UNIVERSIDAD NACIONAL DE CÓRDOBA

TRABAJO ESPECIAL DE LICENCIATURA

Automatización de Cefalometrías utilizando métodos de aprendizaje automático

---

Autor:
Francisco NIEVAS

Director:
Dr. Jorge SÁNCHEZ

Tesis presentada en el cumplimiento de los requisitos
para el grado de Licenciatura en Ciencias de la Computación

Diciembre 2019

UNIVERSIDAD NACIONAL DE CÓRDOBA

Resumen

Facultad de Matemática, Astronomía, Física y Computación

Licenciatura en Ciencias de la Computación

Automatización de Cefalometrías utilizando métodos de aprendizaje automático

por Francisco NIEVAS

La cefalometría es un estudio médico que logra diagnosticar problemas dentarios, esqueléticos ó problemas estéticos. Se realiza sobre un trazado obtenido del calco de líneas de las estructuras blandas y duras (piel y hueso respectivamente) a partir de una radiografía lateral de la cara, obtenida del paciente. Una vez obtenido el calco se procede a marcar ciertos *puntos cefalométricos*, además de líneas y ángulos característicos para poder realizar el estudio en sí. En este trabajo especial de la Licenciatura en Ciencias de la Computación se propone utilizar modelos de aprendizaje automático para la generación de cefalometrías. Dichos modelos detectan los puntos cefalométricos en imágenes de rayos x, acelerando el proceso de cómputo del estudio. Se presentaron arquitecturas novedosas, las mismas combinan una arquitectura de un Autoencoder y el uso de redes neuronales convolucionales con capas Inception para asociar a una imagen de entrada un mapa de probabilidades sobre la misma. Se compararon diferentes modelos, llegando a mostrar que estos tienen un excelente desempeño para esta tarea.

UNIVERSIDAD NACIONAL DE CÓRDOBA

Abstract

Facultad de Matemática, Astronomía, Física y Computación

Licenciatura en Ciencias de la Computación

Automatización de Cefalometrías utilizando métodos de aprendizaje automático

por Francisco NIEVAS

Cephalometry is a medical study that is used to diagnose dental, skeletal or aesthetic problems. This study involves tracing the structures of the skin and bone from a lateral x-ray image of the face which is obtained from the patient. Once this is done, cephalometric points need to be marked on top of this tracing besides specific lines and angles. This work proposes to use machine learning models to generate the aforementioned studies. These models can detect cephalometric points in X-ray images, thus reducing the time it takes to conduct this study. These novel architectures presented hereby combine both an autoencoder and the use of convolutional neural networks with inception layers, which is used to associate an input map with a probability map. Different models were compared, showing their excellent performance of this task.

Reconocimientos

- A mi familia, que fueron mi soporte a lo largo de los años para que pueda terminar mi carrera.
- A Jorge Sanchéz, por su experiencia, conocimiento y motivación que me orientó en la investigación de este trabajo final.
- A Ayelén, mi novia y compañera emocional en todo este trayecto.
- A Juan Porta, que me acompañó durante la tesis y me dio un montón de conocimiento en física y matemática
- A mis amigos Agustín, Juan, Trucco y todos con los que compartí dentro y fuera de las aulas. Aquellos amigos de la Facultad, que se convierten en amigos de vida y aquellos que serán mis colegas, gracias por todo su apoyo.
- A Martín, Cindi, Seba y Agus, con los cuales compartí el trabajo, mates y charlas
- A la Universidad pública por permitirme estudiar y capacitarme en tan prestigiosa Facultad, la cual fue mi segundo hogar por tantos años.

Índice general

Capítulo 1

Introducción

1.1. Motivación

Las redes convolucionales (CNN en inglés), ocupan el *estado del arte* para diversas tareas en visión por computadora, han demostrado tener éxito para una amplia gama de aplicaciones, incluida la clasificación de imágenes [19, 14] , la segmentación de imágenes [23], la alineación de imágenes [17], la detección de puntos faciales [4], la estimación de posturas humanas [22], la detección de lineas en caminos [11], entre otras tareas.

En la actualidad, en el campo de la medicina es donde se ha visto una gran tendencia al uso de CNN para automatizar el proceso de detección y diagnóstico de enfermedades [6, 18, 7, 5]

El problema de detectar puntos característicos (definidos según cada tema en particular) para el ámbito de la medicina es muy importante, esto se debe a la posibilidad de mejorar la precisión y el tiempo que demoran algunos estudios médicos, logrando así que se mejore la calidad de vida de las personas. Uno de estos estudios es la cefalometría cuantitativa. El análisis cefalométrico se utiliza en odontología, y especialmente en ortodoncia, para medir el tamaño y las relaciones espaciales de los dientes, las mandíbulas y el cráneo. Este análisis informa la planificación del tratamiento, cuantifica los cambios durante el tratamiento y proporciona datos para la investigación clínica.

La identificación de puntos cefalométricos en los cefalogramas laterales es un problema difícil. El cráneo es un objeto 3D altamente complejo que, en un radiografía, se proyecta en un solo plano 2D, lo que lleva a estructuras superpuestas. Además, la asimetría facial, las variaciones de posicionamiento de la cabeza durante la adquisición de la imagen y la distorsión radiográfica hacen que los contornos izquierdo y derecho no se superpongan perfectamente, lo que lleva a estructuras duplicadas. Esto, combinado con la variación anatómica individual, en particular en casos patológicos, hace que sea muy difícil posicionar de manera confiable los puntos cefalométricos.

Actualmente en la práctica clínica, las posiciones de los puntos cefalométricos se identifican de forma manual o semiautomática, lo cual es muy tedioso, requiere mucho tiempo y es propenso a inconsistencias entre el mismo ortodoncista y entre diferentes ortodoncistas (errores *inter-observador* e *intra-observador* respectivamente). Los distintos niveles de entrenamiento y experiencia en ortodoncia tienen un impacto en las variaciones inter-observador, y las limitaciones de tiempo y otros compromisos pueden tener un impacto en la intra-observador [9]

Algunas formas de optimizar la marcación de los puntos cefalométricos es el uso de algún software que permita ayudar o aconsejar al profesional a la hora de marcar los puntos. Este tipo de software no elimina completamente la necesidad de contar con un profesional, sino que le provee herramientas para hacer su labor mas

sencilla. Uno de estos es el caso de CefMed , que permite realizar todo el marcado de los puntos cefalométricos desde su plataforma, sin necesidad de tener en formato físico la imagen de rayos x del paciente.

Sin embargo, no existe una solución que sea completamente automática y lo suficientemente precisa para este problema, por lo cual se ha convertido en un foco de investigación científica en los últimos años. Actualmente, se dispone de bases de datos públicas y etiquetadas (o sea, ya analizadas previamente por profesionales expertos), con lo cual se hace fácil el poder investigar y desarrollar experimentos y comparar el desempeño del mismo con una metodología estándar

En el año 2014, con el fin de fomentar la búsqueda de una solución a este problema, el Simposio Internacional de Imágenes Biomédicas del IEEE (*IEEE International Symposium on Biomedical Imaging* 2014 **(ISBI 2014)** presentó la competencia "Automatic Cephalometric X-Ray Landmark Detection Challenge"[26]. El objetivo de esta competencia fue investigar tecnologías adecuadas para la detección de puntos cefalométricos en imágenes de Rayos-X y proporcionar un marco de evaluación estándar con un conjunto de datos reales. En estas competencias se utilizaron en su mayoría modelos de detección de landmarks basados en el método de Random Forest para segmentación y posterior asignación de los puntos

Una de las soluciones presentadas fue la de **Ibragimov** et al. [13], planteó una solución utilizando teoría de juegos y *Random Forest*. **Arik** et al. 2017 [3], propuso un sistema basado en redes neuronales convolucionales para para la detección de los puntos cefalométricos, la misma, generaba un mapa de verosimilitud de los puntos cefalométricos basados en la intensidad de los píxeles, luego, se combinaba con un modelo de Random Forest para la generación de un mapa basado en la distribución de cada punto con respecto al resto. Esta idea logró superar a los modelos presentados, para una precisión de 2mm, que en odontología se considera el máximo error aceptable. Juan Ignacio Porta [21] introdujo una arquitectura novedosa, la cual combina el uso de CNN con capas Inception y la arquitectura de un Autoencoder para asociar a una imagen de entrada un mapa de probabilidades sobre la misma.

Capítulo 2

Marco Teórico

2.1. Cefalometrías

La cefalometría se realiza sobre un trazado obtenido del calco de líneas de las estructuras blandas y duras (piel y hueso respectivamente) a partir de una radiografía lateral de la cara, obtenida del paciente. Una vez obtenido el calco se procede a marcar ciertos *puntos cefalométricos*, dependiendo del tipo de cefalometría y análisis que se esté realizando (por ejemplo, Bjork, Steiner, Ricketts). Luego, se calculan varias mediciones lineales y angulares desde las posiciones de los *puntos cefalométricos*. La precisión con la que se ubican estos puntos tiene un impacto directo en los resultados de los análisis realizados y las decisiones de tratamiento resultantes. Según unas normas determinadas, se comparan los resultados obtenidos y permite diagnosticar al paciente. A continuación se describen diferentes tipos de cefalometrías, según cada autor. Mostrando los puntos cefalométricos y el tipo de uso que posee cada una.

2.1.1. Cefalometría de Bjork-Jarabak

El análisis de Bjork-Jarabak es útil para determinar las características del crecimiento en sus aspectos cualitativos y cuantitativos, es decir, dirección y potencial de crecimiento. Para este cefalograma se necesita marcar 6 puntos cefalométricos (Puntos N-S-Ar-Go-Me-Pg. Puntos 2, 5, 23, 21, 19 y 17 de la Figura 2.1 respectivamente). Uno de ellos (Go) es construido con la información de 2 puntos cefalométricos auxiliares (PiC y PiR). Por lo tanto en nuestro caso necesitamos identificar un total de 7 Puntos cefalométricos que se pueden observar en la Figura 2.2, además de las estructuras de tejido blando y óseos para poder llevar a cabo la cefalometría en sí.

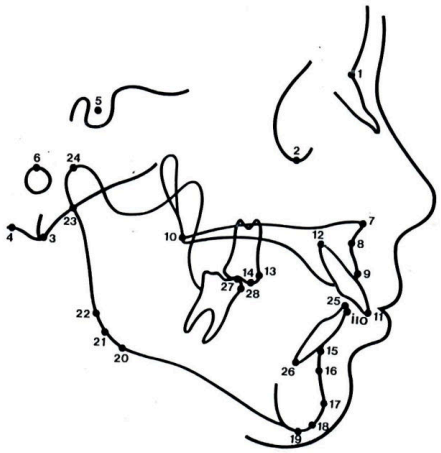


FIGURA 2.1: Todos los puntos cefalométricos del autor Bjork-Jarabak.

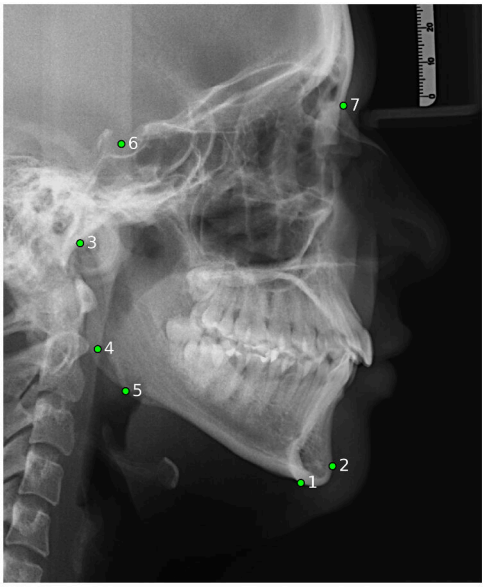


FIGURA 2.2: Puntos cefalométricos necesarios para realizar la cefalometría de Bjork-Jarabak marcados sobre una radiografía. Referencias en la tabla 2.1

CUADRO 2.1: Lista de puntos cefalométricos de Bjork-Jarabak

$N°$ de Landmark	nombre anatómico
L1	Mentoniano (Me)
L2	Pogonio (Pg)
L3	Articular (Ar)
L4	Póstero inferior de la rama mandibular (PiR)
L5	Póstero inferior del cuerpo (PiC)
L6	Silla (S)
L7	Nasion (N)

2.1.2. Cefalometría de Ricketts

En la cefalometría de Ricketts se utilizan medidas que diagnostican diferentes problemas, estos se agrupan en 6 campos:

- Campo 1: Se diagnostica si existe algún problema dentario.
- Campo 2: Hace referencia al problema ortopédico. Se interpreta la relación esquelética, entre el maxilar y la mandíbula.
- Campo 3: Se determina la relación de los dientes respecto al hueso, es decir, el problema óseo-dentario.
- Campo 4: El problema estético se analiza en este campo, donde se observa y evalúa la protrusión labial, la longitud del labio superior y la relación entre la comisura labial y el plano oclusal.
- Campo 5: Se relaciona a los maxilares (las estructuras movibles) con la base del cráneo (la estructura amovible). Es decir, la relación cráneo-facial.
- Campo 6: Se evalúa la estructura interna, como por ejemplo el arco mandibular, la longitud del cuerpo de la mandíbula o la altura facial posterior.

Para este cefalograma necesitamos marcar un total de 36 puntos cefalométricos que se ilustran en la Figura 2.3. A diferencia de Bjork-Jarabak, algunos puntos necesitan información de estructuras dentales (que no siempre son fácilmente identificables en las radiografías) dificultando el proceso de marcado e introduciendo posibles errores.

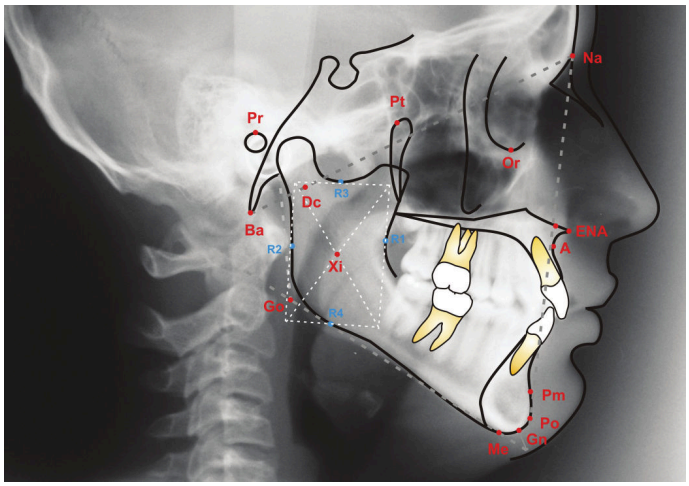


FIGURA 2.3: Algunos puntos cefalométricos de Ricketts marcados sobre RX

2.2. Aprendizaje automático

En este trabajo final se aplicaron técnicas de *aprendizaje automático*. A continuación se hace una introducción general al tema explicando en detalle los métodos utilizados. Se define el aprendizaje automático como un conjunto de algoritmos que pueden detectar automáticamente patrones en datos masivos, y luego utilizar los patrones descubiertos para hacer predicciones, o para realizar otros tipos de toma de decisiones bajo algún criterio. Este "*aprendizaje*" permite a las computadoras realizar tareas específicas de forma autónoma, es decir, sin necesidad de ser programadas específicamente para cada tarea que se desee realizar. El Aprendizaje automático se divide generalmente en dos tipos principales: **Supervisado** y **No supervisado**. El aprendizaje supervisado requiere de datos previamente etiquetados para aprender a realizar su trabajo. Estos datos se dividen en dos subconjuntos, llamados de *entrenamiento* y *evaluación*. El modelo aprende utilizando los datos de entrenamiento, mientras que los de evaluación son reservados para medir el rendimiento del modelo entrenado. En cambio, en el aprendizaje no supervisado, los algoritmos no necesitan de datos etiquetados, pero sí de indicaciones previas, las cuales le enseñan a comprender y analizar la información que le será brindada. Por ejemplo: Agrupar o clasificar información, etc.

2.3. Redes Neuronales

El modelo neuronal mas simple es el *Perceptrón* [24], su estructura está representada en la Figura 2.4. Está compuesto por entradas $\{x_i\}_{i=1}^{D}$, pesos entrenables $\{w_i\}_{i=1}^{D}$ y un término independiente o *bias* w_0 como se muestra en la Eq. (2.1). Dado que el Perceptrón tiene una capa única de salida, se denomina red neuronal de una sola

capa. Sea $x^D \in \mathbb{R}^D$, el valor de salida y se obtiene por una función de activación $f(.)$ tomando la suma de pesos de entrada de la siguiente manera:

$$y(\boldsymbol{x}, \Theta) = f(\sum_{i=1}^{D} x_i w_i + w_0) = f(\boldsymbol{w}^\top \boldsymbol{x} + w_0) \tag{2.1}$$

donde $\Theta = \{w, w_0\}$ es un conjunto de parámetros, $\{w_i\}_{i=1}^{D} \in \mathbb{R}^D$ es un vector de pesos y w_0 es el *bias*. Definamos $z = \boldsymbol{w}^\top \boldsymbol{x} + w_0$. En este trabajo final se utiliza como funciones de activación $f(.)$ a las funciones ***ReLu*** y ***Sigmoide***:

- ⋆ ***ReLu:*** $\sigma(z) = max(0, z)$
- ⋆ ***Sigmoide:*** $\sigma(z) = \dfrac{1}{1 + e^{-z}}$

Si bien es posible aplicar diferentes tipos de funciones de activación para capas diferentes o incluso unidades diferentes, en la literatura es común aplicar el mismo tipo de función de activación para las capas ocultas. Sin embargo, siempre tiene que ser una función no lineal, de lo contrario, se podría representar a la red por una red equivalente compuesta por una única capa que cuente con una matriz de pesos construida a partir de multiplicar cada una de las matrices de pesos de la red original.

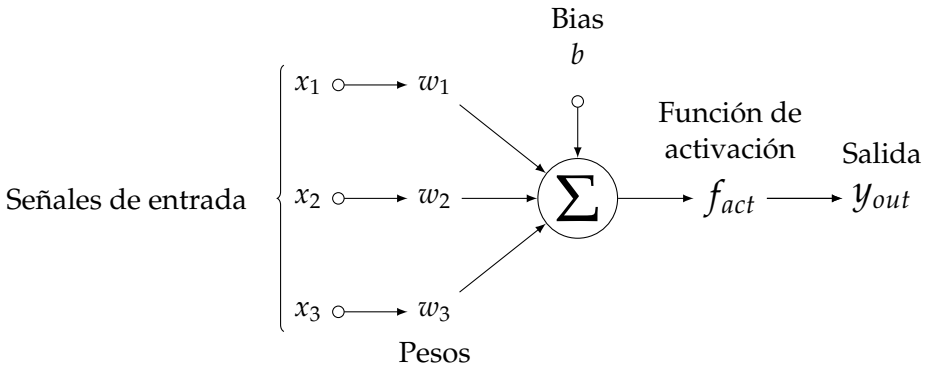


FIGURA 2.4: Esquema de un Perceptrón. x_1, x_2, x_3 son las entradas de la neurona. w_1, w_2, w_3 son las componentes de w

Si empezamos a combinar perceptrones uniendo la salida de uno con la entrada de otro, se forman distintas capas, y por lo tanto lo que obtenemos es una red feed-forward como la de la Figura 2.5. Consiste de al menos tres capas: una de entrada, una de salida y, como mínimo, una capa oculta.

