UNIVERSIDAD NACIONAL DEL SUR

TESIS DE DOCTOR EN CIENCIAS DE LA COMPUTACIÓN

Expansión de la Capacidad de los Filtros Convolucionales en Redes Neuronales

Juan Ignacio Larregui

BAHÍA BLANCA ARGENTINA

2020

Esta página intencionalmente en blanco.

Prefacio

Esta Tesis se presenta como parte de los requisitos para optar al grado Académico de Doctor en Ciencias de la Computación, de la Universidad Nacional del Sur y no ha sido presentada previamente para la obtención de otro título en esta Universidad u otra. La misma contiene los resultados obtenidos en investigaciones llevadas a cabo en el ámbito del Departamento de Ciencias e Ingeniería de la Computación durante el período comprendido entre el 25/11/2014 y el 21/11/2020, bajo la dirección de la Dra. Silvia Mabel Castro.

Juan Ignacio Larregui

UNIVERSIDAD NACIONAL DEL SUR

Secretaría General de Posgrado y Educación Continua

La presente tesis ha sido aprobada el ..../..../..... , mereciendo la calificación de .....(.....)

Esta página intencionalmente en blanco.

Agradecimientos

En primer lugar, quiero agradecer a mi Directora de Tesis, Dra. Silvia M. Castro, no solo por darme un lugar en su grupo de investigación sino por brindarme la libertad necesaria para encontrar mi propio camino dentro de las enormes posibilidades que ofrece nuestra disciplina. Lo recorrido durante este período impactó e impactará por siempre mi futuro de formas imposibles de predecir.

En segundo lugar, a Juan A. Biondi, por introducirme al Deep Learning, mostrarme los avances hasta el momento y hacerme pensar en adoptarlo como mi área principal de desarrollo.

En tercer lugar, al comité evaluador, por sus detalladas y fructíferas devoluciones, y por la curiosidad demostrada durante la defensa de esta tesis.

En cuarto lugar, a mis compañeros del VyGLab, por el ambiente de trabajo y compañerismo sincero mostrado durante todos estos años, los cuales afortunadamente trascendieron lo meramente profesional.

Por último, agradezco al Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET) y a la Universidad Nacional del Sur, instituciones a las cuales tuve el honor de pertenecer como becario, estudiante y docente.

Dedico esta Tesis a mis viejos, Marisa y Raúl, junto al agradecimiento más grande, no solo por permitirme estudiar, lo cual requiere un esfuerzo inmenso para cualquier familia laburante de este país, sino también por dejarme elegir mi camino con total libertad y fomentar mi curiosidad intelectual desde la más temprana edad. Los quiero mucho.

Esta página intencionalmente en blanco.

Resumen

En los últimos años el campo de la Visión Artificial ha experimentado un crecimiento acelerado con el éxito de las Redes Neuronales Artificiales y el Aprendizaje Profundo. La cantidad de datos etiquetados que se han relevado, las mejoras en hardware especializado y las importantes modificaciones introducidas en los algoritmos tradicionales surgidos en la segunda mitad del siglo pasado han posibilitado el avance en problemas complejos que parecían imposibles de abordar pocos años atrás.

En particular, las Redes Neuronales Convolucionales se han convertido en el modelo más popular dentro de este campo de las Ciencias de la Computación. A lo largo de la década del 2010, los trabajos que avanzaron el estado del arte en los diferentes problemas de la Visión Artificial han incluido casi exclusivamente redes de este tipo. Sin embargo, algunos componentes de las Redes Convolucionales han mantenido sus estructuras y definiciones originales. Este es el caso de los filtros convolucionales, los cuales han mantenido su estructura geométrica estática en las últimas décadas.

El objetivo general de esta tesis es explorar las limitaciones inherentes a la estructura tradicional de los filtros convolucionales, proponiendo nuevas definiciones y operaciones para superar las mismas. En esta línea, se presenta una generalización de la definición de los filtros convolucionales, extendiendo el concepto de *dilatación* de los mismos a intervalos continuos sobre las dimensiones espaciales. Adicionalmente, se presenta una nueva definición de la Convolución Dilatada para permitir comportamientos dinámicos durante el proceso de entrenamiento. Basadas en las definiciones introducidas, se proponen las nuevas operaciones de Convolución de Dilatación Adaptativa y Convolución de Dilatación Aleatoria. La primera introduce a las redes convolucionales la capacidad de optimizar la dilatación de los filtros de acuerdo a los datos de entrada, de manera de adaptarse dinámicamente a los cambios semánticos y geométricos presentes en las diferentes escenas. La segunda permite explorar la utilización de filtros de dilataciones aleatorias para simular

transformaciones de escala, con el objetivo de aumentar la invariancia a escala de una red convolucional, una de sus limitaciones más conocidas.

Finalmente, se definieron casos de estudio para Clasificación de Imágenes y Segmentación Semántica, de manera de obtener métricas cuantitativas que permitan evaluar las propuestas realizadas. Se realizaron múltiples entrenamientos de diferentes arquitecturas y configuraciones para redes conocidas en la literatura, mostrando resultados favorables con la inclusión de las operaciones propuestas. Más aún, el diseño de estas es modular, por lo que pueden ser incluidas en arquitecturas arbitrarias.

Abstract

In the last years, the field of Computer Vision has seen incredible success through the adoption of Artificial Neural Networks and Deep Learning. The amount of labeled data, the improvements in specialized hardware, and further development in the traditional algorithms, have enabled advances in complex problems that seemed impossible to approach a few years before.

In particular, these networks have become the most popular models within this field of Computer Sciences. Throughout the last decade, the state-of-the-art research in the different Computer Vision problems had almost exclusively included this type of model. However, the structure of some components of Convolutional Networks has remained almost unaffected. This is the case with convolutional filters, which have kept their original geometric structure in the last decades.

The overall goal of this thesis is to explore the limitations inherent to the traditional structure of the convolutional filters, introducing new definitions and operations to overcome them. In this context, a generalization of the definition of convolutional filters is presented, extending the concept of dilation to continuous intervals in the spatial dimensions. Additionally, a new definition for the Dilated or Atrous Convolution is proposed, which enables dynamic behaviors in the dilation of the filters during the training process. Based on these new definitions, two new operations are presented: the Adaptive Dilation Convolution and the Random Dilation Convolution. The first one introduces the capacity for Convolutional Networks to optimize the dilation of the filters according to the input data, dynamically adapting to the semantic and geometric differences found across scenes. The second, enables the exploration of random dilations to simulate different scale transformations in the data, aiming to increase the scale invariance of these networks, one of their known limitations.

Finally, different study cases were defined for Image Classification and Semantic Seg-

mentation, in order to evaluate the introduced operations using quantitative metrics. Several training experiments were performed, using different architectures and configurations for renowned networks, showing positive results during the inclusion of the proposed operations. Moreover, their design is modular, enabling them to be included in arbitrary architectures.

Certifico que fueron incluidos los cambios y correcciones sugeridos por los jurados.

Firma del Director

Esta página intencionalmente en blanco.

Abreviaturas

ANNs Redes Neuronales Artificiales

ASPP Atrous Spatial Pyramid Pooling

CDC Convolución de Dilatación Continua

CDD Convolución de Dilatación Dinámica

CNNs Redes Neuronales Convolucionales

SGD Stochastic Gradient Descent

Esta página intencionalmente en blanco.

Índice general

Esta página intencionalmente en blanco.

Índice de figuras

Capítulo 1

Introducción

> "What I cannot create, I do not understand"
>
> — Richard Feynman

Gran parte de la comunidad científica ha adoptado esta famosa cita del físico teórico Richard Feynman como el enfoque adecuado ante uno de los problemas abiertos más grandes de la ciencia: la búsqueda de aquello que constituye la **Inteligencia**. Desde los primeros pasos dados por pioneros como Alan Turing a mediados del siglo pasado [Tur48, Tur50], el descubrimiento y desarrollo de algoritmos inteligentes ocupa gran parte de la producción científica actual en lo que es, para muchos, el camino indicado para entender uno de los conceptos más complejos conocidos por la humanidad.

Cuando nos referimos coloquialmente al procesamiento inteligente de información realizado por humanos, solemos pensar en tareas cognitivas de alto nivel, incluyendo el procesamiento racional como una de sus características distintivas. Sin embargo, es común pasar por alto comportamientos inteligentes en tareas que consideramos sencillas donde no se involucra el raciocinio, las cuales pueden tener como resultado respuestas motrices reflejas o instantáneas. Desde el punto de vista del procesamiento de información, resulta claro que el reconocimiento de patrones involucrado, así como la asociación de diferentes tipos de información sensorial y la recuperación de información del pasado constituyen de la misma manera un procesamiento inteligente.

La respuesta ante estímulos visuales es un claro ejemplo de esto. Cada segundo procesamos una enorme cantidad de información visual, realizando a su vez una interpretación semántica y física de la misma de acuerdo al modelo del mundo que formamos con la

experiencia pasada. Desde el punto de vista de las Ciencias de la Computación, estos procesos han sido históricamente difíciles de automatizar por la enorme variabilidad que podemos encontrar en la información sensorial de entrada, resultando en la imposibilidad de diseñar algoritmos que busquen específicamente por determinados patrones y que sean capaces de generalizar su comportamiento ante nuevos datos. Así, los enfoques en los que debamos diseñar o conocer los patrones esperados ante la distribución probabilística de los datos de entrada no parecen ser viables y escalables, por lo que debemos explorar enfoques en los que dichos patrones puedan ser aprendidos y reconocidos a través de la experiencia, de manera similar a lo que sucede con el procesamiento de la información que realizan los organismos biológicos.

Como parte de la exploración científica y bajo estas conclusiones surge el **Conexionismo** y las **Redes Neuronales Artificiales** (ANNs, por sus siglas en inglés) inspiradas en la biología de la corteza cerebral, postulando al procesamiento inteligente de información como un fenómeno emergente en redes de esta naturaleza. Como trabajos fundacionales, McCulloch y Pitts proponen en 1943 el primer modelo computacional de *neurona artificial* [MP43], mientras que Frank Rosenblatt publica en 1958 el algoritmo del **Perceptron** [Ros58], un modelo que funda las bases de las Redes Neuronales Artificiales que se desarrollarían hasta nuestros días.

El Conexionismo pasó por períodos de desinterés en la comunidad científica, principalmente durante la década de 1970 y la primera mitad de los años 90. Sin embargo, incluso durante estos períodos, se publicaron artículos que marcaron un antes y un después en los enfoques ante problemas complejos relacionados con la búsqueda de la Inteligencia Artificial. La década de 2010 trajo consigo el punto más alto de relevancia del Conexionismo, con la explosión de técnicas como el Aprendizaje Profundo [LBH15] y el Aprendizaje por Refuerzo [SHM$^+$16], aventurando a muchos a pensar que el diseño de una Inteligencia Artificial General (una inteligencia comparable a la humana) es un hito alcanzable en los próximos años. Solo el tiempo podrá hacer un juicio de valor sobre estos postulados.

Este trabajo de tesis se enmarca dentro del área de la Visión Artificial, la cual se interesa en algoritmos que puedan interpretar la información visual de manera *inteligente*, de forma comparable a la interpretación que realizamos los humanos. El enfoque tomado es uno puramente *conexionista*, explorando un tipo particular de Redes Neuronales Artificiales denominadas **Redes Neuronales Convolucionales**.

1.1. Contexto

Durante las décadas de 1950 y 1960, Hubel y Wiesel publican una serie de artículos con observaciones del procesamiento de la información visual en ciertos mamíferos y la respuesta neuronal ante cambios en los estímulos sensoriales [HW59, HW62, HW68], por los cuales recibirían en 1981 el Premio Nobel de Fisiología y Medicina. Estas observaciones inspiran a Kunihiko Fukushima a publicar en 1980 un modelo de Red Neuronal Artificial de múltiples capas denominado **Neocognitron** [Fuk80, Fuk88], el cual logra aprender de manera robusta patrones visuales. Este modelo es el predecesor directo de las Redes Neuronales Convolucionales. Cinco años más tarde, Rumelhart *et al.* desarrollarían y popularizarían la técnica de **backpropagation** como algoritmo de aprendizaje de los parámetros de las ANNs, técnica de aprendizaje utilizada hasta la actualidad, mostrando resultados en reconocimiento de formas y predicción de palabras en lenguaje natural [RHW85, RHW86]. En 1989, Yann LeCun implementaría la primera Red Neuronal Convolucional entrenada mediante el algoritmo de *backpropagation*, logrando reconocer imágenes de dígitos escritos a mano [LBD+89b]. Tanto Yann LeCun como Geoffrey Hinton, partícipe de los trabajos en *backpropagation* junto a Rumelhart, se mantendrían activos en el área realizando notables contribuciones en los años venideros y convirtiéndose en dos de las figuras fundamentales para el desarrollo y éxito del Aprendizaje Profundo en los últimos años [LBD+89a, LDS89, LB+95, LBBH98, LeC98, SEZ+13, Hin90, HDFN95, HS06, MH08, KH+09, NH10, SHK+14].

En particular, fue Hinton el primer autor del artículo del año 2006 que suele indicarse como el iniciador del Aprendizaje Profundo [HOT06]. En este trabajo se propone una disposición en serie de múltiples modelos conocidos como *Restricted Boltzmann Machines*, siendo estos entrenados de a uno por vez y de manera consecutiva, mostrando la posibilidad de entrenar ANNs profundas de múltiples capas.

Hinton también participaría, junto a Alex Krizhevsky e Ilya Sutskever, del trabajo que marca el inicio de la explosión que vio el Aprendizaje Profundo en la década de 2010. En este se presenta la Red Neuronal Convolucional **AlexNet** [KSH12], ganando en 2012 el desafío de Clasificación de Imágenes del conjunto de datos ImageNet [DDS+09], siendo esta la primera oportunidad en la que una ANNs obtuvo resultados del estado del arte en Visión Artificial para problemas de esta magnitud. Más aún, la exactitud de esta red en la clasificación superó por más del 10 % al algoritmo ubicado en el segundo

puesto, lo cual marcó un cambio disruptivo en la forma de atacar los problemas de Visión Artificial. A partir de este momento, los algoritmos que empujarían el estado del arte durante los siguientes años serían casi exclusivamente Redes Neuronales Convolucionales con diferentes variantes. Los problemas atacados incluyen **Clasificación de Imágenes** [SZ14, SLJ+15, HZRS16a, SVI+16, XGD+17, TVDJ20], **Clasificación de Videos** [KTS+14, WGH+20], **Detección de Objetos** [GDDM14, RHGS15, RDGF16, TPL20], **Segmentación Semántica** [LSD15, RFB15, BKC17, CZP+18, ZWZ+20], **Segmentación por Instancias** [HGDG17, LWW+20, DLJ+20], **Segmentación Panóptica** [KHG+19, KGHD19, WZG+20, MV20], **Estimación de Pose** [WRKS16, KCQL18, YSWJ19] y **Descripción de Imágenes** [KFF15, WHZS20, ZPZ+20].

En los últimos años se han visto esfuerzos por otorgar mayor capacidad a las CNNs manteniendo el número de parámetros que conforman el modelo. Adicionalmente, se busca que las redes sean invariantes o equivariantes ante ciertas transformaciones que sabemos ocurren en las imágenes naturales, de manera de capturar más precisamente la distribución probabilística de los datos de entrada, considerando que siempre contaremos con un número limitado de muestras frente al número infinito de imágenes posibles. Entre estos esfuerzos, e inspirados en técnicas aplicadas en Procesamiento de Imágenes y de Señales [Mal99, FGMR09], se han introducido variantes que buscan flexibilizar la arquitectura de la red y, en algunos casos, el filtro convolucional en sí mismo [PKS14, PKS15, CPK+14, DQX+17].

En este contexto, se explora en esta tesis la flexibilización de la estructura geométrica de los filtros convolucionales con el objetivo de que una capa convolucional pueda aprender la forma óptima de utilizar el campo receptivo de cada filtro, a diferencia de la estructura fija que se encuentra en las arquitecturas tradicionales.

1.2. Objetivo General y Aportes

El objetivo general de esta tesis consiste en explorar la expansión de la capacidad de los filtros convolucionales utilizados tradicionalmente buscando superar las limitaciones dadas por su estructura geométrica fija. Para ello, se proponen generalizaciones a la definición de Convolución Dilatada, así como dos variantes a la capa convolucional utilizada en Redes Neuronales Convolucionales basadas en estas generalizaciones.

Para la evaluación de las propuestas realizadas y la obtención de resultados comparativos, fue necesaria la implementación de ambas variantes, las cuales pueden ser incluidas como una capa o módulo en redes existentes.

En resumen, los aportes de este trabajo son:

1. **Generalización de la Convolución Dilatada a valores continuos**. La Convolución Dilatada tradicional es una operación discreta en las tasas de dilatación empleadas, es decir, está definida solo para valores enteros, los cuales son interpretados como tasas de dilatación del filtro convolucional. Se define una extensión de la operación sobre valores continuos denominada Convolución de Dilatación Continua (CDC).

2. **Definición de la Convolución de Dilatación Dinámica (CDD)**. Una capa de Convolución Dilatada tradicional utiliza un único valor estático como tasa de dilatación para todos los filtros. Se propone su extensión mediante un mapa de tasas de dilatación que permita transformar el filtro en cada posición espacial de manera independiente. Más aún, un mapa de dilataciones puede ser modificado entre ejemplos de entrenamiento.

3. **Propuesta de la Capa de Convolución de Dilatación Adaptativa**. Se define un módulo para redes convolucionales que utiliza la Convolución de Dilatación Dinámica, donde el mapa de tasas de dilatación es predicho por el mismo módulo a partir de su entrada. Para que esto sea posible en la práctica, el módulo aprende durante el entrenamiento convencional a predecir las tasas óptimas dentro de un rango preestablecido.

4. **Propuesta de la Capa de Convolución de Dilatación Aleatoria**. En búsqueda de una mayor invariancia ante transformaciones de escala en la entrada, se explora la utilización de tasas de dilatación aleatorias para los filtros de un módulo durante la totalidad del proceso de entrenamiento, buscando simular una mayor variabilidad en la escala de los objetos presentes en las muestras sin la necesidad de modificar el conjunto de datos.

5. **Desarrollo e implementación de ambas capas**. Se implementaron los módulos propuestos para su inclusión en arquitecturas de redes existentes.

6. **Comparación de las propuestas frente a arquitecturas de filtros convolucionales de estructura geométrica fija**. Se definieron casos de estudio y se realizaron entrenamientos comparativos introduciendo los módulos propuestos en arquitecturas conocidas para los problemas de Clasificación de Imágenes y Segmentación Semántica.

1.3. Estructura de la Tesis

El presente trabajo de tesis incluye los conceptos relevantes de manera que su lectura sea autocontenida. A continuación se describe su estructura:

- **Capítulo 1 Introducción**. En este capítulo se introducen los objetivos, el marco histórico general y el contexto particular en el cual se desarrolla este trabajo de investigación, así como las contribuciones del mismo.

- **Capítulo 2 Aprendizaje Profundo y Visión Artificial**. En este capítulo se realiza una introducción teórica del Aprendizaje Profundo así como la definición de los conceptos relevantes para la lectura de este trabajo.

- **Capítulo 3 Expansión de los Filtros Convolucionales**. En este capítulo se incluyen las propuestas realizadas así como el trabajo relacionado. Se presentan las limitaciones de la estructura geométrica fija de los filtros convolucionales. Se define la Convolución de Dilatación Dinámica como extensión de la Convolución Dilatada tradicional a partir de su generalización a valores continuos. Se proponen y definen formalmente las capas de Convolución de Dilatación Adaptativa y Aleatoria.

- **Capítulo 4 Evaluación y Casos de Estudio**. En este capítulo se exponen las evaluaciones de las propuestas realizadas en casos de estudio comparativos para los problemas de Clasificación de Imágenes y Segmentación Semántica, incluyendo las módulos propuestos en arquitecturas de redes existentes.

- **Capítulo 5 Conclusiones y Trabajo Futuro**. En este capítulo se presentan las conclusiones del trabajo de investigación y se discuten las posibles líneas de investigación futuras.

Capítulo 2

Aprendizaje Profundo y Visión Artificial

2.1. ¿Por qué Aprendizaje Automatizado para Visión Artificial?

Tradicionalmente, los algoritmos utilizados en el campo de la Visión Artificial no hacen uso de métodos de Aprendizaje Automatizado o *Machine Learning*, o bien lo utilizan como etapa final de un *pipeline* principalmente basado en otras técnicas, en cuyo caso hablamos de *Aprendizaje Automatizado Aplicado*.

Un problema fundamental de la Visión Artificial es la representación de características o *features* en una imagen: características que son consideradas relevantes para resolver el problema en cuestión. No solo es importante la detección automática de dichas características, sino la definición de una representación de las mismas que permita encapsular la información relevante. Esta información puede ser la presencia de bordes, la forma de una curva, información de color o abstracciones de más alto nivel. La elección y el diseño de las características tienen un gran impacto en la solución al problema atacado, y son dependientes de la naturaleza del mismo.

En el acercamiento tradicional a la Visión Artificial, la aplicación de algoritmos de Aprendizaje Automatizado suele ser vista luego de la extracción de las características de la imagen, las cuales son tomadas como entrada para resolver un problema determinado, como puede ser el de la clasificación de imágenes en diferentes categorías. Sin embar-

go, dichas técnicas de Aprendizaje Automatizado no intervienen tradicionalmente en el diseño y detección de las características relevantes. La decisión de *qué* constituye una característica en una imagen y *cómo* representarla es parte del diseño de la solución, por lo que hablamos de una ingeniería manual de características o *Feature Engineering*. Ejemplos notables de algoritmos de detección y representación de características son SIFT [Low99], SURF [BTVG06] y FAST [RD06].

Con el éxito reciente de las redes neuronales artificiales de gran profundidad, bajo el campo de investigación denominado **Aprendizaje Profundo**, esta decisión ya no tiene que ser tomada explícitamente, sino que las características óptimas para un determinado conjunto de datos son *aprendidas* durante el proceso de entrenamiento de la red. De esta manera, el algoritmo de aprendizaje automatizado puede encontrar características que un diseño manual pasaría por alto o que resultan difíciles de representar: ¿cómo representamos un concepto de alto nivel como "rueda de bicicleta"?

Por lo tanto, en problemas donde se cuente con un conjunto de datos que constituya una muestra representativa de la totalidad de los ejemplos que podemos encontrar, es posible obviar el diseño manual de las características y aprender de manera automática las mismas mediante un algoritmo de Aprendizaje Automatizado.

2.2. Aprendizaje Supervisado

Muchos problemas de Aprendizaje Automatizado pueden ser reducidos a encontrar una función $f : X \to Y$, donde X representa el espacio de entradas posibles e Y el espacio de salidas. En el caso de la clasificación de imágenes, por ejemplo, X representa el espacio de imágenes posibles e Y un conjunto de valores interpretados como probabilidades para cada una de las categorías del problema.

En el Aprendizaje Supervisado, en particular, contamos con ejemplos del espacio de entradas $x \in X$ para los cuales conocemos su salida esperada -verdadera- $y \in Y$. Estos ejemplos son denominados datos de *ground truth*. Podemos pensar el conjunto de estos ejemplos como una muestra tomada de una distribución D generadora de datos que no conocemos. De esta manera, estamos asumiendo que X es una variable aleatoria y que los ejemplos de entrenamiento son independientes e idénticamente distribuidos (i.i.d.). El objetivo ideal del aprendizaje es el de encontrar una función f^* que represente

correctamente el mapeo $X \to Y$ para cualquier ejemplo de la distribución D.

Sin embargo, siempre contaremos con un conjunto finito (una muestra) de los ejemplos de entrada posibles. Debemos contentarnos entonces con que f^* prediga los valores correctos para los ejemplos de entrenamiento y, a su vez, esperar un rendimiento similar en nuevos ejemplos que no fueron observados durante el proceso de aprendizaje. Cuando esto sucede, decimos que el modelo *generaliza* correctamente.

