UNIVERSIDAD NACIONAL DE CÓRDOBA

TRABAJO ESPECIAL DE LICENCIATURA

Uso de redes neuronales en el procesamiento de imágenes odontológicas

---

Autor:
Juan PORTA

Directores:
Dr. Francisco A. TAMARIT
Dr. Jorge SANCHEZ

Tesis presentada en el cumplimiento de los requisitos para el grado de Licenciado en Física

en el

Grupo de Teoría de la Materia Condensada
28 de marzo de 2019

UNIVERSIDAD NACIONAL DE CÓRDOBA

Resumen

Facultad de Matemática, Astronomía, Física y Computación

Licenciado en Física

Uso de redes neuronales en el procesamiento de imágenes odontológicas

por Juan PORTA

En este trabajo se aborda el problema de la detección de puntos característicos en imágenes médicas bidimensionales obtenidas mediante técnicas de rayos X, para lo cual se hizo uso de aprendizaje automático basado en redes neuronales artificiales. En particular, se abordó el problema de identificación de puntos característicos en imágenes cefalométricas de uso odontológico, no solo por su alto valor profesional, sino sobre todo porque este problema representa un gran desafío para el uso de inteligencia artificial en el reemplazo de expertos humanos. Se introdujo una arquitectura novedosa, no utilizada anteriormente en la literatura, la cual combina el uso de redes neuronales convolucionadas profundas con núcleos convolucionales de diferentes tamaños (capas inception) y la arquitectura de autoencoder para asociar a una imagen de entrada un mapa de probabilidades sobre la misma. Se utilizaron diferentes redes y se realizó un estudio comparativo, llegando a mostrar que estas redes introducidas tienen un excelente desempeño a la hora de identificar puntos característicos, superando las que se conocen en la bibliografía. En otras palabras, los modelos presentados en este trabajo mostraron una gran presición en detección de posiciones de landmarks, superando hasta en más de 7 puntos porcentuales los desempeños de los mejores modelos presentados en la literatura.

UNIVERSIDAD NACIONAL DE CÓRDOBA

Resumen en inglés

Facultad de Matemática, Astronomía, Física y Computación

Licenciado en Física

Uso de redes neuronales en el procesamiento de imágenes odontológicas

por Juan PORTA

In this work the problem of the detection of characteristic points in two-dimensional medical images obtained by X-ray techniques was addressed, for which automatic learning based on artificial neural networks is used. In particular, the problem of identifying characteristic points in odontologic cephalometric images was addressed, not only because of its high professional value, but above all because this problem represents a great challenge for the use of artificial intelligence in the replacement of human experts. A novel architecture, not previously used in the literature was introduced, which combines the use of deep convolutional neural networks with convolutional kernels of different sizes (inception layers) and the autoencoder architecture to associate an input image with a map of probabilities over it. Different networks were used and a comparative study was carried out, showing that these introduced networks have an excellent performance when identifying characteristic points, surpassing those that are known in the bibliography. In other words, the models presented in this work showed great precision in detecting landmarks positions, surpassing the performance of the best models presented in the literature by more than 7 percentage points.

Agradecimientos

- A mis padres, Beto y Sandra, los que me bancaron toda la vida mis locuras, entre ellas la de estudiar en FaMAF. Los amo con todas mis fuerzas.
- A mi hermana Agus, de toda la vida, y a Juan, desde que están juntos, ejemplos de vida y lucha y amor.
- A Pancho, un excelente director, persona y amigo, y el responsable de que esté haciendo esta tesis.
- A Jorge, resolviendo todas las dudas que surgieron en este proceso.
- A Seba, que permitió que me involucre en este proyecto y confió en mí para desarrollarlo.
- A Vane, compañera de la vida en las buenas y en las malas, pase lo que pase, la persona que me hizo madurar más que nadie y a quien le debo gran parte de haber llegado hasta acá.
- A Fran, ya que me enseñó y me enseña a trabajar en equipo en programación, me formó en bash y desarrolló conmigo las evaluaciones y métricas.
- A Marcos, Mauri y Santi, mis mayores compañeros de estudio, con quienes compartí sufrimiento, mates e insultos en el transcurso de nuestra formación, los cuales nos hermanaron profundamente.
- A la gente del Guri y al Guri como entidad, compañeros de militancia, de vida, amigos de fierro de los que aprendí cosas más importantes y profundas que la Física.
- A Mariano, el barba, que seguro está levantando su copa en mi honor.
- A Cindi, Martín y Agus, que comparten todos los días el trabajo, los mates, los cafés, los almuerzos y mi nerviosismo.
- A los del cole, amigos incondicionales y cables al mundo exterior.
- Al resto de mi familia, que capaz no entienden lo que hago pero me bancan igual.
- A María José e Ignacio, juro que voy a presentar todo a tiempo.
- A Carlos, gracias por las consultas estos últimos días.
- A todos los que me prestaron sus apuntes para estudiar.
- Al pueblo argentino, y en particular a las clases populares, a quienes más le debemos como universitarios.

- A Cristina y Nestor, y a la militancia, por intentar construir la patria que soñamos.
- A la ineludible verdad de que vamos a volver.

Índice general

Dedicado a todos los que me acompañaron en este camino.

Capítulo 1

Introducción

En el desarrollo de los estudios de redes neuronales han tenido especial importancia las técnicas analíticas y numéricas desarrolladas para el estudio de la física de los sistemas desordenados y complejos. En particular, el estudio mecánico estadístico de los vidrios magnéticos, caracterizados por interacciones aleatorias competitivas, ha sido clave para la construcción de los primeros modelos neuronales capaces de imitar los mecanismos de memoria y aprendizaje observados en sistemas naturales. Teniendo en cuenta que una neurona artificial es usualmente un sistema dinámico simple, una red neuronal debe entenderse como un conglomerado de dichas neuronas unidas a través de una arquitectura de conexiones sinápticas, las cuales pueden ser tanto excitatorias como inhibitorias, dando lugar así a la presencia de frustración. En particular, el modelo de Ising con interacciones aleatorias ferro o antiferromagnéticas fue la piedra angular en el desarrollo de redes de memoria asociativa, como el modelo pionero de Hopfield (1982).

En el campo particular de análisis de imágenes, han tomado una importancia notable en los últimos años las redes neuronales artificiales y el deep learning, en específico las Redes Neuronales Convolucionales (CNN, de su versión en inglés).

El problema específico de detección de puntos característicos (definidos por expertos) en imágenes, es un problema con un enorme potencial de aplicación y que conlleva desafíos importantes en términos de aprendizaje automático. Es un problema cuyas soluciones presentan aplicaciones en muchos campos, tanto científicos como técnicos y tecnológicos. Encontrar automáticamente, a partir de imágenes bidimiensionales centros de masa, articulaciones, fallas mecánicas, defectos en materiales y deformaciones son algunos ejemplos de aplicación.

Un ejemplo particular de aplicación es la cefalometría cuantitativa. Se trata de una técnica médica esencial en la elaboración de diagnósticos, evaluaciones de control y postratamiento en la odontología. Consiste en la demarcación en imágenes radiológicas (generalmente de rayos x) de puntos característicos para la posterior determinación de la forma mandibular y craneal, y desviaciones en éstas, a través de las posiciones absolutas y relativas de estos puntos. El procedimiento realizado por los odontólogos, utilizando lápiz y papel vegetal, es costoso en tiempo y complejo de aprender, con lo cual el desarrollo de herramientas de automatización del proceso sería de gran beneficio para éstos. Si bien se trata de un problema especifico,

su resolución es de fácil generalización a otras aplicaciones. No hay hasta ahora una solución adecuadamente precisa para este problema en la literatura publicada, por lo cual se ha convertido en un campo de mucha investigación científica en los últimos años. En particular, se dispone de bases de datos públicas y etiquetadas (o sea, ya analizadas previamente por expertos humanos), con lo cual se hace fácil contrastar el desempeño de los diferentes métodos entre sí.

Una herramienta extremadamente útil en este campo, además de las publicaciones científicas, son los desafíos (competencias) organizados por diferentes instituciones científicas internacionales. En los años 2014 y 2015 se introdujeron en particular competencias específicas de detección automática en imágenes cefalométricas de rayos x con el respaldo del *IEEE International Symposium on Biomedical Imaging*. En estas competencias se utilizaron en su mayoría modelos de detección de landmarks basados en el método llamado Random Forest para segmentación y posterior asignación de los puntos (Wang y col. (2015) y Wang y col. (2016)). En ambos se concluyó que la exactitud de los modelos no era aún suficiente y que la detección de puntos característicos (a los cuales llamaremos a lo largo de la tesina *landmarks*) sigue siendo un problema abierto. En 2017, Arik, Ibragimov y Xing (2017) propusieron un modelo de detección de landmarks aplicando primero una red neuronal convolucional para la generación de un mapa de verosimilitud de los landmarks, basado en la intensidad de los píxeles, y combinándolo luego con un modelo de Random Forest para la generación de un mapa basado en la distribución de cada landmark respecto al resto. El mismo logró superar a los modelos de ambas competencias, para una precisión de 2mm, que en odontología se considera el máximo error aceptable.

El enfoque presentado es el estudio teórico y la posterior implementación un sistema detección automática a través de redes neuronales de puntos característicos en imágenes bidimensionales cefalométricas. Continuando con la idea de Arik, Ibragimov y Xing (2017), se propone una exploración en distintas arquitecturas originales de redes neuronales para desarrollar un modelo que mejore el estado del arte en el problema de detección de landmarks en imágenes cefalométricas. Para esto se hace uso de un conjunto de datos (dataset) público de las competencias "Automatic Cephalometric X-Ray Landmark Detection Challenge 2014" y "Grand Challenges in Dental X-Ray Image Analisys: Automated Detection and Analisys for Diagnosis in Cephalometric X-Ray Image", correspondientes al ISBI 2014 y 2015, respectivamente. Estos nos permiten comparar los resultados con los modelos previos.

Capítulo 2

Marco Teórico

2.1. Aprendizaje Automático

La automatización de procesos es una idea que ha acompañado al humano durante toda su historia. Pensar en que un proceso pueda ser realizado con la menor asistencia humana posible permite optimizar costos, aumentar el rendimiento, y permite incluso la realización de tareas previamente imposibles en pos del mejoramiento de la calidad de vida humana.

La llegada de las computadoras permitió llevar al máximo la automatización de tareas en áreas específicas, en las cuales la base es la realización de grandes cantidades de operaciones aritméticas. En estas tareas el dominio de las computadoras es enorme, habiendo superado por mucho la capacidad humana.

Sin embargo, existen otras tareas para las cuales el humano ha mostrado siempre mejor desempeño, por ejemplo en tareas de reconocimiento de imágenes, interpretación de texto, habilidades artísticas, entre otras.

Además, es claro que el cerebro humano tiene propiedades deseables a la hora de abordar artificialmente tareas que no son simples de imitar con programas de computadora clásicos, como la capacidad de ser tolerante a fallas del sistema, datos incompletos, inconsistentes o ruidosos, y por sobre todo, la capacidad de adaptación y aprendizaje (Hertz y col. (1991)).

Con base en estas limitaciones es que se desarrolló un nuevo paradigma en los sistemas computacionales, el aprendizaje automático. La idea subyacente es la de diseñar algoritmos computacionales que "aprendan" a desarrollar una tarea, es decir, que sin la programación de una serie sucesiva de reglas, y dada una tarea T, una métrica P que evalúa el desempeño en esa tarea, y una experiencia E, el programa pueda desarrollar mejor la tarea T, medida a traves de P, utilizando su experiencia E (Mitchell (1997)).

2.1.1. Métodos de Aprendizaje

Los métodos a través de los cuales un sistema "aprende" a desarrollar una tarea pueden clasificarse, grosso modo, en tres categorías:

- Aprendizaje Supervisado

En el aprendizaje supervisado, el sistema aprende a desarrollar una tarea basándose en datos etiquetados por expertos humanos. Luego de que el sistema devuelve un resultado, el mismo se compara con la etiqueta, considerada la respuesta correcta, y a partir de esa comparación el sistema aprende.

- Aprendizaje por Refuerzo

  En el aprendizaje por refuerzo, a diferencia del supervisado, la información que se le provee al sistema no es la respuesta correcta, sino información sobre si se ha equivocado o no.

- Aprendizaje no Supervisado

  En el aprendizaje no supervisado no se le brinda ninguna información extra al sistema además de los datos de entrada, y la tarea que debe desempeñar es la de extraer características relevantes sobre los datos.

El método que aplicaremos en este trabajo es el aprendizaje supervisado, ya que disponemos de datos etiquetados que queremos que el modelo aprenda.

2.1.2. Regresión versus clasificación

A grandes razgos, existen dos tipos de tareas posibles a la hora de desarrollar un modelo de aprendizaje automático supervisado: regresión y clasificación.

Como su nombre lo indica, una tarea de clasificación consiste en, dado un conjunto de ejemplos separados en clases enumeradas por un experto humano, poder decidir a que clase pertenece un nuevo ejemplo. El clasificador vive entonces en un espacio de funciones $D \to \{0, 1, 2, \cdots, n-1\}$, donde D es el espacio de los ejemplos, y n es el número de clases.

Una tarea de regresión consiste, por otro lado, en devolver una salida continua, perteneciente a $\mathbb{R}^n$, donde n es la dimensión de salida del modelo, que debe coincidir con la dimensión de la etiqueta.

2.1.3. Conjuntos de entrenamiento y evaluación

En aprendizaje supervisado, uno dispone de un conjunto de datos asociados a sus respectivas etiquetas. Éste usualmente se divide en dos subconjuntos, llamados de entrenamiento y de evaluación, respectivamente. Esto se hace para poder entrenar el modelo en base al primer conjunto, y reportar el rendimiento del modelo basado en una métrica medida sobre el segundo conjunto.

Conjunto de entrenamiento

Cuando aplicamos el modelo a un elemento del conjunto de entrenamiento, podemos comparar la salida del mismo con la etiqueta, y entonces modificar los parámetros del modelo tal que la diferencia entre estas sea mínima.

Conjunto de evaluacion

Si queremos saber si nuestro modelo no ha sobreajustado, y por el contrario ha aprendido a extraer las características relevantes de los datos para realizar la tarea pertinente, es necesario analizar las predicciones del modelo sobre datos etiquetados que no hayan sido utilizados para el entrenamiento. En este sentido se construye el conjunto de evaluación, para poder analizar si el modelo tiene la capacidad de resolver correctamente casos nunca vistos, o sea, si tiene capacidad de generalizar.

2.2. ¿Qué és una Red Neuronal Artificial?

Como explicamos en la Introducción, una red neuronal artificial es un conglomerado de Neuronas Artificiales, unidas a través de una arquitectura de conexiones sinápticas, que tiene la capacidad de aprender a desarrollar una tarea, es decir, es una herramienta de aprendizaje automático. Aparecen entonces dos componentes relevantes, los cuales merecen mayor detalle.

2.2.1. Neurona Artificial

Como se mencionó anteriormente, la neurona artificial embebida en redes de aprendizaje automático es un sistema dinámico simple. A diferencia de un modelado artificial de una neurona biológica, en esta neurona artificial no es necesario ni deseable modelar todos los elementos escenciales para la vida de la neurona, sino solo representar aquellos que hacen parte crucial en el almacenamiento y transmisión de información entre estas unidades. Desde el punto de vista más simple, una neurona es una unidad de procesamiento que recibe señales de entrada, y en función de estas señales emite una señal de salida (ver figura 2.1). Para simplificar el análisis supongamos que esas señales de entrada y salida son binarias, es decir: 0 para una señal inhibitoria y 1 para una señal excitatoria. Entonces, el comportamiento de la neurona puede ser modelado de la siguiente manera:

$$\zeta^n = g(f(\vec{\zeta})) = \begin{cases} 0 & \text{si } f(\vec{\zeta}) < 0 \\ 1 & \text{si } f(\vec{\zeta}) \geq 0 \end{cases} \tag{2.1}$$

$$\text{con } \vec{\zeta} = (\zeta^1, \zeta^2, \cdots, \zeta^{n-1}), \quad \zeta^i \in \{0,1\}, \quad i \in 1,2,\cdots,n.$$

La función $f(x)$ representa el comportamiento de la neurona frente a señales inhibitorias o excitatorias enviadas por otras neuronas. Llamamos a la función $g(x)$ *función de activación*, ya que refleja el comportamiento de la salida para un valor de $f(x)$.