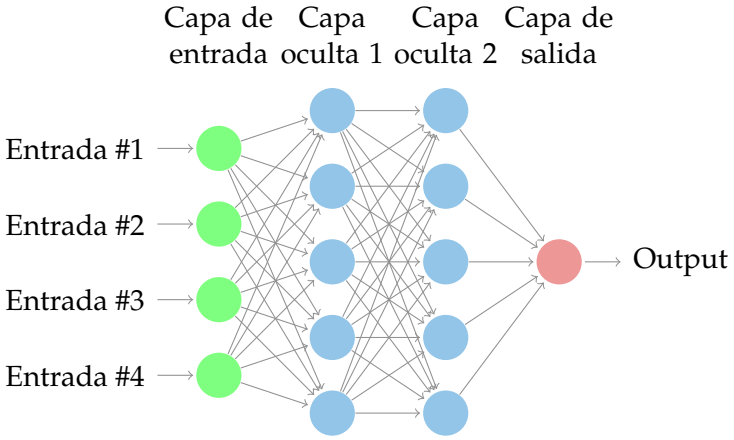


FIGURA 2.5: Arquitectura de una red Feed-Forward

2.3.1. Aprendizaje en redes Feed-Forward

El problema de aprender los parámetros de una red neuronal feed-forward puede formularse como la minimización de una función de error. Dado un conjunto de entrenamiento $\{\boldsymbol{x}_n, \boldsymbol{t}_n\}_{n=1}^{N}$, donde $\boldsymbol{x}_n \in \mathbb{R}^D$ representa una observación o un dato y $\boldsymbol{t}_n \in \{0,1\}^K$ denota un indicador de clase de la forma de *one-of-K*, es decir, para una clase k solo el k-esimo elemento en un vector $\boldsymbol{t}_n$ es 1 y el resto de elementos es 0. Para un problema de clasificación de **K** clases, es común utilizar *cross-entropy* como función de costo, la misma esta definida de la siguiente manera:

$$E(\Theta) = -\frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} ln(y_{nk}) \tag{2.2}$$

donde t_{nk} denota el k-esimo elemento del vector $\boldsymbol{t}_n$ y y_{nk} es el k-esimo elemento del vector de predicciones $\boldsymbol{y}_n$ para $\boldsymbol{x}_n$ La función de costo en la Eq. (2.2) es altamente no lineal y no convexa. Por lo tanto, no hay una solución analítica del conjunto de parámetros que minimiza la ecuación, en cambio, se utiliza un algoritmo de descenso de gradiente actualizando los parámetros de manera iterativa. Para utilizar un algoritmo de descenso de gradiente, se requiere una forma de calcular un gradiente $\nabla E(\Theta)$ evaluado en un conjunto de parámetros Θ.

2.4. Redes Convolucionales

En las redes neuronales multicapa convencionales, las entradas están siempre en forma vectorial. Sin embargo, para imágenes (médicas), la información estructural o entre píxeles o voxels vecinos es otra fuente de información. Por lo tanto, la vectorización destruye inevitablemente dicha información estructural y de configuración en imágenes. Una forma particular de redes neuronales son las redes convolucionales (CNN), normalmente tienen capas convolucionales intercaladas con capas de *pooling* (o *sub-sampling*) y luego seguidas por capas totalmente conectadas (*fully connected layers*) como en una red neuronal multicapa estándar.

Si definimos nuestra función

$$f(\boldsymbol{x}) = \left(\sum_i \boldsymbol{K}_i \star x_i\right) + b \tag{2.3}$$

donde $\star$ representa la operación de convolución discreta, b representa el término independiente o *bias* y $\mathbf{K}_i$ representa un núcleo de convolución, con parámetros ajustables, entonces obtenemos una neurona convolucional. Utilizando los conceptos previamente explicados se puede construir redes feed-forward con neuronas convolucionales, en vez de perceptrones.

La Figura 2.6 muestra un ejemplo de una convolución discreta, en su versión de dos dimensiones (la cual es utilizada en datos de tipo imagen). La grilla de color azul claro se denomina mapa de características de entrada. En el dibujo se representa un solo mapa de características de entrada, pero es común tener varios mapas apilados uno sobre otro, por ejemplo, en el caso de imágenes RGB. Un kernel (de color gris) se desliza sobre el mapa de entrada. En cada ubicación, se calcula el producto entre cada elemento del kernel y el elemento de entrada que se superpone y se resumen los resultados para obtener la salida en la ubicación actual.

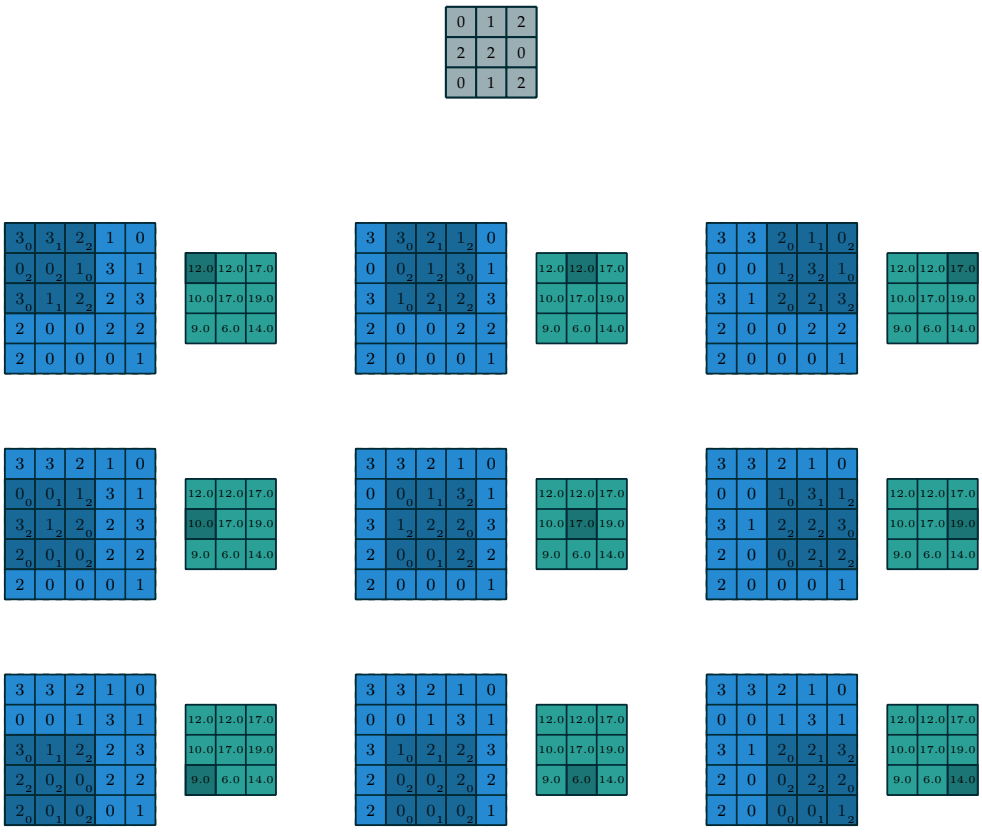


FIGURA 2.6: Ejemplo de convolución discreta en dos dimensiones. El núcleo se presenta en gris, la entrada en celeste y la salida en verde. Imagen extraída de [8]

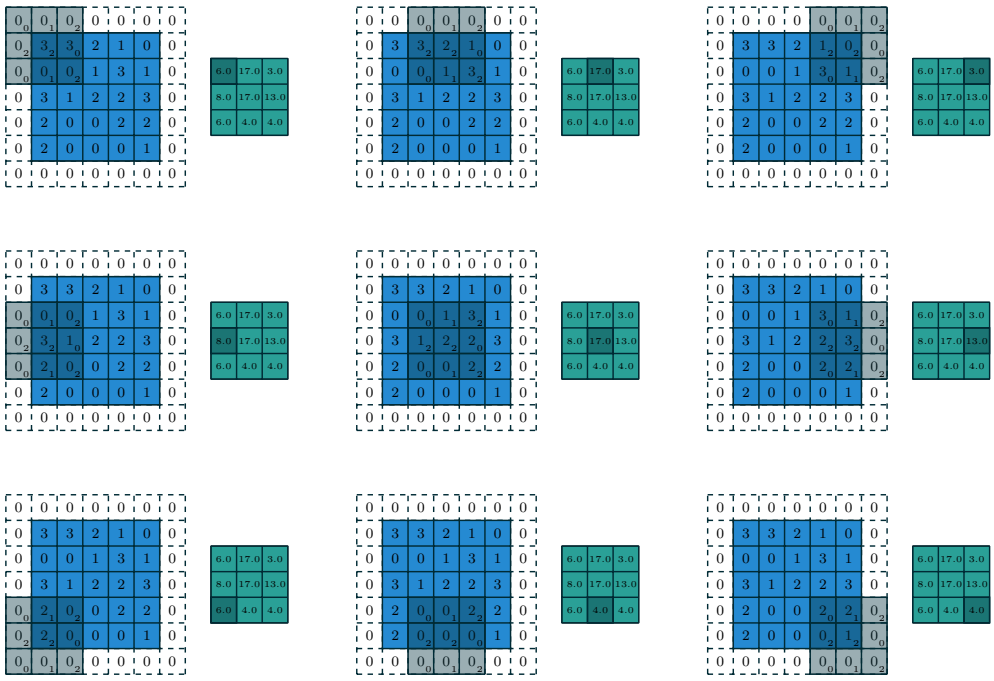


FIGURA 2.7: Ejemplo de convolución discreta con Padding. Imagen extraída de [8]

Las redes convolucionales poseen diversos *parámetros* que modifican el comportamiento de la misma. En primer lugar, en las capas convolucionales es posible determinar el tamaño del kernel con el cual se van a realizar las operaciones de convolución. Un kernel grande puede lograr aprender mayor información pero a un costo computacional mayor. En segundo lugar, en las capas convolucionales se puede utilizar padding, el mismo permite aumentar la dimensión de salida de una convolución añadiendo información en los mapas de entrada. Existen varias formas de añadir información: Con ceros, con valores iguales a los bordes de la imagen, valor medio de la imagen, etc. En la Figura 2.7 se puede observar una convolución con padding de ceros. Por último, se puede utilizar una técnica de sub-muestreo o Stride. Esto constituye una forma de reducir la dimensión de salida de una convolución. El valor de *Stride* indica cada cuantos índices realizo una operación de convolución. Como alternativa, puede ser interpretado como la cantidad de la salida que se retiene. Por ejemplo, mover el kernel por saltos de dos es equivalente a mover el kernel por saltos de uno pero manteniendo solo elementos de salida impares. Figura 2.8

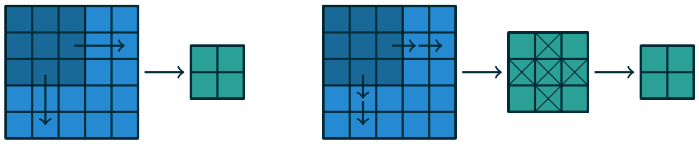


FIGURA 2.8: Una forma alternativa de ver los *strides* En lugar de trasladar el kernel de 3 x 3 en incrementos de $s = 2$ (izquierda), el kernel se traduce en incrementos de 1 y solo se retiene uno de los elementos de salida $s = 2$ (derecha). Imagen extraída de [8]

2.4.1. Capas de *pooling*

Una capa de *pooling* se aplica luego de una capa convolucional para comprimir los mapas de activación de la capa convolucional anterior. Específicamente, cada mapa de características en una capa de pooling se vincula con un mapa de características en la capa de convolución, y cada unidad en un mapa de características de la capa de agrupación se calcula en función de un subconjunto de unidades en su campo receptivo. En la figura 2.9 se puede observar una comparación entre una capa convolucional con kernel de 3x3 y una capa de pooling con kernel de 9x9, se puede observar como las capa de pooling logra reducir y comprimir la información reteniendo las características mas importantes del mapa de activación.

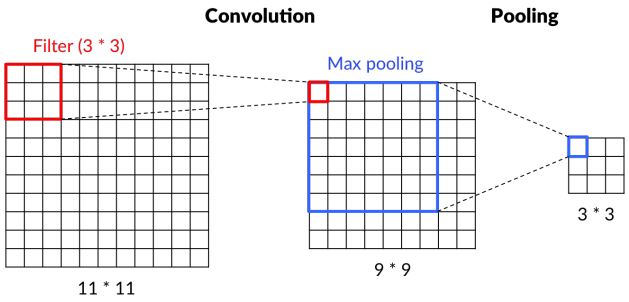


FIGURA 2.9: Comparación entre capa convolucional y capa de pooling

Capítulo 3

Propuesta

El objetivo propuesto es el de automatizar el proceso de generación de las cefalometrías de Bjork-Jarabak y Ricketts. Como se explicó en 2.1, una de las etapas consiste en marcar puntos cefalométricos dependiendo del tipo de cefalometría que se quiera realizar. Para realizar un modelo de aprendizaje supervisado es necesario contar con información etiquetada, hasta el momento la única información con la que se contaba era el dataset público de la competencia "Grand Challenges in Dental X-Ray Image Analisys. Challenge #1: Automated Detection and Analisys for Diagnosis in Cephalometric X-Ray Image". El dataset esta compuesto por un conjunto de *entrenamiento* de 150 imágenes y 2 conjuntos de *evaluación* de 100 y 150 imágenes respectivamente. El mismo contiene imágenes de pacientes de entre 6 y 60 años. Contiene diecinueve de los puntos cefalométricos mas populares como puntos de referencia para detectar. La Figura 3.1 muestra los puntos utilizados en la competencia. Los tamaños de las imágenes originales son de 2400 x 1935 píxeles, y la resolución es de 0,1 mm / píxeles en ambas direcciones. Las imágenes fueron comprimidas a un tercio de la imagen original (tomando el promedio de cada parche de 3 x 3) para fines de reducción de dimensionalidad, esto reduce la complejidad computacional sin perder información significativa, resultando una imagen de 800x645.

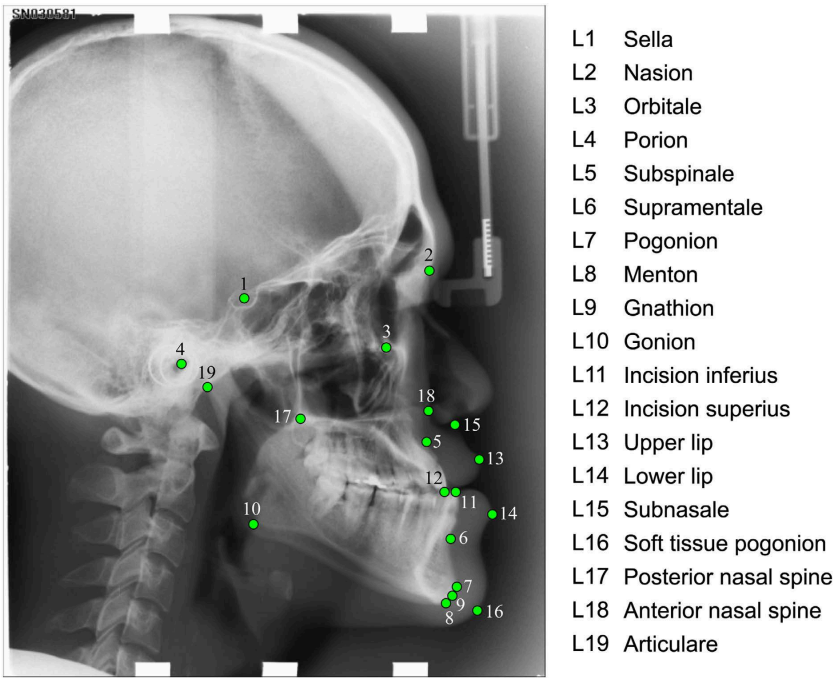


FIGURA 3.1: 19 puntos cefalométricos de la competencia ISBI 2014. Imagen extraída de [15]

Los diecinueve puntos del dataset no conforman ningún cefalograma en específico. Por esta razón, no es posible entrenar un modelo de aprendizaje automático que detecte todos los puntos de un cefalograma. Sin embargo, es útil para poder comprobar la *performance* del modelo y poder compararlo con otros modelos existentes en la literatura. Para solucionar este problema se necesitaba crear un dataset que contenga información de los puntos cefalométricos de cada uno de los distintos cefalograma, Bjork-Jarabak y Ricketts en este caso.

3.1. Creando un dataset

Trabajando en conjunto con la empresa CefMed se logró conseguir imágenes etiquetadas de distintos tipos de cefalogramas. Con estos datos, idealmente, se podría realizar un modelo de aprendizaje automático para detectar y etiquetar automáticamente imágenes de rayos x para cada tipo de cefalograma. Cuando se analizaron los datos se encontraron varios problemas:

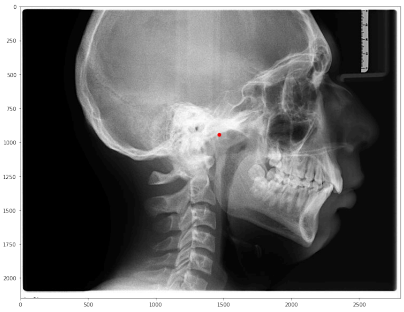
1. Las imágenes de rayos x poseían distintos tamaños (distintas relaciones de aspecto) y resoluciones (mm / píxeles).
2. Las imágenes no fueron tomadas con los mismos equipos, ni por los mismos operarios.
3. Algunas imágenes, originalmente, no eran digitales y se les había aplicado algún proceso de digitalización como escaneado o directamente una fotografía ya sea sobre un monitor o un negatoscopio.
4. Las imágenes no tenían los mismos niveles de contraste y brillo.
5. Las imágenes fueron etiquetadas por distintos profesionales.

Dados estos problemas y más que fueron apareciendo a lo largo de este trabajo final es que armar un dataset no es tarea fácil. Para cada uno de estos problemas se tuvo que pensar en una solución.

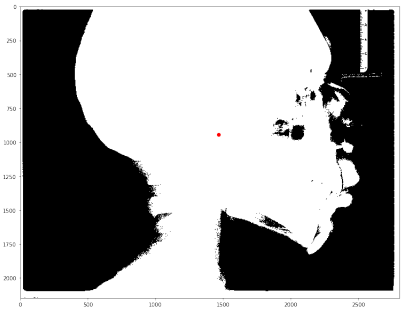
Como se mencionó anteriormente, las imágenes fueron etiquetadas por distintos profesionales. Según un experimento [1, Capítulo 2] la diferencia en la estimación de los puntos cefalométricos varía de 0,4mm a 3,7 mm (DE = 0,2-2,5 mm) para un mismo examinador, y de 0,6 a 5,3 mm (DE = 0,2-3,2 mm) para distintos examinadores. Según los profesionales en el área, un cefalograma tiene una tolerancia de 2mm de error. Esto significa que un punto puede estar ubicado a 2mm de su distancia real y los valores producidos por el estudio cefalométrico siguen siendo aceptables. Por lo tanto crear un dataset con datos provenientes de distintos profesionales aumentaría la varianza en las etiquetas de cada punto, logrando que la tarea sea mas difícil de realizar. Debido a esto, cada *dataset* fue construido con datos provenientes de un único médico. Para solucionar el problema de que las imágenes no tenían los mismos niveles de brillo y contraste se tuvo que normalizar los datos para que coincidieran niveles de brillo, colores (algunas imágenes eran RGB con tintes azulinos). Ya que las imágenes poseían diferentes relaciones de aspecto se necesitó elaborar un procedimiento para poder recortar las imágenes normalizando el tamaño de las imágenes, pero que también mantenga la información de cada una. Para ello, se tomaron dos aproximaciones diferentes: Recortar utilizando binarización y utilizar información extra para detectar la zona de interés.

