UNIVERSIDAD
DE GRANADA

Facultad de Ciencias
Escuela Técnica Superior de Ingenierías Informática y de Telecomunicaciones

Grado en Ingeniería Informática y Matemáticas

TRABAJO DE FIN DE GRADO

Localización de Regiones de Interés Utilizando Aprendizaje Profundo

Presentado por:
Laura Gómez Garrido

Tutor:
Jesús Chamorro Martínez
Ciencias de la Computación e Inteligencia Artificial

Curso académico 2019-2020

Localización de Regiones de Interés Utilizando Aprendizaje Profundo

Laura Gómez Garrido

Laura Gómez Garrido *Localización de Regiones de Interés Utilizando Aprendizaje Profundo*.
Trabajo de fin de Grado. Curso académico 2019-2020.

Responsable de tutorización

Jesús Chamorro Martínez
Ciencias de la Computación e Inteligencia Artificial

Grado en Ingeniería Informática y Matemáticas

Facultad de Ciencias
Escuela Técnica Superior de Ingenierías Informática y de Telecomunicaciones

Universidad de Granada

Declaración de originalidad

D./Dña. Laura Gómez Garrido

Declaro explícitamente que el trabajo presentado como Trabajo de Fin de Grado (TFG), correspondiente al curso académico 2019-2020, es original, entendida esta, en el sentido de que no ha utilizado para la elaboración del trabajo fuentes sin citarlas debidamente.

En Granada a 22 de octubre de 2020

Fdo: Laura Gómez Garrido

Dedicatoria (opcional)
Ver archivo `preliminares/dedicatoria.tex`

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

Agradecimientos del libro (opcional, ver archivo `preliminares/agradecimiento.tex`).

Summary

An english summary of the project (around 800 and 1500 words are recommended).
File: `preliminares/summary.tex`

Introducción

De acuerdo con la comisión de grado, el TFG debe incluir una introducción en la que se describan claramente los objetivos previstos inicialmente en la propuesta de TFG, indicando si han sido o no alcanzados, los antecedentes importantes para el desarrollo, los resultados obtenidos, en su caso y las principales fuentes consultadas.

Ver archivo `preliminares/introduccion.tex`

Parte I.

Conceptos previos

En esta parte, introduciremos una serie de conceptos previos necesarios para poder hablar con propiedad en adelante. Explicaremos los conceptos de red neuronal y red neuronal convolucionada. Veremos que se las redes neuronales son aproximadores universales y, además, conoceremos el concepto operación no local así como ver su consistencia.

1. Red neuronal

1.1. El problema de clasificar una imagen.

Cuando observamos una imagen, podemos localizar varios elementos, a partir de los cuáles, esta se encuentra compuesta con tan sólo un vistazo. Sin embargo, para un ordenador no se trata de algo tan sencillo puesto que sólo es capaz de ver un gran conjunto de números que no tienen por qué tener relación alguna entre sí.

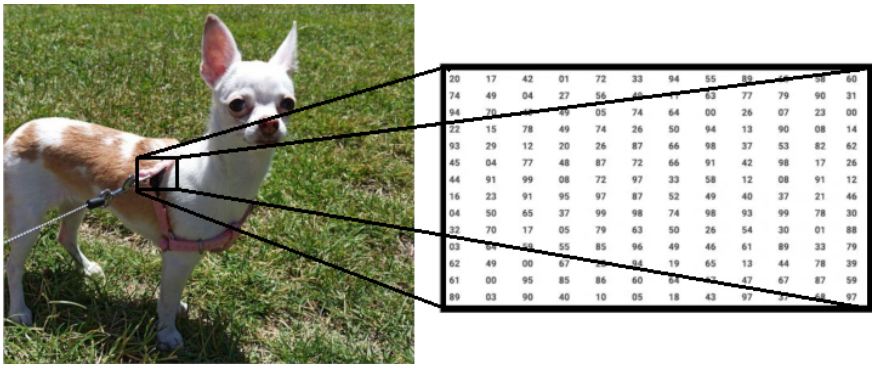


Figura 1.1.: Cómo un ordenador ve una imagen

Idealmente, en esta imagen desearíamos que fuera capaz de identificar que se trata de un perro, en concreto de un chiguagua, de pelaje blanco y manchas color café que se encuentra sobre un césped. Estos pocos datos, que para nosotros parecen tan triviales, necesitan de horas y horas de computación para ser obtenidos a partir de una imagen cualquiera.

Dejamos todos estos detalles, a los cuales esperamos poder llegar en secciones futuras, y simplificamos el problema a tener un conjunto de etiquetas y buscar con cuál de todas ellas tiene mayor relación nuestra imagen.

Una primera idea sería utilizar utilizar *k-nearest neighbor classifier*, en adelante *k*-NN. Este clasificador consiste en, para cada etiqueta, determinar las *k* imágenes cuya distancia, siendo Manhattan y Euclídea las más comunes, entre los píxeles de nuestra imagen sea menor. La etiqueta idónea será aquella cuya dicha distancia sea menor considerando las *k* imágenes.

De esta sencilla propuesta surgen diversos problemas. Las métricas más comunes de este clasificador suelen darle mayor importancia al valor concreto de los píxeles, en lugar de a las formas que de las que está compuesta la figura, provocando que valores como los colores de fondo pueden influir más en la clasificación que los propios píxeles de la figura que queremos clasificar. En el ejemplo de la imagen del chiguagua, si considerásemos las etiquetas verde y perro, tendríamos que por lo general como respuesta el color verde, pese a que

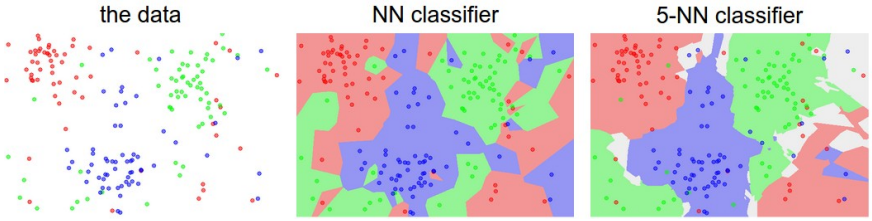


Figura 1.2.: Un ejemplo de la diferencia entre un *Nearest Neighbor classifier* y un *5-Nearest Neighbor classifier* con puntos bidimensionales y tres clases.[aut]

estamos más interesados por la mascota en sí. Otro gran problema sería la baja escalabilidad que nos proporciona esta solución, al incrementarse enormemente el costo computacional de clasificación conforme aumenta el número de imágenes a comparar, ya sea por aumentar el número de etiquetas o nutrir de más datos las ya existentes.

Debido a la falta de escalabilidad de este método de clasificación no paramétrico, existe un gran problema ante el incremento de los datos de entrenamiento. Un primer ejemplo de la búsqueda de aumentar la escalabilidad del algoritmo sin un aumento de los tiempos de clasificación, podrían ser los clasificadores lineales.