De aquí se pueden observar dos cosas: primero, que el comportamiento de la neurona artificial, aún con la arbitrariedad en la función $f(x)$, es bastante simple. Segundo, que el comportamiento de la neurona depende escencialmente de estar conectada con otras neuronas.

Una forma eficiente de agregar las señales de entrada, es mediante una función $f(\vec{\zeta})$ lineal, es decir:

$$f(\vec{\zeta}) = \sum_{i=0}^{n-1} w_i \zeta^i \tag{2.2}$$

donde $\zeta^0 = 1$ y w_0 es un término independiente llamado *bias*.

Esto nos brinda simplicidad en el cálculo y nos permite modificar los pesos w_i para darle mayor importancia a algunas entradas que a otras.

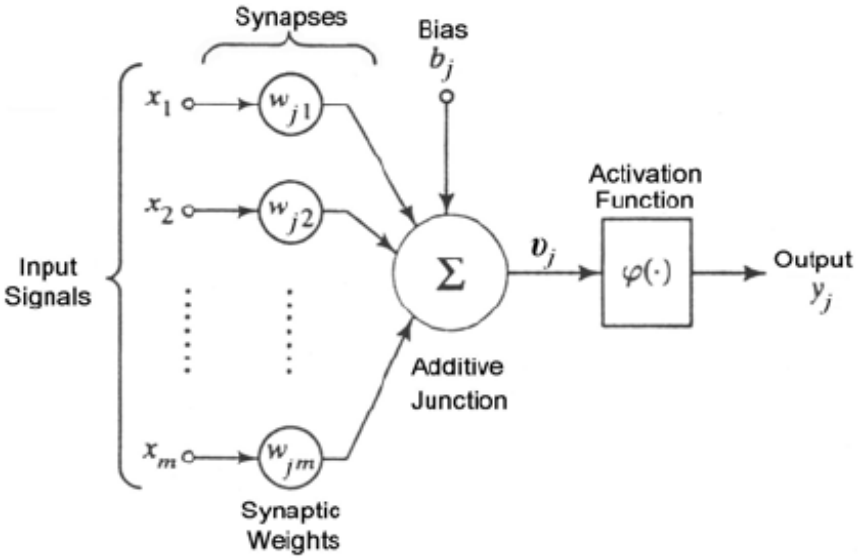


FIGURA 2.1: Esquema de funcionamiento de una neurona artificial.
Imagen extraída de Guarnieri, Pereira y Chou (2006)

2.2.2. Arquitectura de Conexiones

Si analizamos el modelo de neurona detallado en las ecuaciones 2.1 y 2.2, podemos notar que hemos asumido tácitamente algunas cosas:

- Las neuronas no están autoconectadas (su salida no funciona como entrada de la misma neurona).
- En principio, todas las otras neuronas se conectan a ésta neurona. (suponiendo que existen n neuronas en total)

Esto nos está brindando información de cómo están conectadas estas neuronas. La información completa de las conexiones de cada neurona de la red es lo que se conoce como Arquitectura de Conexiones. Hay que tener en cuenta que cuando n es grande, la complejidad de la Arquitectura puede aumentar notoriamente, sobre todo si suponemos que no todas las neuronas se conectan con todas las demás.

A grandes razgos, podríamos clasificar a las arquitecturas de redes neuronales en dos tipos, según como se produce el flujo de información en la red:

1. Redes Prealimentadas (Feed-Forward en inglés), que se caracterizan por que el grafo que las representa no presenta ciclos, es decir, la información fluye en una dirección definida, desde un inicio hasta un final.

2. Redes Recurrentes, que permiten un comportamiento cíclico en el conjunto total o en un subconjunto de neuronas.

En nuestro problema nos interesarán particularmente las arquitecturas del primer tipo.

2.2.3. Redes Feed-Forward

Vamos a analizar la estructura general de las redes Feed-Forward, en las cuales el proceso de aprendizaje es particularmente simple de entender. Una red Feed-Forward tiene la particularidad de que se pueden caracterizar a sus neuronas componentes como neuronas de entrada, neuronas ocultas y neuronas de salida. Lo más simple es que en estos modelos las neuronas estén organizadas en capas, en donde cada neurona de una capa tiene conexiones de entrada con neuronas de la capa inmediatamente anterior y conexiones de salida con la capa inmediatamente siguiente. Existen excepciones dentro de estos modelos, como conexiones direccionadas dentro de una misma capa o conexiones que se "saltan" una o varias capas, y conectan capas que no son consecutivas. El tipo más simple dentro de las redes Feed-Forward en capas es el Perceptrón multicapa, el cual fué estudiado en detalle originalmente por Rosenblatt (1961), y cuyo modelado matemático es el siguiente:

$$\zeta_i^j = g\left(\sum_{k=0}^{n_{i-1}} w_{i,k}^j \zeta_{i-1}^k\right) \tag{2.3}$$

donde:

- ζ_i^j es la salida de la neurona j en la capa i.
- n_{i-1} es el número de neuronas en la capa $i-1$.
- $w_{i,k}^j$ es el peso de la conexión entre la neurona ζ_i^j en la capa i y la neurona ζ_{i-1}^k en la capa $i-1$.
- $g(x)$ es una función de activación no lineal.

La figura 2.2 nos muestra un esquema de la arquitectura de un perceptrón de 3 capas (notar que la capa de entrada no se tiene en cuenta). Las unidades de la capa de entrada cumplen la única función de alimentar a la red con características o *features* de entrada. Luego vienen una o varias capas ocultas o capas intermedias, seguidas de la capa de salida en donde se lee el resultado computado. Queda claro a partir de la figura que la red es direccionada, no posee bucles y no hay conexiones entre neuronas de una misma capa (Hertz y col. (1991)).

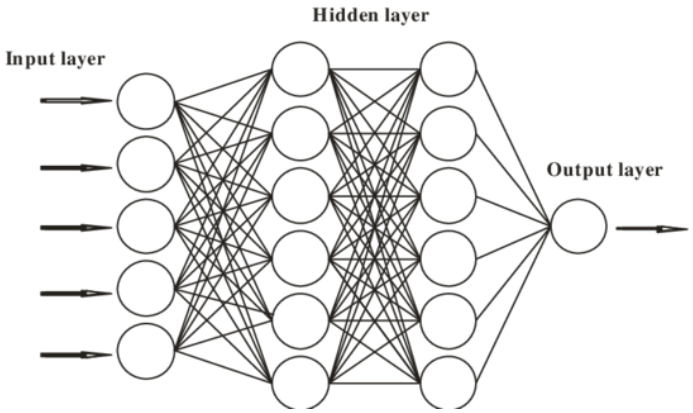


FIGURA 2.2: Esquema de una red Feed-Forward tipo Perceptrón multicapa, la versión más simple de red Feed-Forward. Imagen extraída de Torre y col. (2015)

2.3. Aprendizaje de la red

Como se vió en la sección 2.2, las redes neuronales tienen la capacidad de aprender a realizar una tarea en específico. Ésto se realiza aprendiendo los pesos w_i^j de las conexiones. Para realizar este entrenamiento se introducirán algunos conceptos relevantes.

2.3.1. Función de Costo y Backpropagation

Para que una FFNN aprenda, necesita comparar las salidas que genera con las respuestas etiquetadas por humanos. La medida del error que tiene la salida para reproducir la etiqueta es llamada Función de Costo. Si denotamos tanto a la dimensión de la salida del modelo "$\hat{y}$" como a la de las etiquetas "y" por n, la función de costo va de $\mathbb{R}^{2n}$ a $\mathbb{R}$, es decir, es una función que recibe un resultado del modelo, y su etiqueta correspondiente, y devuelve un valor escalar. La función de costo tiene que ser mínima para una salida igual a la etiqueta, y debe estar acotada inferiormente en el espacio de parámetros w_i^j.

Para generar aprendizaje, se busca minimizar la función de costo en el espacio de los parámetros, para lo cual se utiliza alguna técnica de Descenso por Gradiente. Para ésto, debemos calcular el gradiente de la función de costo con respecto a los parámetros w_i^j. La forma de hacer esto para RNA estructuradas en capas, es mediante un método llamado Backpropagation (propagación de errores hacia atrás), el cual se basa en aplicar iterativamente la regla de la cadena de diferenciación sobre cada capa, desde el final hasta el principio de la red, haciendo uso de los gradientes de la capa calculada anteriormente.

Dentro de los algoritmos de descenso por gradiente, vamos a mencionar algunos de ellos:

- Descenso por gradiente clásico

Es un método iterativo de primer orden para encontrar mínimos. Dada una función $F(\vec{X})$, un paso en la actualización es:

$$\vec{\theta}_{t+1} = \vec{\theta}_t - \lambda \nabla_\theta J(\vec{\theta}_t).$$

donde $J(\theta)$ es el costo promediado sobre todo el conjunto de entrenamiento, θ es el conjunto de parámetros, y ∇_θ es el gradiente con respecto a los parámetros. λ es un valor llamado tasa de actualización, y determina cuan rápido nos movemos hacia el mínimo.

- Descenso por gradiente estocástico en minibatch con momento

  Este método, aplicado en Arik, Ibragimov y Xing (2017), se diferencia del anterior en que en vez de calcular la función de costo sobre todo el conjunto de entrenamiento, elige aleatoriamente una cantidad b de elementos, llamada *batch*, y calcula el gradiente sobre este valor de la función de costo. Además permite la inclusión de un parámetro llamado momento α, que funciona de manera similar a un momento lineal en física, es decir, para el cálculo del paso $t+1$, existe un término asociado al gradiente del paso t, y otro término, pesado por α, que lleva al optimizador a avanzar en la dirección del paso $t-1$, es decir, tiene una "resistencia" a cambiar su dirección de movimiento. Podemos ver el comportamiento de un paso en la actualización, definiendo:

  $$\vec{\theta}_{t+1} = \vec{\theta}_t + \Delta\vec{\theta}_{t+1},$$

  donde ahora definimos a $\Delta\vec{\theta}_{t+1}$ como:

  $$\Delta\vec{\theta}_{t+1} = -\lambda \nabla_\theta J(\vec{\theta}_t) + \alpha \Delta\vec{\theta}_t$$

  Aquí $\Delta\vec{\theta}_t$ sólo depende del valor de los parámetros θ y del gradiente de la función de costo con respecto a θ en el tiempo $t-1$.

- Adam

  Este método, basado en descenso por gradiente estocástico, tiene la particularidad de calcular dinámicamente el valor del momento, presenta una eficiencia computacional mayor que algoritmos previos, y tiene un buen comportamiento para problemas con gran cantidad de datos y/o parámetros. (Kingma y Ba (2014))

2.3.2. Función de Activación

Se desprende de lo anterior que la dependencia de la función de costo con respecto a los parámetros debe ser diferenciable o, al menos, Lipschitz continua. Esto no ocurre con la función de activación tipo escalón definida en la ecuación 2.1, que

tiene un punto de discontinuidad en 0. Se debe entonces buscar una función de activación que responda correctamente a las entradas de esa capa, y que satisfaga las condiciones de continuidad y diferenciabilidad.

Algunas de las funciones más comunes que satisfacen estas condiciones son (ver figura 2.3):

- **ReLU**

$$g(x) = max(0, x)$$

- Sigmoide

$$g(x) = \frac{1}{1 + e^{-x}}$$

- Tangente hiperbólica

$$g(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

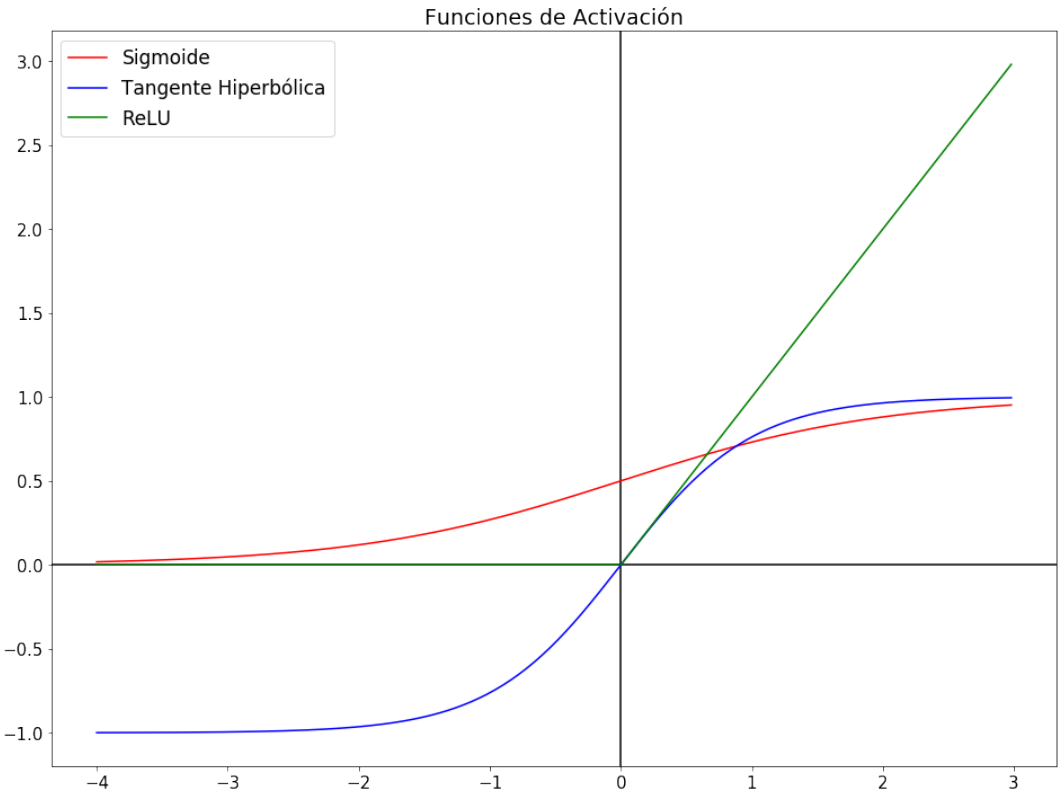


FIGURA 2.3: Gráfica de distintas funciones de activación. Es importante observar que las imágenes de las mismas son distintas, siendo $(0, 1)$ para la función sigmoide, $(-1, 1)$ para la tangente hiperbolica, y $[0, \infty)$ para la función ReLU.

De estas funciones, la más estable numéricamente, que genera menos problemas de extinción de gradiente, y que ha mostrado empíricamente mejores resultados en problemas de imágenes es la función **ReLU**.

2.4. Redes Neuronales Convolucionales

Pensemos ahora en una función $f(\vec{\zeta})$ cuya imagen no esté restringida al conjunto de los reales. Nuestra neurona ahora vive en el espacio de las funciones $\mathbb{R}^n \to \mathbb{R}^m$, con m la dimensión de salida.

2.4.1. Neuronas Convolucionales

Si definimos nuestra función

$$f(\vec{\zeta}) = \left(\sum_i \mathbf{K}_i \star \boldsymbol{\zeta}_i \right) + b \tag{2.4}$$

donde $\star$ representa la operación de convolución discreta, b representa el término independiente o *bias* y $\mathbf{K}_i$ representa un núcleo de convolución, conformado por parámetros ajustables, entonces obtenemos una neurona convolucional. Esta operación sigue siendo lineal, conmutativa, asociativa, y posee un elemento de identidad.

