UNIVERSITAT POLITÈCNICA DE VALÈNCIA

MIARFID Máster en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

DSIC DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Identificación, segmentación y regresión de elementos para modelado 3D de pies

Trabajo Fin de Máster

Máster en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Autor: Joaquín Sanchiz Navarro

Tutor: Francisco José Abad Cerdá

Tutor externo: Eduardo Parrilla Bernabé

2020-2021

Resumen

En este trabajo se describe el desarrollo de un sistema de segmentación capaz de clasificar a nivel de pixel las clases fondo, folio, pie izquierdo y pie derecho. Comienza con un modelo de segmentación PSPNet, que irá evolucionando a medida que se va avanzando en su estudio. El resultado será un mapa de segmentación con los identificadores de cada clase como valor de pixel. Además, se realizará un proceso de localización de las esquinas del folio para conocer su posición en el espacio 2D mediante una regresión del mapa de calor de los keypoints, definidos como una gaussiana 2D de valores de probabilidad de apariencia. Con el mapa de segmentación y la posición de las esquinas, se puede conocer el tamaño del pie relativo al folio y sus medidas antropométricas para un modelado 3D del mismo, siendo este el objetivo final del proyecto. Los modelos implementados se desarrollarán buscando optimizar los resultados y el rendimiento, para un funcionamiento robusto, rápido y preciso.

Palabras clave: segmentación, regresión de keypoints, aprendizaje profundo, redes neuronales convolucionales, modelado 3D

Abstract

This project describes the development of a segmentation system capable of classifying between background, paper, left foot and right foot classes at a pixel level. It will begin with a proposed segmentation model, PSPNet, which will evolve as the project progresses. The result will be a segmentation map with the identifiers of each class as pixel value. In addition, a task of locating the corners of the folio will be carried out to know their position in 2D space through a regression of the heat map of the keypoints, defined as a 2D Gaussian of appearance probability values. With the segmentation map and the position of the corners, it is possible to know the size of the foot relative to the sheet and its anthropometric measurements for a 3D modeling, this being the goal of the project. The implemented models will be developed in order to optimize results and performance, for robust, fast and accurate operation.

Keywords : segmentation, keypoint regression, deep learning, convolutional neural networks, 3D modeling.

Tabla de contenidos

1. Introducción 10
1.1 Descripción del problema 10
1.2 Objetivos 11
1.3 Análisis 11
1.4 Estructura del proyecto 11
2. Estado del arte 13
2.1 Segmentación de imagen 13
2.1.1 Segmentación semántica 13
2.1.2 Segmentación de instancia 14
2.2 Redes neuronales convolucionales 15
2.3 Convolución 15
2.3.1 Deconvolución o convolución traspuesta 15
2.3.2 Convolución dilatada 17
2.3.3 MobileConv o SeparableConv 17
2.4 Historia de las redes convolucionales de segmentación 18
2.4.1 Alexnet 18
2.4.2 VGG 18
2.4.3 ResNet 18
2.4.4 Fully Convolutional Network 19
2.4.5 SegNet 20
2.4.6 DeepLab 20
2.4.7 PSPNet 20
2.5 Localización de puntos clave 20
2.5.1 Pose Machines (2014) 21
2.5.2 Convolucional Pose Machines 21
2.5.3 Stacked Hourglass Networks for Human Pose Estimation 21
2.5.4 Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields 22
2.5.5 Cascaded Pyramid Network 22
2.5.6 Simple pose 23
3. Conjunto de datos 24
3.1 Conjunto de datos de segmentación 24

3.1.1 Tamaño de entrada 24
3.1.2 Normalización de las imágenes de entrada 25
3.1.3 Datos del ground truth 27
3.1.4 Limpieza de ground truth 27
3.1.5 Codificación 28
3.1.6 Problemas en los datos de segmentación 28
3.2 Conjunto de datos de predicción de puntos clave 30
3.2.1 Heatmaps 30
4. Modelos 31
4.1 Modelo de segmentación propuesto 31
4.1.1 Backbone 32
4.1.2 Pyramid Pooling Module 33
4.1.3 Función de pérdida 33
4.1.4 Optimizador 33
4.2 Modelo de localización de keypoints propuesto 35
4.2.1 SimplePose 35
4.2.2 Función de pérdida 36
4.2.3 Optimizador 36
5. Desarrollo de modelos finales 37
5.1 Estudio del modelo de segmentación 37
5.1.1 Métricas 37
5.1.2 Evolución del modelo de PSPNet 38
5.2 Estudio del modelo de localización de puntos clave 47
5.2.1 Métricas 47
5.2.2 Evolución del modelo de regresión de puntos 48
6. Conclusiones 53
7. Trabajos futuros 54
8. Bibliografía 55

Tabla de figuras

Figure 1. Tipos de algoritmos de segmentación [3] 13
Figure 2: Etiquetas semanticas [4] 13
Figure 3: Segmentación de imagen [4] 14
Figure 4: Segmentación de instancia [5] 14
Figure 5: Areas de Brodmann [7] 15
Figure 6: Convolución [8] 15
Figure 7: Vecino más cercano [9] 16
Figure 8: Interpolación bi-lineal [10] 16
Figure 9: Bed of Nails [9] 16
Figure 10: Convolución dilatada [11] 17
Figure 11: Separable Convolution [12] 17
Figure 12: Conexion residual [17] 18
Figure 13: Bottleneck [17] 19
Figure 14: Keypoint Heatmaps [22] 21
Figure 15: Supervisión intermedia [23] 21
Figure 16: Modulo Hourglass [25] 22
Figure 17: Secuencia de modulos [26] 22
Figure 18: Comparativa entre modulos encoder-decoder [27] 23
Figure 19: Entrada y ground truth 24
Figure 20: Distribución de los valores de pixel sin normalizar para muestra de 1000 imágenes 26
Figure 21: Distribución de los valores de pixel normalizados para muestra de 1000 imágenes 26
Figure 22: Pixeles de clase medio por imagen 27
Figure 23: Errores de pixel 27
Figure 24: Corrección de errores de pixel mediante vecino más cercano 28
Figure 25: Representación de clusters 29
Figure 26: Ejemplo de etiquetas semánticas mediante clustering 29
Figure 27: Heatmaps del folio 30
Figure 28: Ejemplo de mapa de segmentacion de CityScapes [28] 31
Figure 29: Arquitectura de PSPNet [21] 32
Figure 30: Perdida supervisada de PSPNet [21] 32
Figure 31: Pyramid Pooling Module [21] 33
Figure 32: Representación del descenso por gradiente 34
Figure 33: Estructura encoding-decoding de SimplePose [27] 35
Figure 34: Heatmap en puntos clave 36
Figure 35: Representación del dice coef 37
Figure 36: Matriz de confusión del modelo base de PSPNet 38
Figure 37: A la izquierda la entrada, posteriormente el ground truth y al final la salida 39
Figure 38: Precisión modelo 42
Figure 39: Gráfico de precision de arquitecturas [29] 43
Figure 40: Precisión de modelo 43
Figure 41: Precisión de modelo 44
Figure 42: Resultado de segmentación 45

Figure 43: Resultado de segmentación 45
Figure 44: Resultado de segmentación 46
Figure 45: Mapa de error de etiqueta 46
Figure 46: Comparativa de tiempos de inferencia entre el modelo base y modelo final (tiempo/imagen) 47
Figure 47: Pérdida del modelo 49
Figure 48: Predicción de salida 49
Figure 49: Predicción errónea sobre punto ocluido 49
Figure 50: Pérdida con Focal Loss 50
Figure 51: Predicción del modelo 50
Figure 52: Pérdida de modelo con OHKM 51
Figure 53: Error en predicción de punto ocluido 51
Figure 54: Predicción correcta de punto ocluido 51
Figure 55: Predicción con ResNet50 de backbone 52
Figure 56: Predicción con disminucion de tamaño de gaussiana 52
Figure 57: Desplazamiento en la segmentación 54

1. Introducción

Este trabajo de fin de master se encuadra en un proyecto más ambicioso que busca obtener una reconstrucción 3D del pie del usuario a partir de un conjunto de fotografías realizadas por un dispositivo móvil mediante el uso de la app 3D Avatar Feet [1].

El Instituto de Biomecánica (IBV) es un centro tecnológico que estudia el comportamiento del cuerpo humano y su relación con los productos, entornos y servicios que utilizan las personas. El inicio de la actividad del centro se remonta a 1976 y en la actualidad el instituto es un centro concertado entre el Instituto Valenciano de Competitividad Empresarial (IVACE) y la Universitat Politècnica de València (UPV).

Concretamente, este trabajo tiene dos objetivos. El primero es la creación y aplicación de un sistema capaz de segmentar fotografías realizadas por cualquier dispositivo en 4 clases diferentes: fondo, folio, pie izquierdo y pie derecho. El objetivo de la segmentación es realizar la extracción de la información necesaria para calcular el tamaño del folio y del pie para una posterior reconstrucción 3D de la extremidad del sujeto.

Debido a lo concreto del dominio de este problema, es necesario utilizar una base de datos antropométrica creada por el Instituto de Biomecánica de Valencia a lo largo de varios años, para poder utilizar estos datos en un entrenamiento de modelos de aprendizaje automático.

El segundo objetivo del proyecto es la localización dentro de la imagen de las esquinas del folio, con la idea de obtener información para el modelado 3D del sujeto. Las imágenes recibidas por el sistema presentan una alta variabilidad, por factores como la forma del pie, la orientación, la distancia frente a la cámara, iluminación, color de piel, errores de usuario, características del dispositivo, etc. Debido a esta complejidad, es necesaria la aplicación de técnicas de *machine learning* que ofrezcan mayor robustez frente al problema en comparación con las técnicas clásicas, sensibles a todos estos factores. El fin en sí mismo es integrar todas estas técnicas en la aplicación de modelado 3D del pie para dispositivos móviles.

1.1 Descripción del problema

La reconstrucción 3D de objetos a partir de imágenes 2D ha recibido mucho interés en los últimos años [2]. Obtener un modelo 3D de partes del cuerpo humano como las extremidades habilita la capacidad de estudiarlo, digitalizarlo, adaptar la ergonomía de sistemas que lo tienen en cuenta e incluso la posibilidad de rehacerlo físicamente y que guarde una alta similitud con la realidad. Para que todas estas reconstrucciones tengan una utilidad real, es necesario realizar un tratamiento de la información para trabajar con los datos de la manera adecuada.

Desde el Instituto de Biomecánica de Valencia, se ha venido creando aplicaciones para realizar reconstrucciones 3D mediante la toma de varias fotografías que revelen datos como la orientación, posición de la cámara, tamaño relativo, etc. En este proyecto se trabajará en el caso concreto de la reconstrucción 3D del pie.

Para generar un modelo 3D del pie, el IBV cuenta con una aplicación guiada que permite sacar tres imágenes, dos vistas laterales y una vista superior. Para que el proceso funcione, el pie debe estar colocado sobre un folio A4 o formato americano. Conociendo la localización de las esquinas del folio, se puede conocer los parámetros de la vista de la cámara, permitiendo así crear un modelo más preciso.

1.2 Objetivos

Los objetivos principales a tratar en este trabajo son los siguientes:

- Realizar un modelo capaz de segmentar en 4 clases las imágenes recogidas por la aplicación y capaz de identificar la posición de las 4 esquinas del folio, mediante librerías de machine learning.
- Mejorar los resultados de las técnicas clásicas de segmentación y localización de esquinas en cuanto a precisión y velocidad.
- Realizar una comparación y describir la evolución de los diferentes modelos que han sido tratados, teniendo en cuenta parámetros como precisión, coste computacional, tiempo de inferencia etc.

1.3 Análisis

Para mejorar de forma robusta los métodos de segmentación y localización de esquinas propuestos por el proyecto, encontramos las siguientes restricciones:

- Las alteraciones de brillo, contraste y otros atributos de las imágenes que se salgan de lo común pueden resultar en salidas problemáticas del modelo clásico actual.
- Imágenes con una resolución pobre pueden dar lugar a segmentaciones poco precisas.
- Imágenes movidas pueden dar lugar a fallos en el proceso
- La existencia de un modelo muy pesado puede dar lugar a respuestas lentas, y se necesitan soluciones rápidas para el usuario de la aplicación.
- Los servicios desarrollados se desplegarán en servidores donde se intentará minimizar el costo, por lo que adaptar los modelos a procesadores sin GPU reducirá gastos de mantenimiento.

1.4 Estructura del proyecto

La memoria se ha estructurado en los siguientes capítulos:

- Se realizará un repaso del estado del arte de las diferentes técnicas de segmentación y de localización de puntos clave, así como del uso de las herramientas necesarias para llevar a cabo estas tareas y los modelos y técnicas que han sido utilizadas para resolver estos problemas anteriormente.
- Posteriormente se describirá cómo se han tratado los datos para su uso por los diferentes modelos y para la obtención de los mejores resultados.

- A continuación, se detallarán los modelos propuestos para la resolución de dichas tareas, describiendo los diferentes módulos que los componen.
- Una vez se ha descrito el funcionamiento de los modelos, se describirá las modificaciones realizadas para adaptar estas herramientas a nuestro problema, procurando mejorar factores como la precisión, eficiencia, tiempo de inferencia y coste computacional. Se estudiará y comparará cada modificación para intentar guiar el proyecto a un resultado óptimo.
- Finalmente se presentarán las conclusiones y el trabajo futuro de este proyecto a modo de cierre.

2. Estado del arte

En el siguiente apartado, se describen las diferentes técnicas de segmentación y detección de puntos clave existentes, pasando por las herramientas utilizadas para llevar a cabo dichas tareas.

2.1 Segmentación de imagen

En Visión por Computador, la segmentación de imagen es una forma de disgregar una imagen digital en múltiples regiones de acuerdo a diferentes propiedades a nivel de pixel. A diferencia de la clasificación y detección de objetos, es normal enfocar este problema como una tarea a nivel de pixel, ya que la información contextual de la imagen es muy importante a la hora de segmentar diferentes regiones. La segmentación permite extraer información de interés para facilitar procesos de análisis.

Hay dos tipos principales de algoritmos de segmentación de imagen: segmentación semántica y segmentación de instancia. Existe otro tipo llamada segmentación panóptica, que es la versión unificada de los dos procesos de segmentación básicos [*Figure* 1].

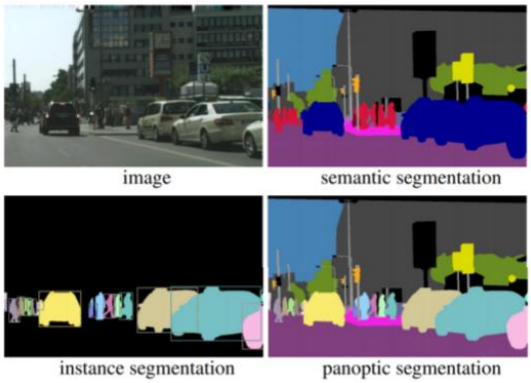


Figure 1. Tipos de algoritmos de segmentación [3]

2.1.1 Segmentación semántica

La segmentación semántica describe el proceso de asociar una etiqueta de clase a cada pixel de una imagen [*Figure* 2].

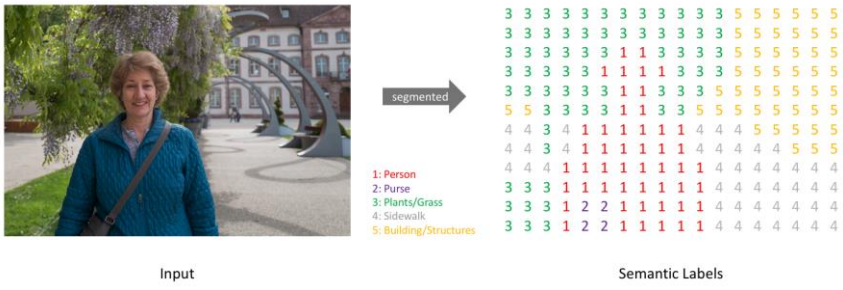


Figure 2: Etiquetas semanticas [4]

Dicho proceso se utiliza para reconocer conjuntos de píxeles agrupados en diferentes categorías y de diferentes formas. Por ejemplo, un vehículo de conducción autónoma necesita identificar vehículos, peatones, señales de tráfico, aceras y otros elementos de la carretera.

Este tipo de técnicas no solo se utilizan para vehículos autónomos. Su ámbito abarca imagen médica, inspección industrial, clasificación de terreno visible desde imágenes de satélite, y visión robótica, entre otras [*Figure* 3].

Figure 3: Segmentación de imagen [4]

La segmentación semántica se diferencia de la detección de objetos en que no se predice ninguna caja de inclusión alrededor de un elemento, ni tampoco se detectan diferentes instancias de la misma clase. En el caso de la , aparecen múltiples coches, pero todos llevan la misma etiqueta de clase. Este tipo de algoritmos son interesantes pues permiten que el objeto o clase de interés abarque diferentes áreas de la imagen a nivel de pixel, detectando objetos con formas irregulares que no encajarían en una caja de inclusión.

2.1.2 Segmentación de instancia

La segmentación de instancias [5] es el proceso mediante el cual buscamos detectar un objeto en una escena y generar una máscara que nos permita extraer con mayor precisión el objeto detectado, puede verse como el conjunto de dos procesos: primero detectar el área rectangular que contiene el objeto y luego obtener la máscara que segmenta dicho objeto [*Figure* 4].