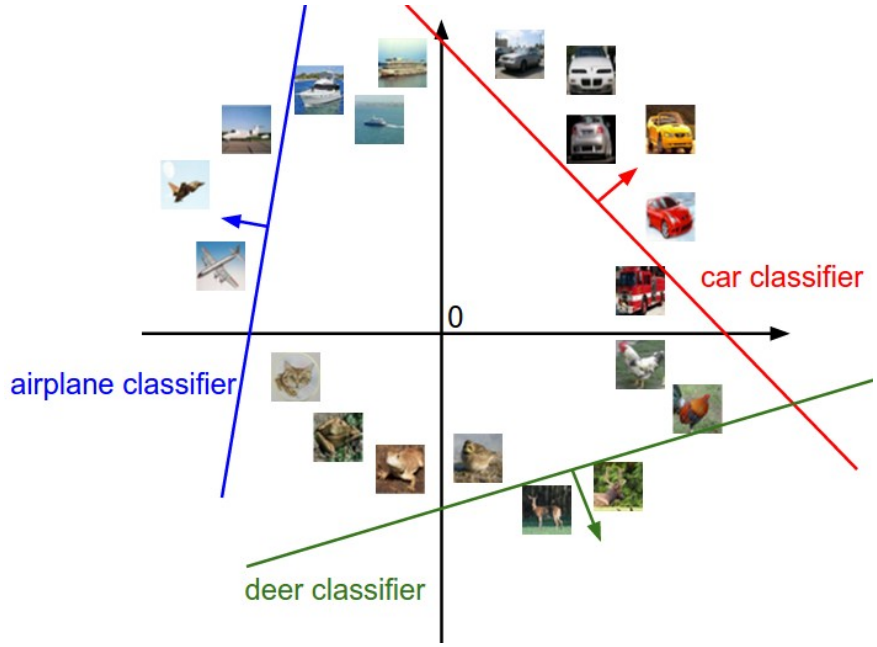


Figura 1.3.: Ejemplo de representación de un clasificador lineal con tres etiquetas.[aut]

Consideremos $D \in \mathbb{N}$ la dimensión de nuestras imágenes o datos de entrada y $K \in \mathbb{N}$ la cantidad de etiquetas o categorías bajo las cuales pueden ser clasificadas. Siendo $N \in \mathbb{N}$ el número de ejemplos que utilizaremos para entrenar nuestro clasificador, tendremos que para cada dato de entrenamiento $x_i \in \mathbb{R}^D$ $i = 1, ..., N$ le corresponde una etiqueta $y_i \in \{1, ..., K\}$

tal que juntos conforman el par (x_i, y_i) de imagen y categoría a la que pertenece. Utilizando estos datos, podremos entrenar el clasificador lineal que será una función del tipo

$$f(x) = W \cdot x + b \quad W \in M_{K,D}(\mathbb{R}) \quad b \in \mathbb{R}^K \quad \forall x \in \mathbb{R}^D,$$

con la cual estaremos dividiendo el espacio de resultados utilizando hiperplanos. Existen múltiples tipos de clasificadores lineales, como por ejemplo una *máquina de vectores de soporte multiclase (Multiclass SVM)* o un clasificador SoftMax que tienen como principal diferencia la función de pérdida que utilizan para penalizar las etiquetas incorrectas.

1.2. El modelo de una neurona

Cuando comenzamos planteando nuestro problema buscábamos conseguir clasificar una imagen con una probabilidad de acertar similar o superior a la humana. Teníamos el problema de que un ordenador no era capaz de pensar o razonar de la misma forma que un ser humano y buscábamos una forma de clasificar esta información a pesar de ello. Es aquí donde consideraremos los modelos de redes neuronales, que buscan ser capaces de simular cómo funciona un cerebro humano para aprender cómo reconocer distintos elementos de la misma forma que nosotros lo hemos ido haciendo a lo largo de nuestra vida.

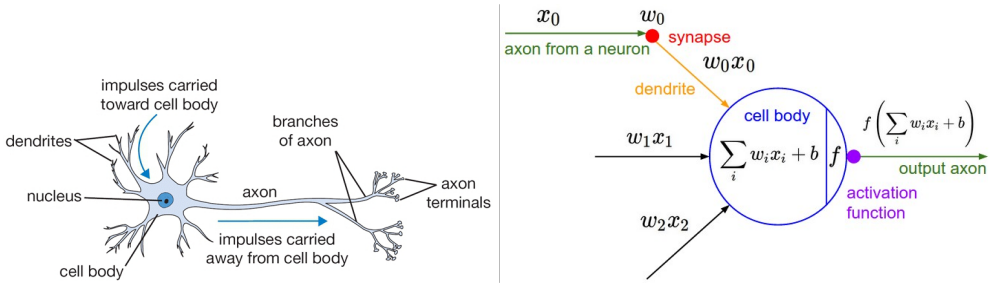


Figura 1.4.: Comparación entre una neurona biológica (izquierda) y el modelo matemático (derecha) [aut]

Nuestro cerebro está formado por múltiples neuronas interconectadas entre sí que están constantemente transmitiéndose información y aprendiendo a través de todos los datos que reciben. Si nos fijamos en 1.4 podemos ver que una neurona real esta formada por dentritas que son quienes, a través del proceso de sinapsis, reciben la entrada de información, que es asimilada y transformada por su núcleo antes de ser transmitida a la siguientes neuronas a través de las divisiones de su axón en caso de que supere un cierto umbral y se active.

Si consideramos que tenemos D dentritas y que por la i-ésima dentrita recibimos la información x_i con un peso o fuerza de sinapsis w_i, tendríamos que nuestro núcleo trabaja con el vector de información $(w_1 x_1, ..., w_D x_D)$ que podemos estimar como $\sum_i w_i x_i$ y sumarle una determinada constante b propia de la neurona. El umbral de activación y la señal enviada al resto de neuronas será la evaluación a través de una *función de activación* Def. 1.1. Cabe destacar la similitud existente entre el modelo de una neurona y un clasificador lineal.

Definición 1.1. Una *función de activación* es una función $f : \mathbb{R}^n \to \mathbb{R}^n$ siendo $n \in \mathbb{N}$ el número de neuronas de la capa a evaluar.

A continuación, mencionaremos los ejemplos más representativos utilizados como funciones de activación:

FIXME: Sugerencia: Añadir una gráfica para cada función de activación

- *Función sigmoide* o *logística* $\sigma : \mathbb{R}^n \to [0,1]^n$ definida como $\sigma_i(x_i) = \frac{1}{1+e^{-x_i}}$. Se suele utilizar en la última capa de nuestra red, cuando nuestras imágenes pueden pertenecer a varias clases o etiquetas al mismo tiempo al ser el resultado para cada componente independiente del resto.

- *Función softmax* utilizada normalmente en la última capa donde hay tantas neuronas como etiquetas y en el problema de clasificación donde una imagen puede pertenecer únicamente a una sola etiqueta o caso. Se trata de una modificación de función logística que normaliza la salida para obtener la probabilidad de pertenecer a cada clase obteniendo así que la suma de todas las salidas de esta capa sería 1. Aquí, la i-ésima neurona tendría función de activación $\sigma_i(x) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$ y utilizaría $L(x) = \frac{1}{N}\sum_{i=1}^{N} -\ln \frac{e^{\sigma_{y_i}(x)}}{\sum_{j=1}^{D} e^{\sigma_j(x)}} = \frac{1}{N}\sum_{i=1}^{N}(-\sigma_{y_i}(x) + \ln \sum_{j=1}^{D} e^{\sigma_j(x)})$ como función de pérdida para el entrenamiento.