3.1.1. Recortando imágenes utilizando binarización

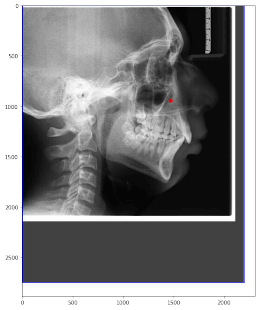
Una vez obtenidas las imágenes que se iban a utilizar, se tenían que recortar para que todas tengan una misma relación de aspecto. Como primera opción optó por calcular el centroide de la imagen. Para ello, se binarizaron las imágenes con un cierto *threshold*, es decir, valores menores que el *threshold* se asignan a *cero* y los mayores a *uno*. Con esta información se computa la ubicación espacial promedio (promedio de todas las coordenadas x y coordenadas y) de los elementos que tienen asignados *unos*, y en esa ubicación se coloca un *box* con una relación de aspecto fija según la entrada de los modelos convolucionales existentes, es decir de 800px X 600px. Como se puede ver en la Figura 3.2, al recortar con el *box* algunas partes estaban fuera de la imagen y se tenia que rellenar ese espacio faltante con información extra. Se probó con: *ceros*, valor promedio de la imagen, valores constantes según los bordes de la imagen original. Este método no generó buenos resultados para ninguna de las configuraciones antes mencionadas ya que introducía ruido innecesario en las imágenes.



(A)



(B)



(C)

FIGURA 3.2: Crop de RX utilizando centroides. 3.2a Imagen original, 3.2b imagen con *threshold*, 3.2c imagen recortada utilizando valores promedios para rellenar información faltante.

3.1.2. Recortando imágenes utilizando estructura blanda

Al ver que el método anterior no obtuvo los resultados deseados se decidió buscar otro método, que fuera mas eficaz para recortar imágenes y que utilice toda la información extra con la cual se contaba de cada imagen. Para cada una, además de marcar los puntos cefalométricos, los ortodoncistas marcan los perfiles blandos y duros de las imágenes de rayos X. Es decir, la piel y los huesos. Con esa información podemos deducir de la imagen original donde se encuentra la región de interés para nosotros.

La información de cada estructura estaba representada en forma de un conjunto de puntos, que luego se utilizan para hacer una interpolación y lograr una línea que se adapte al contorno que se quiere marcar. Utilizando estos puntos, dados en forma de coordenadas (x, y), se construyó un *box* que contenía a todos los puntos de todas las estructuras, esto se repitió para cada imagen y se guardó las medidas de cada *box*. Luego, se hizo un análisis estadístico y se pudo calcular un *box* con las medidas promedio. Esto se hizo con dos fines: Primero, realizar un análisis exploratorio de los datos, para poder obtener una idea del tipo de imágenes con las que se contaba. Segundo, obtener la relación de aspecto que tendrían que tener las imágenes recortadas.

Con los datos obtenidos, se llegó a la conclusión de que la relación de aspecto de las imágenes difería de la del dataset del Challenge. Por lo tanto, se fijó un tamaño de *box* de (790Px x 653Px) para que tenga un tamaño similar a las imanes del Challenge. Utilizando este *box* el procedimiento era el siguiente: Primero se tomaba una imagen sin procesar, en ella se detectaban todos los puntos (cefalométricos y de estructuras) para poder ubicar el *box*. En el caso de que el tamaño del box sea chico, se agrandaba manteniendo la misma relación de aspecto, es decir aumentando de cada lado del box de manera equitativa, hasta que todos los puntos estaban dentro del mismo. Una vez que se determinó el box, se corta la imagen y esta lista para usarse. Si se toma un tamaño de box mas grande, para poder utilizarla es necesario comprimirla, pero dado que tiene la misma relación de aspecto que la entrada de nuestro modelo esto se puede hacer sin deformar la imagen.

En la Figura 3.3 se puede visualizar el procedimiento antes mencionado. En azul se encuentra el *box* (el cual necesito ser expandido dadas las dimensiones de la imagen) que será utilizado para recortar la imagen. Esta imagen recortada tendrá que ser reducida a un tamaño de (790x653) para luego poder ser utilizada. Con distintas lineas de colores se observan las estructuras de tejidos blandos y duros. Los puntos sobre las lineas son las coordenadas (x, y) que son utilizadas para chequear que el *box* esta bien ubicado.

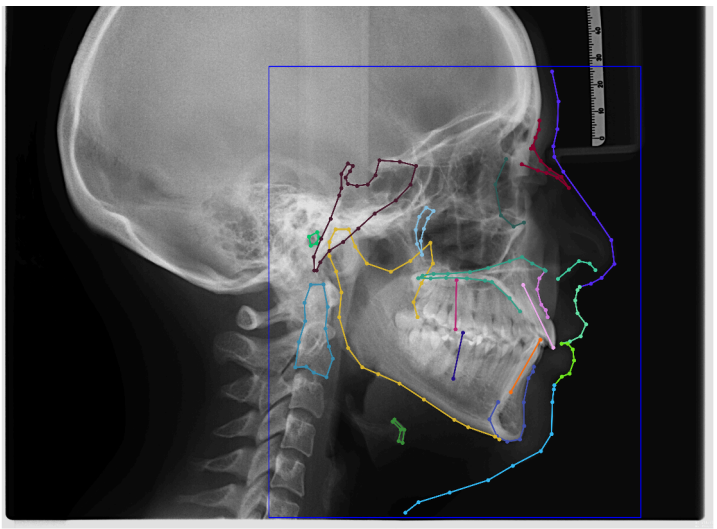


FIGURA 3.3: Imagen original. En azul se muestra un *box* que abarca todos los puntos de las estructuras y por donde sera recortada. Cada línea de color representa una estructura distinta.

3.2. Capas Inception

Hay una manera simple pero poderosa de crear mejores modelos de aprendizaje profundo. Simplemente se puede hacer un modelo más grande, ya sea en términos de profundidad, es decir, número de capas o número de neuronas en cada capa. Pero como es de imaginarse, puede crear complicaciones: Más grande el modelo, más propenso a sobre ajustar, esto es particularmente notable cuando los datos de entrenamiento son pocos. Si aumentamos el número de parámetros significaría un aumento de recursos computacionales.

Supongamos, por ejemplo, que una capa en nuestro modelo de aprendizaje profundo ha aprendido a enfocarse en partes individuales de una cara como la que aparece en la Figura 3.4. La siguiente capa de la red probablemente se centraría en la cara general de la imagen para identificar los diferentes objetos presentes allí. Ahora para hacer esto realmente, la capa debe tener los tamaños de los filtros apropiados para detectar diferentes objetos. En la Figura 3.5 se puede observar que el tamaño de las estructuras para el caso de imágenes de rayos x son pequeñas, estas varían de persona en persona según edad, etnia, etc.

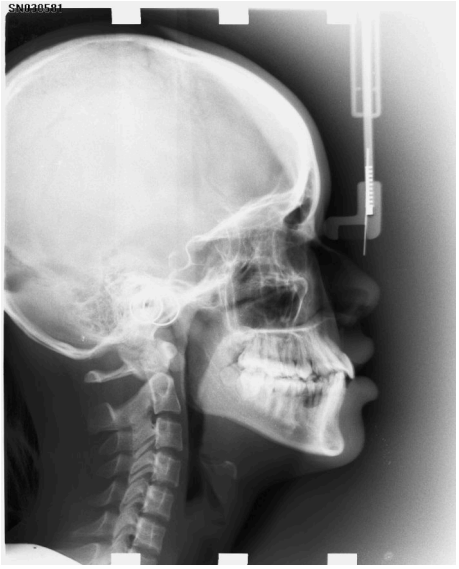


FIGURA 3.4: Imagen de rayos x extraída del dataset del ISBI 2014.

FIGURA 3.5: Izquierda: 2X zoom de la imagen original. Derecha: 4X zoom de la imagen original

Aquí es donde la capa de *Inception* [25] se destaca. Permite que las capas internas busquen y elijan qué tamaño de filtro será relevante para conocer la información requerida. Entonces, incluso si el tamaño de las estructuras en la imagen es diferente, por ejemplo por diferencias en la edad del paciente como en observa en la Figura 3.6, la capa funciona en consecuencia para reconocer las estructuras. Para la primera imagen, probablemente tomaría un tamaño de filtro más alto, mientras que tomaría uno más bajo para la segunda imagen. Se prefiere un filtro más grande para la información que se distribuye de manera global, y se prefiere un filtro más pequeño para la información que se distribuye de manera local.

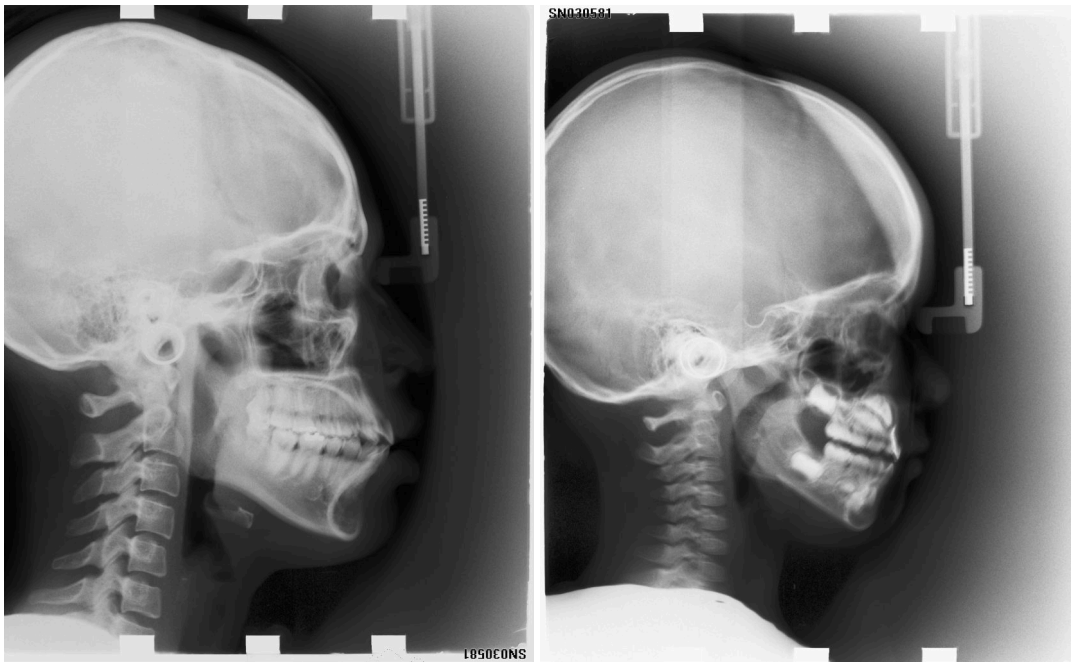


FIGURA 3.6: Dos imágenes provenientes del dataset. Se observa como la diferencia de edad logra que el tamaño de las estructuras sea distinto.

A lo largo del tiempo distintas versiones e iteraciones de las capas *Inception* fueron presentadas. En este trabajo final se hizo uso de las capas *InceptionD* e *InceptionE*. En la Figura 3.7 y 3.8 se puede observar con detalle la estructura del cada módulo.

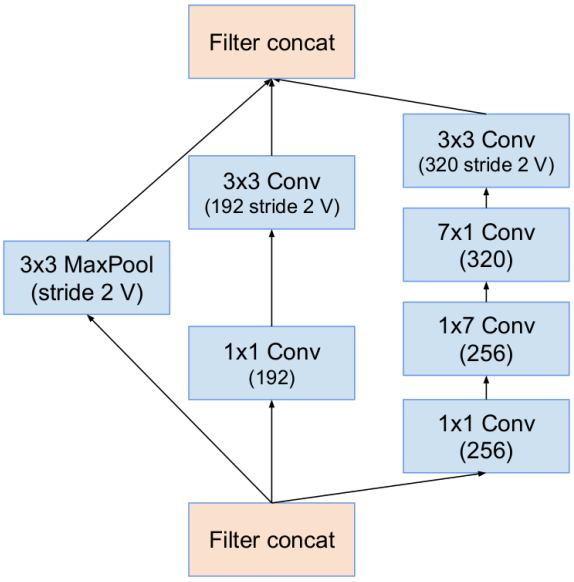


FIGURA 3.7: Capa *InceptionD*. (Fuente Inception v4 [25])

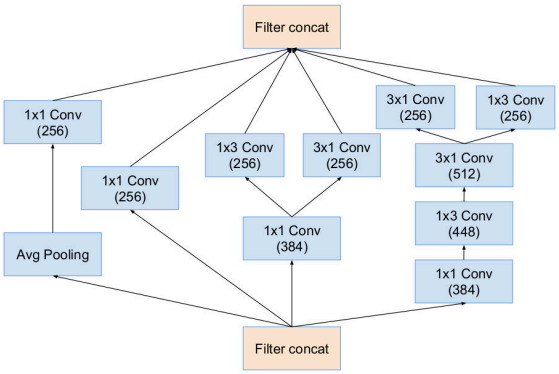


FIGURA 3.8: Capa *InceptionE*. (Fuente Inception v4 [25])

Sin embargo, utilizar estas capas en redes convolucionales tiene como consecuencia un gran costo computacional. Por lo tanto, se hicieron modificaciones en la cantidad de filtros de cada capa, obteniendo así versiones mas mas "livianas". En la Figuras 3.9 y 3.10 se pueden observar los cambios que se realizaron.

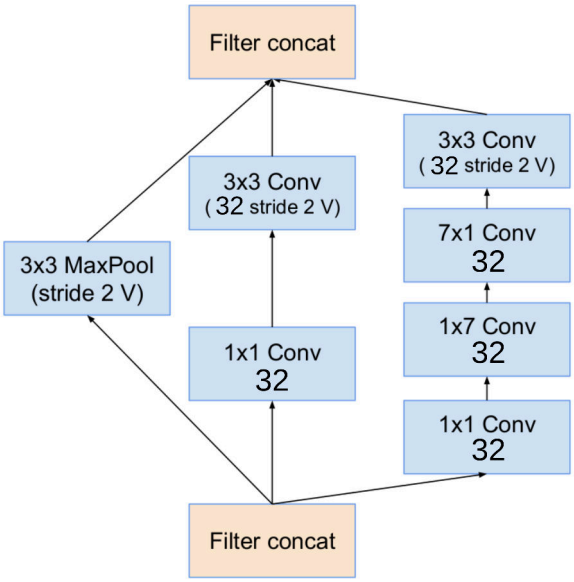


FIGURA 3.9: Capa *InceptionD* modificada.

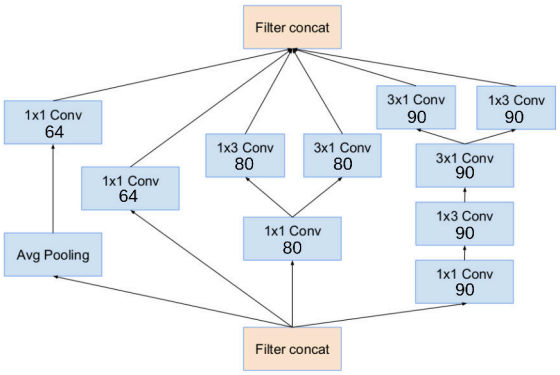


FIGURA 3.10: Capa *InceptionE* modificada.

3.3. Autoencoder

Un *Autoencoder* es un tipo de red neuronal artificial utilizada para aprender codificaciones eficientes de datos de manera no supervisada. La estructura de un *Autoencoder* se puede observar en la Figura 3.11. Esta dividido en dos partes: *Encoder* y *Decoder*. El objetivo del Encoder es aprender una representación (codificación) para un conjunto de datos, típicamente para la reducción de dimensionalidad, entrenando a la red para ignorar el ruido"de la señal. Junto con el lado de reducción, se aprende un lado de reconstrucción, es decir el Decoder, donde la red intenta generar a partir de la codificación una representación lo más cercana posible a su entrada original, de ahí su nombre. Si tomamos de ejemplo una imagen de rayos x, la primera capa del Encoder puede aprender a codificar características básicas como esquinas o bordes, la segunda capa analiza la salida de la primera capa y luego codifica características menos locales como la punta de una nariz, la tercera puede codificar una nariz completa, etc. Las capas del *Decoder* aprenden a decodificar la representación en su forma original lo mejor posible, tratando de reconstruir la imagen original de entrada.

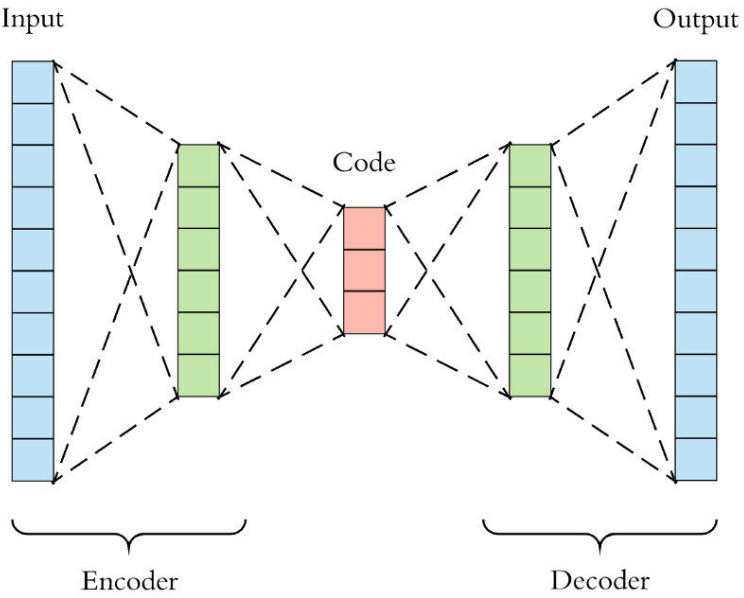


FIGURA 3.11: Esquema de las partes de un *Autoencoder*

A continuación se explica la arquitectura de un Autoencoder con capas *Inception*. Se toma el caso en que la imagen de entrada tiene un tamaño de 800x645 debido al dataset con el que se lo utilizó. Luego se explicará como se modificó para utilizar con otros Datasets que poseen tamaños de imágenes distintas. La arquitectura del *Encoder* se puede observar en la Figura 3.12. La misma esta compuesta por 3 capas de Convoluciones 2-D, cada una seguida por una capa de normalización y como función de activación una *Relu*. La primera capa de Convolución 2-D posee un *Stride* de 2, las dos siguientes poseen un *Stride* de 1. De las tres Convoluciones 2-D solo la última posee *Padding* de 1. A continuación de la última capa de Convolución 2-D hay una capa de *max pooling* con kernel de 2. Seguido se tienen 3 capas *Inception*. Una *InceptionD* y dos *InceptionE*. Al final del *Encoder* tenemos una imagen de 98x79 (aproximadamente una reducción de dimensionalidad de 8 veces). Continuando con la arquitectura de *Decoder* (ver Figura 3.13), se puede observar que posee 5 capas de Convoluciones transpuestas 2-D; la primera y la cuarta tienen *Stride* de 1 el resto cuenta con un *Stride* de 2. A la salida del *Autoencoder* se añade una función *Sigmoide* para obtener un escalar en el intervalo (0,1)