Durante el aprendizaje debemos conocer qué tan lejos se encuentra el modelo del rendimiento esperado. Para ello, definimos una función $L(\tilde{y}, y)$ para un valor verdadero y y una predicción $\tilde{y} = f(x)$. Esta función cuantifica el error de predicción o la *pérdida* del modelo para cada ejemplo de entrenamiento x.

El aprendizaje, entonces, se reduce a encontrar una función f^* que minimice la pérdida L para los n ejemplos de entrenamiento (x_i, y_i):

$$f^* = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} L(f(x_i), y_i). \tag{2.1}$$

En la práctica, f será una función parametrizada por un conjunto de parámetros θ, por lo que buscaremos el conjunto θ^* que minimice la pérdida L.

Podemos ver que el Aprendizaje Supervisado depende directamente de la utilización de los datos de *ground truth*. De ahí su nombre: supervisamos el aprendizaje mediante datos de entrada cuya salida es conocida e informada durante el proceso de entrenamiento.

2.2.1. Regularización

La ecuación 2.1 tiene una solución trivial: una función que tome, para cada ejemplo x del conjunto de entrenamiento, el valor y asociado, y sea igual a 0 en el resto del dominio. Podemos variar este último valor, reemplazando 0 por cualquier escalar, y tendremos otra solución trivial. Vemos entonces que el problema no tiene una única solución, sino infinitas. Sin embargo, como se mencionó anteriormente, buscamos una función f^* que no solo se comporte de manera esperada para los ejemplos de entrenamiento, sino para todo ejemplo del espacio X. Las soluciones triviales mencionadas tendrán una pérdida nula para los ejemplos vistos, pero no así para nuevos ejemplos no contemplados durante el aprendizaje. Debemos entonces favorecer soluciones que *generalicen* a nuevos ejemplos por sobre aquellas que no lo hagan. La pregunta que surge inmediatamente es ¿cómo podemos favorecer una solución sobre otra si solo contamos con los ejemplos de entrenamiento y

ambas cometen el mismo error en ellos?

Una estrategia consiste en preferir soluciones simples por sobre soluciones complejas. Si dos funciones minimizan la pérdida, consideramos que la función de menor complejidad constituye una mejor solución al problema. Para formalizar este concepto durante el proceso de entrenamiento, incluimos un término de regularización en la función de pérdida que representa la complejidad del modelo, de manera de minimizar no solo el error cometido en la predicción, sino también la complejidad de la función ajustada:

$$f^* = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} L(f(x_i), y_i) + R(f), \tag{2.2}$$

siendo R una función escalar que no depende de los datos, sino de la función ajustada.

Intuitivamente, la penalización de soluciones complejas introduce la asunción de que una solución de mayor complejidad es menos probable de ser la correcta que una de menor complejidad. Es decir, consideramos más probable a una solución que requiera la complejidad *justa* que indican las observaciones, y no una complejidad adicional *innecesaria* introducida por el diseño del proceso de aprendizaje.

2.3. Regresión Lineal

La manera más simple de modelar una variable dependiente a partir de una (o varias) variable/s independiente/s parte de asumir que dicha relación es lineal, por lo que el modelo consistirá en una combinación lineal de las variables de entrada. Formalmente, buscamos modelar una variable y a partir de N variables de entrada $x_{1...N}$ de manera que

$$y = \sum_{i}^{N} k_i x_i = k_1 x_1 + k_2 x_2 + ... + k_N x_N + b, \tag{2.3}$$

donde los factores $k_{1...N}$ y b son parámetros del modelo. El parámetro b es denominado *bias* y permite que se encuentren soluciones que no pasen por el origen.

Si organizamos los parámetros y las variables de entrada en dos vectores k y x, respectivamente, podemos definir la ecuación 2.3 en forma matricial:

$$y = k^T x + b \tag{2.4}$$

Llamamos **Regresión Lineal** al proceso mediante el cual estimamos los parámetros $k_{1...N}$, b.

Una función de pérdida comúnmente utilizada en la Regresión Lineal es el cuadrado del error cometido en la predicción, es decir:

$$L(\hat{y}, y) = (\hat{y} - y)^2, \tag{2.5}$$

por lo que el problema se reduce, en ese caso, a encontrar los parámetros k^* y b^* de una función lineal f^* de manera que

$$f^*_{k,b} = \arg\min_{k,b} \frac{1}{m} \sum_{i=1}^{m} (k^T x_i + b - y_i)^2. \tag{2.6}$$

Equivalentemente, podemos decir que buscamos la función f^* que minimiza el Error Cuadrático Medio (ECM) de las predicciones. Adicionalmente, podemos incluir un término de regularización al igual que en la ecuación 2.2 para favorecer soluciones con parámetros de menor magnitud, de manera de evitar pesos desproporcionados sobre algunas variables, buscando que la predicción dependa en mayor medida de la totalidad de las variables de entrada.

En la práctica, al emplear un método de Regresión Lineal se debe tener en cuenta que la asunción de la relación lineal entre las variables representa un gran sesgo en el proceso de modelado. En este caso, estamos intentando resolver un problema N dimensional ajustando una línea, plano o hiperplano de acuerdo al número N de variables de entrada, por lo que la capacidad de modelado es limitada y evidentemente será insuficiente en caso de que la relación real entre las variables sea no lineal.

2.4. Regresión Logística

En la Regresión Logística, nuestro objetivo no es el de predecir el valor numérico de una variable dependiente, sino clasificar el ejemplo de entrada como perteneciente a una de múltiples posibles clases. Las variables dependientes representarán una distribución probabilística discreta, donde el valor de salida de cada una de ellas estará asociado a la masa de la distribución para dicha categoría, es decir, a la probabilidad del ejemplo de entrada de pertenecer a dicha clase. Cuando las clases en consideración se reducen a dos, hablamos de Regresión Logística Binaria.

Para abordar el problema de clasificación binaria, podemos utilizar las ideas desarrolladas en Regresión Lineal y modificar la ecuación 2.4:

$$y = f(k^T x + b), \tag{2.7}$$

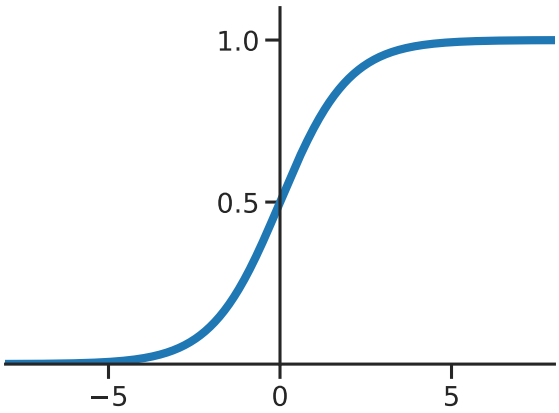


Figura 2.1: Función logística.

siendo la función f una función *sigmoide*, que transforma la salida de la combinación lineal de la entrada en un valor en el intervalo $(0, 1)$ que podemos interpretar como la probabilidad del ejemplo de entrada de pertenecer a una clase. De esta manera, un ejemplo *pertenece* o *no pertenece* a una categoría particular.

Una función sigmoide comúnmente utilizada es la *función logística*, muchas veces llamada, incorrectamente, función sigmoide:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.8}$$

Podemos ver en la Figura 2.1 que la función logística se acerca asintóticamente a 0 o a 1 cuando su entrada tiende a $-\infty$ o a $+\infty$, respectivamente.

Si generalizamos este concepto para clasificar un ejemplo entre múltiples clases, hablamos de Regresión Logística Multinomial. En ella, el valor de predicción y es un vector de tantos valores como clases posibles, asociándose cada valor a una de dichas clases. Normalizando el vector de manera que sus componentes tomen valores en el intervalo $(0, 1)$ y su suma sea siempre igual a 1, podemos interpretarlo como una distribución discreta donde cada valor representa la probabilidad del ejemplo de pertenecer a cada una de las clases. Una función comúnmente utilizada para normalizar la predicción es la *función Softmax*, que generaliza la función logística para casos multinomiales de K clases:

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, \quad \text{para } j = 1, ..., K. \tag{2.9}$$

Interpretamos los valores de entrada a la función Softmax como probabilidades logarítmicas no-normalizadas. Al utilizar la exponenciación, las convertimos en probabilidades no-normalizadas, las cuales finalmente normalizamos dividiendo por su suma total.

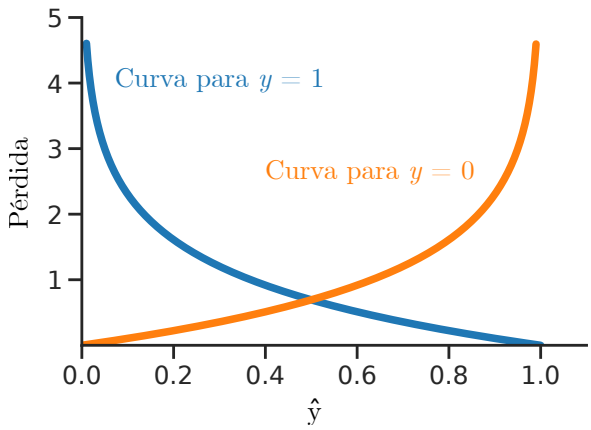


Figura 2.2: Entropía Cruzada.

Al igual que en la Regresión Lineal, podemos cuantificar el error de predicción mediante una función de pérdida. En la Regresión Logística utilizamos Entropía Cruzada entre la distribución real y la distribución predicha. Para el caso binario, siendo y la clasificación real, con valor 1 si el ejemplo pertenece a la clase y 0 en caso contrario, e $\hat{y}$ la probabilidad predicha, definimos la pérdida como:

$$L(\hat{y}, y) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}) \tag{2.10}$$

En cualquier caso, solo uno de los términos de la función no será nulo, dependiendo el valor real de y. El signo negativo en ambos términos se debe a que $\hat{y} \in (0, 1)$, por lo que ambos logaritmos serán negativos. Como podemos ver en la Figura 2.2, el valor real y determina la orientación de la función de pérdida, la cual en ambos casos decrece a medida que el valor predicho $\hat{y}$ se acerca a y.

En el caso multinomial, $\hat{\mathbf{y}}$ será un vector de probabilidades para las clases. Las componentes del vector $\mathbf{y}$ serán iguales a 0 excepto aquella asociada a la clase verdadera del ejemplo, la cual será igual a 1. Esta codificación es denominada *one-hot*. La función de pérdida será en este caso:

$$L(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_i y_i \log(\hat{y}_i) \tag{2.11}$$

Los términos de la suma serán nulos, excepto en la componente asociada a la clase del ejemplo, debido a la codificación del vector $\mathbf{y}$. En ese caso, la pérdida se reduce al logaritmo (negativo) de la predicción para dicha clase, como sucede en el caso binario.

(a)

(b)

Figura 2.3: Una imagen (a) cambia su interpretación semántica al desordenar sus píxeles (b).

2.5. La Imagen como Entrada

Una imagen es un conjunto de píxeles representado en una matriz de dos dimensiones (tres si consideramos los canales de color) con una restricción particular: el ordenamiento de los píxeles en estas dimensiones espaciales es único y no puede ser alterado. Si es alterado, hablamos de una imagen diferente a la original, y quizás perdamos la coherencia semántica de la imagen, al menos para nuestra interpretación como humanos (ver Figura 2.3).

En un gran número de problemas, el reordenamiento de las diferentes dimensiones del dato no modifica al dato en sí, tratándose del mismo ejemplo en uno u otro caso. Esto no es cierto si el dato en cuestión es una imagen. Tener en cuenta esta restricción espacial resulta fundamental para resolver problemas de Visión Artificial.

2.6. Redes Neuronales

En un gran número de problemas, la relación entre las entradas a un modelo y la salida esperada es una relación no lineal. Los modelos lineales, si bien simples y fáciles de implementar, tienen una limitada *capacidad* y no podrán representar correctamente dicha relación, tanto en problemas de regresión como en problemas de clasificación. Las **Redes Neuronales Artificiales**, o simplemente Redes Neuronales, son modelos complejos capaces de representar, potencialmente, funciones de arbitraria complejidad.

Una Red Neuronal es en su esencia una función no lineal del vector de entradas x,

parametrizada por θ, un vector de parámetros también conocido como vector de *pesos*. Los valores específicos de estos parámetros son ajustados durante el proceso de entrenamiento de la red. Una propiedad inherente a la construcción de las redes, y responsable en gran parte del éxito que estas muestran actualmente, es la organización jerárquica de las operaciones computadas, que permite reconocer y *aprender* características de la entrada a diferentes niveles de abstracción y complejidad.

Como su nombre lo indica, una Red Neuronal consta de un conjunto de unidades o *neuronas* conectadas de manera específica, cada una de las cuales realiza una serie de cálculos. Una red puede ser vista como un grafo de cómputos dirigido en el que las neuronas representan los nodos y los arcos están dados por el flujo de cálculos realizados, lo que impone una dependencia u orden en las operaciones. Un arco dirigido desde una neurona n_i hacia otra n_j indica que la salida de n_i será una de las entradas de n_j.

Las operaciones del grafo son realizadas secuencialmente, partiendo desde la entrada de la red. Al existir una dependencia entre las operaciones, en cada paso solo será posible calcular la salida de un cierto conjunto de neuronas, constituido por aquellas para las cuales se encuentra disponible la entrada (dependiente de operaciones previas ya realizadas). A este conjunto de neuronas lo llamamos *capa* y se desprende implícitamente que las neuronas de una misma capa no pueden estar conectadas entre sí.

El número de neuronas, el número de capas y las conexiones entre las neuronas definen la *arquitectura* de una red neuronal.

2.6.1. La Unidad Básica: la Neurona

Dada una entrada n-dimensional $x \in \mathbb{R}^n$, una neurona artificial computa su salida (activación) como:

$$a = \sigma(w^T x + b) \tag{2.12}$$

Donde w es un vector n-dimensional de parámetros o pesos, b es el parámetro de *bias* y σ es una función de activación no lineal. La función de activación es la que permite que una red neuronal ajuste un modelo no lineal de la entrada y reconozca patrones complejos; sin dicha función, a pesar de la organización jerárquica de sus unidades, una red neuronal no sería más que una secuencia de transformaciones lineales sobre la entrada, es decir, un modelo lineal.

La neurona calcula, entonces, el producto punto entre el vector de entrada y el vector

Figura 2.4: Valores de verdad dados por una función XOR representados en diferentes colores, con $Azul = Falso$. Los ejes representan las dos dimensiones (entradas) a la operación lógica, pudiendo tomar los valores $0 = Falso$ y $1 = Verdadero$. Nótese la imposibilidad de separar el espacio de acuerdo a los valores de verdad de salida mediante una única línea.

de parámetros *entrenables*, transformando el resultado de acuerdo a una función de activación no lineal. En la práctica, el factor b se incluye en una última dimensión adicional en el vector de pesos w. De la misma manera se expande el vector x asignando un valor igual a 1 en dicha dimensión, de manera que la suma del factor escalar b en la ecuación 2.12 sea realizada al calcular el producto punto:

$$w = \begin{bmatrix} w_1 \\ \vdots \\ w_n \\ b \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \\ 1 \end{bmatrix}, \quad a = \sigma(w^T x) = \sigma\left(\begin{bmatrix} w_1 & \dots & w_n & b \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \\ 1 \end{bmatrix} \right) \tag{2.13}$$

Como se puede observar en la ecuación 2.13, una neurona es un modelo lineal seguido de la aplicación de una función de activación. Una consecuencia de este modelo es la imposibilidad de una única neurona de representar una compuerta lógica XOR, al no ser posible la separación de su salida de manera lineal (ver Figura 2.4). Este hecho fue demostrado por Minsky y Papert [MP69] a finales de la década de 1960, generando una gran confusión en la comunidad científica al interpretarse erróneamente algunas de las conclusiones y provocando la pérdida de interés en las redes neuronales por más de una década.

2.6.1.1. Funciones de Activación

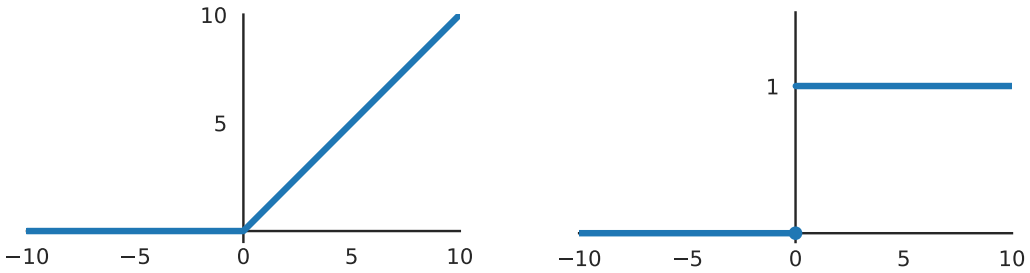
Históricamente, las funciones de activación más utilizadas fueron funciones monótonas y sigmoides que permiten aproximar el concepto de activación binaria en una neurona (o bien la neurona se activa de acuerdo a la entrada recibida, o bien permanece inactiva) y restringen el valor de salida de la neurona entre un máximo y un mínimo. Ejemplos de este tipo de funciones son la *tangente hiperbólica* y la función *logística* mencionada en la Sección 2.4.

Sin embargo, estas funciones presentan un inconveniente cuando son insertadas en redes de múltiples capas de profundidad. Debido a que sus valores de salida se acercan asintóticamente a un mínimo y a un máximo a medida que la entrada se hace menor o mayor, respectivamente, la derivada de dichas funciones respecto a la entrada es cercana a 0 en gran parte del dominio. Como se detalla en la Sección 2.7.3, las redes neuronales son entrenadas propagando el gradiente del error cometido en la predicción desde las capas finales hacia las capas iniciales, por lo que un gradiente cercano a 0 en una neurona de la red impide que la señal de entrenamiento continúe propagándose hacia neuronas de capas previas.

En el año 2000 [HSM+00] se introdujo una nueva función de activación, el *rectificador*:

$$\sigma(x) = x^+ = max(0, x) \tag{2.14}$$

Una neurona que utiliza esta función de activación es denominada *Unidad Lineal Rectificada* o ReLU, por sus siglas en inglés (Rectified Linear Unit) [NH10]. Una de sus ventajas respecto a las funciones mencionadas anteriormente es el hecho de la eliminación de la saturación de la salida para valores positivos del dominio. El rectificador contribuye



(a) Rectificador **(b)** Heaviside

Figura 2.5: Rectificador (a) y su derivada, la función de Heaviside (b).

a eliminar el problema del gradiente cercano a 0, siendo su derivada la función escalón de Heaviside, como puede verse en la Figura 2.5. La derivada en 0 no está definida, aunque en la práctica suele establecerse en 0.

Las ventajas del rectificador en el entrenamiento de redes neuronales de múltiples capas fueron puestas de manifiesto en 2011 [GBB11]. Actualmente es la función de activación más popular y se la considera uno de los aportes responsables del éxito en la utilización de redes profundas durante la década de 2010. Se han propuesto múltiples variantes en los últimos años, entre las que se encuentran ELU [CUH15], LeakyReLU [MHN13] y su generalización, PReLU [HZRS15].

2.6.2. Feedforward Neural Networks

Como se mencionó anteriormente, una neurona es una unidad capaz de representar una función lineal sobre un vector de entradas $x \in \mathbb{R}^n$, por lo que representa un modelo con limitada capacidad para resolver problemas no linealmente separables.

Sin embargo, un conjunto de neuronas es capaz de resolver problemas más complejos separando el espacio de las entradas mediante múltiples líneas o hiperplanos. En la Figura 2.6 podemos observar cómo dos rectas pueden resolver el problema de la función XOR planteado anteriormente, permitiendo modelar dicha operación mediante dos neuronas *en una misma capa.*

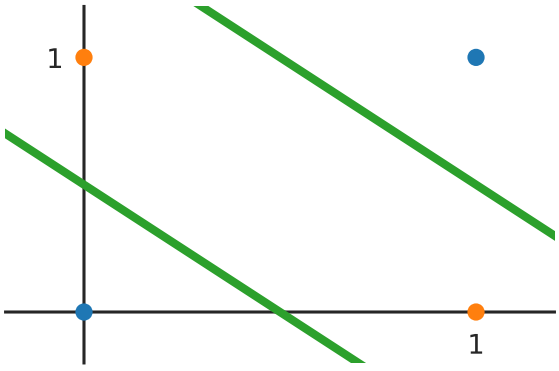


Figura 2.6: Los valores de verdad dados por una función XOR pueden ser separados mediante dos líneas, por lo que esta operación es representable mediante una capa de dos neuronas.

En el año 1989, Cybenko probó que una red neuronal de una única capa puede aproximar cualquier función continua si se utiliza una función de activación sigmoide [Cyb89]. Sin embargo, el número de neuronas requerido y el algoritmo de entrenamiento de los pesos de las neuronas no se especifica. Más aún, la disposición de neuronas en una única capa dista de ser la arquitectura más eficiente en términos de cantidad de neuronas necesarias, facilidad de entrenamiento y minimización del error cometido en la aproximación. En el año 1991, Hornik mostró que el potencial de las redes neuronales como aproximadores universales se pone de manifiesto al disponer las neuronas en múltiples capas, es decir, donde las salidas de ciertas neuronas constituyen la entrada de otras [Hor91].

Podemos ver que las redes neuronales tienen el potencial de resolver problemas de complejidad arbitraria. Las limitaciones en la práctica están dadas principalmente por el hardware utilizado y la cantidad de datos de entrenamiento disponibles.

2.6.2.1. Redes Neuronales Totalmente Conectadas

En una red neuronal podemos considerar al vector de entrada x como una *capa de entrada*. De la misma manera, la salida de la red puede ser considerada una *capa de salida*. Las capas de neuronas internas, que contienen los pesos de la red, son denominadas *capas ocultas*.

Una disposición natural de las neuronas de una red resulta de conectar todas las neuronas de una capa con todas las neuronas de la capa siguiente. Esta disposición se denomina **Red Neuronal Totalmente Conectada**, o *Fully-Connected Neural Network*. El vector de entrada x estará entonces completamente conectado con todas las neuronas de la primera capa. Denominando w_i al vector de pesos de una neurona i (incluyendo el término correspondiente al *bias* como se indicó en la ecuación 2.13), podemos definir una matriz de pesos W_1 en la que cada fila contenga los pesos de una neurona de la capa. De esta manera, podemos calcular las activaciones de la primera capa como:

$$a_1 = \sigma(W_1 x) \tag{2.15}$$

siendo σ la función de activación de las neuronas computada elemento a elemento. El vector de activaciones a_1 tendrá tantas dimensiones como neuronas formen la capa.

Repitiendo el proceso, podemos definir una segunda capa que tome como entrada el vector a_1 y tenga asociada una matriz de pesos W_2, y así sucesivamente para crear una

red neuronal de la cantidad de capas deseada, quedando definida como una secuencia de multiplicaciones de matrices y funciones no lineales aplicadas elemento a elemento. Por ejemplo, una red de cuatro capas es una función computada como:

$$f(x) = W_4\sigma(W_3\sigma(W_2\sigma(W_1x))) \tag{2.16}$$

En la capa de salida no solemos aplicar la función de activación σ; en algunos casos podemos utilizar una función de activación diferente, dependiendo del modelo que estemos ajustando. Esto sucede, por ejemplo, en redes para problemas de clasificación, en las que encontramos en la última capa funciones de activación que transforman los valores de salida a valores interpretables como una distribución probabilística sobre las categorías disponibles, como se desarrolló en la Sección 2.4.

2.6.2.2. Inconvenientes de las Redes Neuronales Totalmente Conectadas para Visión Artificial

Estas redes presentan inconvenientes al ser utilizadas en problemas de Visión Artificial, donde la entrada a la red es una imagen o un conjunto de ellas, ya que:

- Cada neurona *observa* la totalidad de la imagen, teniendo asociado un peso diferente para cada dimensión de entrada (correspondiente a un píxel en la primera capa, con tres o más pesos si consideramos el valor de píxel asociado a cada canal de color). La relación espacial que existe entre estos valores en la imagen no es tenida en cuenta. Si pensamos en el reordenamiento de los píxeles de la imagen, basta con reordenar apropiadamente el vector de pesos de cada neurona para que el resultado sea el mismo. Sin embargo, la imagen ya no puede ser considerada la misma.

