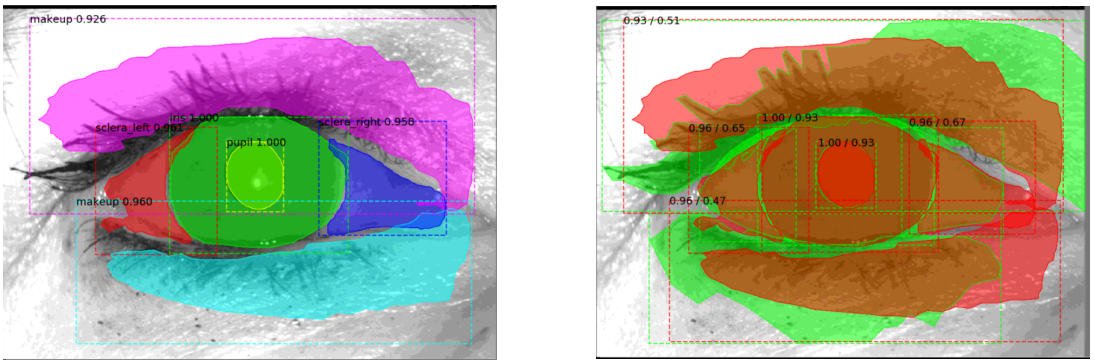


Figura 4.32: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.29:

Class	IoU
Iris	0.92874
Pupil	0.92923
Sclera right	0.66976
Sclera left	0.65328
MakeUp	0.46871
MakeUp	0.50756
mIoU	0.69288

Tabla 4.29: IoU por clase y mIoU promedio de cada clase. Se reporta dos veces la clase maquillaje puesto que existen dos áreas del ojo que presentan esta clase.

1.3.5. Prueba 05

Según se muestra en el gráfico de la prueba anterior, no se presencia progreso para el desempeño del modelo conforme aumentan las épocas, por lo cual se decide incrementar en 10 unidades el learning rate y disminuyendo a 50 las épocas con motivo de respaldar el número escogido para el learning rate. Dado lo antes dicho, los hiper-parámetros utilizados para esta prueba fueron:

Parameter	**Value**
Weights	COCO
Images per GPU	2
Image Min Dimension	800
Image Max Dimension	1024
Learning Rate	0.001
Weight Decay	0.0001
Detection Min Confidence	0.9
Epochs	50
Batch Size	2
Steps	80
Layers	heads
Data Augmentation	No

Tabla 4.30: Los steps corresponden al calculo: $steps = [TI/(BatchSize)] * 2$, donde TI corresponde a las 80 imágenes de entrenamiento usadas.

Obteniendo los siguientes resultados de mIoU por época:

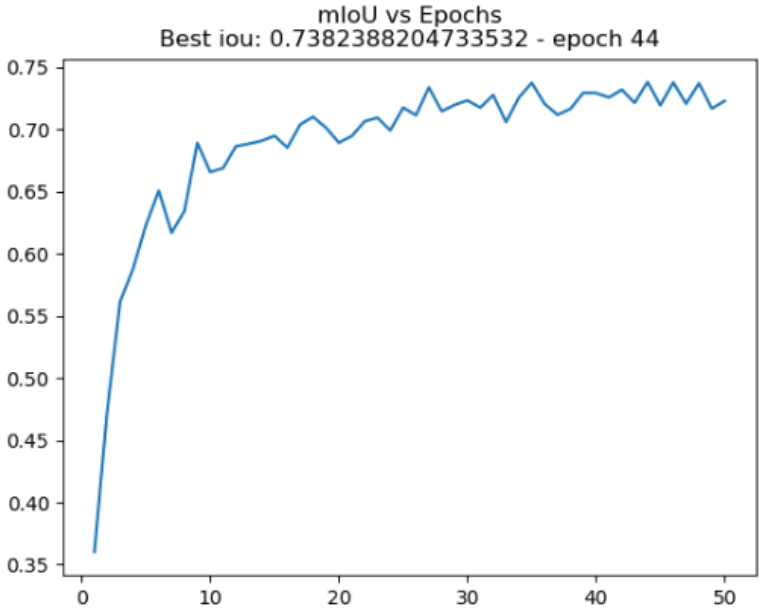


Figura 4.33: Resultados de mIoU por época de la prueba 05 con ResNet101 por backbone. El modelo con mejor rendimiento fue el de la época 44 con un mIoU de 0.73823 para todo el set de validación. El eje-X corresponde a las épocas, mientras que el eje-Y corresponde al mIoU.

La figura 4.34 muestra los resultados de segmentación con el mejor modelo para la presente prueba:

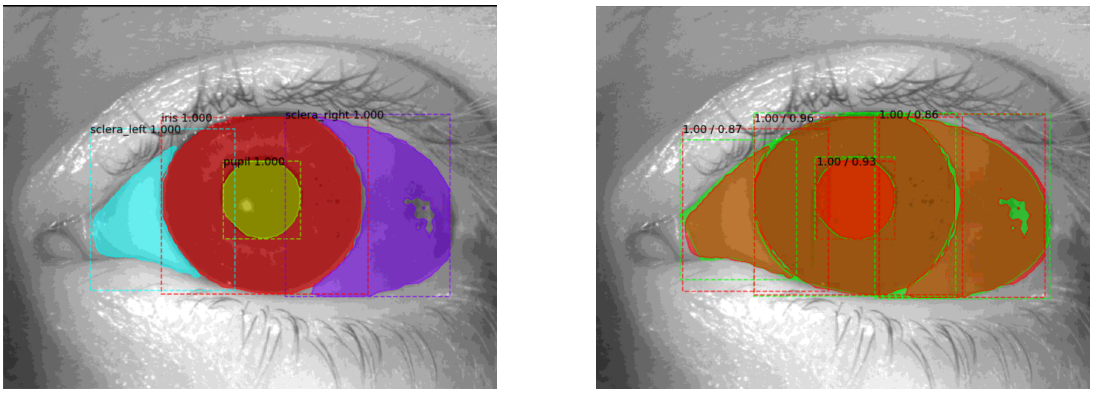


Figura 4.34: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.31:

Class	IoU
Iris	0.96133
Pupil	0.93254
Sclera right	0.86136
Sclera left	0.86847
mIoU	0.90592

Tabla 4.31: IoU por clase y mIoU promedio de cada clase.

La figura 4.35 muestra los resultados de segmentación con el mejor modelo para una imagen con presencia de maquillaje:

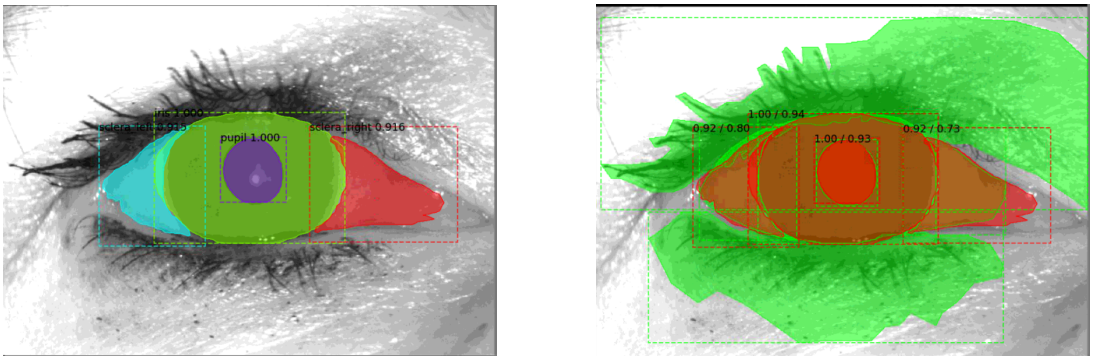


Figura 4.35: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.32:

Class	IoU
Iris	0.93503
Pupil	0.92833
Sclera right	0.72518
Sclera left	0.79684
MakeUp	0.00000
MakeUp	0.00000
mIoU	0.56423

Tabla 4.32: IoU por clase y mIoU promedio de cada clase. Se reporta dos veces la clase maquillaje puesto que existen dos áreas del ojo que presentan esta clase.

1.3.6. Prueba 06

El desempeño del modelo vuelve a remontar por sobre 0.7, lo cual confirma que un learning rate con valor de 0.001 es el optimo para este caso. Se realizan entonces pruebas con los pesos de IMAGENET y evaluar sus resultados. Los hiper-parámetros utilizados para esta prueba fueron:

Parameter	Value
Weights	IMAGENET
Images per GPU	2
Image Min Dimension	800
Image Max Dimension	1024
Learning Rate	0.001
Weight Decay	0.0001
Detection Min Confidence	0.9
Epochs	50
Batch Size	2
Steps	80
Layers	heads
Data Augmentation	No

Tabla 4.33: Los steps corresponden al calculo: $steps = [TI/(BatchSize)] * 2$, donde TI corresponde a las 80 imágenes de entrenamiento usadas.

Obteniendo los siguientes resultados de mIoU por época:

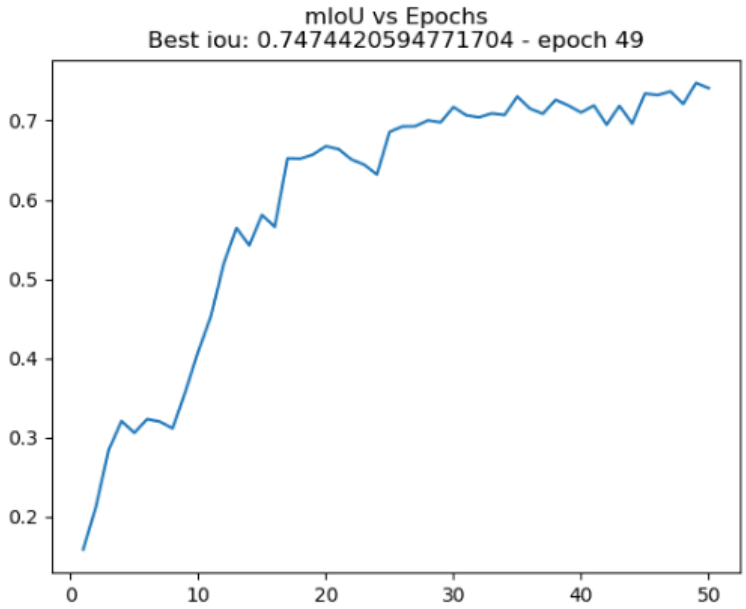


Figura 4.36: Resultados de mIoU por época de la prueba 06 con ResNet101 por backbone. El modelo con mejor rendimiento fue el de la época 49 con un mIoU de 0.74744 para todo el set de validación. El eje-X corresponde a las épocas, mientras que el eje-Y corresponde al mIoU.

La figura 4.37 muestra los resultados de segmentación con el mejor modelo para la presente prueba:

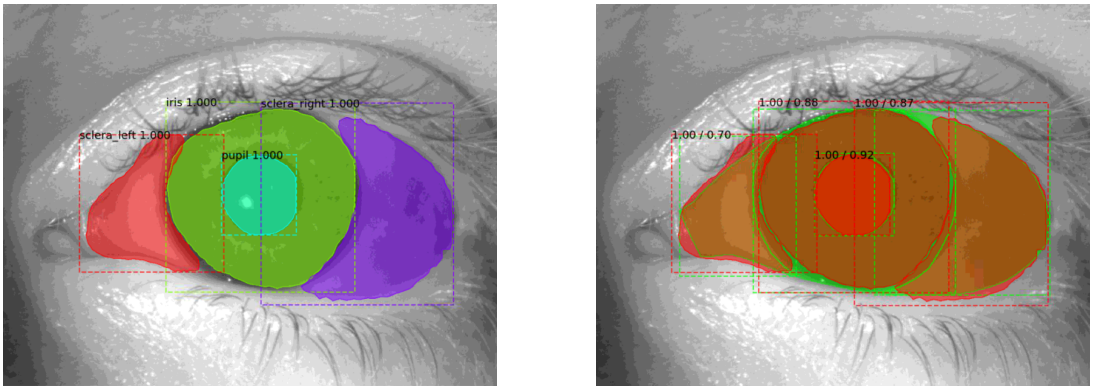


Figura 4.37: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.34:

Class	IoU
Iris	0.87923
Pupil	0.91791
Sclera right	0.87063
Sclera left	0.69757
mIoU	0.84134

Tabla 4.34: IoU por clase y mIoU promedio de cada clase.

La figura 4.38 muestra los resultados de segmentación con el mejor modelo para una imagen con presencia de maquillaje:

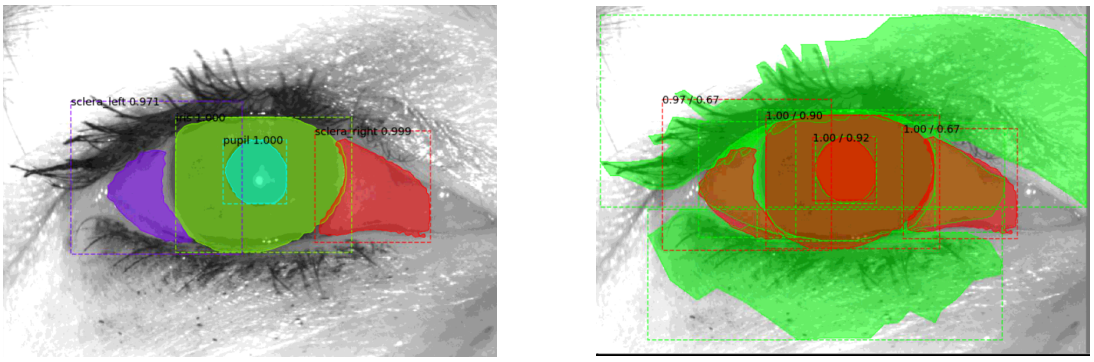


Figura 4.38: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.35:

Class	IoU
Iris	0.89579
Pupil	0.92440
Sclera right	0.66677
Sclera left	0.67267
MakeUp	0.00000
MakeUp	0.00000
mIoU	0.52662

Tabla 4.35: IoU por clase y mIoU promedio de cada clase. Se reporta dos veces la clase maquillaje puesto que existen dos áreas del ojo que presentan esta clase.

1.3.7. Prueba 07

Dada la obtención de mejores resultados que con los pesos de COCO, es realizado una segunda prueba con los pesos de IMAGENET, esta vez incrementando la cantidad de épocas para el modelo. Los hiper-parámetros utilizados para esta prueba fueron:

Parameter	**Value**
Weights	IMAGENET
Images per GPU	2
Image Min Dimension	800
Image Max Dimension	1024
Learning Rate	0.001
Weight Decay	0.0001
Detection Min Confidence	0.9
Epochs	100
Batch Size	2
Steps	80
Layers	heads
Data Augmentation	No

Tabla 4.36: Los steps corresponden al calculo: $steps = [TI/(BatchSize)] * 2$, donde TI corresponde a las 80 imágenes de entrenamiento usadas.

Obteniendo los siguientes resultados de mIoU por época:

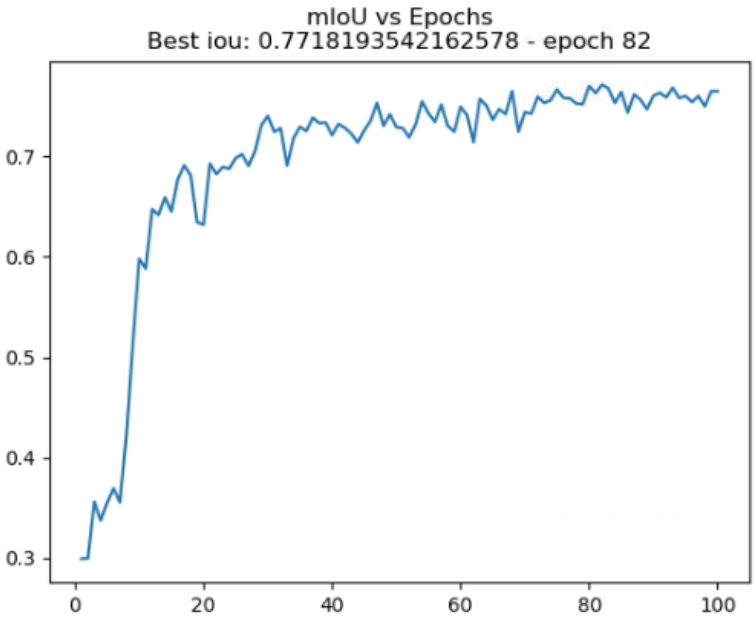


Figura 4.39: Resultados de mIoU por época de la prueba 07 con ResNet101 por backbone. El modelo con mejor rendimiento fue el de la época 82 con un mIoU de 0.77181 para todo el set de validación. El eje-X corresponde a las épocas, mientras que el eje-Y corresponde al mIoU.