- *Función ReLU*, cuyo nombre completo sería *Rectified Linear Unit*. Se suele utilizar en las capas intermedias de nuestra red y es de la forma $f(x) = \max(0, x)$.

- *Función tanh* se trata de de una centralización de la función sigmoide. $Tanh(x) = 2\sigma(x) - 1$.

1.3. La red completa

Una red red neuronal es un conjunto de neuronas dividas en varias *capas* de forma que las neuronas de una capa pueden estar unidas o no con las neuronas de la capa anterior y de la siguiente formando así un grafo acíclico dirigido.

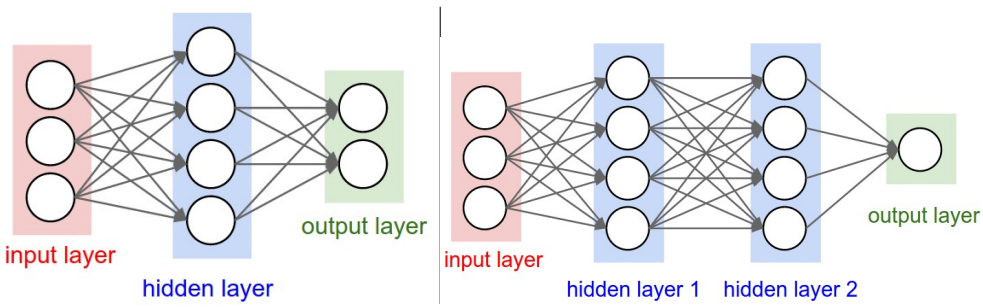


Figura 1.5.: Ejemplos de redes totalmente conectadas (*fully-connected*)

La primera capa recibe el nombre de *capa de entrada* que posee D neuronas, es decir, tantas como la dimensión de nuestros datos. Por otro lado tenemos la *capa de salida* con K neuronas, tantas como etiquetas o clases de clasificación tengamos. El resto de neuronas se distribuyen dentro de las *capas ocultas* cuya distribución y conexiones dependen del modelo en concreto

que queramos desarrollar, quedando a nuestro criterio.

Así, podemos afirmar que una red neuronal es la aplicación sucesiva de funciones de la forma

$$F(x) = \sum_{j=1}^{n} \alpha_j \sigma(w_j^T x + b_j) \quad \forall x \in \mathbb{R}^n,$$

donde $w_j \in \mathbb{R}^n$, $\alpha_j, b_j \in \mathbb{R}$ serán fijas una vez entrenada la red y $\sigma : \mathbb{R} \to \mathbb{R}$ la función de activación elegida en cada capa.

FIXME el siguiente párrafo desaparece y en su lugar aparecerá los algoritmos escritos. Así no es tan sacado de la manga

A continuación toca hablar del entrenamiento de la red completa, a los conceptos que ya conocíamos del clasificador lineal **??** se añade el concepto de *Backpropagation* o *Propagación hacia atrás de errores* que hace uso de la *regla de la cadena* para, a través de los cálculos locales realizados por cada neurona, obtener la función completa de clasificación. Así, buscamos que las neuronas se auto organicen para ser capaces de localizar patrones similares de forman que sepan cómo reaccionar ante la presencia de ruido o datos incompletos. Esto hace a través de la comunicación de sus valores locales y gradientes entre las distintas neuronas, comenzando por la capa de salida y avanzando hacia atrás. FIXME: Mencionar importancia del Subgradiente para el cálculo del gradiente, al no estar definido para cualquier función.

FIXME: Definir epoch y la validación cruzada

2. Teoremas de Aproximación Universal

Hasta ahora se ha definido una estructura con una serie de parámetros con la esperanza de poder llegar a resolver el problema planteado. Sin embargo, en ningún momento hemos utilizado ningún resultado que nos asegure que dicha estructura puede llegar a resolver el problema que deseamos. A continuación, mostraremos diversos resultados, con diversos niveles de generalidad, que muestran cómo esta estructura es un aproximador universal y por ello resuelve nuestro problema. Estos resultados se dividen en dos grandes categorías *anchura indeterminada* y *profundidad indeterminada*.

2.1. Anchura indeterminada

2.1.1. George Cybenko, 1989

Cybenko, en [Cyb89], demostró la capacidad de aproximación en el caso de anchura indeterminada y profundidad fijada para funciones de activación sigmoidales Def. 2.1. A partir de esta versión clásica, se logró considerar otras funciones de activación, e incluso, demostrar que era gracias a la arquitectura en sí misma, y no a la función de activación, que las redes neuronales eran aproximadores universales [KH91]. Aquí estudiaremos la versión del teorema para funciones continuas, pero debemos destacar que en el mismo artículo se mencionan también versiones para otros espacios de funciones.

Definición 2.1. Una función $\sigma : \mathbb{R} \to \mathbb{R}$ es *sigmoidal* si

$$\lim_{t \to +\infty} \sigma(t) = 1 \quad y \quad \lim_{t \to -\infty} \sigma(t) = 0.$$

Definición 2.2. Dada una σ-álgebra M, una *medida con signo* μ sobre M es una función del conjunto $\mu : M \to [-\infty, +\infty]\sigma$-aditiva.

Definición 2.3. Una medida se dice *de Borel* si está definida sobre una σ-álgebra de Borel, es decir, la engendrada por los abiertos del espacio.

Definición 2.4. Una *medida con signo regular de Borel* sobre una σ-álgebra M es una medida con signo que cumple

$$\mu(E) = \inf\{\mu(V) : E \subset V, V \text{ abierto}\} = \sup\{\mu(C) : C \subset E, C \text{ cerrado}\}$$

para todo conjunto de Borel $E \in M$

En adelante, utilizaremos I_n para referirnos al cubo unidad n-dimensional, $[0,1]^n$ y para el espacio de las medidas finitas con signo regulares de Borel sobre I_n utilizaremos $M(I_n)$. Además, $C(I_n)$ denotará el espacio de funciones continuas en I_n.

Definición 2.5. Una función $\sigma : \mathbb{R} \to \mathbb{R}$ es *discriminatoria* si, para una medida $\mu \in M(I_n)$ con

$$\int_{I_n} \sigma(w^T x + b) d\mu(x) = 0$$

para todo $w \in R^n$ y $b \in \mathbb{R}$, implica que $\mu = 0$.