- Una imagen desplazada un píxel en cualquier dirección hará que los pesos asociados a cada píxel varíen por completo y, como consecuencia, el resultado de las operaciones computadas. Vemos entonces que es deseable la invariancia o equivariancia traslacional para problemas de Visión Artificial: buscamos detectar las características particulares de una imagen independientemente de su posición espacial en la misma.

- El número de parámetros a entrenar es muy grande. Para una imagen de 100x100 píxeles en escala de grises, tendremos 10.000 parámetros en una sola neurona de la

primera capa. Considerando que buscamos implementar redes de múltiples capas, con un gran número de neuronas en cada una de ellas, la cantidad de parámetros constituye un problema tanto para el hardware actual como para la cantidad de datos con la que podemos contar, siendo una regla general la necesidad de una mayor cantidad de datos si buscamos entrenar más parámetros.

Estos inconvenientes han llevado al desarrollo de redes especialmente diseñadas para problemas de Visión Artificial, que tienen en cuenta las características particulares del empleo de imágenes como datos de entrada. Las más populares son denominadas Redes Neuronales Convolucionales o *Convolutional Neural Networks*.

2.6.3. Redes Neuronales Convolucionales

Las Redes Convolucionales interpretan la entrada respetando su ordenamiento espacial, lo que las hace adecuadas para reconocimiento de patrones en datos visuales; a su vez, su diseño resuelve los inconvenientes enumerados anteriormente en Redes Totalmente Conectadas.

Consideremos una ventana de una imagen de entrada de $k \times k \times c$ elementos, donde c es la cantidad de canales (por ejemplo, canales RGB de la imagen). Podemos desplazar esta ventana en las dos dimensiones espaciales de la imagen de manera de cubrir la totalidad de esta, superponiendo o no las ventanas a medida que nos desplazamos. En cada posición de la ventana podemos definir la entrada de $k \times k \times c$ elementos a una determinada neurona de la primera capa. Tendremos entonces tantas neuronas como posiciones posibles de la ventana en la imagen. Vemos que, a diferencia de las Redes Totalmente Conectadas, una neurona ya no *observa* la totalidad de la entrada. Más aún, en una Red Convolucional dichas neuronas comparten los parámetros. Una interpretación equivalente es considerar que contamos con una única neurona de $k \times k \times c$ parámetros entrenables, que es desplazada a través de la entrada, calculando su activación en cada una de las posiciones en las que se ubica. Esta operación de desplazamiento no es otra cosa que la convolución discreta de un *filtro* o *kernel* con la entrada (de allí el nombre de estas redes), donde el filtro está dado por los pesos de la neurona.

Podemos entrenar en cada capa tantos de estos filtros como deseemos. El conjunto de estos filtros formará una *capa convolucional*. En general, dado un tensor de entrada de $h_{in} \times w_{in} \times c_{in}$ elementos, la salida de la capa será otro tensor de $h_{out} \times w_{out} \times c_{out}$. Cada

canal de salida albergará la respuesta a un filtro particular sobre el tensor de entrada. Podemos ver entonces la respuesta a cada filtro como una matriz de dos dimensiones, donde cada ubicación espacial denota la respuesta del filtro al tomar como entrada la ventana centrada en dicha posición.

El número de elementos por el que desplazamos la ventana en determinada dirección se denomina *stride*. Por ejemplo, con un $stride = 1$ tenemos que $h_{in} = h_{out}$ y $w_{in} = w_{out}$ (asumiendo la utilización de *zero-padding*, técnica que consiste en expandir los bordes de la entrada asignando valores nulos con el objetivo de poder ubicar la ventana en el centro de las posiciones correspondientes a bordes y esquinas).

La configuración de una capa convolucional queda completamente definida especificando 4 hiperparámetros:

- El tamaño F de los filtros. Suelen emplearse filtros cuadrados, por lo que basta con especificar un valor entero. Adicionalmente, cada filtro tendrá siempre un valor de *bias*.
- La cantidad K de filtros a aplicarse. Esta cantidad definirá los canales de salida, con $K = c_{out}$.
- El valor de *stride* S o desplazamiento de las ventanas.
- La utilización o no de *zero-padding* P en los bordes de la entrada.

El número de parámetros a entrenar en una capa convolucional queda determinado por el tamaño y número de los filtros, y por los canales de entrada. Debido a que se trata de una operación de convolución, el tamaño de las dimensiones espaciales de la entrada no afecta el número de parámetros de la capa. Como consecuencia, una capa convolucional puede recibir entradas de diferentes dimensiones espaciales sin verse afectada en su funcionamiento.

La cantidad de parámetros por cada filtro es $F \times F \times c_{in}$. La cantidad total de parámetros de la capa será entonces $F \times F \times c_{in} \times K$, más K *biases*.

Los filtros suelen tener tamaños impares pequeños, siendo los filtros de 3×3 los más utilizados. Nótese cómo la extensión espacial de los filtros es pequeña, pero su extensión *en profundidad* es a través de la totalidad de los canales, los cuales pueden contarse en decenas o cientos, especialmente en las capas ocultas de la red.

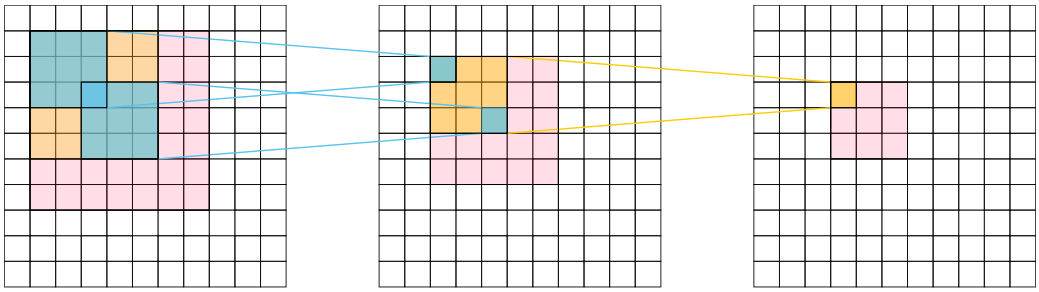


Figura 2.7: El campo receptivo de un filtro respecto a la entrada de la red aumenta a medida que nos movemos hacia capas más profundas. En la figura, el filtro correspondiente a cada capa es de 3×3 elementos en todos los casos (en color celeste para la primera capa de la izquierda, en amarillo para la segunda y en rosa para la tercera). Podemos ver en cada capa, según el color correspondiente, el campo receptivo efectivo que tiene un filtro, es decir, cuáles son las regiones que afectan los elementos sobre los que se aplicará. Se puede apreciar cómo un filtro de 3×3 elementos en la tercera capa *observa* una región de 7×7 elementos en la primera.

2.6.3.1. Campo Receptivo

La disposición en capas de las redes convolucionales permite el reconocimiento jerárquico de patrones en las imágenes. Las capas más cercanas a la entrada, o *menos profundas*, reconocerán patrones simples y universales como bordes y puntos, mientras que las capas superiores combinarán patrones de capas inferiores para encontrar características complejas.

Para que esto sea posible, es necesario que los filtros de las capas superiores tengan un *campo receptivo* que permita *observar* buena parte de la imagen. Los filtros de la primera capa tendrán, naturalmente, un campo receptivo igual a su tamaño. Los de la segunda capa, al ser aplicados sobre respuestas de los filtros de la primera, tendrán un campo receptivo mayor (ver Figura 2.7). De esta manera, observamos que la adición de capas a la red aumenta el campo receptivo de los filtros en las capas finales. Sin embargo, se necesitan decenas de capas para abarcar una región considerable de una imagen. Por ejemplo, dada una entrada de 100×100 píxeles, considerada pequeña según los estándares actuales, y utilizando filtros de 3×3 en toda la red, necesitamos 24 capas para alcanzar un campo receptivo de 49×49 en la última capa, es decir, para abarcar aproximadamente la cuarta parte de la imagen.

Por supuesto, existe la posibilidad de aumentar el campo receptivo de una capa me-

diante la utilización de filtros de mayores dimensiones espaciales; sin embargo, también aumenta considerablemente el número de parámetros a entrenar. Es más eficiente, en términos de número de parámetros, la adición de capas con filtros de menor tamaño: una capa de filtros de 5×5 contará con 25 parámetros (en un solo filtro o canal), mientras que dos capas consecutivas de 3×3, con un campo receptivo efectivo de 5×5, sumarán $3 \times 3 + 3 \times 3 = 18$ parámetros. Más aún, dos capas representan un modelo de mayor complejidad que una sola de ellas, por lo que también ganamos capacidad de representación.

Para alcanzar un campo receptivo considerable, entonces, necesitamos un gran número de capas consecutivas, especialmente al tener en cuenta que el aumento del tamaño de los filtros no es una solución deseable. Sin embargo, un gran número de capas se traduce en un gran número de parámetros a entrenar, lo que vuelve más complejo el entrenamiento del modelo, más aún cuando se cuenta con una cantidad de datos limitada. Adicionalmente, encontramos rápidamente limitaciones de hardware si tenemos en cuenta que, en la práctica, la totalidad del modelo suele permanecer en memoria de GPU.

Una solución al problema del campo receptivo y a las limitaciones de hardware es la utilización de capas de *pooling*.

2.6.3.2. Pooling

Una estrategia muy utilizada para aumentar el campo receptivo de los filtros y al mismo tiempo controlar el tamaño de los tensores de activación es aplicar capas de *pooling* en ciertos puntos de la red. Estas capas, al igual que las convolucionales, definen ventanas sobre la entrada. Sin embargo, no se aplica un filtro en cada posición, sino que se calcula una operación de submuestreo de la ventana independientemente para cada canal. Los tipos de pooling más utilizados son **average pooling**, donde el valor de salida se calcula como la media de los valores de entrada y **max pooling**, cuya salida es el valor máximo de la ventana [RHBL07]. Este último es el más utilizado por haber mostrado mejores resultados experimentales.

El *stride* en las capas de *pooling* suele ser establecido de manera que no exista superposición entre ventanas. Una configuración habitual viene dada por un $stride = 2$ y un tamaño de ventana de 2×2, reduciendo a la mitad el tamaño del tensor de entrada en cada una de las dimensiones espaciales, es decir, reduciendo el tensor a un cuarto de su tamaño original. Debe notarse que el *pooling* no agrega parámetros a la red.

Intuitivamente, podemos entender el *max pooling* de la siguiente manera: si la característica que el filtro busca reconocer está presente en la ventana, existirá una activación de considerable magnitud, la cual será *seleccionada* por el *max pooling*. Si bien perdemos resolución espacial, detectamos la presencia de cierta característica dentro de la ventana en la que se aplica la operación.

Como consecuencia de una operación de *pooling*, el tensor de salida de una capa se reduce en sus dimensiones espaciales y los filtros de la siguiente capa aumentan su campo receptivo debido a que son aplicados sobre posiciones que, antes de la operación de *pooling*, no eran abarcables por la misma ventana. Adicionalmente, se gana invariancia espacial dentro de cada ventana: perdemos la ubicación exacta de la activación que fue propagada, mientras que el resultado de la operación hubiese sido el mismo si dicha activación se hubiese encontrado en cualquiera de las otras ubicaciones.

2.7. Optimización

Habiéndose definido e inicializado una red neuronal, el siguiente paso es el de optimizar sus parámetros θ de manera de minimizar la función de pérdida sobre los ejemplos de entrenamiento. Debido a la complejidad de la función representada por una red y al gran número de neuronas en cada una de sus múltiples capas, resulta imposible encontrar una solución analítica para θ. Más aún, las redes neuronales aproximan funciones no lineales arbitrarias (definidas implícitamente por su arquitectura), haciendo que se trate de un problema de optimización no convexo, por lo que existirá un número potencialmente infinito de mínimos locales. El método numérico más utilizado en redes neuronales es denominado *Gradient Descent*, y tiene múltiples variantes.

2.7.1. Gradient Descent

En determinado momento, cualquiera sea el valor de un parámetro de un modelo, podemos actualizarlo de dos maneras: o bien aumentamos su magnitud, o bien la disminuimos. Al tomar una de estas decisiones nos alejamos o acercamos a su valor óptimo. Vemos, entonces, que una estrategia válida para la optimización del parámetro es calcular la dirección en la que debemos realizar la actualización (no necesariamente su magnitud). Esta es la idea en la que se basa el algoritmo **Gradient Descent**: como su nombre lo

indica, calcula la dirección de actualización correcta de acuerdo al gradiente de la función de pérdida respecto al parámetro, es decir, considera cómo cambia el error cometido por el modelo al modificar el parámetro en cuestión. Mediante dicho gradiente podemos conocer la dirección en la cual la pérdida *aumentaría*, por lo que actualizando el parámetro en la dirección contraria disminuimos la pérdida.

La función de pérdida depende de un gran número de parámetros, especialmente en redes neuronales. Podemos visualizar dicha función como una superficie multidimensional, donde cada punto del dominio representa un conjunto diferente de valores para todos los parámetros, es decir, un modelo. El valor de la función en cada punto indica el error cometido por el modelo asociado **en la totalidad del conjunto de entrenamiento**. Durante el proceso de entrenamiento, siempre nos encontramos en un punto determinado de dicha superficie. Al aplicar *Gradient Descent* calculamos la dirección de *descenso más pronunciado* en la vecindad del punto en el que nos encontramos y actualizamos el conjunto de parámetros en dicha dirección, de manera de disminuir la pérdida. Repetimos este proceso hasta alcanzar la convergencia.

Formalmente, en cada paso de entrenamiento, actualizamos un parámetro θ_i de la siguiente manera:

$$\theta_i \leftarrow \theta_i - \alpha \frac{\partial L}{\partial \theta_i} \tag{2.17}$$

El factor α controla la magnitud de la actualización en la dirección requerida y es denominado *tasa de aprendizaje* (*learning rate*). Si utilizamos un valor muy pequeño, el parámetro requerirá un mayor número de actualizaciones hasta alcanzar el valor óptimo, pero corremos el riesgo de permanecer en un mínimo local. Si aumentamos el valor, este puede ser lo suficientemente grande como para desplazarnos en la superficie hacia un punto en el que la pérdida sea mayor, pudiendo incluso hacer que el proceso diverja.

La tasa de aprendizaje es un hiperparámetro de la red y su valor es crítico para el correcto entrenamiento de la misma. Generalmente es establecido de manera empírica y se utilizan estrategias de actualización de su valor a medida que transcurre el proceso, siendo común la disminución progresiva de su valor de manera continua o en actualizaciones discretas cada cierto número de iteraciones.

2.7.2. Stochastic Gradient Descent

Actualmente, las redes neuronales son utilizadas para modelar problemas complejos en los que los datos de entrada son multidimensionales, como sucede con datos visuales, y en los que se tiene un gran número de ejemplos de entrenamiento. El algoritmo original de *Gradient Descent* requiere que se calcule, en cada iteración, el error de predicción que comete la red para la totalidad del conjunto de datos. Esto resulta inviable por limitaciones de hardware, considerando que estos conjuntos de datos pueden contar con cientos de miles de ejemplos. Más aún, deberíamos calcular las predicciones de todos los ejemplos del *dataset* para completar una única actualización en los parámetros, imposibilitando el proceso de entrenamiento en términos de tiempo de ejecución.

Sin embargo, podemos aplicar la misma idea utilizando solo un ejemplo de entrenamiento por iteración. En este caso, la superficie representada por la función de pérdida será diferente a aquella que tenemos al considerar la totalidad del *dataset*, como también el gradiente del error con respecto a los parámetros. El problema de optimización habrá cambiado (y cambiará en cada iteración al variar el ejemplo de entrada), por lo que la actualización de los parámetros ya no será en la dirección de descenso más pronunciado para el *dataset*, sino *para el ejemplo seleccionado* en cada paso de entrenamiento.

A pesar de estas consideraciones, en la práctica, el algoritmo ha demostrado ampliamente su utilidad en la optimización de modelos. El orden en el que son seleccionados los ejemplos de entrenamiento es aleatorio, lo que hace del algoritmo un proceso estocástico, dándole su nombre: **Stochastic Gradient Descent** (**SGD**). Por otra parte, esta variante presenta ciertas ventajas:

- Al variar la función de pérdida iteración a iteración, disminuyen las probabilidades de mantenernos dentro de un mínimo local,

- Considerar un ejemplo por iteración introduce ruido en el proceso de optimización, lo que funciona como una regularización del modelo y colabora en la generalización de este a nuevos ejemplos.

En el entrenamiento de redes neuronales, especialmente redes de Aprendizaje Profundo, no se suele utilizar un único ejemplo de entrenamiento por iteración, sino un pequeño conjunto de estos, de manera de acelerar el entrenamiento sin perder los beneficios que

provee el SGD. A este conjunto de ejemplos se lo conoce como *mini-batch*, por lo que en este caso hablamos de **Mini-Batch Gradient Descent**.

Variantes de SGD

Para lograr una convergencia más rápida, se han propuesto técnicas y variantes del algoritmo de SGD original.

Una técnica común se denomina **Momentum** y consiste en acumular las direcciones de los gradientes de todos los pasos de entrenamiento como una media móvil exponencial (*exponential moving average*), utilizando el gradiente del último paso para actualizar la dirección del gradiente acumulado. La actualización de los parámetros se realiza de acuerdo a este último, logrando un efecto de *memoria* para con las direcciones anteriores. El objetivo es el de aproximar la dirección real del gradiente para todos los ejemplos vistos, siendo en este caso una aproximación de primer orden, ya que acumulamos la media del gradiente. Otros optimizadores como **Adagrad** [DHS11] y **RMSProp** [TH12] aproximan el gradiente con términos de segundo orden. **Adam** [KB14], uno de los optimizadores más utilizados actualmente, mantiene aproximaciones tanto de primer como de segundo orden, y puede ser visto como una combinación de RMSProp y Momentum.

2.7.3. Backpropagation

Como se ha mencionado, para actualizar un parámetro debemos conocer la derivada de la función de pérdida o error de predicción cometido con respecto a dicho parámetro. En una red neuronal, la complejidad de las conexiones y la propagación de la activación de cada neurona en operaciones de capas posteriores imposibilitan el cálculo directo de dicha derivada, por lo que se requiere de otra estrategia.

El algoritmo de *Backpropagation* [RHW85] posibilita el cálculo de gradientes haciendo uso de la regla de la cadena de las derivadas; si tenemos una composición de funciones

$$(h \circ (g \circ f))(x) = h(g(f(x))), \tag{2.18}$$

podemos calcular la derivada de la función h con respecto a x como:

$$\frac{dh}{dx} = \frac{dh}{dg}\frac{dg}{df}\frac{df}{dx} \tag{2.19}$$

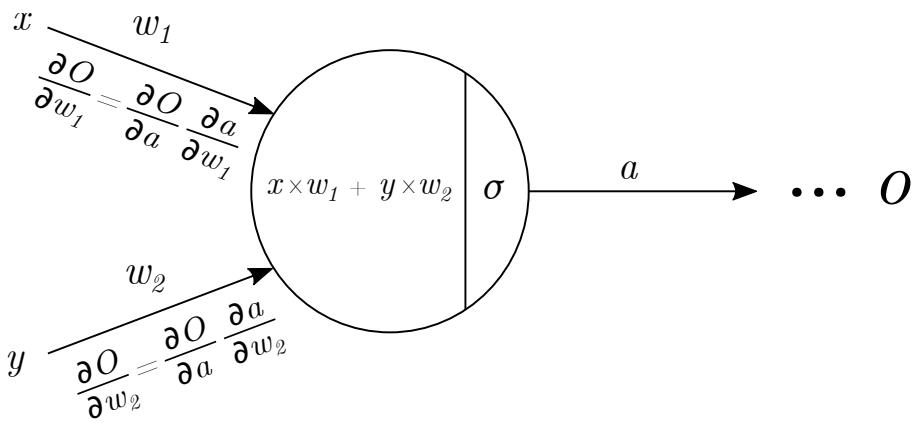


Figura 2.8: Una neurona con entradas x e y, parámetros w_1 y w_2, y una función de activación σ. Podemos calcular el efecto que tiene un parámetro w_i sobre cualquier operación posterior O utilizando la regla de la cadena de las derivadas.

En una red neuronal, la entrada a una neurona está compuesta por la salida de ciertas neuronas de la capa previa. Es decir, tenemos una composición de las funciones representadas por las neuronas.

El efecto que tendrá la salida de una neurona sobre cualquier operación posterior en una red puede ser medido como la derivada de dicha operación posterior con respecto a la activación de la neurona. A su vez, el gradiente de la activación de la neurona con respecto a sus parámetros o pesos, representa el efecto que producen los parámetros sobre la activación cuando son modificados. Por regla de la cadena, podemos entonces calcular el efecto de los parámetros de una neurona con respecto a cualquier operación posterior de la red, como la derivada de la operación respecto a la activación de la neurona, multiplicada por el gradiente de la activación con respecto a los parámetros. Esto puede ser visto gráficamente en la Figura 2.8.

En una red neuronal, siempre podemos calcular la derivada de la función de pérdida con respecto a las salidas de la última capa, ya que estas representan la única entrada de la función de pérdida. Adicionalmente, cualquier neurona de la red puede calcular su gradiente local, es decir, el gradiente de su activación con respecto a sus propios parámetros, ya que este es independiente del resto de las operaciones de la red. De esta manera, la última capa conoce los dos gradientes necesarios para actualizar sus parámetros: el gradiente local y el gradiente de la función de pérdida con respecto a su salida. A su vez, cada parámetro de la última capa de la red está directamente asociado a la activación de una neurona de la capa previa, por lo que podemos conocer su gradiente. Esto último,

sumado al conocimiento del gradiente local de cada neurona de la capa previa, permite conocer los gradientes de los parámetros de esta capa. Así, sucesivamente, propagamos el gradiente por la red en dirección contraria a la utilizada al computar la predicción. Una vez alcanzada la primera capa, sabremos los gradientes para los parámetros de cada neurona de la red, por lo que podremos actualizarlos de acuerdo a la política establecida por el algoritmo de optimización, y así completar un paso de entrenamiento.

2.8. Consideraciones Adicionales y Otras Técnicas

2.8.1. Inicialización de los Parámetros de una Red

Antes de comenzar el entrenamiento, debemos inicializar los parámetros de la red. La inicialización constituye una decisión importante, ya que las predicciones para los primeros ejemplos de entrenamiento dependerán casi exclusivamente de los valores asignados, y será en gran medida por estos que la optimización logre o no comenzar el proceso de entrenamiento con éxito.

Considerando que las activaciones de las capas contendrán valores positivos y negativos, aproximadamente centrados en 0, parece razonable inicializar todos los pesos de la red en 0 y dejar que el algoritmo de optimización actualice sus valores hacia donde se requiera. Sin embargo, esta inicialización haría que las activaciones de todos los filtros sean iguales, por lo que también lo serían sus actualizaciones en cada paso de entrenamiento, y nunca se rompería la simetría. Es decir, estaríamos entrenando un único filtro en cada una de las capas, replicado innecesariamente.

Por esta razón, es práctica común utilizar inicializaciones con valores aleatorios pequeños, de acuerdo a distribuciones centradas en 0. En Redes Convolucionales, en particular, suele utilizarse una distribución normal truncada, donde no existen valores desplazados a más de un cierto número de desvíos estándar de la media, de manera de evitar inicializaciones de gran magnitud (positiva o negativa).

2.8.2. Regularización en Redes Neuronales

La forma más empleada de regularización en redes neuronales es denominada *weight decay*. En esta, utilizamos la magnitud de los parámetros entrenables como medida de la

complejidad de la función que ajusta la red, prefiriendo parámetros cercanos a cero por sobre parámetros de gran magnitud. Intuitivamente, estamos forzando a la red a realizar la predicción utilizando la totalidad de sus parámetros en lugar de unos pocos de gran peso.