La operación de convolución discreta, en su versión de dos dimensiones (la opción por defecto en datos tipo imagen), tiene la siguiente forma (ver figura 2.4):

$$C[m,n] = \sum_{u=-l}^{l} \sum_{v=-l}^{l} K[u,v]\zeta[m-u, n-v] \tag{2.5}$$

Donde:

- $K[u,v]$ es el elemento del núcleo $\mathbf{K}$ en la fila u y columna v
- $\mathbf{K}$ tiene $(2l+1) \times (2l+1)$ elementos
- $\boldsymbol{\zeta}$ es 2-dimensional, de $d \times d$ elementos
- $\mathbf{C}$ es la salida de la convolución, también 2-dimensional, en principio de $(d-2l) \times (d-2l)$ elementos

Nuestra función de activación vive ahora en el espacio de funciones $\mathbb{R}^{(d-2l)\times(d-2l)} \to \mathbb{R}^{(d-2l)\times(d-2l)}$, aplicando la activación a cada elemento de la salida de $f(\boldsymbol{\zeta})$.

La ventaja principal que proporciona el uso de una CNN es la capacidad de preservar la localidad y la espacialidad de la información. A diferencia de una red completamente conectada, en las CNN cada elemento de la matriz de salida de una neurona posee información de una región de la matriz de entrada que es acotada y simplemente conexa espacialmente. Ésto es especialmente útil a la hora de analizar datos de entrada que presenten relación espacial, como sucede con las imágenes.

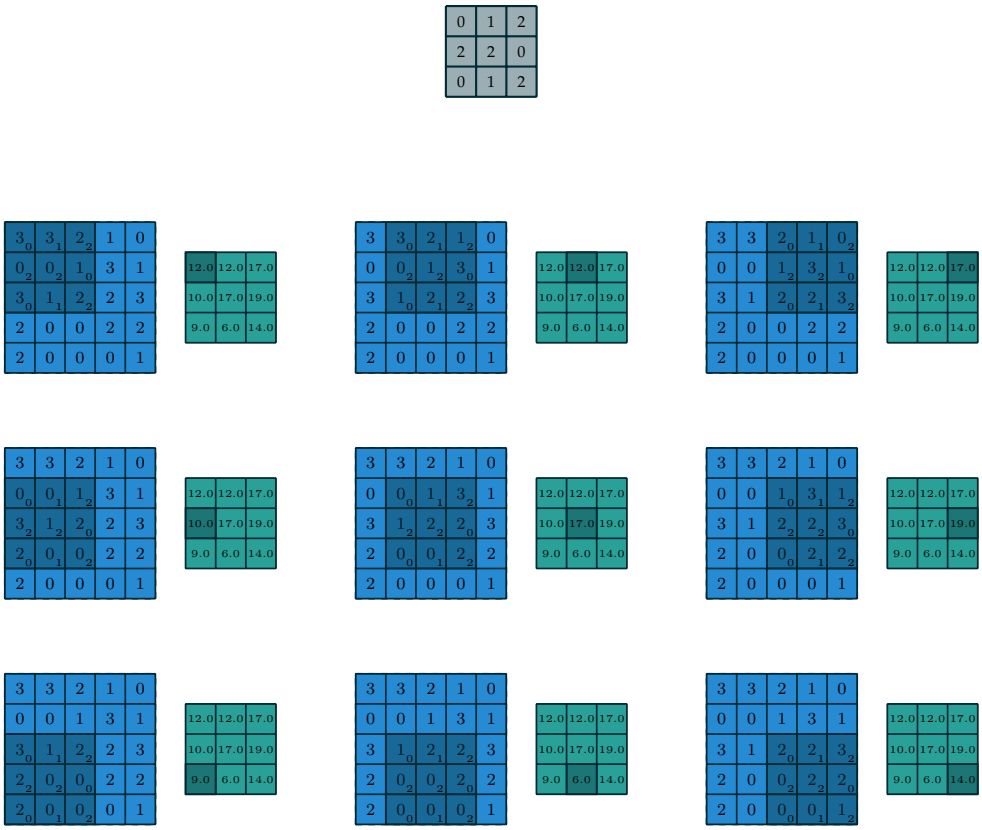


FIGURA 2.4: Ejemplo de convolución discreta, en su forma más simple. El núcleo se presenta en gris, la entrada en celeste y la salida en verde. Imagen extraída de Dumoulin y Visin (2016).

2.4.2. Capa Convolucional

Analizando la ecuación 2.4, podemos ver que una neurona en una capa tiene varios núcleos convolucionales, asociados a distintas entradas ζ_i. Estas entradas son llamadas "canales", y se corresponden a las salidas generadas por neuronas distintas pertenecientes a la capa anterior. Entonces, la salida de una capa convolucional es 3-dimensional, con 2 dimensiones espaciales y una de canales o "profundidad". Para que las dimensiones espaciales de salida sean consistentes, todos los núcleos convolucionales en una capa deben ser del mismo tamaño. Entonces, para simplificar la notación, dada una capa convolucional a la cual le entran c canales de dimensión $d \times d$, decimos que cada neurona aplica un único núcleo de dimensión $(2l+1) \times (2l+1) \times c$, que en realidad corresponde a c núcleos apilados a través de la dimensión de los canales (ver figura 2.5).

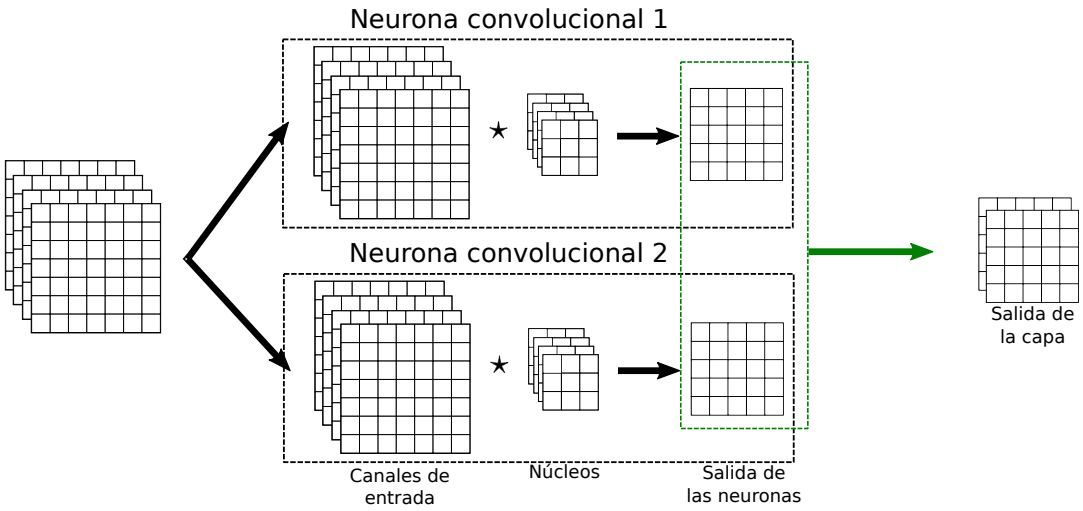


FIGURA 2.5: Esquema de capa convolucional con 2 neuronas de núcleo 3×3 (en realidad $3 \times 3 \times 4$, donde la cantidad de canales del núcleo queda implícitamente definida por la cantidad de canales de entrada).

2.4.3. Parámetros de la convolución

Existen una serie de parámetros, además del tamaño del kernel, que permiten modificar la forma en que opera una capa convolucional. Estos son los siguientes (ver Dumoulin y Visin (2016)):

- Relleno o *padding*: La operación de padding me permite aumentar la dimensión de salida de una convolución, permitiendome mayor libertad en la dimensión de salida que la que me permite el tamaño del kernel. La forma de realizarlo es rellenar los contornos de la entrada de la capa, es decir, si la entrada originalmente era de dimensión $\mathbb{R}^{d \times d}$, un padding de (l, l) agrega l filas y l columnas en los bordes horizontales y verticales superior e inferior de la imagen, generando una entrada de dimensión $\mathbb{R}^{(d+2l)\times(d+2l)}$. La opción por defecto es rellenar con ceros, pero existen otros tipos de relleno, como continuar el borde, condiciones de contorno cíclicas o rellenar con la media de la imagen, entre otros posibles.

- Paso o *stride*: Así como la operación de padding permite aumentar la dimensión de salida, el stride me permite reducirla más que la reducción generada por el kernel. El paso me dice cada cuantos píxeles de la entrada aplico la operación de convolución de la ecuación 2.5, es decir, el intervalo con el cual tomo los m, n en la función $G[m, n]$. Un stride de (s, s) genera una salida:

$$\mathbf{G} = \begin{bmatrix} G[0,0] & G[0,s] & G[0,2s] & \cdots \\ G[s,0] & G[s,s] & & \\ G[2s,0] & & \ddots & \\ \vdots & & & \ddots \end{bmatrix}$$

y la dimensión espacial de $\mathbf{G}$ es $\mathbb{R}^{([d-2l-1]/s+1)\times([d-2l-1]/s+1)}$, si el argumento $[d-2l-1]/s+1$ es entero, o la parte entera del mismo en su defecto.

2.4.4. Operaciones usuales en Redes Convolucionales

Existen además de las capas convolucionales y las funciones de activación, otras operaciones usuales de común aplicación en las redes convolucionales. Vamos a hacer hincapié en las dos más comunes, y que son de aplicación en este trabajo.

- *Pooling*: La operación de pooling es una operación no paramétrica (no se aprenden pesos) que sirve para reducir la dimensión espacial de una imagen o activación. La operación tiene en común con la convolución en que opera a través de un kernel de una dimensión dada, tiene un stride definido y permite la operación de padding. Las formas más usuales de aplicar un pooling, dado un tamaño del kernel k, son las siguientes:
  - Max Pooling: dada una ventana $k \times k$ de la entrada, tomo el máximo de los valores de esa ventana como salida.
  - Average Pooling: dada una ventana $k \times k$ de la entrada, tomo el promedio de los valores de esa ventana como salida.
- *Batch Normalization*: La operación de batch normalization o BatchNorm toma una entrada del tamaño del batch elegido, y la normaliza tal que la media de los valores sobre el batch sea 0 y su varianza sea 1. Esto nos permite mejorar el tiempo de cómputo y la convergencia a mejores mínimos, logrando que ningún elemento de la activación se vuelva demasiado dominante sobre el resto.

2.4.5. Campo receptivo e información local

Analizando una red convolucional de varias capas, podemos definir el campo receptivo de un pixel en la salida de una dada capa. Esto es, la región de la imagen de entrada de la red que, a través de las capas anteriores, aporta información para el cálculo del valor de ese pixel. Se puede ver (figura 2.6) que, mientras más capas haya entre la imagen de entrada y el pixel de salida, mayor será la dimensión espacial del campo receptivo. También depende del tamaño de los núcleos de las capas anteriores, siendo mayor la dimensión espacial del campo receptivo mientras mayor sea la dimensión espacial de los núcleos. Sin embargo, como se explica en Luo y col., 2016, el campo receptivo efectivo de una salida puede ser en realidad mucho menor que el teórico, principalmente por la mayor influencia de los píxeles centrales en el mismo. Entonces, teniendo en cuenta que para las tareas que se buscan desarrollar a través de redes convolucionales no basta con la información de intensidad local de la imagen, sino que es necesario tener información global de las relaciones de posición, ángulo y forma de las estructuras generalmente extensas que conforman la imagen, es conveniente pensar en arquitecturas que logren expresar de maneras más precisas las estructuras no locales.

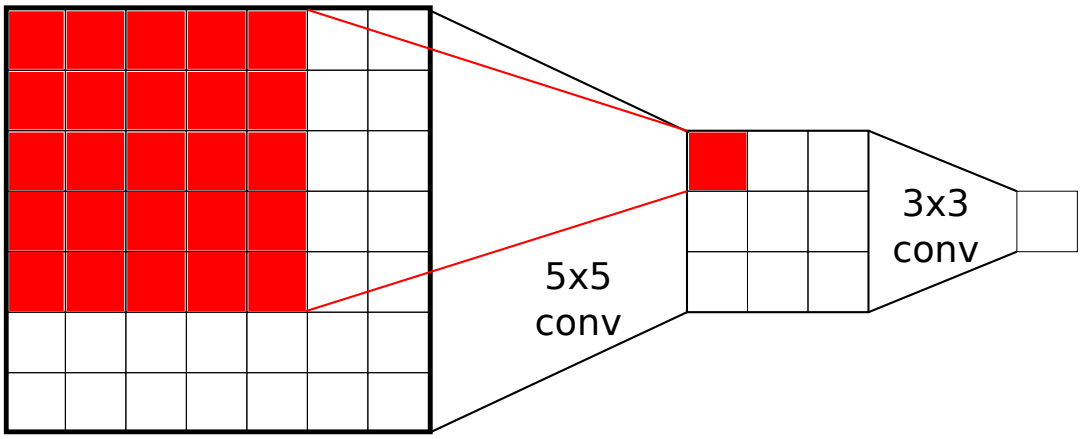


FIGURA 2.6: Esquema de representación del campo receptivo teórico a través de operaciones convolucionales. La región roja en la primera capa representa el campo receptivo del píxel rojo en la segunda capa, mientras que la primera capa completa es el campo receptivo del píxel de la tercera capa.

2.4.6. Capa Inception

En Szegedy y col. (2015) se propuso una idea novedosa para generar distintos niveles de localidad en el campo receptivo, que pudieran aportar información de regiones cercanas y lejanas al area de interés, mediante la aplicación de una arquitectura específica de capa conocida como capa inception. (ver figura 2.7)

La idea detrás de la capa inception es que, dada una neurona convolucional, el campo receptivo de la salida con respecto a las activaciones de la capa inmediatamente anterior está fijado por el tamaño del núcleo. Entonces, si se aplican nucleos de distinto tamaño en paralelo, y luego se unen las salidas, se genera una activación que tiene información de distintos campos receptivos, es decir, obtenida a partir de distintos niveles de localidad.

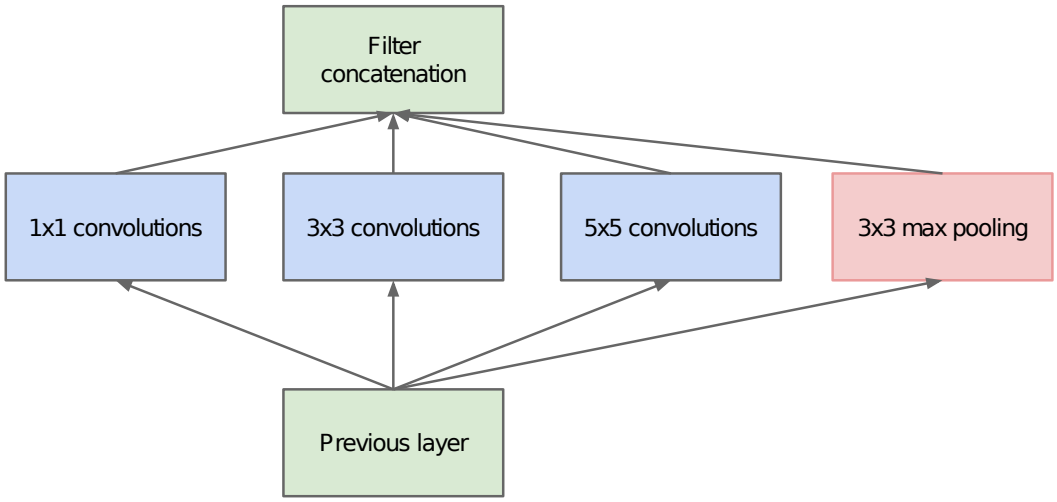


FIGURA 2.7: Esquema de la versión naïve del módulo inception. Como se puede ver, se aplican distintas capas convolucionales o de pooling en paralelo, y luego se las concatenan en la dimensión de los canales. Figura extraída de Szegedy y col. (2015).