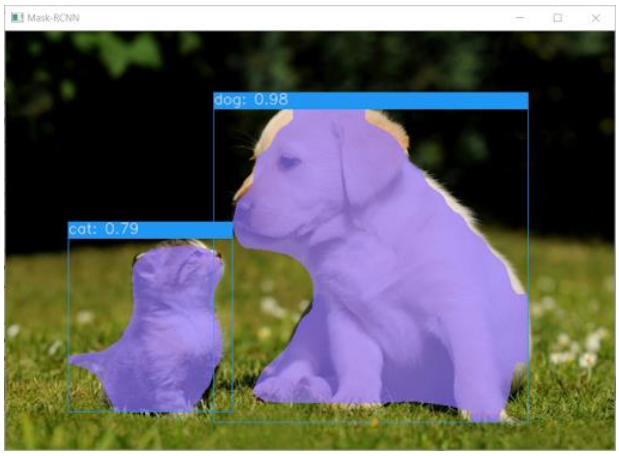


Figure 4: Segmentación de instancia [5]

2.2 Redes neuronales convolucionales

Después de muchos estudios sobre el cerebro de los mamíferos, los investigadores han encontrado que partes específicas del cerebro se activan reaccionando a diferentes tipos de estímulos. Por ejemplo, algunas zonas de las Áreas de Brodmann [*Figure* 5] se activan cuando ven formas verticales, otras partes cuando ven formas horizontales, y otras cuando ven formas específicas, colores, caras, etc [6].

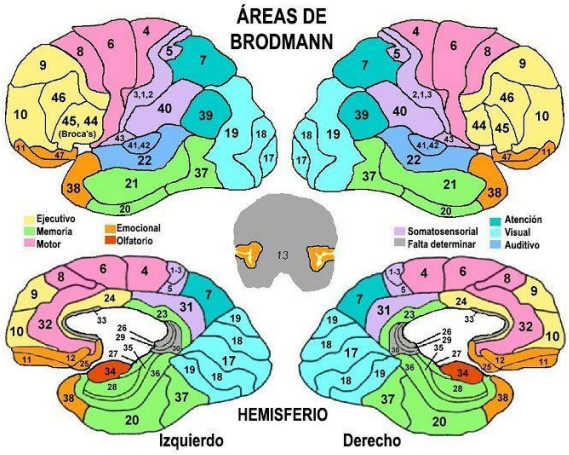


Figure 5: Areas de Brodmann [7]

Siguiendo esta idea, comienzan a aparecer las primeras Redes Neuronales Convolucionales, que son un tipo de redes neuronales cuyos bloques están compuestos por capas convolucionales. Una capa convolucional no es más que un conjunto de matrices denominados kernels o filtros que se usan para operaciones matemáticas en una matriz de características como una imagen.

2.3 Convolución

Una convolución 2D es una operación simple: se empieza con un filtro que se va desplazando sobre una entrada 2D, realizando una multiplicación sobre la entrada y el kernel. Dicho kernel repite el proceso para cada posición de la matriz por la que se desplaza, convirtiendo una matriz 2D de características en otra matriz 2D de características [Figure 6].

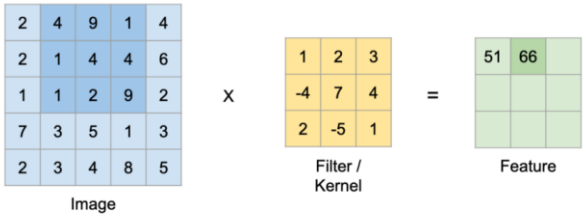


Figure 6: Convolución [8]

2.3.1 Deconvolución o convolución traspuesta

Una deconvolución es la operación inversa a la convolución. Con una convolución se puede aplicar un filtro, y usando una deconvolución sería posible recuperar la señal o imagen de entrada.

Es un proceso computacional generalmente usado para incrementar la dimensión de un mapa de características de entrada. Se diferencia de los métodos clásicos de upsampling en la posibilidad de aprender los parámetros del filtro que mejor mantienen la información.

- **Nearest Neighbor**: En el caso del vecino más cercano, como su nombre indica, se copia el valor de un pixel de entrada en una vecindad de tamaño "K", donde "K" depende de la salida esperada [*Figure* 7].

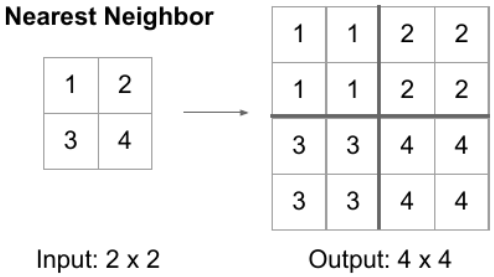


Figure 7: Vecino más cercano [9]

- **Bilineal**: En el caso de la interpolación bi-lineal, se produce una media ponderada basada en la distancia entre los pixeles de entrada y los pixeles de salida con dimensión aumentada [Figure 8].

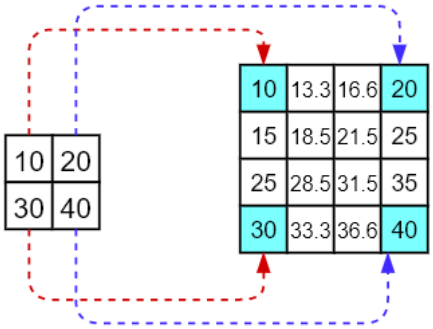


Figure 8: Interpolación bi-lineal [10]

- **Bed Of Nails**: En el caso del Bed Of Nails, se copian los valores de los pixeles de entrada en sus posiciones correspondientes y se rellena con 0 los pixeles restantes [Figure 9].

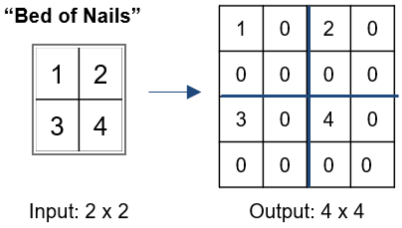


Figure 9: Bed of Nails [9]

2.3.2 Convolución dilatada

La operación de convolución dilatada es muy parecida a la convolución normal, salvo en que el filtro o kernel contiene una separación entre sus pixeles. Esta separación, conocida como ratio de dilatación, permite obtener información más contextual en cada filtro, algo muy interesante dentro del ámbito semántico. Como se puede observar en la Figura, a medida que se va aumentando el ratio de dilatación, los pixeles sobre los que se opera el filtro se van distanciando entre sí [Figure 10].

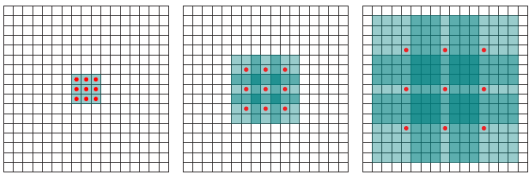


Figure 10: Convolución dilatada [11]

2.3.3 MobileConv o SeparableConv

Son un tipo de convolución especial de procesamiento de señal básico. Esta convolución realiza dos operaciones:

- Depthwise convolution: Inicialmente se realiza una convolución sin cambiar la cantidad de los filtros. Esto se hace mediante la convolución de 3 kernels. Cada kernel itera cada canal de la imagen individualmente.
- Pointwise convolution: Posteriormente, se produce una operación de convolución mediante un kernel de 1x1 (sobre el caso anterior, un filtro de 1x1x3). Si anteriormente se tenían 3 filtros, ahora se tiene 1. Creando, por ejemplo, 256 kernels de 1x1x3, se pueden obtener 256 filtros.
- Con una imagen de entrada de 12x12 RGB y 256 kernels de 5x5 para una imagen de salida de (8,8), la convolución original realiza 256x3x5x5x8x8 = 1.228.800 multiplicaciones.
- En la Separable Conv, en el proceso de depthwise convolution se realizan operaciones entre 3 kernels de 5x5x1 que se mueven 8x8 veces para una imagen de salida de (8,8). Esto son 3x5x5x8x8 = 4800 multiplicaciones. Posteriormente en la pointwise convolution, se operan 256 kernels de 1x1x3 8x8 veces. Esto son 256x1x1x3x8x8 = 49.152. Sumado al estado anterior son 52,052 operaciones comparado con 1.228.800 multiplicaciones para obtener la misma cantidad de filtros [Figure 11].

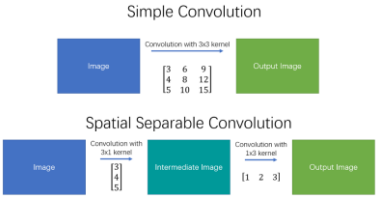


Figure 11: Separable Convolution [12]

2.4 Historia de las redes convolucionales de segmentación

2.4.1 Alexnet

En 2012 AlexNet [13] batió el récord de clasificación de ILSVRC, bajando el menor error alcanzado de 25.8% a 16.4%, hecho que marcó un antes y un después para la clasificación de imágenes y la visión por computador en general, ya que inició una explosión de diferentes arquitecturas convolucionales que bajarían el error a 3.57% en 4 años [14].

2.4.2 VGG

En 2014 GoogleNet y VGG [15] bajaron el error de AlexNet a 6.7% y 7.3% respectivamente.

2.4.3 ResNet

En 2015, la ResNet [16] de Microsoft con conexiones residuales superó el estado del arte bajando el error a 3.57%, no solo batiendo el récord de clasificación de imágenes, sino que batió el resto de algoritmos de 'Detección y Localización' y 'Segmentación Semántica'.

Las conexiones residuales [*Figure* 12] han supuesto un gran avance en la visión por computador. Este tipo de conexiones son especiales ya que son capaces de aprender funciones residuales con referencia a las conexiones de entrada, en vez de aprender funciones no referenciadas. Una conexión residual se basa en concatenar información de salida de una convolución con información de entrada, obteniendo como resultado una combinación de ambas informaciones. Esto de cara a problemas de segmentación es muy interesante, ya que la relación entre la entrada y la salida es muy alta, por lo que se entrena una cierta modificación de la entrada.

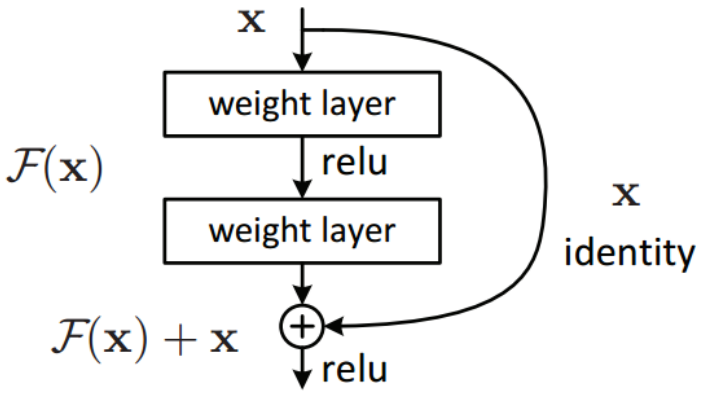


Figure 12: Conexion residual [17]

Este tipo de conexiones también permiten un mayor flujo del gradiente de propagación del error, que en redes con mucha densidad de capas se ve desvanecido por las funciones de activación de las neuronas (ver apartado siguiente).

Desvanecimiento de gradiente

En machine learning, el problema de desvanecimiento de gradiente es la dificultad encontrada para entrenar redes neuronales mediante métodos de aprendizaje basados en el descenso por gradientes y de retro propagación del error. En tales métodos, cada uno de los pesos de la red neuronal recibe una actualización proporcional a la derivada parcial de la función del error con respecto al peso actual en cada iteración del entrenamiento.

En algunos casos, el gradiente de error se va desvaneciendo a valores muy pequeños, impidiendo que el peso cambie su valor.

La ResNet también incluye los bloques conocidos como bottleneck [*Figure* 13]. Son módulos residuales que utilizan convoluciones 1x1 para crear un cuello de botella. El uso de este tipo de módulos reduce el número de parámetros y de producto de matrices. La idea es hacer bloques residuales lo más finos posibles para incrementar la profundidad de la red y tener menos parámetros.

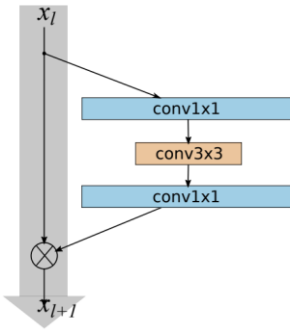


Figure 13: Bottleneck [17]

2.4.4 Fully Convolutional Network

En 2014 aparece también la FCN [18] (Fully Convolutional Network for Semantic Segmentation), que populariza el uso de redes convolucionales end-to-end para la segmentación semántica. Este tipo de redes siguen una arquitectura encoder-decoder, donde se produce una extracción de características en el proceso de encoding y se opera la predicción al final del decoder. Los autores de FCN observan que las capas fully-connected pueden ser vistas como una convolución con filtros que cubre toda la entrada.

Con los modelos de extracción de características como VGG, los mapas de características todavía necesitan ser ampliados debido a la reducción dimensional del pooling, que va bajando el tamaño del filtro drásticamente. En vez de usar interpolación bilineal, se introducen las capas de deconvolución, filtros que aprenden la interpolación minimizando el error. Estas capas producen segmentaciones erróneas debido a la pérdida de información que se produce en la reducción de la dimensionalidad del pooling. Es por esto por lo que se empiezan a introducir conexiones puente o shortcuts para obtener mejores resultados añadiendo información contextual previa a la deconvolución.

2.4.5 SegNet

En 2015 se publica ***SegNet*** [19] (Encoder-decoder para segmentación de imagen) en donde, a diferencia de FCN, se introducen shortcuts que conectan los pools del encoder con las deconvoluciones del decoder, consiguiendo una segmentación más eficiente, pero con peores resultados.

2.4.6 DeepLab

A finales de 2015 se empiezan a utilizar las convoluciones dilatadas en el artículo "Multi-Scale Contect Aggregation by Dilated Convolutions" [20], que usa dichos filtros para predicción densa, proponiendo un módulo de contexto que usa convoluciones dilatadas para la agregación a múltiples escalas.

DeepLab versión 2 sale a la luz en el artículo "Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution and Fully Connected CRFs" en 2016 usando convoluciones dilatadas y proponiendo una pirámide de dilatación espacial para la extracción de características a nivel contextual.

2.4.7 PSPNet

A finales de 2016 PSPNet "Pyramid Scene Parsing Network" [21] propone un módulo de agrupamiento en pirámide para agregar el contexto. El módulo de agrupamiento en pirámide permite capturar esta información aplicando grandes filtros en las capas de agrupamiento. Esta arquitectura utiliza una ResNet modificada con convoluciones dilatadas como extractor de características, luego un módulo de agrupamiento en pirámide que concatena los mapas de características de la ResNet con características ampliadas con filtros que cubren diferentes ratios de la imagen.

2.5 Localización de puntos clave

Conocer la localización de puntos clave o "keypoints" [*Figure* 14] de un objeto o imagen es una tarea que puede servir para problemas complejos de clasificación e identificación, concretamente en aquellos donde existe una gran variación de formas diferentes que influyen en la apariencia visual de la información, como identificar animales salvajes, conocer la pose de un cuerpo en un instante determinado o localizar puntos de interés en caras para tareas de reconocimiento facial.

La tarea de detección de keypoints se formula típicamente de manera diferente a la tarea de detección de objetos con regresión de bounding boxes. Está demostrado que es muy difícil realizar una regresión a nivel de pixel, por lo que este tipo de tareas se formula como un problema de estadística en el que la salida del modelo debe contener la probabilidad de cada pixel de pertenecer al punto clave que se quiere estimar.

Esta información se aprende mediante mapas de calor de probabilidades asociadas a la apariencia de cada punto clave, generalmente mediante gaussianas con centro en el pixel que se quieren identificar. La obtención de dicha información no es para nada trivial, por lo que resulta muy costoso la obtención de este tipo de datos para entrenar modelos, en muchos casos teniendo que resolverse de manera supervisada.

Figure 14: Keypoint Heatmaps [22]

Se han realizado gran variedad de estudios de localización de puntos clave, la gran mayoría de ellos relacionados con la identificación de la pose humana, por lo que los datos que existen sobre problemas de localización de puntos clave tratan en su mayoría este problema (MPII y MS-COCO datasets).

2.5.1 Pose Machines (2014)

Uno de los primeros modelos [23] en estimar la pose humana mediante redes convolucionales. Propone una topología de modelo en cascada con supervisión intermedia [*Figure* 15] para ir mejorando la estimación de las predicciones.

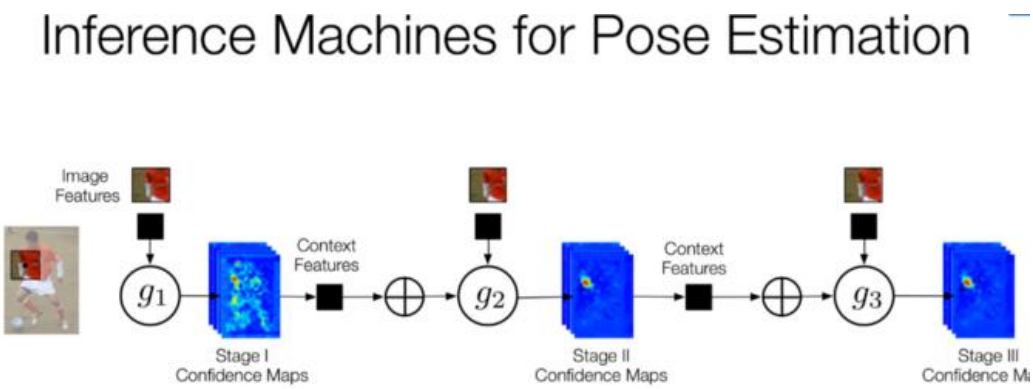


Figure 15: Supervisión intermedia [23]

Este modelo basado en random forest contiene módulos individuales que se encargan de aprender la posición de diferentes partes del cuerpo.

2.5.2 Convolucional Pose Machines

Modelo que sigue la idea del Pose Machines [24] de 2014 pero esta vez en vez de usar random forest usa redes neuronales convolucionales para aprender los parámetros.