Pensando en el caso extremo en el que solo tenemos un ejemplo de entrenamiento, sin regularización, los parámetros asociados a características importantes de dicho ejemplo crecerían indefinidamente en magnitud para maximizar la precisión en la predicción (minimizar la diferencia entre la predicción y la salida esperada, haciendo cada vez mayor la entrada a la última función sigmoide de manera de acercarse asintóticamente a una probabilidad igual a 1). Si introducimos un segundo ejemplo luego del entrenamiento, los parámetros que la red consideró importantes, los cuales tendrán una gran magnitud, lo eran específicamente para el primer ejemplo, por lo que es probable que el error de predicción para el segundo ejemplo sea alto.

Podemos considerar el caso general de múltiples ejemplos de entrenamiento de manera similar: no importa el tamaño de este conjunto, siempre podrá ser visto como una muestra pequeña frente a los infinitos ejemplos posibles. Para no *sobreajustar* los parámetros al conjunto de entrenamiento, buscando una buena *generalización* en las predicciones, forzaremos una magnitud pequeña en los parámetros mediante el factor de regularización. De esta manera, ciertos parámetros encontrarán una contradicción en las señales de actualización: para reducir el error de predicción deben crecer, pero el factor de regularización los forzará a decrecer. La única manera para la red de continuar minimizando la función de pérdida es mediante la utilización de otros parámetros de la red que seguramente "no hubiesen sido tenidos en cuenta" sin la influencia que ejerce la regularización.

En la práctica suelen emplearse las normas $L1$ y $L2$ de los parámetros de la red, siendo la regularización $L2$ la más utilizada. Esta última puede definirse como

$$R_{L2}(\theta) = \frac{1}{2}\lambda \sum_i \theta_i^2 \tag{2.20}$$

donde λ representa el peso que se le da al factor de regularización en la función de pérdida. El escalar $\frac{1}{2}$ es un factor de normalización: como se verá posteriormente, los parámetros serán actualizados de acuerdo al gradiente, por lo que la actualización del parámetro θ_i vendrá dada por $\lambda\theta_i$ en lugar de $2\lambda\theta_i$, lo que elimina una multiplicación en la implementación del algoritmo sin afectar el proceso de optimización.

La regularización $L1$, por su parte, se define como

$$R_{L1}(\theta) = \lambda \sum_i |\theta_i| \tag{2.21}$$

En la práctica, al forzar los parámetros a mantenerse cercanos a cero, el cuadrado de cada parámetro será menor a su valor absoluto, por lo que la regularización $L2$ suele tener menor influencia que la regularización $L1$. Para compensar este hecho, podemos aumentar el valor del hiperparámetro λ en la regularización $L2$.

En general, la regularización $L2$ suele dar mejores resultados que la $L1$. Esta última tiende a forzar a ciertos parámetros a volverse prácticamente iguales a cero, por lo que podemos interpretar que selecciona ciertos parámetros de la red. Si no estamos interesados en la selección de parámetros, la regularización $L2$ suele ser la mejor opción. También existe la posibilidad de combinar ambos tipos de regularización, en lo que se denomina *Elastic Net Regularization* [ZH05].

2.8.3. Data Augmentation

Otra forma de evitar el *sobreajuste* de los parámetros es mediante *Data Augmentation*. Como su nombre lo indica, consiste en aumentar los datos de entrenamiento disponibles mediante su manipulación y transformación, de manera de generar nuevos ejemplos de entrenamiento válidos. En el caso de datos visuales, podemos generar nuevos ejemplos mediante rotaciones, espejado, recortes, cambios en la iluminación o color, cambios en el enfoque, e incluso aplicando transformaciones perspectivas leves o difuminando la imagen.

Toda transformación del ejemplo original es válida siempre que la etiqueta asociada siga siendo válida luego de la transformación. Por ejemplo, si aplicamos un espejado horizontal a la imagen de un perro, esta sigue representando la imagen de un perro, por lo que es un nuevo ejemplo válido de entrenamiento. Sin embargo, si espejamos una imagen que solo contiene texto, el ejemplo dejará de tener sentido bajo la misma transformación.

2.8.4. Dropout

Una estrategia comúnmente utilizada en el aprendizaje automatizado consiste en entrenar múltiples modelos buscando que cada uno de ellos aproveche diferentes características de los datos de entrenamiento. Para la predicción final se combinan las predicciones

de todos los modelos, esperando de esta manera una mejor generalización y un menor sobreajuste que el de un modelo único.

En redes neuronales, una forma de implementar esta idea es mediante la utilización, en cada paso de entrenamiento, de un subconjunto aleatorio de neuronas de la red. Así, podremos considerar que contamos con un modelo diferente en cada paso de entrenamiento, donde la red con todas sus neuronas activas representa el conjunto de modelos entrenados, el cual realizará la predicción final. Esta técnica es denominada *Dropout* [SHK+14] y ha gozado de una gran popularidad en los últimos años.

Para implementarla, se establece un hiperparámetro p que indica la probabilidad $1-p$ de una neurona de encontrarse inactiva, en cuyo caso su salida es forzada a cero, por lo que no tendrá participación en la predicción y no será entrenada. La proporción esperada de neuronas inactivas en cada paso será entonces $1-p$. Alternativamente, podemos ver al *Dropout* como una técnica de regularización, ya que anular aleatoriamente la salida de una neurona es equivalente a introducir ruido de acuerdo a una variable aleatoria de Bernoulli con parámetro p. Introduciendo ruido aleatorio, dificultamos el sobreajuste de los parámetros a las respuestas específicas de los ejemplos de entrenamiento.

Esta técnica es comúnmente utilizada en capas Totalmente Conectadas. No suele aplicarse en capas convolucionales debido a la menor cantidad de parámetros y a la correlación espacial de los datos de entrada. Especialmente en Redes Convolucionales, la técnica de *Batch Normalization* ha permitido alcanzar mejores resultados experimentales.

2.8.5. Batch Normalization

Durante el proceso de entrenamiento de una red neuronal los parámetros de las capas son modificados iteración a iteración, por lo que las distribuciones de entrada a cada capa, dependientes de los parámetros de la capa anterior, se ven afectadas de manera permanente. Este cambio es denominado *Covariate Shift* y presenta inconvenientes en el entrenamiento de los parámetros de la red, especialmente los de capas superiores, ya que las variaciones serán más pronunciadas en dichas capas al ser producto de la acumulación de variaciones de todas las capas previas. Tenemos entonces una variación permanente de las distribuciones de entrada a las capas, mientras que la distribución de las respuestas esperadas para los ejemplos de entrenamiento se mantiene igual.

Una solución a este inconveniente es utilizar *Batch Normalization* [IS15], técnica que

consiste en controlar la variación en las activaciones de cada capa normalizando sus distribuciones. Controlando la media y la varianza de las activaciones *estabilizamos* las distribuciones de entrada a las capas, disociando el entrenamiento de una capa del entrenamiento de las anteriores, ya que reducimos el impacto de los cambios en los parámetros de capas previas. De esta manera, hacemos más robustas las capas superiores frente a cambios en las capas inferiores de la red.

Cabe mencionar que al normalizar las activaciones no estamos perdiendo información específica del ejemplo ni las respuestas a los distintos filtros en cada ubicación espacial, ya que estamos aplicando simplemente una transformación lineal.

Como su nombre lo indica, la técnica de *Batch Normalization* tiene en cuenta las estadísticas del *mini-batch* de ejemplos de entrenamiento. Formalmente, definiendo al *mini-batch* de m ejemplos como $\mathcal{B} = \{x_{1...m}\}$, las operaciones realizadas para su aplicación son:

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \tag{2.22}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \tag{2.23}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \tag{2.24}$$

$$y_i \leftarrow \gamma\hat{x}_i + \beta \equiv BN_{\gamma,\beta}(x_i) \tag{2.25}$$

Vemos que, en primer lugar, calculamos la media $\mu_{\mathcal{B}}$ del *mini-batch* (ec. 2.22) y su varianza $\sigma^2_{\mathcal{B}}$ (ec. 2.23). Luego, normalizamos el *mini-batch* de manera de tener media igual a 0 y varianza igual a 1, substrayendo $\mu_{\mathcal{B}}$ y diviendo por $\sigma_{\mathcal{B}}$, siendo $\hat{x}_i$ el resultado de esta transformación (ec. 2.24). El factor ϵ es un escalar pequeño para otorgar estabilidad numérica y evitar la división por cero. Finalmente, escalamos y desplazamos $\hat{x}_i$ mediante γ y β, respectivamente (ec. 2.25). Los parámetros γ y β son aprendidos durante el entrenamiento y son únicos en cada una de las capas. Sin estos, se limitarían las características que una capa puede representar.

La técnica de *Batch Normalization* es ampliamente utilizada en Aprendizaje Profundo y ha facilitado el proceso de entrenamiento, permitiendo tasas de aprendizaje más altas, haciendo más flexibles las inicializaciones de los parámetros y acelerando su aprendizaje. Adicionalmente, tiene efectos de regularización, ya que la media y la varianza utilizadas en la normalización son calculadas sobre el *mini-batch* y no sobre el conjunto completo

de entrenamiento, por lo que estamos introduciendo ruido aleatorio y, por consiguiente, haciendo más difícil el sobreajuste de los parámetros. En la práctica, sin embargo, suele utilizarse en conjunto con otra técnica de regularización, comúnmente *regularización L2*.

Esta página intencionalmente en blanco.

Capítulo 3

Expansión de los Filtros Convolucionales

Las Redes Neuronales Convolucionales han probado su gran capacidad como modelos de aproximación de funciones generales, así como en la resolución práctica de problemas de alta complejidad en el área de Visión Artificial.

La variabilidad que se espera encontrar en las muestras para un problema determinado (en escala, rotación, perspectiva, transformaciones de cuerpo rígido, entre otras) constituye el factor más importante en la dificultad de los problemas de clasificación y segmentación de imágenes. Para obtener modelos que mantengan un buen rendimiento a pesar de esta variación, es decir, que ataquen el problema en cuestión a través de la distribución real de las muestras de entrada, se cuenta con dos alternativas:

- Obtener una gran variabilidad en la muestra de entrenamiento, de manera de contemplar en el conjunto de datos las diferentes poses y transformaciones que se puedan encontrar en los objetos representados en las imágenes,
- Diseñar las Redes Neuronales Convolucionales de manera de contemplar estas transformaciones como parte de su arquitectura, logrando invariancia a dichas transformaciones sin la necesidad de adquirir una mayor cantidad de datos.

La primera opción es un abordaje del problema por fuerza bruta para obtener invariancia ante las diferentes transformaciones. En un caso hipotético en el que se cuenta con infinitos datos, la primera opción sería suficiente. Sin embargo, en la práctica los conjuntos de datos son limitados y el etiquetado de las imágenes tiene un alto costo de

obtención, principalmente cuando debe ser realizado por un humano. En Segmentación Semántica, por ejemplo, donde la etiqueta para una imagen está constituida por una clasificación de clase para cada píxel, el costo de etiquetado es aún más alto.

La segunda opción es la única que resulta viable en la práctica, aunque requiere de modificaciones en el diseño de las redes. A pesar del éxito que han sabido demostrar en los últimos años las Redes Convolucionales, su componente fundamental, el filtro convolucional, ha mantenido tradicionalmente su estructura geométrica fija. Resulta natural, por lo comentado anteriormente, explorar la extensión de los filtros convolucionales de manera de flexibilizar los cómputos que realizan, especialmente la estructura geométrica de la entrada que es muestreada en cada posición espacial.

Las propuestas de modificación directa al filtro convolucional surgen con dos objetivos principales:

- Aumentar la invariancia ante transformaciones de la entrada, por ejemplo, invariancia a diferentes escalas.
- Obtener modelos de mayor capacidad manteniendo el número de parámetros y/o la cantidad de cómputos involucrados.

En este capítulo se presenta un análisis de las limitaciones que presentan los filtros de estructura geométrica fija, así como diferentes propuestas para flexibilizar la estructura del filtro convolucional en búsqueda de superar estas limitaciones.

3.1. Trabajo Relacionado

El diseño de arquitecturas de Redes Neuronales que alcancen mayores niveles de invariancia o equivariancia ante diferentes transformaciones fue explorado por varios trabajos mediante diferentes estrategias.

Gens y Domingos [GD14] proponen una arquitectura alternativa a las Redes Convolucionales que obtiene representaciones de la entrada con mayor invariancia en Grupos de Simetría arbitrarios, demostrando su propuesta con resultados favorables ante transformaciones afines.

Diferentes trabajos adoptan el enfoque de ingeniería de características buscando modificar las representaciones obtenidas por las redes de Aprendizaje Profundo de manera

explícita, con el objetivo de aumentar la invariancia ante distintas transformaciones. Los trabajos incluyen convoluciones con la Transformada Wavelet [BM13] y bancos de filtros transformados para obtener invariancia ante escala [KSJ14] y ante diferentes transformaciones lineales [SL12].

Otra estrategia que permite obtener resultados similares, así como filtrar información no relevante, está constituida por mecanismos de atención en los que los cómputos son restringidos a una o varias regiones particulares de la entrada. Girshick *et al.* [GDDM14], entre otros notables aportes, combinaron algoritmos de Proposición de Regiones (*Region Proposal*) dentro de Redes Convolucionales de manera de obtener vectores de características ricas para cada región propuesta, produciendo un cambio disruptivo en el estado del arte del problema de Detección de Objetos. Por su parte, el trabajo de Erhan *et al.* [ESTA14] demostró que las Redes Convolucionales son capaces de hacer regresión de Regiones de Interés (*ROIs*), lo que puede interpretarse como un mecanismo de atención sobre regiones de la entrada.

Generalizando sobre las ideas de invariancia ante transformaciones arbitrarias y mecanismos de atención, las Redes de Transformación Espacial (*Spatial Transformer Networks*) fueron propuestas en 2015 por Jaderberg *et al.* [JSZ+15]. Este trabajo propone la inclusión de un módulo que toma como entrada un mapa de características, computa un conjunto de parámetros que modelan una transformación (de complejidad arbitraria) y aplica la transformación a dicho mapa. Los parámetros de la transformación son predichos por la misma red, la cual aprende a predecir los parámetros convenientes durante el proceso de entrenamiento convencional del problema que se esté atacando sin la necesidad de incluir información adicional de *ground truth*.

Los diferentes trabajos mencionados avanzaron el estado del arte para diferentes tareas dentro del campo del Aprendizaje Profundo, aunque sin modificar la estructura geométrica de los filtros convolucionales, los cuales continuaban siendo aplicados en posiciones espaciales adyacentes de la entrada. El primer trabajo en proponer una modificación directa al filtro convolucional en estas redes fue publicado por [CPK+14]. En este se presenta la red *DeepLab* para atacar el problema de Segmentación Semántica (predicción densa por píxel). La propuesta incluye la introducción del filtro de **Convolución Dilatada** (*Atrous Convolution*), el cual mantiene el número de posiciones espaciales en los que se aplica un filtro cuadrado, aunque extiende espacialmente las dimensiones de dicho

filtro para abarcar una región mayor en la entrada. En este primer trabajo, esta modificación al filtro convolucional solo fue incluida en las capas finales, reemplazando las capas Totalmente Conectadas por capas de Convolución Dilatada. Posteriormente, un nuevo trabajo del mismo grupo de investigación propuso la adición de un conjunto de capas paralelas que utilizan la Convolución Dilatada con diferentes variaciones en la extensión espacial, y una posterior concatenación de las salidas de estas capas, con la intención de combinar filtros aplicados con diferentes dilataciones. A este conjunto de capas se lo denominó ASPP (del inglés *Atrous Spatial Pyramid Pooling*). Esta nueva arquitectura, denominada *DeepLabv2* [CPK+17] avanzó el estado del arte en Segmentación Semántica en *benchmarks* del reconocido conjunto de datos Pascal VOC 2012 [EEVG+15], así como se posicionó en los primeros puestos de las tablas comparativas para otros conjuntos de datos como Cityscapes [COR+16]. Subsecuentes artículos presentaron actualizaciones a estas redes, con modificaciones a la ASPP (*DeepLabv3*) [CPSA17] y a la arquitectura general de la red, así como el uso de la Convolución Dilatada Separable, una combinación entre las Convoluciones Dilatada y Separable que permite reducir el número de parámetros (*DeepLabv3+*) [CZP+18].

A diferencia de las distintas variantes de la red *DeepLab*, que explotan la replicación paralela de capas de Convolución Dilatada, el trabajo presentado por Yu y Koltun [YK15] propone un módulo que utiliza múltiples capas de Convolución Dilatada de manera secuencial, con sucesivos incrementos en la extensión espacial de los filtros de cada capa. El diseño de este módulo tiene como objetivo agregar información contextual a diferentes escalas sin la necesidad de introducir un submuestreo de su entrada, a diferencia de la aplicación de la operación de *pooling*.

La modificación al filtro que propone la Convolución Dilatada utilizada por los trabajos citados mantiene el concepto original del filtro convolucional de una capa, en la cual el mismo filtro (manteniendo la misma extensión espacial en el caso de la Convolución Dilatada) es aplicado en toda la entrada por igual. Es decir, no se cuenta con un mecanismo que permita modificar la estructura espacial del filtro dependiendo de la posición de la entrada en la que se está aplicando. El primer trabajo que flexibiliza esta característica de las capas convolucionales fue publicado en 2017 por Dai *et al.* [DQX+17]. En este se introdujo la noción de Convolución Deformable, en la cual la estructura del filtro a aplicar puede tomar una forma geométrica arbitraria, a diferencia de la estructura rectangular

convencional, y varía para cada posición de la entrada en la que se aplica. La estructura geométrica del filtro en cada posición de la entrada es obtenida a través de un conjunto de vectores predichos por la misma red que funcionan como vectores de desplazamiento desde la posición central del mismo para cada uno de sus elementos. Posteriormente, un artículo incluyendo modificaciones sobre la propuesta original fue publicado por el mismo grupo de investigación en 2019 [ZHLD19], en el cual la Convolución Deformable predice un conjunto adicional de valores de modulación que permiten variar la amplitud o peso que se le otorga a cada una de las posiciones espaciales de la entrada, otorgando mayor flexibilidad. En ambas propuestas, la estructura geométrica arbitraria parece indicar la necesidad de una gran cantidad de datos adicionales para evitar el *sobreajuste* o para lograr una buena predicción de los vectores, más aún teniendo en cuenta la necesidad de predecir al menos 18 valores por cada posición espacial de la entrada (para filtros de dimensiones similares a filtros cuadrados de 3×3 elementos, prediciendo vectores de dos dimensiones para cada uno de ellos).

Por otra parte, los trabajos mencionados fueron evaluados únicamente como módulos o capas dentro de redes de gran complejidad, lo que dificulta aislar la influencia que tienen las modificaciones propuestas sobre los resultados finales. Modelos complejos de millones de parámetros pueden presentar comportamientos caóticos en los que las mejoras que se esperan intuitivamente por modificaciones introducidas pueden no corresponderse con las causas reales (y muchas veces no previstas) de dichas mejoras. Un caso célebre en el que se confundió las causas de las mejoras en el rendimiento se evidenció con la introducción de la técnica de entrenamiento de Dropout [SHK+14]. La exclusiva evaluación frente a tareas de gran complejidad se relaciona con la práctica de la comunidad científica del área de Aprendizaje Profundo de otorgar implícitamente mayor validez a aquellos trabajos que logran avances en los *benchmarks* estandarizados para cada problema, a pesar de ser estos generalmente incrementales y de no presentar cambios sustanciales en los algoritmos.

A continuación se define formalmente la Convolución Dilatada, se detallan sus limitaciones y se presentan propuestas para superar las mismas.

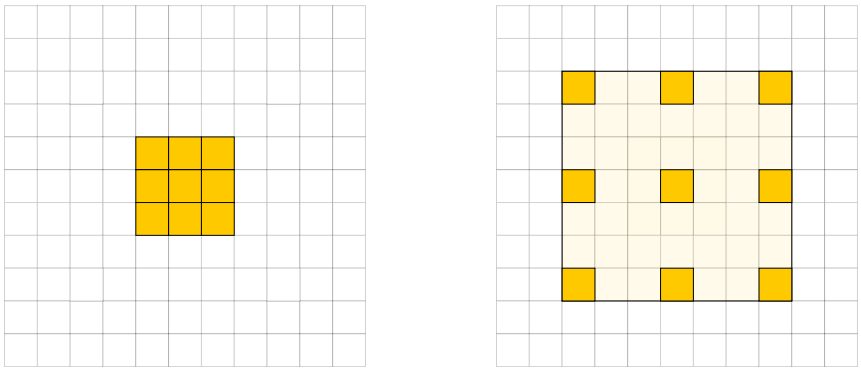
3.2. Convolución Dilatada

Como se vio en el capítulo 2, uno de los aspectos importantes a tener en cuenta en una red neuronal convolucional es el campo receptivo asociado a las neuronas, especialmente a las neuronas finales o en capas cercanas a la que realiza finalmente la predicción. Asimismo, se vio que podemos aumentar dicho campo receptivo de dos maneras:

- Aumentando la cantidad de capas de la red, de manera que las neuronas finales adquieran un mayor contexto espacial,
- Utilizando filtros (*kernels*) de mayor tamaño.

Ambas opciones tienen la desventaja de aumentar el número de parámetros o pesos a entrenar. La alternativa propuesta como parte de la red *DeepLab* citada anteriormente es la utilización de la denominada **Convolución Dilatada** (*Atrous Convolution*).

Con el objetivo de aumentar el campo receptivo de una neurona, este tipo de capa convolucional recrea la utilización de filtros cuadrados de mayor tamaño, aunque fijando el número efectivo de pesos en los filtros como si se tratara de filtros de un tamaño menor, comúnmente de 3×3. Para ello, un filtro convencional es *dilatado* desde el centro hacia afuera hasta alcanzar el tamaño deseado, desplazando las posiciones originales de los pesos del filtro hacia los extremos diagonales y centrales (ver Figura 3.1). De esta manera, obtenemos un kernel cuadrado de mayor tamaño en el que solo asociamos pesos en determinadas posiciones. Las ubicaciones restantes no tendrán peso asociado, lo que es equivalente a asumir que su peso tiene un valor fijo igual a 0. El nivel de dilatación



(a) **(b)**

Figura 3.1: Convolución mediante un filtro de 3×3 (a) sin dilatación, (b) con una tasa de dilatación igual a 3.

utilizado se indica mediante la *tasa de dilatación*, que define la distancia entre posiciones con pesos asociados. Una convolución con tasa de dilatación igual a 1 es equivalente a una convolución convencional.

Una capa convolucional dilatada, entonces, mantiene el número de pesos a entrenar aunque permite que los filtros puedan calcular combinaciones lineales de entradas que se encuentran en posiciones no adyacentes y potencialmente distantes en las dimensiones espaciales. Esto puede resultar particularmente útil en capas profundas de la red, donde cada entrada representa una característica con abstracciones de alto nivel, aumentando considerablemente el contexto que una neurona es capaz de *percibir*.

La utilidad de la Convolución Dilatada ha sido demostrada utilizando tanto una secuencia de capas convolucionales dilatadas que aumenten progresivamente el campo receptivo [YK15] como también múltiples capas paralelas con diferentes tasas de dilatación, técnica denominada *Atrous Spatial Pyramid Pooling* (ASPP) [CPK+17]. En los últimos años, esto ha permitido mejorar los rendimientos especialmente en problemas de segmentación semántica, permitiendo aumentar el campo receptivo de una capa sin pérdida de resolución espacial, a diferencia del uso de *pooling* o técnicas similares.

3.2.1. Definición como Generalización de Convolución Discreta

La Convolución Dilatada puede ser pensada como una generalización de la operación de Convolución Discreta en la que previamente se muestrean las entradas sin la restricción de que estas deban ser adyacentes en las dimensiones espaciales.

Formalmente, definimos el operador de Convolución Discreta $*$ en dos dimensiones como:

$$(I * k)(\mathbf{p}) = \sum_{\mathbf{s}+\mathbf{t}=\mathbf{p}} I(\mathbf{s})k(\mathbf{t}) \tag{3.1}$$

con $I : \mathbb{Z}^2 \to \mathbb{R}$ una función que representa la entrada bidimensional para una determinada posición espacial y $k : \Omega_f \to \mathbb{R}$, con $\Omega_f = [-f, f]^2 \cap \mathbb{Z}^2$, un filtro cuadrado de $(2f + 1)^2$. La posición espacial donde se centra el filtro en la entrada está representada por $\mathbf{p} \in \mathbb{Z}^2$.