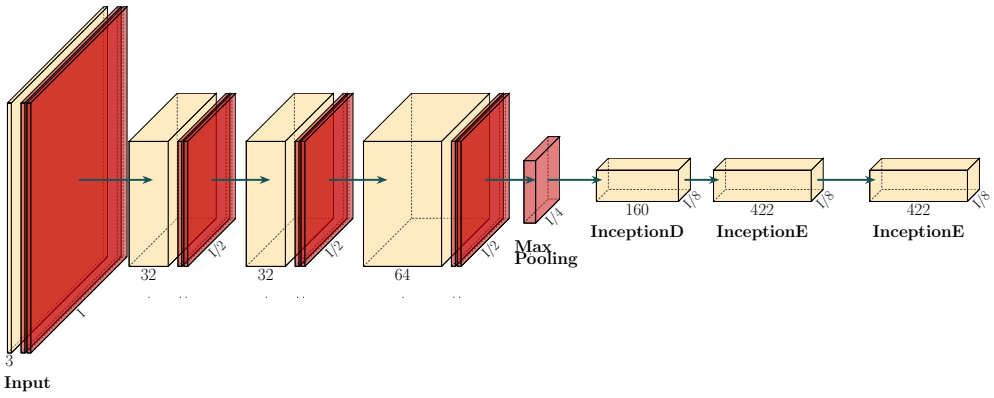


FIGURA 3.12: Arquitectura de un Encoder con capas *Inception*

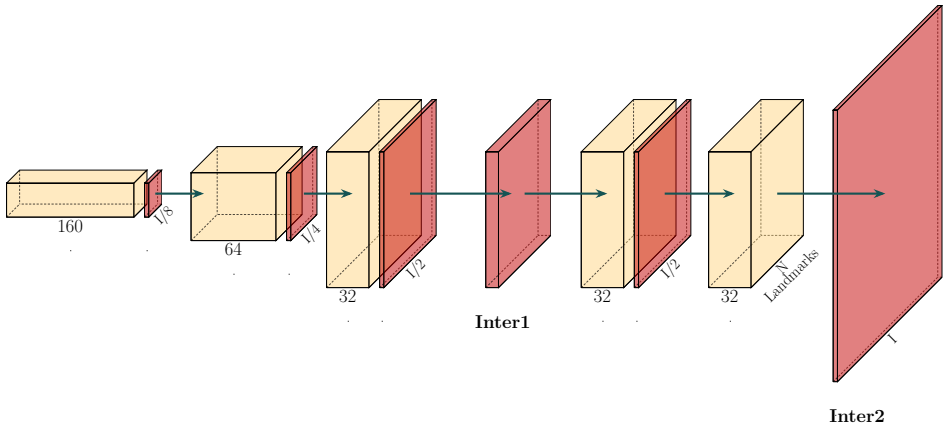


FIGURA 3.13: Arquitectura de un Decoder con capas *Inception*

3.4. Mapas de probabilidad

Como se mostró en secciones anteriores, los autoencoders son útiles en escenarios donde se necesita recrear la información de entrada de la red. Por lo tanto, no podemos utilizar directamente un Autoencoder para detectar puntos cefalométricos en una imagen. Sin embargo, podemos entrenar el modelo para una tarea distinta pero que nos sirva para encontrar los puntos dentro de una imagen. En todos los datasets se contaba con la información de los puntos cefalométricos etiquetados de la forma de coordenadas (x, y) para cada imagen, utilizando estas coordenadas se construyeron mapas de activación para cada punto cefalométrico. Los mapas de probabilidades son generados teniendo en cuenta la ubicación (real) del punto y en esos lugares se centró una función gaussiana con σ relacionado con las restricciones impuestas en las métricas del problema, es decir, detectar con un error máximo de 2mm la posición del Landmark. En las Figuras 3.14 y 3.15 se puede observar el resultado de este procedimiento. En el caso de las imágenes del dataset del ISBI que tienen un tamaño de 800x645 lo que nos quedaría luego de este proceso es un conjunto de mapas de activación de la forma $(19, 800, 645)$, es decir, uno por cada punto cefalométrico. Por lo tanto, utilizando estos mapas de activación y la imagen original como entrada de un Autoencoder, se lo puede entrenar para que replique esta información. En el proceso logrando que el autoencoder aprenda a generar mapas de activación para cada punto, donde las zonas de mayor activación se puede tomar como la predicción de las coordenadas de cada punto.

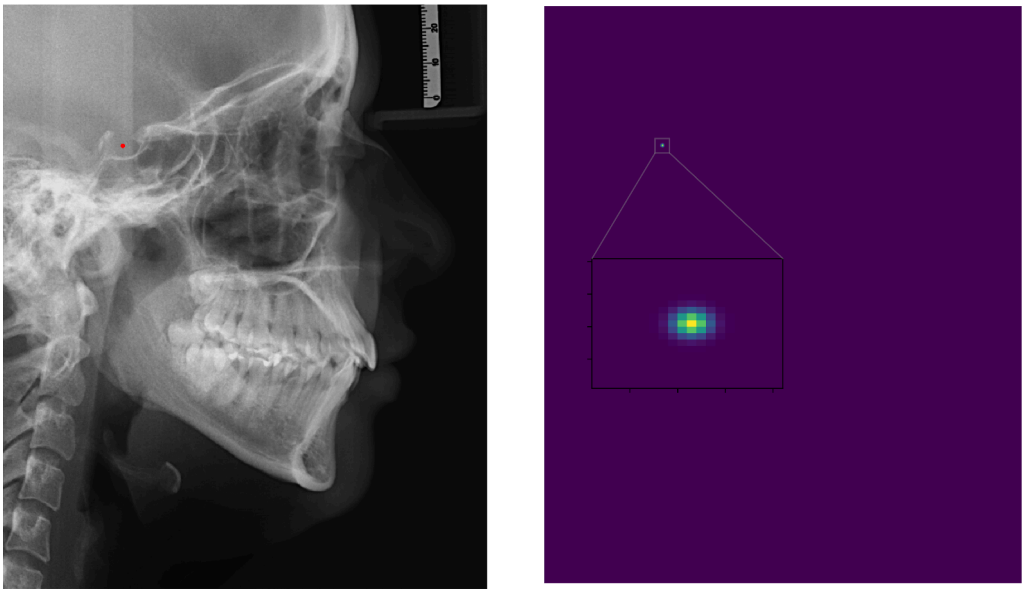


FIGURA 3.14: Izquierda: Imagen original con el punto Silla marcado en rojo (ver 2.2). Derecha: Mapa de calor generado con una activación gaussiana centrada en las coordenadas del punto.

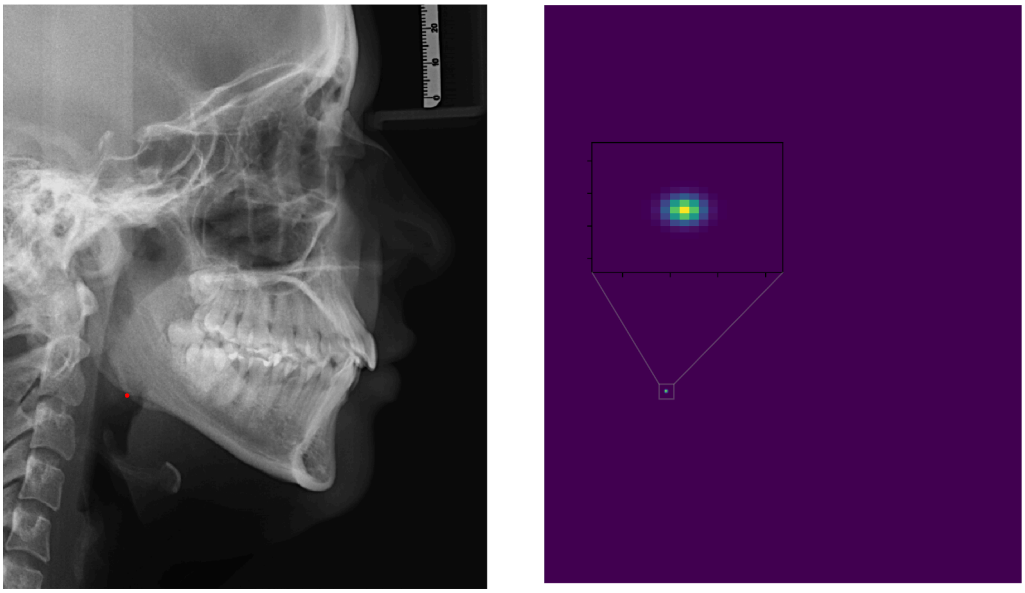


FIGURA 3.15: Izquierda: Imagen original con el punto Gonion marcado en rojo (ver 2.2). Derecha: Mapa de calor generado con una activación gaussiana centrada en las coordenadas del punto.

3.4.1. Función de costo

Cuando se utilizan modelos como los Autoencoders se aborda el problema como una regresión, es decir, se está tratando de ajustar una función cuya imagen pertenece a $\mathbb{R}^n$. Esto nos llevaría a pensar que la función de costo adecuada es el Error Cuadrático Medio, habitual en problemas de regresión, pero si tenemos en cuenta la

particularidad de nuestra salida, la imagen de nuestro modelo es $[0,1]^n \subseteq \mathbb{R}^n$ y estamos comparando distribuciones de probabilidad, por lo tanto una función adecuada para esto es la Entropía Cruzada Binaria. En este caso, se hará uso de una versión modificada, que incluye la salida Sigmoide dentro de la función de costo, para mayor estabilidad numérica, y que además permite pesar la clase positiva para realizar un intercambio entre *recall* y *precisión*. Con esto en cuenta, la función de costo queda definida de la siguiente manera:

$$\frac{1}{N}\sum_{i=1}^{N} -w_i[p_i y_i \cdot log\ \sigma(x_i) + (1-y_i) \cdot log\ (1-\sigma(x_i))] \tag{3.1}$$

donde: N es el número de elementos en el batch, w_i es un factor de peso (opcional) entre los elementos del batch, y_i es la clase a comparar (0 o 1), x_i es la salida del modelo, $x_i \in [0,1] \subseteq \mathbb{R}$, p_i es el peso otorgado a la clase positiva, σ es la función Sigmoide El peso p_i será considerado un hiper-parámetro a ajustar, que nos permitirá cambiar el modelo variando el hiper-parámetro.

3.4.2. Autoencoder Inception con dataset nuevo

En base a los dataset creados se procedió a realizar cambios en la estructura del Autoencoder Inception. La nueva arquitectura recibe imágenes de (790Px x 653Px) En cuanto a la salida, ahora solo tenemos 7 (y 36 respectivamente) capas de salida dependiendo del tipo de Cefalometría que estemos tratando de predecir. Mas allá de los cambios antes mencionados, la arquitectura es idéntica a la de la sección 3.3

3.5. Pruebas preliminares

En la siguiente sección se describen los diferentes experimentos preliminares que se realizaron con el fin de obtener mayor conocimiento del problema y poder explorar distintos acercamientos a la solución.

Para el primer experimento se tomó como base la arquitectura del Autoencoder Inception 3.4.2, en el mismo se duplicaron la cantidad de filtros de las capas convolucionales del *Decoder*. Esto se hizo debido a que la cantidad de parámetros entrenables estaba mayormente localizada en el *Encoder*, por lo tanto la idea era balancear la cantidad de parámetros a lo largo de toda la arquitectura. Logrando que ambas partes de la arquitectura tengan la mismas capacidad para poder aprender la tarea para la cual se los estaba entrenando. En la Figura 3.16 se puede observar la nueva arquitectura del *Decoder*

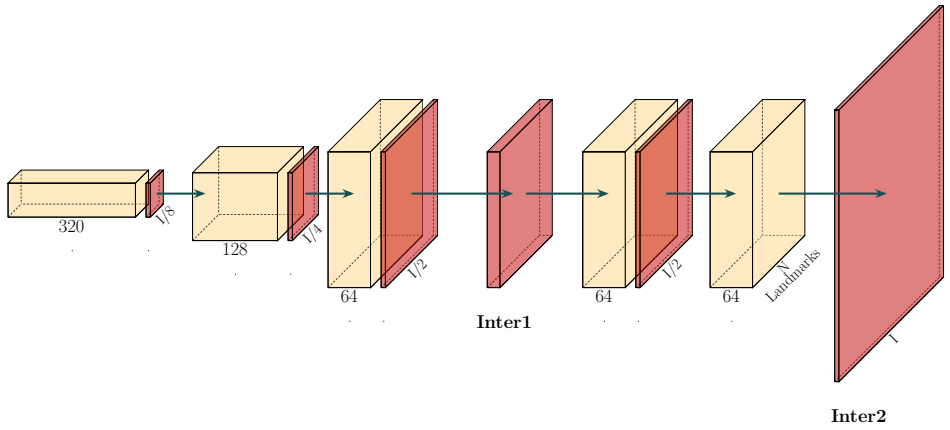


FIGURA 3.16: *Decoder* de Modelo Autoencoder Complex. Se duplicaron la cantidad de filtros en todas las capas convolucionales

Para el segundo experimento, al igual que el primero, se tomó la arquitectura base de Autoencoder Inception 3.4.2. A este modelo se le añadieron mas capas convolucionales en el *Decoder* continuando con la hipótesis de que balancear la cantidad de parámetros entrenables del modelo iba a lograr mejores resultados.

3.5.1. Menpo

Para el tercer experimento, se intentó utilizar un modelo de alineación de imágenes. Un modelo de apariencia activa (*Active Appearance Model* ó AAM para abreviar) es un modelo estadístico deformable tanto de la forma y como de la apariencia de un objeto. Durante el entrenamiento tiene como objetivo recuperar una descripción paramétrica de un determinado objeto a través de un algoritmo de optimización. Se creó un modelo de forma con los puntos cefalométricos de las imágenes de entrenamiento. La idea era que el *AAM* ayude a las predicciones del Autoencoder a tener mejor precisión ya que este modelo cuenta con una mayor información espacial entre los distintos puntos cefalométricos. En este experimento se tomó la arquitectura base del Autoencoder Inception 3.4.2, con las predicciones del Autoencoder, es decir las coordenadas de cada punto, se les aplicó un *AAM* entrenado sobre los puntos cefalométricos. Como resultado, el *AAM* genera unas nuevas predicciones, también en forma de coordenadas. Se utilizó la libreria de Menpo para poder ejecutar este experimento, la misma cuenta con un conjunto de herramientas que proporcionan soluciones integrales para el modelado deformable 2D y 3D.

3.5.2. Modelo de formas

Para el cuarto experimento se utilizaron las ideas de **Ibragimov** et al. [12]. En el mismo, se presenta un *framework* basado en teorías de juegos para la segmentación de imágenes basada en puntos de referencia. La detección de puntos de referencia (o *landmarks*) se formula como un juego, en el que los landmarks son jugadores, los puntos candidatos son estrategias y las probabilidades de que los puntos candidatos representen un Landmark son pesos (*payoffs*), determinados de acuerdo con la similitud de las intensidades de la imagen y las relaciones espaciales entre los puntos candidatos en la imagen objetivo y sus landmarks correspondientes en imágenes del conjunto de entrenamiento. Se intentó replicar tomando como puntos candidatos a las mayores n activaciones de un mapa de probabilidad y con eso resolver un problema de optimización basado en teoría de juegos como menciona Ibragimov. Sin

embargo, el costo computacional de resolver este problema hizo que fuera imposible realizar pruebas extensas por lo que eventualmente se terminó descartando esta idea.

3.5.3. Distintas funciones de activación

Se intentó utilizar distintas funciones de activación para el Autoencoder. Las funciones elegidas fueron: ***Leaky ReLU*** y ***PReLU***. La motivación para usar ***Leaky Relu*** es que no se enfrenta a el problema de que las activaciones sean nulas cuando la neurona tiene valores inferiores a 0, esto bloquea completamente el aprendizaje en ReLU debido a gradientes 0 en la parte negativa.

PReLU está inspirada en ***Leaky ReLU***, tiene un impacto insignificante en la precisión en comparación con ReLU. A diferencia de ***Leaky ReLU***, ***PReLU*** sustituye el valor 0.01 por un parámetro α.

Las mismas están definidas como:

$$LeakyReLU : f(x) = \begin{cases} 0{,}01x & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$$

$$PReLU : f(x) = \begin{cases} \alpha x & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases} \quad \alpha \in [\infty, 0)$$

3.6. Modelos propuestos

En la siguiente sección se explican las distintas mejoras realizadas sobre el modelo original. Como primer acercamiento, se modificó la forma en la que se inicializan los pesos de las capas convolucionales siguiendo la idea de Xavier [10]. La motivación para la inicialización de Xavier en las redes neuronales es inicializar los pesos de la red para que las funciones de activación no se inicien en regiones saturadas o muertas. En otras palabras, queremos inicializar los pesos con valores aleatorios que no sean "demasiado pequeños"ni "demasiado grandes". La inicialización de Xavier establece los pesos de una capa en utilizando valores elegidos por una distribución uniforme aleatoria U

$$\mathrm{U} \sim \left[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}} \right] \tag{3.2}$$

donde donde n_i es el número de conexiones entrantes de la capa y n_{i+1} es el número de conexiones salientes desde esa capa.

3.6.1. Modelo Wider

La segunda modificación consistió en aumentar el campo receptivo de las capas convolucionales. El campo receptivo se puede definir brevemente como la región en el espacio de entrada del cual puede llegar a obtener información una característica de una CNN en particular. Se modificó el tamaño de los *kernel* de las capas convolucionales, además de agregar una capa Inception en el en el Encoder. En la Figura 3.17 se pueden observar los cambios a la arquitectura. A continuación se lista en detalle las modificaciones que se le aplicaron a la arquitectura:

- Aumento de tamaño de *kernel* en la primera capa convolucional del Encoder. Se paso de 3x3 a 5x5 y se cambió el *padding* de 1 a 2.
- Adición de capa Inception E como segunda capa del *Encoder*
- Adición de capa convolucional y cambio de cantidad de filtros en las siguientes capas para que la disminución de dimensionalidad sea mas suave.

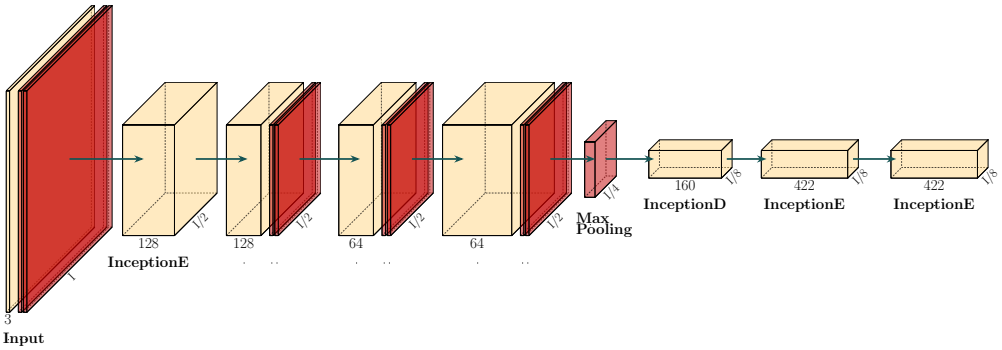


FIGURA 3.17: Arquitectura del Encoder en el modelo Wider. Se modificó el tamaño de los filtros, padding además de añadir una capa Inception más.

3.6.2. Modelo Wider Paddup

La tercera modificación que se le hizo al modelo consistió en cambiar la forma en la que el *Decoder* reconstruye la "imagen" (mapa de probabilidad). El problema más destacado asociado con la convolución transpuesta es la presencia de artefactos en forma de tablero de ajedrez en las imágenes de salida como se muestra en la Figura 3.18.

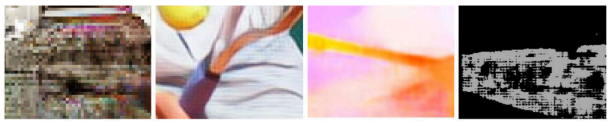


FIGURA 3.18: Distintos niveles de artefactos. Imagen extraída de [2].

Por lo tanto, se reemplazaron las capas Convolucionales Transpuestas por una combinación de *upsampling* con convoluciones básicas siguiendo la idea de [20]. Cada capa convolucional transpuesta fue reemplazada por lo siguiente: En primera instancia se aplica un *upscaling* utilizando la técnica de *nearest-neighbor* al mapa de probabilidades, seguido de un padding constante, finalmente se aplica una Convolución.