2.4.7. Autoencoders

En las tareas de reducción de dimensionalidad, extracción de características o *features*, inicialización de parámetros y extracción de ruido, existe una herramienta basada en redes neuronales que ha demostrado grandes desempeños, el autoencoder. En su forma mas básica, un autoencoder funciona como dos redes feed-forward una después de la otra: el encoder, una red que se caracteriza por que la dimensión de salida es menor a la de entrada, y el decoder, una red cuya dimensión de salida es mayor a la de entrada y que satisface que (ver figura 2.8):

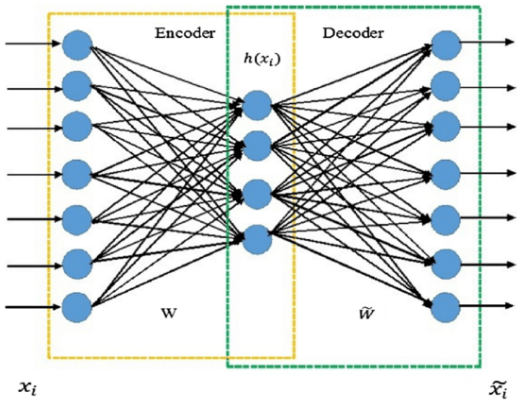


FIGURA 2.8: Ejemplo de la arquitectura básica de un autoencoder. Figura extraída de Ahmed, L. Dennis Wong y Nandi (2018).

$$input_1 \in \mathbb{R}^n \rightarrow output_1 = encoder(input_1) \in \mathbb{R}^m, m < n$$

$$input_2 = output_1 \rightarrow output_2 = decoder(input_2) \in \mathbb{R}^n.$$

La idea en esta arquitectura es que el modelo aprenda a reproducir la entrada con el menor error posible, usualmente medido a través del error cuadrático medio, es decir:

$$\frac{1}{N}\sum_{i=1}^{N}\|input_1^i - output_2^i\|^2.$$

donde N es el número de ejemplos en el batch.

A la información almacenada en $output_1$ se le llama usualmente "espacio latente", y es una codificación de la imagen en un espacio de dimensión inferior, que almacena la información más importante de la misma.

2.5. Cefalometría Odontológica

La cefalometría es el estudio y medición de la cabeza humana, usualmente a través de imágenes médicas como radiografías. En general tiene diversas áreas de aplicación, y dentro de ellas una de las más comunes es la odontología y, en particular, la ortodoncia. En este campo, la cefalometría como medición de las estructuras craneales, maxilares y dentales es un método que, empleando radiografías orientadas, obtiene medidas lineales y angulares de los elementos anatómicos de craneo y cara, ofreciendo información para la elaboración de los análisis cefalométricos (Vellini-Ferreira (2002)).

El análisis cefalométrico odontológico juega un rol importante en el diagnóstico clínico, tratamiento y cirugía. Estos describen la interpretación de las estructuras óseas, dentales y de bordes blandos del paciente, y permiten el análisis ortodóntico y el planeamiento del tratamiento (Wang y col. (2016)).

En los momentos tempranos del desarrollo de la ortodoncia, el único registro era el paciente. Se podía medir y describir sobre él, pero solo de manera externa y superficial. El principal objetivo del tratamiento ortodóntico consistía entonces en la alineación de los dientes visibles. Sin embargo, con el tiempo la evolución de los objetivos de la ortodoncia para elaborar un mejor diagnóstico, en base a conocer la naturaleza de la enfermedad y conocer para ello las características de normalidad, llevó a la exploración de nuevos métodos de diagnóstico (Arcieri y col. (2016)). En base a esto es que se desarrollan diferentes cefalogramas, como el presentado en Ricketts (1982), uno de los más completos en cuanto a características analizadas.

Teniendo en cuenta la complejidad de la realización de los cefalogramas (ver figura 2.9), y que estos consumen una gran cantidad de tiempo, tanto a la hora de aprender la técnica por parte de los ortodoncistas, como a la hora de la realización misma, es que se plantea como un problema interesante el desarrollo automático de los mismos.

Durante las últimas décadas se ha explorado la detección automática de puntos cefalométricos. Cardillo y Sid-Ahmed (1994) usaron correspondencia de patrones (*template matching*) y operadores morfológicos en escala de grises para identificación automática de puntos cefalométricos. El-Feghi, Sid-Ahmed y Ahmadi (2004) usaron técnicas de aprendizaje automático basadas en sistemas *neuro-fuzzy* (combinación entre redes neuronales artificiales y sistemas difusos) y clustering k-means para análisis cefalométricos automáticos. Yue y col. (2006) combinaron extracción de parches de imagen con un modelo de forma basado en análisis de componentes principales.

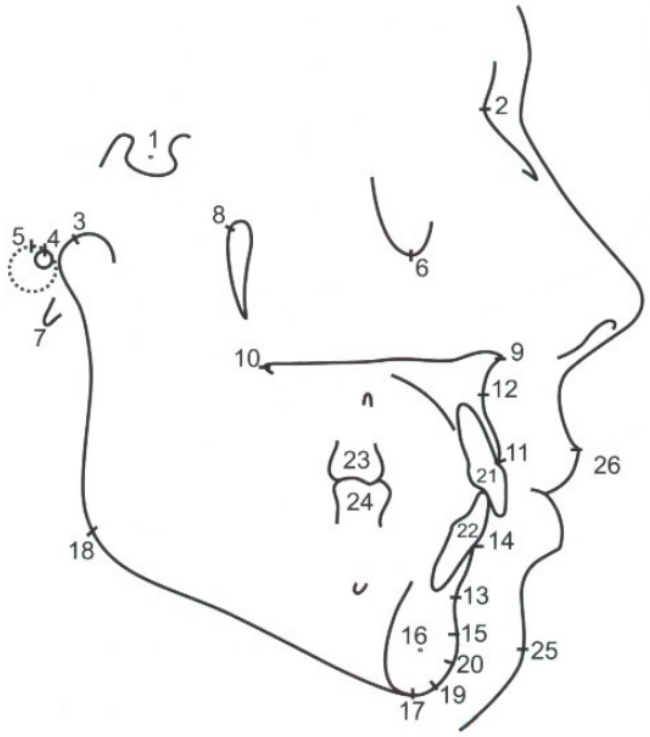


FIGURA 2.9: Ejemplo de cefalograma. Se puede ver la demarcación de puntos cefalométricos y estructuras anatómicas. Imagen extraída de Vellini-Ferreira (2002)

Recientemente, Wang y colaboradores (Wang y col. (2015) y Wang y col. (2016)) organizaron dos competencias públicas en cefalometría automática y registraron el desempeño del estado del arte en algoritmos de detección de puntos cefalométricos. Los mejores algoritmos de esas competencias están basados en *Random Forest* para clasificación de patrones de intensidad para puntos cefalométricos individuales, y análisis estadísticos de forma para explotar la relación espacial entre puntos.

Arik, Ibragimov y Xing (2017) proponen el primer modelo de red neuronal convolucional profunda para la generación de mapas de probabilidad de intensidad en la detección de landmarks, combinado con un modelo estadístico de forma. Vamos a explicar mas detalladamente este modelo, ya que será el modelo base para comparar con los desarrollados en el presente trabajo.

2.5.1. Descripción del modelo de Arik, Ibragimov y Xing (2017)

En este trabajo, el problema planteado para la generación de un mapa de probabilidades basado en intensidad se basa en la evaluación de parches de la imagen original, con un dado tamaño, para determinar si el pixel central de ese parche es o no un landmark en específico. Denotamos por $\mathbf{I}_{(x_i,y_i)}$ al parche $N \times N$ centrado en el landmark l, con N lo suficientemente grande como para reconocer visualmente que el pixel (x_i, y_i) representa al landmark. El objetivo es encontrar L funciones (con L el número de landmarks) $g_l : [0,255]^N \times [0,255]^N \rightarrow [0,1]$ (suponiendo que los valores

de los pixeles están en el rango $[0, 255]$) tales que $g_l(\mathbf{I}_{(x_i,y_i)})$ sea una estimación de la probabilidad de que el pixel (x_i, y_i) sea el landmark l. Idealmente, debería valer 1 solo cuando el pixel i es el landmark l, y 0 para cualquier otro pixel. Entonces, el problema se plantea como una clasificación binaria, con etiquetas 0 y 1.

Para poder transformar el problema en una clasificación, se realizan una serie de transformaciones al conjunto de datos:

Primero, para reducir el costo computacional sin perder demasiada información relevante, se reduce la dimensión espacial de las imágenes en $h/3 \times w/3$, donde h y w son el alto y el ancho de las imágenes originales, respectivamente.

Para la construcción del conjunto de entrenamiento, ee eligen píxeles de manera aleatoria en un círculo de radio r_1 en el entorno del landmark verdadero, y se extiende la definición de clase positiva (1) a estos puntos.

Luego, se eligen aleatoriamente puntos en un anillo entre radios r_2 y r_3, con $r_1 < r_2 < r_3$, y se etiquetan estos puntos como puntos de clase negativa (0). (ver figura 2.10)

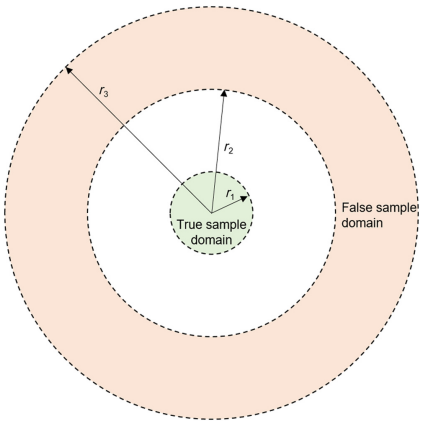


FIGURA 2.10: Esquema del dominio de ejemplos verdaderos y falsos. Los píxeles son extraídos aleatoriamente de estas regiones. (Arik, Ibragimov y Xing (2017))

A partir de estos puntos se generan parches de 81×81 centrados en los mismos, y éstos se utilizan como entrada de la red.

La arquitectura de la red es la de una red convolucional clásica, apilando capas convolucionales, activaciones ReLU y operaciones de Max Pooling, hasta que a la salida de la última capa convolucional, cuando la dimensión de la activación llega a $1 \times 1 \times 25$, se aplica una red completamente conexa, al estilo perceptrón multicapa, para obtener una salida escalar, la cual va entre cero y uno, y es una estimación de la probabilidad de que el pixel central sea el landmark (ver figura 2.11).

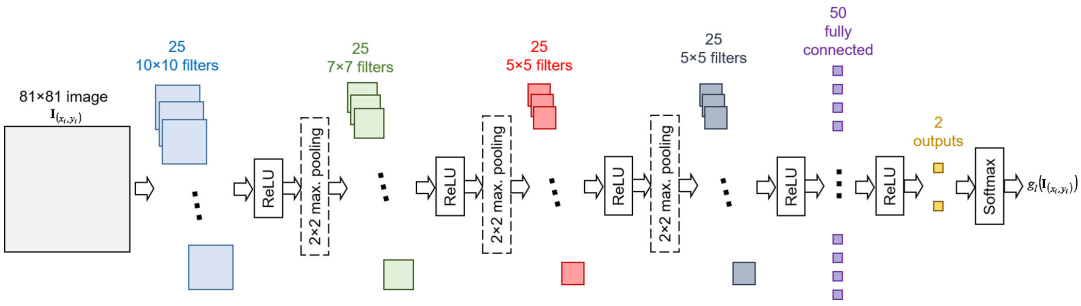


FIGURA 2.11: Esquema de la red utilizada en Arik, Ibragimov y Xing (2017).

A modo de prueba, se reprodujo la red convolucional de este trabajo, siguiendo las especificaciones detalladas en el mismo. En la figura 2.12 se puede observar el mapa de probabilidad de un landmark para una imagen del conjunto de evaluación.

FIGURA 2.12: Mapa de probabilidad generado para el punto L1 o Sella a partir del modelo de Arik, Ibragimov y Xing (2017).

En este capítulo hemos realizado una revisión conceptual del trabajo abordado en esta tesina, comenzando por la descripción del uso de redes neuronales para aprendizaje atomático, siguiendo por definición del método de back propagation para entrenar redes feed forward, y describiendo las redes convolucionales, que utilizaremos en el próximo capítulo. Hemos descripto también el problema de imágenes que nos interesa aboradar como así también hemos presentado una revisión bibliográfica breve y hemos descripto con cierto grade de detalle el trabajo que, a nuestro conocimiento actual, mejor resolvió el problema en cuestión. En los próximos 3 capítulos abordaremos la parte original de esta tesis.

Capítulo 3

Metodología

En este capítulo explicaremos los modelos neuronales que utilizaremos en el capítulo siguiente. En particular presentaremos las arquitecturas utilizadas, así como las funciones de costo, descripción del conjunto de datos y algoritmos de actualización de parámetros a aplicar en este trabajo. Vale destacar que las arquitecturas presentadas en este trabajo, a las cuales llamamos "autoencoder restringido", son originales en la literatura.

3.1. Conjunto de Datos

El conjunto de datos está compuesto por 400 radiografías cefalométricas obtenidas de 400 pacientes con una edad entre 6 y 60 años. Los cefalogramas fueron adquiridos en formato TIFF con el dispositivo Soredex CRANEXr Excel Ceph machine (Tuusula, Finland), y el software Soredex SorCom (3.1.5, version 2.0), y la resolución de las imágenes es de 1935×2400 píxeles. (ver figura 3.1)

Para la evaluación, especialitas marcaron manualmente 19 landmarks en cada imagen y estos datos fueron revisados por dos doctores en medicina experimentados. La posición de los landmarks considerada como verdadera es el promedio de los marcados realizados por los dos doctores (Wang y col., 2016). La variabilidad media intra-observador es de 1.73 y 0.90 *mm* para los dos expertos, mientras que la variabilidad media inter-observador es de 1.38 *mm* (Arik, Ibragimov y Xing, 2017).

El dataset fue particionado de la misma manera que en el IEEE ISBI 2015 Challenge, con 150 imágenes de entrenamiento, 150 imágenes en el conjunto de test1 y 100 imágenes en el conjunto de test2.

Las imágenes fueron reducidas por 3 en las dos dimensiones espaciales, tomando el promedio de cada parche 3×3 para reducción de complejidad del modelo.

Puede observarse a modo de ejemplo la localización de cada landmark y su nombre en la figura 3.1 y en la tabla 3.1, respectivamente.

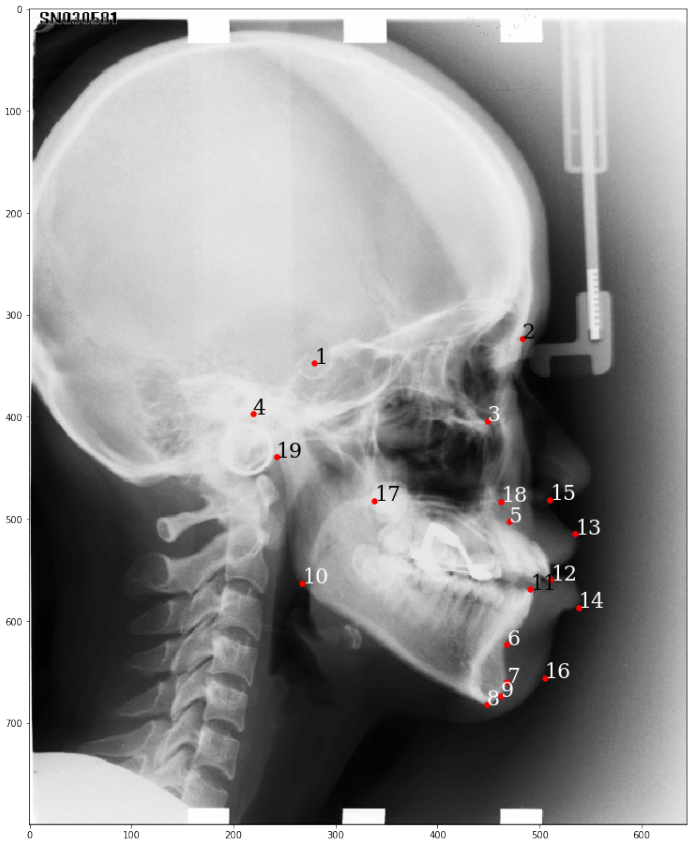


FIGURA 3.1: Imagen del dataset con los landmarks marcados en rojo. (Wang y col., 2016)

TABLA 3.1: Lista de puntos anatómicos (landmarks)

$N°$ de landmark	nombre anatómico
L1	Sella
L2	Nasion
L3	Orbitale
L4	Porion
L5	Subspinale
L6	Supramentale
L7	Pogonion
L8	Menton
L9	Gnathion
L10	Gonion
L11	Lower incisal incision
L12	Upper incisal incision
L13	Upper lip
L14	Lower lip
L15	Subnasale
L16	Soft tissue pogonion
L17	Posterior nasal spine
L18	Anterior nasal spine
L19	Articulate

3.2. Hardware y software utilizados

El hardware utilizado para el desarrollo, entrenamiento y testeo de los modelos fue el siguiente:

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.