Como se mencionó en el capítulo 2, en una capa convolucional, un filtro de dos dimensiones espaciales se extiende en una tercera dimensión a través de los canales de la entrada, como sucede con los tres canales *RGB* de una imagen. La ecuación 3.1 solo ha-

ce referencia a las dos dimensiones espaciales por simplicidad en la notación, aunque se debe tener en cuenta que la operación se extiende a través de la totalidad de una tercera dimensión.

Continuando, definimos el operador de Convolución Dilatada $*_r$ como una generalización de la Convolución Discreta:

$$(I *_r k)(\mathbf{p}) = \sum_{\mathbf{s}+r\mathbf{t}=\mathbf{p}} I(\mathbf{s})k(\mathbf{t}) \tag{3.2}$$

para una tasa de dilatación $r \in \mathbb{Z}^{\geq}$.

3.2.2. Limitaciones de la Convolución Dilatada

No obstante su demostrada utilidad, la definición de Convolución Dilatada presenta múltiples limitaciones relacionadas con la definición de la tasa de dilatación a utilizar. En particular, dicha tasa de dilatación:

- Debe ser indicada durante la definición de la capa en la arquitectura de la red, es decir, antes de comenzar el entrenamiento. Sin embargo, en este punto no tenemos información que nos ayude a definir la tasa de dilatación óptima a utilizar, por lo que es común recurrir a un proceso de prueba y error hasta dar con el valor que nos permita alcanzar el mejor rendimiento en el conjunto de validación. Es fácil ver el problema de este acercamiento: cada experimento de entrenamiento consume un tiempo considerable (en algunos casos, semanas), por lo que evitar este proceso no solo es deseable sino, en muchas oportunidades, necesario.

- Es utilizada por igual en la totalidad de las neuronas de la capa, independientemente de su posición espacial. Sin embargo, es natural pensar que cada posición puede tener una dilatación óptima diferente, dependiendo del contexto y del contenido semántico de la imagen de entrada. Por ejemplo, un objeto cercano a la cámara abarcará un mayor número de píxeles que el mismo objeto en una posición alejada, por lo que parece apropiado utilizar una dilatación mayor para aquellos filtros que se apliquen en las posiciones del primer objeto.

- No es modificada entre ejemplos de entrada, ya que es un parámetro especificado en la definición de la capa convolucional. Por este motivo, la misma tasa será aplicada

en todos los ejemplos por igual, a pesar del contenido semántico de los mismos o cambios en la naturaleza de las escenas.

Las limitaciones enumeradas desaparecen si podemos asignar una tasa de dilatación independiente en cada una de las posiciones espaciales y modificarla con cada ejemplo de entrenamiento. Adicionalmente, resulta de suma importancia encontrar la tasa de dilatación óptima para cada posición. A continuación se propone la definición dinámica de estas como una solución frente a las limitaciones expuestas.

3.3. Convolución de Dilatación Dinámica

Para atacar las limitaciones que presenta la Convolución Dilatada resulta necesario definir tasas de dilatación dinámicas durante el proceso de entrenamiento, independientes en cada posición espacial y apropiadas de acuerdo al contenido semántico de cada ejemplo. Podemos pensar entonces en un *mapa de tasas de dilatación*, una matriz de dos dimensiones conteniendo las tasas de dilatación a utilizar en cada una de las posiciones espaciales. Este mapa podrá ser generado, predicho o leído de manera dinámica durante el proceso de entrenamiento. Denominaremos **Convolución de Dilatación Dinámica** (**CDD**) a una operación de convolución que utilice un mapa para definir dinámicamente las tasas de dilatación a utilizar. Las estrategias utilizadas para la generación u obtención del mapa de dilataciones darán lugar a diferentes tipos de convolución que se proponen como parte de este trabajo de tesis.

3.3.1. En Búsqueda de Tasas de Dilatación Óptimas

Una pregunta natural que podemos formularnos al pensar en el mapa de dilataciones es: ¿qué constituye una tasa de dilatación óptima en una determinada posición?

Podemos ensayar ciertas intuiciones como potenciales respuestas a esta pregunta. Tal vez la más inmediata surge al considerar la transformación perspectiva que sufren los objetos en una imagen natural. Como se mencionó en la Sección 3.2 al desarrollar las limitaciones de la Convolución Dilatada tradicional, resulta natural pensar que las tasas de dilatación en objetos cercanos a la cámara deban ser mayores a aquellas en objetos alejados, ya que la captura del contexto del objeto requiere de traslaciones en el espacio

de la imagen y estas están directamente atadas a la transformación perspectiva de la cámara. Sin embargo, las siguientes cuestiones son difíciles de resolver a ciencia cierta:

- La intuición de requerir tasas de dilatación mayores en objetos cercanos a la cámara no especifica el valor explícito que tomaría la tasa de dilatación óptima. ¿Cómo decidir a ciencia cierta el valor óptimo exacto, más allá de una comparación cualitativa?

- A pesar de parecer correcta, no podemos saber *a priori* si dicha intuición constituye una base firme para supervisar el entrenamiento de una red. Es decir, no tenemos la certeza necesaria para transformar dicha intuición en datos de *ground truth*, ya que, por definición, no calificarían como tales. Más aún, dentro de la complejidad modelada por una red neuronal profunda, es imposible predecir cómo estos datos afectarían el rendimiento post-entrenamiento de la red.

- Esta intuición solo toma en cuenta aspectos relacionados con la geometría de la imagen. Otros factores, como el contenido semántico de la vecindad en determinada posición, la etiqueta asociada al objeto, la cercanía de bordes u otros aspectos desconocidos podrían afectar el valor de la tasa de dilatación óptima. Resulta imposible determinar la totalidad de los factores que influyen sobre esta y la ponderación que debemos hacer de ellos.

Pero fundamentalmente, aunque ataquemos u obviemos estos problemas, resulta extremadamente costoso requerir del etiquetado adicional de un conjunto de datos para la sola definición del mapa de dilataciones. Podemos concluir, entonces, que un acercamiento que nos permita generar dinámicamente el mapa de dilataciones, sin contar con datos adicionales, nos brinda el escenario más apropiado. A continuación proponemos diferentes tipos de capas de Convolución de Dilatación Dinámica de manera de atacar las limitaciones detalladas hasta el momento.

3.3.2. Convolución de Dilatación Adaptativa

Consideremos una red convolucional de arquitectura arbitraria en la que tenemos una capa de Convolución de Dilatación Dinámica, la cual requiere la definición de un mapa de tasas de dilatación. Una alternativa para la generación dinámica de este mapa es que

la propia red, o una subred de esta, prediga específicamente las tasas de dilatación a utilizar en cada posición espacial. Denominaremos a esta alternativa **Convolución de Dilatación Adaptativa**.

Tomemos como ejemplo la definición de una capa paralela a la capa de CDD que tenga como único objetivo predecir las tasas de dilatación que esta última utilizará para el ejemplo de entrenamiento que se está procesando. Es decir, ambas capas toman la misma entrada, pero la salida de una de ellas se interpretará como las tasas de dilatación a utilizar por la otra (ver Figura 3.2). La capa de predicción de tasas de dilatación puede extenderse a una subred paralela de arquitectura arbitraria, siempre que su salida respete las dimensiones espaciales del mapa de tasas de dilatación a utilizar en la capa de CDD. Dicha salida será entonces una matriz de dos dimensiones en la que el valor en cada posición será interpretado como la tasa de dilatación a utilizar en dicha posición por la capa de CDD. De esta manera, permitimos que una capa o subred de la propia red a entrenar prediga las tasas de dilatación de otra capa de la red, generando dinámicamente las tasas para cada posición, para cada ejemplo de entrenamiento.

Al abordar la generación del mapa de tasas de dilatación de esta manera, debemos también abordar el problema del entrenamiento de la subred de generación. El objetivo será que dicha subred mejore la predicción de tasas de dilatación óptimas a medida que transcurre su entrenamiento. Debemos encontrar, entonces, un mecanismo para supervisar dicho entrenamiento.

El abordaje trivial es el de contar con datos de *ground truth* que permitan supervisar la salida de la subred y obtener el error cometido mediante una función de pérdida.

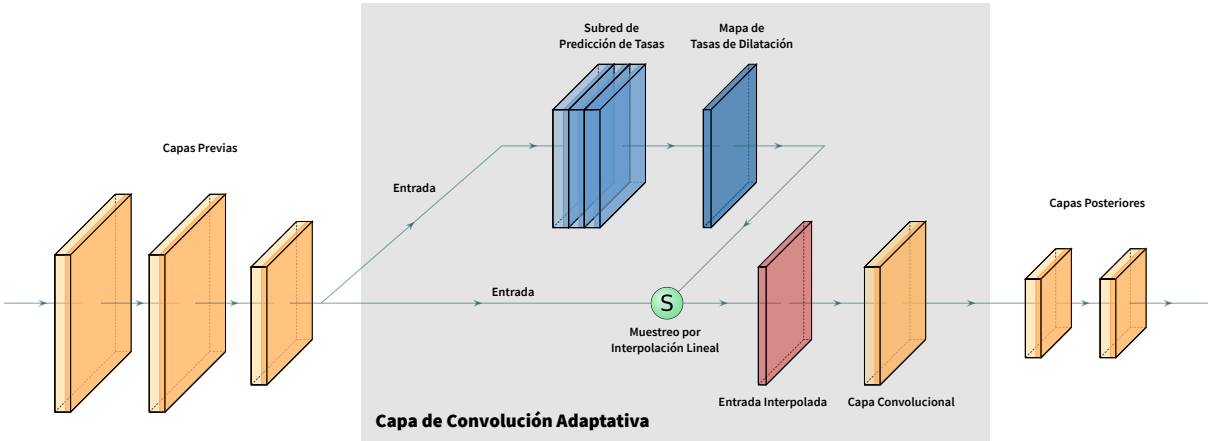


Figura 3.2: Capa de Convolución de Dilatación Adaptativa.

Sin embargo, como se mencionó anteriormente, no podemos definir de manera precisa e inequívoca la tasa de dilatación óptima en cada posición de una imagen de un conjunto de datos.

Resultaría conveniente, entonces, entrenar la subred de predicción del mapa de dilataciones sin la necesidad de contar con datos adicionales a los ya utilizados durante el entrenamiento de la red principal. Una consecuencia inmediata al considerar este acercamiento es que todas las operaciones utilizadas en la predicción y utilización del mapa de dilataciones deben ser diferenciables. Por este motivo, las tasas a utilizar en el proceso no podrán ser discretizadas, siendo necesaria la utilización de tasas de dilatación continuas. A continuación, definimos formalmente la utilización de dilataciones continuas para luego describir en detalle el proceso de entrenamiento de la subred de predicción de dilataciones.

3.3.2.1. Muestreo Diferenciable mediante Interpolación Lineal

Denominaremos r a una tasa de dilatación continua y consideraremos adicionalmente las dos tasas de dilatación enteras más cercanas a r, dadas por $\lceil r \rceil$ y $\lfloor r \rfloor$. Definiremos la **Convolución de Dilatación Continua** (**CDC**) como la interpolación lineal entre las dos operaciones de Convolución Dilatada con dichas tasas. Formalmente:

$$(I *_r k)(\mathbf{p}) = \left[(\lceil r \rceil - r) \sum_{\mathbf{s_1}+\lfloor r \rfloor \mathbf{t}=\mathbf{p}} I(\mathbf{s_1}) k(\mathbf{t}) \right] + \left[(r - \lfloor r \rfloor) \sum_{\mathbf{s_2}+\lceil r \rceil \mathbf{t}=\mathbf{p}} I(\mathbf{s_2}) k(\mathbf{t}) \right] \tag{3.3}$$

para una tasa de dilatación $r \in \mathbb{R}_{\geq 0}$.

Como puede observarse en la ecuación 3.3, la Convolución de Dilatación Continua calcula la suma pesada de las dos Convoluciones Dilatadas con tasas $\lceil r \rceil$ y $\lfloor r \rfloor$, utilizando los mismos parámetros para el filtro k en ambos casos. El peso de cada término es inversamente proporcional a la distancia entre la tasa de dilatación r predicha y la tasa de dilatación entera empleada.

La interpolación lineal calculada en cada posición prepara la entrada para la aplicación de los filtros convolucionales de la capa. Lo que diferencia a una capa de Convolución de Dilatación Continua de una capa convolucional tradicional es el muestreo previo a la aplicación de los filtros. Una vez preparada la entrada, no existen diferencias adicionales.

La operación de interpolación lineal es diferenciable excepto en los valores enteros, en este caso, correspondientes a dilataciones enteras. En la práctica, sin embargo, encontrar exactamente valores enteros es extremadamente improbable, por lo que podemos obviar

los casos no diferenciables sin temor a afectar el entrenamiento, pensando a la interpolación lineal como una operación diferenciable en el contexto del entrenamiento de una red. Aun en el caso de encontrar dilataciones enteras, sus aportes al gradiente propagado son cancelados, por lo que podemos considerarlos equivalentes a tener un gradiente igual a 0.

3.3.2.2. Aprendiendo las Tasas de Dilatación

Al contar con una subred de predicción de dilataciones, buscamos que sus capas sean correctamente ajustadas a medida que consideramos el error cometido para cada ejemplo de entrenamiento. En este caso, el error será el de considerar una tasa de dilatación sub-óptima.

Para ilustrar claramente el proceso de entrenamiento y la propagación del gradiente hacia la subred de predicción de dilataciones, consideraremos lo que sucede durante la aplicación de un filtro limitándonos a una única posición espacial. Asumiremos también que la profundidad de la entrada en la dimensión de los canales es igual a 1, como si se tratase de la primera capa de una red en la que la entrada es una imagen en escala de grises.

Denominaremos r a la tasa de dilatación predicha por la subred de predicción de dilataciones. Dado que r es un valor continuo, la dilatación del filtro hará que sus posiciones no se encuentren *alineadas* con las posiciones de la entrada, sino que estarán entre las posiciones que resultarían de considerar las tasas $\lceil r \rceil$ y $\lfloor r \rfloor$, como se puede ver en la Figura 3.3a.

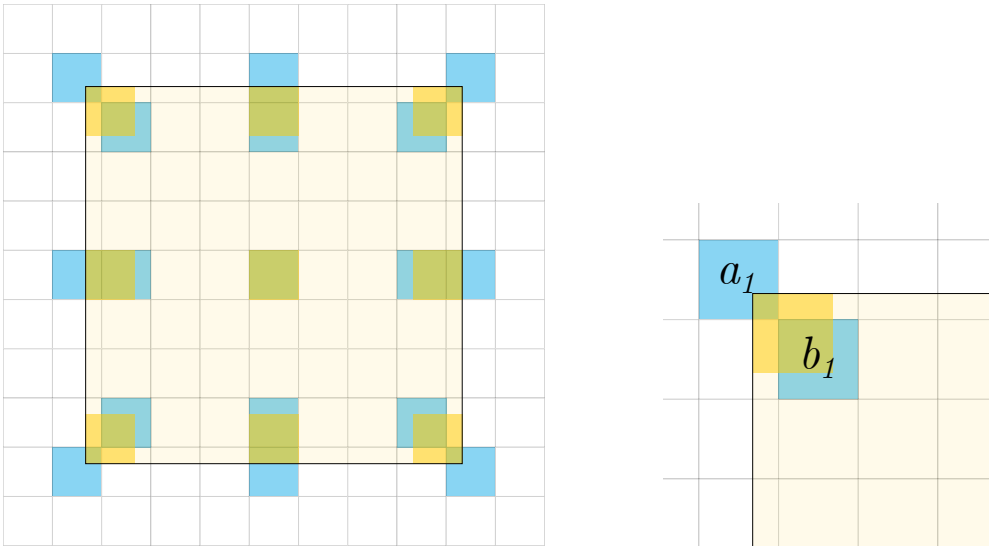
Nos centraremos en lo que sucede en la posición superior izquierda del filtro y luego lo extenderemos al resto de las posiciones. Llamaremos a_1 y b_1 a los valores que toma la entrada en las posiciones dadas por $\lceil r \rceil$ y $\lfloor r \rfloor$, respectivamente (ver Figura 3.3b). Calcularemos la interpolación lineal s_1 entre los valores a_1 y b_1:

$$s_1 = w_c \, a_1 + w_f \, b_1$$

con

$$w_c = r - \lfloor r \rfloor, \qquad w_f = \lceil r \rceil - r$$

los pesos aplicados de acuerdo a la inversa de la distancia lineal entre r y las posiciones enteras más cercanas, respetando $w_c + w_f = 1$. El valor s_1 será posteriormente multiplicado



(a) **(b)**

Figura 3.3: (a) Filtro dilatado mediante una tasa de dilatación continua $r = 3{,}2$. Las posiciones finales no se *alinean* con las posiciones de la entrada. En celeste se observan los valores de la entrada que serán tenidos en cuenta al calcular la interpolación lineal. (b) En la posición superior izquierda del filtro se interpolarán los valores a_1 y b_1 que toma la entrada.

por el valor f_1 del filtro convolucional:

$$o_1 = f_1 \, s_1$$

La salida o_1 será posteriormente sumada al resultado de las restantes operaciones del filtro, en el resto de las posiciones espaciales, que serán calculadas de la misma manera.

Durante la propagación del gradiente en un paso de entrenamiento, eventualmente conoceremos la derivada parcial de la pérdida L respecto a la operación $f_1 \, s_1$, esto es, $\frac{\partial L}{\partial o_1}$. Inmediatamente, podremos conocer la derivada de la pérdida respecto al parámetro f_1 del filtro, pero también respecto al valor interpolado s_1. Considerando siempre la posición espacial mencionada, tenemos:

$$\frac{\partial L}{\partial s_1} = \frac{\partial L}{\partial o_1} \frac{do_1}{ds_1}$$

con $\frac{do_1}{ds_1} = f_1$ y $\frac{\partial L}{\partial o_1}$ el valor acumulado del gradiente propagado desde la función de pérdida hasta la operación que estamos considerando.

Conocer $\frac{\partial L}{\partial s_1}$ es equivalente a conocer cómo debería cambiar la salida de la interpolación s_1 para disminuir la pérdida. A su vez, sabemos que s_1 depende directamente de los pesos

w_c y w_f, por lo que podemos también conocer cómo debemos modificar dichos pesos:

$$\frac{\partial L}{\partial w_c} = \frac{\partial L}{\partial s_1}\frac{\partial s_1}{\partial w_c}, \qquad \text{con } \frac{\partial s_1}{\partial w_c} = a_1$$

$$\frac{\partial L}{\partial w_f} = \frac{\partial L}{\partial s_1}\frac{\partial s_1}{\partial w_f}, \qquad \text{con } \frac{\partial s_1}{\partial w_f} = b_1$$

Ambos pesos dependen de la tasa de dilatación r, predicha por la subred de predicción de dilataciones. Tenemos:

$$\frac{dw_c}{dr} = 1, \qquad \frac{dw_f}{dr} = -1$$

Podemos calcular cómo afecta la tasa de dilatación r a la salida de la interpolación s_1 como:

$$\begin{aligned}\frac{\partial s_1}{\partial r} &= \frac{\partial s_1}{\partial w_c}\frac{dw_c}{dr} + \frac{\partial s_1}{\partial w_f}\frac{dw_f}{dr} \\ &= a_1\,(1) + b_1\,(-1) \\ &= a_1 - b_1\end{aligned}$$

Finalmente, la derivada parcial de la pérdida respecto a la tasa de dilatación r es expresada como:

$$\begin{aligned}\frac{\partial L}{\partial r} &= \frac{\partial L}{\partial s_1}\frac{\partial s_1}{\partial r} \\ &= \frac{\partial L}{\partial s_1}(a_1 - b_1) \\ &= \frac{\partial L}{\partial s_1}a_1 - \frac{\partial L}{\partial s_1}b_1\end{aligned} \tag{3.4}$$

Como se mencionó, los valores a_1 y b_1 son valores conocidos de la entrada. Vemos entonces que podemos calcular la derivada parcial de la función de pérdida con respecto a la tasa de dilatación predicha para una determinada posición en un filtro. A su vez, siendo r la salida de la subred de predicción, podemos continuar propagando esta derivada hacia las capas de dicha subred, entrenando los parámetros de los filtros de manera que r se *mueva* de acuerdo al gradiente propagado de la pérdida. Puesto de otra manera, en cada paso de entrenamiento tenemos la capacidad de modificar los parámetros de la subred de predicción de manera de obtener una mejor tasa de dilatación r. Así, entrenamos la subred de predicción de dilataciones sin necesidad de datos adicionales, mediante la utilización de tasas continuas e interpolando linealmente posiciones adyacentes.

En la ecuación 3.4 podemos observar claramente cómo la derivada de la pérdida no depende de los pesos w_c y w_f, sino de los valores de entrada a_1 y b_1, con signo contrario. Los valores adyacentes de la entrada *competirán* por acercar o alejar la tasa r hacia su posición. Podemos ilustrar esta competencia entre los valores de entrada mediante un ejemplo. Consideremos que el gradiente propagado de la función de pérdida nos indica que la interpolación s_1 debe tener un valor menor al que se calculó. Esto es equivalente a decir que la derivada $\frac{\partial L}{\partial s_1}$ es positiva (incrementando s_1 incrementamos la pérdida). Esta derivada se encuentra en ambos términos de la ecuación 3.4, por lo que:

$$\frac{\partial L}{\partial r} > 0, \qquad \text{si } a_1 > b_1$$
$$\frac{\partial L}{\partial r} < 0, \qquad \text{si } b_1 > a_1$$

De esta manera, si $a_1 > b_1$ la derivada de la pérdida respecto a la tasa r será también positiva, indicando que debemos disminuir r para disminuir la pérdida. Disminuyendo r nos acercamos hacia la posición del valor b_1, el cual era el menor entre ambos valores de la entrada, otorgándole más peso y efectivamente disminuyendo el valor interpolado s_1, como indicaba el gradiente de la pérdida originalmente.

Durante el desarrollo previo consideramos únicamente lo que sucede para la posición superior izquierda del filtro, y pensando en una única aplicación de dicho filtro en la matriz/tensor de entrada. Generalizando para el resto de las posiciones del filtro, la derivada de la pérdida respecto a la tasa de dilatación predicha será la suma de las contribuciones en cada una de las posiciones de acuerdo a la ecuación 3.4. Esta derivada representará la contribución de la aplicación del filtro en una única posición espacial de la entrada. Sin embargo, al tratarse de una operación de convolución, el filtro será aplicado en múltiples posiciones, por lo que las capas de la subred de predicción de dilataciones serán entrenadas de acuerdo al gradiente de la pérdida respecto a la totalidad de las tasas predichas para cada posición, es decir, de acuerdo a la suma de las contribuciones del mapa de dilataciones.

3.3.2.3. Intervalo de Movimiento

El único hiperparámetro de una capa de Convolución de Dilatación Adaptativa que debemos definir estáticamente es el intervalo $[r_{inf}, r_{sup}]$ de movimiento de la tasa de dilatación. Si bien $r_{inf}, r_{sup} \in \mathbb{R}_{\geq 0}$, asumiremos la utilización de extremos enteros.

Podemos considerar el intervalo más simple, donde la distancia $r_{sup} - r_{inf} = 1$. En este ejemplo, en la totalidad de las tasas de dilatación r utilizadas se cumplirá $r_{inf} = \lfloor r \rfloor$ y $r_{sup} = \lceil r \rceil$, por lo que la capa se reduce a la interpolación lineal entre dos capas con tasas fijas r_{inf} y r_{sup}. En este caso, su utilización puede ser en detrimento del rendimiento de la red: estamos definiendo explícitamente dos capas de dilataciones diferentes (que comparten parámetros), pero solo utilizaremos una estadística (la suma pesada) de la información que ambas brindan. Puesto de otra manera, estamos quitando capacidad al modelo, a pesar de requerir los recursos de memoria de un modelo de mayor capacidad.