Lema 2.1. *Cualquier función sigmoidal medible y acotada es discriminatoria. En particular, las funciones sigmoidales continuas son discriminatorias.*

Demostración. □

Teorema 2.1. *Sea σ una función discriminatoria continua. Entonces, las sumas finitas de la forma*

$$G(x) = \sum_{i=1}^{n} \alpha_i \sigma(w_i^T x + b)$$

son densas en $C(I_n)$. En otra palabras, dados $f \in C(I_n)$ y $\varepsilon > 0$, existe una suma, $G(x)$, de la forma anterior, para la cual

$$|G(x) - f(x)| < \varepsilon \quad \forall x \in I_n.$$

Demostración. □

Teorema 2.2. *Sea σ una función sigmoidal continua. Entonces, las sumas finitas de la forma*

$$G(x) = \sum_{i=1}^{n} \alpha_i \sigma(w_i^T x + b)$$

son densas en $C(I_n)$. En otra palabras, dados $f \in C(I_n)$ y $\varepsilon > 0$, existe una suma, $G(x)$, de la forma anterior, para la cual

$$|G(x) - f(x)| < \varepsilon \quad \forall x \in I_n.$$

Demostración. Para esta demostración, se combina Lema 2.1 y Teorema 2.1, notando que las funciones sigmoidales continuas satisfacen las condiciones de este lema. □

2.1.2. Kurt Hornik, 1989 y 1991

En 1898 [HSW89], Hornik demostró que las redes neuronales con una capa oculta que utiliza funciones de aplastamiento (sigmoidal Def. 2.1 y no decreciente según la *definición 2.3* del mismo artículo) arbitrarias son capaces de aproximar cualquier función Borel-medible de un espacio finito unidimensional con cualquier grado deseado de precisión.

Más tarde, en 1991 [KH91], extendió los teoremas de Cybenko mostrando que no necesariamente tenía que utilizar funciones de activación sigmoidales para los espacios de funciones considerados. A continuación enunciaremos el teorema para el espacio de funciones continuas.

Teorema 2.3. *Sea $\sigma : \mathbb{R} \to \mathbb{R}$ una función continua, acotada y no constante. Entonces, las sumas finitas de la forma*

$$G(x) = \sum_{i=1}^{n} \alpha_i \sigma(w_i^T x - b)$$

son densas en $C(X)$ para todos los subconjuntos compactos X de $\mathbb{R}^m$.

2.2. Profundidad indeterminada

2.2.1. Zhou Lu, Hongmin Pu, Feicheng Wang, Zhiquang Hu y Liwei Wang, 2017

Estos autores demostraron [LPW+17] el caso de profundidad indeterminada para funciones Lebesgue-integrables y una función de activación ReLU. Este teorema fue presentado como una versión dual de las demostraciones para anchura indeterminada y abre el camino para nuevas demostraciones en el caso de anchura indeterminada.

Teorema 2.4. *Para cualquier función Lebesgue-integrable $f : \mathbb{R}^n \to R$ y cualquier $\varepsilon > 0$, existe A una red totalmente conectada con función de activación ReLU y una anchura $d_m \leq n + 4$, tal que la función F_A representada por esta red satisface*

$$\int_{\mathbb{R}^n} |f(x) - F_A(x)| dx < \varepsilon.$$

2.2.2. Patrick Kidger y Terry Lyons, 2019

Una de las variantes más recientes [KL19], fue presentada para el caso de una función de activación no afín, que sea continuamente diferenciable y con derivada no nula en al menos un punto. Destaca porque con sus consideraciones abarca las funciones de activación utilizadas en la práctica, incluyendo las funciones de activación polinómicas. En el artículo, se consideran otras extensiones o variaciones al teorema que enunciaremos.

Definición 2.6. Sea $\sigma : \mathbb{R} \to \mathbb{R}$ y $n, m, k \in \mathbb{N}$. Entonces $NN^{\sigma}_{n,m,k}$ representa la clase de funciones $R^n \to \mathbb{R}^m$ descritas por una red neuronal hacia adelante con n neuronas en la capa de entrada, m neuronas en la capa de salida, y un numero arbitrario de capas ocultas, para las cuales k neuronas tienen como función de activación la función σ. Cada neurona de la capa de salida tiene la función de activación identidad.

Teorema 2.5. *Sea $\sigma : \mathbb{R} \to \mathbb{R}$ una función continua y no polinómica que es continuamente diferenciable en al menos un punto, con derivada no nula en dicho punto. Sea $K \subset \mathbb{R}^n$ un compacto. Entonces $NN^{\sigma}_{n,m,n+m+2}$ es denso en $C(K; \mathbb{R}^m)$ con respecto a la norma del supremo.*

Teorema 2.6. *Sea $\sigma : \mathbb{R} \to \mathbb{R}$ una función polinómica y no afín. Sea $K \subset \mathbb{R}^n$ un compacto. Entonces $NN^{\sigma}_{n,m,n+m+2}$ es denso en $C(K; \mathbb{R}^m)$ con respecto a la norma del supremo.*

3. Redes neuronales convolucionadas (CNN)

Hasta ahora, todo lo que hemos hablado es válido para casi cualquier contexto puesto que no hemos hecho ninguna presunción en cuanto a la estructura de los datos o sobre las peculiaridades que estos presentan. Para nosotros todos los datos eran un vector con D componentes a los que les correspondía una etiqueta y no le dábamos importancia a la estructura interna que estos pudieran llegar a presentar.

Para la edición y manipulación de imágenes, es común utilizar la operación de convolución como filtro. Las redes neuronales son aquellas que utilizan estos filtros, de ahí su nombre, para analizar las distintas características de las imágenes. Si bien, podemos fijar los pesos manualmente para utilizar algunos de los filtros más comunes en la edición de imágenes, se ha comprobado experimentalmente que por lo general se obtienen mejores resultados si es la propia red quien fija estos pesos a través de un proceso de aprendizaje.

En adelante, dejaremos de considerar que nuestros datos de entrada tienen una estructura de vector y consideraremos que estamos trabajando con matrices tridimensionales, es decir, nuestros datos estarán organizados como si fueran ortoedros. Esto lo representaremos como $M_{a,h,c}(\mathbb{R})$ con $a, h, c \in \mathbb{N}$ donde a representará la anchura, h la altura y c el número de canales. Para simplificarlo, cada entrada $X \in M_{a,h,c}(\mathbb{R})$ representará c imágenes en escala de grises de dimensiones $a \times h$. Como detalle, una imagen RGB está representada como tres imágenes en escala de grises($c = 3$) donde cada canal representa un color primario y con la combinación de los tres obtenemos una imagen a color.

3.1. Capas Convolucionadas o Convolutionals

Internamente, cada neurona de una capa convolucional posee un *kernel* o *filtro* $W \in M_{r,s,c}(\mathbb{R})$ donde r, s y c son parámetros prefijados y una variable $b \in \mathbb{R}$ bias. Para cada entrada $X \in M_{a,h,c}(\mathbb{R})$ tomamos una sección $x^{RS} \subset X$ donde $x^{RS} = (x^{RS}_{ijk})_{ijk}$ $i = R, ..., R + r$, $j = S, ..., S + s$ y $k = 1, ..., c$ con $R = 1, ..., a - r$ y $S = 1, ..., h - s$. Así, cada neurona realiza una *convolución matricial* y suma la variable b bias:

$$f(x^{RS}) = \sum_{k=1}^{c} \sum_{i=1}^{r} \sum_{j=1}^{s} w_{i,j,k} \cdot x^{RS}_{i+R,j+S,k} + b$$

Siendo esta su función de puntuación de las neuronas correspondientes a dicho kernel. A este filtro le corresponderán tantas neuronas como sean necesarias para cubrir todos los datos de entrada. Una capa de convolución, podrá tener tantos filtros como se quieran y cada uno de ellos tendrá tantas neuronas como sean necesarias para cubrir toda la imagen. Visualmente, se transforma un ortoedro en otro.