El software utilizado fue:

Python 3.6

Librería PyTorch

3.3. Descripción de Modelos

3.3.1. Autoencoder Convolucional

Como desarrollo original de esta tesina, se inicia estudiando la generación de mapas de verosimilitud a través de un autoencoder convolucional simple "restringido". La arquitectura está compuesta de la aplicación secuencial de capas convolucionales y operaciones de max pooling primero, hasta llevar a un mínimo la cantidad de variables que almacenan información (el espacio latente), y la aplicación secuencial posterior de capas convolucionales transpuestas, las cuales permiten aumentar el dominio espacial aprendiendo los parámetros (ver Dumoulin y Visin (2016)). La restricción impuesta a nuestro autoencoder proviene de la tarea asignada. Como queremos que la red reproduzca a la salida un mapa de verosimilitud que indique la probabilidad de cada pixel de ser un landmark, no se entrena a la red a reproducir la imagen de entrada, si no un mapa con las dimensiones espaciales de la imagen, construido manualmente, que tiene una distribución de probabilidad gaussiana centrada en el landmark correcto (etiquetado), con un valor de σ relacionado con las restricciones impuestas en las métricas del problema, es decir, detectar con un error máximo de $2mm$ la posición del landmark. La idea de entrenar con un mapa gaussiano y no con un mapa binario que solo valga $p = 1$ en la posición correcta

del landmark, proviene de permitirle al modelo una mayor libertad de equivocarse dentro de los márgenes permitidos, y prevenir un sobreajuste.

El algoritmo de actualización elegido aquí es el algoritmo Adam, el cual presenta notables mejoras en el tiempo de convergencia con respecto a los algoritmos de descenso por gradiente tradicionales (Kingma y Ba, 2014).

3.3.2. Autoencoder Inception

El siguiente modelo desarrollado parte de dos ideas: primero, la generación de una red más profunda, capaz de abstraer más la información de la imagen de entrada, y segundo, lograr que la red obtenga de manera más eficiente información de distintos campos receptivos, es decir, de distintas escalas espaciales. Para ello, luego de reducir la dimensión espacial de los datos de entrada a través de capas convolucionales simples y operaciones de max pooling al inicio del encoder, se procede a aplicar capas basadas en las versiones *InceptionD* e *InceptionE* de la implementación de la red Inception-V3 en PyTorch. En este punto, se procede a aumentar nuevamente la dimensión espacial a través de convoluciones transpuestas. Al igual que en el modelo anterior, el autoencoder sigue las mismas restricciones respecto a la reproducción de un mapa de gaussianas.

En este modelo, el optimizador elegido es el mismo que en 3.3.1.

Se desarrollan dos variantes de este modelo, que difieren únicamente en la cantidad de núcleos convolucionales transpuestos aplicados en las capas del decoder, con 32 y 64 núcleos. Estos modelos serán llamados **autoencoder inception 32** y **autoencoder inception 64**, respectivamente.

En la figura 3.2 se puede observar el esquema general de la arquitectura, mientras que en las figuras 3.3 y 3.4 se pueden observar en mayor detalle las capas aplicadas en cada instancia del encoder y del decoder, respectivamente, para las dos variantes del modelo.

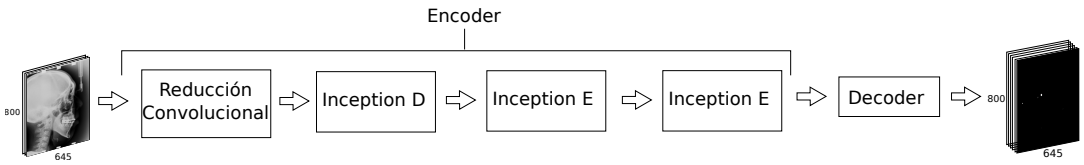
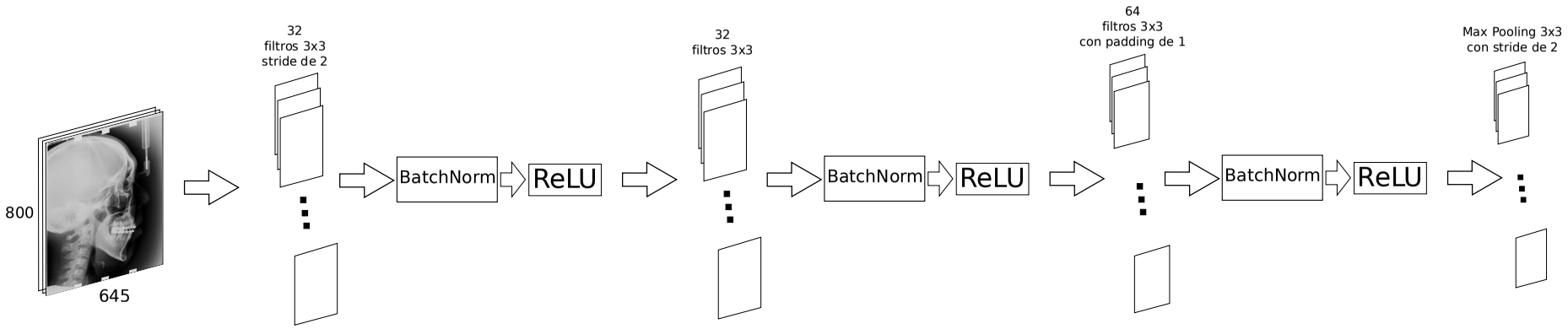


FIGURA 3.2: Esquema general del Autoencoder Inception

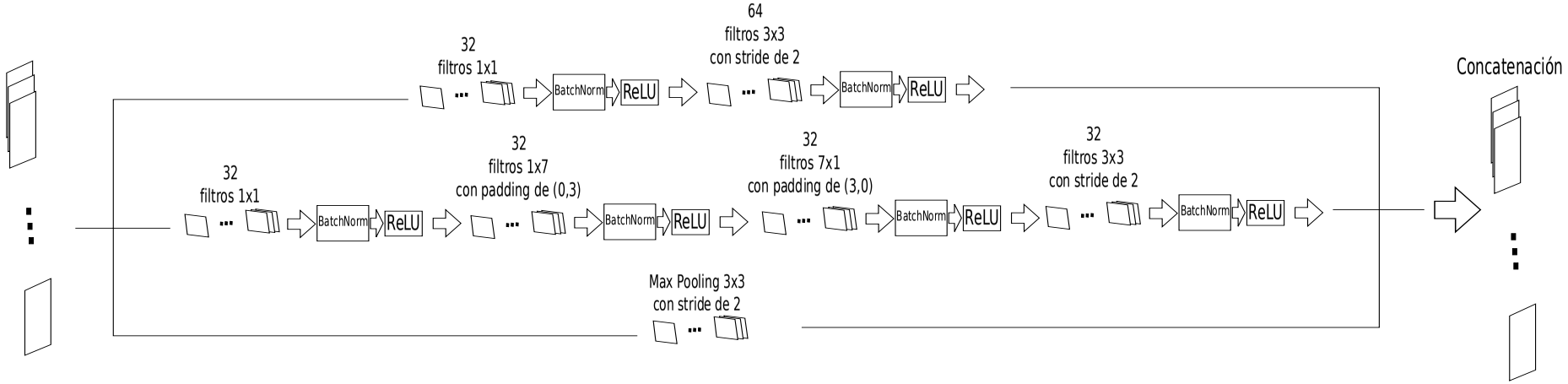
3.4. Función de Costo

Vamos a explicar en esta sección las funciones de costo utilizadas para cada modelo, acompañada de una breve fundamentación del motivo de su elección.

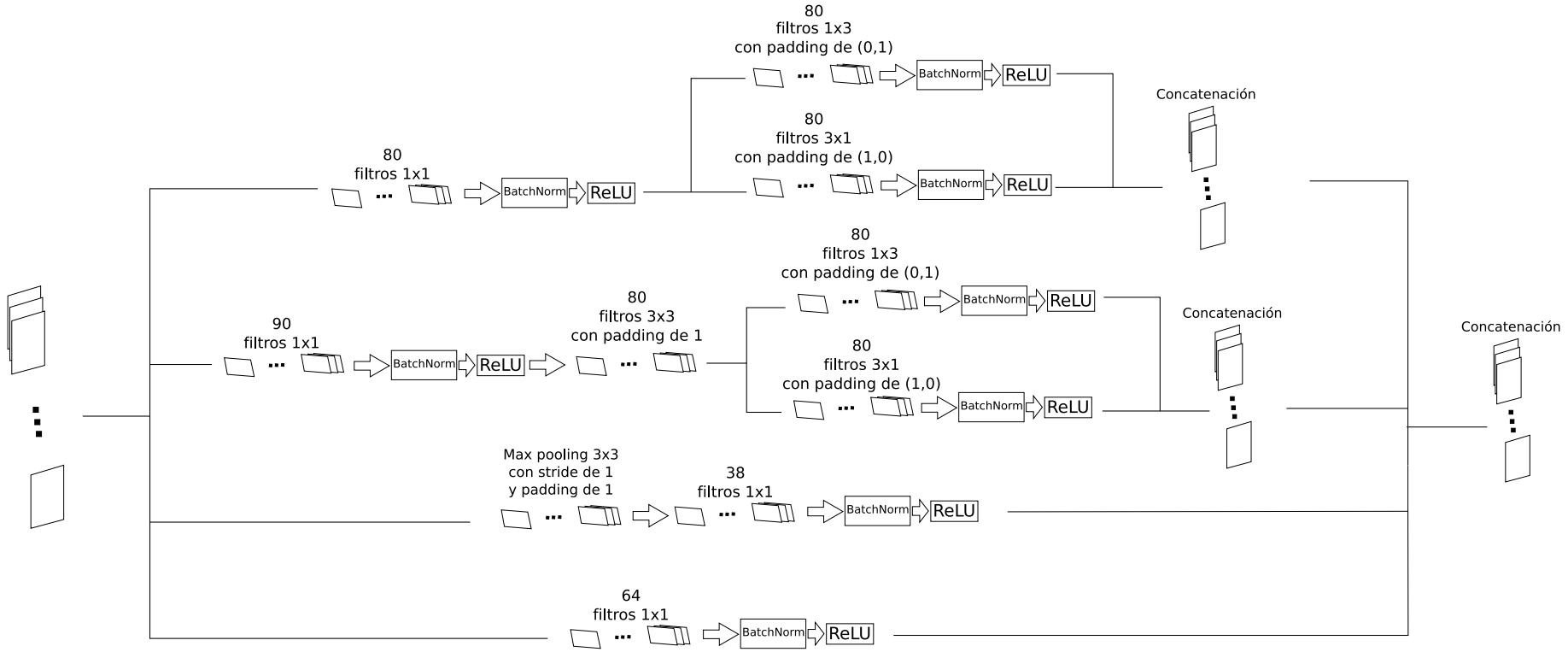
En los modelos 3.3.1 y 3.3.2, y en los autoencoders en general, se aborda el problema como una regresión, es decir, se está tratando de ajustar una función cuya



(A) Capas de reducción convolucional. El objetivo de estas capas es reducir la dimensión espacial de la imagen de entrada aprendiendo los parámetros útiles para la transmisión de información relevante a través de la red.

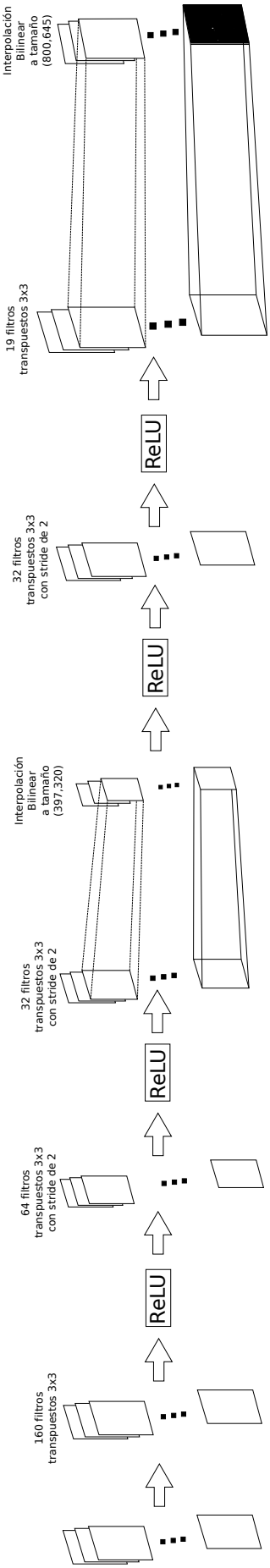


(B) Versión modificada de la capa InceptionD implementada en PyTorch. El objetivo de esta capa es la reducción de dimensión espacial mientras se combina información de distintos campos receptivos.



(C) Versión modificada de la capa InceptionE implementada en PyTorch. El objetivo de esta capa es combinar información de distintos campos receptivos.

FIGURA 3.3: Capas componentes del encoder en la arquitectura Autoencoder inception.



(A) decoder de la primera versión de la red Autoencoder Inception. la reducción de canales en la última capa convolucional transpuesta es de 32 a 19.

(B) decoder de la segunda versión de la red Autoencoder Inception. la reducción de canales en la última capa convolucional transpuesta es de 64 a 19.

FIGURA 3.4: decoders de las distintas versiones de la red Autoencoder Inception.

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 me permite pesar la clase positiva para realizar un intercambio entre *recall* y *precision*. 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 hiperparámetro a ajustar, que nos permitirá cambiar el modelo variando el hiperparámetro.

3.5. Selección del landmark

Una vez generados los mapas de probabilidad, es necesario elegir una técnica para determinar únicamente la posición del landmark. Existen varios métodos (como el utilizado en Ibragimov y col. (2014)) basados en la forma de las distribuciones de los landmarks, pero para estos modelos utilizaremos una metodología más simple, aplicando un filtro gaussiano al mapa de probabilidad (para filtrar posibles detecciones con alta probabilidad en zonas de baja probabilidad) y luego eligiendo el punto con mayor probabilidad después de aplicar el filtro.

3.6. Métricas

Para comparar modelos y determinar cual de ellos es el mejor, es necesario definir métricas que nos sirvan para cuantificar el desempeño de los mismos. En esta sección explicaremos cuales métricas se utilizan en este trabajo.

3.6.1. 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 o *sucess detection rate* p_z con presición menor a z mm se formula como (Wang y col. (2016)):

$$p_z = \frac{\#\{j : \|L_d(j) - L_r(j)\| < z\}}{\#\Omega} \times 100\,\%. \tag{3.2}$$

donde L_d, L_r representan la localización del landmark detectado y del landmark etiquetado, respectivamente, z denota la presición en la medición, en nuestro caso cuatro valores: $z = 2mm$, $2.5mm$, $3mm$ y $4mm$; $j \in \Omega$, y $\#\Omega$ representa el número de detecciones realizadas.