La capa de Convolución de Dilatación Adaptativa brinda mayor flexibilidad cuando consideramos intervalos mayores, esto es, cuando $r_{sup} - r_{inf} \geq 2$. En estos casos, la cantidad de memoria requerida aún es equivalente a la de dos capas convolucionales convencionales, pero obtenemos la flexibilidad de múltiples tasas de dilatación.

Si quisiéramos considerar las distintas tasas de dilatación enteras de este intervalo utilizando capas de dilatación tradicionales, deberíamos definir explícitamente cada una de ellas. Esto introduciría un número de parámetros equivalente al de $r_{sup} - r_{inf} + 1$ capas convolucionales. Sin embargo, cada una de las tasas de dilatación consideradas es aplicada en cada posición espacial, a pesar de no tratarse de la tasa óptima o no aportar información relevante al momento de realizar la predicción en las capas subsiguientes. En contrapartida, una capa de Dilatación Adaptativa permite utilizar diferentes dilataciones, manteniendo constante la cantidad de memoria utilizada a pesar de un incremento en el intervalo de movimiento de las tasas. Puesto de otra manera, la cantidad de recursos necesarios en capas de convolución dilatada tradicional crece linealmente con el número de tasas consideradas, mientras que permanece constante en el tipo de capa propuesta.

3.3.3. Convolución de Dilatación Aleatoria

Durante el tratado de los datos de entrenamiento, es una práctica habitual *aumentar* los mismos mediante diferentes transformaciones de la imagen para alterarla y multiplicar la cantidad de muestras de entrenamiento. Dicha alteración es realizada de manera que la etiqueta de la imagen siga siendo aplicable. En problemas de Segmentación Semántica, por ejemplo, se aplican las mismas transformaciones tanto en la imagen como en la etiqueta (en ese caso, mapa de clases por píxel).

Una transformación habitual consiste en escalar la imagen de manera de atacar la

limitación de las Redes Convolucionales ante las diferentes escalas de los objetos presentes en las muestras. De esta manera, se busca aumentar la invariancia a escala del modelo introduciendo las transformaciones en el conjunto de datos, pero no en el diseño de la red.

Por otra parte, si pensamos en el campo receptivo de la Convolución Dilatada, este se encuentra relacionado con la escala de los objetos presentes en la imagen o mapa de características de entrada. Reducir la escala de una imagen manteniendo la estructura de los filtros convolucionales es similar a mantener la escala de la imagen, pero esta vez aumentando la dilatación de los filtros proporcionalmente. Si aplicamos este razonamiento, estaríamos evitando la necesidad de multiplicar los datos de entrada ante transformaciones aleatorias de escala, esta vez modificando el diseño mismo de la arquitectura de la red convolucional de manera que se apliquen tasas de dilatación aleatorias.

La capa de **Convolución de Dilatación Aleatoria** es un tipo de Convolución de Dilatación Dinámica en la que se utiliza una tasa de dilatación aleatoria para cada posición espacial generada dinámicamente. Mediante esta estrategia podemos pensar que en cada paso de entrenamiento tenemos una imagen que es transformada a escala de manera independiente para cada posición espacial. Con el transcurso de los pasos de entrenamiento, una red que incluya una capa de Convolución de Dilatación Aleatoria tiende a considerar la totalidad de las combinaciones posibles de transformaciones de escala, aumentando su invariancia ante dicha transformación.

Otra interpretación de la aplicación de tasas aleatorias es como alternativa a la inclusión de la *Atrous Spatial Pyramid Pooling* propuesta en la red *DeepLabv2* [CPK+17]. Es decir, reemplazamos un conjunto de capas paralelas de Convolución Dilatada con diferentes tasas de dilatación fijas por una única capa de Convolución de Dilatación Aleatoria que utiliza tasas de dilatación dinámicas, forzando a la red a aprender información multiescala de cada muestra de entrenamiento. Adicionalmente, en caso de ser aplicable debido a conocimiento previo que pueda ser deducido del conjunto de datos, la frecuencia de utilización de las diferentes tasas de dilatación podría ser modulada, lo que se corresponde con introducir un sesgo para ciertas escalas. En este trabajo de tesis asumiremos tasas de dilatación aleatorias tomadas de una distribución uniforme.

Capítulo 4

Evaluación y Casos de Estudio

En este capítulo se analizan experimentalmente las propuestas realizadas en el Capítulo 3. Las evaluaciones se realizan para las tareas predictivas de **Clasificación de Imágenes** (predicción de la clase a la que pertenece la imagen) y **Segmentación Semántica** (predicción densa por píxel). Los casos de estudio se evaluaron mediante diferentes conjuntos de datos, dependiendo de la información de etiquetado (*ground truth*) disponible para cada uno de ellos.

4.1. Clasificación de Imágenes

En esta sección se estudian las diferencias en rendimiento al incluir o no las modificaciones propuestas en el Capítulo 3 para la tarea predictiva de **Clasificación de Imágenes**. Para realizar un estudio comparativo se optó por tomar el bloque de capas más utilizado en arquitecturas de redes de Aprendizaje Profundo: el bloque con conexión residual de redes tipo ResNet [HZRS16a]. Utilizando e interconectando varios de estos bloques, se crearon diferentes arquitecturas y múltiples configuraciones para cada una de ellas. A continuación se detalla el procedimiento utilizado y los resultados obtenidos.

4.1.1. Conjunto de Datos

Para la evaluación se utilizó el conjunto de datos estándar para clasificación **CIFAR-10** [KH+09]. Consta de 60.000 imágenes RGB de 32x32 píxeles divididas en 10 clases, con 6.000 imágenes por clase. Las clases son disjuntas (i.e. una imagen pertenece a una única clase).

Se utilizó la presentación sugerida por los autores, con 50.000 imágenes previamente apartadas para formar el subconjunto de entrenamiento y las 10.000 imágenes restantes para el subconjunto de validación. Ambos subconjuntos se encuentran balanceados con respecto a la cantidad de imágenes incluidas por cada clase.

4.1.2. Métrica de Evaluación

La métrica utilizada para evaluar el rendimiento de los entrenamientos es *exactitud*, definida como:

$$exactitud = \frac{predicciones\ correctas}{total\ de\ predicciones}$$

Resulta natural utilizar esta métrica en problemas de Clasificación de Imágenes ya que cuantifica de manera binaria lo que la última capa de predicción de las redes modela como distribución probabilística. Esta métrica es adecuada para conjuntos de datos balanceados, condición respetada por el conjunto de datos **CIFAR-10**. De lo contrario, su valor presenta un sesgo por la sobrerepresentación de las clases mayoritarias.

4.1.3. Arquitecturas Utilizadas

ResNet (del inglés *Residual Net*, Red Residual) es una arquitectura muy utilizada en el campo del Aprendizaje Profundo, introducida originalmente por He *et al.* [HZRS16a].

El artículo introduce un **bloque residual** que consta de una serie de capas convolucionales y funciones de activación ReLU, con la particularidad de que la salida de la última capa es sumada a la entrada al bloque. Esto permite, entre otras cosas, que el gradiente se propague por caminos paralelos, donde uno de los caminos recorre un menor número de capas, *saltando* entre bloques y evitando el problema de desvanecimiento del gradiente antes de alcanzar las capas cercanas a la entrada.

Los bloques residuales introdujeron mejoras inmediatas para el entrenamiento de las redes y han sido objeto de numerosas modificaciones en los últimos años, produciendo trabajos en los que se introducen diferentes versiones de los mismos. Para la presente evaluación experimental se optó por utilizar la variante con pre-activación propuesta por los autores del artículo original [HZRS16b].

En la literatura se definen comúnmente diferentes arquitecturas de redes residuales dependiendo del número de bloques empleados en serie. Cada bloque, a su vez, consta de

2 capas convolucionales. Así, el número total de capas convolucionales de una red ResNet es igual a 2 veces el número de bloques utilizados. En este trabajo de tesis se utilizan las arquitecturas **ResNet14** y **ResNet20**, las cuales difieren según el número de bloques residuales, siendo 6 bloques en el caso de ResNet14 y 9 bloques en el caso de ResNet20.

Todas las arquitecturas detalladas a continuación utilizan una primera capa convolucional aplicada a la entrada y una última capa Totalmente Conectada. Estas dos capas son las que, sumadas a las capas de los bloques residuales, le dan el nombre a cada una de las redes ResNet. Adicionalmente, para reducir las dimensiones de la entrada a la capa totalmente conectada se utiliza una capa de *average pooling*.

De manera de comparar cuantitativamente las definiciones y propuestas desarrolladas en el Capítulo 3, se incluyeron distintas modificaciones a la arquitectura ResNet original. Como se mencionó en dicho capítulo, las Capas de Convolución de Dilatación Adaptativa y Aleatoria pueden ser incluidas como módulos en arquitecturas arbitrarias de Redes Convolucionales, lo que permite realizar una observación directa de las diferencias en rendimiento que puedan introducir. Las modificaciones introducidas se realizaron:

- Agregando una capa convolucional adicional previa a la capa de *average pooling*. La capa utilizada es una Convolución Separable [SM14], de manera de minimizar el número de parámetros nuevos introducidos. Esta arquitectura funcionará como control y la denominaremos *ResNet Convencional*.

- Agregando una capa de Convolución de Dilatación Adaptativa. La operación de convolución dentro de la capa de Dilatación Adaptativa es también Separable, de manera de mantener el mismo número de parámetros en ambas arquitecturas y asegurar que las diferencias en rendimiento no sean introducidas por este factor. La tasa de dilatación puede tomar dinámicamente un valor continuo en el intervalo [1, 4]. Esta arquitectura pretende evaluar la propuesta realizada en la Sección 3.3.2 del Capítulo 3 y la denominaremos *ResNet con Tasas Adaptativas*.

- Agregando una capa de Convolución de Dilatación Aleatoria, como se propone en la Sección 3.3.3. La operación de convolución dentro de la capa de Dilatación Aleatoria es también Separable, al igual que en los otros casos. Las tasas de dilatación aleatorias pueden tomar valores enteros dentro del conjunto {1, 2, 3, 4}. La distri-

bución probabilística utilizada para el muestreo de las tasas aleatorias es uniforme. Esta arquitectura pretende evaluar las propuestas realizadas en la Sección 3.3.3 del Capítulo 3 y la denominaremos *ResNet con Tasas Aleatorias*.

Por otra parte, las tres arquitecturas fueron entrenadas bajo diferentes configuraciones. Al tratarse de redes ResNet, se incluyeron configuraciones adicionales con conexiones residuales en los módulos propuestos de manera de contar con resultados experimentales para arquitecturas formadas exclusivamente por módulos con este tipo de conexiones:

- ResNet Convencional sin conexión residual, i.e., la capa de convolución separable se encuentra en serie con el resto de las capas.
- ResNet Convencional con conexión residual, i.e., la salida de la capa de convolución separable es adicionada a su entrada.
- ResNet con Tasas Adaptativas sin conexión residual.
- ResNet con Tasas Adaptativas con conexión residual.
- ResNet con Tasas Aleatorias sin conexión residual.
- ResNet con Tasas Aleatorias con conexión residual.

4.1.4. Metodología de Entrenamiento

Para cada una de las arquitecturas y configuraciones detalladas se realizaron 5 entrenamientos diferentes partiendo de una inicialización aleatoria en cada uno de ellos, de manera de evaluar los rendimientos minimizando las variaciones que puedan darse debido a la naturaleza estocástica de la inicialización de los parámetros. Observaremos entonces el rendimiento promedio sobre los 5 entrenamientos para cada configuración, aunque también podremos observar el intervalo entre los valores mínimo y máximo de acuerdo a la métrica utilizada.

Para finalizar el entrenamiento en cada caso, se siguió una política de *early stopping*. La misma consiste en finalizar el entrenamiento del modelo si el rendimiento del mismo sobre el conjunto de datos de validación no mejora luego de un cierto número n de épocas, siendo n un hiperparámetro de la política de entrenamiento. Para la totalidad de los entrenamientos llevados a cabo se estableció el valor habitual de $n = 5$.

Por otra parte, con los primeros entrenamientos realizados para establecer la metodología a aplicar, se observó que al superarse las primeras 20 épocas de entrenamiento, de manera aproximada, el rendimiento de los modelos sobre el conjunto de datos de evaluación se estabilizaba si no se modificaba la tasa de aprendizaje. Se decidió, entonces, conducir la totalidad de los entrenamientos de manera que la tasa de aprendizaje del modelo sea disminuida en un orden de magnitud al finalizarse la época de entrenamiento número 25. Esto produce una visible mejora en el rendimiento desde la época 26, observada en la totalidad de los resultados presentados a continuación.

Los parámetros son inicializados de manera aleatoria según la *Inicialización He* [HZRS15]. El peso del término para el error de regularización en la función de pérdida global para todos los entrenamientos es de 0,002. Adicionalmente, se hace uso de la técnica de *Batch-Normalization*.

Comenzando en la siguiente página, pueden verse los resultados experimentales obtenidos, comparando cada una de las arquitecturas propuestas contra la arquitectura control, en sus distintas configuraciones, según corresponda.

4.1.5. Resultados Experimentales

Se presentan los resultados para las redes ResNet14 y ResNet20 en sus distintas configuraciones, comparados con las configuraciones convencionales para cada arquitectura.

ResNet con Tasas Adaptativas

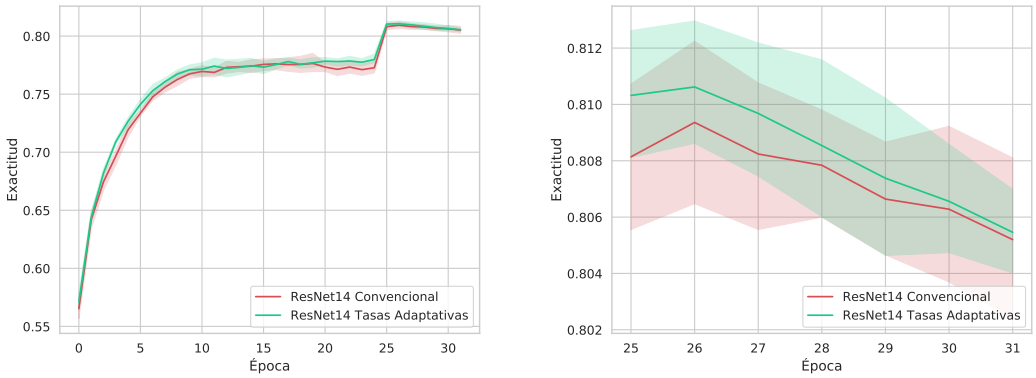


Figura 4.1: Comparación para CIFAR-10, ResNet14 con Tasas Adaptativas vs. ResNet14 Convencional, (a) totalidad de las épocas de entrenamiento hasta la aplicación de *early-stopping*, (b) visualización aumentada a partir de la época 25, donde se alcanza la mayor *exactitud*.

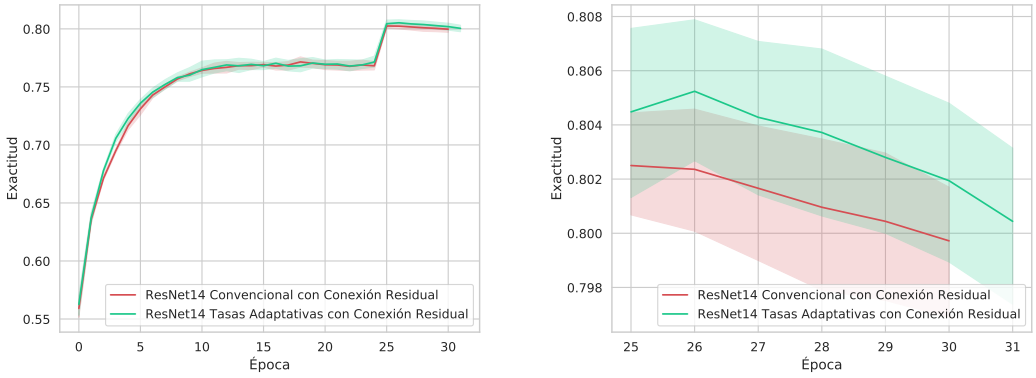
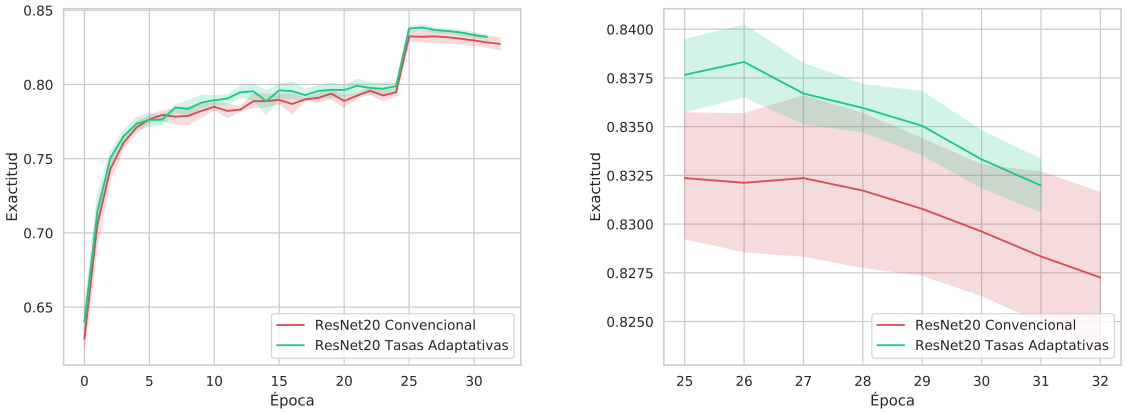
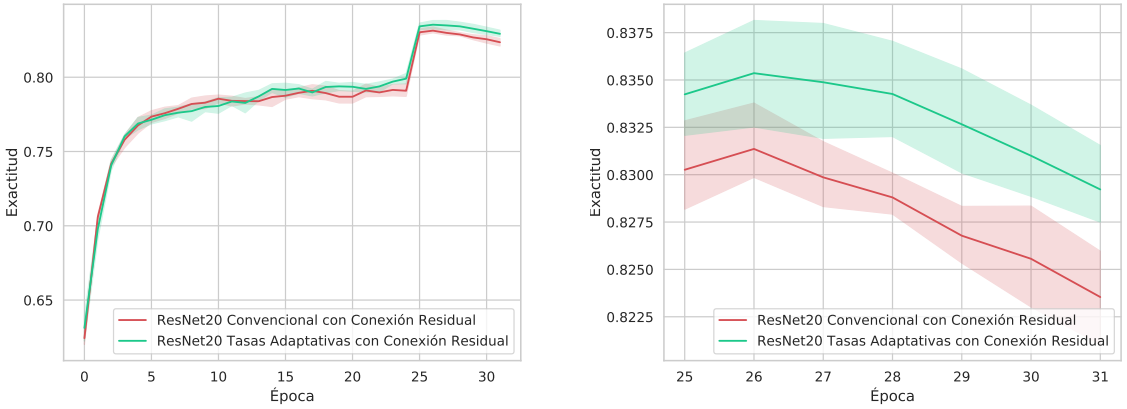


Figura 4.2: Comparación para CIFAR-10, ResNet14 con Tasas Adaptativas vs. ResNet14 Convencional, ambas con conexión residual en la última capa, (a) totalidad de las épocas de entrenamiento hasta la aplicación de *early-stopping*, (b) visualización aumentada a partir de la época 25, donde se alcanza la mayor *exactitud*.



(a) **(b)**

Figura 4.3: Comparación para CIFAR-10, ResNet20 con Tasas Adaptativas vs. ResNet20 Convencional, (a) totalidad de las épocas de entrenamiento hasta la aplicación de *early-stopping*, (b) visualización aumentada a partir de la época 25, donde se alcanza la mayor *exactitud*.



(a) **(b)**

Figura 4.4: Comparación para CIFAR-10, ResNet20 con Tasas Adaptativas vs. ResNet20 Convencional, ambas con conexión residual en la última capa, (a) totalidad de las épocas de entrenamiento hasta la aplicación de *early-stopping*, (b) visualización aumentada a partir de la época 25, donde se alcanza la mayor *exactitud*.

ResNet con Tasas Aleatorias

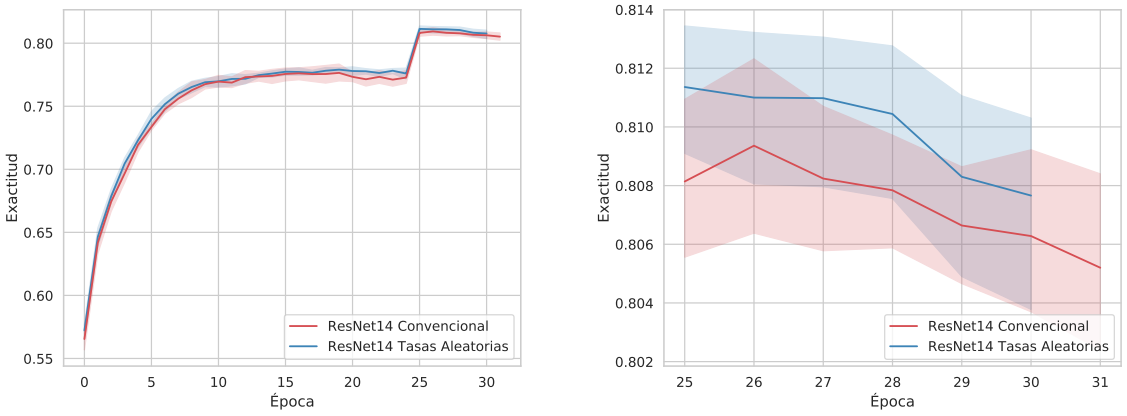


Figura 4.5: Comparación para CIFAR-10, ResNet14 con Tasas Aleatorias vs. ResNet14 Convencional, (a) totalidad de las épocas de entrenamiento hasta la aplicación de *early-stopping*, (b) visualización aumentada a partir de la época 25, donde se alcanza la mayor *exactitud*.