La figura 4.40 muestra los resultados de segmentación con el mejor modelo para la presente prueba:

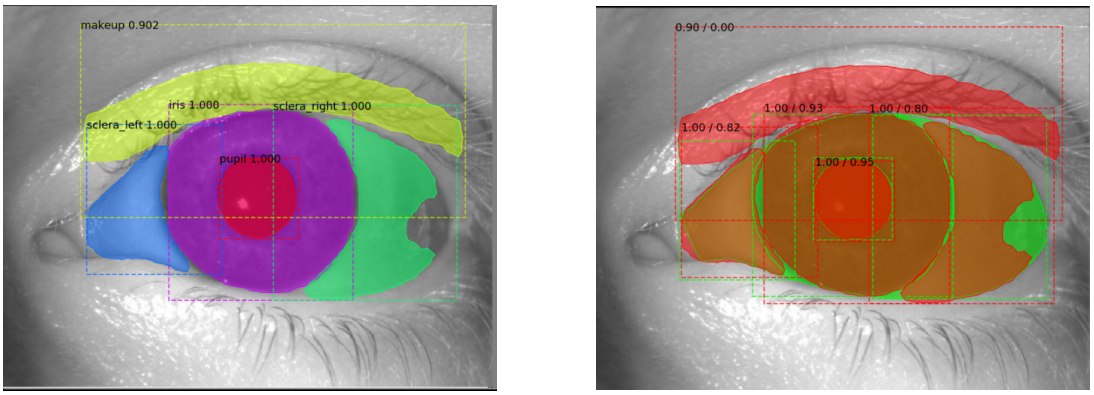


Figura 4.40: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.37:

Class	IoU
Iris	0.93393
Pupil	0.94517
Sclera right	0.79808
Sclera left	0.81972
MakeUp	0.00000
mIoU	0.69938

Tabla 4.37: IoU por clase y mIoU promedio de cada clase. Fue detectado maquillaje donde no había presencia del mismo. Ante este falso positivo, su IoU es 0.0.

La figura 4.41 muestra los resultados de segmentación con el mejor modelo para una imagen con presencia de maquillaje:

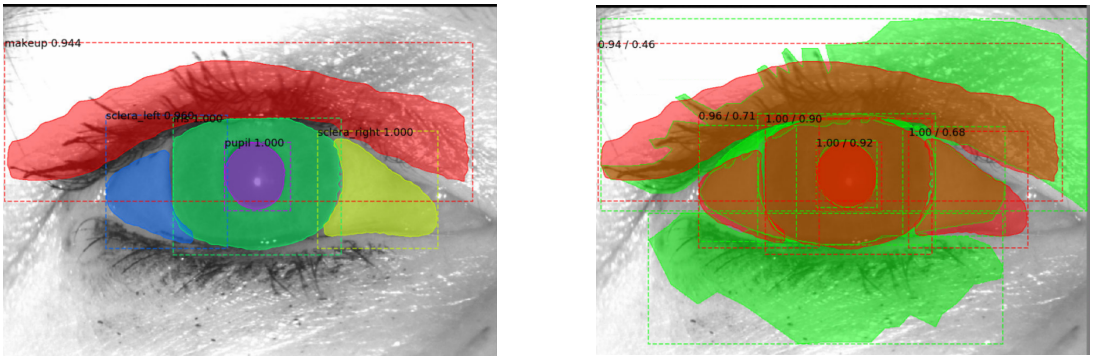


Figura 4.41: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.38:

Class	IoU
Iris	0.90340
Pupil	0.92343
Sclera right	0.67705
Sclera left	0.70998
MakeUp	0.45706
MakeUp	0.00000
mIoU	0.61182

Tabla 4.38: IoU por clase y mIoU promedio de cada clase. Se reporta dos veces la clase maquillaje puesto que existen dos áreas del ojo que presentan esta clase.

1.3.8. Prueba 08

Hasta ahora se presentan los mejores resultados con los pesos de IMAGENET, por lo cual se incrementará el número de épocas. Los hiper-parámetros utilizados para esta prueba fueron:

Parameter	Value
Weights	IMAGENET
Images per GPU	2
Image Min Dimension	800
Image Max Dimension	1024
Learning Rate	0.001
Weight Decay	0.0001
Detection Min Confidence	0.9
Epochs	150
Batch Size	2
Steps	80
Layers	heads
Data Augmentation	No

Tabla 4.39: Los steps corresponden al calculo: $steps = [TI/(BatchSize)] * 2$, donde TI corresponde a las 80 imágenes de entrenamiento usadas.

Obteniendo los siguientes resultados de mIoU por época:

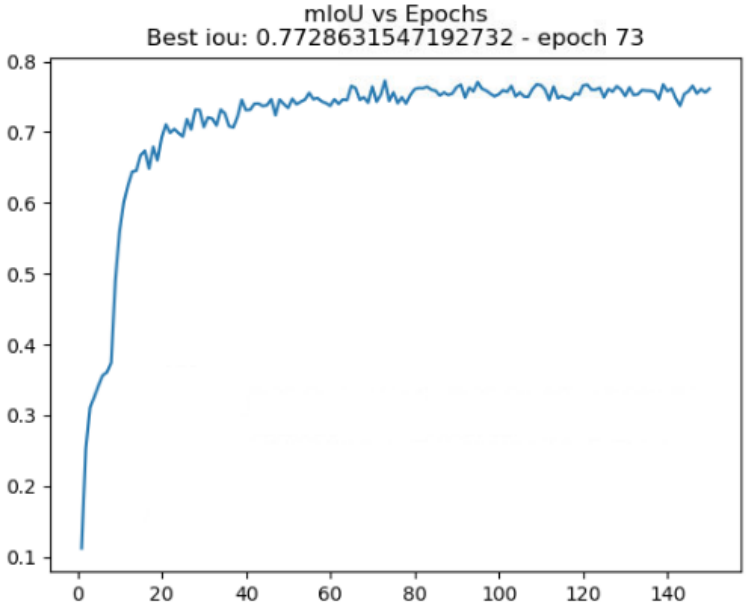


Figura 4.42: Resultados de mIoU por época de la prueba 08 con ResNet101 por backbone. El modelo con mejor rendimiento fue el de la época 73 con un mIoU de 0.77286 para todo el set de validación. El eje-X corresponde a las épocas, mientras que el eje-Y corresponde al mIoU.

La figura 4.43 muestra los resultados de segmentación con el mejor modelo para la presente prueba:

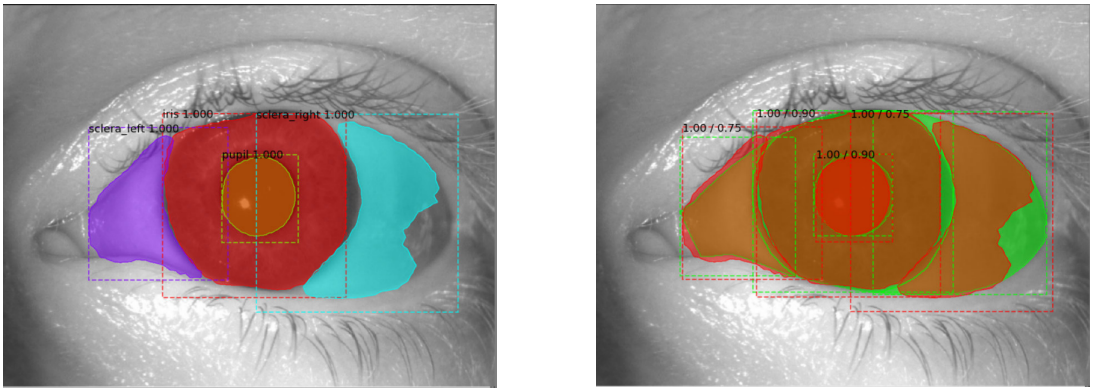


Figura 4.43: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.40:

Class	IoU
Iris	0.89648
Pupil	0.89906
Sclera right	0.75023
Sclera left	0.74639
mIoU	0.82304

Tabla 4.40: IoU por clase y mIoU promedio de cada clase.

La figura 4.44 muestra los resultados de segmentación con el mejor modelo para una imagen con presencia de maquillaje:

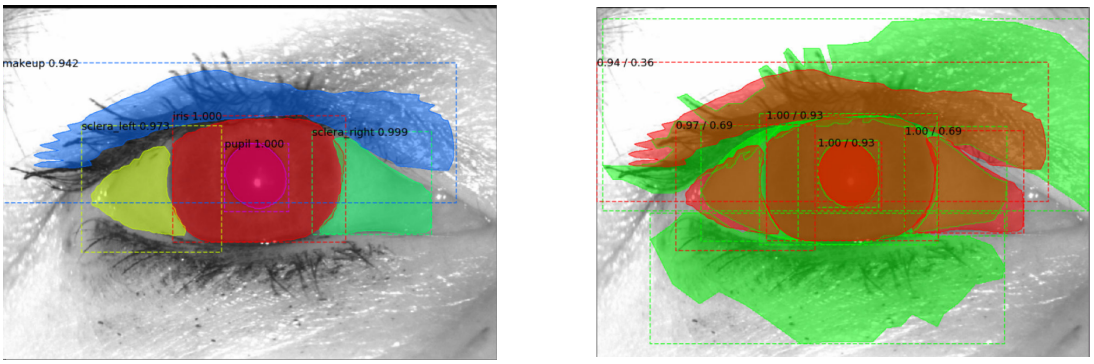


Figura 4.44: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.41:

Class	IoU
Iris	0.92599
Pupil	0.92547
Sclera right	0.69064
Sclera left	0.69405
MakeUp	0.36118
MakeUp	0.00000
mIoU	0.59955

Tabla 4.41: IoU por clase y mIoU promedio de cada clase. Se reporta dos veces la clase maquillaje puesto que existen dos áreas del ojo que presentan esta clase.

1.3.9. Prueba 09

No hubo mayor incremento del rendimiento dado el aumento de épocas para los pesos de IMAGENET. Se retoman pruebas con los pesos de COCO para comenzar a aplicar la técnica de data augmentation y aumentando la cantidad de steps. Aumentar los steps significa que la red será entrenada mas veces con las mismas imágenes, pero dado que se está aplicando data augmentation, estas imágenes serán distintas en cada iteración. Los hiper-parámetros utilizados para esta prueba fueron:

Parameter	**Value**
Weights	COCO
Images per GPU	2
Image Min Dimension	800
Image Max Dimension	1024
Learning Rate	0.001
Weight Decay	0.0001
Detection Min Confidence	0.9
Epochs	200
Batch Size	2
Steps	120
Layers	heads
Data Augmentation	Yes

Tabla 4.42: Los steps corresponden al calculo: $steps = [TI/(BatchSize)] * 3$, donde TI corresponde a las 80 imágenes de entrenamiento usadas.

Obteniendo los siguientes resultados de mIoU por época:

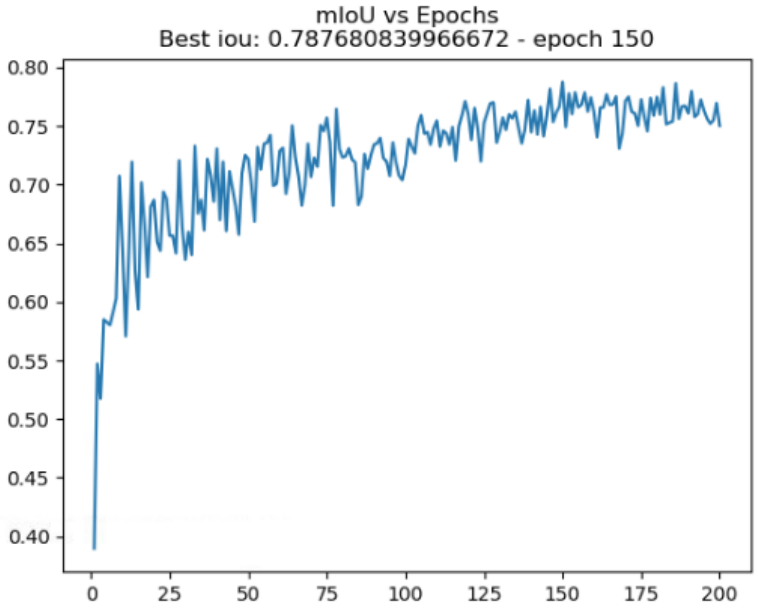


Figura 4.45: Resultados de mIoU por época de la prueba 09 con ResNet101 por backbone. El modelo con mejor rendimiento fue el de la época 150 con un mIoU de 0.78768 para todo el set de validación. El eje-X corresponde a las épocas, mientras que el eje-Y corresponde al mIoU.

La figura 4.46 muestra los resultados de segmentación con el mejor modelo para la presente prueba:

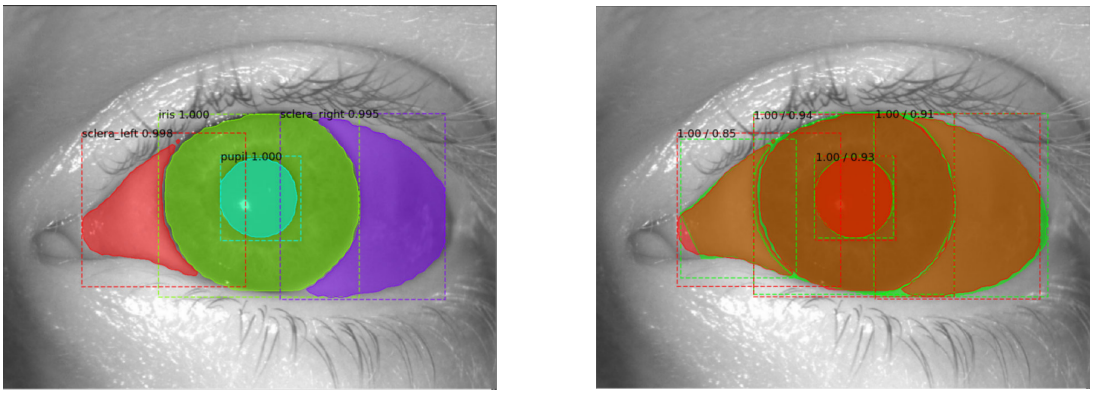


Figura 4.46: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.43:

Class	IoU
Iris	0.94218
Pupil	0.93244
Sclera right	0.91048
Sclera left	0.84582
mIoU	0.90773

Tabla 4.43: IoU por clase y mIoU promedio de cada clase.

La figura 4.47 muestra los resultados de segmentación con el mejor modelo para una imagen con presencia de maquillaje:

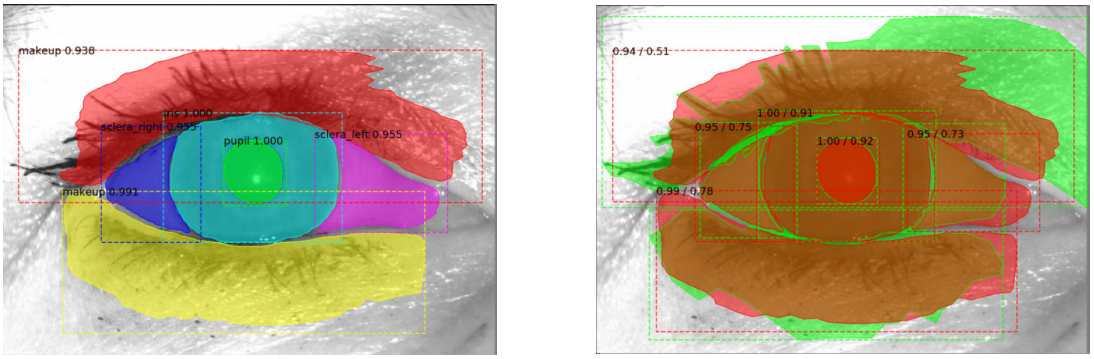


Figura 4.47: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.44:

Class	IoU
Iris	0.91170
Pupil	0.91619
Sclera right	0.74973
Sclera left	0.72554
MakeUp	0.78410
MakeUp	0.50877
mIoU	0.76600

Tabla 4.44: IoU por clase y mIoU promedio de cada clase. Se reporta dos veces la clase maquillaje puesto que existen dos áreas del ojo que presentan esta clase.

1.3.10. Prueba 10

Hubo un incremento deseable por parte del rendimiento del modelo, lo cual confirma que aplicar data augmentation resulta en un impacto positivo para el entrenamiento. Para el presente entrenamiento se reduce en 10 unidades el learning rate y se aplica un entrenamiento con alto numero de épocas tal como se muestra en la siguiente tabla:

Parameter	Value
Weights	COCO
Images per GPU	2
Image Min Dimension	800
Image Max Dimension	1024
Learning Rate	0.0001
Weight Decay	0.0001
Detection Min Confidence	0.9
Epochs	200
Batch Size	2
Steps	120
Layers	heads
Data Augmentation	Yes

Tabla 4.45: Los steps corresponden al calculo: $steps = [TI/(BatchSize)] * 3$, donde TI corresponde a las 80 imágenes de entrenamiento usadas.

Obteniendo los siguientes resultados de mIoU por época:

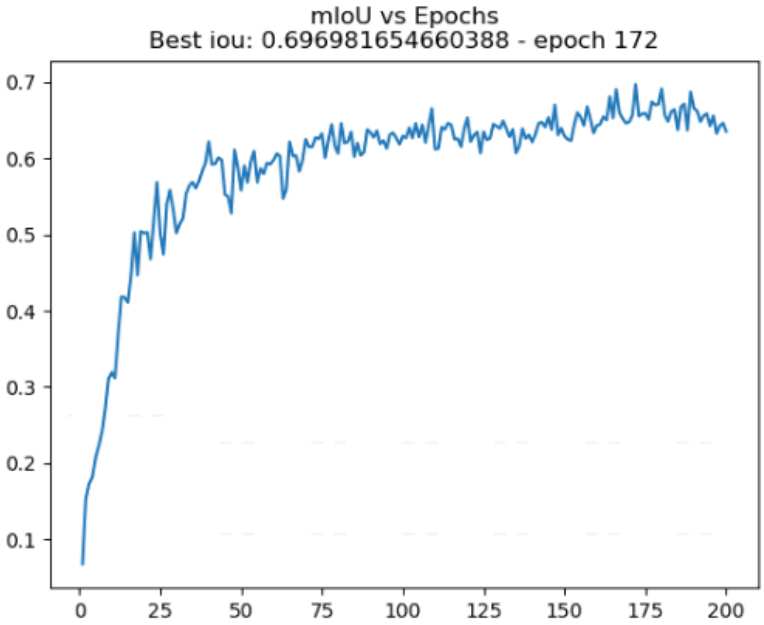


Figura 4.48: Resultados de mIoU por época de la prueba 10 con ResNet101 por backbone. El modelo con mejor rendimiento fue el de la época 172 con un mIoU de 0.69698 para todo el set de validación. El eje-X corresponde a las épocas, mientras que el eje-Y corresponde al mIoU.