3.6.3. Modelos Gris

La cuarta modificación consistió en modificar la estructura para que reciba imágenes en escala de grises. Este cambio fue hecho debido a que el modelo estaba siendo entrenado con imágenes RGB que contenían información que no es relevante para el modelo. Por lo tanto, se añadió una transformación que convertía los datos a escala de grises y en cuanto a la arquitectura del modelo se modificó la cantidad de canales de entrada de 3 (RGB) a 1 (escala de grises). Este cambio se realizó en

el modelo original y además a los distintos experimentos que se mencionan en esta sección para poder observar si tenían cambios en su rendimiento. La transformación realiza un promedio en los 3 canales y logra resultados como los de la imagen 3.19.

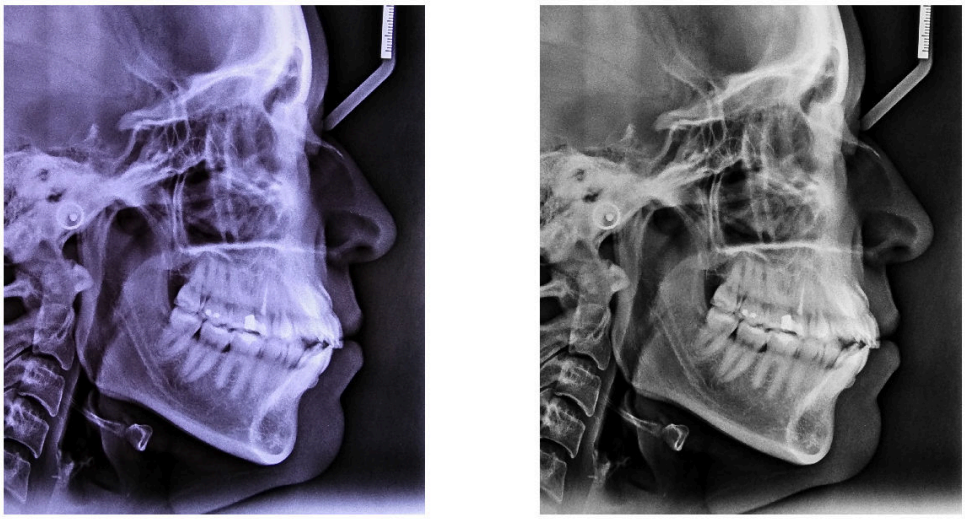


FIGURA 3.19: Transformación de imágenes a escala de grises. Izquierda: Imagen original. Derecha: Imagen luego de la transformación.

3.6.4. Modelos con información extra de puntos

El quinto experimento consistió en utilizar la información extra de las imágenes para poder mejorar el modelo. Como se mencionó en la sección 3.1, se contaba con información de la localización de los perfiles blandos y duros, con esta información se crearon nuevos mapas de probabilidad modelando las zonas donde se encontraban dichas estructuras. Con esta información, la tarea de identificar los puntos puede ser más fácil debido a que todos los puntos son marcados sobre algún perfil (blando o duro). En un primer comienzo se optó por agregar información de la zona de la mandíbula, Silla y del Basion, ya que los puntos ubicados en esas zonas son los mas difíciles de identificar, incluso para profesionales expertos. Para la generación de los mapas de probabilidad se utilizó un criterio similar al utilizado en los puntos cefalométricos. Previamente se necesitó realizar una interpolación entre los puntos para poder obtener un linea o trazado de la estructura en sí y sobre ese trazado ir aplicando funciones gaussianas para generar un mapa sobre la estructura. En la Figura 3.20 se puede observar a la izquierda los puntos originales para ambas estructuras y a la derecha el resultado del procedimiento mencionado. Se anexaron dos mapas de probabilidad (uno de cada estructura) a la entrada de la red, modificando ligeramente la arquitectura para permitir dos capas de entrada más. Este experimento se realizó tomando como base a la arquitectura del Autoencoder Wider.

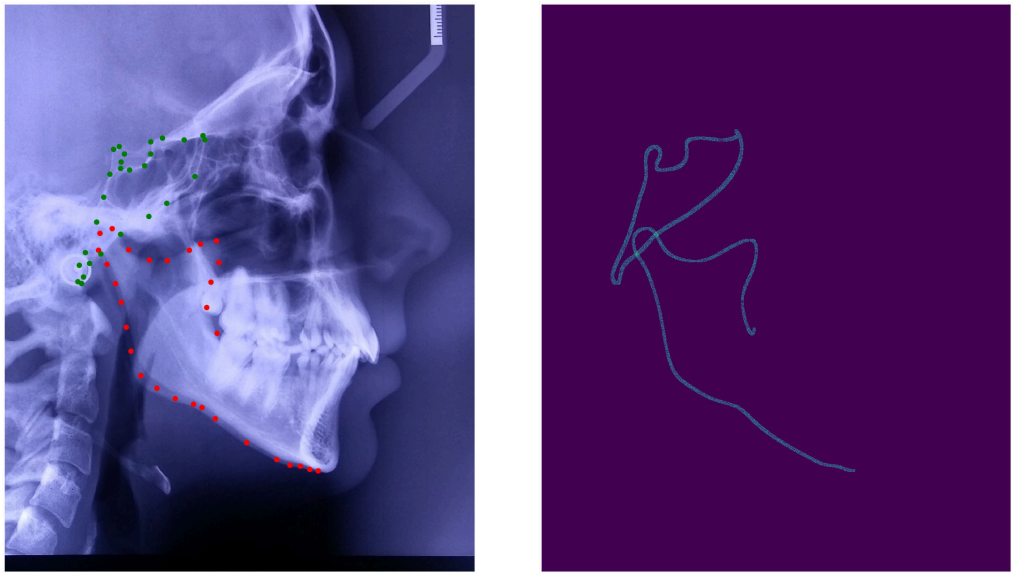


FIGURA 3.20: Mapa de probabilidad generado para la zona de la mandíbula, Silla y zona del Basion. Izquierda: Puntos originales. Derecha: Mapa de probabilidad generado.

3.6.5. Box para encontrar activaciones

El sexto experimento trató de resolver uno de los problemas que los modelos tenían, los mapas de probabilidad generados por el modelo poseían activaciones positivas en zonas donde un punto cefalométrico era imposible de que se encuentre. Para resolver esto, en cada conjunto de datos se utilizó la información de los datos de entrenamiento para calcular una zona donde cada punto esta localizado en promedio. A estas zonas se las calculó en relación a otros puntos (que la red detecta correctamente) para tener mayor confiabilidad a la hora de generar predicciones sobre una imagen nunca vista. El procedimiento consistió en medir todas las distancias medias (en el eje X e Y) entre los distintos puntos, con esta información para cada punto se arma un *box* ubicado con las distancias medias a los puntos Pogonio y Silla, y el tamaño del box esta dado por las distancias mas cortas y mas grandes vistas en el conjunto de entrenamiento (mas un cierto margen de error). En la Figura 3.21 se observa en verde al punto Silla y con Naranja los puntos Nasion y PiC. Con una cota de color rojo se denota la distancia en el eje Y entre el punto Silla y PiC y con una cota color azul se marca la distancia en el eje Y entre el punto Silla y el Nasion. Para los puntos Pogonio y Silla, si bien el modelo los identifica siempre bien, se propuso utilizar la misma idea para evitar falsas detecciones. Para ello, se calcularon las zonas donde se ubican en promedio (según sus coordenadas X e Y) con los datos de entrenamiento y se siguió el mismo procedimiento que con el resto de puntos cefalométricos. En la Figura 3.22 se puede observar una imagen de test en la cual se detectan dos zonas con grandes probabilidades para el punto Gonion, en azul la zona correcta y en rojo una falsa detección. Debido a esta técnica la falsa activación no es tomada en cuenta y la predicción es calculada sobre la zona adecuada.

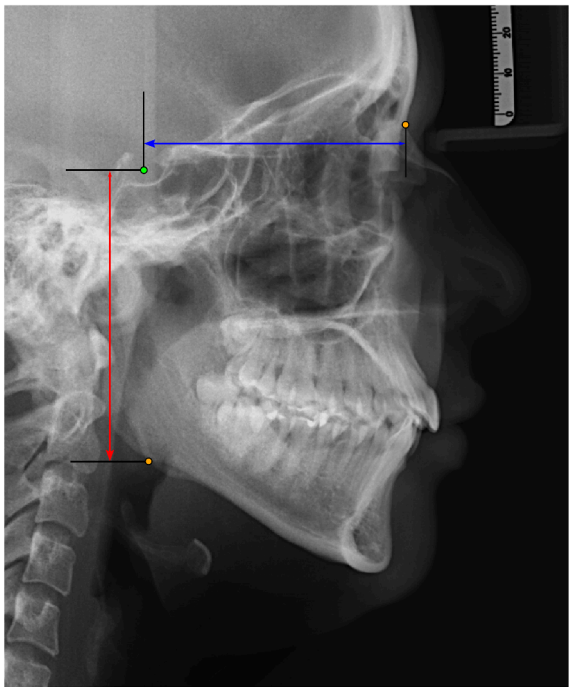


FIGURA 3.21: Calculo de box. En rojo distancia entre punto Silla y PiC eje X. En azul distancia entre punto Silla y Nasion.

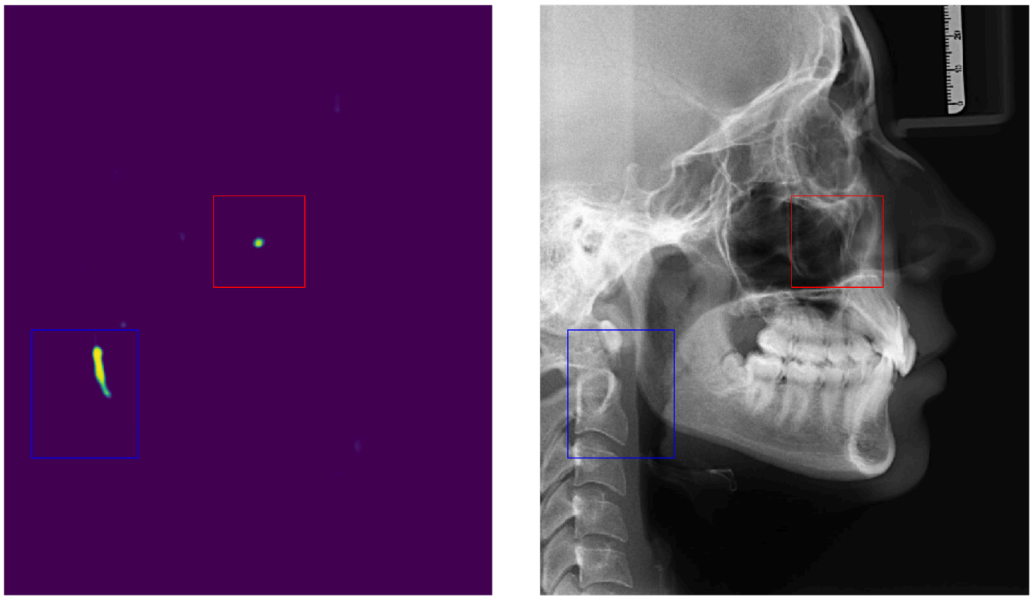


FIGURA 3.22: Izquierda: Mapa de activación del modelo convolucional. Derecha: Imagen original. En azul, se observa un rectángulo calculado a partir de los datos de entrenamiento. En rojo, una falsa detección fuera del rectángulo.

3.6.6. Esqueletizacion y Tangente

En el análisis de formas, el esqueleto de una forma es una versión delgada de esa forma que es equidistante de sus límites. El esqueleto generalmente enfatiza las propiedades geométricas y topológicas de la forma, como su conectividad, topología, longitud, dirección y ancho. La esqueletización es el resultado del proceso de adelgazamiento, que disminuye el contorno del objeto en cuestión hasta alcanzar una estructura de un píxel en promedio. En la Figura 3.23 se puede observar un ejemplo de un algoritmo de esqueletización en el cual partiendo de una imagen de una letra calcula un esqueleto que representa la forma de la misma. Los esqueletos se usan ampliamente en visión por computadora, análisis de imágenes, reconocimiento de patrones y procesamiento de imágenes digitales para propósitos tales como reconocimiento óptico de caracteres, reconocimiento de huellas digitales, inspección visual o compresión.

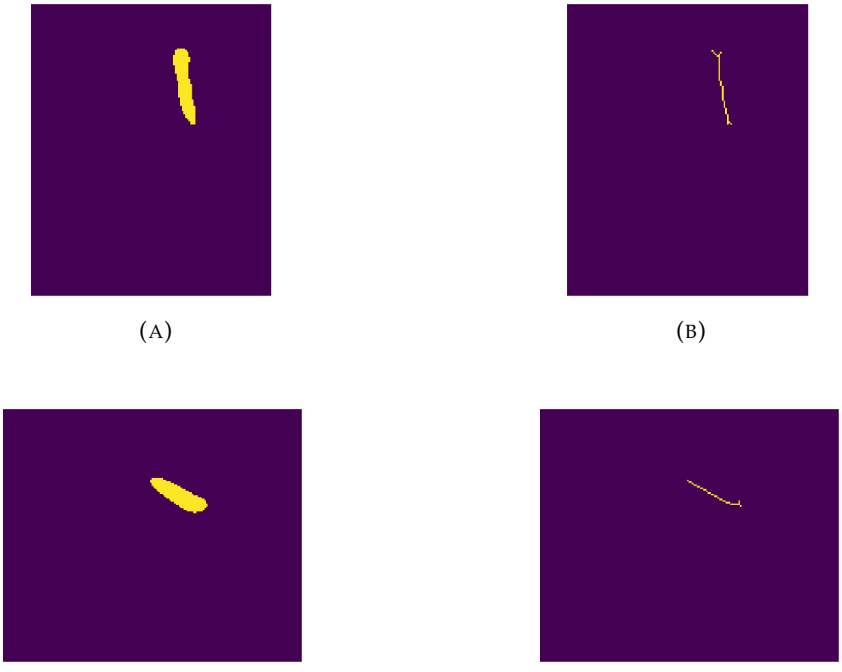
FIGURA 3.23: Ejemplo de algoritmo de esqueletización.

Utilizando este esqueleto se llevó a cabo un experimento para ayudar a localizar el punto Gonion (Go), el mismo se marca con la intersección de dos rectas, la primera armada con los puntos Articular y Póstero inferior de la rama mandibular (**PiR**), mientras que la segunda con el punto Mentoniano y con el Póstero inferior del cuerpo (**PiC**). Ver Figura 2.1, puntos 23 y 22 para la primera línea, 19 y 20 para la segunda. Tanto el PiR como el PiC son puntos difíciles de marcar, aún para personas expertas en el tema, logrando que estas líneas no sean bien identificadas y propaguen el error hacia el punto Gonion. Se realizó un experimento a partir de los mapas de probabilidad de los puntos PiR y PiC (Figuras 3.24a y 3.24c), se les aplicó un proceso de esqueletización (Figuras 3.24b y 3.24d). Una vez conseguidos los esqueletos de cada una de las zonas se procedió a detectar ambos puntos según las reglas determinadas por los estudios cefalométricos:

- PiR: El punto mas posterior de la rama. Es decir, para las imágenes laterales con las que se estuvo trabajando implicaría el punto mas a la izquierda de la estructura mandibular.
- PiC: El punto mas inferior del cuerpo. Es decir, de la mandíbula el punto mas inferior.

En la figura 3.25 se puede observar como se marcan las dos líneas en cuestión. En violeta y verde se pueden observar los esqueletos de los puntos PiR y PiC respectivamente, con esa información se trazaron las dos lineas. En amarillo, entre los

puntos Articular y PiR, en naranja entre el Mentoniano y el PiC. Además, se puede observar los *box* generados siguiendo el experimento de la sección 3.6.5.



(A) (B)

(C) (D)

FIGURA 3.24: Esqueleto

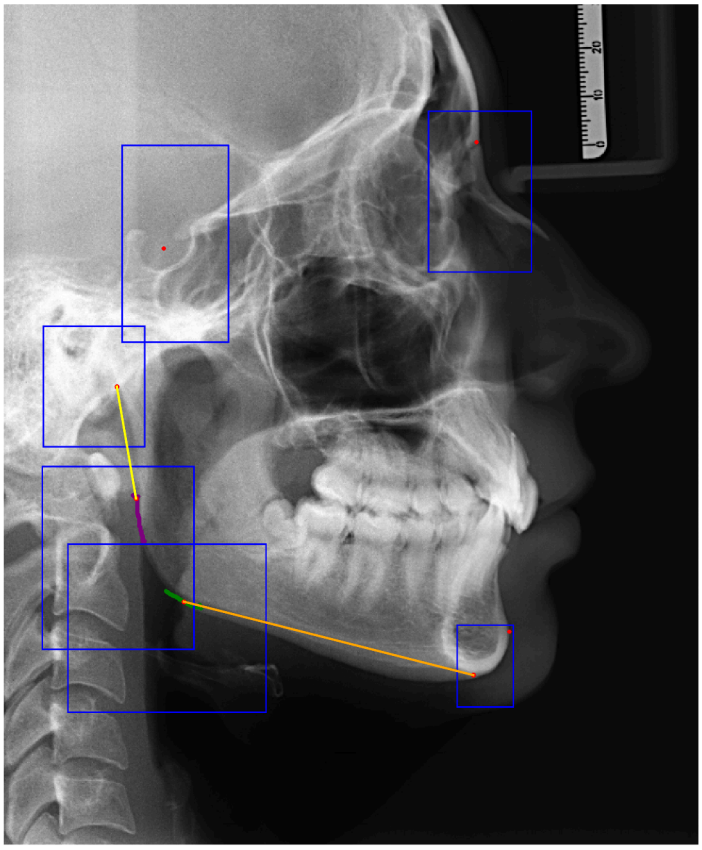


FIGURA 3.25: Ejemplo de imagen con boxes y tan

3.6.7. Modelo CordConv

Liu, Rosanne, et al. [16] propusieron añadir a las redes convolucionales información espacial de una manera novedosa. La solución propuesta se llama CoordConv y funciona dando acceso a la convolución a sus propias coordenadas de entrada mediante el uso de canales de coordenadas extras. La capa CoordConv propuesta es una extensión a la capa convolucional estándar en el cual los canales adicionales se instancian y se llenan con información de coordenadas (es decir, información constante que no se entrena), después se los concatena en la dimensión de los canales. La Figura 3.26 muestra como se añade la información para los canales i y j. Concretamente el canal de coordenadas i es una matriz $h \times w$ con su primera fila llena de 0, su segunda fila con 1, es el tercero con 2, etc. El canal de coordenadas j es similar, pero con sus columnas rellenas con valores constantes en lugar de sus filas.

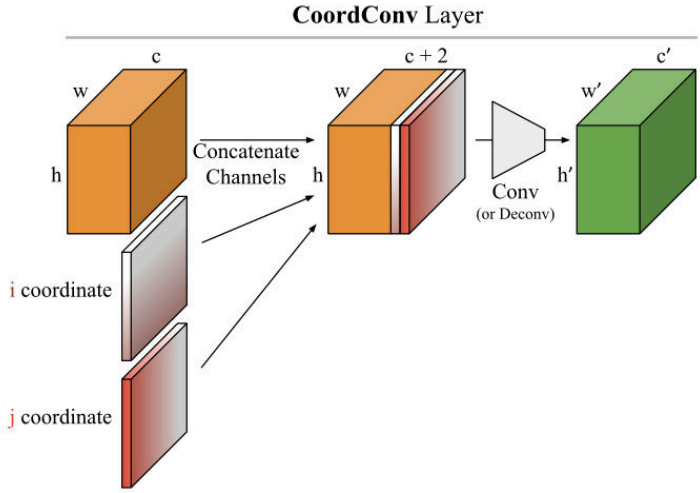


FIGURA 3.26: Proceso de creación de una capa CoordConv. Imagen extraída de [16].