2.5.3 Stacked Hourglass Networks for Human Pose Estimation

Al igual que la tarea de segmentación, la localización de puntos clave necesita tener en cuenta información contextual a diferentes niveles, por lo que Newell et al. [25] propone una estructura de tipo reloj de arena o Hourglass [*Figure* 16]. Las características son procesadas sobre todas las diferentes escalas y consolidadas posteriormente para capturar las relaciones espaciales asociadas al cuerpo. Este proceso se repite iterativamente en conjunto con supervisión intermedia para mejorar el funcionamiento de la red. En general, el modelo consiste en un conjunto de bloques encoding-decoding con módulos residuales que van capturando características locales como caras y manos a medida que se reduce la dimensionalidad y que posteriormente se relacionan en el proceso de decoding mediante concatenaciones de los bloques de encoding y de decoding realizando un proceso de upsampling en estos últimos para recuperar la dimensionalidad original y obtener características más semánticas.

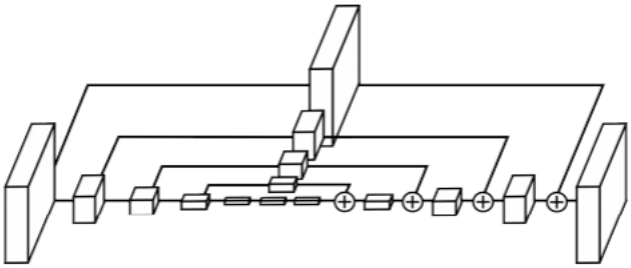


Figure 16: Modulo Hourglass [25]

Este bloque se repite en cascada con supervisión intermedia para ir mejorando la precisión de la red a medida que avanzan los bloques, terminando con la predicción de los mapas de calor de los keypoints.

2.5.4 Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields

Este modelo sigue la idea de los bloques Hourglass con supervisión intermedia [26], pero además es capaz de realizar una asociación entre los diferentes puntos clave denotando así segmentos y esqueletos humanos en tiempo de inferencia. El modelo se divide en dos ramas, una primera rama que realiza una predicción de los puntos clave y una segunda rama que codifica en forma de "PAFs" o "Part Affinity Fields" las agrupaciones de puntos clave en base a su dirección relativa con respecto a los diferentes segmentos. Esta arquitectura es capaz de estimar 17 puntos por cada esqueleto humano y es capaz de detectar múltiples poses en una sola inferencia [*Figure* 17].

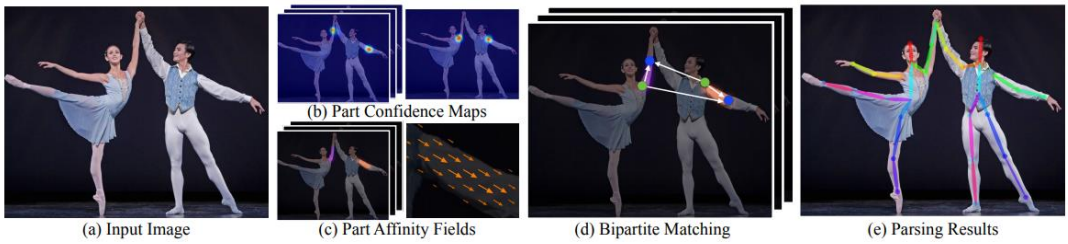


Figure 17: Secuencia de modulos [26]

2.5.5 Cascaded Pyramid Network

Este modelo utiliza un enfoque parecido al propuesto por Newell et. al, pero se divide en dos partes. Primero se presenta un módulo llamado "GlobalNet" que utiliza la información extraída por cada bloque de un extractor de características, ResNet en este caso, para hacer la predicción de los puntos clave. El resultado de la predicción de cada bloque va a una rama de pérdida que computa el MSE para aprender la localización de los heatmaps de una forma "supervisada". La salida de estos bloques va a un segundo módulo denominado "RefineNet" que afina aún más el resultado mediante módulos bottleneck. Posterior a esta extracción de características, ocurre un upsampling para concatenar todos los filtros. Se pasarán por otra convolución que obtiene información del conjunto completo y supervisada por una pérdida MSE con Online Hard Keypoint Mining (OHKM).

Este tipo de pérdida permite la focalización sobre los casos más complejos del problema, los puntos ocluidos. Es una técnica que consiste en propagar la pérdida de

las predicciones que más pérdida generan, de esta manera se le "enseña" a la red los casos más complicados y obviamos los triviales.

2.5.6 Simple pose

Este modelo aparece como uno de los más sencillos para la estimación de la pose [27]. Se basa en un conjunto de capas de deconvolución añadidas a un extractor de características, ResNet en este caso. Los métodos anteriores combinan upsampling y parámetros de filtros convolucionales, en cambio Simple Pose combina ambas tareas en la deconvolución. Este modelo es supervisado por una pérdida L2 como los anteriores [Figure 18].

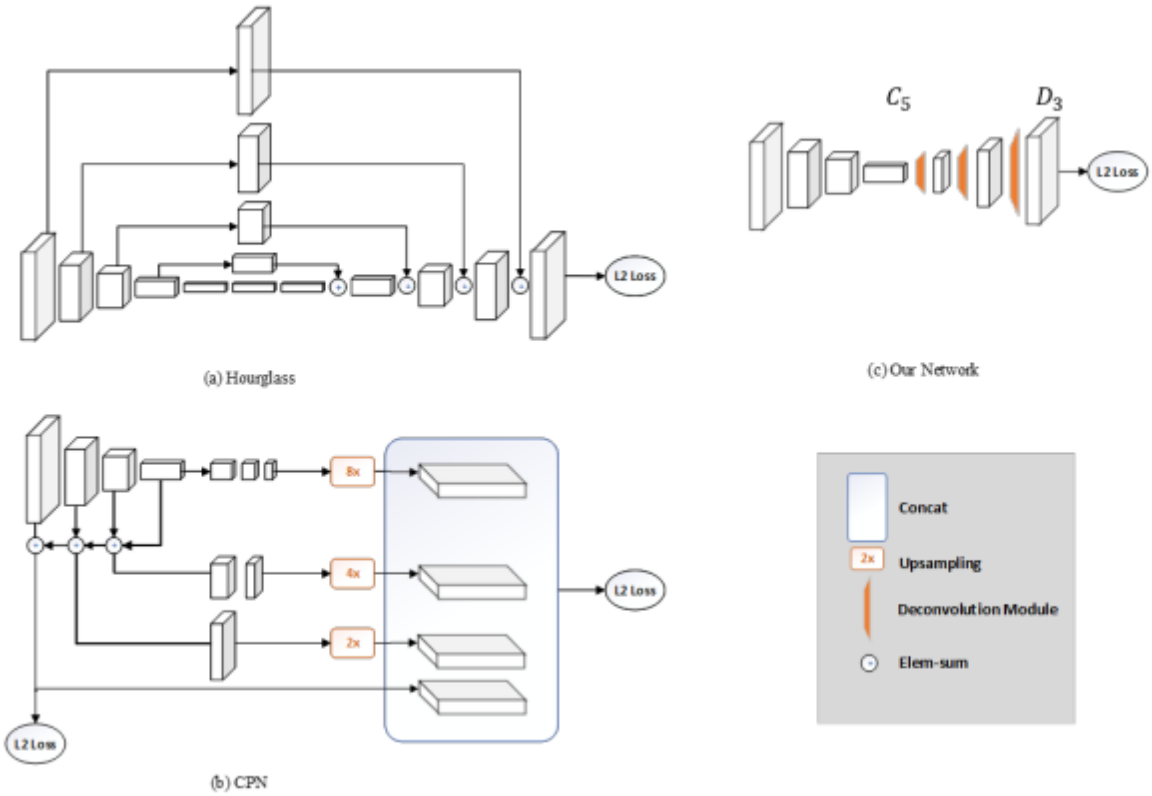


Figure 18: Comparativa entre modulos encoder-decoder [27]

3. Conjunto de datos

Para este proyecto se han utilizado datos proporcionados por el Instituto de Biomecánica de Valencia. Se han proporcionado dos conjuntos de datos diferentes, uno para segmentación y otro para localización de puntos clave. A continuación, se explican detalladamente los datos utilizados, su procesado y su normalización.

3.1 Conjunto de datos de segmentación

Para esta tarea se cuenta con 6084 imágenes que se han dividido en 4870 muestras de entrenamiento y 1214 muestras de validación. Estas imágenes están compuestas por una imagen de entrada y un ground truth como mapa de segmentación de salida [*Figure* 19].

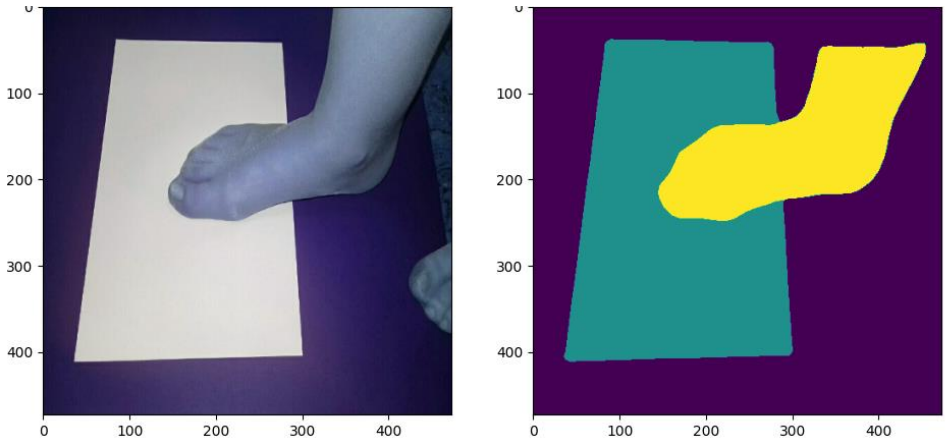


Figure 19: Entrada y ground truth

Estas imágenes son de tamaños que varían entre (160,120) y (400, 268) con valores de pixel RGB entre 0 y 255. Los mapas de segmentación contienen valores de pixel entre 0 y 3, correspondientes a las siguientes clases:

Id	Clase
0	Background
1	Folio
2	Pie izquierdo
3	Pie derecho

Table 1: Etiquetas de clase

Sobre estas imágenes se realizará un preprocesado que ayudará al modelo a extraer la información de manera adecuada.

3.1.1 Tamaño de entrada

El tamaño de las imágenes de entrada debe ser común, por lo que se debe escalar todo el dataset a un tamaño concreto. Se propone utilizar imágenes cuadradas ya que el extractor de características de segmentación que se utiliza en el estado del arte ("ResNet") utiliza imágenes cuadradas. Sobre el tamaño de entrada se propone (473,473), ya que es el tamaño de entrada del modelo inicialmente propuesto por los investigadores del IBV.

3.1.2 Normalización de las imágenes de entrada

Normalizar los datos es una técnica de preprocesado que suele formar parte de la preparación de los datos para el aprendizaje automático.

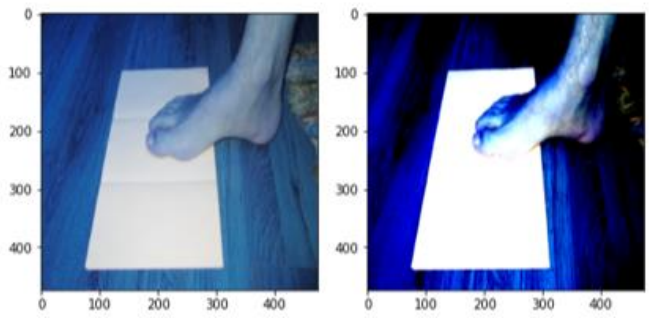
Consiste en cambiar los valores a una distribución más interesante sin distorsionar la relación de los valores.

Con la idea de proporcionar al modelo un conjunto de datos más sencillo de procesar, se analiza la variación de los valores de entrada. Dichos valores cubren un rango entre 0 y 255. Reducir la variación de los datos y proyectarlos entre 0 y 1 puede ayudar a la red a mejorar su precisión.

- **Estandarización**: Mediante la normalización por la media y la desviación típica, se escala nuestro dataset para que tenga una variación más baja. De esta manera seguirá las propiedades de una distribución normal estándar con media en 0 y desviación típica 1. Los valores normalizados se calculan de la siguiente forma:

$$z = \frac{x - \mu}{\sigma}$$

  Siendo x el valor del canal de la imagen (RGB), z el valor normalizado, μ la media y σ la desviación típica.
  Se puede observar que normalizando el dataset, los elementos como el folio y el pie aparecen más contrastados con el fondo que antes de la normalización.



- **Min-max**: otro tipo de normalización es la conocida como min-max, que escala los datos para que estén contenidos en un rango fijo, normalmente entre 0 y 1.

Se muestra a continuación una comparativa de la normalización por la media y la desviación típica sobre los canales de las imágenes [*Figure* 20] [*Figure* 21]:

Pixel	R	Rz	G	Gz	B	Bz
1	15	-0.97439764	26	-1.1569542	64	-0.79871122
2	14	-0.98871648	25	-1.17247691	63	-0.81455546

Table 2: Ejemplo de estandarizacion

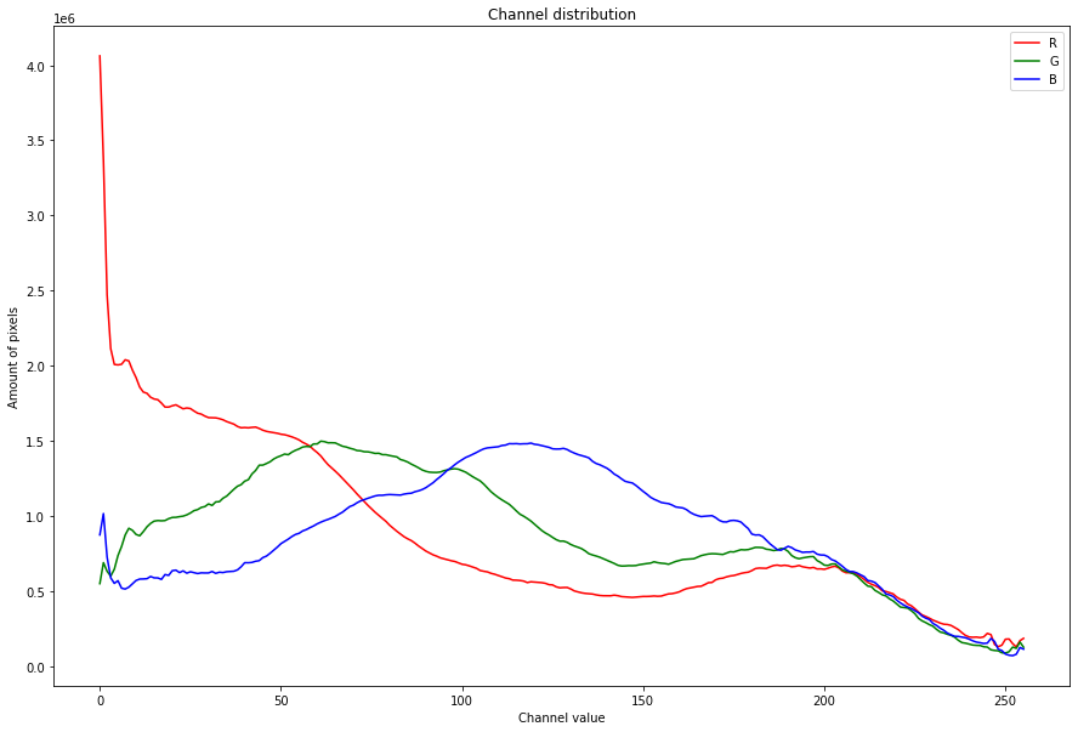


Figure 20: Distribución de los valores de pixel sin normalizar para muestra de 1000 imágenes

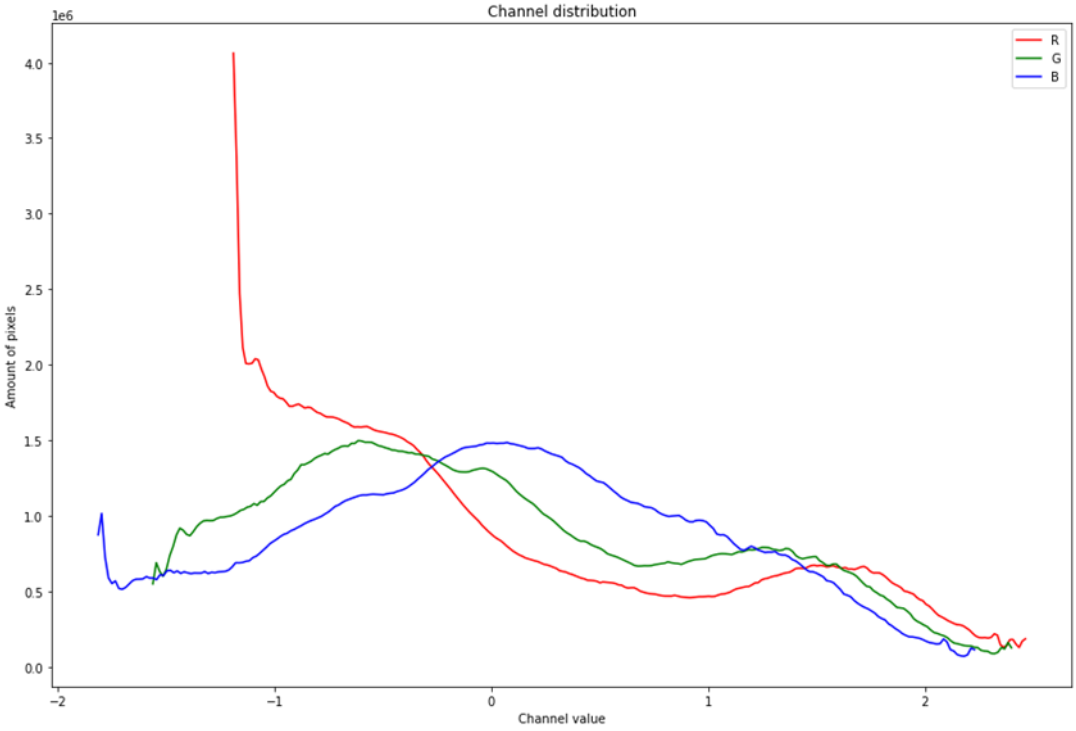


Figure 21: Distribución de los valores de pixel normalizados para muestra de 1000 imágenes

Hay que destacar que la normalización de los datos de entrada supone un gasto de memoria importante. Anteriormente se contaban con imágenes de la forma 473,473,3 que suponen 671.187 valores enteros entre 0 y 255 almacenables en variables tipo unsigned int de 8 bits, o sea 671Kb por imagen. Al normalizar las imágenes, se cuenta con 671.187 valores flotantes de 32 bits, o sea cada imagen pesa 4 veces más.