La figura 4.49 muestra los resultados de segmentación con el mejor modelo para la presente prueba:

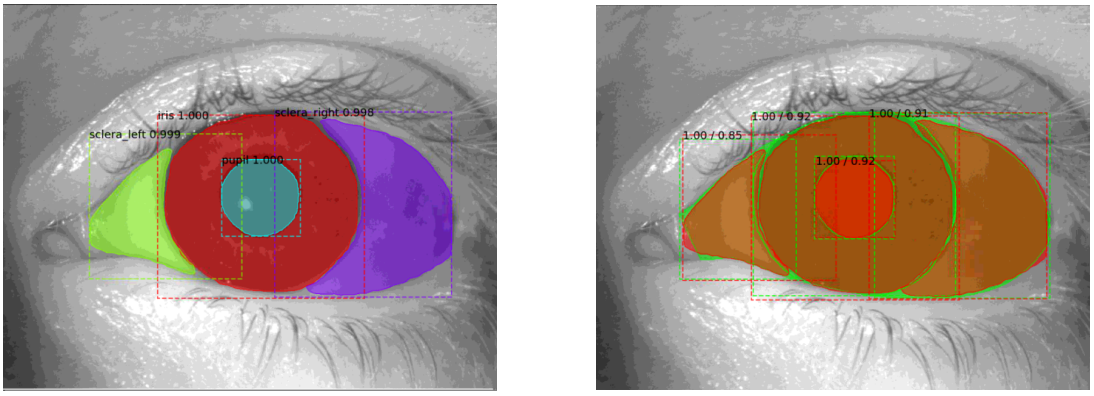


Figura 4.49: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.46:

Class	IoU
Iris	0.92192
Pupil	0.92058
Sclera right	0.91352
Sclera left	0.84760
mIoU	0.90091

Tabla 4.46: IoU por clase y mIoU promedio de cada clase.

La figura 4.50 muestra los resultados de segmentación con el mejor modelo para una imagen con presencia de maquillaje:

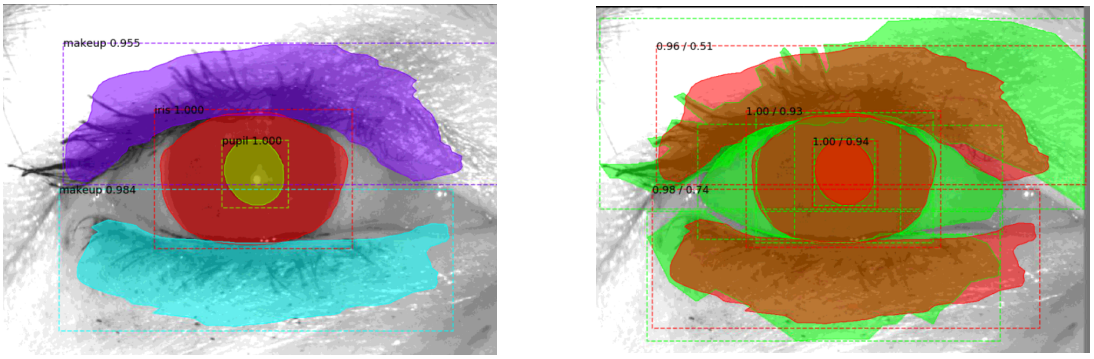


Figura 4.50: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.47:

Class	IoU
Iris	0.92912
Pupil	0.94174
Sclera right	0.00000
Sclera left	0.00000
MakeUp	0.73951
MakeUp	0.51291
mIoU	0.52054

Tabla 4.47: IoU por clase y mIoU promedio de cada clase. Se reporta dos veces la clase maquillaje puesto que existen dos áreas del ojo que presentan esta clase.

1.3.11. Prueba 11

Reducir el learning rate redujo casi en 10 puntos el desempeño final del modelo. Se estima que al incrementar la cantidad de steps se logrará un desempeño mayor. Los hiper-parámetros utilizados para esta prueba fueron:

Parameter	**Value**
Weights	COCO
Images per GPU	2
Image Min Dimension	800
Image Max Dimension	1024
Learning Rate	0.0001
Weight Decay	0.0001
Detection Min Confidence	0.9
Epochs	100
Batch Size	2
Steps	400
Layers	heads
Data Augmentation	Yes

Tabla 4.48: Los steps corresponden al calculo: $steps = [TI/(BatchSize)] * 10$, donde TI corresponde a las 80 imágenes de entrenamiento usadas.

Obteniendo los siguientes resultados de mIoU por época:

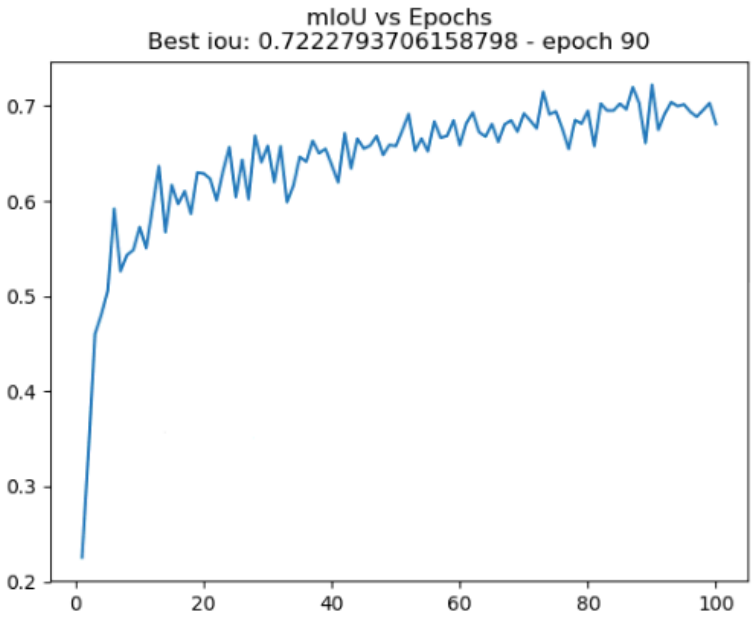


Figura 4.51: Resultados de mIoU por época de la prueba 11 con ResNet101 por backbone. El modelo con mejor rendimiento fue el de la época 90 con un mIoU de 0.72227 para todo el set de validación. El eje-X corresponde a las épocas, mientras que el eje-Y corresponde al mIoU.

La figura 4.52 muestra los resultados de segmentación con el mejor modelo para la presente prueba:

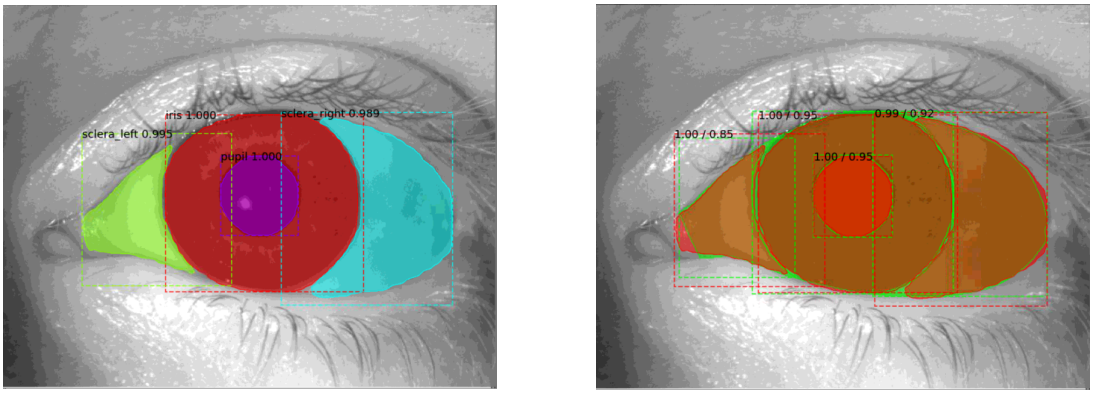


Figura 4.52: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.49:

Class	IoU
Iris	0.95032
Pupil	0.95324
Sclera right	0.92062
Sclera left	0.85498
mIoU	0.91979

Tabla 4.49: IoU por clase y mIoU promedio de cada clase.

La figura 4.53 muestra los resultados de segmentación con el mejor modelo para una imagen con presencia de maquillaje:

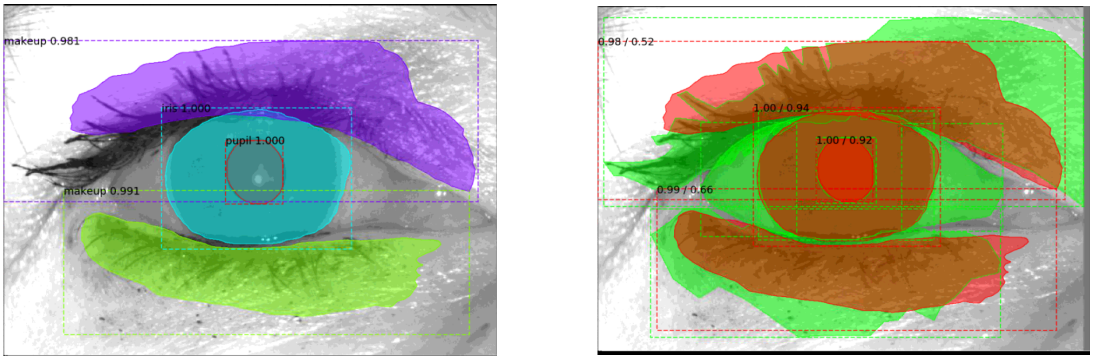


Figura 4.53: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.50:

Class	IoU
Iris	0.94389
Pupil	0.92163
Sclera right	0.00000
Sclera left	0.00000
MakeUp	0.65799
MakeUp	0.51787
mIoU	0.50689

Tabla 4.50: IoU por clase y mIoU promedio de cada clase. Se reporta dos veces la clase maquillaje puesto que existen dos áreas del ojo que presentan esta clase.

1.3.12. Prueba 12

Se aprecia un incremento del desempeño del modelo, pero es menor que la prueba 09, por lo cual se vuelve a aumentar el learning rate 10 unidades. Además, se mantiene el numero de steps. Los hiper-parámetros utilizados para esta prueba fueron:

Parameter	Value
Weights	COCO
Images per GPU	2
Image Min Dimension	800
Image Max Dimension	1024
Learning Rate	0.001
Weight Decay	0.0001
Detection Min Confidence	0.9
Epochs	100
Batch Size	2
Steps	400
Layers	heads
Data Augmentation	Yes

Tabla 4.51: Los steps corresponden al calculo: $steps = [TI/(BatchSize)] * 10$, donde TI corresponde a las 80 imágenes de entrenamiento usadas.

Obteniendo los siguientes resultados de mIoU por época:

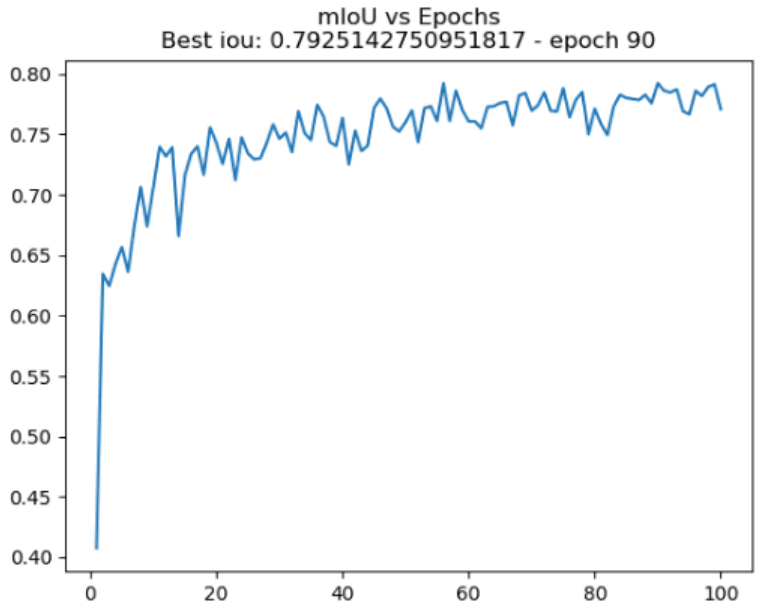


Figura 4.54: Resultados de mIoU por época de la prueba 12 con ResNet101 por backbone. El modelo con mejor rendimiento fue el de la época 90 con un mIoU de 0.79251 para todo el set de validación. El eje-X corresponde a las épocas, mientras que el eje-Y corresponde al mIoU.

La figura 4.55 muestra los resultados de segmentación con el mejor modelo para la presente prueba:

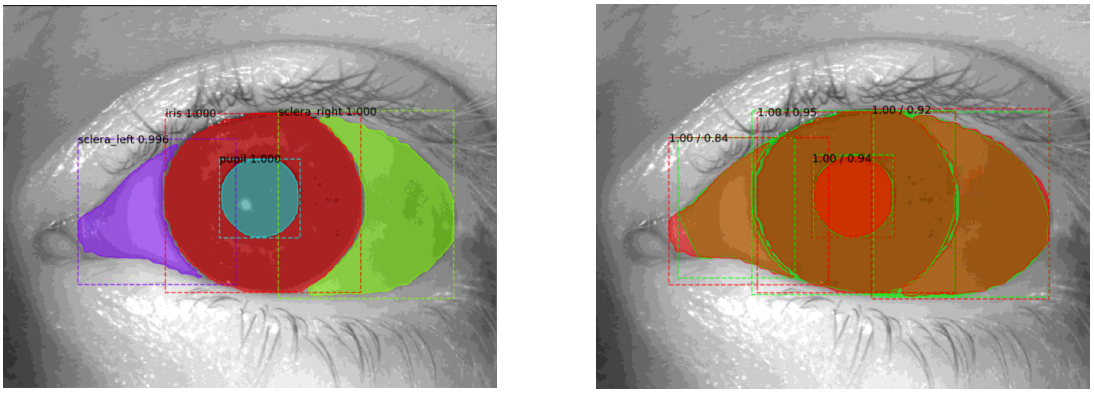


Figura 4.55: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.52:

Class	IoU
Iris	0.95431
Pupil	0.94453
Sclera right	0.91633
Sclera left	0.84199
mIoU	0.91429

Tabla 4.52: IoU por clase y mIoU promedio de cada clase.

La figura 4.56 muestra los resultados de segmentación con el mejor modelo para una imagen con presencia de maquillaje:

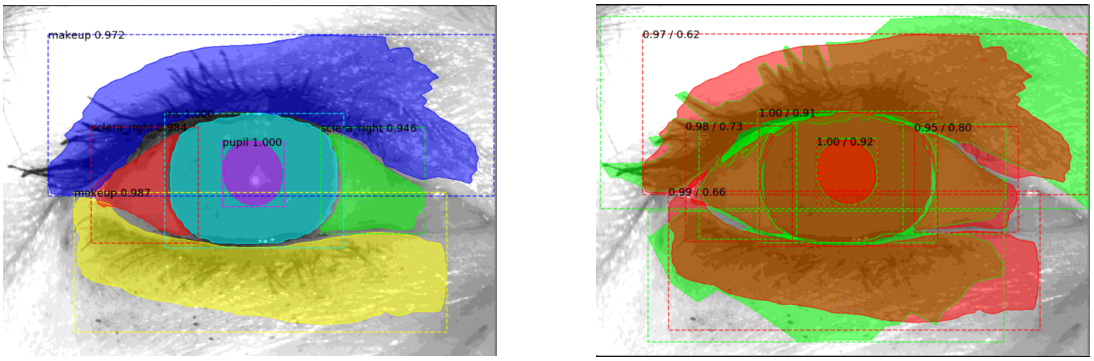


Figura 4.56: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.53:

Class	IoU
Iris	0.91188
Pupil	0.91858
Sclera right	0.73164
Sclera right	0.79811
Sclera left	0.00000
MakeUp	0.66392
MakeUp	0.61519
mIoU	0.66276

Tabla 4.53: IoU por clase y mIoU promedio de cada clase. Se reporta dos veces la clase maquillaje puesto que existen dos áreas del ojo que presentan esta clase. También se reporta dos veces la clase esclera izquierda mientras que la clase esclera derecha no fue detectada.

1.3.13. Prueba 13

Incrementar la cantidad de steps y reducir el learning rate incremento aún más el desempeño final del modelo. Se procede a aumentar las épocas con los mismos hiperparámetros utilizados por la prueba anterior. Los hiper-parámetros utilizados para esta prueba fueron:

Parameter	**Value**
Weights	COCO
Images per GPU	2
Image Min Dimension	800
Image Max Dimension	1024
Learning Rate	0.001
Weight Decay	0.0001
Detection Min Confidence	0.9
Epochs	150
Batch Size	2
Steps	400
Layers	heads
Data Augmentation	Yes

Tabla 4.54: Los steps corresponden al calculo: $steps = [TI/(BatchSize)] * 10$, donde TI corresponde a las 80 imágenes de entrenamiento usadas.