Figura 3.1.: Capa de Convolución. La zona grisácea corresponde al resultado de un filtro.

En esta demo del Curso de Stanford sobre Convolutional Neural Networks for Visual Recognition podemos ver el funcionamiento de dos filtros 3×3 ($r = 3, s = 3$) a una entrada $x \in M_{7,7,3}$. Nótese, que el filtro no es aplicado en todas las submatrices sino que avanza dos posiciones tanto vertical como horizontalmente, es decir, la capa posee un *paso* o *stride* de 2. En la formulación anterior, se ha supuesto que el paso es de tamaño 1. Además, en la demo se ha completado la matriz x con 0 hasta tener una dimensión de $9 \times 9 \times 3$ para asegurarnos de que recorremos todas las posiciones de x con el filtro. Tanto el paso como si la matriz es completada con algún otro número o no son parámetros que se prefijan al crear la capa, comunes a todos los filtros y controlan la dimensión de salida de la capa.

3.2. Capas de Agrupación o Pooling

FIXME: Same convoluciones

3.3. Ejemplos

FIXME: ¿Apéndices o distribuido en el pdf? Cosas a mencionar:

- Cómo se ven tras cada neurona y/o capa (cómo pooling y convolution modifican la imagen)
- Cómo los filtros modifican una imagen visualmente

4. Operador no local

En la sección anterior se ha visto que una operación de convolución posee un kernel que limita la cantidad de posiciones que son observadas de forma que la cantidad de parámetros de la capa de convolución es dependiente al tamaño de este núcleo. Esto tiene como desventaja que, para cada posición de nuestra entrada de datos, las posiciones que influyen al resultado de la convolución están restringidas a un subconjunto de tamaño fijo de los datos de entrada por lo que se suele decir que es una operación *local*. Si quisiéramos utilizar todos los datos de entrada para cada posición de la imagen, tendríamos un aumento considerable de parámetros a utilizar, con los consecuentes problemas de memoria y entrenamiento.

El objetivo de esta sección es definir un tipo de operación que sea capaz de extraer características de nuestros datos de entrada utilizando toda su dimensión y sin tener un aumento drástico del número de parámetros a entrenar, será a lo que llamaremos un operador no local. La necesidad de este tipo de operación nace del hecho de que, por lo general, las imágenes a estudiar no suelen ser un conjunto de *collage* sin relación entre sí, sino que el entorno de nuestra imagen puede llegar a ser de utilidad a la hora de distinguir qué objetos estamos clasificando.

Para simplificar la definición, nos restringiremos al contexto de espacios vectoriales de dimensión finita sobre el cuerpo de los números reales.

Definición 4.1. Sea $x = (x_1, ..., x_D) \in \mathbb{R}^D$ una señal de entrada, $u : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ un producto escalar, $v : \mathbb{R} \to \mathbb{R}$ función real evaluada y $C : \mathbb{R}^D \to \mathbb{R}$. Se define una operación no local genérica o *generic non-local operation* $f : \mathbb{R}^D \to \mathbb{R}^D$ con $f(x) = (f_1(x_1), ..., f_D(x_D))$ aquella donde:

$$f_i(x_i) = \frac{1}{C(x)} \sum_{j=1}^{D} u(x_i, x_j) v(x_j) \quad \forall i = 1, ..., D$$

En este contexto, $v(x_j)$ será una representación de la señal de x entrada en la posición j, u representará una relación, por ejemplo la afinidad, entre las posiciones i y j.

Proseguiremos enunciando varios ejemplos de funciones que pertenezcan a dicha definición para mostrar la diversidad que podemos elegir a la hora de implementar un modelo que siga esta estructura. Por simplicidad, consideraremos que $v(x_j) = W_v x_j$ donde W_v es una matriz de pesos que tiene que ser aprendida, siendo esta fácilmente implementable como una convolución de núcleo de tamaño 1.

1. *Gaussiana* $u(x_i, x_j) = e^{x_i^T x_j}$ y $C(x) = \sum_{j=1}^{D} u(x_i, x_j)$.
2. *Embedded Gaussian* $u(x_i, x_j) = e^{\phi_i(x_i)^T, \theta_j(x_j)}$ donde $\phi_i(x_i) = W_{\phi_i} x_i$ y $\theta_j(x_j) = W_{\theta_j} x_j$ y $C(x) = \sum_{j=1}^{D} u(x_i, x_j)$. Nótese que el módulo *self-attention* [VSP+17] es un caso particular de este ejemplo.

3. *Dot-product similarity* $u(x_i, x_j) = \phi_i(x_i)^T \theta_j(x_j)$ donde $\phi_i(x_i) = W_{\phi_i} x_i$, $\theta_j(x_j) = W_{\theta_j} x_j$ y $C(x) = D$.

4. *Non-local means* [BCM05]

Para poder afirmar que un operador no local cumple con lo que buscamos, primero se debe demostrar que las operaciones de este tipo nos dan el resultado deseado. Para ello, primero deberemos de enunciar una serie de conceptos.

4.1. Conceptos previos

En adelante, el conjunto Ω será un espacio muestral y $\mathcal{A}$ será una *σ-álgebra* sobre dicho Ω. Además, P será una probabilidad aplicada a los elementos de la σ-álgebra $\mathcal{A}$. Esto será representado como $(\Omega, \mathcal{A}, P)$ siendo un espacio probabilístico y será donde trabajemos.

Definición 4.2 (Proceso estocástico)**.** Un *proceso estocástico* es una familia de variables aleatorias $\{X_t\}_{t \in T}$ en T, donde T es un conjunto ordenado arbitrario y cada variable aleatoria está definida sobre un espacio de probabilidad $(\Omega, \mathcal{A}, P)$.

A partir de ahora, siempre que nos refiramos a un proceso nos estaremos refiriendo a un proceso estocástico bajo la definición anterior.

Definición 4.3 (Proceso con incrementos independientes)**.** Un proceso estocástico tiene *incrementos independientes* si $\forall n > 1, \forall t_1 < \ldots < t_n \in T$

$$X_{t_1}, X_{t_2} - X_{t_1}, \ldots, X_{t_n} - X_{t_{n-1}} \text{son variables aleatorias independientes.}$$

Definición 4.4 (Proceso con incrementos estacionarios)**.** Un proceso estocástico tiene *incrementos estacionarios* si $\forall s < t \in T, \forall h \in T$

$$(X_t - X_s) \sim (X_{t+h} - X_{s+h}).$$

$X \sim Y$ indica que X sigue la misma distribución que Y.