3.1.3 Datos del ground truth

Analizando la distribución de clases en los diferentes mapas de segmentación del dataset se obtiene que, de media por imagen, entorno a un 63% de los pixeles se corresponden a la clase 0 o background, un 25% se corresponden a la clase 1 o folio, y un 11% se corresponden a la clase 2 y 3, pie izquierdo o derecho [*Figure* 22].

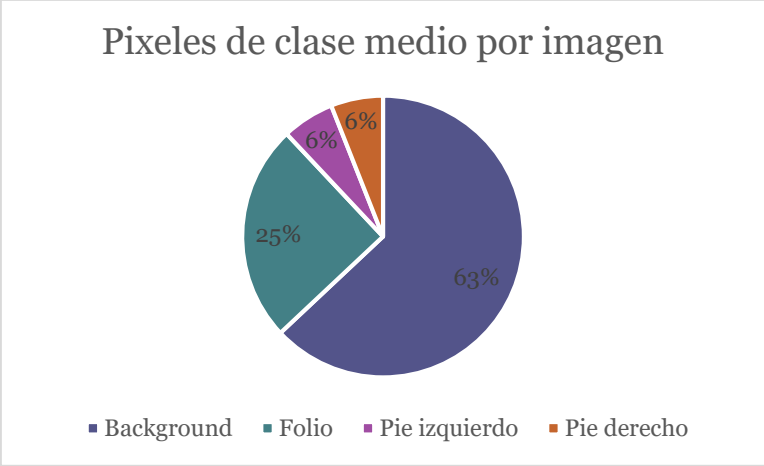


Figure 22: Pixeles de clase medio por imagen

Es importante tener en cuenta esta información, puesto que existe un desequilibrio de clases en el dataset, y esto puede hacer que el modelo aprenda más una clase que otra, favoreciendo a la clase con mayor cantidad de datos en puntos de duda, como bordes entre clases.

3.1.4 Limpieza de ground truth

Analizando de cerca los pixeles del ground truth, se encuentran errores en los límites de cada clase [*Figure* 23].

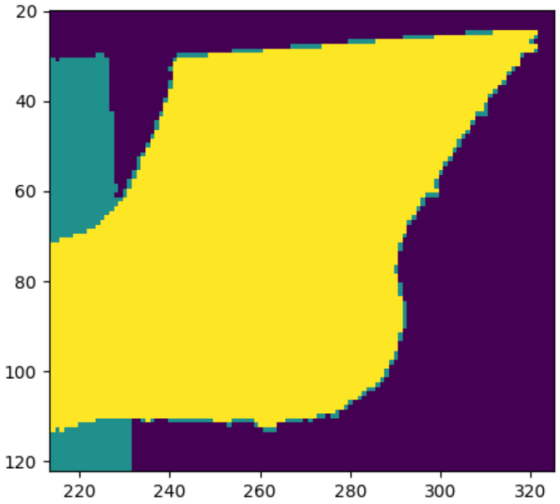


Figure 23: Errores de pixel

Esto se debe a que en el proceso de segmentación clásico que se utilizó para construir el dataset no se utilizó correctamente, posiblemente por una mala elección de los algoritmos de interpolación.

Por ello, se realizará una limpieza de todo el dataset analizando la vecindad de cada pixel con una ventana de 5 pixeles y asignándole a cada pixel analizado el valor del máximo correspondiente a dicha ventana, resultado en bordes limpios [Figure 24].

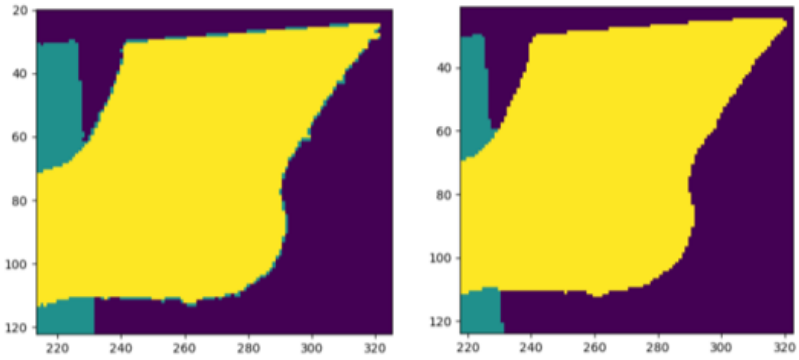


Figure 24: Corrección de errores de pixel mediante vecino más cercano

3.1.5 Codificación

El proceso de codificación es parecido al de normalización. Consiste en crear una representación de los datos de salida que el modelo pueda manejar. Como lo que interesa conocer es la probabilidad de cada pixel de pertenecer a una clase, se codificará la salida con una representación one-hot. La representación one-hot permite especificar qué neurona se activará para cada clase, o en nuestro caso, qué filtro se activará para cada clase. La codificación one-hot resulta de la siguiente manera:

Clase	Fondo	Folio	Pie izquierdo	Pie derecho
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1

Table 3: Codificación one-hot

De esta forma, se está especificando que la probabilidad de activación para el filtro 1 en la clase 0 es de 1, y la probabilidad de activación de las demás es 0. En cambio, la probabilidad de activación del filtro 2 para la clase 1 es 1, y la de las demás es 0. En situaciones reales, el valor de la clase que más probabilidad acumula será cercana a 1, y la probabilidad restante se repartirá entre los otros filtros, con valores cercanos a 0, siendo la suma de la probabilidad de todos los filtros igual a 1.

3.1.6 Problemas en los datos de segmentación

Normalmente se pueden encontrar los siguientes problemas en los datos para tareas de segmentación:

- Error en la creación del dataset: como se ha mencionado anteriormente, lo normal es que el dataset de segmentación no sea 100% perfecto. Los ground truth de las imágenes de pies presentaban

algunos errores, y los datasets como CityScapes presentan alrededor de un 17.6% de error cometido por el experto encargado en definir los mapas de segmentación según el artículo de la PSPNet [21].

- Los valores de pixel no son categóricos: a veces se pueden encontrar datasets que no presentan valores de pixel correspondientes a las etiquetas de clase. Por ejemplo, si existen 4 clases (rojo, verde, amarillo y azul) es posible que se encuentran valores de pixel correspondientes a dichos colores en los tres canales RGB, es decir, un pixel rojo tendría valores entorno a (255,0,0), rango que puede variar. Estos datos no sirven para problemas de segmentación, ya que no corresponden a etiquetas de clase. El caso de CityScapes [28] es un ejemplo. El dataset contenía muchos valores de pixel diferentes, por lo que para obtener los valores de etiquetas es necesario operar un algoritmo de clasificación automático como un clúster. Conociendo el número de clases, se puede entrenar un modelo de mixturas de gaussianas que sea capaz de agrupar los diferentes valores de pixel en clúster, que corresponderán nuestras etiquetas de clase [*Figure* 25].

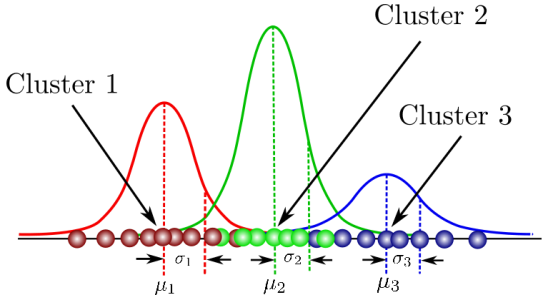


Figure 25: Representación de clusters

Estos datos presentaran errores, pero pueden servir como entrenamiento. La siguiente imagen muestra la salida de un modelo de mixturas de gaussianas aplicado sobre el conjunto de CityScapes. En él se pueden apreciar diferentes errores, sobre todo en fronteras de clases como farolas u otros elementos pequeños [*Figure* 26].

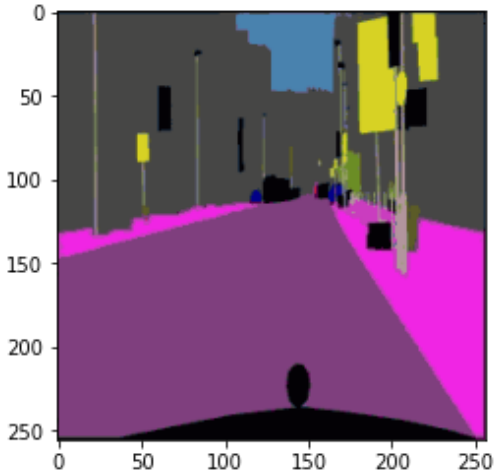


Figure 26: Ejemplo de etiquetas semánticas mediante clustering

3.2 Conjunto de datos de predicción de puntos clave

El conjunto de datos para la tarea de predicción de puntos clave está compuesto por las mismas imágenes de entrada, pero en este caso están acompañadas por 4 pares de valores que componen la información correspondiente a las esquinas del folio.

Inicialmente, se propuso realizar una regresión de los pares de valores asociados a cada esquina, pero existe tanta variación entre los posibles valores que es muy difícil que un modelo pueda aprender a regresar dichos valores. Por ello se aproxima este problema como una tarea de localización de heatmaps.

Para ello, es necesario crear los mapas de calor correspondientes a cada esquina sobre todo el dataset.

3.2.1 Heatmaps

Para la creación de los mapas de calor, se construye una gaussiana 2D con centro en el punto correspondiente a cada esquina y radio desconocido que se puede considerar como hiperparámetro. Si se elige un radio muy pequeño, al modelo le costará mucho aprenderlo, en cambio, si se elige un radio demasiado grande, al modelo le faltará precisión. Se genera un mapa por cada esquina, concatenando todos los mapas en un vector de forma (alto,ancho,4) al ser 4 esquinas. La mayor parte del mapa está constituido por valores de pixel cercanos a 0 y, a medida que se acerca a la posición del punto clave, los valores comienzan a acercarse a 1, teniendo como máximo el centro de la gaussiana [Figure 27].

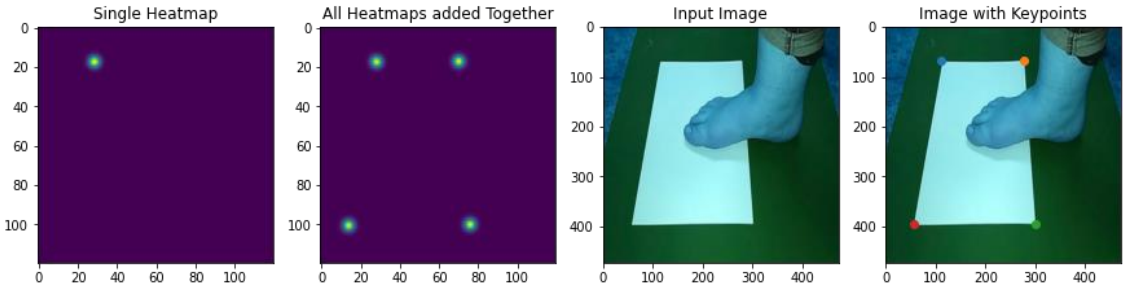


Figure 27: Heatmaps del folio

El tamaño del mapa de calor también es un hiperparámetro. Tamaños más grandes implicarán mayor coste computacional, y tamaños más pequeños implicarán un error añadido en el re-escalado del resultado.

4. Modelos

A continuación, se describe los modelos propuestos para solucionar los diferentes problemas. Se dividirá esta sección en la parte de segmentación, donde se tratará los aspectos relativos al modelado de la arquitectura para la tarea, y predicción de puntos clave.

4.1 Modelo de segmentación propuesto

El modelo propuesto por el Instituto de Biomecánica de Valencia es la Piramid Scene Parsing Network o PSPNet [21]. Se realizará una implementación en Keras de este modelo para la realización de pruebas.

La PSPNet es un modelo propuesto para solucionar uno de los principales problemas de las Fully Convolutional Networks, que es la falta de información contextual. Para un mejor entendimiento de la escena, se propone la topología en pirámide, que es capaz de combinar características globales y locales haciendo la predicción final más similar al caso real.

Este modelo se ha diseñado para tareas de segmentación complejas y gran cantidad de datos, como el CityScapes, un conjunto de datos de imágenes de diferentes ciudades europeas y su correspondiente mapa de segmentación. CityScapes proporciona la información necesaria para que un vehículo autónomo sea capaz de identificar los diferentes elementos que necesita conocer para poder efectuar su función, implicando la segmentación de alrededor de 30 clases [*Figure* 28].

Figure 28: Ejemplo de mapa de segmentacion de CityScapes [28]

PSPNet propone una estrategia de pérdida supervisada intermedia, algo que ayudará en la propagación del error, y una estrategia de convoluciones dilatadas sobre el extractor de características (ResNet en este caso) [*Figure* 29].

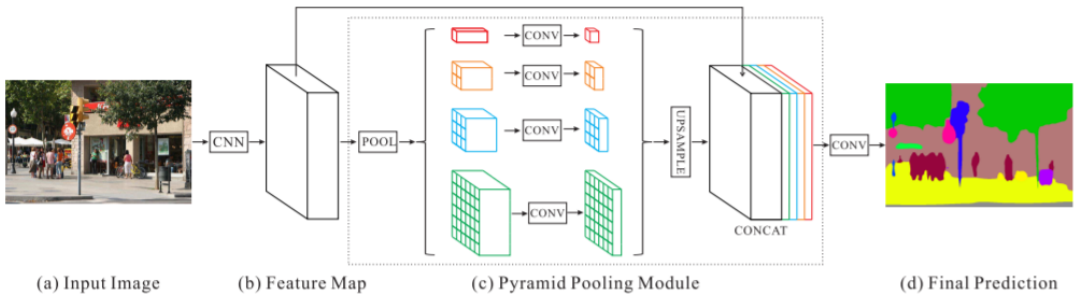


Figure 29: Arquitectura de PSPNet [21]

4.1.1 Backbone

La arquitectura de la PSPNet está compuesta por un visor inicial o extractor de características, que en este caso es una ResNet101 dilatada pre-entrenada con ImageNet. Este visor tiene como tarea conseguir una combinación de filtros de características iniciales. Utilizar la ResNet como extractor de características es interesante en un modelo de segmentación por las conexiones residuales comentadas anteriormente. Este modelo va reduciendo la dimensionalidad de los mapas mediante MaxPooling, que va quedándose con las características más interesantes de cada mapa, hasta alcanzar un tamaño de mapa relativamente bajo. Para este problema, se incorporan ratios de dilatación a las convoluciones del extractor de características, con la idea de acumular más información contextual.

Como la densidad de la ResNet101 es muy alta, para no caer en el problema del desvanecimiento de gradiente, se propone una rama alternativa de pérdida supervisada como en la Inception de Google. Esto implica conectar la salida de uno de los bloques de la ResNet, concretamente después del bloque 4, con una pequeña red que interprete esos mapas de características y calcule una salida para computar el error con respecto al ground truth. La existencia de esta rama auxiliar permite que se propague una parte del error desde una situación más alta en la ResNet, permitiendo que el gradiente de error llegue hasta las capas más altas. Este gradiente de error se pondera con respecto al gradiente inicial, calculándose así el 40% del error intermedio supervisado [*Figure* 30].

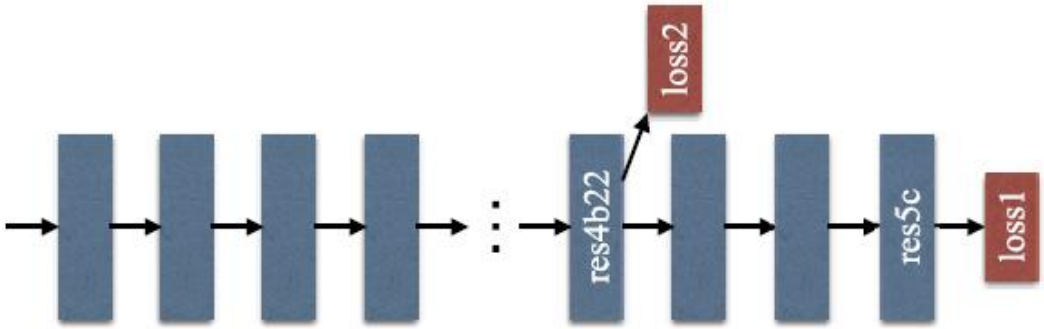


Figure 30: Perdida supervisada de PSPNet [21]

Para conectar con el módulo piramidal, se propone incrementar el tamaño de los mapas finales de la ResNet a 1/8 del tamaño inicial de la entrada para una mejor

interpretación de la información contextual. Esto se realiza con capas de UpSampling, que operan una interpolación bilineal hasta el tamaño deseado.