Obteniendo los siguientes resultados de mIoU por época:

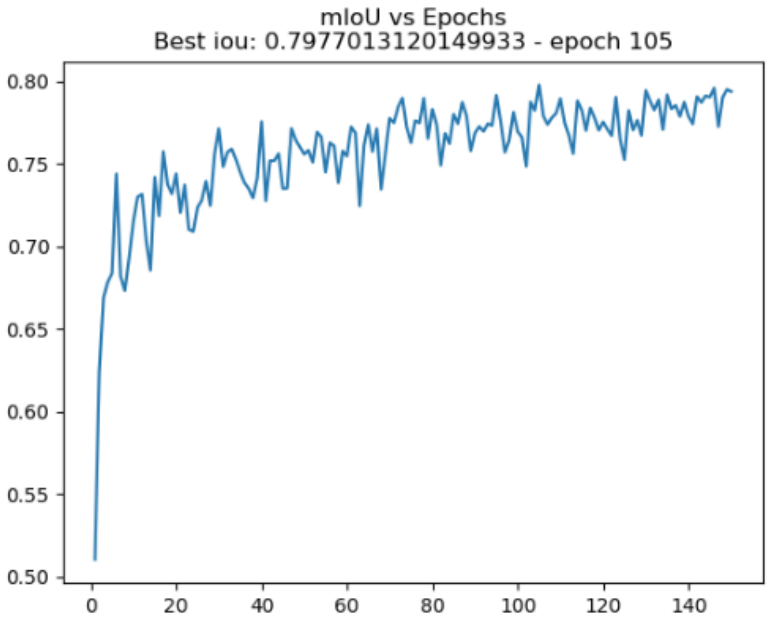


Figura 4.57: Resultados de mIoU por época de la prueba 13 con ResNet101 por backbone. El modelo con mejor rendimiento fue el de la época 105 con un mIoU de 0.79770 para todo el set de validación. El eje-X corresponde a las épocas, mientras que el eje-Y corresponde al mIoU.

La figura 4.58 muestra los resultados de segmentación con el mejor modelo para la presente prueba:

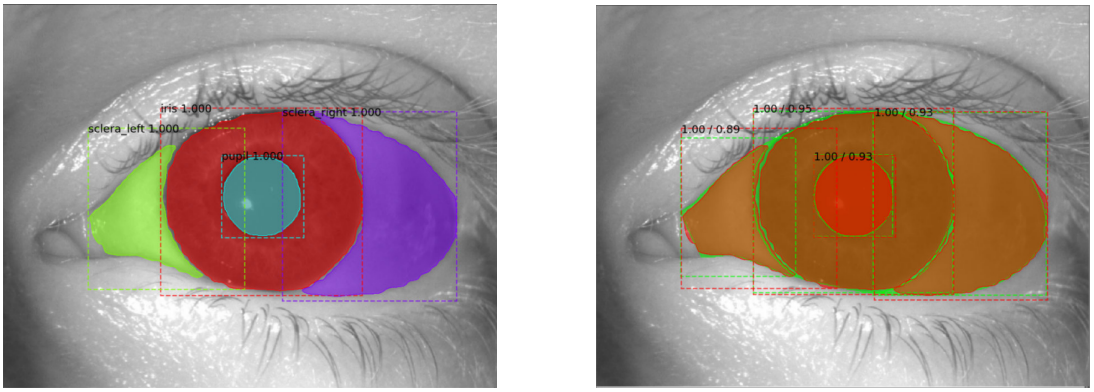


Figura 4.58: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.55:

Class	IoU
Iris	0.94567
Pupil	0.93192
Sclera right	0.93400
Sclera left	0.88619
mIoU	0.92444

Tabla 4.55: IoU por clase y mIoU promedio de cada clase.

La figura 4.59 muestra los resultados de segmentación con el mejor modelo para una imagen con presencia de maquillaje:

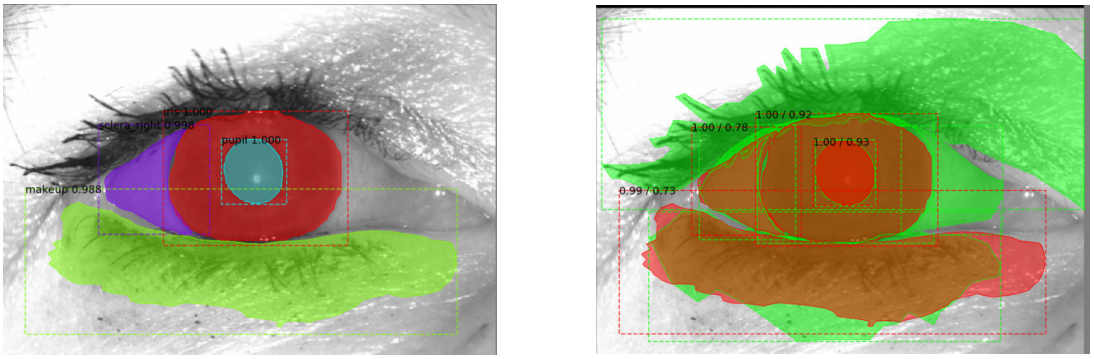


Figura 4.59: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.56:

Class	IoU
Iris	0.92027
Pupil	0.93356
Sclera right	0.77505
Sclera left	0.00000
MakeUp	0.72796
MakeUp	0.00000
mIoU	0.55947

Tabla 4.56: IoU por clase y mIoU promedio de cada clase. Se reporta dos veces la clase maquillaje puesto que existen dos áreas del ojo que presentan esta clase. Se reporta además que la clase esclera derecha no fue detectada.

1.3.14. Prueba 14

Al aumentar las épocas se aprecia un incremento menor en el rendimiento del modelo, ante lo cual para la presente prueba se reducen las épocas y se aumentan aún más los steps. Los hiper-parámetros utilizados para esta prueba fueron:

Parameter	Value
Weights	COCO
Images per GPU	2
Image Min Dimension	800
Image Max Dimension	1024
Learning Rate	0.001
Weight Decay	0.0001
Detection Min Confidence	0.9
Epochs	100
Batch Size	2
Steps	800
Layers	heads
Data Augmentation	Yes

Tabla 4.57: Los steps corresponden al calculo: $steps = [TI/(BatchSize)] * 20$, donde TI corresponde a las 80 imágenes de entrenamiento usadas.

Obteniendo los siguientes resultados de mIoU por época:

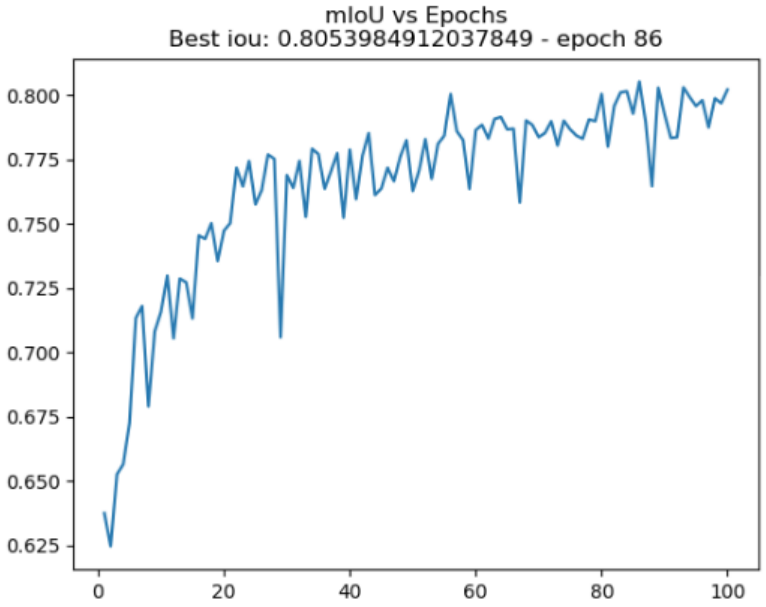


Figura 4.60: Resultados de mIoU por época de la prueba 14 con ResNet101 por backbone. El modelo con mejor rendimiento fue el de la época 86 con un mIoU de 0.80539 para todo el set de validación. El eje-X corresponde a las épocas, mientras que el eje-Y corresponde al mIoU.

La figura 4.61 muestra los resultados de segmentación con el mejor modelo para la presente prueba:

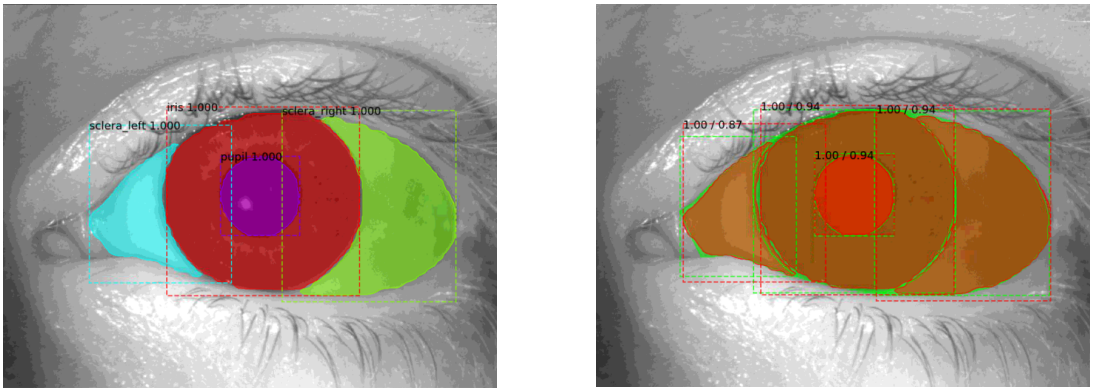


Figura 4.61: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.58:

Class	IoU
Iris	0.94079
Pupil	0.94174
Sclera right	0.94303
Sclera left	0.86957
mIoU	0.92378

Tabla 4.58: IoU por clase y mIoU promedio de cada clase.

La figura 4.62 muestra los resultados de segmentación con el mejor modelo para una imagen con presencia de maquillaje:

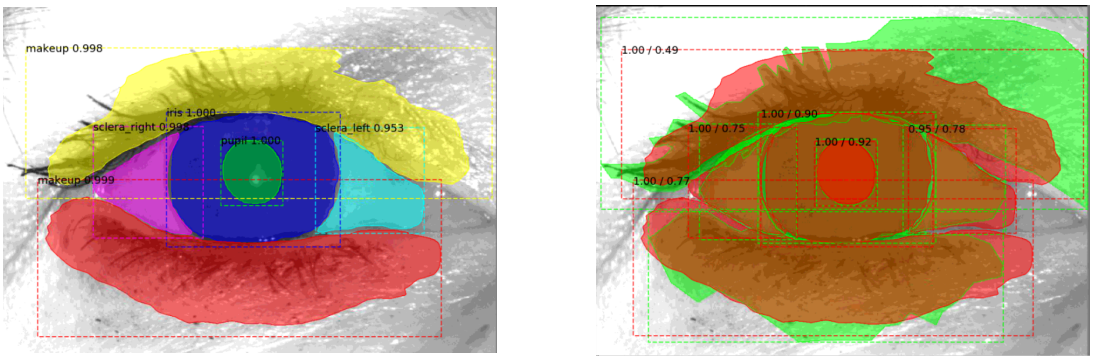


Figura 4.62: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.59:

Class	IoU
Iris	0.90401
Pupil	0.91741
Sclera right	0.74831
Sclera left	0.78313
MakeUp	0.77371
MakeUp	0.49083
mIoU	0.76957

Tabla 4.59: IoU por clase y mIoU promedio de cada clase. Se reporta dos veces la clase maquillaje puesto que existen dos áreas del ojo que presentan esta clase. Se reporta además que la clase esclera derecha no fue detectada.

1.3.15. Prueba 15

Gracias al aumento de steps el desempeño final del modelo logró superar los 0.8 de mIoU, con lo cual se procede a realizar pruebas con un gran número de épocas, un gran número de steps y reentrenando las 3 primeras capas del backbone. Los hiper-parámetros utilizados para esta prueba fueron:

Parameter	**Value**
Weights	COCO
Images per GPU	2
Image Min Dimension	800
Image Max Dimension	1024
Learning Rate	0.001
Weight Decay	0.0001
Detection Min Confidence	0.9
Epochs	200
Batch Size	2
Steps	800
Layers	3+
Data Augmentation	Yes

Tabla 4.60: Los steps corresponden al calculo: $steps = [TI/(BatchSize)] * 20$, donde TI corresponde a las 80 imágenes de entrenamiento usadas.

Obteniendo los siguientes resultados de mIoU por época:

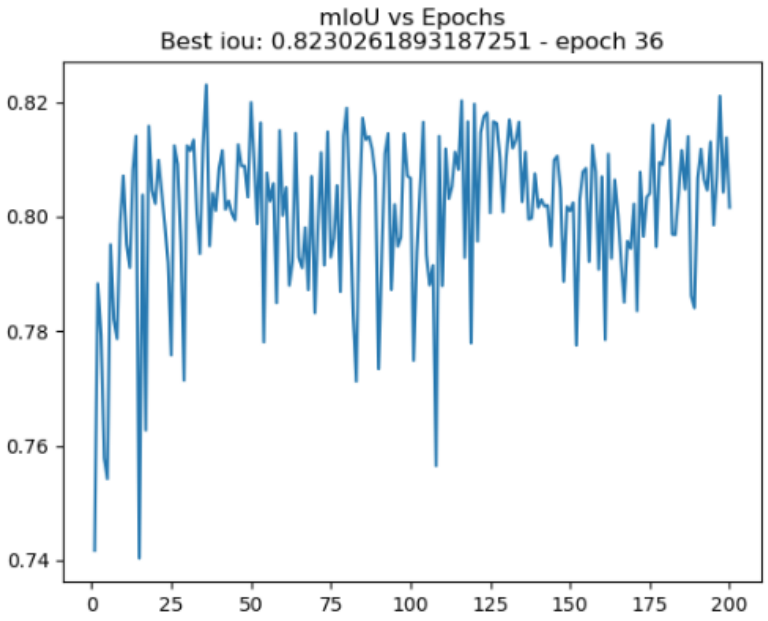


Figura 4.63: Resultados de mIoU por época de la prueba 15 con ResNet101 por backbone. El modelo con mejor rendimiento fue el de la época 36 con un mIoU de 0.82302 para todo el set de validación. El eje-X corresponde a las épocas, mientras que el eje-Y corresponde al mIoU.

La figura 4.64 muestra los resultados de segmentación con el mejor modelo para la presente prueba:

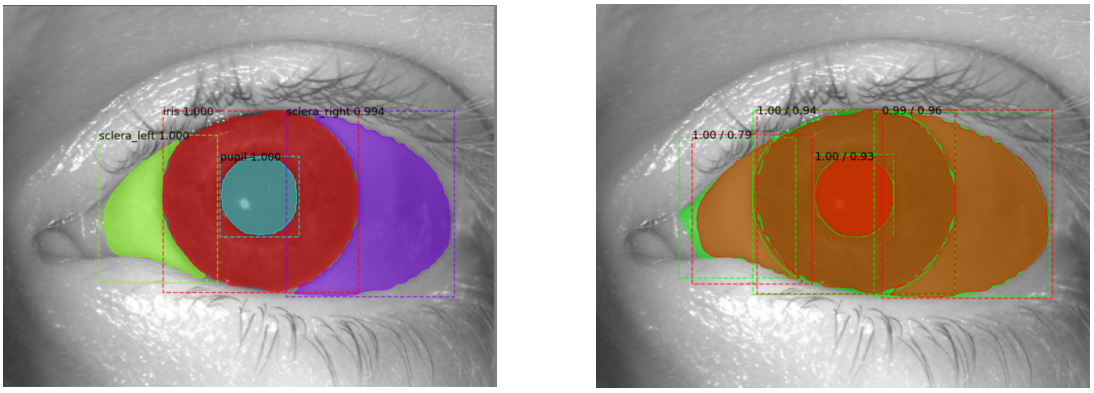


Figura 4.64: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.61:

Class	IoU
Iris	0.94044
Pupil	0.93176
Sclera right	0.95625
Sclera left	0.79448
mIoU	0.90573

Tabla 4.61: IoU por clase y mIoU promedio de cada clase.

La figura 4.65 muestra los resultados de segmentación con el mejor modelo para una imagen con presencia de maquillaje:

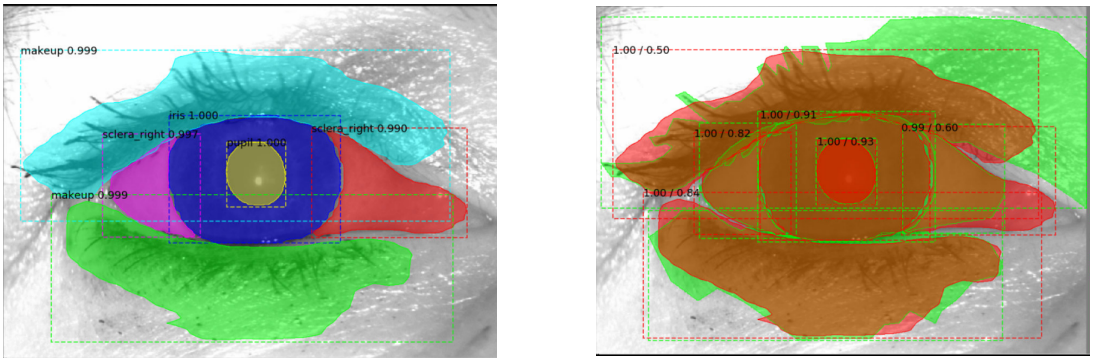


Figura 4.65: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.62:

Class	IoU
Iris	0.90687
Pupil	0.93106
Sclera right	0.82389
Sclera right	0.60159
Sclera left	0.00000
MakeUp	0.49840
MakeUp	0.83763
mIoU	0.65706

Tabla 4.62: IoU por clase y mIoU promedio de cada clase. Se reporta dos veces la clase maquillaje puesto que existen dos áreas del ojo que presentan esta clase. Se reporta además que la clase esclera derecha no fue detectada.