Definición 4.5 (Proceso estacionario)**.** Un proceso estocástico es *estrictamente estacionario* si $\forall n, \forall t_1 < \ldots < t_n \in T, \forall h \in T$

$$(X_{t_1}, \ldots, X_{t_n}) \sim (X_{t_1+h}, \ldots, X_{t_n+h}).$$

A continuación, definiremos el concepto proceso mezclado. En la literatura, los cuatro tipos de mezcla normalmente son referidos como ψ-mezclado, Φ-mezclado (o uniformemente mezclado), ρ-mezclado (o mezcla basada en la correlación maximal) y α-mezclado (o fuertemente mezclado). Cambiaremos la notación a Φ_i-mezcla $i = 1, ..., 4$ respectivamente, de forma que una Φ_i-mezcla implique una Φ_{i+1}-mezcla con $i = 1, ..., 3$.

Por notación, diremos también que $\mathcal{A}_m^n$ será la σ-álgebra inducida por las variables aleatorias Z_j con $m \leq j \leq n$ en el espacio muestral Ω. Entonces:

Definición 4.6 (Proceso mezclado)**.** La secuencia $\{Z_j\}_{j\geq 1}$ es conocida Φ_i-mezclada con $i = 1, ..., 4$ si, para cada $A \in \mathcal{A}_1^k$ y para cada $B \in \mathcal{A}_{k+n}^{\infty}$, las siguientes desigualdades son satisfechas:

- Para Φ_1-mezclada:

$$|P(A \cap B) - P(A)P(B)| \leq \Phi_1(n)P(A)P(B) \text{ con } \Phi_1(n) \downarrow 0 \text{ cuando } n \to \infty$$

- Para Φ_2-mezclada:

$$|P(A \cap B) - P(A)P(B)| \leq \Phi_2(n)P(A) \text{ con } \Phi_2(n) \downarrow 0 \text{ cuando } n \to \infty$$

- Para Φ_3-mezclada:

$$|P(A \cap B) - P(A)P(B)| \leq \Phi_3(n)[P(A)P(B)]^{\frac{1}{2}} \text{ con } \Phi_3(n) \downarrow 0 \text{ cuando } n \to \infty$$

- Para Φ_4-mezclada:

$$|P(A \cap B) - P(A)P(B)| \leq \Phi_4(n) \text{ con } \Phi_4(n) \downarrow 0 \text{ cuando } n \to \infty$$

4.2. Consistencia de un operador no local

La idea es que, bajo supuestos estacionarios, para cada píxel i una operación no local converge a la expectativa condicional de un píxel i observado en un vecindario del píxel. En este caso las condiciones de estacionariedad equivalen a decir que, a medida que crece el tamaño de la imagen, podemos encontrar muchas regiones similares para todos los detalles de la imagen. Es decir, conforme aumenta el tamaño de la imagen es más probable que, si encontramos un objeto perteneciente a determinada categoría, encontremos más objetos pertenecientes a dicha categoría.

Sea V un campo aleatorio (random field) y los datos de entrada v una realización de V. Sea Z una secuencia de variables aleatorias con $Z_i = \{X_i, Y_i\}$ con $Y_i = V(i)$ es real valuada y $X_i = V(I\backslash\{i\})\mathbb{R}^p$ valuada donde I representa el conjunto de posiciones. Una operación no local genérica será un estimador de la esperanza condicionada $E[Y_i|X_i = v(I\backslash i)]$.

Teorema 4.1. *Sea $Z = \{V(N_i\backslash i), V(i)\}_{i\geq 1}$ un proceso estocástico estrictamente estacionario y mezclado. Sea f^n una operación no local genérica aplicada a la secuencia $Z_n = \{V(N_i\backslash\{i\}), V(i)\}_{i\geq 1}^n$. Entonces,*

$$|f^n(j) - [Y_j|X_j = v(I\backslash\{j\})]| \to 0 \; a.s \; \forall j \in 1, ..., n$$

.

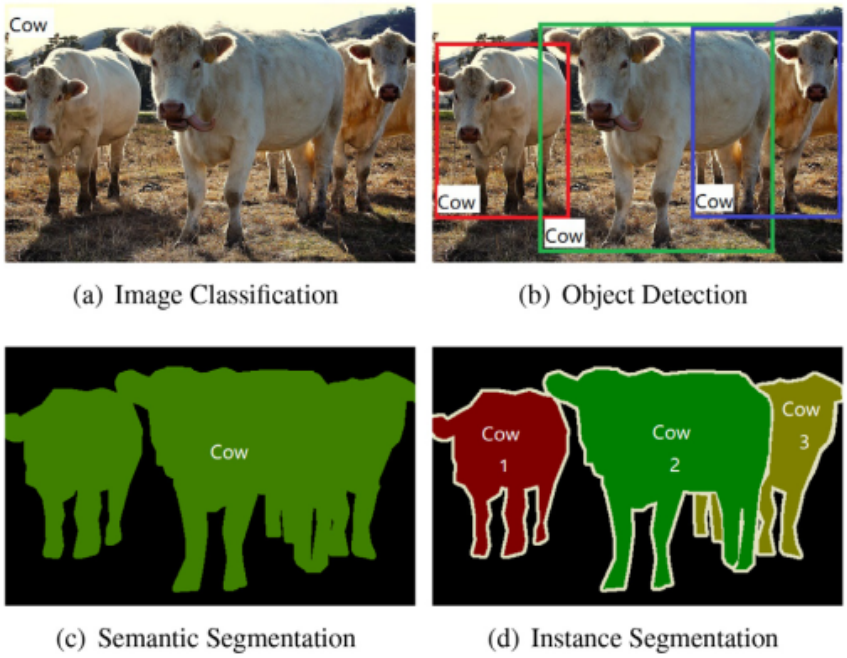
En un contexto más general, la demostración se puede encontrar en [BCM05] bajo la suposición de cualquiera de cuatro tipos de procesos mezclados enunciados en Def. 4.6

5. Paradigmas de detección

Hasta este momento, hemos tratado un problema de clasificación global. Se suponía que la imagen tenía un único elemento que se quisiera clasificar e idealmente este estaría centrado con respecto el centro de la imagen. En adelante, querremos clasificar múltiples elementos pertenecientes a múltiples etiquetas distintas y querremos saber cuántos elementos hay, sus posiciones relativas a la imagen y la categoría a la que pertenecen cada una de ellas.

La gran mayoría de los paradigmas de detección utilizan una red neuronal pre-entrenada como clasificador dentro de sus estructuras. Suele recibir modificaciones en sus últimas capas ya sea sustituyéndolas, pasando por un proceso de *fine-tunning*, o eliminándolas antes de ser añadidas como un elemento invariante durante el entrenamiento del modelo. Estas redes suelen recibir el nombre de *backbone* y algunos modelos permiten la libre elección de estos clasificadores dependiendo del problema concreto que se desee abordar.

Para indicar las posiciones de los elementos, por lo general se utilizarán *bounding-box* o *bbox* que serán rectángulos que contendrán cada uno de los elementos o *pixel level* que colorea cada uno de los píxeles pertenecientes a determinada categoría, ambos con la mayor precisión posible.