En este experimento, se tomó como base la arquitectura del Autoencoder Wider PaddUp y se añadieron dos capas CoordConv. Una al principio del *Encoder* y otra en el *Decoder*.

Capítulo 4

Experimentos

A continuación se presentan los resultados obtenidos con las arquitecturas de redes convolucionales presentadas en el capítulos anteriores. Los modelos propuestos fueron: Autoencoder Xavier, Autoencoder Wider, Autoencoder Wider Paddup y Autoencoder Cord Conv. Sobre esos modelos se probaron distintas variaciones como: Gris (imágenes en escala de grises), Puntos (información de estructuras del cráneo), Box y Esqueletización. Es importante destacar que los siguientes resultados, al momento de escribir este trabajo especial de licenciatura, son los primeros en presentarse sobre Datasets privados y con casos de uso reales. Esto es importante porque permite ver como se desarrollan estos modelos en la vida real, si bien los resultados obtenidos de competencias pueden ofrecer métricas y resultados, nada garantiza que los mismos resultados ocurran en la práctica a la hora de desarrollar un sistema que funcione con alguna de las arquitecturas antes mencionadas.

En la primera parte de esta sección se presenta el problema de entrenamiento, mientras que en la segunda sección se aborda la etapa de evaluación de cada modelo.

4.1. Selección del Landmark desde mapas de probabilidad

Una vez generados los mapas de probabilidad, es necesario establecer alguna técnica para determinar únicamente la posición de cada punto cefalométrico. Se optó por una metodología simple, aplicar un filtro gaussiano al mapa de probabilidad (para filtrar posibles falsas detecciones que posean alta probabilidades en zonas particulares) y luego eligiendo el punto de mayor activación luego del filtro. El parámetro σ del filtro gaussiano fue elegido con las mismas restricciones impuestas por del problema, es decir, detectar con un error máximo de 2mm.

4.2. Hardware y software utilizados

Durante todo el proceso de desarrollo se utilizó un servidor provisto por el Centro de Cómputo de Alto Desempeño de la Universidad Nacional de Córdoba, al igual que para el entrenamiento y testeo de los distintos modelos propuestos. Las especificaciones del servidor son las siguientes:

Computadora Nabucodonosor

- Nodo Supermicro 1027GR-TSF con placa madre X9DRG-HF.
- 2 Xeon E5-2680v2 de 10 núcleos cada uno.
- 64 GiB RAM en 8 módulos de 8 GiB DDR3 1600 MT/s.

- 3 GPU NVIDIA GTX 1080Ti (GP102, 11 GiB GDDR5) conectados por PCIe 3.0 16x.
- 1 SSD 240GiB para Sistema Operativo conectados a SATA-2.
- 3 SSD 1TiB para datos en RAID0 por ZFS conectados a SATA-3.

Versiones de software utilizadas:

- **Python 3.7**
- **PyTorch 1.1.0**
- **OpenCV 4.1.0**

4.3. Entrenamiento

4.3.1. Transformaciones en el conjunto de datos

Antes de entrenar los modelos, existe un conjunto de transformaciones realizadas sobre el conjunto de datos, cada una con un objetivo específico. En todos los modelos de Autoencoders desarrollados además de la reducción de la imagen mencionada en 3.1, se realiza una serie de transformaciones con el fin de aumentar la cantidad de datos que el modelo observa en la fase de entrenamiento, las mismas son:

- Reescalados aleatorios de la imagen (entre el 98 % y 102 %)
- Traslaciones aleatorias horizontales y verticales a la imagen (de hasta 2 % de la imagen en cada dirección)
- Rotaciones aleatorias a la imagen (hasta 5°)
- Normalización del conjunto de imágenes para tener media 0 y varianza 1.

Los cambios de escala, rotación y traslación aleatoria de las imágenes, son realizados al comienzo de cada época de entrenamiento, con el motivo de aumentar artificialmente el tamaño del conjunto de entrenamiento y así lograr que el modelo aprecie mejor las estructuras existentes en las imágenes, y reducir el *overfitting*. Los valores elegidos son valores coherentes con variaciones naturales en la generación de la radiografía. Estos valores son pequeños, debido a que las radiografías cefalométricas deben ser realizadas con una serie de cuidados que no permiten demasiada variación. La normalización del dataset se realiza para facilitar el aprendizaje, ya que no hay una dimensión preferencial en la escala presentada.

4.3.2. Hiperparámetros

En esta sección determinaremos los valores de los hiperparámetros del modelo tales como la tasa de aprendizaje ϵ, el peso de regularización λ, el peso de la clase positiva p, el tamaño del batch y el número de épocas de aprendizaje. Se realizó una búsqueda en grilla similar a la de [21]. Los valores obtenidos fueron: ϵ de 10^{-3}, λ de 10^{-5} y p de 2500. El número de épocas de aprendizaje se estableció en 300 épocas y el tamaño de batch se modificó en base a cada experimento debido a restricciones de memoria de GPU.

4.3.3. Conjuntos de evaluación

Para el dataset del ISBI, existen dos conjuntos de evaluación, con 150 y 100 imágenes respectivamente. Con los datos provistos por la empresa CefMed se crearon 5 conjuntos de datos siguiendo las estrategias antes mencionadas, cada uno con su respectivo sub-conjunto para entrenamiento y test. Los conjuntos fueron armados de forma tal que cada uno de ellos fuera etiquetado por un profesional distinto, el resultado de esto fue que las imágenes de cada conjunto provengan de distintos centros médicos, lo que significó una diferencia de resoluciones, proporciones de aspecto, colores, etc. Los conjuntos 1, 2, 3 son de cefalometrías de Bjork, el conjunto 4 esta formado por la unión de Conjuntos 2 y 3; y el conjunto 5 son de cefalogramas de Ricketts.

4.4. Métricas

4.4.1. Coeficiente de detecciones exitosas

Para cada Landmark, los doctores marcan la localización de un solo píxel en vez de un área. Si la diferencia absoluta entre el punto detectado y el punto de referencia no es mas grande que z mm, la detección es considerada exitosa. De otra manera, la detección es considerada fallida. Entonces, el coeficiente de detecciones exitosas [26] (*SDR* para abreviar) p_z con precisión menor a z mm se formula como:

$$p_z = \frac{\#\{j : \|L_d(j) - L_r(j)\| < z\}}{\#\Omega} \times 100\,\%. \tag{4.1}$$

donde L_d, L_r representan la localización del Landmark detectado y del Landmark etiquetado, respectivamente, z denota la precisión en la medición, en nuestro caso $z = 2mm$ por las restricciones del problema, aunque se en algunas ocasiones se tomarán valores como: 2,5mm, 3mm y 4mm; $j \in \Omega$, y $\#\Omega$ representa el número de detecciones realizadas. Una aclaración importante es que para poder calcular la métrica correctamente se tiene que tener en cuenta las distintas resoluciones de cada imagen, es decir, la relación *pixel / mm* de cada una.

4.4.2. Error radial medio

El error radial [26] se define como $R = \sqrt{\Delta x^2 + \Delta y^2}$, donde $\Delta x, \Delta y$ son las distancias absolutas en las direcciones x e y respectivamente, entre el Landmark detectado y el etiquetado. El error radial medio o *mean radial error* (MRE) y la desviación estándar o *standard deviation* (SD) asociada, se calculan como:

$$MRE = \frac{\sum_{j=1}^{N} R_i}{N}, \qquad SD = \sqrt{\frac{\sum_{j=1}^{N} (R_i - MRE)^2}{N-1}} \tag{4.2}$$

Donde N es el número de imágenes en el conjunto de test.

4.5. Resultados

En esta sección presentaremos los resultados de los distintos experimentos presentados en la sección 3.6. Para cada experimento se eligió un conjunto de datos sobre cual entrenar los modelos. En los siguientes gráficos, el *Success detection rate*

está calculado en promedio para los 7 (ó 36) puntos cefalométricos. Los modelos fueron entrenados por 300 épocas y las métricas se calcularon cada 10 épocas.

4.5.1. Autoencoder básico

En el primer experimento se llevó a cabo para todos los conjuntos de datos disponibles con el fin de obtener un *baseline* con el cual comparar los distintos experimentos. En las Figuras 4.1, 4.3 y 4.4 por la parte superior se pueden observar los resultados para los conjuntos de datos 1, 3 y 4 respectivamente con la métrica *SDR*. Se puede observar que el modelo tiene un comportamiento similar para esos conjuntos de datos. Alrededor de la época 100 de entrenamiento los modelos sufren de *overfitting*, es decir, se sobre ajustan a los datos de entrenamiento logrando que no pueda generalizar la tarea de predicción para imágenes de validación. Los modelos alcanzan una *SDR* de **0.65** en promedio para los 7 puntos cefalométricos en sus respectivos conjuntos de *test*. Si examinamos las métricas para cada punto cefalométrico (Ver Tabla 4.1) se observa que el modelo puede predecir la ubicación de muchos puntos cefalométricos con una exactitud elevada, pero para los puntos 4 y 5 (ver Figura 2.2) tiene un coeficiente de detecciones exitosas muy bajo, logrando que la métrica promedio baje. Si miramos las Figuras 4.1, 4.2, 4.3, 4.4, 4.5 y 4.6 por la parte inferior de cada una se pueden observar los valores de la métrica *MRE*. Se observan resultados similares salvo para el conjunto 6 donde los valores iniciales son mayores que el resto.

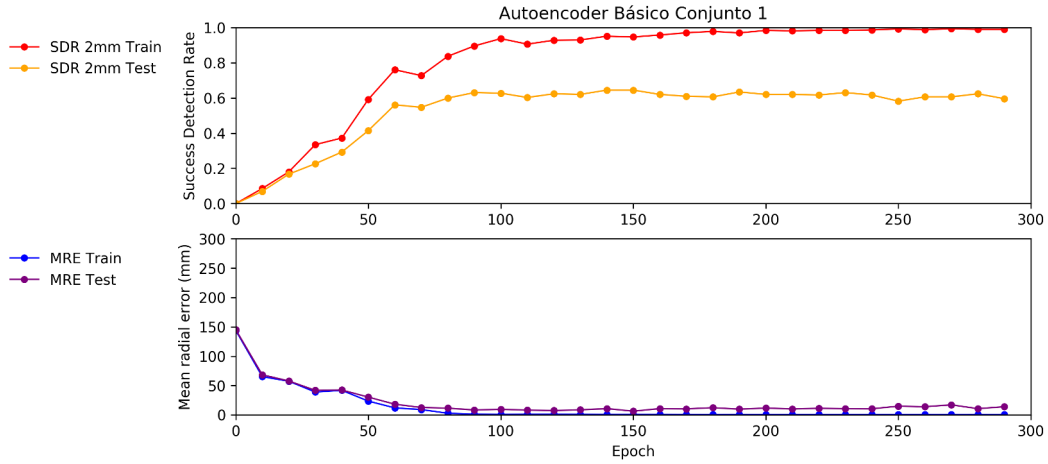


FIGURA 4.1: Coeficiente de detecciones exitosas (SDR) a 2 mm y error radial medio (MRE) en los conjuntos de entrenamiento y test 1 en función de las épocas de aprendizaje para modelo Autoencoder Básico

CUADRO 4.1: Coeficiente de Detecciones exitosas por Landmark. Modelo Autoencoder Básico entrenado sobre Conjunto 1

	Conjunto de Entrenamiento				Conjunto de Test			
	2 mm	2.5 mm	3 mm	4 mm	2 mm	2.5 mm	3 mm	4 mm
L1.00	0.96	0.98	1.00	1.00	0.85	0.90	0.93	0.93
L2.00	0.94	0.96	0.99	0.99	0.76	0.80	0.83	0.90
L3.00	0.98	0.99	1.00	1.00	0.76	0.76	0.80	0.90
L4.00	0.98	0.98	0.98	0.98	0.15	0.20	0.24	0.32
L5.00	0.70	0.79	0.85	0.91	0.29	0.44	0.46	0.51
L6.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
L7.00	0.95	0.97	0.98	0.98	0.66	0.71	0.80	0.83
Promedio	0.93	0.95	0.97	0.98	0.64	0.69	0.72	0.77

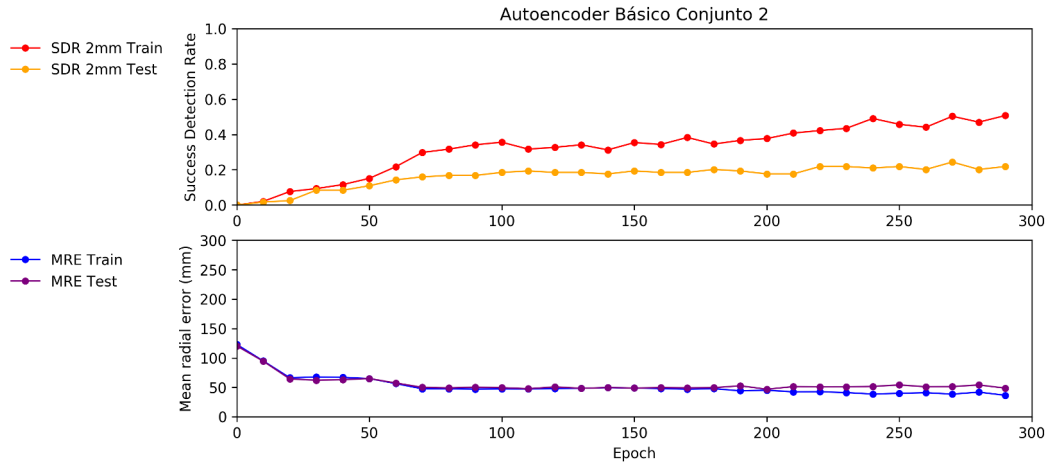


FIGURA 4.2: Coeficiente de detecciones exitosas (SDR) a 2 mm y error radial medio (MRE) en los conjuntos de entrenamiento y test 2 en función de las épocas de aprendizaje para modelo Autoencoder Básico.

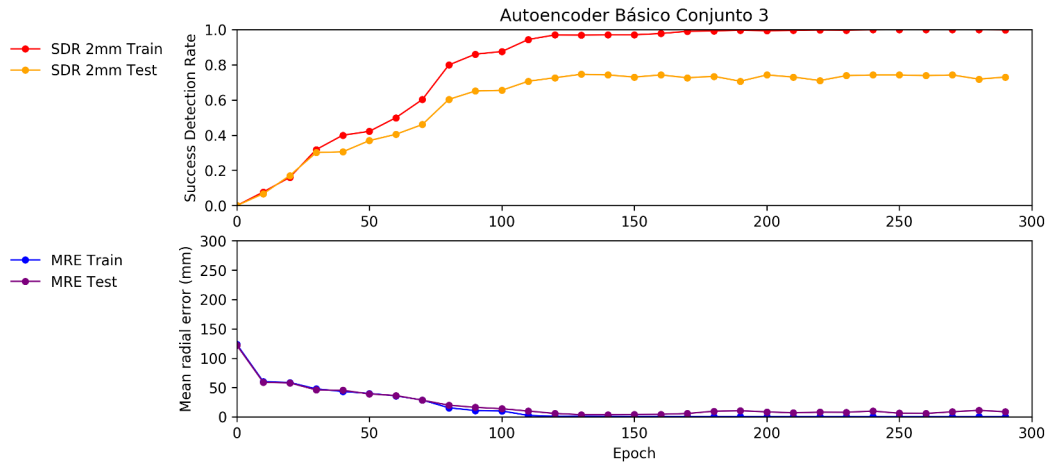


FIGURA 4.3: Coeficiente de detecciones exitosas (SDR) a 2 mm y error radial medio (MRE) en los conjuntos de entrenamiento y test 3 en función de las épocas de aprendizaje para modelo Autoencoder Básico.

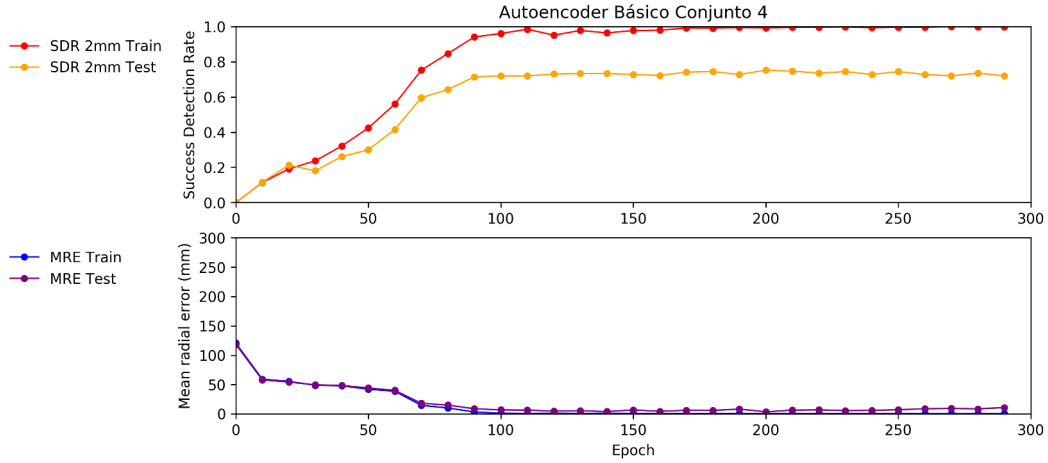


FIGURA 4.4: Coeficiente de detecciones exitosas (SDR) a 2 mm y error radial medio (MRE) en los conjuntos de entrenamiento y test 4 en función de las épocas de aprendizaje para modelo Autoencoder Básico

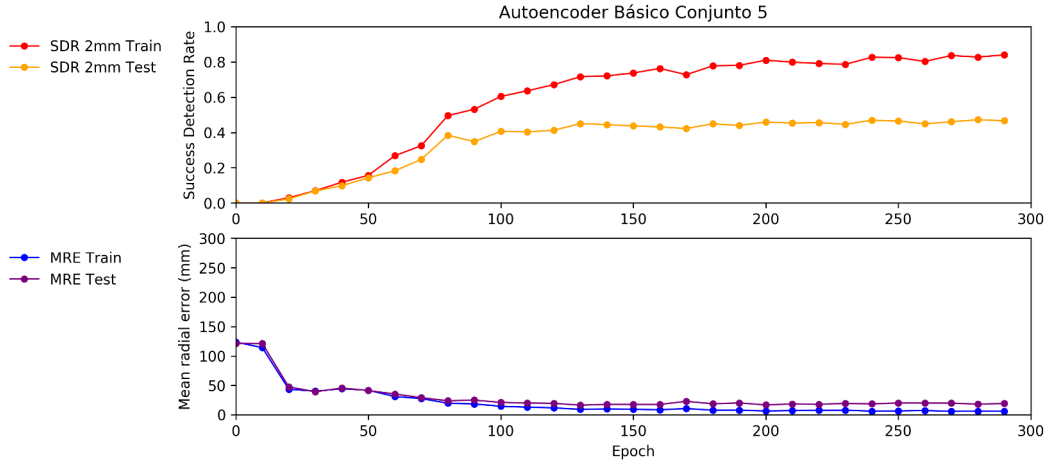


FIGURA 4.5: Coeficiente de detecciones exitosas (SDR) a 2 mm y error radial medio (MRE) en los conjuntos de entrenamiento y test 5 en función de las épocas de aprendizaje para modelo Autoencoder Básico.