1.3.16. Prueba 16

Reentrenar el backbone produjo un incremento de 0.80 a 0.82 de mIoU. Dado el gran numero de steps y de épocas, se estima que el rendimiento del modelo incrementará dada una reducción de 10 unidades del learning rate. Los hiper-parámetros utilizados para esta prueba fueron:

Parameter	Value
Weights	COCO
Images per GPU	2
Image Min Dimension	800
Image Max Dimension	1024
Learning Rate	0.0001
Weight Decay	0.0001
Detection Min Confidence	0.9
Epochs	200
Batch Size	2
Steps	800
Layers	3+
Data Augmentation	Yes

Tabla 4.63: Los steps corresponden al calculo: $steps = [TI/(BatchSize)] * 20$, donde TI corresponde a las 80 imágenes de entrenamiento usadas.

Obteniendo los siguientes resultados de mIoU por época:

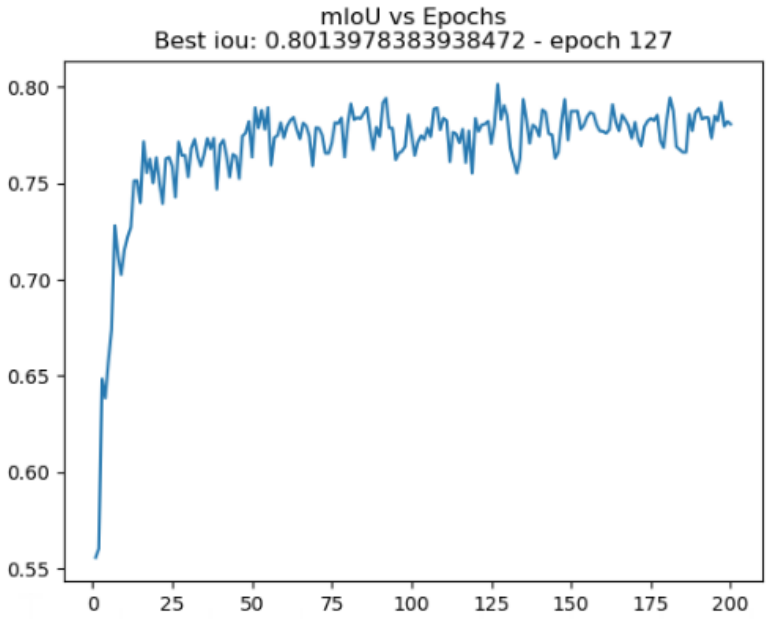


Figura 4.66: Resultados de mIoU por época de la prueba 16 con ResNet101 por backbone. El modelo con mejor rendimiento fue el de la época 127 con un mIoU de 0.80139 para todo el set de validación. El eje-X corresponde a las épocas, mientras que el eje-Y corresponde al mIoU.

La figura 4.67 muestra los resultados de segmentación con el mejor modelo para la presente prueba:

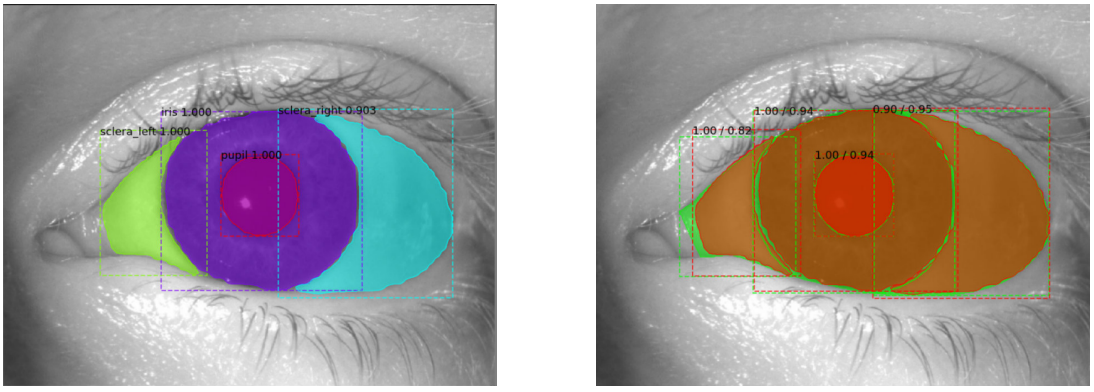


Figura 4.67: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.64:

Class	IoU
Iris	0.94095
Pupil	0.94137
Sclera right	0.94665
Sclera left	0.81674
mIoU	0.91143

Tabla 4.64: IoU por clase y mIoU promedio de cada clase.

La figura 4.68 muestra los resultados de segmentación con el mejor modelo para una imagen con presencia de maquillaje:

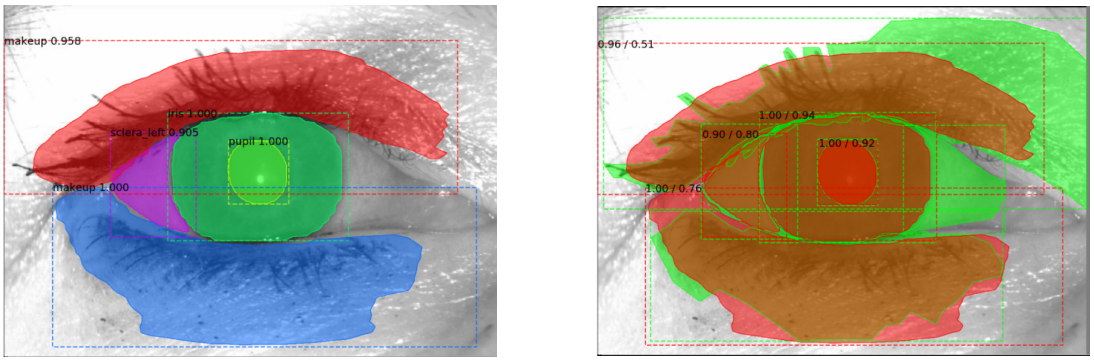


Figura 4.68: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.65:

Class	IoU
Iris	0.93634
Pupil	0.91633
Sclera right	0.00000
Sclera left	0.79501
MakeUp	0.76331
MakeUp	0.50740
mIoU	0.65306

Tabla 4.65: IoU por clase y mIoU promedio de cada clase. Se reporta dos veces la clase maquillaje puesto que existen dos áreas del ojo que presentan esta clase. Se reporta además que la clase esclera derecha no fue detectada.

1.3.17. Prueba 17

Ante el decremento del learning rate, desempeño del modelo se redujo de 0.82 a 0.80. Se estima que entrenando un nivel más del backbone ayudará a remontar el desempeño del modelo. Los hiper-parámetros utilizados para esta prueba fueron:

Parameter	**Value**
Weights	COCO
Images per GPU	2
Image Min Dimension	800
Image Max Dimension	1024
Learning Rate	0.0001
Weight Decay	0.001
Detection Min Confidence	0.9
Epochs	200
Batch Size	2
Steps	800
Layers	4+
Data Augmentation	Yes

Tabla 4.66: Los steps corresponden al calculo: $steps = [TI/(BatchSize)] * 20$, donde TI corresponde a las 80 imágenes de entrenamiento usadas.

Obteniendo los siguientes resultados de mIoU por época:

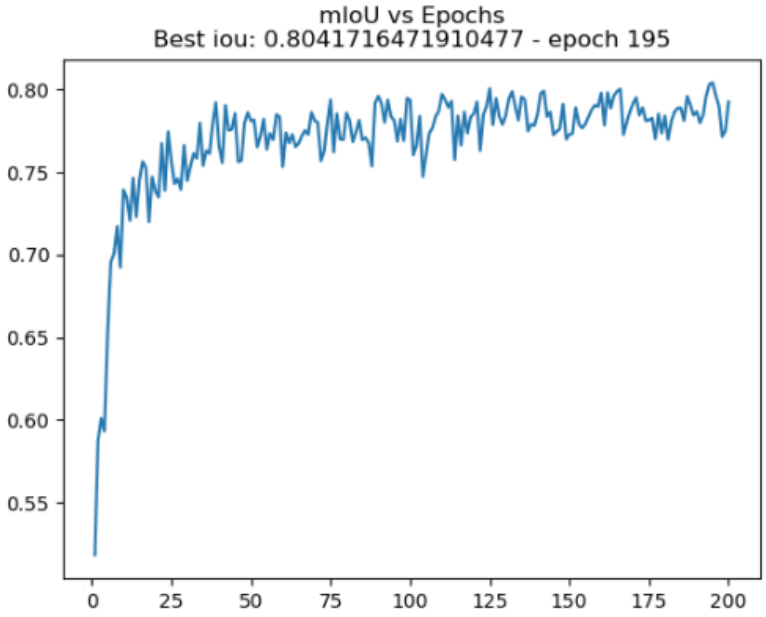


Figura 4.69: Resultados de mIoU por época de la prueba 17 con ResNet101 por backbone. El modelo con mejor rendimiento fue el de la época 195 con un mIoU de 0.80417 para todo el set de validación. El eje-X corresponde a las épocas, mientras que el eje-Y corresponde al mIoU.

La figura 4.70 muestra los resultados de segmentación con el mejor modelo para la presente prueba:

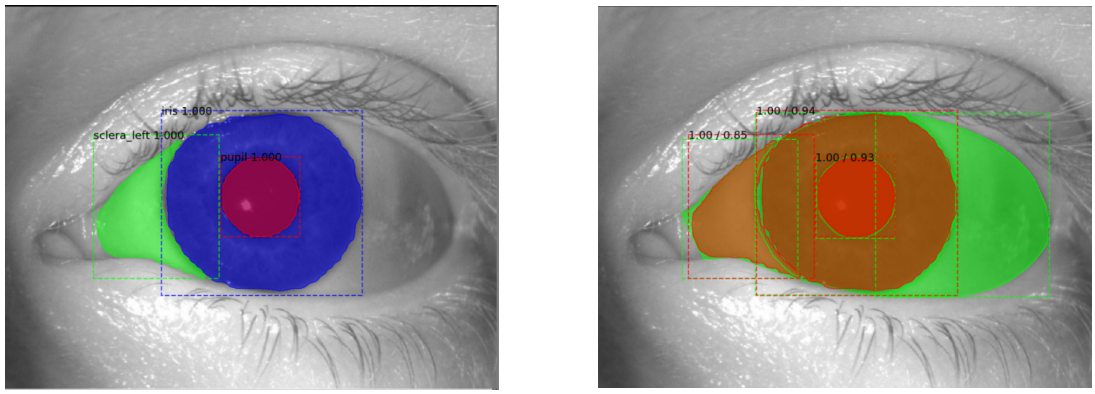


Figura 4.70: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.67:

Class	IoU
Iris	0.93865
Pupil	0.93050
Sclera right	0.00000
Sclera left	0.84737
mIoU	0.67913

Tabla 4.67: IoU por clase y mIoU promedio de cada clase. Se reporta que la esclera derecha no fue detectada.

La figura 4.71 muestra los resultados de segmentación con el mejor modelo para una imagen con presencia de maquillaje:

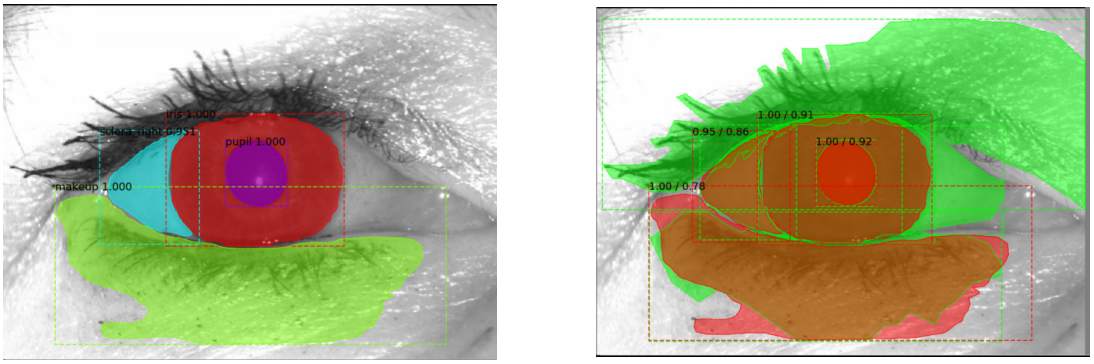


Figura 4.71: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.68:

Class	IoU
Iris	0.91481
Pupil	0.91666
Sclera right	0.00000
Sclera left	0.86039
MakeUp	0.78466
MakeUp	0.00000
mIoU	0.57942

Tabla 4.68: IoU por clase y mIoU promedio de cada clase. Se reporta dos veces la clase maquillaje puesto que existen dos áreas del ojo que presentan esta clase. Se reporta además que la esclera derecha no fue detectada.

1.3.18. Prueba 18

Dadas las dos pruebas anteriores, para la problemática actual se estima que un learning rate de 0.001, 200 épocas y 800 steps son parametros ideales para obtener resultados por sobre 0.8 de mIoU. La presente prueba busca confirmar lo antes mencionado, así que los hiper-parámetros utilizados para esta prueba fueron:

Parameter	Value
Weights	COCO
Images per GPU	2
Image Min Dimension	800
Image Max Dimension	1024
Learning Rate	0.001
Weight Decay	0.0001
Detection Min Confidence	0.9
Epochs	200
Batch Size	2
Steps	800
Layers	3+
Data Augmentation	Yes

Tabla 4.69: Los steps corresponden al calculo: $steps = [TI/(BatchSize)] * 20$, donde TI corresponde a las 80 imágenes de entrenamiento usadas.

Obteniendo los siguientes resultados de mIoU por época:

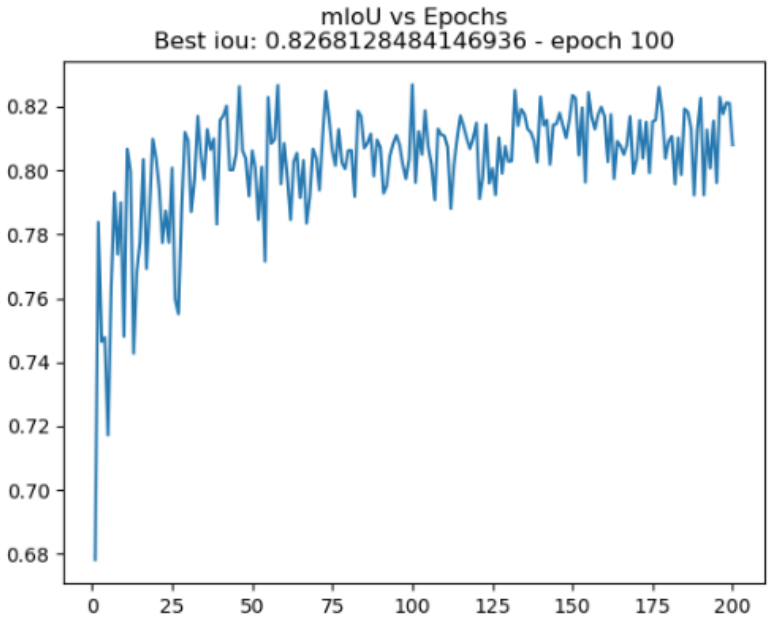


Figura 4.72: Resultados de mIoU por época de la prueba 18 con ResNet101 por backbone. El modelo con mejor rendimiento fue el de la época 100 con un mIoU de 0.82681 para todo el set de validación. El eje-X corresponde a las épocas, mientras que el eje-Y corresponde al mIoU.

La figura 4.73 muestra los resultados de segmentación con el mejor modelo para la presente prueba:

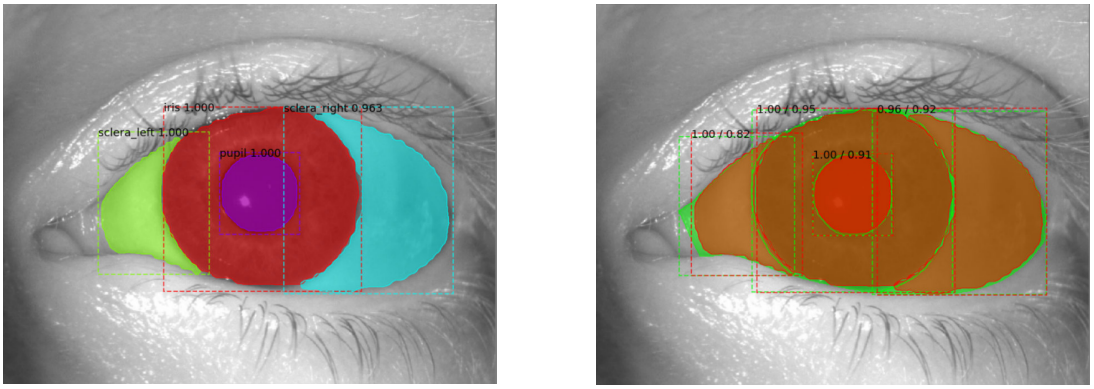


Figura 4.73: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.70:

Class	IoU
Iris	0.94601
Pupil	0.91378
Sclera right	0.92056
Sclera left	0.82177
mIoU	0.67913

Tabla 4.70: IoU por clase y mIoU promedio de cada clase.