4.1.2 Pyramid Pooling Module

El módulo piramidal fusiona características de diferentes escalas. Inicialmente se opera un AveragePooling para reducir la dimensionalidad a los tamaños de pool deseados (los propuestos por el articulo original son 1,2,3 y 6). De esta manera, se obtienen mapas de características con convoluciones de 1x1, 2x2, 3x3 y 6x6, cubriendo así un amplio abanico de informaciones muy locales, otras un poco más contextuales, y otras más globales. Estos mapas de características se reescalan mediante capas de upsampling al tamaño de entrada del módulo, es decir 1/8, y se concatenan con los mapas de características del extractor inicial. La última convolución procesará características extraídas por la ResNet101 y el módulo piramidal concatenadas en un mismo bloque que contiene lo necesario para comprender la escena en su mayoría. Como se quiere obtener la probabilidad de pertenencia de un pixel a una clase, se coloca una capa convolucional con activación softmax que operará la predicción, y posteriormente se realizará una interpolación bilineal de vuelta al tamaño inicial de entrada [Figure 31].

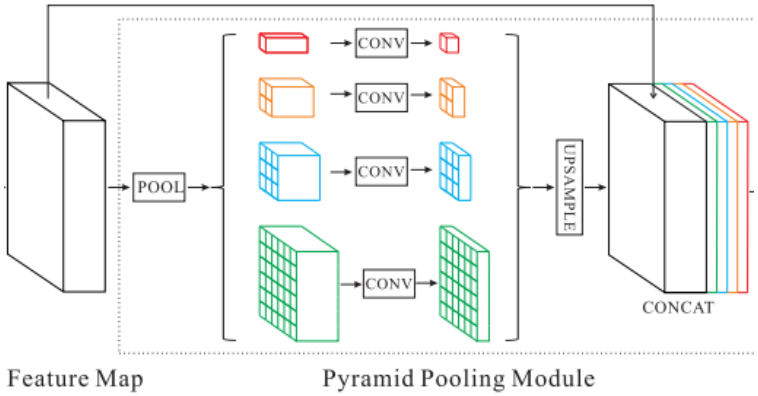


Figure 31: Pyramid Pooling Module [21]

4.1.3 Función de pérdida

La función de pérdida propuesta para este problema es la entropía cruzada categórica. Es una medida de precisión para variables categóricas. En la entropía cruzada categórica es más difícil la diferenciación y la convergencia, presenta escala univariante, es simétrica y fácil de interpretar por el descenso por gradiente.

$$\mathcal{L}(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij}\log(p_{ij})$$

4.1.4 Optimizador

El optimizador propuesto es el Stocastic Gradient Descent (SGD), o descenso estocástico del gradiente.

El descenso estocástico o gradiente de descenso incremental, es una aproximación estocástica del gradiente descendiente usado para minimizar una función objetivo que se escribe como una suma de funciones diferenciables. Este optimizador trata de encontrar mínimos o máximos por iteración.

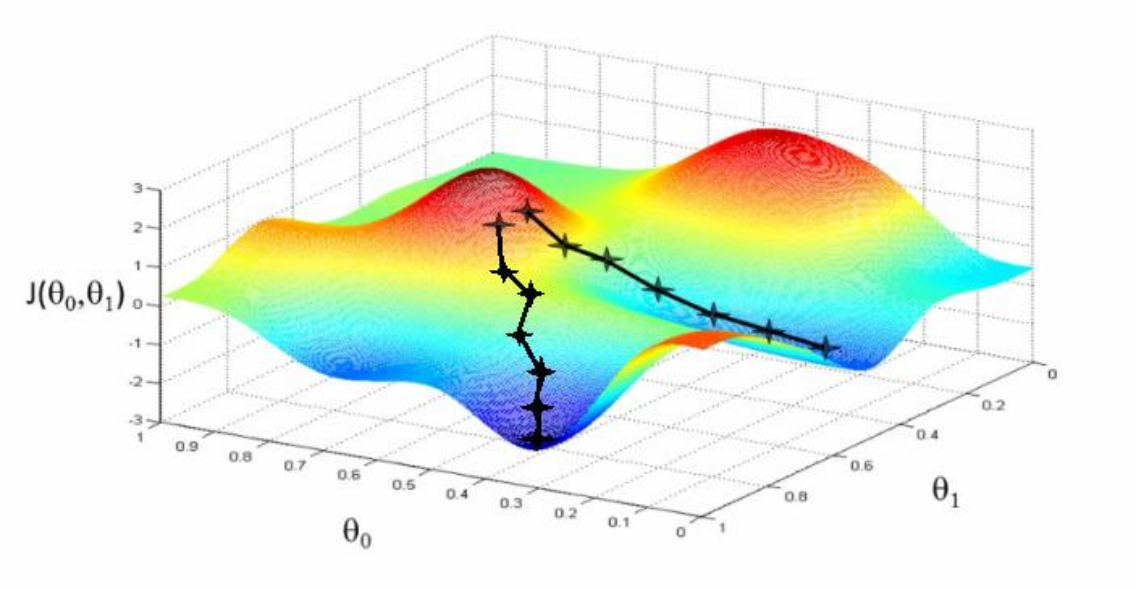


Figure 32: Representación del descenso por gradiente

Al igual que en la función del gradiente descendente, el gradiente indica la dirección en la que la función tiene la ratio de aumento más pronunciada, aunque no indica hasta dónde se debe avanzar en esa dirección [*Figure* 32].

4.2 Modelo de localización de keypoints propuesto

El modelo de localización de puntos clave propuesto es el conocido como Simple Pose.

4.2.1 SimplePose

Como ya se ha mencionado anteriormente, SimplePose [27] ofrece una aproximación sencilla al problema de la estimación de puntos clave corporales y pose humana. Combinando una ResNet como encoder y un conjunto de convoluciones traspuestas como decoder [Figure 33], es capaz de extraer características interesantes para la pose humana, y realizar una predicción muy acertada sobre los puntos clave corporales.

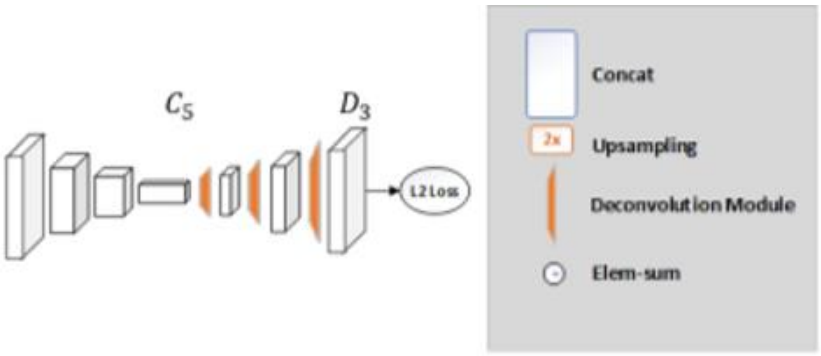


Figure 33: Estructura encoding-decoding de SimplePose [27]

Para probar el modelo, se utilizará una versión preentrenada de Gluon sobre la que se entrena nuestro dataset. La idea es que como este modelo está ya entrenado para localizar puntos clave y aprender su forma, podrá aprender a detectar las esquinas del folio y aprender la estructura del papel para regresión de puntos ocluidos.

La versión de Gluon sigue una estructura channel-first, por lo que se tiene que adaptar nuestros tensores de entrada y salida a este tipo de codificación.

- **Channel-first** propone una estructura de datos de la forma (canales, largo, ancho), esto significa que la forma en la que tendrán que entrar los datos al modelo es mediante 3 vectores de valores de pixel del tamaño de la imagen, correspondientes a los valores r, g y b.
- **Channel-last** propone una estructura de datos diferente, de la forma (largo, ancho, canales), lo que supone una matriz de entrada del tamaño de las imágenes con 3 valores diferentes para cada posición de la matriz, correspondientes a los valores de cada canal r, g y b.

Además, SimplePose propone utilizar un tamaño de entrada de (256, 192) que ha sido evaluado como hiperparámetro. Al ser la forma corporal más larga que ancha generalmente, el modelo se apoya en esta evidencia para proporcionar más información y facilitar el trabajo de identificación de pose en su problema original de localización de puntos clave corporales. El problema propuesto deberá acomodar las imágenes de entrada a dicho tamaño.

4.2.2 Función de pérdida

La función de pérdida propuesta para este problema es la L2 o "MSE". El Mean Squared Error mide la diferencia entre el valor estimado y el valor que se intenta estimar, en promedio. Es una medida intuitiva para calcular la precisión del estimador.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

En este caso, lo que se intenta estimar es la gaussiana 2D correspondiente al mapa de calor de cada keypoint. Esto implicará valores cercanos a 0 en la mayoría del mapa de calor y pocos valores cercanos a 1, indicando la proximidad de un punto clave. L2 penalizará la diferencia de los valores del mapa de calor a nivel de cada pixel para intentar simular el mapa de activación de cada keypoint [*Figure* 34].

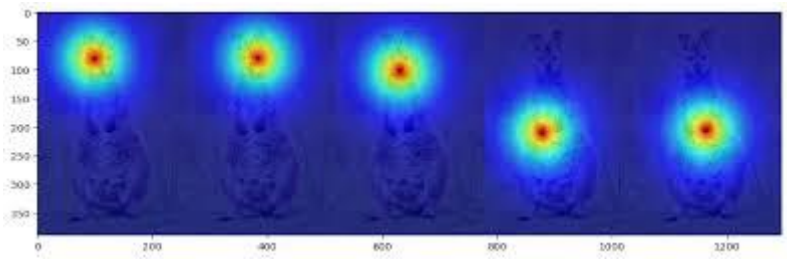


Figure 34: Heatmap en puntos clave

Por ello, no se puede utilizar una función de activación softmax como en el caso anterior, ya que no se está aprendiendo la probabilidad de un pixel de pertenecer a una clase, sino que se está aprendiendo el valor de activación de cada pixel. Lo más lógico sería utilizar funciones de activación lineales entre 0 y 1.

4.2.3 Optimizador

El optimizador propuesto para el modelo SimplePose es el Adam o "Adaptative momentum estimation". El algoritmo de Adam es diferente al de descenso por gradiente tradicional. Adam no solo calcula la tasa de aprendizaje de parámetros adaptativos en función del valor medio del primer momento como RMSProp, sino que también hace uso completo del valor medio del segundo momento del gradiente (la varianza no centrada).

5. Desarrollo de modelos finales

Partiendo de los modelos propuestos, se realizará una discusión y se propondrá una serie de modificaciones justificadas con un estudio con la finalidad de adaptar las diferentes redes a los problemas a resolver. De nuevo, se dividirá esta sección en el caso de segmentación y el caso de predicción de los puntos clave.

El estudio ha sido realizado en Google Colab, una plataforma de investigación de modelos de redes neuronales. Esta plataforma ofrece máquinas virtuales con GPUs potentes para este tipo de trabajos. Para el estudio se ha tenido que contar con las limitaciones de los recursos ofrecidos por este entorno.

5.1 Estudio del modelo de segmentación

Para la realización del estudio sobre los modelos de segmentación se han utilizado las siguientes métricas.

5.1.1 Métricas

Pixel accuracy

La precisión a nivel de pixel nos permite conocer el número de píxeles mal clasificados y el número de píxeles bien clasificados. Es una medida estándar para medir la precisión de un modelo, pero en los casos de segmentación puede no ser la métrica más interesante para conocer la validez de un modelo.

Se supone el caso en el que se pretende realizar una tarea de segmentación binaria donde existe una clase background y una clase de interés. La clase de interés ocupa un 1% de la imagen y la clase background ocupa el 99% restante. Si un modelo de segmentación realizara una predicción completa de background, la precisión a nivel de pixel sería del 99%, ya que ha cometido un 1% de error en su predicción final correspondiente a la clase de interés. Esto nos puede hacer pensar que nuestro modelo es válido para el problema, pero nada más lejos de la realidad. Para evaluar correctamente el modelo habría que utilizar una métrica que tuviera en cuenta la cantidad de clases para operar una ponderación, y por ello se utilizará el conocido como “dice coeficient”.

Dice coef

El Dice coefficient o F1 Score computa el área de solape entre la predicción y el ground truth multiplicado por 2 dividido entre el número total de pixeles en ambas segmentaciones [*Figure* 35].

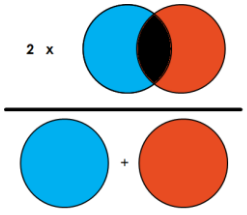


Figure 35: Representación del dice coef

En el caso de segmentación multiclase, el F1 medio se calcula obteniendo el F1 de cada clase y dividiéndolo entre el número de clases.

5.1.2 Evolución del modelo de PSPNet

Partiendo del modelo original de Caffe convertido a modelo congelado de TensorFlow, se realizarán pruebas sobre el dataset después del entrenamiento para comprobar su funcionamiento.

El modelo inicial ha sido entrenado de la siguiente manera:

Train	4870 imágenes normalizadas por la media
Validación	1214 imágenes normalizadas por la media
Función de pérdida	Entropía cruzada categórica
Optimizador	SGD

Table 4: Configuración de entrenamiento

Y testeado sobre muestras de validación presenta los siguientes resultados:

Mean Pixel Accuracy	0.9576%
Dice coef	0.70%

Table 5: Resultados PSPNet base

El porcentaje de acierto sobre las diferentes clases es el siguiente [*Figure* 36]:

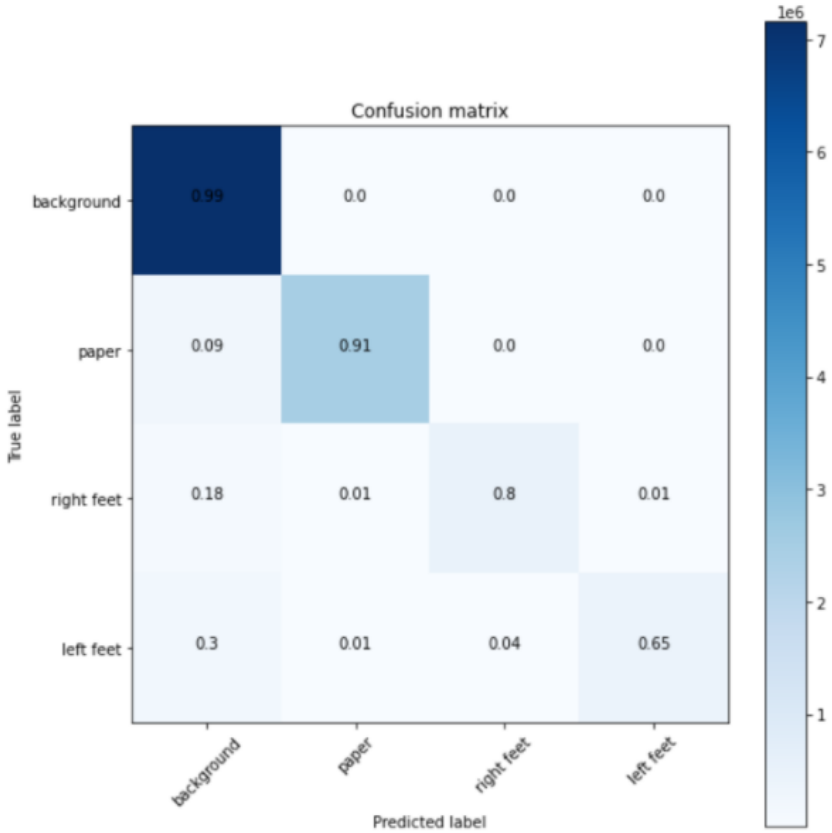


Figure 36: Matriz de confusión del modelo base de PSPNet

La predicción sobre la imagen resulta de la siguiente manera [Figure 37] :

Figure 37: A la izquierda la entrada, posteriormente el ground truth y al final la salida

Como se puede observar, se produce una pérdida de definición en la segmentación de la imagen.

Se propone reducir las capas de upsampling y por lo tanto reducir de manera menos drástica la dimensionalidad del extractor de características. Esto implica eliminar el último bloque de la ResNet para alcanzar un tamaño de mapa de características de 30,30 en vez de 15,15.

Al tener menos profundidad en el módulo extractor de características, es interesante cambiar el lugar de la rama auxiliar de pérdida para que siga estando a mitad del modelo, por lo que se mueve a mitad del bloque 3.