3.6.2. Error radial medio

El error radial 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 estandar o *standard deviation* (SD) asociada, se calculan como (Wang y col. (2016)):

$$MRE = \frac{\sum_{j=1}^{N} R_i}{N}, \qquad SD = \sqrt{\frac{\sum_{j=1}^{N} (R_i - MRE)^2}{N-1}} \tag{3.3}$$

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

Capítulo 4

Resultados

En este capítulo se presentan los resultados obtenidos con las arquitecturas de redes convolucionales presentadas en el capítulo anterior. Es importante destacar que estos resultados, obtenidos con arquitecturas originales, permiten alcanzar, hasta nuestro conocimiento actual, el mejor desempeño de la literatura en lo que respecta a la detección de puntos característicos en imágenes cefalométricas. En la primera sección presentaremos el problema de entrenamiento, en la segunda sección abordaremos la etapa evaluación de cada modelo considerado.

4.1. Entrenamiento

4.1.1. Transformaciones en el conjunto de datos

Antes de entrenar los modelos, existe una serie de transformaciones realizadas sobre el conjunto de datos, cada una con un objetivo específico.

En todos los modelos de autoencoders restringidos desarrollados, tanto el simple como las dos versiones inception, además de la reducción de la imagen mencionada en 3.1, realizamos una serie de transformaciones en los datos:

- Se realizan reescalados aleatorios de la imagen (entre el 98 % y 102 %)
- Se realizan traslaciones aleatorias horizontales y verticales a la imagen (de hasta 2 % de la imagen en cada dirección)
- Se realizan rotaciones aleatorias a la imagen (hasta 5°)
- Se normaliza el conjunto de imágenes para tener media 0 y varianza 1.

Los cambios de escala, rotación y traslación aleatoria de las imágenes, son realizadas con el motivo de aumentar artificialmente el conjunto de entrenamiento, para poder apreciar mejor las estructuras existentes en las imágenes, y reducir el sobreajuste. 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.

4.1.2. Hiperparámetros

En esta sección determinaremos los valores de los hiperparámetros de entrenamiento (aquellos parámetros del modelo que no se aprenden automáticamente), tales como la tasa de aprendizaje, el peso de regularización, el momento, el peso de la clase positiva, el tamaño del batch y el número de épocas de aprendizaje.

Los valores determinados previamente son:

- Tamaño del batch: 5
- Número de épocas: 250

La elección del tamaño del batch tiene que ver con la imposibilidad de cargar más imágenes en memoria de la GPU, mientras que el número de épocas se elige lo suficientemente grande como para asegurarnos de que la función de costo llegue a un mínimo.

Por otro lado, los valores del peso de la clase positiva p, la tasa de aprendizaje ϵ y el peso de regularización λ, son elegidos mediante una busqueda en grilla, tales que generen los mejores resultados en un conjunto de validación. Para reducir el costo computacional, la busqueda en grilla se hará primero con el peso p, con valores predefinidos de tasa $\epsilon = 10^{-3}$ y del peso $\lambda = 10^{-5}$. Luego de elegido el mejor valor para p, se realiza una busqueda conjunta de los valores de λ y ϵ, hasta obtener el modelo con mejores predicciones, basados en las métricas establecidas.

Búsqueda de hiperparámetros en grilla

- Autoencoder simple

  Realizamos la búsqueda en grilla para el modelo autoencoder simple restringido. Se procede a realizar la búsqueda en grilla del hiperparámetro p, para esto se buscará en un entorno del valor $p = 2500$, el cual demostró buen desempeño en pruebas preliminares. Los valores para la búsqueda son {2000, 2500, 3000, 4000, 5000, 6000}. Para realizar la búsqueda de hiperparámetros se divide aleatoriamente la muestra de entrenamiento en dos submuestras: una de entrenamiento, con un 80 % de los datos, y una de validación, con un 20 % de los datos. Los modelos se entrenan sobre la submuestra de entrenamiento, y se evalúan a través de alguna métrica su desempeño en el conjunto de validación. Para estos modelos utilizaremos como métrica el coeficiente de detecciones exitosas para 2 mm.

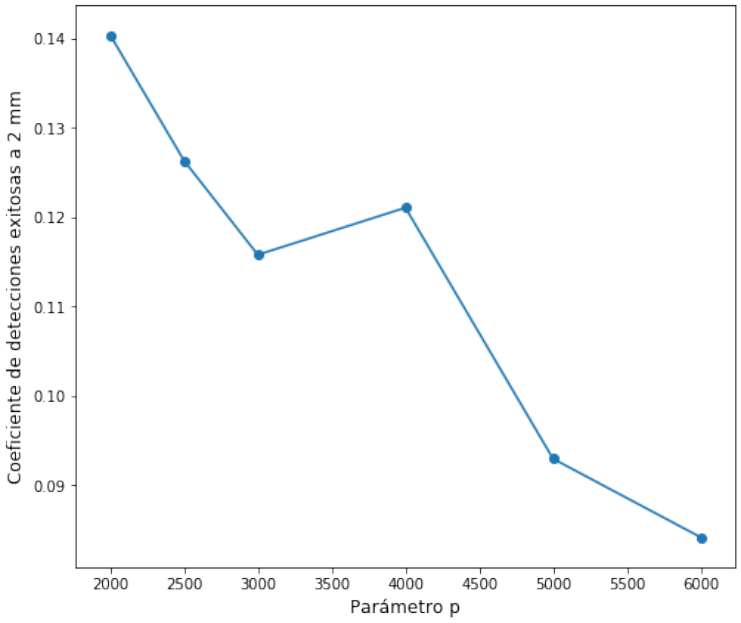


FIGURA 4.1: Resultados de la búsqueda en grilla sobre el hiperparámetro p para el modelo Autoencoder simple. El modelo que presenta el mejor resultado es $p = 2000$.

Se encuentra, con un valor de la métrica de $14,0\,\%$, que el modelo que presenta un mejor desempeño en el conjunto de validación es para un valor de $p = 2000$. De aquí en adelante se utilizará este valor para p en el modelo.

Ahora se realiza la búsqueda en grilla conjunta de los hiperparámetros ϵ y λ, ambos en el conjunto $\{10^{-3}, 10^{-4}, 10^{-5}\}$, valores usuales en la literatura para estos parámetros. La división de la muestra de entrenamiento y validación se realiza de la misma manera que para el hiperparámetro p, y se utiliza la misma métrica.

		tasa de aprendizaje ϵ:		
		$\epsilon = 10^{-3}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-5}$
peso de regularización λ :	$\lambda = 10^{-3}$	4.2 %	0.4 %	0 %
	$\lambda = 10^{-4}$	10.9 %	0 %	0 %
	$\lambda = 10^{-5}$	14.0 %	0.4 %	0 %

TABLA 4.1: Resultados de la búsqueda en grilla sobre los hiperparámetros ϵ y λ para el modelo Autoencoder simple.

Se encuentra, con un valor de la métrica de $14.0\,\%$, que el modelo que presenta un mejor desempeño en el conjunto de validación es para valores de ϵ y λ de

10^{-3} y 10^{-5}, respectivamente. De todas maneras, como los valores encontrados no son comparables a los valores de la literatura, se descarta este modelo y se continúa con las dos variantes del autoencoder inception, ya que asumimos que el problema de este modelo es la falta de complejidad y la localidad del mismo.

- Autoencoder Inception 32

  Realizamos la busqueda en grilla para la variante de 32 núcleos del modelo autoencoder inception.

  Se procede a realizar la búsqueda en grilla del hiperparámetro p, de la misma manera que en el autoencoder simple.

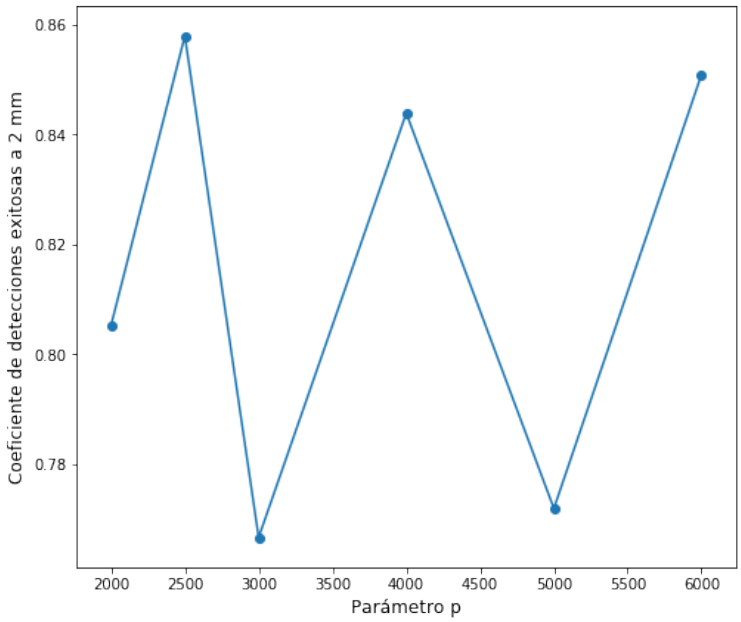


FIGURA 4.2: Resultados de la búsqueda en grilla sobre el hiperparámetro p para el modelo Autoencoder Inception 32. El modelo que presenta el mejor resultado es $p = 2500$.

Se encuentra, con un valor de la métrica de 85, 8 %, que el modelo que presenta un mejor desempeño en el conjunto de validación es para un valor de $p = 2500$. De aquí en adelante se utilizará este valor para p en el modelo.

Ahora se realiza la búsqueda en grilla conjunta de los hiperparámetros ϵ y λ, siguiendo el método explicado en el autoencoder simple.

peso de regularización λ :	tasa de aprendizaje ϵ: $\epsilon = 10^{-3}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-5}$
$\lambda = 10^{-3}$	5.6 %	5.6 %	4.4 %
$\lambda = 10^{-4}$	85.1 %	51.2 %	4.4 %
$\lambda = 10^{-5}$	85.8 %	74.9 %	4.2 %

TABLA 4.2: Resultados de la búsqueda en grilla sobre los hiperparámetros ϵ y λ para el modelo Autoencoder inception 32.

Se encuentra, con un valor de la métrica de 85.8 %, que el modelo que presenta un mejor desempeño en el conjunto de validación es para valores de ϵ y λ de 10^{-3} y 10^{-5}, respectivamente. Entonces, de aquí en adelante se utilizarán estos valores para el modelo.

- Autoencoder Inception 64

  Realizamos la busqueda en grilla para la variante de 64 núcleos del modelo autoencoder inception.

  Se procede a realizar la búsqueda en grilla del hiperparámetro p, de la misma manera que en el autoencoder simple.

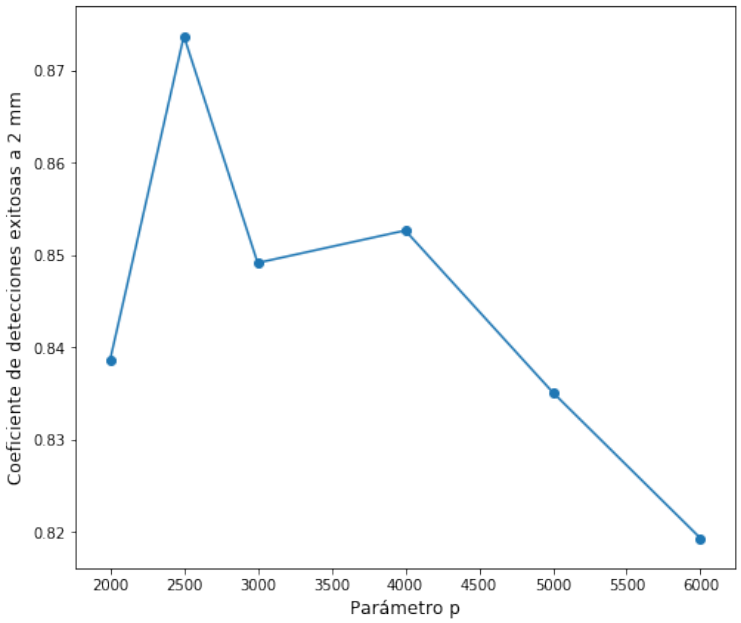


FIGURA 4.3: Resultados de la búsqueda en grilla sobre el hiperparámetro p para el modelo Autoencoder Inception 64. El modelo que presenta el mejor resultado es $p = 2500$.

Se encuentra, con un valor de la métrica de 87,37 %, que el modelo que presenta un mejor desempeño en el conjunto de validación es para un valor de $p = 2500$. De aquí en adelante se utilizará este valor para p en el modelo.

Ahora se realiza la búsqueda en grilla conjunta de los hiperparámetros ϵ y λ, de la misma manera que se realizó para los otros dos modelos.

peso de regularización λ :	tasa de aprendizaje ϵ: $\epsilon = 10^{-3}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-5}$
$\lambda = 10^{-3}$	5.6 %	7.7 %	4.4 %
$\lambda = 10^{-4}$	81.6 %	77.0 %	7.5 %
$\lambda = 10^{-5}$	87.4 %	81.1 %	7.5 %

TABLA 4.3: Resultados de la búsqueda en grilla sobre los hiperparámetros ϵ y λ para el modelo Autoencoder inception 32.

Se encuentra, con un valor de la métrica de 87.4 %, que el modelo que presenta un mejor desempeño en el conjunto de validación es para valores de ϵ y λ de 10^{-3} y 10^{-5}, respectivamente. Entonces, de aquí en adelante se utilizarán estos valores para el modelo.

4.2. Evaluación de los modelos

4.2.1. Conjuntos de evaluación

Como se mencionó en 3.1, existen dos conjuntos de evaluación, con 150 y 100 imágenes respectivamente. De los resultados de Wang y col. (2016) y Arik, Ibragimov y Xing (2017), podemos ver que esos modelos presentan un peor desempeño en las detecciones del segundo conjunto de evaluación, como se muestra en la figura 4.4.

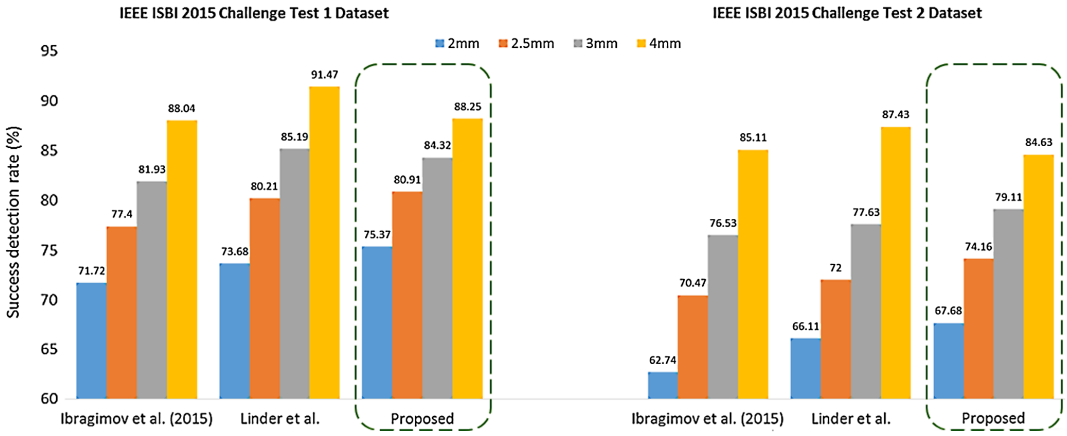


FIGURA 4.4: Comparación de los modelos de Wang y col. (2016) y Arik, Ibragimov y Xing (2017) en las métricas de coeficiente de detecciones exitosas para 2, 2.5, 3 y 4 mm. El modelo etiquetado como proposed es el perteneciente a este último. Imagen extraída de Arik, Ibragimov y Xing (2017).