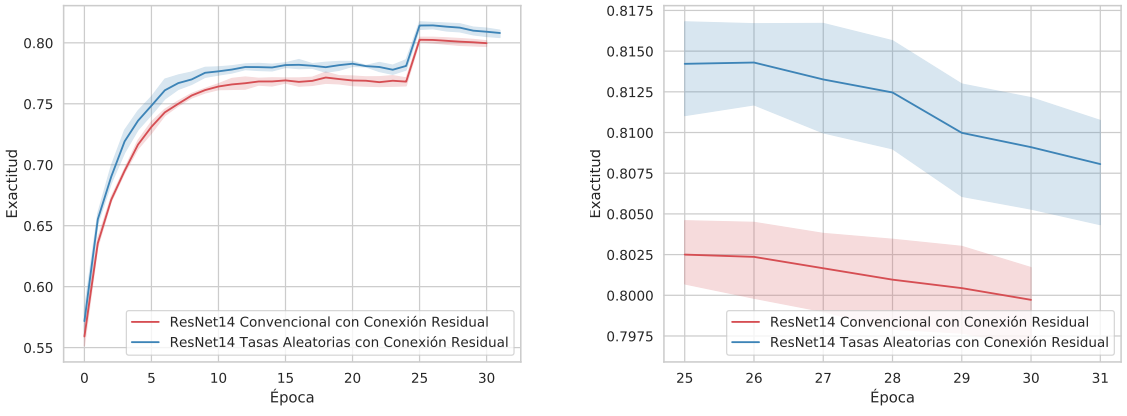
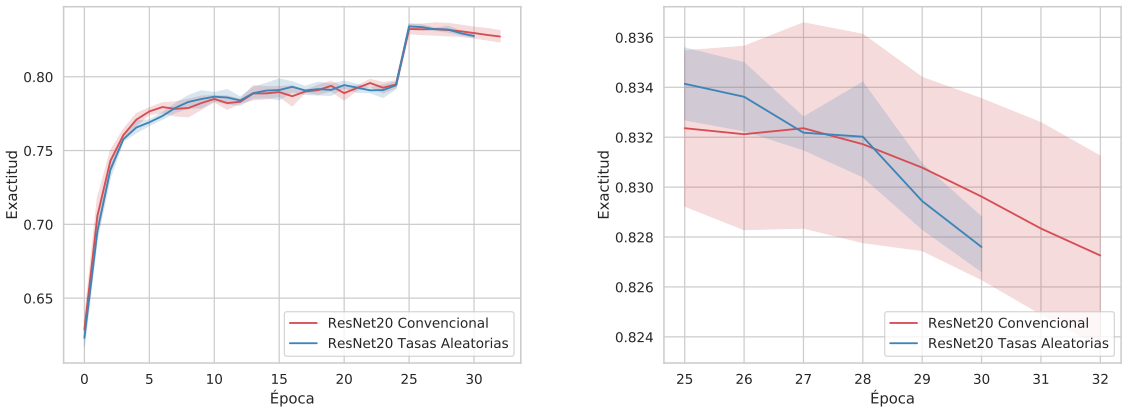
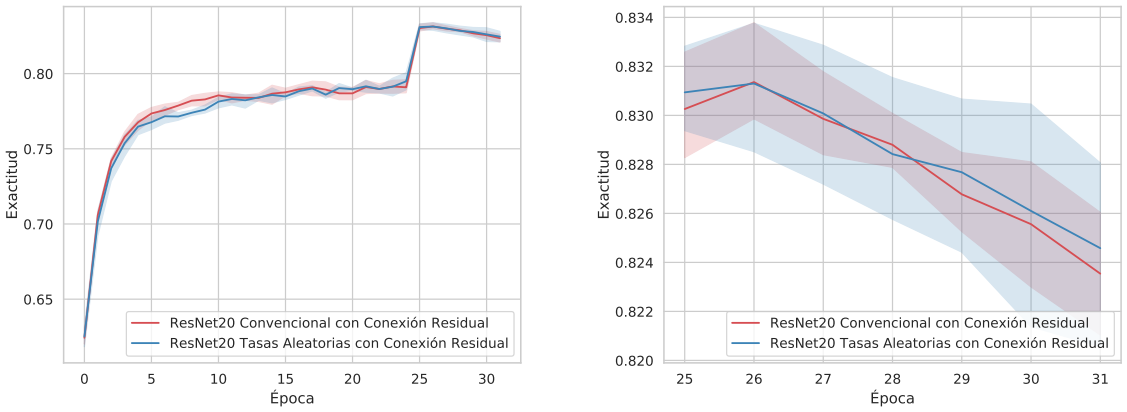


Figura 4.6: Comparación para CIFAR-10, ResNet14 con Tasas Aleatorias vs. ResNet14 Convencional, ambas con conexión residual en la última capa, (a) totalidad de las épocas de entrenamiento hasta la aplicación de *early-stopping*, (b) visualización aumentada a partir de la época 25, donde se alcanza la mayor *exactitud*.



(a) **(b)**

Figura 4.7: Comparación para CIFAR-10, ResNet20 con Tasas Aleatorias vs. ResNet20 Convencional, (a) totalidad de las épocas de entrenamiento hasta la aplicación de *early-stopping*, (b) visualización aumentada a partir de la época 25, donde se alcanza la mayor *exactitud*.



(a) **(b)**

Figura 4.8: Comparación para CIFAR-10, ResNet20 con Tasas Aleatorias vs. ResNet20 Convencional, ambas con conexión residual en la última capa, (a) totalidad de las épocas de entrenamiento hasta la aplicación de *early-stopping*, (b) visualización aumentada a partir de la época 25, donde se alcanza la mayor *exactitud*.

4.1.6. Discusión de los Resultados para Clasificación de Imágenes

En las distintas figuras se puede observar cómo la media de los valores de *exactitud* de las configuraciones que incluyen las propuestas realizadas en el Capítulo 3 presentan

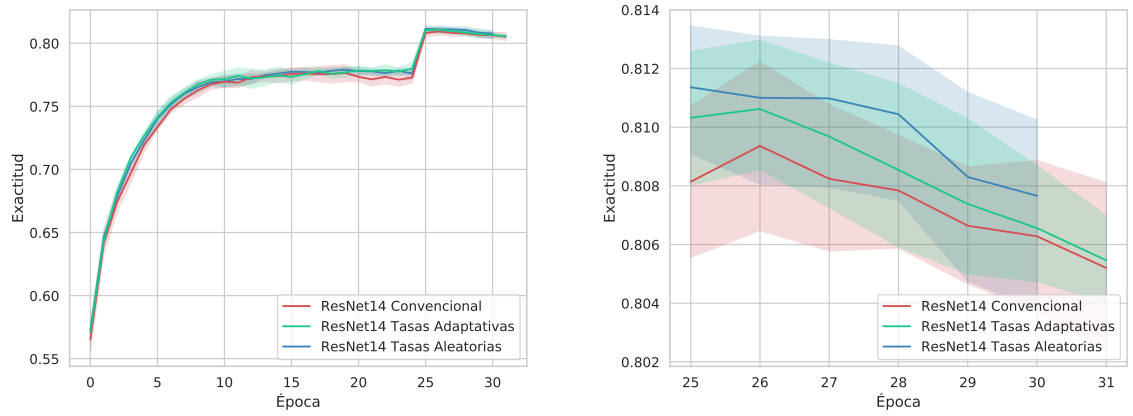
una mejora respecto a la media de las configuraciones convencionales.

Intuitivamente, esperamos que en todos los casos el rendimiento de la configuración de Tasas Adaptativas sea mayor al de Tasas Aleatorias, por ser el primero un proceso de optimización de las tasas de dilatación, mientras el segundo un proceso aleatorio. Sin embargo, como se observa en las figuras 4.9 y 4.10, para la red ResNet14 observamos el caso inverso, donde el rendimiento es superior para la configuración de Tasas Aleatorias. Una posible interpretación es que la configuración de Tasas Aleatorias funcione efectivamente como un proceso de *generalización*, como se detalló en la Sección 3.3.3 del Capítulo 3, donde la Capa de Convolución de Dilatación Aleatoria equivale en el límite a todas las posibles combinaciones de tasas de dilatación. Si bien la combinatoria de los distintos mapas de tasas de dilatación es órdenes de magnitud superior a la cantidad de mapas diferentes que podemos obtener en cualquier entrenamiento finito en la práctica, este efecto puede ser el que introduzca las mejoras observables respecto a la configuración de Tasas Adaptativas. Al ser ResNet14 una red de unas pocas capas, el número de combinaciones diferentes necesarias para producir una mejora es menor a dicho número en redes de mayor complejidad. Por este motivo, puede que el efecto de *generalización* ante diferentes escalas ejerza una influencia mayor en redes pequeñas, pero que su influencia se desvanezca a medida que aumentamos el número de capas, pudiendo ser interpretado como un efecto de regularización en redes de mayor complejidad.

Para la red ResNet20, en las figuras 4.7 y 4.8 vemos que las configuraciones de Tasas Aleatorias tienen un rendimiento similar a las configuraciones convencionales. Adicionalmente, en las figuras 4.11 y 4.12 donde se compara las configuraciones propuestas y las configuraciones convencionales, vemos el comportamiento esperado en el que las configuraciones de Tasas Adaptativas son superiores a las configuraciones tanto convencionales como de Tasas Aleatorias.

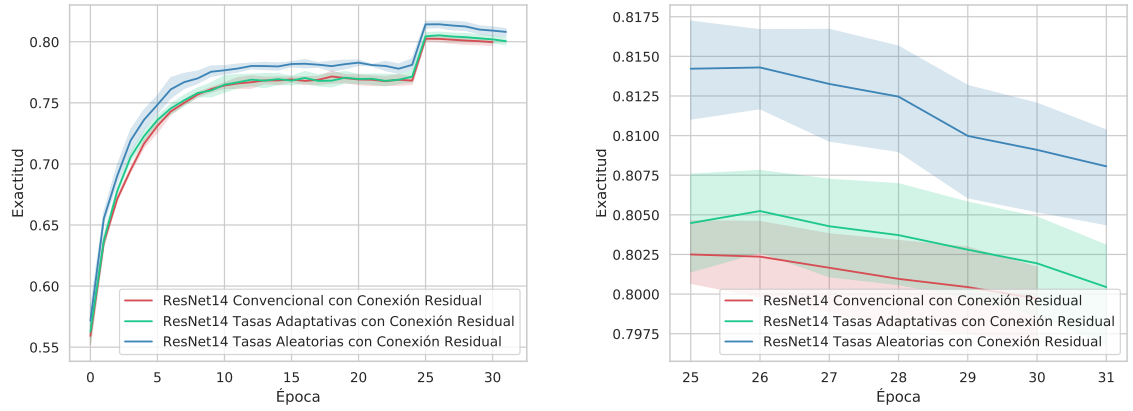
Ambas redes ResNet presentadas en la evaluación fueron diseñadas intencionalmente con un número pequeño de capas. El objetivo de la presente sección no es el de maximizar la métrica de evaluación para Clasificación de Imágenes hacia métricas comparables a la de los diferentes *benchmarks* existentes, sino el de realizar un estudio comparativo de cómo las propuestas introducidas influencian la capacidad de los modelos no-lineales, manteniendo a estos últimos lo suficientemente simples de manera que podamos concluir que las discrepancias en rendimiento son introducidas por las diferencias en las configu-

raciones entre los modelos que se comparan.



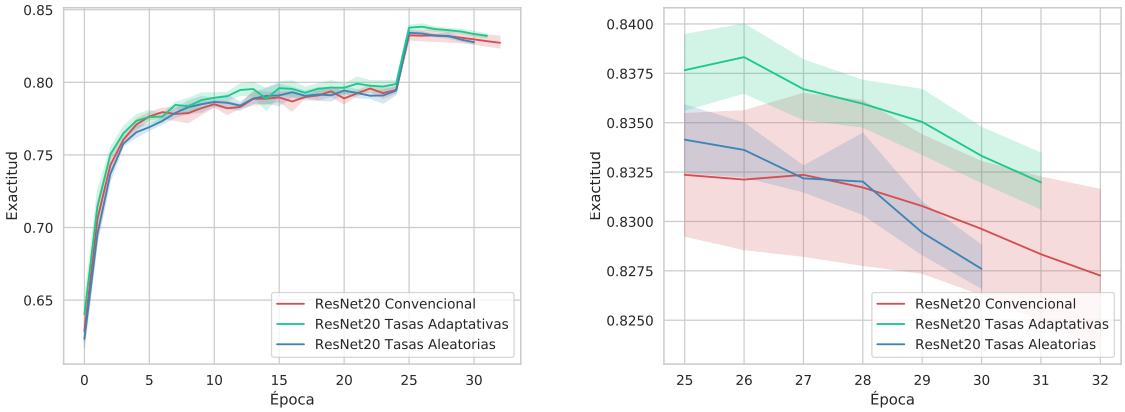
(a) **(b)**

Figura 4.9: Comparación para CIFAR-10, ResNet14 con Tasas Aleatorias vs. ResNet14 Convencional vs. ResNet14 con Tasas Adaptativas, (a) totalidad de las épocas de entrenamiento hasta la aplicación de *early-stopping*, (b) visualización aumentada a partir de la época 25, donde se alcanza la mayor *exactitud*.



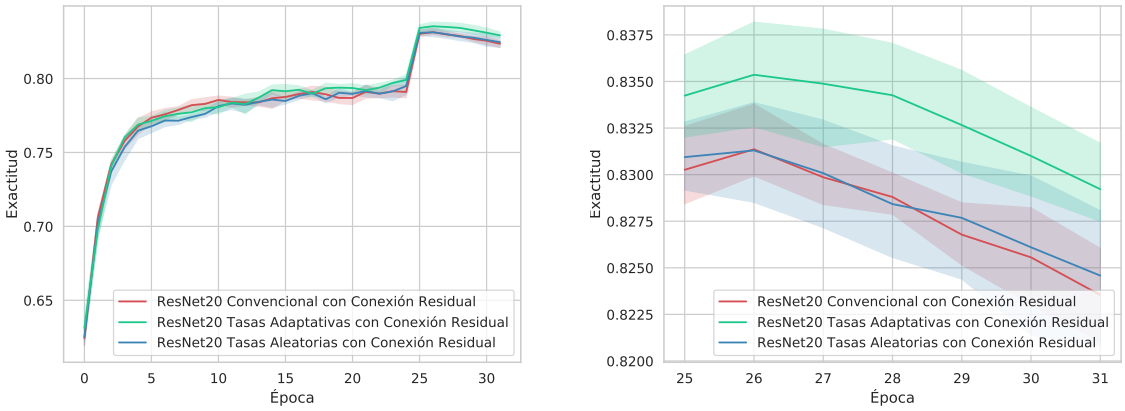
(a) **(b)**

Figura 4.10: Comparación para CIFAR-10, ResNet14 con Tasas Aleatorias vs. ResNet14 Convencional vs. ResNet14 con Tasas Adaptativas, todas con conexión residual en la última capa, (a) totalidad de las épocas de entrenamiento hasta la aplicación de *early-stopping*, (b) visualización aumentada a partir de la época 25, donde se alcanza la mayor *exactitud*.



(a) **(b)**

Figura 4.11: Comparación para CIFAR-10, ResNet20 con Tasas Aleatorias vs. ResNet20 Convencional vs. ResNet20 con Tasas Adaptativas, (a) totalidad de las épocas de entrenamiento hasta la aplicación de *early-stopping*, (b) visualización aumentada a partir de la época 25, donde se alcanza la mayor *exactitud*.



(a) **(b)**

Figura 4.12: Comparación para CIFAR-10, ResNet20 con Tasas Aleatorias vs. ResNet20 Convencional vs. ResNet20 con Tasas Adaptativas, todas con conexión residual en la última capa, (a) totalidad de las épocas de entrenamiento hasta la aplicación de *early-stopping*, (b) visualización aumentada a partir de la época 25, donde se alcanza la mayor *exactitud*.

4.2. Segmentación Semántica

De manera de introducir las propuestas realizadas en este trabajo de tesis en una aplicación de mayor complejidad, se entrenó la red **Deeplabv3+** [CZP+18] para la tarea predictiva de **Segmentación Semántica**, tanto en su versión original como incluyendo las modificaciones que se proponen en el Capítulo 3.

La Segmentación Semántica consiste en predecir la clase a la que pertenece cada uno de los píxeles de la imagen de entrada. De esta manera, la predicción de la red será una matriz de las mismas dimensiones espaciales que la imagen. Un píxel pertenece a una única clase de todas las posibles.

4.2.1. Conjunto de Datos

La evaluación se realizó sobre el conjunto de datos de imágenes urbanas para predicción semántica **Cityscapes** [COR+16], ampliamente utilizado en algoritmos para vehículos autónomos. La etiqueta para cada imagen es representada por una matriz de las mismas dimensiones espaciales que la imagen, donde el valor para cada elemento representa la clase a la que pertenece el píxel asociado. Consta de 5.000 imágenes con etiquetas finas, donde la delineación de los objetos presentes es precisa, y 20.000 imágenes adicionales con etiquetado grueso, donde la delineación es aproximada. Para la evaluación experimental se utilizaron las imágenes con etiquetado fino.

4.2.2. Métrica de Evaluación

La métrica utilizada para evaluar el rendimiento de los entrenamientos es el *Índice de Jaccard*, también denominada *Intersección sobre la Unión*.

En el caso de predicción densa por píxel, podemos pensar que tenemos una predicción independiente para cada una de las clases de objetos presentes en las imágenes. Tenemos entonces, para cada clase, dos matrices binarias de las dimensiones espaciales de la imagen: una indicando aquellos píxeles que se predijeron como pertenecientes a la clase y otra que representa la información verdadera o de *ground truth*, indicando aquellos píxeles que realmente pertenecen a la clase. Denominando *conjunto P* a los píxeles de la predicción y *conjunto V* a los píxeles verdaderos, definimos el Índice de Jaccard para una clase c

como:

$$J_c = \frac{P_c \cap V_c}{P_c \cup V_c}.$$

Podemos ver claramente cómo la métrica representa el número de píxeles predichos que se intersecan con los verdaderos, divididos por el número total de píxeles diferentes representados. En el caso de una predicción perfecta, la intersección será exacta, y la unión se hará entre dos grupos que contienen los mismos elementos, por lo que numerador y denominador serán iguales, dando en ese caso un índice igual a 1. Para el caso de una predicción completamente errónea, la intersección será vacía, por lo que el numerador será igual a 0, y por consiguiente también el índice.

La métrica de evaluación J para cada imagen se calcula obteniendo la suma total de píxeles, para la totalidad de clases, tanto para la intersección como para la unión, y calculando la relación entre dichas sumas:

$$J = \frac{\sum_c P_c \cap V_c}{\sum_c P_c \cup V_c}$$

Para evaluar un modelo sobre el conjunto de datos, utilizaremos la media del Índice de Jaccard sobre todas las imágenes de evaluación i, métrica denominada $mIoU$ (*mean Intersection over Union*):

$$mIoU = \frac{1}{N} \sum_{i=1}^{N} J_i$$

4.2.3. Metodología de Entrenamiento

Para cada una de las configuraciones detalladas se realizó un entrenamiento diferente utilizando la técnica de *fine-tuning*, inicializando los parámetros de la red con un conjunto de parámetros pre-entrenados para una tarea similar, de manera de evitar comenzar el entrenamiento con valores aleatorios tomados de una distribución probabilística. En los resultados presentados a continuación, las diferentes configuraciones parten del mismo pre-entrenamiento o *checkpoint* de la subred Xception [Cho17]. Deeplabv3+ es una ampliación de la red Xception, adicionando capas en las etapas finales. Estas realizan la predicción densa por píxel, recibiendo como entrada las salidas de diferentes capas en la subred Xception. El pre-entrenamiento utilizado es el incluido con la implementación original de la red en la librería TensorFlow [ABC+16]. Las capas adicionales a las

de la subred Xception son inicializadas de manera aleatoria según la *Inicialización He* [HZRS15].

La reutilización de entrenamientos es muy común al momento de entrenar redes de un gran número de parámetros, como sucede con *Deeplabv3+*, aprovechando entrenamientos realizados para otras tareas similares. El área que estudia la reutilización de entrenamientos entre diferentes tareas se denomina Transferencia de Aprendizaje (*Transfer Learning*).

Se respetó la estrategia original de Deeplabv3+ para la actualización de la tasa de aprendizaje durante el entrenamiento en problemas de Segmentación Semántica. Se determina un valor inicial para la misma y se disminuye linealmente con los pasos de entrenamiento hasta alcanzar un valor nulo. Para los entrenamientos realizados se estableció una tasa de aprendizaje inicial $\alpha = 0{,}01$. El peso del término para el error de regularización en la función de pérdida global es de 0,002. Se utilizó la técnica de *Batch-Normalization* como en el artículo original.

4.2.4. Configuraciones Utilizadas

Se comparan tres configuraciones:

- Deeplabv3+ Convencional. Esta arquitectura funcionará como control. La red Deeplabv3+ incluye originalmente tres capas convolucionales paralelas de diferentes tasas de dilatación (fijas), iguales a 6, 12 y 18.

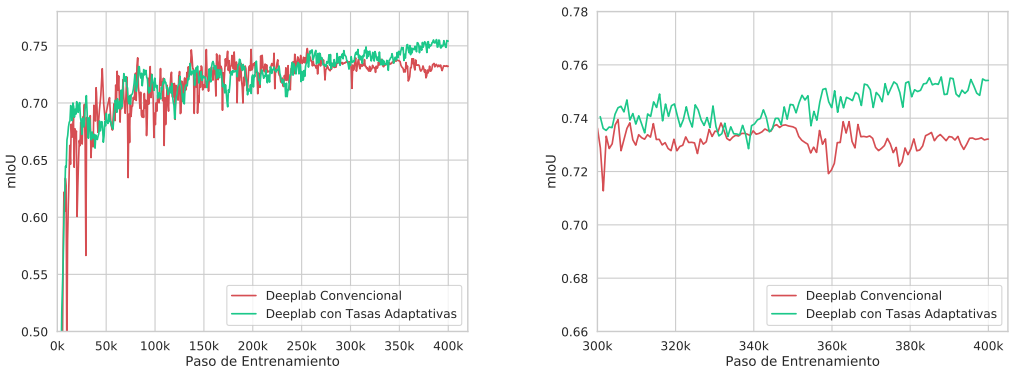
- Deeplabv3+ con Tasas Adaptativas. Las tasas de dilatación varían dinámicamente dentro del intervalo [6, 18], respetando los límites de las tasas de dilatación utilizadas en la configuración Convencional. Nótese la posibilidad de variar las tasas de dilatación dentro del intervalo continuo definido por las tasas de la configuración Convencional, pero en una sola capa en lugar de las tres capas paralelas originales.

- Deeplabv3+ con Tasas Aleatorias. Las tasas de dilatación aleatorias pueden tomar valores dentro del conjunto {6, 12, 18}. La distribución utilizada para el muestreo de valores aleatorias es uniforme. Nótese la utilización del mismo conjunto de tasas de dilatación que en la configuración Convencional, pero en una sola capa en lugar de las tres capas paralelas originales.

A continuación se muestran los resultados experimentales para las distintas configuraciones detalladas.

4.2.5. Resultados Experimentales

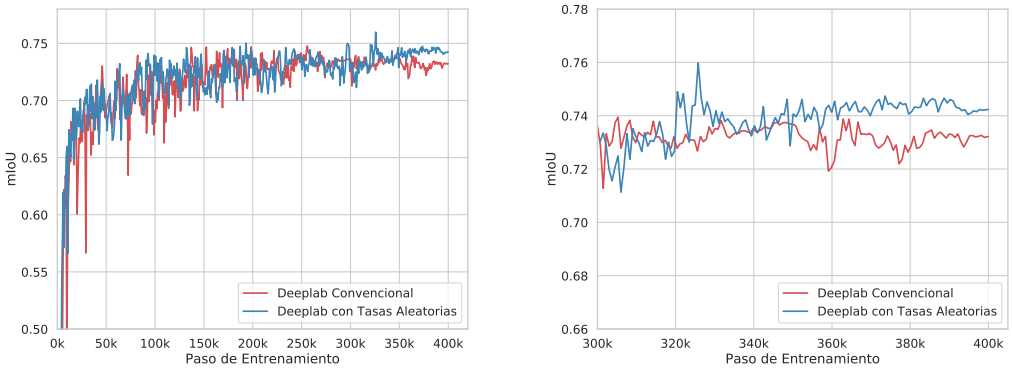
Deeplabv3+ con Tasas Adaptativas



(a) **(b)**

Figura 4.13: Comparación para Cityscapes, Deeplabv3+ con Tasas Adaptativas vs. Deeplabv3+ Convencional, (a) comparación para los 400k pasos de entrenamiento, (b) visualización aumentada para los últimos 100k pasos, donde el rendimiento máximo de ambas redes comienza a estabilizarse.

Deeplabv3+ con Tasas Aleatorias



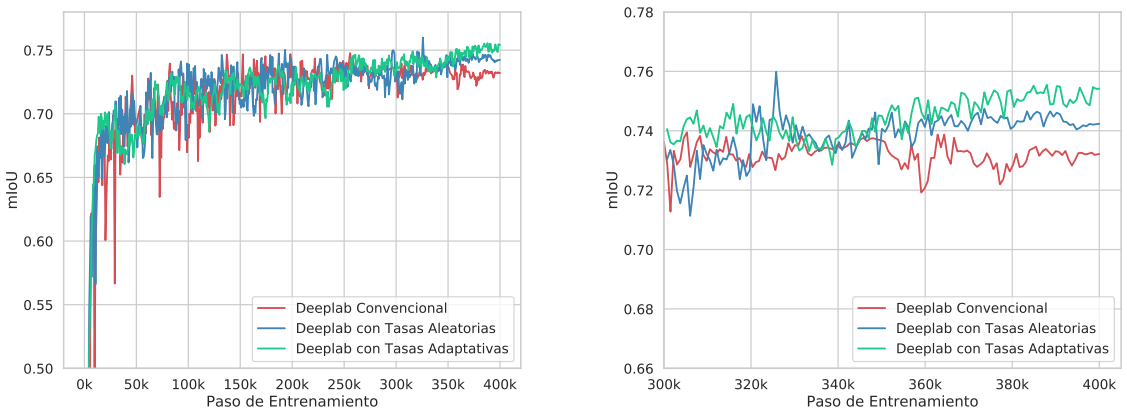
(a) **(b)**

Figura 4.14: Comparación para Cityscapes, Deeplabv3+ con Tasas Aleatorias vs. Deeplabv3+ Convencional, (a) comparación para los 400k pasos de entrenamiento, (b) visualización aumentada para los últimos 100k pasos, donde el rendimiento máximo de ambas redes comienza a estabilizarse.