Tamaño de pools

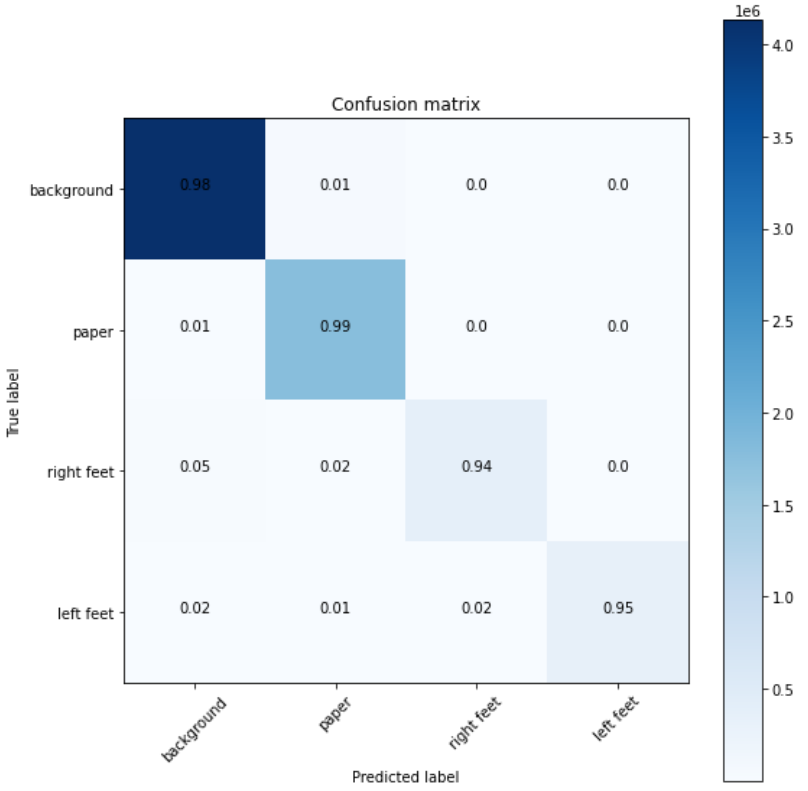
Se prueban diferentes combinaciones de tamaños de pool para ver cual se adapta mejor al problema. Además, se obtiene la estrategia de learning rate propuesta por el artículo de la PSPNet, la poly learning rate policy, que va reduciendo el learning rate a medida que van pasando las epochs y teniendo en cuenta el número total de epochs de entrenamiento:

$$lr = lr_0 \times (1 - \frac{i}{T_i})^{power}$$

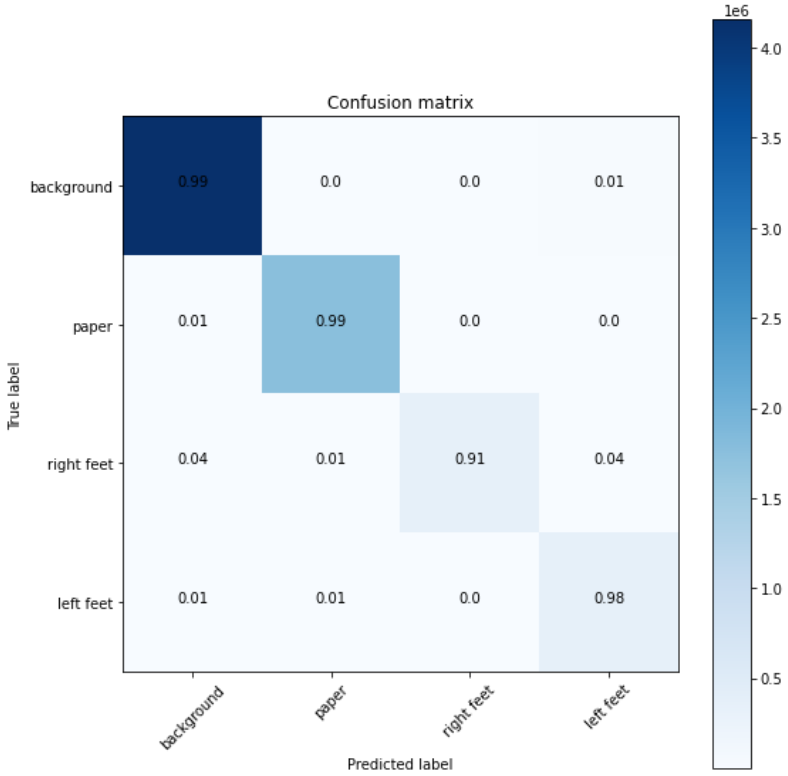
Los datos de entrenamiento y sus correspondientes resultados son los siguientes:

Learning rate	Tamaño de batch	Epoch	Optimizador	Métrica
Poly LR Policy con base 1e-3	8 imágenes	10	RMSProp	Pixel Acc

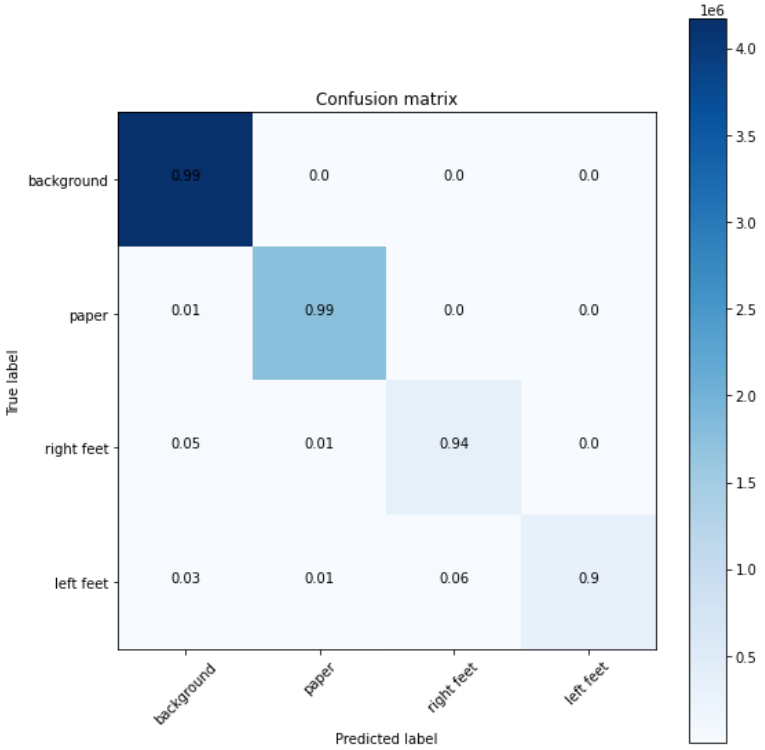
Modelo	Entrenamiento	Validación
ResNet50PSPNet1236	0.9818	0.9714



Modelo	Entrenamiento	Validación
ResNet50PSPNet1238	0.9816	0.9712



Modelo	Entrenamiento	Validación
ResNet50PSPNet1356	0.9813	0.9701



Se adjunta una comparación de las segmentaciones de los diferentes modelos:

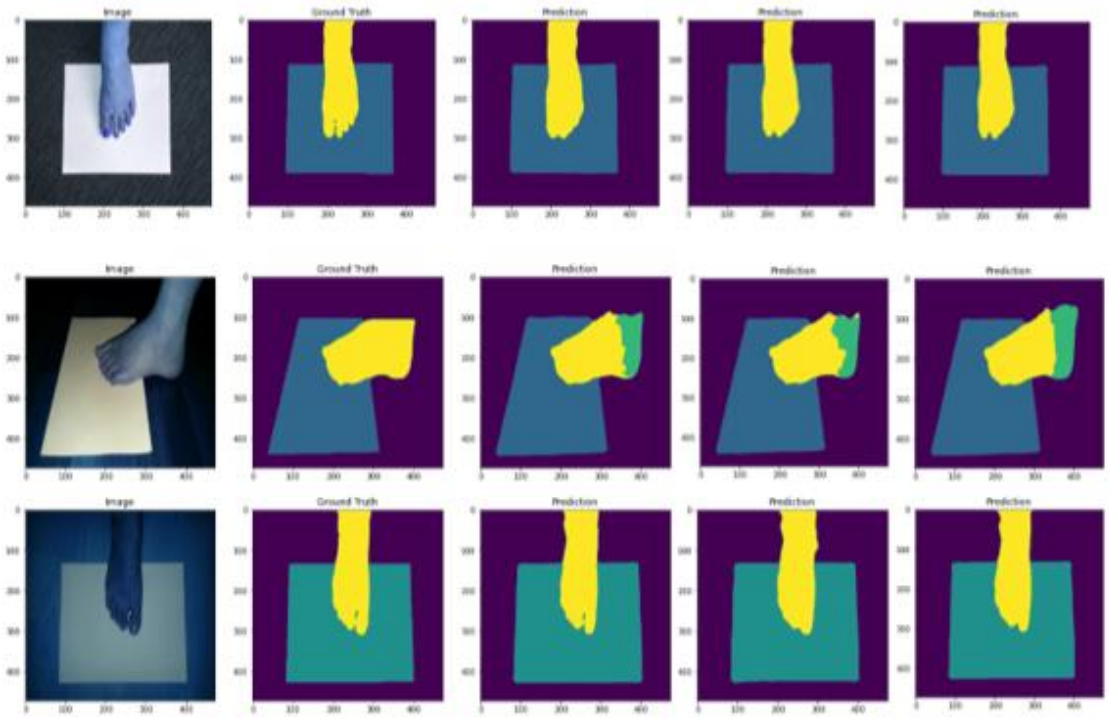


Figure: Primera columna como entrada, segunda como ground truth, tercera prediccion 1236, cuarta prediccion 1238, quinta predicción 1356

Analizando los datos se puede llegar a los siguientes resultados:

- La reducción de la profundidad de la ResNet50 no parece afectar mucho ni a los resultados finales de segmentación ni a las métricas.
- La mayoría de los errores son provocados desde la clase background, posiblemente debido al desbalanceo de datos de entrenamiento. Como la clase background es la que más pixeles presenta en el dataset, es posible que la red esté sobre-aprendiendo esta información.
- El tamaño de los pools no parece afectar demasiado al problema. Las diferentes combinaciones tienen sus ventajas y sus inconvenientes, pero parece que la mejor combinación es la propuesta en el artículo original, es decir, 1236.

Transfer Learning con la ResNet

Comparando el dataset correspondiente al problema con el dataset de ImageNet, se puede observar que no guardan relación entre ellos. Aun así, como la ResNet funciona como buen extractor de características e independientemente del dataset es capaz de encontrar buenas combinaciones de filtros. Se congela los dos primeros bloques con la idea de conservar los patrones de imagen más sencillos como bordes y esquinas, y se entrenará los siguientes que se encargan de patrones más abstractos y que se necesitan entrenar para nuestro problema. A demás, se prueba a normalizar los datos entre 0 y 1.

Con estos cambios se obtiene una mejora de en torno a 0.01 puntos en precisión a nivel de pixel tanto en entrenamiento como en validación [*Figure* 38]:

Modelo	Entrenamiento	Validación
ResNet50FrozenPSPNet1356	0.9901	0.9801

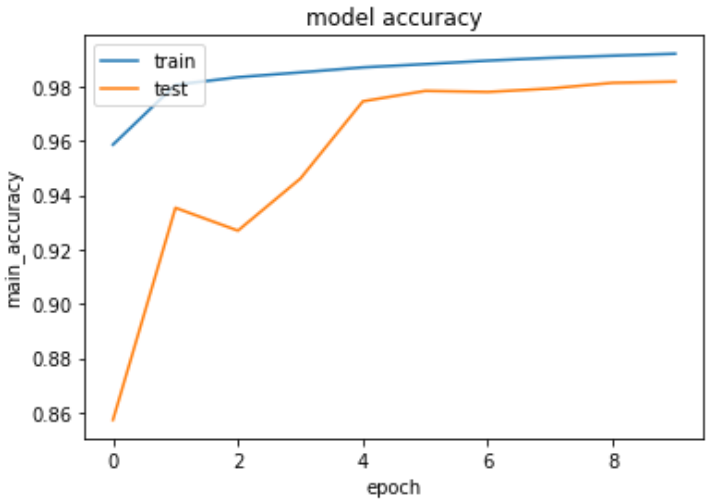


Figure 38: Precisión modelo

EfficientNet como extractor de características

Con la llegada de nuevos modelos de visión por computador, cambiar el extractor de características por uno más novedoso puede ser algo interesante. Diversos estudios sitúan la nueva EfficientNet de Google como uno de los visores más interesantes y con mejores resultados tanto en precisión como en velocidad [29] [*Figure* 39].

La EfficientNet presenta cambios interesantes con respecto a la ResNet. Sus convoluciones son del tipo MobileConv o SeparableConv, por lo que, para la misma cantidad de filtros, operará más rápido.

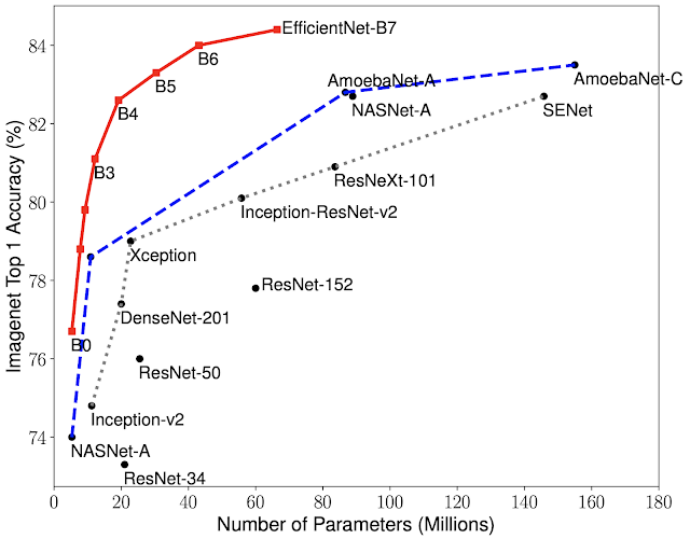


Figure 39: Gráfico de precision de arquitecturas [29]

Cambiando el extractor de características y siguiendo la misma política de congelación de bloques que en la ResNet, es decir, congelar los dos primeros bloques, colocar la rama de supervisión intermedia a mitad del visor (bloque 4) y obtener la salida del último bloque con 30 pixeles de tamaño de filtro, se obtiene un rendimiento similar con un visor con 10M de parámetros menos, la EfficientNetB3 [*Figure* 40]:

Modelo	Entrenamiento	Validación
Efficient3PSPNet1236	0.9913	0.9803

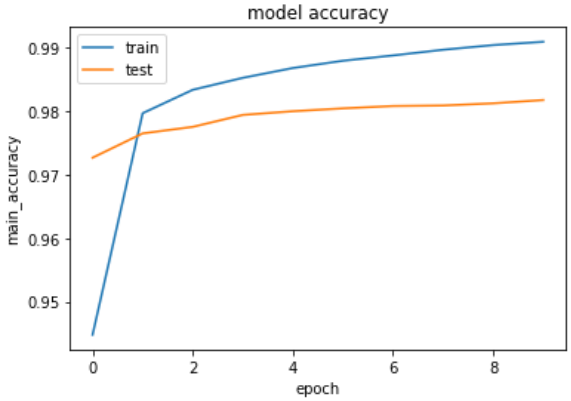


Figure 40: Precisión de modelo

Probando el modelo más ligero y más rápido, se obtiene una precisión parecida con un modelo total de 15M de parámetros [*Figure* 41].

Modelo	Train	Validación
Efficient0PSPNet1236	0.9885	0.9800

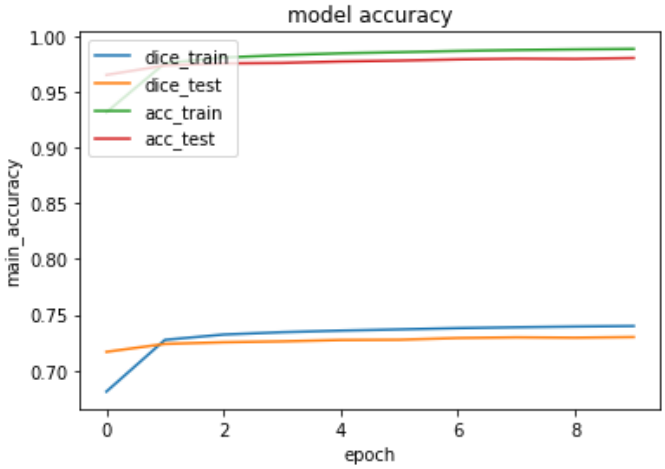


Figure 41: Precisión de modelo

El modelo más denso preentrenado con ImageNet, la EfficientNetB5, nos presenta los siguientes resultados con un modelo de 23M de parámetros:

Modelo	Train	Validación
Efficient5PSPNet1236	0.9903	0.9820

Data Augmentation sobre el dataset

Aumentar el dataset es una buena forma de mejorar los resultados de validación si se realiza correctamente, ya que prepara al modelo frente a unas imágenes que se parecerán a las que se verán en la realidad. El proceso de Data Augmentation se realizará combinando el ImageDataGenerator de Keras con funciones implementadas en flujo de ejecución:

- Rotación de las imágenes 5º
- Zoom out sobre las imágenes en un rango de un 10%
- Se realizará flip horizontal con una probabilidad del 50%. Las imágenes de segmentación cambiarán la etiqueta de identificación del pie, ya que el flip cambia su clase de izquierdo a derecho o viceversa.

En el caso real del problema, previa a la segmentación existe un modelo encargado de validar las imágenes para que no se procesen imágenes con errores. Puesto que es crucial que se vean tanto el folio al completo como el pie, no se realiza zoom de acercamiento. Se obtienen los siguientes resultados:

Modelo	Entren. Pixel Acc	Entren. F1	Validación	Val F1
Efficient4PSPNet1236	0.9927	0.7456	0.9920	0.8082
Efficient5PSPNet1236	0.9888	0.7419	0.9886	0.7395

A continuación se muestran algunas imágenes del resultado de la segmentación del modelo Efficient4PSPNet1236 [*Figure* 42].

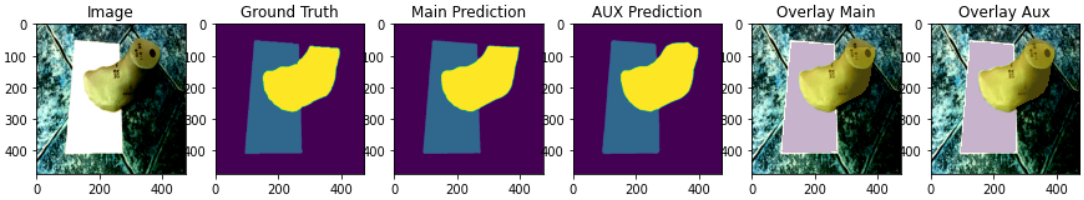


Figure 42: Resultado de segmentación

Como se puede observar, el modelo presenta complicaciones a la hora de apurar los bordes del folio, es decir, la frontera entre la clase 0 y la clase 1. Para intentar corregir este problema, se propone modificar la función de pérdida.