La figura 4.74 muestra los resultados de segmentación con el mejor modelo para una imagen con presencia de maquillaje:

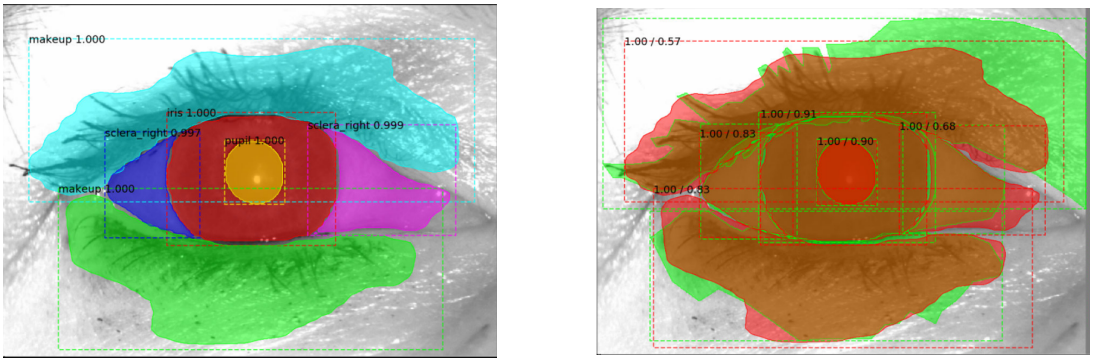


Figura 4.74: La imagen de la izquierda corresponde a las máscaras predichas por el modelo, mientras que la imagen de la derecha corresponde a la superposición de las máscaras predichas por el modelo con las máscaras marcadas manualmente.

Se reportan los IoU de cada clase en la tabla 4.71:

Class	IoU
Iris	0.90524
Pupil	0.90018
Sclera right	0.67597
Sclera right	0.83074
Sclera left	0.00000
MakeUp	0.56724
MakeUp	0.82551
mIoU	0.67213

Tabla 4.71: IoU por clase y mIoU promedio de cada clase. Se reporta dos veces la clase maquillaje puesto que existen dos áreas del ojo que presentan esta clase. Se reporta además que la esclera derecha no fue detectada.

1.3.19. Resumen y análisis de pruebas ResNet101

Se presenta finalmente una tabla resumiendo las pruebas anteriores:

Prueba	Epochs	LR	DA	Steps	Layers	Weights	mIoU
01	100	0.001	No	80	heads	COCO	0.75289
02	100	0.0001	No	80	heads	COCO	0.69135
03	150	0.0001	No	80	heads	COCO	0.71202
04	200	0.0001	No	80	heads	COCO	0.71329
05	50	0.001	No	80	heads	COCO	0.73823
06	50	0.001	No	80	heads	IMAGENET	0.74744
07	100	0.001	No	80	heads	IMAGENET	0.77181
08	150	0.001	No	80	heads	IMAGENET	0.77286
09	200	0.001	Yes	120	heads	COCO	0.78768
10	200	0.0001	Yes	120	heads	COCO	0.69698
11	100	0.0001	Yes	400	heads	COCO	0.72227
12	100	0.001	Yes	400	heads	COCO	0.79251
13	150	0.001	Yes	400	heads	COCO	0.79770
14	100	0.001	Yes	800	heads	COCO	0.80539
15	200	0.001	Yes	800	3+	COCO	0.82302
16	200	0.0001	Yes	800	3+	COCO	0.80139
17	200	0.0001	Yes	800	4+	COCO	0.80417
18	200	0.001	Yes	800	3+	COCO	0.82681

Tabla 4.72: Tabla de resumen de pruebas con ResNet101, donde la prueba 02 obtuvo el peor desempeño, mientras que la prueba 18 obtuvo el mejor desempeño (en términos de mIoU).

Adicionalmente, se reportan los mIoU y desviación estándar de la mejor prueba (18) separados por cada clase de la carpeta de validación en la tabla 4.73:

Class	mIoU	std
Iris	0.85999	0.05785
Pupil	0.92151	0.02527
Sclera left	0.88574	0.07957
Sclera right	0.77849	0.25269
MakeUp	0.36032	0.40372

Tabla 4.73: mIoU de cada clase del set de validación (50 imágenes).

De las pruebas realizados con ResNet101 se puede rescatar que:

- Reentrenar desde la tercera capa del backbone provee mejores resultados.

- La cantidad de épocas tiene un impacto positivo sobre el rendimiento final del modelo.
- Aplicar data augmentation tiene un impacto positivo en el rendimiento final de la red.
- Aumentar los steps para obtener más ejemplos del data augmentation tiene un impacto positivo en el rendimiento final de la red.
- Todo lo anterior combinado (aumentar steps, epocas, aplicar data augmentation y reentrenar desde la tercera capa del backbone) tiene un gran impacto en el desempeño final de la red (ver prueba 18).
- Con ResNet101 por backbone, la red Mask R-CNN demora 0,2075 segundos en promedio realizar la predicción de máscaras de una imagen.

1.4. Análisis y comparación de ResNet50 y ResNet101

Al realizar la comparación entre el mejor resultado de ResNet50 y el mejor resultado de ResNet101 (Ver tabla 4.74), se rescata que ResNet101 obtuvo un mejor desempeño (mIoU: 0.82561) reentrenando solo 3 niveles del backbone, mientras que ResNet50 obtuvo un ligero menor desempeño (mIoU: 0.82069) reentrenando todas sus capas. Se estima que reentrenando todas las capas de ResNet101 con los mismos hiperparámetros de su última prueba, se obtendrá un desempeño aún mayor que el presentado en su prueba final (Ver prueba 1.3.18).

Prueba	Backbone	Epochs	LR	DA	Steps	Layers	Weights	mIoU	TPIs
05	ResNet50	200	0.001	Yes	800	all	COCO	0.82069	0,1487
18	ResNet101	200	0.001	Yes	800	3+	COCO	0.82681	0,2075

Tabla 4.74: Mejores resultados ResNet50 y ResNet101. LR: Learning Rate, DA: Data Augmentation, TPIs: Tiempo Promedio por Imagen en segundos.

2. Experimento con DenseNet

Para las pruebas realizadas en esta sección fue utilizada la base de datos OpenEDS [18] descrita en la metodología 02 (Ver capítulo 1, sección 5.2), dividiendo la totalidad de la base de datos en 3 distintas carpetas:

- **train:** 8,916
- **test:** 1,440
- **validation:** 2,403

Como fue detallado en la metodología 02, esta base de datos dispone de las marcas necesarias para cada clase (iris - pupila - esclera) solo para los sets de train y validation, mientras que el set de test es utilizado para generar las predicciones del algoritmo y subir los resultados a la página de EvalAI para así obtener la metrica de evaluación del modelo (Ver capítulo 3, métrica 4.11).

2.1. Arquitectura Propuesta FC-DenseNet-10

Para la competencia realizada por *EvalAI*[4] las características de las *DenseNets* (Ver subsección 2.8 del marco teórico) las hacen perfectas para tareas de segmentación semántica puesto que aplican conexiones

[4] https://evalai.cloudcv.org/web/challenges/challenge-page/353/overview

atajo y supervisión multi-escala.

Las redes *DenseNet56, DenseNet67 y DenseNet103* fueron diseñadas para representar características complejas y así cumplir con la tarea de segmentar elementos de ciudades tales como autos, personas, edificaciones, árboles y otros [17]. Por lo tanto, una arquitectura simple fue desarrollada para así reducir el número de bloques densos. Así, el modelo propuesto para el trabajo de segmentación de iris es uno compuesto de un camino de muestreo descendente con una capa de transición hacia abajo (TD) más un camino de muestreo ascendente con una capa de transición hacia arriba (TU) en vez de 4 transiciones (2TU + 2TD) como fue propuesto por el modelo original. Los bloques densos de las *DenseNets* son útiles dado que conducen características de bajo nivel desde capas anteriores directamente a características de alto nivel de las capas superiores, permitiendo así el reuso eficiente de información. Una representación gráfica del modelo propuesto se muestra en la figura 4.75:

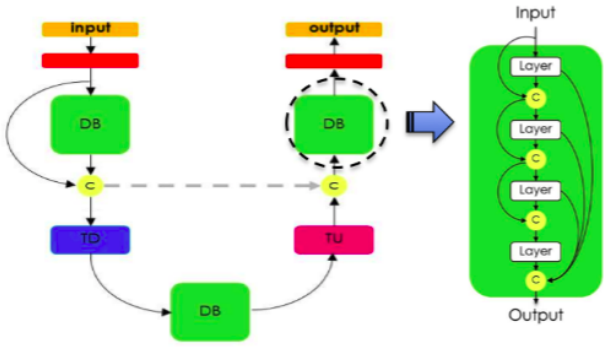


Figura 4.75: Diagrama de arquitectura de *FC-DenseNet-10* propuesta para segmentación semántica de iris. El modelo presentado esta compuesto de un camino de muestreo descendente con una capa de transición hacia abajo (TD) y un camino de muestreo ascendente con una capa de transición hacia arriba (TU). Cada bloque denso (DB) tiene 3, 4 y 3 capas respectivamente.

Se presenta una tabla con el detalle del modelo propuesto:

Arquitectura	
Input	m = 3
3 x 3 Convolution	m = 48
DB (3 layers) + TD	m = 112
DB (4 layers)	m = 112
DB (3 layers) + TU	m = 256
3 x 3 Convolution	m = c
Softmax	

Tabla 4.75: Detalles de la arquitectura propuesta del modelo FC-DenseNet-10. La siguiente notación es usada: DB (Dense Block), TD (Transition Down), TU (Transition Up), M (numero total de mapas de caracteristicas al final de un bloque) y C (número de clases). C = 4.

Y finalmente se muestra una tabla de comparación de cantidad de parámetros entre distintos modelos:

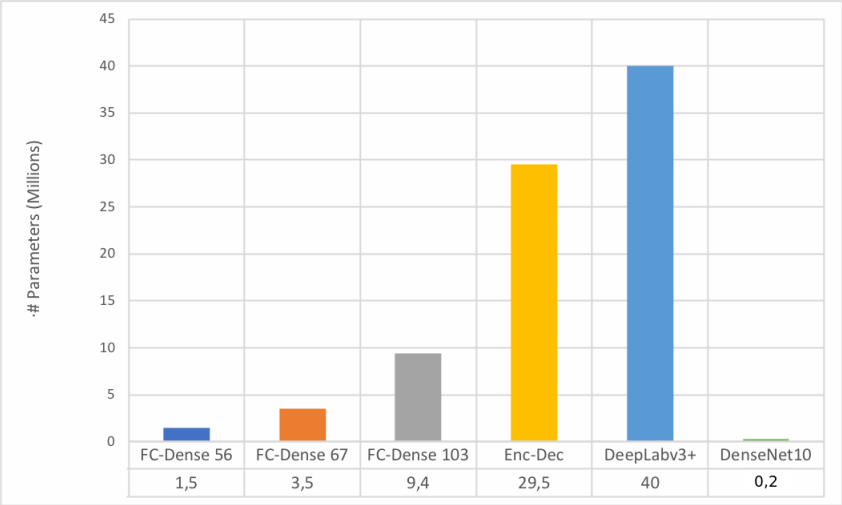


Figura 4.76: Comparación de número de parámetros. FC-DenseNet-10 (202,084 parámetros totales) supera todos los otros modelos. El eje X representa los modelos, mientras que el eje Y representa la cantidad de parámetros en millones.

2.2. Resultados

Para obtener la mejor arquitectura con el menor número de parámetros manteniendo una métrica de IoU alta, tres tipos de arquitecturas de DenseNet fueron analizadas. DenseNet56 (1,5 millones de parametros), DenseNet67 (3,5 millones de parametros) y DenseNet101 (9,4) millones de parámetros. Según el estado del arte, estos modelos tradicionales alcanzan valores reducidos de error para tareas relacionadas con segmentación, pero con un coste de millones de parámetros dada las tareas complejas para las cuales fueron diseñados. Sin embargo, como las implementaciones de modelos DenseNet no utilizan modelos pre-entrenados (ResNet, MobileNet, VGG) como backbone, la cantidad de parámetros pueden ser reducidos y el modelo puede ser ajustado según el problema a resolver, por lo cual 14 modelos distintos fueron entrenados tal como se muestra en la tabla 4.76:

Model Name	Model Params	mIoU	Score
FC-DenseNet_1	52,924	0.8613	0.9306
FC-DenseNet_2	52,924	0.8727	0.9363
FC-DenseNet_3	165,360	0.9206	0.9603
FC-DenseNet_4	165,360	0.9239	0.9619
FC-DenseNet_5	165,360	0.9245	0.9622
FC-DenseNet_6	202,084	0.9200	0.9600
FC-DenseNet_7	202,084	0.9290	0.9645
FC-DenseNet_8	202,084	0.9269	0.9634
FC-DenseNet_9	202,084	0.9297	0.9648
FC-DenseNet_10	202,084	0.9334	0.9667
FC-DenseNet_11	202,084	0.9347	0.9673
FC-DenseNet_12	202,084	0.9174	0.9587
FC-DenseNet_13	**202.084**	**0.9429**	**0.9714**
FC-DenseNet_14	237,868	0.9265	0.9632

Tabla 4.76: Resultados de los mejores modelos de segmentación semántica basados en DenseNet 56-67-103. El mejor resultado de acuerdo con la evaluación de EvalAI está destacado en negrita. La primera columna muestra el nombre de los modelos, la segunda columna muestra la cantidad de parametros y las columnas tercera y cuarta muestran el IoU promedio y la puntuación alcanzada por la métrica propuesta por EvalAI.

Cada prueba expuesta en la tabla fue entrenado desde cero (inicialización de la red con parámetros aleatorios) y fue aplicada la técnica de *data augmentation* al set de entrenamiento con la finalidad de obtener una mayor cantidad de imágenes de entrenamiento y así conseguir una capacidad de generalización superior por parte de los modelos entrenados. Un ejemplo del data augmentation es mostrado en la figura 4.77:

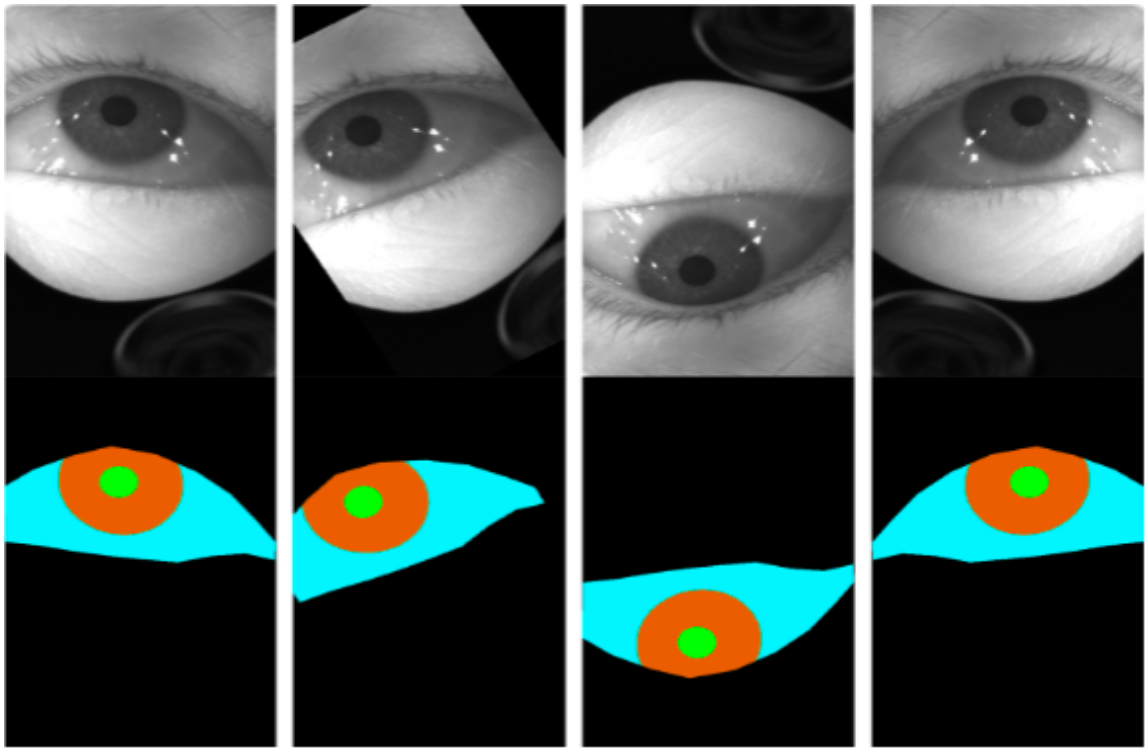


Figura 4.77: Ejemplo de Data Augmentation aplicado sobre una imágen y sus anotaciones. La primera columna corresponde a la imagen original, mientras que las demás se encuentran rotadas, espejada vertical y horizontalmente. Imágenes obtenidas de [18].