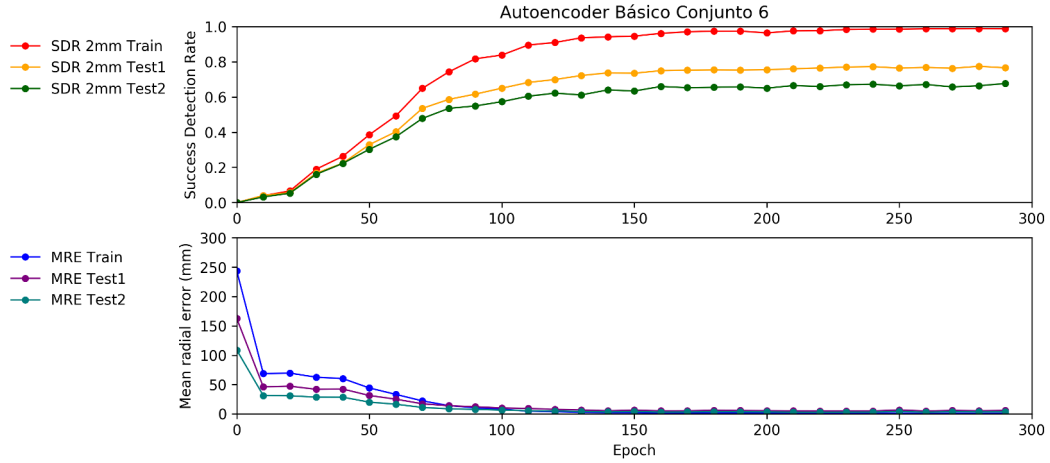


FIGURA 4.6: Coeficiente de detecciones exitosas (SDR) a 2 mm y error radial medio (MRE) en los conjuntos de entrenamiento y test 6 en función de las épocas de aprendizaje para modelo Autoencoder Básico.

Un resultado importante de mencionar son los resultados observados en los conjuntos de datos 2 y 3 (4.2 y 4.3). En el conjunto 2 de Test se logró obtener una *SDR* de **0.24** mientras que en el conjunto 3 de Test una de **0.75** representando estos los valores mínimos y máximos observados. En la Figura 4.7 se realizó una comparación entre los modelos en sus respectivos conjuntos de Test.

Además, se analizaron los resultados obtenidos sobre el conjunto de datos 6 (Ver Figura 4.6), es decir, sobre el dataset público del Challenge del ISBI. Los valores de las métricas son similares a los de los conjuntos de datos 1, 3 y 4. Algo importante de aclarar es que los valores de *SDR* comienzan con valores mas altos que el resto para luego normalizarse al igual que los demás.

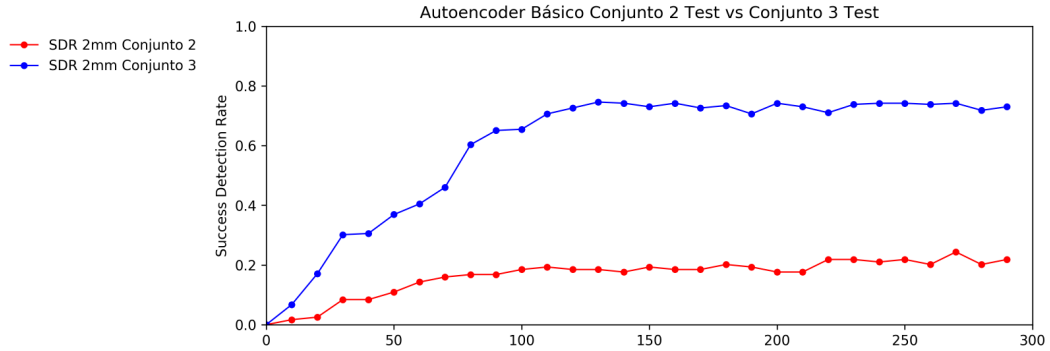


FIGURA 4.7: Coeficiente de detecciones exitosas (SDR) a 2 mm para el modelo Autoencoder Básico en los conjuntos de test 2 y 3.

4.5.2. Xavier

En la Figura 4.8 se puede observar la comparación de la *SDR* del modelo Autoencoder Básico contra el modelo que utiliza inicialización según Xavier para el conjunto 2 de Test. Los resultados obtenidos fueron similares y no se obtuvo una mejora. En la Figura 4.9 se realiza la misma comparación pero con el conjunto de datos 3 obteniendo la misma conclusión.

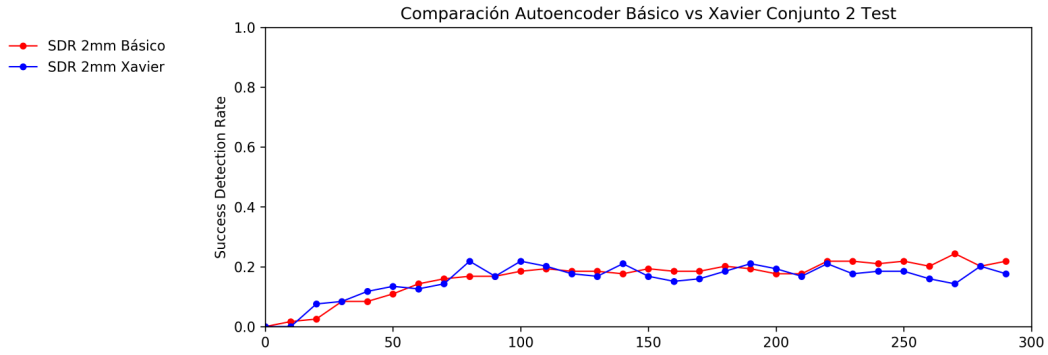


FIGURA 4.8: Comparación del coeficiente de detecciones exitosas (SDR) a 2 mm para los modelos Autoencoder Básico y Autoencoder Xavier en el conjunto de test 2.

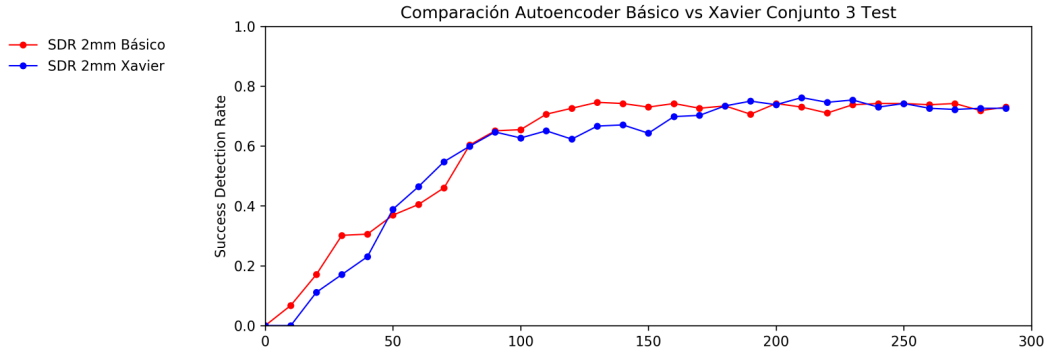


FIGURA 4.9: Comparación del coeficiente de detecciones exitosas (SDR) a 2 mm para los modelos Autoencoder Básico y Autoencoder Xavier en el conjunto de test 3.

4.5.3. Modelo Wider

En la Figura 4.10 se puede observar la comparación de la *SDR* del modelo Autoencoder Básico contra el modelo Autoencoder Wider, para el conjunto 1 de Test. El modelo básico logra una *SDR* máxima de **0.64** mientras que el modelo Wider **0.70**. Por otro lado, en la Figura 4.11 se observa la misma comparación pero con el conjunto de datos 3 obteniendo una *SDR* máxima de **0.75** para el modelo básico mientras que el modelo Wider una de **0.84**.

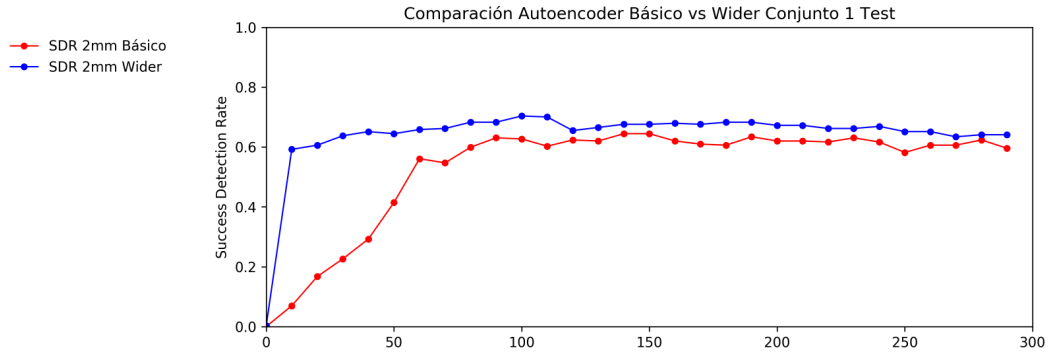


FIGURA 4.10: Comparación del coeficiente de detecciones exitosas (SDR) a 2 mm para los modelos Autoencoder Básico y Autoencoder Wider en el conjunto de test 1.

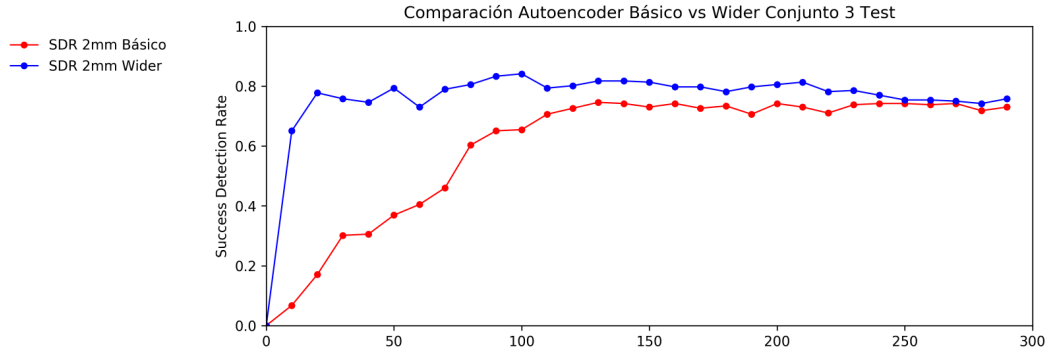


FIGURA 4.11: Comparación del coeficiente de detecciones exitosas (SDR) a 2 mm para los modelos Autoencoder Básico y Autoencoder Wider en el conjunto de test 3.

4.5.4. Modelo Wider Paddup

En las Figuras 4.12 y 4.13 se puede observar la comparación del modelo Wider con el modelo Wider Paddup para los conjuntos 1 y 3. No se observaron mejoras sustanciales con este experimento y los valores obtenidos tanto para la *SDR* y *MRE* fueron similares al del modelo Wider.

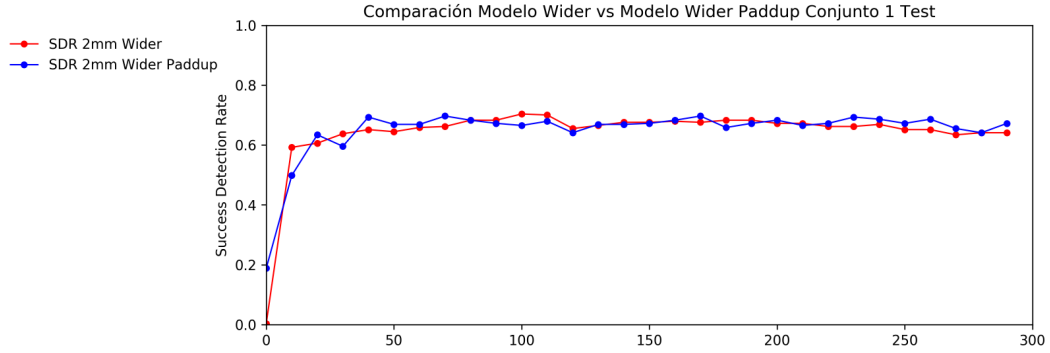


FIGURA 4.12: Comparación del coeficiente de detecciones exitosas (SDR) a 2 mm para los modelos Autoencoder Wider y Autoencoder Wider Paddup en el conjunto de test 1.

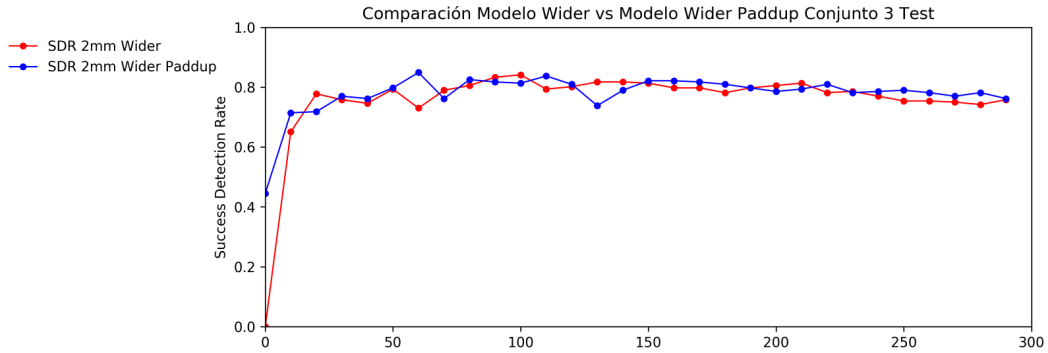


FIGURA 4.13: Comparación del coeficiente de detecciones exitosas (SDR) a 2 mm para los modelos Autoencoder Wider y Autoencoder Wider Paddup en el conjunto de test 3.

Este modelo se destacó en el conjunto 2 (Ver Figura 4.14) logrando los mejores resultados obtenidos sobre este conjunto de datos. Si comparamos con el Autoencoder Básico, donde teníamos una *SDR* de **0.24** en Test, tenemos que el Modelo Wider Paddup logra valores máximos de **0.76**.

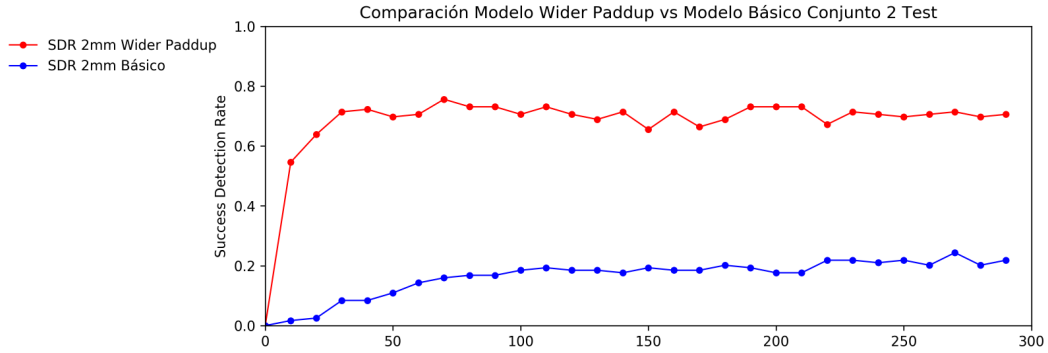


FIGURA 4.14: Comparación del coeficiente de detecciones exitosas (SDR) a 2 mm para los modelos Autoencoder Wider Paddup y Autoencoder Básico en el conjunto de test 2.

4.5.5. Modelo Wider Gray

El experimento de añadir imágenes en escala de grises se realizó sobre el modelo Wider. En las Figuras 4.15 y 4.16 se puede observar la comparación del modelo Wider con el modelo Wider pero utilizando imágenes en escala de grises para los conjuntos 1 y 3. No se observaron mejoras sustanciales con este experimento y los valores obtenidos tanto para la *SDR* y *MRE* fueron similares al del modelo Wider.

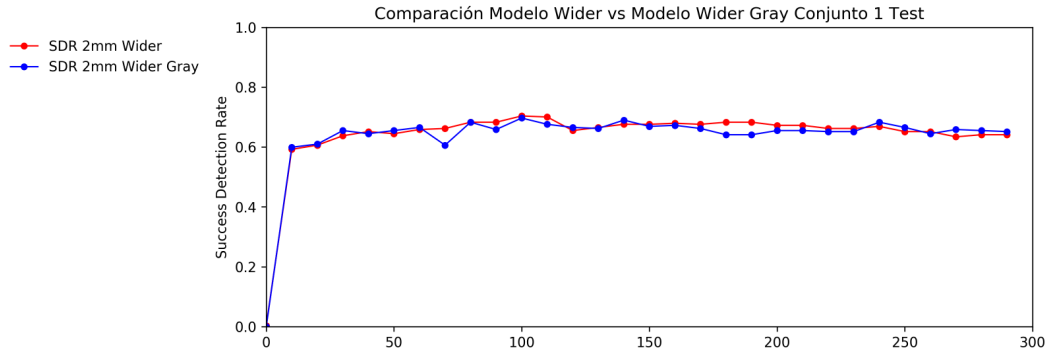


FIGURA 4.15: Comparación del coeficiente de detecciones exitosas (SDR) a 2 mm para los modelos Autoencoder Wider y Autoencoder Wider Gray en el conjunto de test 1.

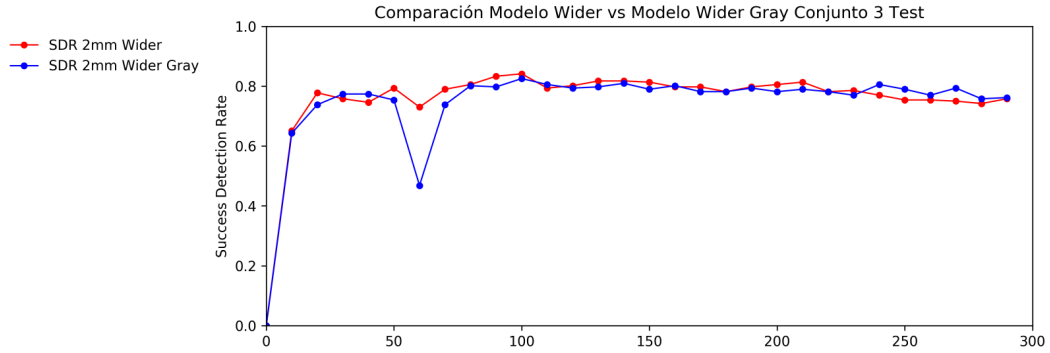


FIGURA 4.16: Comparación del coeficiente de detecciones exitosas (SDR) a 2 mm para los modelos Autoencoder Wider y Autoencoder Wider Gray en el conjunto de test 3.

4.5.6. Points

Se comparó los resultados obtenidos con el modelo Autoencoder Wider Points, es decir, el que utiliza información de las estructuras del cráneo, para el conjunto de entrenamiento 2 contra el mejor resultado obtenido hasta este punto para ese mismo conjunto, es decir, el modelo Wider Paddup (Ver Figura 4.17).

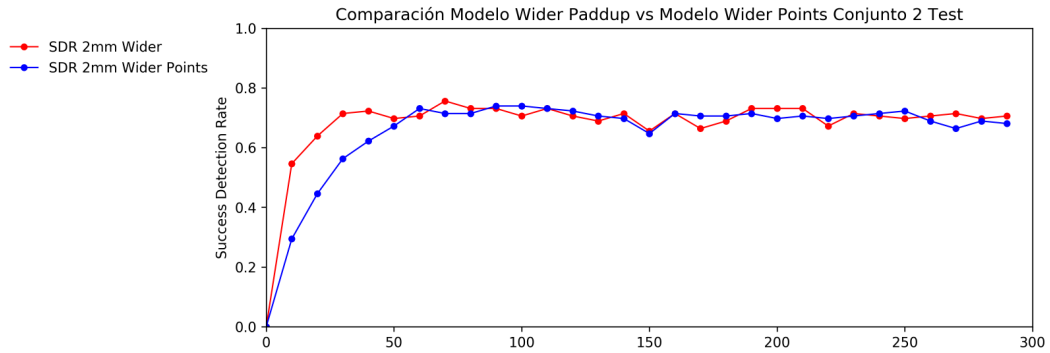


FIGURA 4.17: Comparación del coeficiente de detecciones exitosas (SDR) a 2 mm para los modelos Autoencoder Wider Paddup y Autoencoder Wider Points en el conjunto de test 2.