Función de pérdida

El problema de la diferencia entre la cantidad de muestras de cada clase es estado del arte en los problemas de segmentación. Lo más sencillo es realizar una ponderación del peso que tiene cada clase sobre la función de pérdida, para forzar a la red a que penalice más los errores de clases con menos muestras, y menos los errores de las clases con más muestras.

$$-\sum_{c=1}^{M} \propto y_{o,c} \log(p_{o,c})$$

Entropía cruzada ponderada

Para ello se utilizará la entropía cruzada ponderada, ponderando de la siguiente manera:

Clase	0	1	2	3
Ponderación	0.1	1.0	2.0	2.0

Modelo	Train Pixel Acc	Train F1	Validación	Val F1
Efficient4PSPNet1236	0.9794	0.7107	0.9803	0.7334

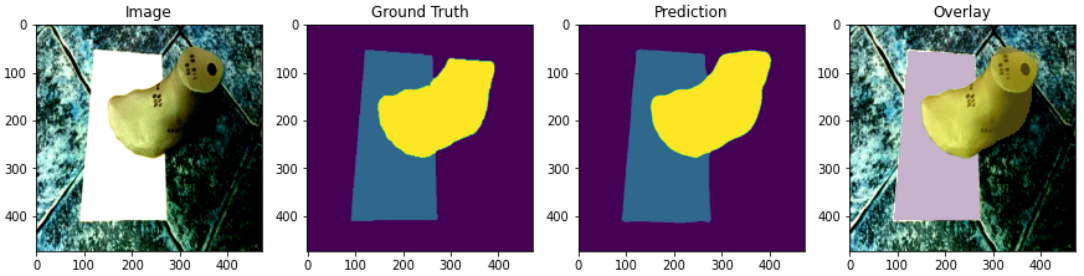


Figure 43: Resultado de segmentación

No mejora los resultados, pero parece apurar mejor las fronteras entre clases [*Figure* 43].

Focal Loss

Se analizará también la Focal Loss. Es una función de pérdida propuesta por Lin et. al de Facebook, que realiza una ponderación entre clases, pero además calcula la pérdida de diferente manera en función de si son muestras sencillas o complicadas. Esto lo hace en función de la pérdida calculada por dicha muestra.

$$FL = -\sum_{i=1}^{C=2} (1 - s_i)^{\gamma} t_i log(s_i)$$

Clase	0	1	2	3
Ponderación	0.15	0.45	0.9	0.9

Modelo	Train Pixel Acc	Train F1	Validación	Val F1
Efficient4PSPNet1236	0.9887	0.7405	0.9879	0.7385

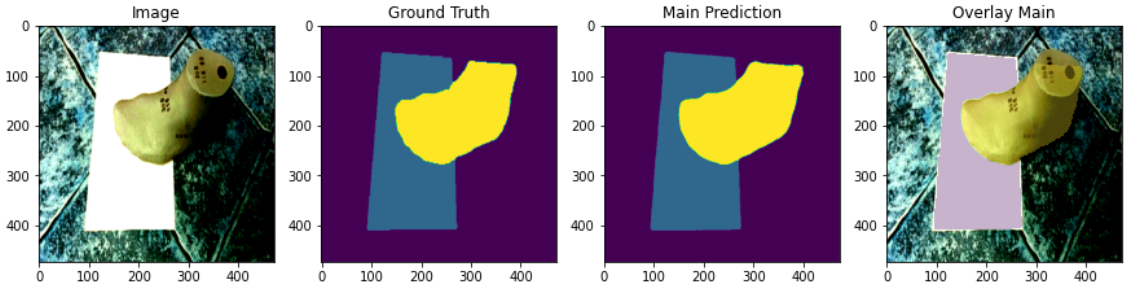


Figure 44: Resultado de segmentación

Como se puede observar, presenta unos resultados similares a la entropía cruzada ponderada [*Figure* 44].

A continuación, se muestra un mapa de error en la comparativa entre la entropía cruzada y la focal loss [*Figure* 45].

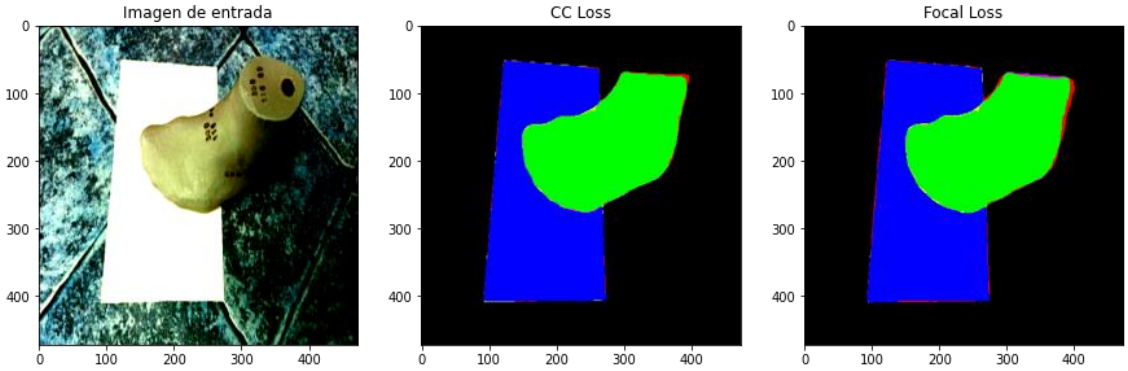


Figure 45: Mapa de error de etiqueta

Se puede comprobar que la entropía cruzada presenta una segmentación más limpia en comparación con la Focal Loss, que en algún caso concreto mejora los resultados, pero no en el caso general.

Tiempo de inferencia

A continuación, se estudiará el tiempo de inferencia del modelo modificado, comparado con el modelo inicial de segmentación. La inferencia se ha realizado en una Nvidia Tesla T4 de 16Gb de RAM ofrecida por Google Colab. Se prueba el modelo final EfficientB4PSPNet sobre el CaffeResNet101PSPNet [*Figure* 46].

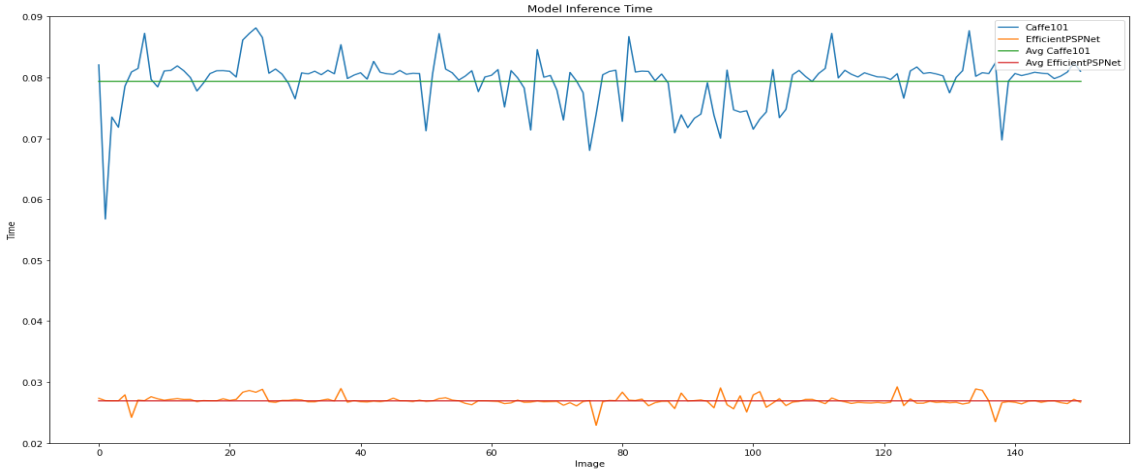


Figure 46: Comparativa de tiempos de inferencia entre el modelo base y modelo final (tiempo/imagen)

La EfficientPSPNet es casi 3 veces más rápida que la PSPNet original, permitiendo realizar 33 inferencias por segundo, lo que permitiría realizar una segmentación fluida en tiempo real. En cambio, el modelo original infiere 12 imágenes por segundo, muy lejos de la segmentación en tiempo real. Ambos modelos se han congelado en un FrozenGraph de TensorFlow para optimizar la inferencia.

5.2 Estudio del modelo de localización de puntos clave

Para el estudio del modelo de localización de puntos clave se han utilizado las siguientes métricas.

5.2.1 Métricas

MSE

En estadística, el **error cuadrático medio** de un estimador mide el promedio de los errores al cuadrado, es decir, la diferencia entre el estimador y lo que se estima. Indica cuánto de lejos está el valor estimado sobre el valor real.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{Y}_i - Y_i\right)^2$$

MAE

El error absoluto medio es una medida de la diferencia entre dos variables continuas. Considerando dos series de datos (unos calculados y otros observados) relativos al mismo fenómeno, el error absoluto medio sirve para cuantificar la precisión del modelo de una forma general.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|\hat{Y}_i - Y_i\right|$$

Heatmap Accuracy

Es una métrica propia de Gluon MX, el entorno sobre el que se trabajará con el modelo propuesto de Simple Pose [30]. Indica la precisión en un rango de 0-1 del modelo sobre los heatmaps.

5.2.2 Evolución del modelo de regresión de puntos

El modelo propuesto ha sido entrenado de la siguiente manera:

Train	4371 imágenes segmentadas
Validación	1873 imágenes segmentadas
Función de pérdida	MSE (L2)
Optimizador	Adam

Y testeado sobre las muestras de validación presenta los siguientes resultados:

Heatmap Accuracy	0.998%

Dado que el modelo de SimplePose propuesto más ligero supera los 34M de parámetros, se buscará una opción más simple para resolver el problema. En este caso, se prioriza tiempo de inferencia frente a precisión, ya que el resultado de la regresión se enviará a unos cálculos que afinan el resultado, por lo que nos interesa que este proceso sea eficiente y sobre todo que pueda adivinar donde se encuentran los puntos ocluidos.

Inicialmente se propone seguir la estructura Hourglass propuesta por el modelo Pose AE. El modelo funciona de la siguiente manera: las convoluciones y el max pooling son usadas para procesar las características a un nivel de resolución muy bajo. En cada max pooling, la red se ramifica y se aplican más convoluciones. Al llegar al nivel de resolución más bajo (15x15 en nuestro caso), la red encadena una serie de upsamplings y combinaciones de mapas de características a lo largo de los diferentes niveles. Como es una topología simétrica, a cada bloque le corresponde otro en la secuencia top-down. Al final de cada bloque Hourglass, hay una convolución 1x1 que realiza la predicción final del mapa a nivel de pixel. Todos los bloques convolucionales presentan funciones de activación de tipo ReLu excepto las últimas. Como se quiere hacer una regresión del valor de pixel, se colocará una función de activación lineal que tomará valores entre 0 y 255 para el centro de la gaussiana.

Se comenzará probando un modelo con bloques convolucionales bottleneck que alcanzan hasta 128 filtros y dos bloques Hourglass con supervisión intermedia.

Learning rate	Tamaño de batch	Epoch	Optimizador	Métrica
Reduce On Plateau	8 imagenes	50	Adam	MSE,MAE

Se alcanza un error muy bajo de 1.3e-5 en validación en la última salida, resultando en unos valores muy interesantes [*Figure* 47].

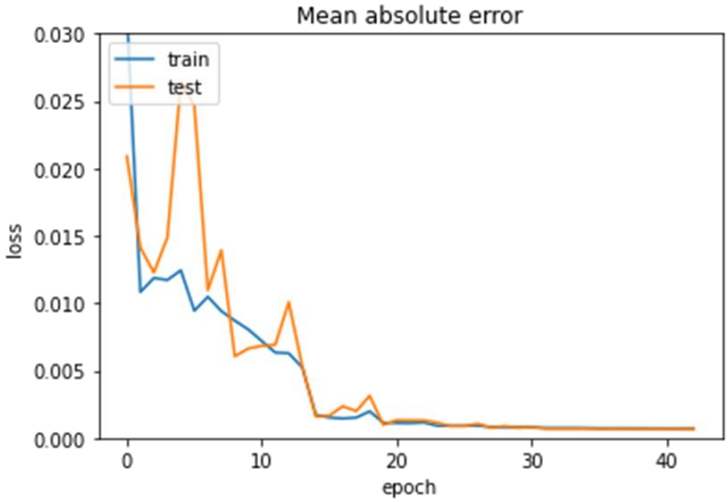


Figure 47: Pérdida del modelo

Se puede observar la precisión del modelo [*Figure* 48]:

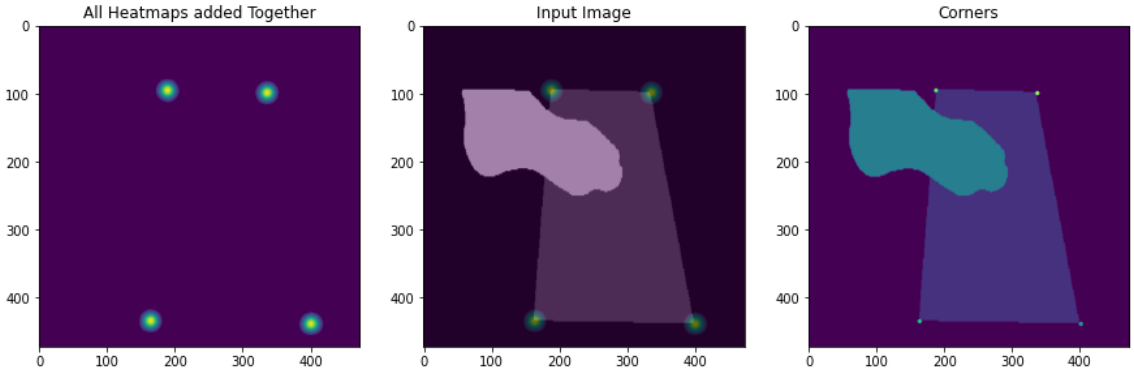


Figure 48: Predicción de salida

Se encuentra un caso donde no funciona bien el modelo, el caso de los puntos ocluidos. Como se puede observar en la siguiente imagen, aparentemente el modelo interpreta la unión del pie y del folio como una esquina al estar esta oculta [*Figure* 49]:

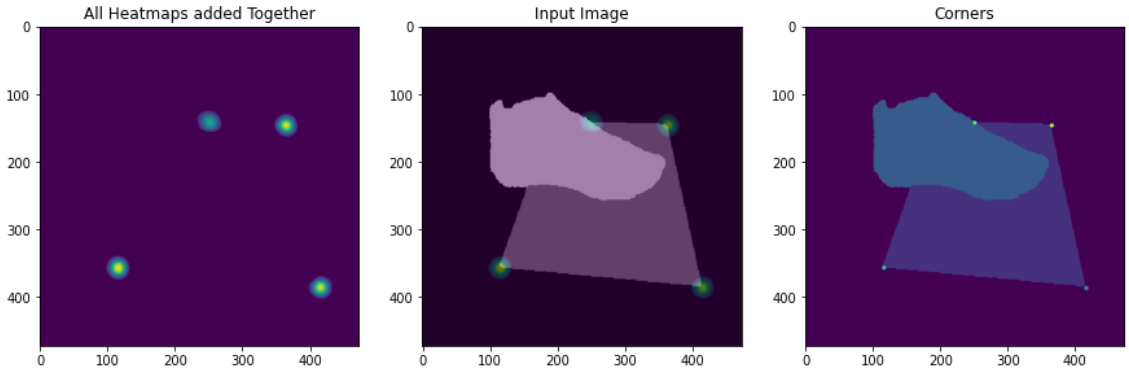


Figure 49: Predicción errónea sobre punto ocluido

Para solventar este problema, se deben dividir los puntos en dos conjuntos, puntos visibles y puntos ocluidos. Con estos dos conjuntos diferenciados, aplicando una penalización mayor al error de los puntos ocluidos se puede forzar al modelo a que encuentre alguna forma de localizar dichos puntos, aprendiendo la forma del folio de alguna forma. Puesto que no se tienen estos dos conjuntos diferenciados, se proponen dos modificaciones en la función de pérdida:

Adaptación de la Focal Loss

Se propone aplicar un factor de decrecimiento del error a los puntos cuyo error de salida esté por debajo de un umbral concreto, y otro distinto a aquellos que superan el umbral. De esta forma se marca una línea entre los puntos fáciles de aprender y los puntos difíciles. Presenta unos resultados parecidos, pero no consigue solucionar el problema de los puntos ocultos [*Figure* 50].

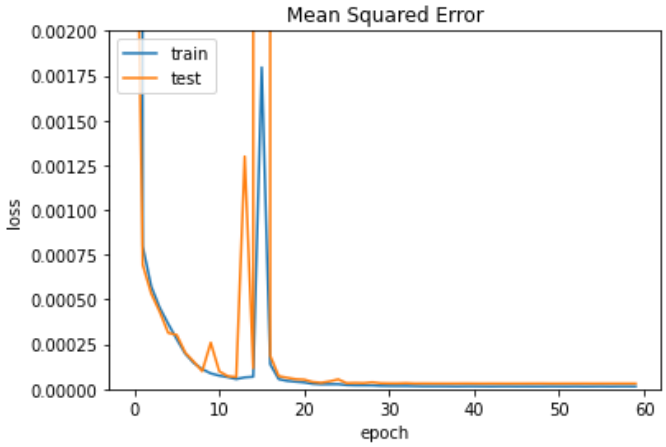


Figure 50: Pérdida con Focal Loss

Con la idea de extraer más información y realizar una comprobación entre la localización de puntos y la segmentación, se realiza dicha regresión sobre la imagen de entrada RGB, siendo esto una propuesta del IBV. Se encuentra una precisión parecida, aunque el mapa de segmentación presenta tan solo 4 valores de pixel diferentes por lo que debería ser más sencillo de interpretar para una red convolucional [Figure 51].