Adicionalmente, todas las pruebas fueron realizadas en un computador Intel I7 con 32 GB de RAM y una GPU 1080TI, fue utilizado el optimizador RMS [125] con un *learning rate* desde 0,1 a 0,0001 con un *decay* de 0,995. En promedio, cada modelo demoró alrededor de 4 días en entrenarse con 100 épocas y el número de capas por cada bloque denso (TU y TD. Ver marco teórico 3, sección 2.8) fueron reducidos desde 5 a 3 y 4 respectivamente.

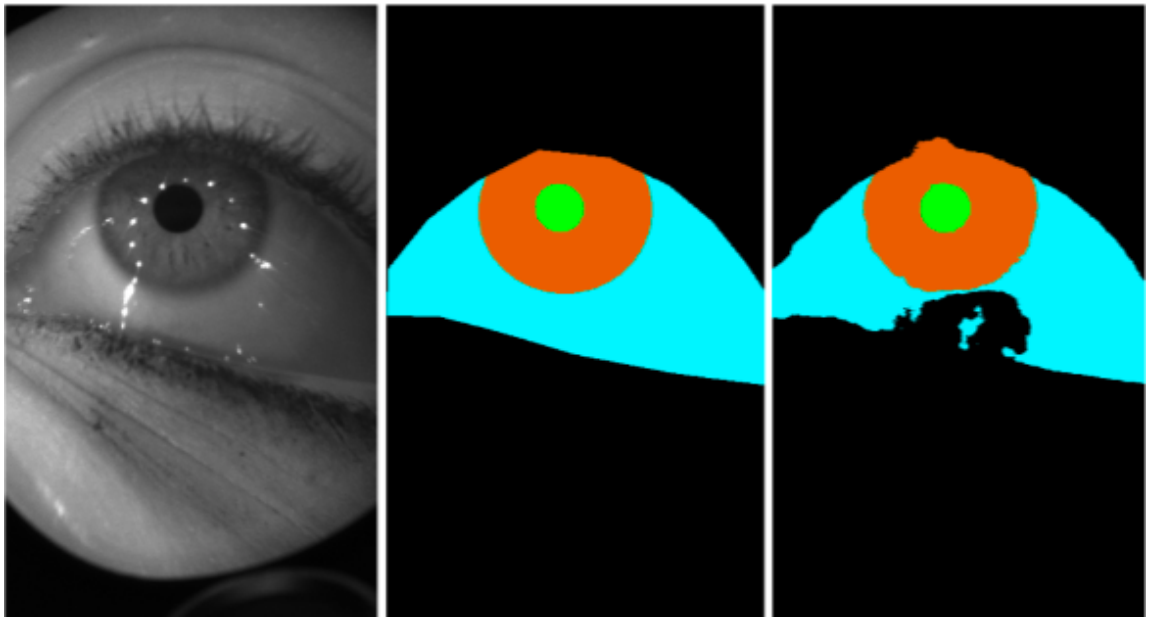


Figura 4.78: Ejemplo de segmentación de baja calidad. La imagen de la izquierda corresponde a la imagen original, mientras que la del medio es su equivalente a marcado manual (ground truth) y la imagen de la derecha es la segmentación obtenida de un modelo entrenado. Se observa una baja calidad de segmentación de para la esclera. Primeras dos imágenes obtenidas de [18], mientras que la tercera es de elaboración propia.

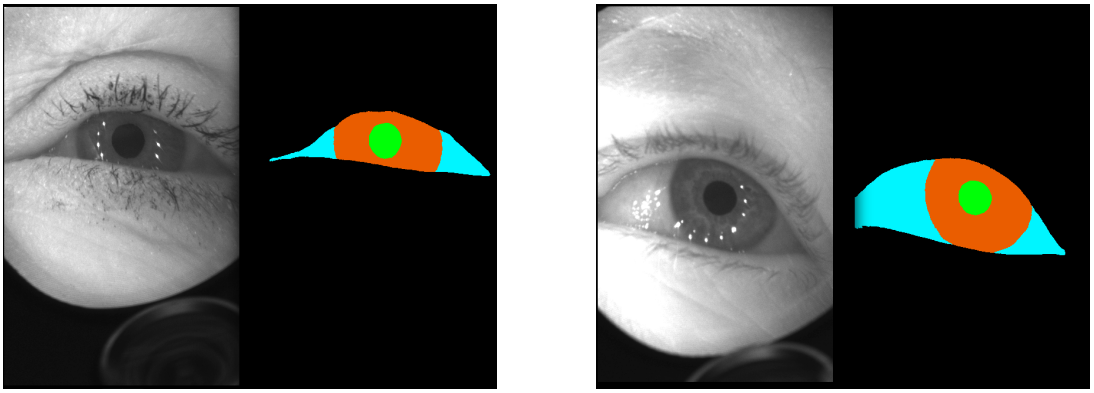


Figura 4.79: Resultados de modelo FC-DenseNet_13. A la izquierda la imagen original y a la derecha las máscaras obtenidas por el modelo. Imágenes obtenidas de [18].

La figura 4.80 reporta el mejor resultado obtenido por el modelo FC-DenseNet10. Fue obtenido un alto desempeño y un bajo ratio de error. Obsérvese que el modelo FC-DenseNet-10 fue entrenado con un alto número de ejemplos y con pocos parámetros (complejidad de modelo).

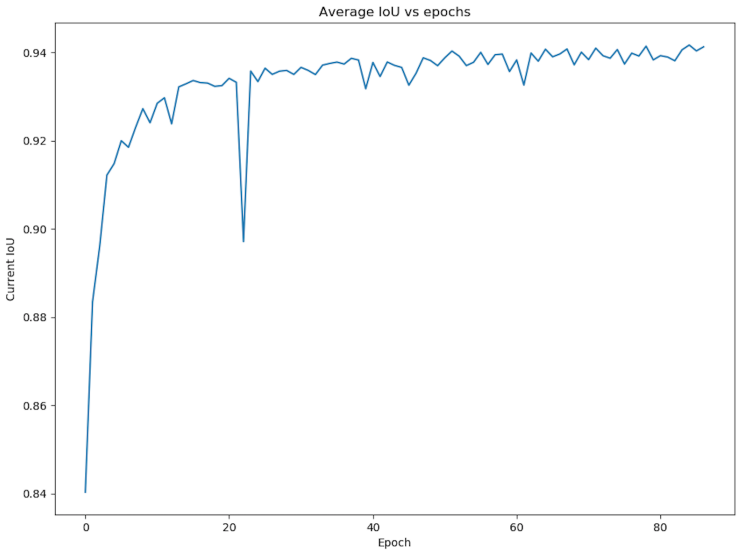


Figura 4.80: Curva de validación (IoU) del mejor modelo FC-DenseNet-10 propuesto.

La figura 4.81 reporta la perdida (categorical-cross-entropy) [126] del mejor resultado obtenido por el modelo FC-DenseNet10. La inestabilidad y ruido del entrenamiento fueron decrementando conforme aumentaban las épocas. Un alto rendimiento fue alcanzado con un ratio de error muy bajo. Esta función de pérdida examina cada píxel individualmente y lo compara con las predicciones de las clases (máscaras binarias para cada una de las clases por segmentar).

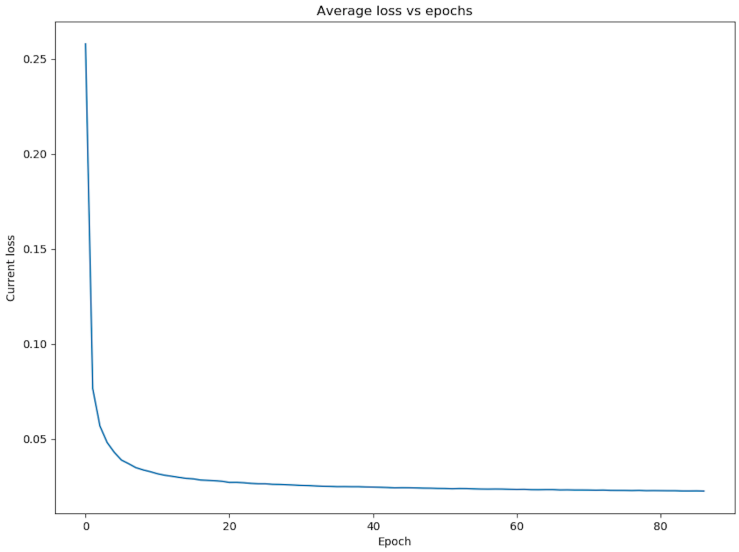


Figura 4.81: Resultados de pérdida vs épocas.

La tabla 4.77 muestra los resultados del experimento.

Segmentación Semántica	Experimento FC-DenseNet
N° de imágenes	8,916
Tiempo Entrenamiento (Hrs)	96 aprox.
Épocas	100
Pasos por época	9,040
Dimensión de imágenes	640 x 400
Learning rate	0.001
Data Aug	Si
Cantidad de parámetros	202,084
Tiempo por predicción (seg)	0.08
Tamaño (MB)	4,0 MB
mIoU	0.9429
EvalAI score	0,9714

Tabla 4.77: Resultados experimento FC-DenseNet_13.

CAPÍTULO 5

Conclusiones

Respecto a la primera hipótesis, se concluye que técnicas de segmentación asociadas al campo de Deep Learning son capaces de extraer de manera automática información relevante del iris. Cabe destacar que no es necesaria una arquitectura tan complicada como lo es Mask R-CNN, sino que una arquitectura personalizada y más simple como DenseNet es capaz de cumplir incluso con mayor precisión y en menor tiempo el mismo trabajo realizado por esta primera herramienta. Lo mencionado queda demostrado en cada uno de los experimentos realizados en la sección de experimentos (Ver capítulo 4).

Respecto a la segunda hipótesis, se concluye que las técnicas usadas son capaces de extraer cada una de las zonas del ojo en espectro NIR, segmentando con un desempeño competente con el estado del arte las zonas de mayor relevancia (iris y pupila). Debido a áreas con iluminación directa al ojo, las zonas de las escleras representan un mayor desafío puesto que su valor de color en píxel es muy similar a zonas externas a la estructura del ojo tales como el arco de la ceja, la cuenca del ojo y secciones de los párpados. Mientras que para el maquillaje se presentaron las siguientes situaciones adversas:

1. Al momento de crear la base de datos fue generalizado el término maquillaje, es decir que aparte de marcar el maquillaje oscuro, también fueron marcadas zonas con maquillaje tipo sombra, lo cual implicó un decremento considerable del IoU promedio por cada modelo de Mask R-CNN, pero no así para DenseNet puesto que esta fue entrenada con una base de datos sin maquillaje marcado.

2. Algunas personas tienen mayor abundancia de cejas, pestañas y/o pestañas más gruesas en ambos párpados, lo que originó situaciones de falsos positivos, lo cual también redujo el IoU ponderado por el modelo.

3. Maquillaje moderado o fino no es detectado por el modelo debido a la falta de estos ejemplos en el set de entrenamiento.

Según las tablas 4.17 y 4.73 del capítulo 4, la clase maquillaje es la que peor IoU promedio por clase presentó mientras que la clase pupila lidera con el mejor IoU promedio por clase. Según los resultados se estiman dos escenarios:

- **Mejor caso:** Un ojo sin maquillaje y con los párpados cubriendo la menor zona posible de iris y escleras presentará una excelente segmentación para todas sus clases disponibles.

- **Peor caso:** Un ojo con distintos tipos de maquillaje y con los párpados cubriendo zonas de iris, pupila y esclera presentará un gran desafío para la segmentación.

Y respecto a la tercera y última hipótesis, se obtienen resultados competitivos con el estado del arte [57, 27] para Mask R-CNN y para DenseNet respectivamente.

Realizando un análisis entre los resultados de los experimentos de Mask R-CNN y de DenseNet, se rescata que la capacidad de generalización de las redes está fuertemente relacionado con la cantidad de imágenes disponibles para entrenar. Es decir, entre más grande sea la base de datos de imágenes disponible para entrenar, mayor rendimiento final y capacidad de generalización tendrá una red. Esto se ve contrastado en las 8,916 imágenes de entrenamiento de DenseNet versus las 80 imágenes de entrenamiento de Mask R-CNN, esto último dada la cantidad excesiva de horas requeridas para segmentar correctamente la base de datos a utilizar. Si bien en ambos casos fue utilizada la técnica de data augmentation, los ejemplos de entrenamiento para Mask R-CNN no superaron el valor de las 1,600 unidades. Además, cabe destacar que cada experimento de DenseNet demoró entre 2 y 4 días mientras que los experimentos de Mask R-CNN no demoraron mas de 2 días, esto debido a la cantidad de imágenes de ambos datasets.

Finalmente, no es necesaria una arquitectura tan compleja como lo es Mask R-CNN (63,754,935 parámetros), pero sí es necesaria una arquitectura compleja más allá de 50,000 parámetros aproximadamente si se busca una métrica competitiva con el estado del arte (Ver tabla de resultados de DenseNet 4.76, capítulo 4). La arquitectura propuesta de DenseNet fue capaz de responder ante el problema de segmentación un 60 % más rápido, con un 99.683 % menos parámetros que Mask R-CNN y con un mejor mIoU resultante tal como se muestra en la tabla 5.1:

Model	**TPIs**	**mIoU**
Mask RCNN (ResNet50)	0.1487	0.82069
Mask R-CNN (ResNet101)	0.2075	0.82681
DenseNet	0.081	0.9429

Tabla 5.1: Comparación de tiempo de predicción y mIoU por modelo. DenseNet lidera en tiempo y mIoU.

Para trabajos futuros se estima realizar las siguientes tareas:

1. Marcar más ejemplos para el framework Mask R-CNN y realizar nuevos entrenamientos.
2. Entrenar Mask R-CNN con la base de datos OpenEDS.
3. Entrenar Mask R-CNN con una base de datos en distintos espectros visuales (RGB, L*a*b, HSV, etc).
4. Entrenar DenseNet con la base de datos de Mask R-CNN.
5. Entrenar DenseNet con una base de datos en distintos espectros visuales (RGB, L*a*b, HSV, etc).
6. Realizar cross-validation entre las bases de datos utilizadas para este trabajo.
7. Realizar predicciones sobre el canal Rojo de imágenes en espectro RGB, esto dado que el canal R está más cercano al espectro NIR.
8. Implementar un backbone distinto a la familia ResNet para Mask R-CNN.
9. Explorar otros tipos de arquitecturas (similar a lo realizado con DenseNet) para segmentar las zonas del ojo.
10. Implementar una métrica ponderada de IoU respecto a la cantidad de píxeles de cada clase.
11. Utilizar ambas redes para segmentar pestañas y eliminarlas del iris.

12. Realizar segmentación de instancia con Mask R-CNN para segmentar distintos iris de un conjunto de personas presentes en una sola imagen.

Referencias

[1] K. W. Bowyer, S. E. Baker, A. Hentz, K. Hollingsworth, T. Peters, and P. J. Flynn, "Factors that degrade the match distribution in iris biometrics," *Identity in the Information Society*, vol. 2, pp. 327–343, Dec. 2009.

[2] L. Masek *et al.*, *Recognition of human iris patterns for biometric identification*. PhD thesis, Master's thesis, University of Western Australia, 2003.

[3] A. Dutta and A. Zisserman, "The VGG image annotator (VIA)," *arXiv preprint arXiv:1904.10699*, 2019.

[4] Gringer, "Linear visible spectrum." , url: https://commons.wikimedia.org/wiki/File:Linear_visible_spectrum.svg, 2008. Last accessed 09-05-2010.

[5] Martin, "Artificial intelligence: How will it affect legal practice – and when?," 2016. Last accessed 04-06-2019.

[6] "CS231n Convolutional Neural Networks for Visual Recognition." , url: http://cs231n.github.io/convolutional-networks. Last accessed 2019-05-14.

[7] F. Chollet, *Deep Learning with Python*. Manning Publications, 2017.

[8] "Image Filtering - A comprehensive tutorial towards 2D convolution and image filtering." , url: https://bit.ly/2PnUwSr. Last accessed 2019-06-11.

[9] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015.

[10] J. Jordan, "An overview of semantic image segmentation." , url: https://www.jeremyjordan.me/semantic-segmentation/. Last accessed 2018-06-11, 2018.

[11] W. Abdulla, "Splash of Color: Instance Segmentation with Mask R-CNN and TensorFlow." , url: https://bit.ly/2CxbfKI. Last Accessed 2018-06-11.

[12] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask r-CNN," in *2017 IEEE International Conference on Computer Vision (ICCV)*, IEEE, oct 2017.

[13] J. Hui, "Image Segmentation with Mask R-CNN," *Medium*, 04 2018. url: https://bit.ly/2R6LrZW. Last accessed 2019-06-26.

[14] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2117–2125, 2017.

[15] J. Hui, "Understanding Feature Pyramid Networks for object detection (FPN)," *Medium*. url: https://bit.ly/2HdbpKR. Last accessed 2019-06-26.

[16] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," *CoRR*, vol. abs/1608.06993, 2016.

[17] S. Jégou, M. Drozdzal, D. Vázquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," *CoRR*, vol. abs/1611.09326, 2016.

[18] S. J. Garbin, Y. Shen, I. Schuetz, R. Cavin, G. Hughes, and S. S. Talathi, "Openeds: Open eye dataset," *arXiv preprint arXiv:1905.03702*, 2019.