(a) Image Classification (b) Object Detection

(c) Semantic Segmentation (d) Instance Segmentation

Figura 5.1.: Detección, clasificación y segmentación. [WSH19]

5.1. Dos pasos o basados en RCNN

El problema de detección se divide en dos etapas: una generación de propuestas y la realización de predicciones sobre estas propuestas. Como paradigmas de detección, destacan la familia *R-CNN* y los que se encuentran basados en esta.

Esta familia surge en noviembre de 2013 con *R-CNN* [GDDM13] que utiliza Selective Search [**?**] para generar 2000 *bbox* de propuestas que son redimensionadas para coincidir con las dimensiones de entrada una CNN pre-entrenada. Esta CNN debe volver a entrenarse, extrayéndole la última capa y añadiéndole una *máquina de vectores de soporte (SVM)* con las categorías originales más una nueva clase llamada "fondo"que engloba a todas las propuestas que no pertenecen a ninguna categoría. Finalmente, las propuestas clasificadas son combinadas a través de un modelo de regresión lineal para así obtener una *bbox* con mejor ajuste y precisión.

La principal diferencia que introduce *Fast R-CNN* [Gir15] reside en que, en lugar de pasar por la CNN todas las propuestas de Selective Search, nuestra CNN recibe como entrada la imagen completa reduciendo el coste computacional al analizar una única vez las zonas de solapamiento entre propuestas. Además, las propuestas dejan de ser escaladas para coincidir con una determinada dimensión y se introduce en una capa especial llamada *Region of Interest Pooling Layer (RoI pooling)* que extrae un vector de longitud fija que será la entrada de las siguientes ramas. A continuación, el modelo se divide en dos ramas: un clasificador SoftMax y un modelo de regresión que calcula las *bbox*.

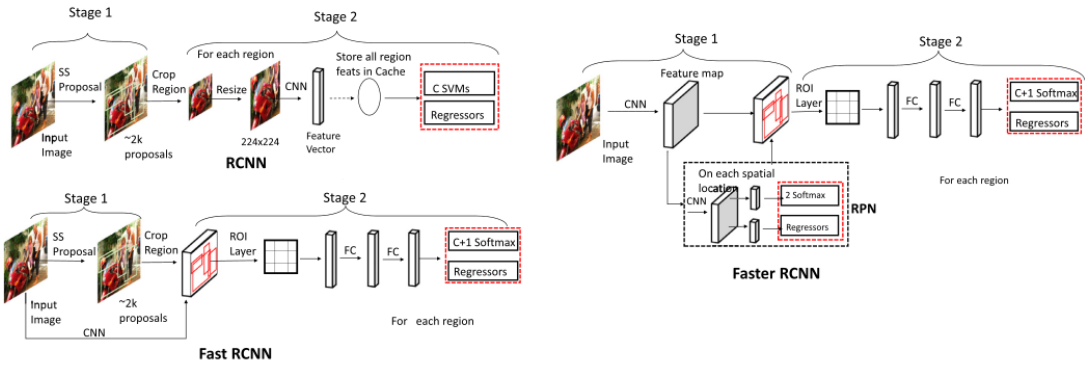


Figura 5.2.: Comparativa de algunos modelos de la familia R-CNN. [WSH19]

En *Faster R-CNN* [RHGS15] se sustituye Selective Search por una red neuronal de generación propuestas, conocida como *anchor boxes*, que no sólo reduce el tiempo de cómputo de la generación de propuestas sino que aporta un menor número de propuestas con mayor calidad y precisión. Un buen apunte de la red de generación de propuestas, es que esta no es pre-entrenada antes de ser añadido al modelo sino que aprende de forma conjunta con toda la red.

La última mejora de esta familia viene representada por *Mask R-CNN* [HGDG17] que sus-

tituye la capa RoI pooling introducida en el modelo Fast R-CNN por una RoI alignment que retiene más información de las características obtenidas por la CNN compartiendo la misma estructura de entrada y salida de datos con la RoI pooling. Mask R-CNN aprovecha esto para realizar predicciones del tipo pixel level con mayor precisión a través de una interpolación bilineal que se ejecuta paralelamente con las capas totalmente conectadas de los modelos anteriores. Así, en el mismo punto que sus predecesores eran divididos en tres ramas independientes, Mask R-CNN se divide en tres ramas: un clasificador SoftMax, un regresor de *bbox* y un otro regresor de pixel level.

Mencionar Mesh R-CNN https://arxiv.org/abs/1906.02739
Mencionar R-FCN https://arxiv.org/pdf/1605.06409.pdf
Libra R-CNN https://arxiv.org/pdf/1904.02701.pdf

5.2. Un paso o basados en YOLO

Estos modelos destacan por no hacer una separación directa de la generación de propuestas y la predicción de estas. Destaca *YOLO* y sus sucesivas mejoras, así como los múltiples algoritmos basados en ellas, pero existen muchos más modelos que pueden ser identificados con esta estructura.

En junio de 2015 aparece *You Only Look Once (YOLO)* [RDGF15], un nuevo algoritmo que pretende realizar al mismo tiempo la generación de propuestas y la clasificación de estas, buscando así una mayor eficiencia computacional. Para ello, tras redimensionar la imagen en 448x448 píxeles, el algoritmo divide la imagen en una cuadrícula cuyas celdas se encargan de realizar un número fijo de propuestas indicando en cada una de ellas la probabilidad que tiene de ser un objeto, la *bbox* en la que se encuentra y la probabilidad de pertenecer a cada una de las clases de objetos de las que disponemos. Finalmente, descarta aquellas propuestas con baja probabilidad de ser un objeto y utiliza un algoritmo de *non-max supression* [BSCD17] que unifica y combina las *bbox* con las máximas áreas compartidas.

Buscando una mejora de la predicción pero sin perder el enfoque de la velocidad, surge *YOLO9000* o *YOLOv2* [RF16] que sigue la misma estructura que su predecesor pero sustituyendo algunos fragmentos por otros con mayor rendimiento. YOLOv2 redimensiona la imagen original a 416x416 píxeles y sustituye las capas totalmente conectadas que utilizaba para la generación de propuestas por un modelo de *anchor boxes* [RHGS15] junto con otros cambios menores.

La siguiente versión *YOLOv3* [RF18] decide no darle tanta importancia a la velocidad y centrarse en solucionar los problemas de detección presentados por sus antecesores. Mientras que YOLOv2 utiliza una arquitectura con 30 capas, YOLOv3 utiliza 106 capas totalmente convolucionadas e introduce la utilización de *residual blocks*, *skip connections* y *upsampling*. Esta nueva arquitectura, realiza la detección de objetos en tres escalas distintas en diferentes profundidades de la red Darknet-53 y utilizando una estructura piramidal [LDG+16] para comunicar la detección entre las diferentes escalas.

Mencionar YOLOv4 https://arxiv.org/pdf/2004.10934.pdf

5.3. Non-local neural networks

En [WGGH17] se introduce el concepto de *non-local neural networks* que utiliza una arquitectura ascendente y descendente, *up-down architecture* [PLCD16], para extraer las características de un clasificador a distintos niveles de profundidad y utilizarlas para segmentar semánticamente la imagen Figura 5.1.