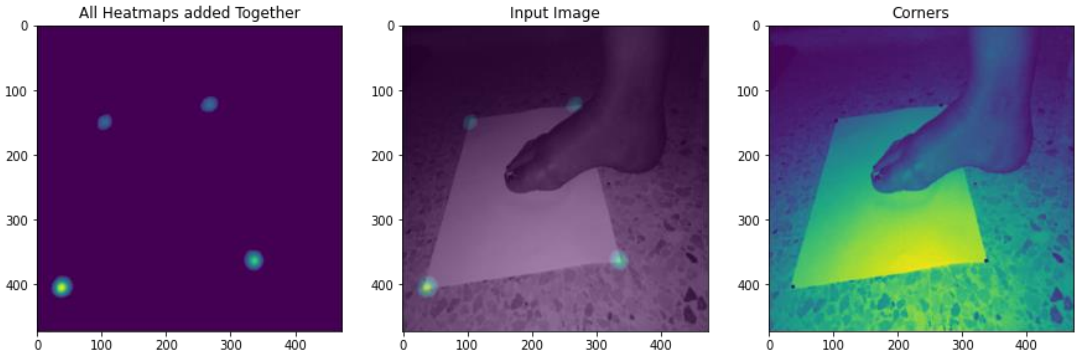


Figure 51: Predicción del modelo

Online Hard Keypoint Mining (OHKM)

Es una técnica que consiste en backpropagar solo el error de los k puntos con mayor error, de esta forma el modelo se focaliza más en los casos más complicados. Normalmente está preparada para problemas donde existe una mayor cantidad de puntos, pero en nuestro caso al buscar tan solo 4, se computará la pérdida de los 3 puntos más conflictivos (con mayor error en la salida). Al igual que la Focal Loss, presenta unos resultados parecidos [*Figure* 52]:

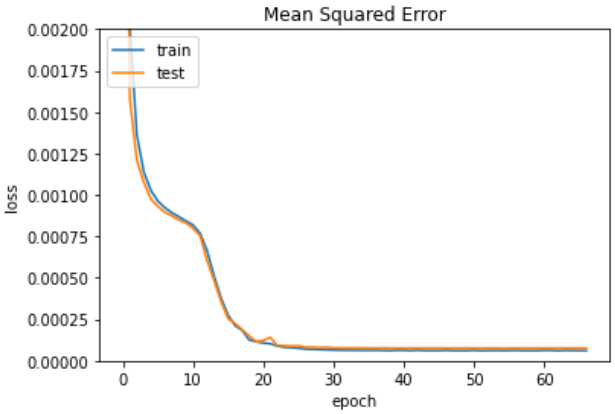


Figure 52: Pérdida de modelo con OHKM

Tampoco soluciona el problema de los puntos ocluidos [*Figure* 53]:

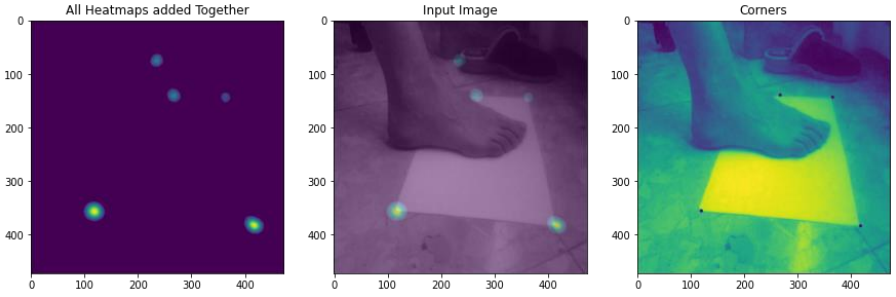


Figure 53: Error en predicción de punto ocluido

Modificación SimplePose

Finalmente, al no poder resolver estos casos concretos, se probará con el modelo inicialmente propuesto para comprobar que efectivamente es robusto frente a esta situación [*Figure* 54].

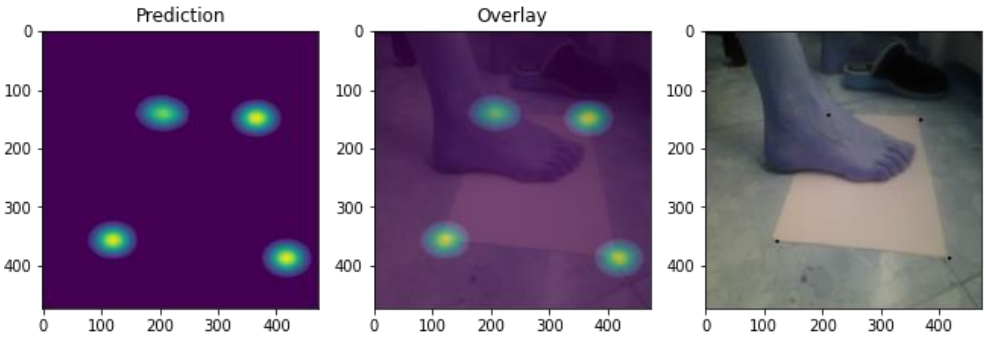


Figure 54: Predicción correcta de punto ocluido

Como se puede observar, la precisión de los resultados disminuye, por lo que se prueba a aumentar la profundidad del visor y se utiliza una ResNet50 en vez de la ResNet18 utilizada anteriormente. Los resultados son parecidos, por lo que no merece la pena la modificación [*Figure* 55]:

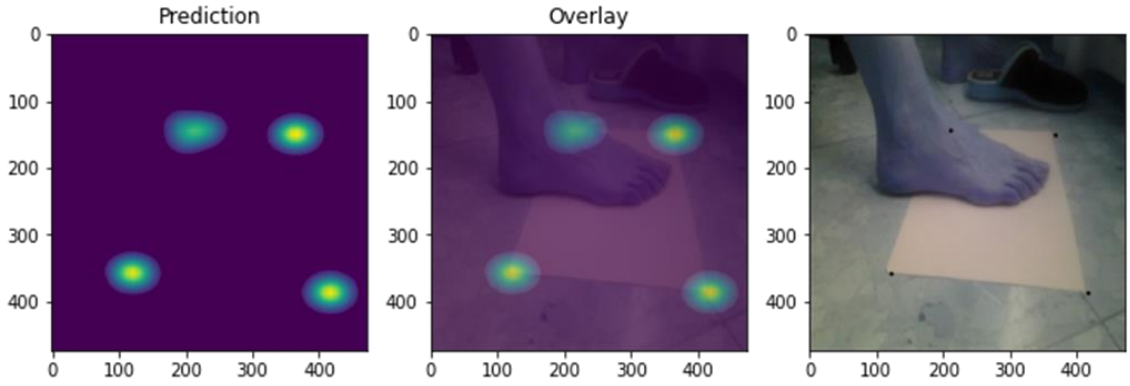


Figure 55: Predicción con ResNet50 de backbone

Se prueba a disminuir el tamaño de la gaussiana, mejorando la precisión significativamente [*Figure* 56]:

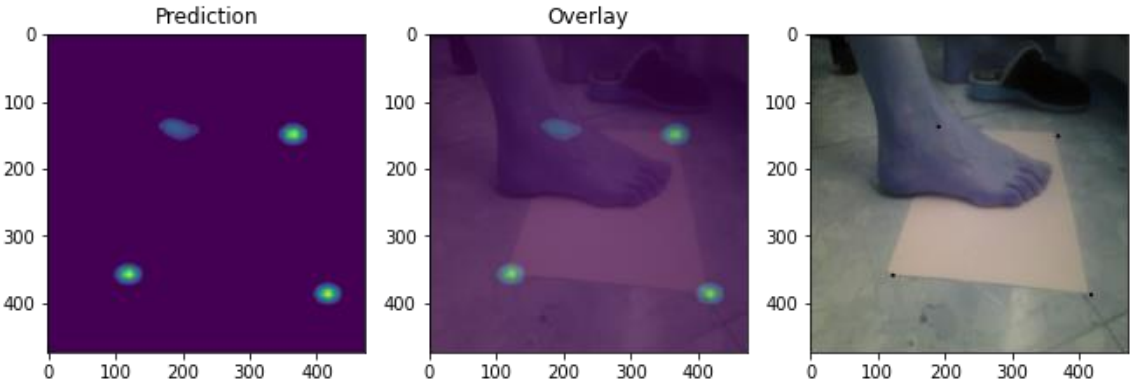


Figure 56: Predicción con disminucion de tamaño de gaussiana

Tiempo de inferencia

El entorno de Gluon MXNet donde se ha entrenado el modelo SimplePose propone exportar el modelo a ONNX [31], formato universal para las redes neuronales. Con la idea de mejorar el tiempo de inferencia sobre la infraestructura real del problema (servidor Intel sin GPU) se propone optimizar el modelo mediante la herramienta OpenVino [32]. Este sistema realiza una modificación en la arquitectura del modelo optimizándola para funcionar en procesadores Intel, mejorando así los tiempos de inferencia. Testeando los tiempos de inferencia obtenemos unos resultados de entorno a 0.15 segundos en GPU con el modelo ONNX y aproximadamente el mismo tiempo de inferencia con el modelo de OpenVino en CPU.

6. Conclusiones

Como conclusión de este proyecto, se realizará un repaso de todo lo realizado y obtenido punto por punto, además de algunos razonamientos posteriores al proyecto, remarcando el éxito sobre los objetivos propuestos.

Se ha aplicado inteligencia artificial a un proceso anteriormente desarrollado por técnicas clásicas, mejorando la robustez del sistema. Anteriormente el proceso de segmentación fallaba sobre imágenes que se salieran de lo establecido en el problema, pero ahora existe un módulo de segmentación capaz de entender la imagen y de proporcionar precisión a los resultados incluso frente a imágenes con sombras, rotadas, saturadas y poco contrastadas.

Se ha mejorado el modelo de PSPNet propuesto mediante un mejor tratamiento de los datos y una modificación de la arquitectura inicial, resultando en un incremento en la precisión y una disminución en el tiempo de inferencia, dando una experiencia más sencilla para el usuario.

Durante la realización del proyecto, se han trabajado técnicas de clasificación automática mediante clustering sobre problemas de segmentación típicos como CityScapes.

Se han abordado diferentes topologías, técnicas, optimizadores y funciones de pérdida resultando en un proyecto muy robusto.

Se ha aprendido a abordar problemas de regresión de puntos clave mediante la predicción de mapas de calor, algo muy aplicable a otras disciplinas.

Se ha realizado un estudio sobre la regresión de puntos clave que es extrapolable a muchas disciplinas. No se ha podido mejorar en gran parte el modelo propuesto debido a la falta de datos, pero se supone que la aproximación ha resultado buena.

Se ha experimentado con optimizadores de modelos neuronales como ONNX y OpenVino, que trasladan estos problemas a un caso real donde el tiempo de inferencia es vital.

En lo referente al problema de regresión de esquinas, como se comentó en el desarrollo del proyecto, seguramente con la existencia de datos que diferenciaran las esquinas ocluidas de las esquinas visibles se podría haber mejorado los casos más complicados. Algo que se probó pero que no resultó fue el desarrollo de una función de pérdida que tuviera en cuenta la diferencia de las pendientes que conformaban los segmentos enfrentados de cada folio, resultando en un trabajo matemático interesante. Algo que falta en el proyecto es la inclusión de los módulos trabajados en la secuencia de extracción de información para la resolución final del modelo 3D, pero por cuestiones de tiempo no se trataron estos temas, focalizando el estudio a los módulos presentados. A pesar de estas cuestiones, considero haber aprendido mucho con este proyecto y haber disfrutado en el proceso.

7. Trabajos futuros

En lo respectivo a los trabajos futuros, cabe destacar la conversión del formato del modelo de segmentación al formato ONNX para optimizar los tiempos de inferencia. El formato ONNX es novedoso y aún sigue en pruebas, funcionando bien con algunos modelos y mal con otros. Después de realizar numerosas pruebas, por alguna razón, las salidas del modelo ONNX aparecían desplazadas [*Figure* 57], pero probando el mismo optimizador de arquitecturas con el modelo de SimplePose, los resultados eran satisfactorios. En un futuro cuando este formato esté más desarrollado, sería interesante volver a probar el modelo de segmentación para ver si se sigue consiguiendo la misma precisión en un tiempo de inferencia más rápido.

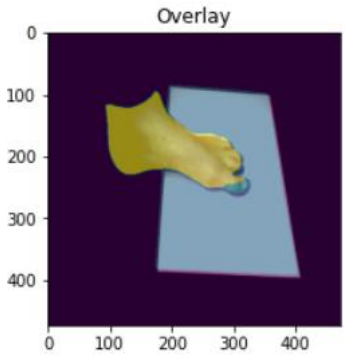


Figure 57: Desplazamiento en la segmentación

Sobre el modelo de regresión de esquinas, cabe destacar la efectividad con la que se ejecutaban las predicciones mediante el modelo tipo Hourglass, fallando a su vez en los casos de los puntos no ocluidos. Con un dataset que especifique los puntos no ocluidos, se podría tratar estos casos difíciles de una manera diferente para forzar a la máquina a ser efectiva frente a estas situaciones. Sería interesante reentrenar el modelo cuando existan los datos para probar su efectividad y comprobar si efectivamente es capaz de resolver este problema.

Con los módulos ya listos, en un futuro cuando se adapten los servidores y el código de la aplicación para estos sistemas, sería interesante probar la aplicación para ver cómo funciona y testear los tiempos de inferencia comparándolos con los anteriores para ver en que medida se ha mejorado el funcionamiento de la app.

8. Bibliografía

[1] IBV. [Online]. Available: https://antropometria.ibv.org/3d-avatar-feet/.

[2] IEEE, "IEEE," [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1640454?casa_token=LcvkMomOQ5UAAAAA:gbqrU-itXONiBt-XiqstAE2IXsGtfxy_NEqInEacXWOH-zImRktZ-2qsX-NETW4O1LleFR5wkLo.

[3] C. Chen, "A survey on deep learning for localizacion and mapping towards the age of spatial machine intelligence," [Online]. Available: https://www.researchgate.net/publication/342409316_A_Survey_on_Deep_Learning_for_Localization_and_Mapping_Towards_the_Age_of_Spatial_Machine_Intelligence.

[4] D. Gupta, "Deep Learning Semantic Segmentation Keras," 2019. [Online]. Available: https://divamgupta.com/image-segmentation/2019/06/06/deep-learning-semantic-segmentation-keras.html.

[5] "Acodigo," 2017. [Online]. Available: http://acodigo.blogspot.com/2019/08/segmentacion-de-instancias-con-opencv_16.html.

[6] Wikipedia, "Areas de brodmann," [Online]. Available: https://en.wikipedia.org/wiki/Brodmann_area.

[7] PsicoActiva, "Las Areas de Brodmann (Localizacion y funcion)," [Online]. Available: https://www.psicoactiva.com/blog/las-areas-brodmann-localizacion-funcion/.

[8] TowardsDataScience, "Convolutional Neural Networks, a beginners guide," [Online]. Available: https://towardsdatascience.com/convolution-neural-networks-a-beginners-guide-implementing-a-mnist-hand-written-digit-8aa60330d022.

[9] Kharshit, "Autoencoder downsampling and upsampling," 2019. [Online]. Available: https://kharshit.github.io/blog/2019/02/15/autoencoder-downsampling-and-upsampling.

[10] K. Sanjar, "Improved U-Net FCN Model for Skin Lesion Semantic Segmentation," [Online]. Available: https://www.researchgate.net/publication/341645510_Improved_U-Net_Fully_Convolutional_Network_Model_for_Skin-Lesion_Segmentation.

[11] Programmerclick. [Online]. Available: https://programmerclick.com/article/65511302670/.

[12] TowardsDataScience, "Basic introduction to separable convolutions," [Online]. Available: https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728.

[13] A. Krizhevsky, "ImageNet Classification with Deep Convolutional Neural Networks," 2012. [Online]. Available: https://papers.nips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[14] TowardsDataScience, "Alexnet, the architecture that challenged cnns," [Online]. Available: https://towardsdatascience.com/alexnet-the-architecture-that-challenged-cnns-e406d5297951.

[15] K. Simonyan, "Very Deep Convolutional Networks for Large-Scale Image Recognition," 2014. [Online]. Available: https://arxiv.org/abs/1409.1556.

[16] K. He, "Deep Residual Learning for Image Recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385.

[17] TowardDataScience, "Residual blocks building blocks of resnet," [Online]. Available: https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec.

[18] J. Long, "Fully Convolutional Networks for Semantic Segmentation," 2014. [Online]. Available: https://arxiv.org/abs/1411.4038.

[19] A. Kendall, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," 2015. [Online]. Available: https://arxiv.org/abs/1511.00561.

[20] L.-C. Chen, "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs," 2017. [Online]. Available: https://arxiv.org/abs/1606.00915.

[21] H. Zhao, "Pyramid Scene Parsing Network," 2017. [Online]. Available: https://arxiv.org/abs/1612.01105.

[22] HackerNews. [Online]. Available: https://www.gushiciku.cn/pl/23BG.

[V. Ramakrishna, "Pose Machines: Articulated Pose Estimation," 2014. [Online].
2 Available: https://www.ri.cmu.edu/pub_files/2014/7/poseMachines.pdf.
3
]

[S.-E. Wei, "Convolutional Pose Machines," 2016. [Online]. Available:
2 https://arxiv.org/abs/1602.00134.
4
]

[A. Newell, "Stacked Hourglass Networks for Human Pose Estimation," 2016.
2 [Online]. Available: https://arxiv.org/abs/1603.06937.
5]

[Z. Cao, "Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields,"
2 2016. [Online]. Available: https://arxiv.org/abs/1611.08050.
6
]

[J. Li, "Simple Pose: Rethinking and Improving a Bottom-up Approach for Multi-
2 Person Pose Estimation," 2019. [Online]. Available:
7] https://arxiv.org/abs/1911.10529.

["CityScapes," [Online]. Available: https://www.cityscapes-dataset.com/.
2
8
]

["Google," 2019. [Online]. Available:
2 https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html.
9
]

[M. Net, "Gluon," [Online]. Available:
3 https://cv.gluon.ai/build/examples_pose/dive_deep_simple_pose.html.
0
]

["ONNX," [Online]. Available: https://onnx.ai/.
3
1]

[OpenVino. [Online]. Available: https://docs.openvinotoolkit.org/latest/index.html.
3
2
]