4.2.2. Resultados

En esta sección presentaremos los resultados de los distintos modelos en las métricas presentadas en 3.6. La evaluación se realiza tanto en promedio sobre los 19 landmarks, como sobre cada landmark en particular.

TABLA 4.4: Coeficiente de Detecciones exitosas para cada landmark [%]. Modelo autoencoder inception 32.

	Conjunto de test 1				Conjunto de test 2			
	2 mm	2.5 mm	3 mm	4 mm	2 mm	2.5 mm	3 mm	4 mm
L1	97.33	97.33	98	98	98	98	98	98
L2	78.66	82.66	90.66	94.66	86	94	94	99
L3	86	92	96	98.66	38	62	81	94
L4	62.66	73.33	78	86	72	77	84	89
L5	59.33	72	82	89.33	80	89	96	98
L6	78	86.66	92	98.66	46	56	75	90
L7	90.66	96	96.66	99.33	99	100	100	100
L8	92	95.33	98	99.33	98	99	100	100
L9	89.33	95.33	98	98.66	99	99	100	100
L10	52	65.33	75.33	88.66	60	71	77	92
L11	91.33	95.33	96.66	98	91	96	98	98
L12	96.66	98	98.66	98.66	94	96	98	99
L13	85.33	96.66	98.66	99.33	10	38	64	97
L14	99.33	100	100	100	70	92	97	100
L15	89.33	94	95.33	98	89	94	97	100
L16	85.33	92	95.33	98	1	4	12	36
L17	94	97.33	98.66	98.66	88	93	97	98
L18	80.66	87.33	91.33	96	90	95	97	99
L19	69.33	78.66	86	93.33	81	89	92	98
promedio	83.82	89.22	92.91	96.39	73.16	81.16	87.21	93.95

TABLA 4.5: Coeficiente de Detecciones exitosas para cada landmark [%]. Modelo autoencoder inception 64.

	Conjunto de test 1				Conjunto de test 2			
	2 mm	2.5 mm	3 mm	4 mm	2 mm	2.5 mm	3 mm	4 mm
L1	98	98.66	98.66	98.66	99	99	99	99
L2	78.66	88	90	94.66	86	92	95	98
L3	80.66	93.33	95.33	99.33	36	62	79	96
L4	61.33	75.33	80.66	90	75	81	85	92
L5	64	78.66	86	94	64	75	87	94
L6	80	88.66	91.33	96.66	27	42	54	70
L7	87.33	94	96.66	99.33	100	100	100	100
L8	90.66	96	97.33	98.66	98	98	100	100
L9	86.66	90.66	96.66	98	98	100	100	100
L10	47.33	64	74	86.66	54	64	76	87
L11	92.66	97.33	98.66	98.66	94	96	96	98
L12	93.33	95.33	96.66	97.33	93	95	96	99
L13	88	96.66	98	99.33	12	42	67	98
L14	95.33	99.33	100	100	36	69	90	99
L15	88	95.33	97.33	98	89	96	98	99
L16	84	88.66	94.66	98.66	5	13	23	49
L17	92.66	96	97.33	98.66	89	94	96	97
L18	80.66	88.66	92.66	96.66	89	96	98	99
L19	69.33	81.33	84.66	90.66	76	84	87	91
promedio	82.04	89.79	92.98	96.52	69.47	78.84	85.58	92.89

Como podemos observar en las tablas 4.4 y 4.5, el comportamiento observado en la figura 4.4 se mantiene para nuestros modelos, presentando una mayor dificultad de detección en las imágenes del segundo conjunto de test. Si bien vemos que hay landmarks como el L10 que presentan gran dificultad de detección en ambos conjuntos, existen otros como el L3, L6, L13 y L16 que son detectados aceptablemente en el primer conjunto de test, pero no así en el segundo. Creemos que esto pueda deberse a que los conjuntos de entrenamiento y ambos conjuntos de test no hubiesen sido generados partiendo aleatoriamente un gran conjunto, sino que se hayan dejado a propósito imágenes más problemáticas en el segundo conjunto de test.

Vemos además que, al contrario de lo ocurrido en el conjunto de validación a la hora de calcular los hiperparámetros, el modelo que presenta mejor desempeño a 2 mm sobre ambos conjuntos de test es el autoencoder inception 32.

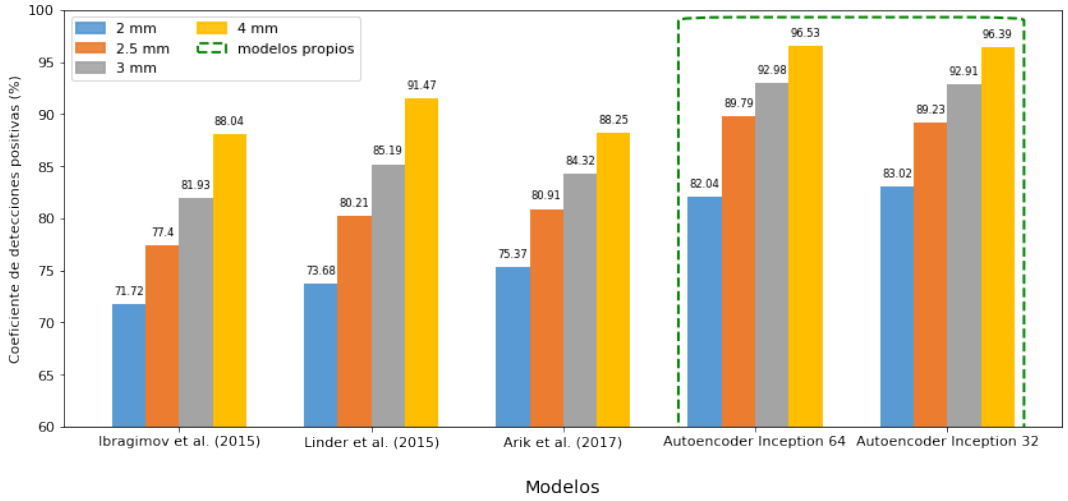


FIGURA 4.5: Resultados de los modelos propios comparados con los modelos de Ibragimov y Col. y Lindner y Cootes presentados en Wang y col. (2016) y el modelo de Arik, Ibragimov y Xing (2017) sobre el conjunto de test 1. Las métricas evaluadas son los coeficientes de detecciones positivas para 2, 2.5, 3 y 4 mm. En recuadro verde los modelos desarrollados en este trabajo.

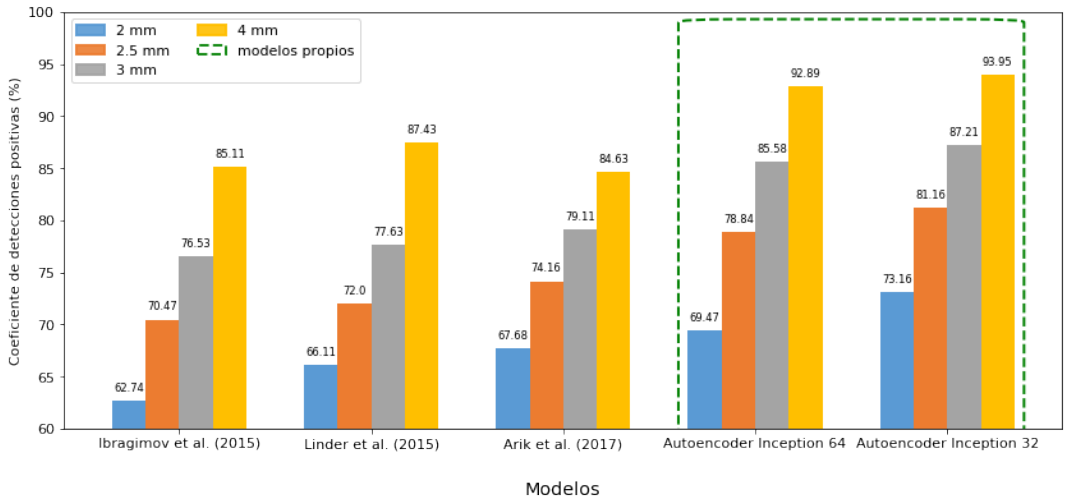


FIGURA 4.6: Resultados de los modelos propios comparados con los modelos de Ibragimov y Col. y Lindner y Cootes presentados en Wang y col. (2016) y el modelo de Arik, Ibragimov y Xing (2017) sobre el conjunto de test 2. Las métricas evaluadas son los coeficientes de detecciones positivas para 2, 2.5, 3 y 4 mm. En recuadro verde los modelos desarrollados en este trabajo.

Como se puede ver en las figuras 4.5 y 4.6, los modelos presentados superan en todas las métricas de coeficiente de detecciones exitosas a los modelos previos. La diferencia para 2 mm entre el mejor modelo y el modelo presentado en Arik, Ibragimov y Xing (2017) es de mas de 7 % para el conjunto de test1 y más de 5 % para el conjunto de test2, una diferencia considerable, sobre todo comparando a su modelo con los dos modelos previos.

Presentaremos ahora algunos ejemplos de detecciones para el modelo autoencoder inception 32, para hacer más ilustrativos los resultados.

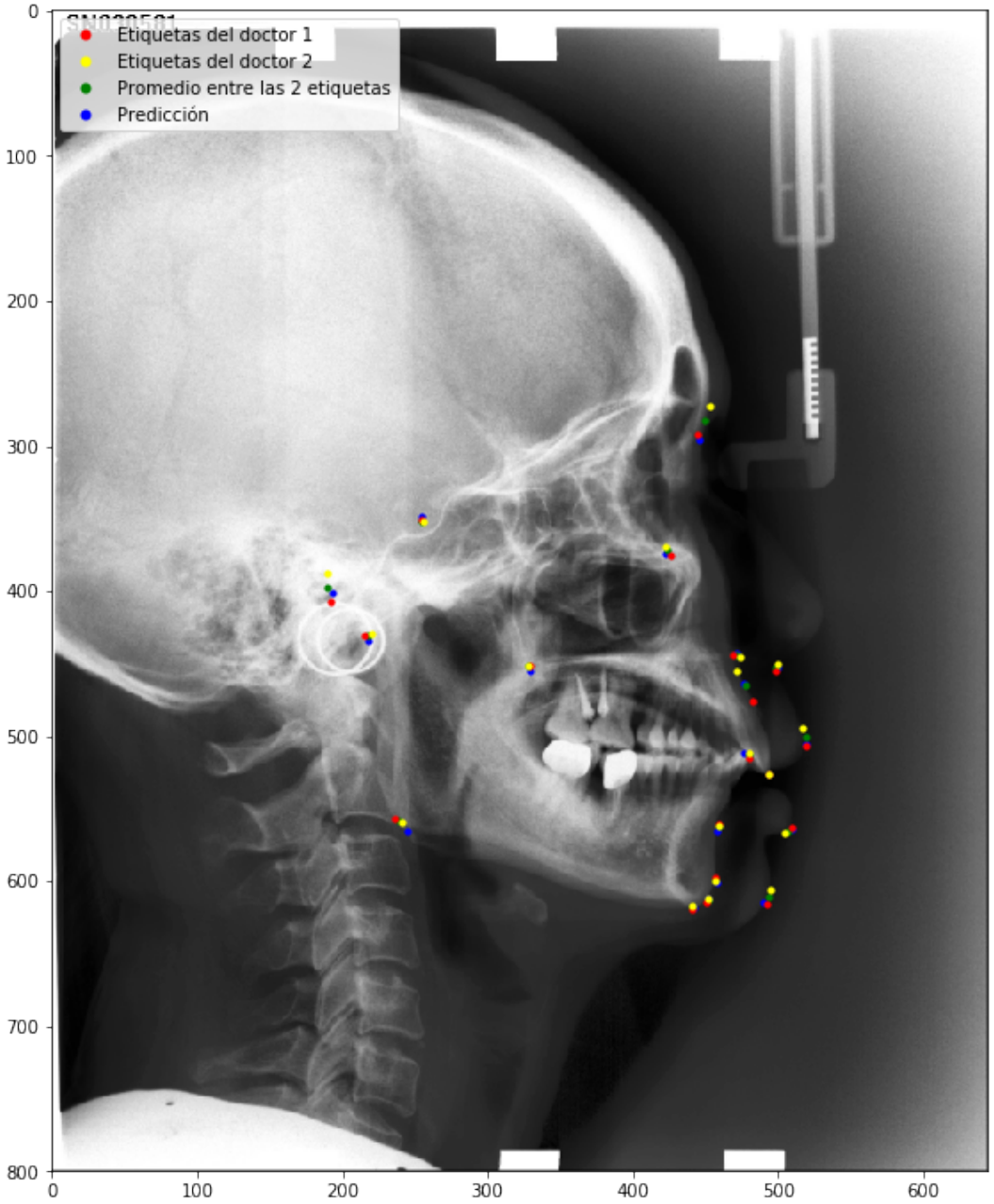


FIGURA 4.7: Detecciones y etiquetas sobre una imagen del conjunto de test 1. Recordar que los puntos considerados como verdaderos son el promedio entre las etiquetas de los dos doctores, marcados en verde.

Como se puede ver en la figura 4.7, existen landmarks en donde los dos doctores difieren considerablemente en sus etiquetas. Esto nos muestra que puede existir un umbral insuperable en la presición para los modelos, basados en que la demarcación de los puntos por parte de los doctores no es lo suficientemente precisa.

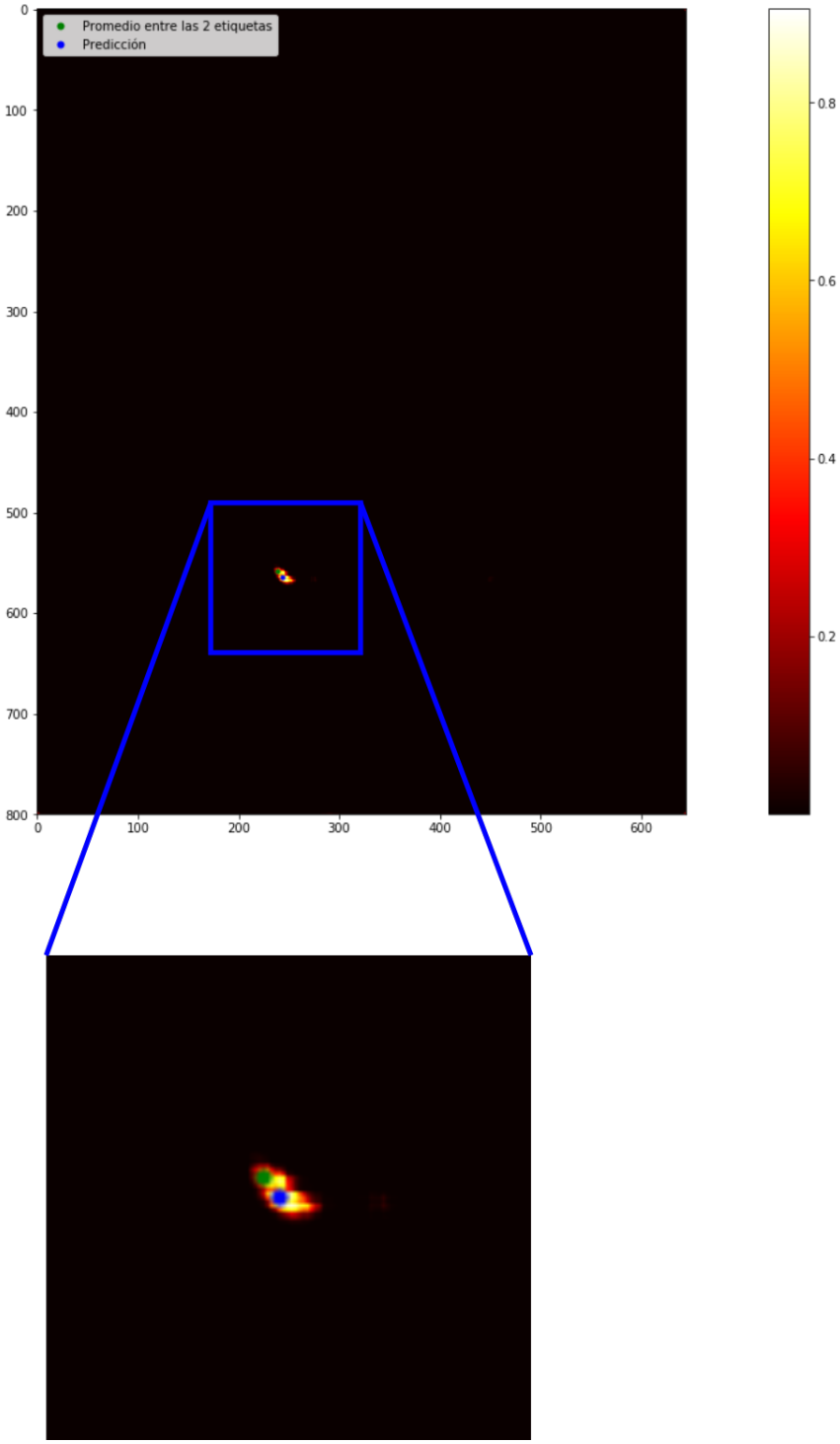


FIGURA 4.8: detecciones y etiquetas del landmark 10 sobre su respectivo mapa de probabilidad de una imagen del conjunto de test 1.