Además, se realizó un experimento comparando el modelo Autoencoder Wider Points con una versión modificada que utilizaba imágenes en escala de grises, el resultado se puede observar en la Figura 4.18, los valores obtenidos fueron similares, una *SDR* de **0.74** para el modelo Wider Points y una de **0.75** para el modelo Wider Points Gray.

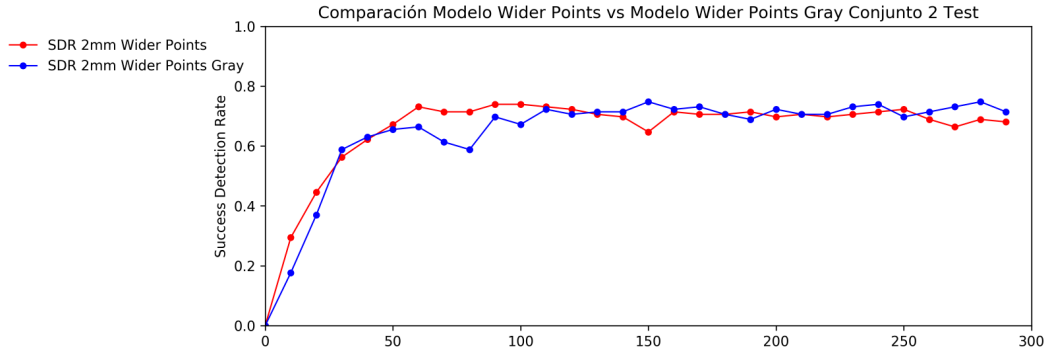


FIGURA 4.18: Comparación del coeficiente de detecciones exitosas (SDR) a 2 mm para los modelos Autoencoder Wider Points y Autoencoder Wider Points Gray en el conjunto de test 2.

4.5.7. Box

El experimento que consistió en añadir un *box* para localizar los puntos restringiendo la zona de búsqueda para la activación máxima se realizó sobre el modelo Autoencoder Wider Paddup. No hubo cambios significantes para la *SDR* a 2mm (Ver Figura 4.21 y 4.22), sin embargo logró que la cantidad de falsas predicciones en zonas donde no pertenecen los puntos, disminuya. Esto se puede observar para los valores de la métrica a 3mm y 4mm, que por las restricciones del problema no son relevantes.

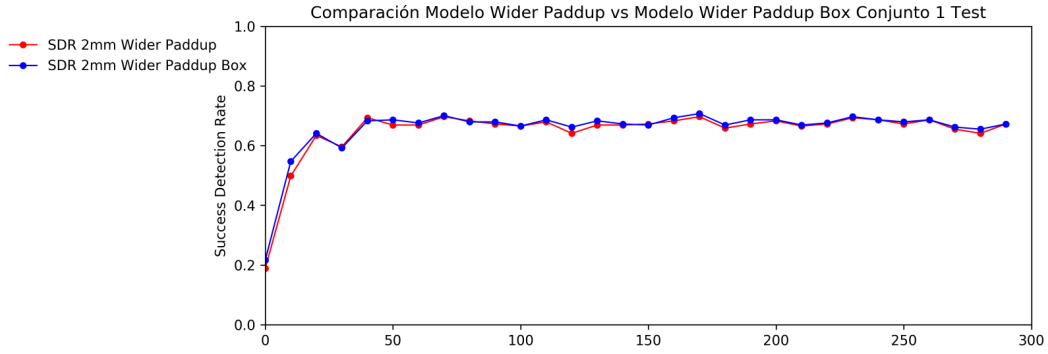


FIGURA 4.19: Comparación del coeficiente de detecciones exitosas (SDR) a 2 mm para los modelos Autoencoder Wider Paddup y Autoencoder Wider Paddup Box en el conjunto de test 1.

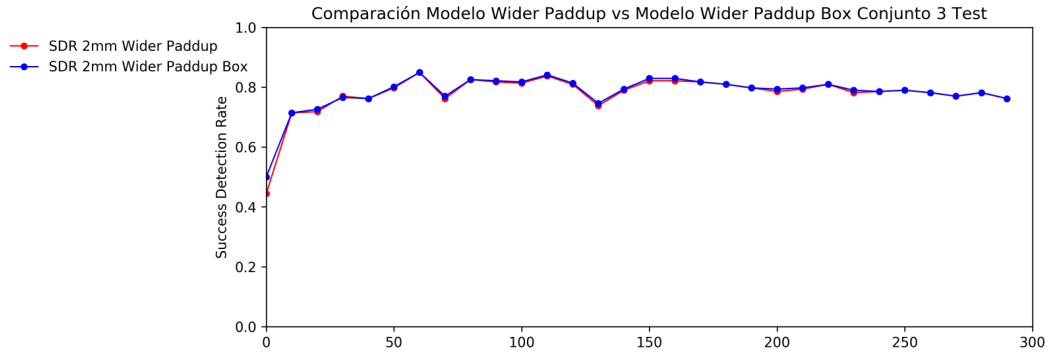


FIGURA 4.20: Comparación del coeficiente de detecciones exitosas (SDR) a 2 mm para los modelos Autoencoder Wider Paddup y Autoencoder Wider Paddup Box en el conjunto de test 3.

4.5.8. Box y Esqueletización

Al experimento de añadir un *box* se le sumó la idea de utilizar Esqueletización para detectar los puntos que tenían malas detecciones. El modelo mejoró la detección de estos puntos pero por un pequeño margen (Ver Tabla 4.2), esto se observa en el punto 4 que mejoró 10 puntos la *SDR*. En las Figuras 4.21 y 4.22 se puede observar la comparación del modelo Autoencoder Wider Paddup Box con el modelo Autoencoder Wider Paddup pero con Esqueletización demostrando que las mejores son imperceptibles.

CUADRO 4.2: Coeficiente de Detecciones exitosas por Landmark. Modelo Autoencoder Wider Paddup Box

	Conjunto de Test Box				Conjunto de Test Esqueletización			
	2 mm	2.5 mm	3 mm	4 mm	2 mm	2.5 mm	3 mm	4 mm
L1.00	0.85	0.93	0.95	1.00	0.90	0.90	0.98	1.00
L2.00	0.80	0.88	0.90	0.98	0.83	0.90	0.93	0.98
L3.00	0.90	0.93	0.93	0.98	0.90	0.93	0.93	0.98
L4.00	**0.27**	0.29	0.39	0.44	**0.37**	0.39	0.39	0.51
L5.00	0.27	0.29	0.39	0.54	0.27	0.34	0.46	0.54
L6.00	0.95	1.00	1.00	1.00	0.98	1.00	1.00	1.00
L7.00	0.85	0.88	0.88	0.98	0.71	0.76	0.78	0.88
Promedio	0.70	0.74	0.78	0.84	0.71	0.75	0.78	0.84

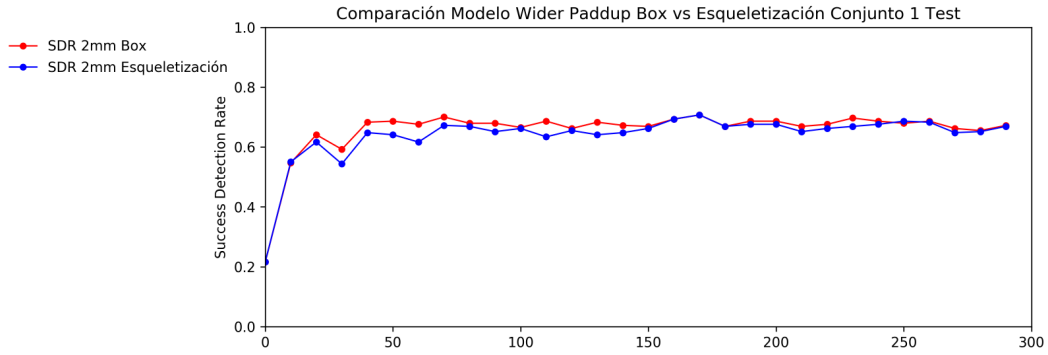


FIGURA 4.21: Comparación del coeficiente de detecciones exitosas (SDR) a 2 mm para los modelos Autoencoder Wider Paddup Box y Modelo Wider Paddup Box con Esqueletización en el conjunto de test 1.

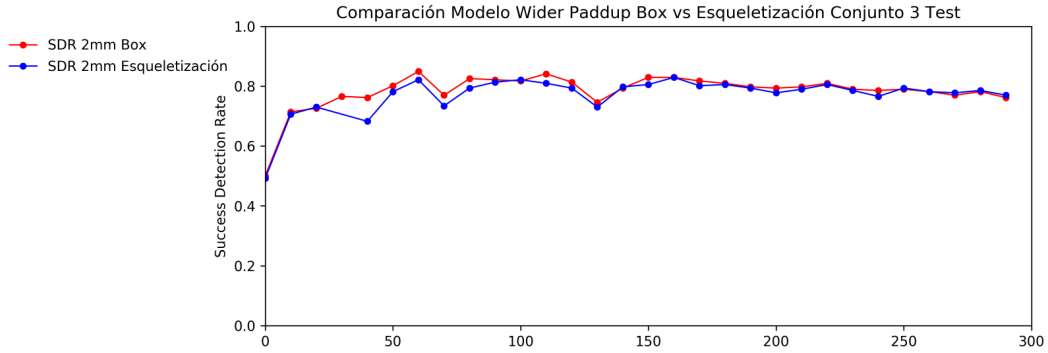


FIGURA 4.22: Comparación del coeficiente de detecciones exitosas (SDR) a 2 mm para los modelos Autoencoder Wider Paddup Box y Modelo Wider Paddup Box con Esqueletización en el conjunto de test 3.

4.5.9. Coord Conv

Para finalizar, se evaluó el desempeño del modelo Autoencoder Wider Paddup utilizando capas Coord Conv. Si lo comparamos con la versión sin estas capas convolucionales (Ver Figuras 4.23 y 4.24) podemos notar un comportamiento similar, concluyendo que no se obtuvieron mejoras.

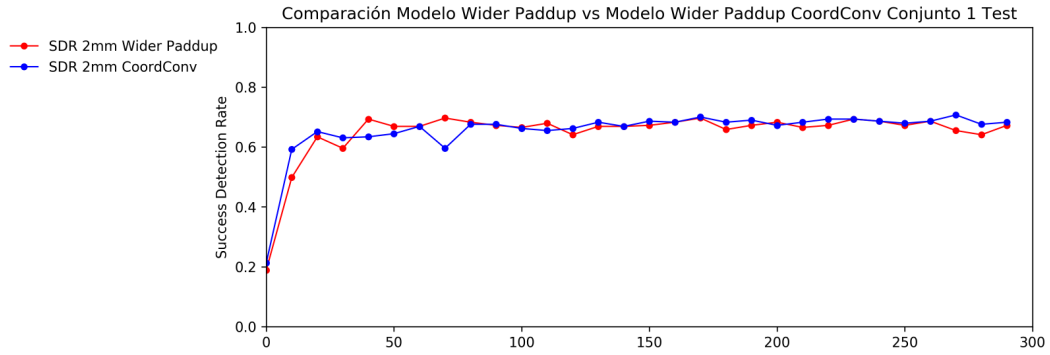


FIGURA 4.23: Comparación del coeficiente de detecciones exitosas (SDR) a 2 mm para los modelos Autoencoder Wider Paddup y Modelo Wider Paddup Coord Conv en el conjunto de test 1.

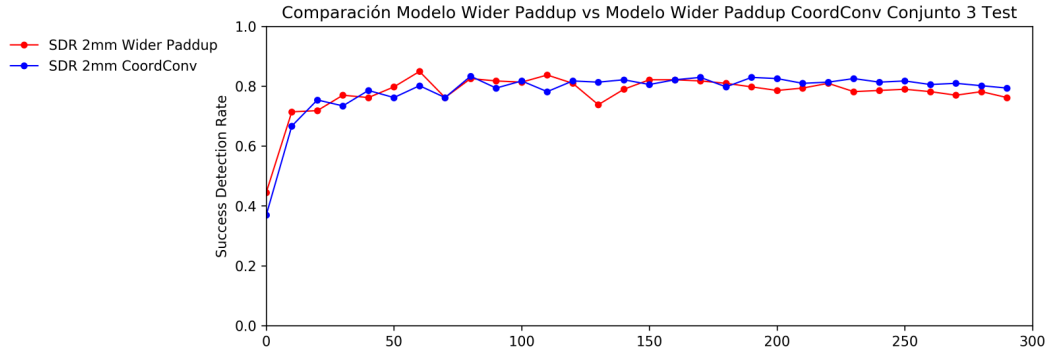


FIGURA 4.24: Comparación del coeficiente de detecciones exitosas (SDR) a 2 mm para los modelos Autoencoder Wider Paddup y Modelo Wider Paddup Coord Conv en el conjunto de test 3.

Capítulo 5

Conclusiones y trabajos a futuro

El analisis cefalométrico es una herramienta para los odontólogos que permite diagnosticar a sus pacientes, pudiendo detectar problemas dentarios problemas dentarios, esqueléticos ó problemas estéticos. La mayoría de profesionales recurre a técnicas manuales para realizar este estudio lo que lleva a la necesidad de contruir herramientas que permitan realizar estos estudios en un menor tiempo.

En este trabajo especial se utilizaron herramientas de Aprendizaje Automático para la generación de cefalometrías. Se presentaron arquitecturas de redes neuronales convolucionales novedosas que estan basadas en Autoencoders, utilizando capas *Inception*. Los distintos modelos presentados se utilizaron para la detección de puntos específicos en imágenes, en particular para la detección de puntos cefalométricos sobre imagenes de rayos x. Incluso con las arduas restricciones del problema, es decir detectar puntos cefalométricos a una distancia de 2mm, demostraron tener buenos resultados para las tareas mencionadas. Algunos aspectos claves a mencionar: La calidad de las imagenes influye de manera considerable en el comportamiento del modelo, imagenes de alta resolución y de equipos que son consistentes a la hora de realizar imagenes de rayos x logran que los resultados sean mejores. Los datasets construidos tienen que ser de imagenes etiquetadas por un solo profesional evitando que hayan, en un mismo dataset, imágenes marcadas por distintos profesionales. También, es posible etiquetar **todas** las imagenes del mismo dataset por distintos profesionales para poder tomar como punto verdadero los datos calculados con el promedio de las marcaciones de los distintos odontólogos. Se intentaron distintos experimentos para añadir información extra al modelo como la localización de las distintas estructuras, esto logró estabilizar el modelo logrando que los mapas de probabilidad generados por los modelos tengan activaciones mas localizadas disminuyendo la cantidad de falsas activaciones.

Este trabajo especial posee muchos temas para seguir explorando en el futuro. Por un lado, la cantidad de imagenes etiquetadas es un problema que afecta la *performance* del modelo, por lo tanto, conseguir un dataset con mas imagenes y etiquetadas por mas profesionales puede disminuir el error intra-observador. Además, se puede explorar la detección de áreas o puntos característicos en distintas áreas de la medicina, por ejemplo, en la detección de tumores en imagenes de rayos x, un área que está cobrando mucho interés en los últimos tiempos. Otra gran incorporación para estos modelos es la información de la ubicación de las distintas estructuras en las imagenes. Investigar la detección de estructuras en imágenes de rayos x puede tanto mejorar las predicciones, como ayudar a automatizar todo el proceso de generación de cefalométrias o cualquier estudio médico que necesite identificar estructuras para su marcado.

Bibliografía

[1] F. J. Aguila. *Manual de Cefalometría*. Editorial Aguiram, 1996.

[2] A. Aitken, C. Ledig, L. Theis, J. Caballero, Z. Wang, and W. Shi. Checkerboard artifact free sub-pixel convolution: A note on sub-pixel convolution, resize convolution and convolution resize. *arXiv preprint arXiv:1707.02937*, 2017.

[3] S. Ö. Arik, B. Ibragimov, and L. Xing. Fully automated quantitative cephalometry using convolutional neural networks. *Journal of Medical Imaging*, 4(1):014501, 2017.

[4] P. N. Belhumeur, D. W. Jacobs, D. J. Kriegman, and N. Kumar. Localizing parts of faces using a consensus of exemplars. *IEEE transactions on pattern analysis and machine intelligence*, 35(12):2930–2940, 2013.

[5] C. Cernazanu-Glavan and S. Holban. Segmentation of bone structure in x-ray images using convolutional neural network. *Adv. Electr. Comput. Eng*, 13(1):87–94, 2013.

[6] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in neural information processing systems*, pages 2843–2851, 2012.

[7] A. de Brebisson and G. Montana. Deep neural networks for anatomical brain segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 20–28, 2015.

[8] V. Dumoulin and F. Visin. A guide to convolution arithmetic for deep learning. *ArXiv e-prints*, mar 2016.

[9] A. P. R. Durão, A. Morosolli, P. Pittayapat, N. Bolstad, A. P. Ferreira, and R. Jacobs. Cephalometric landmark variability among orthodontists and dentomaxillofacial radiologists: a comparative study. *Imaging science in dentistry*, 45(4):213–220, 2015.

[10] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[11] B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue, et al. An empirical evaluation of deep learning on highway driving. *arXiv preprint arXiv:1504.01716*, 2015.

[12] B. Ibragimov, B. Likar, F. Pernus, et al. A game-theoretic framework for landmark-based image segmentation. *IEEE Transactions on Medical Imaging*, 31(9):1761–1776, 2012.

[13] B. Ibragimov, B. Likar, F. Pernus, and T. Vrtovec. Automatic cephalometric x-ray landmark detection by applying game theory and random forests. In *Proc. ISBI Int. Symp. on Biomedical Imaging*, 2014.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[15] C. Lindner, C.-W. Wang, C.-T. Huang, C.-H. Li, S.-W. Chang, and T. F. Cootes. Fully automatic system for accurate localisation and analysis of cephalometric landmarks in lateral cephalograms. *Scientific reports*, 6:33581, 2016.

[16] R. Liu, J. Lehman, P. Molino, F. P. Such, E. Frank, A. Sergeev, and J. Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *Advances in Neural Information Processing Systems*, pages 9605–9616, 2018.

[17] J. L. Long, N. Zhang, and T. Darrell. Do convnets learn correspondence? In *Advances in Neural Information Processing Systems*, pages 1601–1609, 2014.

[18] F. Milletari, N. Navab, and S.-A. Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 565–571. IEEE, 2016.

[19] C. Nebauer. Evaluation of convolutional neural networks for visual recognition. *IEEE Transactions on Neural Networks*, 9(4):685–696, 1998.

[20] A. Odena, V. Dumoulin, and C. Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.

[21] J. I. Porta. *Uso de redes neuronales en el procesamiento de imágenes odontológicas*. Trabajo especial de licenciatura en física, Facultad de Matemática, Física, Astronomía y Computación - UNC, 2019.

[22] D. Ramanan and X. Zhu. Face detection, pose estimation, and landmark localization in the wild. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2879–2886. Citeseer, 2012.

[23] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[24] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[25] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[26] C. Wang, C. Huang, C. Li, and S. Chang. A grand challenge for automated detection of critical landmarks for cephalometric x-ray image analysis. In *IEEE International Symposium on Biomedical Imaging*, 2014.

Los abajo firmantes, miembros del Tribunal de evaluación de tesis, damos fe que el presente ejemplar impreso se corresponde con el aprobado por este Tribunal.