4.2.6. Discusión de los Resultados para Segmentación Semántica

En las figuras 4.13 y 4.14 se puede observar claramente cómo los valores de la métrica de evaluación para las configuraciones que incluyen las propuestas realizadas en el Capítulo 3 se estabilizan en valores mayores a la configuración original. A su vez, podemos observar en la figura 4.15 una comparación incluyendo todas las configuraciones entrenadas, en la que podemos apreciar que el valor de dicha métrica para la configuración de Tasas Adaptativas supera a aquel de las Tasas Aleatorias, lo que es intuitivamente esperado en redes de gran complejidad como Deeplabv3+.

A diferencia de los entrenamientos realizados para las redes ResNet en la Sección 4.1, resulta más riesgoso afirmar que las mejoras de rendimiento que se observaron son consecuencia directa de las modificaciones introducidas. Una red como Deeplabv3+ representa un sistema no-lineal de gran complejidad, por lo que las alteraciones que introduce cualquier modificación resultan difíciles de predecir. Dicho de otra forma, las consecuencias intuitivas que se pueden derivar de las modificaciones introducidas pueden no ser las causantes directas de la mejora en el rendimiento. Esto se ha visto en la última década en varias contribuciones a redes de Aprendizaje Profundo, como en el caso del *Dropout*



(a) **(b)**

Figura 4.15: Comparación para Cityscapes, Deeplabv3+ con Tasas Aleatorias vs. Deeplabv3+ con Tasas Aleatorias vs. Deeplabv3+ Convencional, (a) comparación para los 400k pasos de entrenamiento, (b) visualización aumentada para los últimos 100k pasos, donde el rendimiento máximo de las redes comienza a estabilizarse.

[SHK+14], en donde una cierta técnica introduce una mejora, comprobada repetidas veces de manera empírica, aunque posteriormente se concluye que la mejora observada no era resultado de lo que intuitivamente se creyó como causa, sino que las alteraciones producidas indirectamente por la técnica estaban afectando al sistema de maneras no previstas y esas consecuencias eran las verdaderas causantes de la mejora observada.

No obstante, al realizar los entrenamientos comparativos en una red como Deeplabv3+, la intención es la de observar el comportamiento de las propuestas realizadas al ser introducidas en aplicaciones ampliamente utilizadas y conocidas en la literatura. Se adopta Deeplabv3+ como un caso de estudio, de manera de ilustrar que las modificaciones propuestas pueden ser rápidamente incluidas en redes complejas existentes, en problemas que son atacados tanto en la literatura académica como en implementaciones de aplicaciones comerciales.

Capítulo 5

Conclusiones y Trabajo Futuro

El área de la Visión Artificial ha experimentado un crecimiento acelerado en la década de 2010 a partir de trabajos que marcaron un cambio de paradigma en los enfoques utilizados y la adopción de las Redes Neuronales Convolucionales como el método estándar para procesamiento de información visual [LBH15].

Aumentar la capacidad de estos modelos es un factor importante para el avance del estado del arte. La forma trivial de hacerlo es incrementando la cantidad de parámetros, lo cual aumenta también el número de operaciones que deben realizarse tanto en entrenamiento como en inferencia, así como la memoria total requerida. Esta estrategia colisiona rápidamente con los límites de hardware y tiempo de ejecución, poniendo un techo a su escalabilidad. Por estos motivos, en los últimos años se han multiplicado los esfuerzos para dar a los modelos una mayor capacidad manteniendo el número de parámetros involucrados. El objetivo es la utilización de los parámetros de una forma óptima, aprovechando conocimiento previo que podamos tener del problema.

Cuando una Red Neuronal Convolucional procesa información visual, ciertas características deseables, como la invariancia traslacional, están dadas por la forma en que se utilizan los parámetros del modelo (los filtros comparten dichos parámetros a través de las posiciones espaciales de la entrada). Sin embargo, la invariancia ante otras transformaciones no es una característica presente en la construcción de estos modelos, siendo la variación ante escala una de sus limitaciones conocidas.

5.1. Síntesis de los Aportes de esta Tesis

En este trabajo de tesis hemos explorado modificaciones a los filtros convolucionales existentes con dos objetivos:

- Aumentar la capacidad de los modelos haciendo un uso más eficiente de sus parámetros, sin aumentar su cantidad.
- Construir los filtros convolucionales de manera de obtener menor variabilidad ante transformaciones de escala sobre los objetos de las imágenes de entrada.

A continuación se resumen los aportes de esta tesis a partir de los objetivos mencionados anteriormente:

- **Generalización de la Convolución Dilatada como Convolución de Dilatación Continua (CDC)**. La Convolución Dilatada tradicional permite otorgar flexibilidad en las dimensiones espaciales manteniendo la cantidad de parámetros, *dilatando* el filtro a distancias enteras para aumentar su campo receptivo. La definición propuesta extiende la dilatación a aplicar sobre el filtro a valores continuos.
- **Definición de la Convolución de Dilatación Dinámica (CDD)**. Esta propuesta define un mapa de tasas de dilatación para los filtros convolucionales de una capa. De esta manera, se atacan tres limitaciones de la Convolución Dilatada tradicional: el establecimiento de un valor fijo para las tasas de dilatación de una capa antes de comenzar el entrenamiento, la imposibilidad de aplicar diferentes dilataciones en diferentes posiciones espaciales de la entrada y la imposibilidad de utilizar diferentes dilataciones dependiendo de la imagen de entrada. El mapa de dilataciones propuesto puede ser definido para cada ejemplo del conjunto de datos antes de comenzar el entrenamiento, o puede ser definido durante el mismo de manera dinámica por otro método.
- **Definición de la Convolución de Dilatación Adaptativa**. De manera de poder definir de forma dinámica las dilataciones de los filtros para una capa, el tipo de capa convolucional propuesto predice las tasas de dilatación a partir de la información de entrada a la capa, las cuales se utilizarán para definir el mapa de dilataciones a utilizar por una capa de CDD. Así, la capa propuesta es un módulo conformado

por una Convolución Dinámica y una capa paralela de predicción del mapa de dilataciones. Esta última aprende a predecir las tasas de dilatación óptimas sin modificaciones adicionales al proceso de entrenamiento, haciendo uso de operaciones diferenciables y del algoritmo de *backpropagation*.

- **Definición de la Convolución de Dilatación Aleatoria**. Esta capa utiliza un mapa de dilataciones aleatorio, calculado para cada ejemplo de entrada de manera dinámica durante el entrenamiento. Se explora la posibilidad de que el uso de diferentes tasas de dilatación en diferentes posiciones espaciales aumente la invariancia ante escala del modelo, simulando transformaciones de escala de los objetos presentes en las imágenes sin la necesidad de modificar el conjunto de datos.

- **Evaluación de las capas convolucionales propuestas para casos de estudio de Clasificación de Imágenes y Segmentación Semántica**. Se modificaron arquitecturas conocidas en la literatura para diferentes problemas de Visión Artificial. Se realizaron entrenamientos de diferentes variantes de la arquitectura ResNet [HZRS16a] para Clasificación de Imágenes sobre el conjunto de datos CIFAR-10 [KH+09] y de la arquitectura DeepLabv3+ [CZP+18] para Segmentación Semántica sobre el conjunto de datos Cityscapes [COR+16]. Se observaron resultados favorables que indican la factibilidad del uso de estas propuestas para incrementar el rendimiento de estos modelos en los problemas mencionados. Más aún, los módulos propuestos pueden ser incluidos en arquitecturas existentes sin modificaciones adicionales a las redes.

5.2. Direcciones Futuras de Investigación

Las propuestas realizadas tienen la particularidad de involucrar definiciones generales, pudiendo ser exploradas en diferentes problemas y configuraciones. Se proponen las siguientes líneas de investigación futuras:

- **Evaluación en casos de estudio para otros problemas de Visión Artificial**. Las definiciones de capas convolucionales propuestas son generales y no se limitan a los problemas presentados en los casos de estudio. Se plantea como trabajo futuro la realización de entrenamientos comparativos con diferentes arquitecturas para otros

problemas, como Detección de Objetos y Segmentación por Instancias, que suelen incluir algunas de las arquitecturas más complejas de la literatura. En estos casos, se requerirá el uso del hardware apropiado para que las métricas de evaluación puedan ser comparadas con los resultados del estado del arte.

- **Extensión de las operaciones de Convolución de Dilatación Adaptativa y Aleatoria por canal**. Las propuestas presentadas extienden la operación de convolución permitiendo un cambio en la estructura geométrica del filtro para cada posición espacial de manera independiente. Sin embargo, esta independencia no se extiende a la tercera dimensión de los tensores de entrada, es decir, a los canales que representan las diferentes dimensiones de los vectores de características. El mapa de dilataciones propuesto para la Convolución de Dilatación Dinámica define una matriz de dos dimensiones que contiene una tasa de dilatación para cada posición espacial. Se plantea como trabajo futuro la extensión de este mapa a tres dimensiones, teniendo de esta manera múltiples tasas de dilatación por posición espacial, cada una de ellas asociada a un canal diferente del tensor de entrada. Esta modificación expandiría la capacidad de los filtros aún más, permitiendo que se disocien las operaciones realizadas en las diferentes dimensiones de los vectores de características, los cuales contienen abstracciones de alto nivel en capas profundas de la red.

- **Definición del mapa de dilataciones mediante supervisión externa**. Se ha experimentado con mapas de dilataciones predichos de manera dinámica por la misma red (Convolución de Dilatación Adaptativa) y de manera aleatoria (Convolución de Dilatación Aleatoria). Una tercera opción es utilizar información de *ground-truth* recolectada con otro objetivo y transformarla en un mapa de dilataciones. Por ejemplo, teniendo información de profundidad para cada píxel en la escena (distancia respecto a la cámara), podemos obtener de forma indirecta un mapa de dilataciones. En este caso, una mayor distancia a la cámara se traduce en un objeto más pequeño debido a la transformación perspectiva sufrida, por lo que podemos asignar una tasa de dilatación menor para mantener los elementos del filtro dentro de dicho objeto. De esta manera, un mapa de profundidades puede ser interpretado como un mapa (inverso) de dilataciones: a mayor profundidad, menor

tasa de dilatación, y viceversa. Podemos denominar a esta estrategia Convolución de Dilatación Dinámica con Supervisión Externa.

Esta página intencionalmente en blanco.

Bibliografía

[ABC+16] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: A system for large-scale Machine Learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)* (2016), pp. 265–283.

[BKC17] Badrinarayanan, V., Kendall, A., and Cipolla, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 39*, 12 (2017), 2481–2495.

[BM13] Bruna, J., and Mallat, S. Invariant Scattering Convolution Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence 35*, 8 (2013), 1872–1886.

[BTVG06] Bay, H., Tuytelaars, T., and Van Gool, L. SURF: Speeded Up Robust Features. In *European Conference on Computer Vision* (2006), Springer, pp. 404–417.

[Cho17] Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 1251–1258.

[COR+16] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. The Cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 3213–3223.

[CPK+14] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. Semantic image segmentation with Deep Convolutional Nets and Fully Connected CRFs. *arXiv preprint arXiv:1412.7062* (2014).

[CPK+17] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. Deeplab: Semantic image segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence 40*, 4 (2017), 834–848.

[CPSA17] Chen, L.-C., Papandreou, G., Schroff, F., and Adam, H. Rethinking Atrous Convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587* (2017).

[CUH15] Clevert, D.-A., Unterthiner, T., and Hochreiter, S. Fast and accurate Deep Network Learning by Exponential Linear Units (ELUs). *arXiv preprint arXiv:1511.07289* (2015).

[Cyb89] Cybenko, G. Approximations by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems 2* (1989), 183–192.

[CZP+18] Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. Encoder-decoder with Atrous Separable Convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 801–818.

[DDS+09] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (2009), Ieee, pp. 248–255.

[DHS11] Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research 12*, 7 (2011).

[DLJ+20] Du, X., Lin, T.-Y., Jin, P., Ghiasi, G., Tan, M., Cui, Y., Le, Q. V., and Song, X. SpineNet: Learning scale-permuted backbone for recognition

and localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 11592–11601.

[DQX+17] Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., and Wei, Y. Deformable Convolutional Networks. In *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 764–773.

[EEVG+15] Everingham, M., Eslami, S. A., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. The Pascal Visual Object Classes challenge: A retrospective. *International Journal of Computer Vision 111*, 1 (2015), 98–136.

[ESTA14] Erhan, D., Szegedy, C., Toshev, A., and Anguelov, D. Scalable Object Detection using Deep Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 2147–2154.

[FGMR09] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. Object Detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32*, 9 (2009), 1627–1645.

[Fuk80] Fukushima, K. Neocognitron: A self-organizing Neural Network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36 (1980), 193–202.

[Fuk88] Fukushima, K. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Networks*, 1 (1988), 119–130.

[GBB11] Glorot, X., Bordes, A., and Bengio, Y. Deep Sparse Rectifier Neural Networks. In *Proceedings of the fourteenth International Conference on Artificial Intelligence and Statistics* (2011), pp. 315–323.

[GD14] Gens, R., and Domingos, P. M. Deep Symmetry Networks. In *Advances in Neural Information Processing Systems* (2014), pp. 2537–2545.

[GDDM14] Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate Object Detection and Semantic Segmentation.

In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 580–587.

[HDFN95] Hinton, G. E., Dayan, P., Frey, B. J., and Neal, R. M. The wake-sleep algorithm for Unsupervised Neural Networks. *Science 268*, 5214 (1995), 1158–1161.

[HGDG17] He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 2961–2969.

[Hin90] Hinton, G. E. Connectionist learning procedures. In *Machine Learning*. Elsevier, 1990, pp. 555–610.

[Hor91] Hornik, K. Approximation capabilities of Multilayer Feedforward Networks. *Neural Networks 4*, 2 (1991), 251–257.

[HOT06] Hinton, G. E., Osindero, S., and Teh, Y.-W. A fast learning algorithm for deep belief nets. *Neural Computation 18*, 7 (2006), 1527–1554.

[HS06] Hinton, G. E., and Salakhutdinov, R. R. Reducing the dimensionality of data with Neural Networks. *Science 313*, 5786 (2006), 504–507.

[HSM+00] Hahnloser, R. H., Sarpeshkar, R., Mahowald, M. A., Douglas, R. J., and Seung, H. S. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature 405*, 6789 (2000), 947–951.

[HW59] Hubel, D. H., and Wiesel, T. N. Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology 148*, 3 (1959), 574.

[HW62] Hubel, D. H., and Wiesel, T. N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology 160*, 1 (1962), 106.

[HW68] Hubel, D. H., and Wiesel, T. N. Receptive fields and functional architecture of monkey striate cortex. *The Journal of Physiology 195*, 1 (1968), 215–243.

[HZRS15] He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 1026–1034.

[HZRS16a] He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 770–778.

[HZRS16b] He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in Deep Residual Networks. In *European Conference on Computer Vision* (2016), Springer, pp. 630–645.

[IS15] Ioffe, S., and Szegedy, C. Batch Normalization: Accelerating Deep Network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[JSZ+15] Jaderberg, M., Simonyan, K., Zisserman, A., et al. Spatial Transformer Networks. In *Advances in nNeural Information Processing Systems* (2015), pp. 2017–2025.

[KB14] Kingma, D. P., and Ba, J. Adam: A method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980* (2014).

[KCQL18] Ke, L., Chang, M.-C., Qi, H., and Lyu, S. Multi-scale structure-aware network for human pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 713–728.

[KFF15] Karpathy, A., and Fei-Fei, L. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 3128–3137.

[KGHD19] Kirillov, A., Girshick, R., He, K., and Dollár, P. Panoptic Feature Pyramid Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 6399–6408.

[KH+09] Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images.

[KHG+19] Kirillov, A., He, K., Girshick, R., Rother, C., and Dollár, P. Panoptic Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 9404–9413.

[KSH12] Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS* (2012).

[KSJ14] Kanazawa, A., Sharma, A., and Jacobs, D. Locally Scale-Invariant Convolutional Neural Networks, 2014.

[KTS+14] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. Large-scale video classification with Convolutional Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 1725–1732.

[LB+95] LeCun, Y., Bengio, Y., et al. Convolutional Networks for images, speech, and time series. *The Handbook of Brain Theory and Neural Networks 3361*, 10 (1995), 1995.

[LBBH98] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*, 11 (1998), 2278–2324.

[LBD+89a] LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., and Jackel, L. Handwritten digit recognition with a back-propagation network. *Advances in Neural Information Processing Systems 2* (1989), 396–404.

[LBD+89b] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. Backpropagation applied to handwritten zip code recognition. *Neural Computation 1*, 4 (1989), 541–551.

[LBH15] LeCun, Y., Bengio, Y., and Hinton, G. Deep Learning. *Nature 521*, 7553 (2015), 436–444.

[LDS89] LeCun, Y., Denker, J., and Solla, S. Optimal brain damage. *Advances in Neural Information Processing Systems 2* (1989), 598–605.

[LeC98] LeCun, Y. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/* (1998).

[Low99] Lowe, D. G. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE International Conference on Computer Vision* (1999), vol. 2, Ieee, pp. 1150–1157.

[LSD15] Long, J., Shelhamer, E., and Darrell, T. Fully Convolutional Networks for Semantic Segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 3431–3440.

[LWW+20] Liu, Y., Wang, Y., Wang, S., Liang, T., Zhao, Q., Tang, Z., and Ling, H. CBNet: A Novel Composite Backbone Network Architecture for Object Detection. In *AAAI* (2020), pp. 11653–11660.

[Mal99] Mallat, S. *A Wavelet tour of Signal Processing*. Elsevier, 1999.

[MH08] Maaten, L. v. d., and Hinton, G. Visualizing data using t-SNE. *Journal of Machine Learning Research 9*, Nov (2008), 2579–2605.

[MHN13] Maas, A. L., Hannun, A. Y., and Ng, A. Y. Rectifier nonlinearities improve Neural Network acoustic models. In *Proc. icml* (2013), vol. 30, p. 3.

[MP43] McCulloch, W. S., and Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics 5*, 4 (1943), 115–133.

[MP69] Minsky, M., and Papert, S. Perceptrons.

[MV20] Mohan, R., and Valada, A. Efficientps: Efficient Panoptic Segmentation. *arXiv preprint arXiv:2004.02307* (2020).

[NH10] Nair, V., and Hinton, G. E. Rectified Linear Units improve Restricted Boltzmann Machines. In *ICML* (2010).

[PKS14] Papandreou, G., Kokkinos, I., and Savalle, P.-A. Untangling local and global deformations in Deep Convolutional Networks for image classification and sliding window detection. *arXiv preprint arXiv:1412.0296* (2014).

[PKS15] Papandreou, G., Kokkinos, I., and Savalle, P.-A. Modeling local and global deformations in Deep Learning: Epitomic convolution, multiple instance learning, and sliding window detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 390–399.

[RD06] Rosten, E., and Drummond, T. Machine Learning for high-speed corner detection. In *European Conference on Computer Vision* (2006), Springer, pp. 430–443.

[RDGF16] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You Only Look Once: Unified, real-time Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 779–788.

[RFB15] Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional Networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-assisted Intervention* (2015), Springer, pp. 234–241.

[RHBL07] Ranzato, M., Huang, F. J., Boureau, Y.-L., and LeCun, Y. Unsupervised Learning of invariant feature hierarchies with applications to Object Recognition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition* (2007), IEEE, pp. 1–8.

[RHGS15] Ren, S., He, K., Girshick, R., and Sun, J. Faster R-CNN: Towards real-time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems* (2015), pp. 91–99.

[RHW85] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning internal representations by error propagation. Tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[RHW86] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Nature 323*, 6088 (1986), 533–536.

[Ros58] Rosenblatt, F. The Perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review 65*, 6 (1958), 386.

[SEZ+13] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. Overfeat: Integrated recognition, localization and detection using Convolutional Networks. *arXiv preprint arXiv:1312.6229* (2013).

[SHK+14] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent Neural Networks from overfitting. *The Journal of Machine Learning Research 15*, 1 (2014), 1929–1958.

[SHM+16] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of Go with Deep Neural Networks and Tree Search. *Nature 529*, 7587 (2016), 484–489.

[SL12] Sohn, K., and Lee, H. Learning invariant representations with local transformations. *arXiv preprint arXiv:1206.6418* (2012).

[SLJ+15] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 1–9.

[SM14] Sifre, L., and Mallat, S. Rigid-motion scattering for Image Classification. *Ph. D. thesis* (2014).

[SVI+16] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the Inception architecture for Computer Vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 2818–2826.

[SZ14] Simonyan, K., and Zisserman, A. Very Deep Convolutional Networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[TH12] Tieleman, T., and Hinton, G. Lecture 6.5: Divide the gradient by a running average of its recent magnitude. *Coursera: Neural Networks for Machine Learning* (2012).

[TPL20] Tan, M., Pang, R., and Le, Q. V. Efficientdet: Scalable and efficient Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 10781–10790.

[Tur48] Turing, A. M. Intelligent machinery, 1948.

[Tur50] Turing, A. M. Computing machinery and intelligence. *Mind LIX*, 236 (10 1950), 433–460.

[TVDJ20] Touvron, H., Vedaldi, A., Douze, M., and Jégou, H. Fixing the train-test resolution discrepancy: FixEfficientNet. *arXiv preprint arXiv:2003.08237* (2020).

[WGH+20] Wu, C.-Y., Girshick, R., He, K., Feichtenhofer, C., and Krahenbuhl, P. A Multigrid Method for Efficiently Training Video Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 153–162.

[WHZS20] Wang, T., Huang, J., Zhang, H., and Sun, Q. Visual Commonsense R-CNN. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 10760–10770.

[WRKS16] Wei, S.-E., Ramakrishna, V., Kanade, T., and Sheikh, Y. Convolutional Pose Machines. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 4724–4732.

[WZG+20] Wang, H., Zhu, Y., Green, B., Adam, H., Yuille, A., and Chen, L.-C. Axial-DeepLab: Stand-Alone Axial-Attention for Panoptic Segmentation. *arXiv preprint arXiv:2003.07853* (2020).

[XGD+17] Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. Aggregated residual transformations for Deep Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 1492–1500.

[YK15] Yu, F., and Koltun, V. Multi-scale context aggregation by Dilated Convolutions. *arXiv preprint arXiv:1511.07122* (2015).

[YSWJ19] Yang, L., Song, Q., Wang, Z., and Jiang, M. Parsing R-CNN for instance-level human analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 364–373.

[ZH05] Zou, H., and Hastie, T. Regularization and variable selection via the Elastic Net. *Journal of the Royal Statistical Society: series B (Statistical Methodology) 67*, 2 (2005), 301–320.

[ZHLD19] Zhu, X., Hu, H., Lin, S., and Dai, J. Deformable Convnets v2: More deformable, better results. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 9308–9316.

[ZPZ+20] Zhou, L., Palangi, H., Zhang, L., Hu, H., Corso, J. J., and Gao, J. Unified Vision-Language Pre-Training for Image Captioning and VQA. In *AAAI* (2020), pp. 13041–13049.

[ZWZ+20] Zhang, H., Wu, C., Zhang, Z., Zhu, Y., Zhang, Z., Lin, H., Sun, Y., He, T., Mueller, J., Manmatha, R., et al. Resnest: Split-attention networks. *arXiv preprint arXiv:2004.08955* (2020).