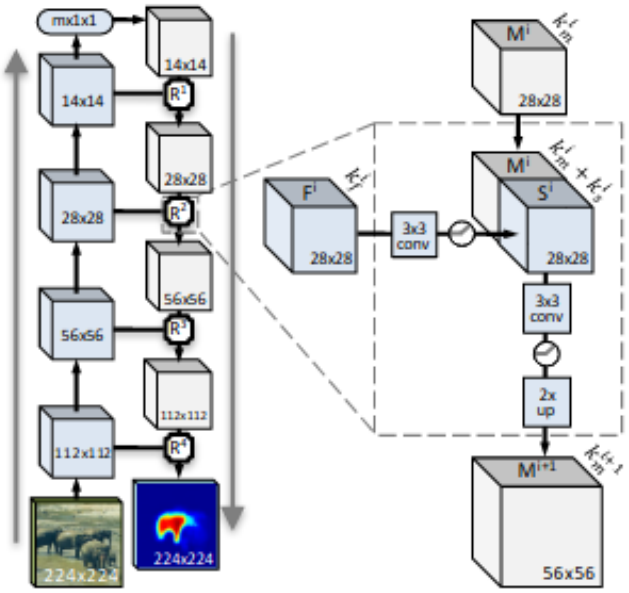


Figura 5.3.: Ejemplo de una arquitectura ascendente y descentente con saltos de conexiones[PLCD16].

En la parte ascendente de este tipo de arquitecturas nos encontraremos con un clasificador de imágenes, del cual extraeremos información a distintos niveles de profundidad que será unificada y analizada en la parte descendente de la arquitectura para poder predecir los distintos segmentos semánticos.

Las *non-local neural networks* se distinguen por la utilización de *non-local block* o bloques no locales como extractores de características en los distintos niveles de profundidad. Estos bloques serán de la forma:

$$z_i = W f_i(x_i) + x_i,$$

donde i representa el índice de una posición de salida, x_i el i-ésimo elemento de la entrada x, f_i la i-ésima coordenada de una operación no local f (Capítulo 4) y W un peso que será entrenado por la red.

5.4. Experimentos

Para el desarrollo de la red neuronal que se ha implementado, se ha utilizado el artículo *Real-time Semantic Segmentation with Fast Attention* [HPC+20] publicado el 9 de julio de 2020. En este artículo se busca implementar una red neuronal capaz de realizar segmentación semántica en tiempo real, sin una gran pérdida de precisión y pudiendo ser ejecutada en dispositivos con pocos recursos.

Siguiendo sus ideas, se ha utilizado como clasificador un ResNet-18 [HZRS15]. Como bloque no local se ha utilizado el, que los autores denominan, *fast attention module*, que implementa como operación no local Capítulo 4 una *dot-product similarity* con $v(x_j) = W_{v_j} x_j$. Para reconstruir las formas a través de las características extraídas nos hemos basado en el modelo U-Net [RFB15].

A. Primer apéndice

Los apéndices son opcionales.
Archivo: `apendices/apendice01.tex`

Glosario

La inclusión de un glosario es opcional.
Archivo: `glosario.tex`

$\mathbb{R}$ Conjunto de números reales.

$\mathbb{C}$ Conjunto de números complejos.

$\mathbb{Z}$ Conjunto de números enteros.

Bibliografía

Las referencias se listan por orden alfabético. Aquellas referencias con más de un autor están ordenadas de acuerdo con el primer autor.

[aut] Varios autores. Convolutional neural networks for visual recognition. [Citado en págs. IX, IX, IX, 4, and 5]

[BCM05] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. A non-local algorithm for image denoising. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02*, CVPR '05, pages 60–65, Washington, DC, USA, 2005. IEEE Computer Society. [Citado en págs. 16 and 17]

[BM12] Joan Bruna and Stéphane Mallat. Invariant Scattering Convolution Networks. *arXiv e-prints*, page arXiv:1203.1513, March 2012. [No citado]

[BSCD17] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S. Davis. Soft-NMS – Improving Object Detection With One Line of Code. *arXiv e-prints*, page arXiv:1704.04503, April 2017. [Citado en pág. 21]

[Cyb89] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989. [Citado en pág. 9]

[GDDM13] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv e-prints*, page arXiv:1311.2524, November 2013. [Citado en pág. 20]

[Gir15] Ross Girshick. Fast R-CNN. *arXiv e-prints*, page arXiv:1504.08083, April 2015. [Citado en pág. 20]

[HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. *arXiv e-prints*, page arXiv:1703.06870, March 2017. [Citado en pág. 20]

[HPC+20] Ping Hu, Federico Perazzi, Fabian Caba Heilbron, Oliver Wang, Zhe Lin, Kate Saenko, and Stan Sclaroff. Real-time Semantic Segmentation with Fast Attention. *arXiv e-prints*, page arXiv:2007.03815, July 2020. [Citado en pág. 23]

[HSW89] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. [Citado en pág. 10]

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. [Citado en pág. 23]

[HZXL19] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. *CoRR*, abs/1904.11491, 2019. [No citado]

[KH91] Kurt and Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991. [Citado en págs. 9 and 10]

[KL19] Patrick Kidger and Terry Lyons. Universal Approximation with Deep Narrow Networks. *arXiv e-prints*, page arXiv:1905.08539, May 2019. [Citado en pág. 11]

[LDG+16] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection. *arXiv e-prints*, page arXiv:1612.03144, December 2016. [Citado en pág. 21]

[LPM15] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025, 2015. [No citado]

[LPW+17] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The Expressive Power of Neural Networks: A View from the Width. *arXiv e-prints*, page arXiv:1709.02540, September 2017. [Citado en pág. 11]

[OMS17] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization. [No citado]

[PLCD16] Pedro O. Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollàr. Learning to Refine Object Segments. *arXiv e-prints*, page arXiv:1603.08695, March 2016. [Citado en págs. IX and 22]

[RDGF15] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. *arXiv e-prints*, page arXiv:1506.02640, June 2015. [Citado en pág. 21]

[RF16] Joseph Redmon and Ali Farhadi. YOLO9000: Better, Faster, Stronger. *arXiv e-prints*, page arXiv:1612.08242, December 2016. [Citado en pág. 21]

[RF18] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement. *arXiv e-prints*, page arXiv:1804.02767, April 2018. [Citado en pág. 21]

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv e-prints*, page arXiv:1505.04597, May 2015. [Citado en pág. 23]

[RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv e-prints*, page arXiv:1506.01497, June 2015. [Citado en págs. 20 and 21]

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. [Citado en pág. 15]

[WGGH17] Xiaolong Wang, Ross B. Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. *CoRR*, abs/1711.07971, 2017. [Citado en pág. 22]

[WSH19] Xiongwei Wu, Doyen Sahoo, and Steven C. H. Hoi. Recent Advances in Deep Learning for Object Detection. *arXiv e-prints*, page arXiv:1908.03673, August 2019. [Citado en págs. IX, IX, 19, and 20]