Si analizamos la figura 4.8, podemos notar que aunque la detección no coincide exactamente con la etiqueta, la posición de la etiqueta tiene una probabilidad considerablemente alta. En base a esto, podemos suponer que aplicando un modelo posterior basado en la forma de las distribuciones de landmarks sobre los mapas de probabilidad, similar al utilizado en Ibragimov y col. (2014), podríamos mejorar notoriamente las detecciones.

4.2.3. Desempeño a través de las épocas

Por último, analizaremos el aprendizaje del modelo a través de las épocas de aprendizaje, evaluando su capacidad de predicción sobre los conjuntos de entrenamiento y evaluación, sobre las métricas "coeficiente de detecciones exitosas a 2 mm (CDE-2mm)" y "error radial medio (ERM)". Como error se aplicó esta última métrica, y no la función de costo, ya que representa de mejor manera el desempeño del modelo a la hora de predecir la posición de los landmarks, que es la tarea específica a desarrollar.

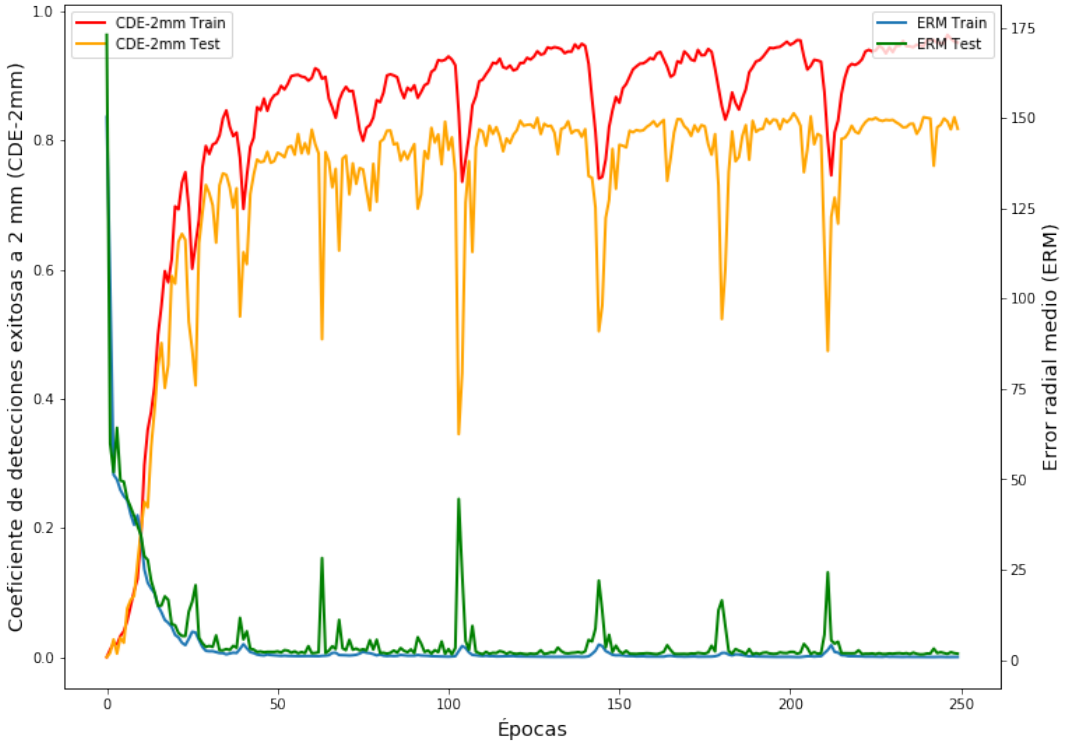


FIGURA 4.9: Coeficiente de detecciones exitosas a 2 mm y error radial medio en los conjuntos de entrenamiento y test 1 en función de las épocas de aprendizaje.

En la figura 4.9 se observa en general el comportamiento esperado para estas métricas sobre ambos conjuntos. El coeficiente de detecciones exitosas crece hasta estancarse para ambos conjuntos, el error radial medio decrece hasta estancarse, y sucede, como es de esperarse, que el modelo tiene un mejor desempeño medido en ambas métricas sobre el conjunto de entrenamiento comparado con el conjunto de evaluación. Sin embargo, es de destacar un comportamiento peculiar cada cierto número de épocas, en las cuales se observa una merma importante de desempeño tanto sobre el conjunto de entrenamiento como sobre el de evaluación. Este comportamiento no se podría explicar como un sobreajuste de los datos, ya que si así fuera sólo se apreciaría sobre el conjunto de evaluación, ni como un problema de subajuste, ya que en el resto de las épocas el modelo se desempeña bien sobre datos de entrenamiento y evaluación. Estimamos que este comportamiento viene asociado a

las operaciones de selección aleatoria del batch como a las transformaciones aleatorias realizadas sobre cada elemento del mísmo, considerando que puede ocurrir que, una vez alcanzado un desempeño bastante óptimo, la nueva forma de presentar los datos conjuntamente con las transformaciones aleatorias realizadas puedan llegar a desestabilizar el modelo, arrancándolo de un mínimo global y llevándolo a un mínimo local. De todas maneras, el comportamiento se corrige con más entrenamiento, y es correspondido en ambos conjuntos de datos, con lo cual se puede evitar quedar en estos puntos si se observa el aprendizaje sobre el conjunto de entrenamiento.

En este capítulo se entrenó y se evaluó el comportamiento de los distintos modelos propuestos en el capítulo anterior sobre conjuntos de entrenamiento y evaluación. En el capítulo siguiente se desarrollarán las conclusiones finales del presente trabajo, y se abordarán propuestas para el posterior mejoramiento de los modelos.

Capítulo 5

Conclusiones

En este trabajo final hemos estudiado el problema de la detección automática de puntos característicos, o landmarks, en imágenes odontológicas obtenidas a partir de técnicas de rayos X. En particular, hemos abordado el problema desde la perspectiva neuronal, utilizando redes neuronales convolucionales profundas. Es importante destacar que este problema odontológico es de particular interés desde el punto de vista de la salud, sobre todo en tratamientos de ortodoncia, ya que la identificación de estos puntos permite a los profesionales el diagnóstico, la definición de los tratamientos y la evaluación y eventual corrección de los mismos. Pero por otro lado, desde el punto de vista del aprendizaje automático, el problema de identificación de estos puntos plantea un desafío importante al mundo de la inteligencia artificial, pues su adecuada localización requiere, para los humanos expertos, la adquisición de habilidades muy específicas que son difíciles de adquirir y sobre todo, de transmitir. El grado de desafío planteado ha llevado incluso a que se realizaran competencias internacionales.

Hemos trabajado con este problema precisamente por su desafío en términos de procesamiento de imágenes, pero es importante decir que llegamos a él gracias a la interacción fructífera que hemos desarrollado en estos últimos meses con la empresa CEFMED (Cefalometría Médica) y en especial con su propietario, el Ing. Sebastián Salgado, con quien hemos generado una cooperación institucional virtuosa.

En los dos primeros capítulos hemos introducido al lector al estado del arte del uso de técnicas neuronales en el aprendizaje automático, las cuales han tenido en los últimos quince años un vigoroso resurgimiento, a partir de las llamadas técnicas profundas, las cuales básicamente han permitido construir y entrenar redes organizadas en un número grande de capas jerárquicas. En particular, después de varias semanas de estudios, exploraciones bibliográficas y diferentes intentos, se optó por el uso de redes neuronales convolucionales. En este sentido, hemos utilizado por primera vez para este problema modelos exclusivamente neuronales, pero además, hemos introducido el concepto de autoencoder restringido, en el cual la entrada es una imagen y la salida, otra imagen en la cual se caracterizan los píxeles buscados mediante un mapa de probabilidades. A esta idea original, que rápidamente mostró ser muy promisoria, se la enriqueció después con el agregado de capas inception.

Hemos comparado por una lado un autoencoder convolucionado y restringido

simple (sin inception), con dos autoenconders convolucionados y restringidos con capas inception, los cuales varían en la cantidad de canales en las capas ocultas del decodificador (32 y 64, respectivamente). Hemos mostrado que la introducción de las capas inception mejora notablemente el desempeño, aunque es necesario destacar que se agrega mucha profundidad, y por lo tanto, complejidad al problema. En particular hay un considerable aumento de parámetros, aunque se debe tener en cuenta que, de todas maneras, cualquier modelo completamente convolucional tiene una cantidad de parámetros muy inferior a un modelo equivalente completamente conectado, o con capas completamente conectadas. Sería provechoso comparar los modelos que utilizan capas inception, con un modelo convolucionado restringido simple de mayor profundidad y cantidad de parámetros, aunque se estima que el gran potencial de la capa inception en nuestros modelos no puede ser replicado mediante la simple secuencia de capas convolucionales simples, sin importar la cantidad de parámetros.

Se realizaron exploraciones de hiperparámetros para encontrar los mejores modelos, y se descartó el modelo autoencoder simple restringido, por no alcanzar valores comparables al estado del arte de la métrica **coeficiente de detecciones exitosas** para ninguna combinación de hiperparámetros. Se evaluaron los modelos restantes sobre dos conjuntos de evaluación, y se los comparó con los mejores modelos del estado del arte. Nuestros modelos mostraron una gran presición en detección de posiciones de landmarks, superando hasta en más de 7 puntos porcentuales los desempeños de los mejores modelos en la literatura sobre la métrica usual.

Este trabajo abre las puertas a posteriores mejoras y complementos. En la parte de mejora, creemos que se debe aumentar el tamaño del conjunto de datos, ya que en problemas comparables en dificultad, los tamaños utilizados suelen ser varios órdenes de magnitud mayores. Otra posibilidad por abordar es la aplicación de modelos basados en formas, como el utilizado en Ibragimov y col. (2014). En lo que respecta a los estudios complementarios, sería muy importante poder utilizar los autoencoders convolucionados y restringidos con capas inception al problema de detección de los bordes blandes de radiografías en general, y a las cefalometrías en particular.

Bibliografía

Ahmed, Hosameldin, M L. Dennis Wong y Asoke Nandi (ene. de 2018). «Intelligent condition monitoring method for bearing faults from highly compressed measurements using sparse over-complete features». En: *Mechanical Systems and Signal Processing* 99, págs. 459-477. DOI: 10.1016/j.ymssp.2017.06.027.

Arcieri, María José y col. (2016). «¿ Es aplicable el Cefalograma de Ricketts en diferentes poblaciones?» En: *Actas odontológicas* 10.2, págs. 12-18.

Arik, Sercan Ö, Bulat Ibragimov y Lei Xing (2017). «Fully automated quantitative cephalometry using convolutional neural networks». En: *Journal of Medical Imaging* 4.1, pág. 014501.

Cardillo, John y Maher A Sid-Ahmed (1994). «An image processing system for locating craniofacial landmarks». En: *IEEE transactions on medical imaging* 13.2, págs. 275-289.

Dumoulin, Vincent y Francesco Visin (2016). «A guide to convolution arithmetic for deep learning». En: *arXiv e-prints*, arXiv:1603.07285, arXiv:1603.07285. arXiv: 1603.07285 [stat.ML].

El-Feghi, Idris, Maher A Sid-Ahmed y Majid Ahmadi (2004). «Automatic localization of craniofacial landmarks for assisted cephalometry». En: *Pattern Recognition* 37.3, págs. 609-621.

Guarnieri, Ricardo A, Enio B Pereira y Sin Chan Chou (2006). «Solar radiation forecast using artificial neural networks in South Brazil». En: *Proceedings of the 8th ICSHMO*, págs. 24-28.

Hertz, J y col. (ene. de 1991). «Introduction To The Theory Of Neural Computation». En: vol. 44. Cap. 1,5. DOI: 10.1063/1.2810360.

Hopfield, J J (1982). «Neural networks and physical systems with emergent collective computational abilities». En: *Proceedings of the National Academy of Sciences* 79.8, págs. 2554-2558. ISSN: 0027-8424. DOI: 10.1073/pnas.79.8.2554. eprint: https://www.pnas.org/content/79/8/2554.full.pdf. URL: https://www.pnas.org/content/79/8/2554.

Ibragimov, Bulat y col. (2014). «Automatic cephalometric X-ray landmark detection by applying game theory and random forests». En: *Proc. ISBI Int. Symp. on Biomedical Imaging*.

Kingma, Diederik P. y Jimmy Ba (2014). «Adam: A Method for Stochastic Optimization». En: *CoRR* abs/1412.6980. arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980.

Lindner, Claudia y col. (2015). «Robust and accurate shape model matching using random forest regression-voting». En: *IEEE transactions on pattern analysis and machine intelligence* 37.9, págs. 1862-1874.

Luo, Wenjie y col. (2016). «Understanding the Effective Receptive Field in Deep Convolutional Neural Networks». En: *Advances in Neural Information Processing Systems 29*. Ed. por D. D. Lee y col. Curran Associates, Inc., págs. 4898-4906. URL: http://papers.nips.cc/paper/6203-understanding-the-effective-receptive-field-in-deep-convolutional-neural-networks.pdf.

Mitchell, Tom (1997). *Machine Learning*. New York: McGraw-Hill. ISBN: 978-0-07-042807-2.

Ricketts, Robert M (1982). *Orthodontic Diagnosis and Planning:–Their roles in preventive and rehabilitative dentistry*. Vol. 2. Rocky Mountain/Orthodontics.

Rosenblatt, Frank (1961). *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. Inf. téc. CORNELL AERONAUTICAL LAB INC BUFFALO NY.

Szegedy, C. y col. (jun. de 2015). «Going deeper with convolutions». En: págs. 1-9. ISSN: 1063-6919. DOI: 10.1109/CVPR.2015.7298594.

Torre, Ana y col. (ene. de 2015). «Prediction of compression strength of high performance concrete using artificial neural networks». En: *Journal of Physics: Conference Series* 582. DOI: 10.1088/1742-6596/582/1/012010.

Vellini-Ferreira, Flavio (2002). «Ortodoncia: Diagnóstico Y Planificación Clínica». En: *Editorial Artes Médicas*, págs. 313-314.

Wang, Ching-Wei y col. (2015). «Evaluation and comparison of anatomical landmark detection methods for cephalometric x-ray images: a grand challenge». En: *IEEE transactions on medical imaging* 34.9, págs. 1890-1900.

Wang, Ching-Wei y col. (2016). «A benchmark for comparison of dental radiography analysis algorithms». En: *Medical image analysis* 31, págs. 63-76.

Yue, Weining y col. (2006). «Automated 2-D cephalometric analysis on X-ray images by a model-based approach». En: *IEEE transactions on biomedical engineering* 53.8, págs. 1615-1623.