[19] A. Rosebrock, "Intersection over Union (IoU) for object detection." url: https://bit.ly/2FBilio. Last accessed 2019-05-14.

[20] J. Hui, "mAP (mean Average Precision) for Object Detection." url: https://bit.ly/2PpmSLc. Last accessed 2019-06-14.

[21] C. S. Bezerra, R. Laroca, D. R. Lucio, E. Severo, L. F. Oliveira, A. S. Britto, and D. Menotti, "Robust iris segmentation based on fully convolutional networks and generative adversarial networks," in *2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pp. 281–288, IEEE, 2018.

[22] M. Arsalan, R. Naqvi, D. Kim, P. Nguyen, M. Owais, and K. Park, "Irisdensenet: Robust iris segmentation using densely connected fully convolutional networks in the images by visible light and near-infrared light camera sensors," *Sensors*, vol. 18, no. 5, p. 1501, 2018.

[23] G. Lin, A. Milan, C. Shen, and I. Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[24] H. Hofbauer, E. Jalilian, A. F. Sequeira, J. Ferryman, and A. Uhl, "Mobile nir iris recognition: Identifying problems and solutions," in *2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pp. 1–9, IEEE, 2019.

[25] E. Jalilian and A. Uhl, "Iris segmentation using fully convolutional encoder–decoder networks," in *Deep Learning for Biometrics*, pp. 133–155, Springer, 2017.

[26] H. Hofbauer, E. Jalilian, and A. Uhl, "Improved iris recognition using cnn noise masks and segmentations,"

[27] Y.-H. Li, P.-J. Huang, and Y. Juan, "An efficient and robust iris segmentation algorithm using deep learning," *Mobile Information Systems*, vol. 2019, 2019.

[28] A. Jain, A. Ross, and S. Prabhakar, "An introduction to biometric recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, pp. 4–20, Jan. 2004.

[29] K. Bonnen, B. F. Klare, and A. K. Jain, "Component-based representation in automated face recognition," *IEEE Transactions on Information Forensics and Security*, vol. 8, pp. 239–253, Jan. 2013.

[30] S. Prabhakar, S. Pankanti, and A. Jain, "Biometric recognition: security and privacy concerns," *IEEE Security & Privacy*, vol. 1, pp. 33–42, Mar. 2003.

[31] R. Devi and P. Sujatha, "A study on biometric and multi-modal biometric system modules, applications, techniques and challenges," in *2017 Conference on Emerging Devices and Smart Systems (ICEDSS)*, IEEE, Mar. 2017.

[32] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar, *Handbook of fingerprint recognition*. Springer Science & Business Media, 2009.

[33] C. Wilson, *Vein pattern recognition: a privacy-enhancing biometric*. CRC press, 2010.

[34] S. Sanderson and J. Erbetta, "Authentication for secure environments based on iris scanning technology," 2000.

[35] J. Daugman, "Information theory and the IrisCode," *IEEE Transactions on Information Forensics and Security*, vol. 11, pp. 400–409, Feb. 2016.

[36] K. W. Bowyer and M. J. Burge, *Handbook of iris recognition*. Springer, 2016.

[37] A. Alqahtani, "Evaluation of the reliability of iris recognition biometric authentication systems," in *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*, IEEE, Dec. 2016.

[38] A. K. Jain, P. Flynn, and A. A. Ross, *Handbook of biometrics*. Springer Science & Business Media, 2007.

[39] N. Othman, B. Dorizzi, and S. Garcia-Salicetti, "Osiris: An open source iris recognition software," *Pattern Recognition Letters*, vol. 82, pp. 124–131, 2016.

[40] R. P. Wildes, "Iris recognition: an emerging biometric technology," *Proceedings of the IEEE*, vol. 85, no. 9, pp. 1348–1363, 1997.

[41] A. J. Mansfield and J. L. Wayman, "Best practices in testing and reporting performance of biometric devices," 2002.

[42] K. W. Bowyer, K. Hollingsworth, and P. J. Flynn, "Image understanding for iris biometrics: A survey," *Computer vision and image understanding*, vol. 110, no. 2, pp. 281–307, 2008.

[43] J. Daugman, "How iris recognition works," in *The essential guide to image processing*, pp. 715–739, Elsevier, 2009.

[44] A. Bird, "DNA methylation patterns and epigenetic memory," *Genes & Development*, vol. 16, pp. 6–21, Jan. 2002.

[45] A. Weaver, "Biometric authentication," *Computer*, vol. 39, pp. 96–97, Feb. 2006.

[46] J. Daugman, "The importance of being random: statistical principles of iris recognition," *Pattern recognition*, vol. 36, no. 2, pp. 279–291, 2003.

[47] M. Regouid, M. Touahria, M. Benouis, and N. Costen, "Multimodal biometric system for ecg, ear and iris recognition based on local descriptors," *Multimedia Tools and Applications*, pp. 1–27, 2019.

[48] J. G. Daugman, "Biometric personal identification system based on iris analysis," Mar. 1994. US Patent 5,291,560.

[49] J. Daugman, "How iris recognition works. proceedings of 2002 international conference on image processing," 2002.

[50] R. P. Wildes, J. C. Asmuth, G. L. Green, S. C. Hsu, R. J. Kolczynski, J. R. Matey, and S. E. McBride, "A system for automated iris recognition," in *Proceedings of 1994 IEEE Workshop on Applications of Computer Vision*, pp. 121–128, IEEE, 1994.

[51] W. W. Boles and B. Boashash, "A human identification technique using images of the iris and wavelet transform," *IEEE transactions on signal processing*, vol. 46, no. 4, pp. 1185–1188, 1998.

[52] S. Lim, K. Lee, O. Byeon, and T. Kim, "Efficient iris recognition through improvement of feature vector and classifier," *ETRI journal*, vol. 23, no. 2, pp. 61–70, 2001.

[53] S.-I. Noh, K. Pae, C. Lee, and J. Kim, "Multiresolution independent component analysis for its iris identification," in *ITC-CSCC: International Technical Conference on Circuits Systems, Computers and Communications*, pp. 1675–1678, 2002.

[54] C. H. Morimoto, T. T. Santos, and A. S. Muniz, "Automatic iris segmentation using active near infra red lighting," in *XVIII Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI'05)*, pp. 37–43, IEEE, 2005.

[55] J. Zuo, N. D. Kalka, and N. A. Schmid, "A robust iris segmentation procedure for unconstrained subject presentation," in *2006 Biometrics Symposium: Special Session on Research at the Biometric Consortium Conference*, pp. 1–6, IEEE, 2006.

[56] Y. Chen, J. Wang, C. Han, L. Wang, and M. Adjouadi, "A robust segmentation approach to iris recognition based on video," in *2008 37th IEEE Applied Imagery Pattern Recognition Workshop*, pp. 1–8, IEEE, 2008.

[57] D. Kerrigan, M. Trokielewicz, A. Czajka, and K. Bowyer, "Iris recognition with image segmentation employing retrained off-the-shelf deep neural networks," *arXiv preprint arXiv:1901.01028*, 2019.

[58] H. Proenca, "Iris recognition: On the segmentation of degraded images acquired in the visible wavelength," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 8, pp. 1502–1516, 2010.

[59] C.-W. Tan and A. Kumar, "Automated segmentation of iris images using visible wavelength face images," in *CVPR 2011 WORKSHOPS*, pp. 9–14, IEEE, 2011.

[60] M. Arsalan, H. Hong, R. Naqvi, M. Lee, M. Kim, D. Kim, C. Kim, and K. Park, "Deep learning-based iris segmentation for iris recognition in visible light environment," *Symmetry*, vol. 9, no. 11, p. 263, 2017.

[61] M. A. Abdullah, J. A. Chambers, W. L. Woo, and S. S. Dlay, "Iris biometrie: Is the near-infrared spectrum always the best?," in *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pp. 816–819, IEEE, 2015.

[62] A. Gangwar and A. Joshi, "Deepirisnet: Deep iris representation with applications in iris recognition and cross-sensor iris recognition," in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 2301–2305, IEEE, 2016.

[63] H. Proença and L. A. Alexandre, "Iris segmentation methodology for non-cooperative recognition," *IEE Proceedings-Vision, Image and Signal Processing*, vol. 153, no. 2, pp. 199–205, 2006.

[64] D. Petrovska and A. Mayoue, "Description and documentation of the biosecure software library," *Project No IST-2002-507634-BioSecure, Deliverable*, 2007.

[65] C. Rathgeb, A. Uhl, and P. Wild, *Iris biometrics: from segmentation to template security*, vol. 59. Springer Science & Business Media, 2012.

[66] A. Uhl and P. Wild, "Weighted adaptive hough and ellipsopolar transforms for real-time iris segmentation," in *2012 5th IAPR international conference on biometrics (ICB)*, pp. 283–290, IEEE, 2012.

[67] A. Uhl and P. Wild, "Multi-stage visible wavelength and near infrared iris segmentation framework," in *International Conference Image Analysis and Recognition*, pp. 1–10, Springer, 2012.

[68] Z. Zhao and K. Ajay, "An accurate iris segmentation framework under relaxed imaging constraints using total variation model," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3828–3836, 2015.

[69] J. R. Matey, R. Broussard, and L. Kennell, "Iris image segmentation and sub-optimal images," *Image and Vision Computing*, vol. 28, no. 2, pp. 215–222, 2010.

[70] H. Proença and L. A. Alexandre, "Iris recognition: Analysis of the error rates regarding the accuracy of the segmentation stage," *Image and Vision Computing*, vol. 28, pp. 202–206, jan 2010.

[71] H. Hofbauer, F. Alonso-Fernandez, J. Bigun, and A. Uhl, "Experimental analysis regarding the influence of iris segmentation on the recognition rate," *Iet Biometrics*, vol. 5, no. 3, pp. 200–211, 2016.

[72] K. W. Bowyer, K. Hollingsworth, and P. J. Flynn, "Image understanding for iris biometrics: A survey," *Computer Vision and Image Understanding*, vol. 110, pp. 281–307, may 2008.

[73] H. Hofbauer, F. Alonso-Fernandez, J. Bigun, and A. Uhl, "Experimental analysis regarding the influence of iris segmentation on the recognition rate," *IET Biometrics*, vol. 5, pp. 200–211, sep 2016.

[74] L. P. Panych, P. D. Jakab, and F. A. Jolesz, "Implementation of wavelet-encoded mr imaging," *Journal of Magnetic Resonance Imaging*, vol. 3, no. 4, pp. 649–655, 1993.

[75] A. K. Jain and F. Farrokhnia, "Unsupervised texture segmentation using gabor filters," *Pattern recognition*, vol. 24, no. 12, pp. 1167–1186, 1991.

[76] D. J. Field, "Relations between the statistics of natural images and the response properties of cortical cells," *Josa a*, vol. 4, no. 12, pp. 2379–2394, 1987.

[77] P. Yao, J. Li, X. Ye, Z. Zhuang, and B. Li, "Iris recognition algorithm using modified log-gabor filters," in *18th International Conference on Pattern Recognition (ICPR'06)*, vol. 4, pp. 461–464, IEEE, 2006.

[78] P. Burt and E. Adelson, "The laplacian pyramid as a compact image code," *IEEE Transactions on communications*, vol. 31, no. 4, pp. 532–540, 1983.

[79] M. Norouzi, D. J. Fleet, and R. R. Salakhutdinov, "Hamming distance metric learning," in *Advances in neural information processing systems*, pp. 1061–1069, 2012.

[80] Y. Zhu, T. Tan, and Y. Wang, "Biometric personal identification based on iris patterns," in *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, vol. 2, pp. 801–804, IEEE, 2000.

[81] N. Liu, H. Li, M. Zhang, J. Liu, Z. Sun, and T. Tan, "Accurate iris segmentation in non-cooperative environments using fully convolutional networks," in *2016 International Conference on Biometrics (ICB)*, pp. 1–8, IEEE, 2016.

[82] D. Crockford, "The application/json media type for javascript object notation (json)," tech. rep., 2006.

[83] G. Van Rossum and F. L. Drake, "Python language reference manual," 2003.

[84] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[85] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," 2014.

[86] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, "Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

[87] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding data augmentation for classification: when to warp?," in *2016 international conference on digital image computing: techniques and applications (DICTA)*, pp. 1–6, IEEE, 2016.

[88] L. Taylor and G. Nitschke, "Improving deep learning using generic data augmentation," *arXiv preprint arXiv:1708.06020*, 2017.

[89] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[90] D. R. Lucio, R. Laroca, E. Severo, A. S. Britto, and D. Menotti, "Fully convolutional networks and generative adversarial networks applied to sclera segmentation," in *2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pp. 1–7, IEEE, 2019.

[91] "NICE-I Evaluation Protocol." http://nice1.di.ubi.pt/evaluation.htm. Last accessed 2018-05-14.

[92] H. Proença and L. A. Alexandre, "Ubiris: A noisy iris image database," in *International Conference on Image Analysis and Processing*, pp. 970–977, Springer, 2005.

[93] H. Yuen, J. Princen, J. Illingworth, and J. Kittler, "Comparative study of hough transform methods for circle finding," *Image and vision computing*, vol. 8, no. 1, pp. 71–77, 1990.

[94] C. Rathgeb, A. Uhl, P. Wild, and H. Hofbauer, "Design decisions for an iris recognition sdk," in *Handbook of Iris Recognition* (K. Bowyer and M. J. Burge, eds.), Advances in Computer Vision and Pattern Recognition, Springer, second edition ed., 2016.

[95] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-CNN: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1137–1149, jun 2017.

[96] Y.-H. Li and P.-J. Huang, "An accurate and efficient user authentication mechanism on smart glasses based on iris recognition," *Mobile Information Systems*, vol. 2017, pp. 1–14, 2017.

[97] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition,"

[98] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 2481–2495, dec 2017.

[99] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions,"

[100] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[101] P. Nishad and R. Manicka Chezian, "Various colour spaces and colour space conversion," *International Journal of Global Research in Computer Science (UGC Approved Journal)*, vol. 4, pp. 44–48, 01 2013.

[102] G. H. Joblove and D. Greenberg, "Color spaces for computer graphics," in *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '78, (New York, NY, USA), pp. 20–25, ACM, 1978.

[103] J. Zwinkels, "Light, electromagnetic spectrum," *Encyclopedia of Color Science and Technology*, pp. 1–8, 01 2015.

[104] H. W. Siesler, Y. Ozaki, S. Kawata, and H. M. Heise, *Near-infrared spectroscopy: principles, instruments, applications*. John Wiley & Sons, 2008.

[105] P. Joshi, *Artificial Intelligence with Python*. Packt Publishing Ltd, 2017.

[106] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 3, pp. 210–229, jul 1959.

[107] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[108] N. Le Roux, Y. Bengio, and A. Fitzgibbon, "Improving first and second-order methods by modeling uncertainty," 2011.

[109] A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019.

[110] L. Deng, D. Yu, *et al.*, "Deep learning: methods and applications," *Foundations and Trends® in Signal Processing*, vol. 7, no. 3–4, pp. 197–387, 2014.

[111] M. van Gerven and S. Bohte, "Editorial: Artificial neural networks as models of neural information processing," *Frontiers in Computational Neuroscience*, vol. 11, dec 2017.

[112] T. Huang, "Computer vision: Evolution and promise," 1996.

[113] L. Xu, M. Jackowski, A. Goshtasby, D. Roseman, S. Bines, C. Yu, A. Dhawan, and A. Huntley, "Segmentation of skin cancer images," *Image and Vision Computing*, vol. 17, pp. 65–74, jan 1999.

[114] M. R. Endsley, "Autonomous driving systems: A preliminary naturalistic study of the tesla model s," *Journal of Cognitive Engineering and Decision Making*, vol. 11, no. 3, pp. 225–238, 2017.

[115] M. Pesaresi and J. A. Benediktsson, "A new approach for the morphological segmentation of high-resolution satellite imagery," *IEEE transactions on Geoscience and Remote Sensing*, vol. 39, no. 2, pp. 309–320, 2001.

[116] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[117] Z.-Q. Zhao, P. Zheng, S. tao Xu, and X. Wu, "Object detection with deep learning: A review,"

[118] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

[119] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," *CoRR*, vol. abs/1608.06993, 2016.

[120] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," 2016.

[121] W. Abdulla, "Mask r-cnn for object detection and instance segmentation on keras and tensorflow." https://github.com/matterport/Mask_RCNN, 2017.

[122] "NVIDIA Corporation, Data Sheet: Quadro M6000 24GB, 2016. dirección:." https://www.nvidia.com/object/quadro-product-literature.html. Last accessed 2019-11-26.

[123] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International journal of computer vision*, vol. 111, no. 1, pp. 98–136, 2015.

[124] A. B. Jung, K. Wada, J. Crall, S. Tanaka, J. Graving, S. Yadav, J. Banerjee, G. Vecsei, A. Kraft, J. Borovec, C. Vallentin, S. Zhydenko, K. Pfeiffer, B. Cook, I. Fernández, W. Chi-Hung, A. Ayala-Acevedo, R. Meudec, M. Laporte, *et al.*, "imgaug." https://github.com/aleju/imgaug, 2019. Online; accessed 25-Sept-2019.

[125] A. Gulli and S. Pal, *Deep Learning with Keras*. Packt Publishing Ltd, 2017.

[126] Z. Zhang and M. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," in *Advances in neural information processing systems*, pp. 8778–8788, 2018.