

APRENDIZAJE PROFUNDO APLICADO A CLASIFICACIÓN DE ESCENAS DE PROPIEDADES INMOBILIARIAS

Ignacio Gabriel Franco

Trabajo final de grado presentado al Departamento de Matemática Aplicada de la Universidad Católica de Santiago del Estero para optar al grado académico de Ingeniería en Informática.

Diciembre 2019
Rafaela, Argentina

Director: Mariano Ferrero

Resumen

En este trabajo se investigó sobre la aplicación de técnicas de Aprendizaje Profundo aplicado a la clasificación de Escenas de Propiedades Inmuebles. Se revisaron trabajos previos sobre clasificación de escenas del mismo tipo y de otros con el fin de conocer qué enfoques podrían funcionar mejor. Se plantearon y comprobaron hipótesis que incluyen el entrenamiento de Redes Neuronales Convolucionales desde cero y el Aprendizaje Mediante Transferencia de Redes Neuronales Preentrenadas. A través de la validación de las hipótesis se obtuvieron modelos capaces de predecir determinados conjuntos de escenas de propiedades inmuebles. Se analizaron las predicciones de los dos mejores modelos obtenidos, haciendo énfasis en el error que cometan a partir de las probabilidades que entregan. Gracias a esta investigación se llegó a la conclusión de que es posible entrenar modelos de Aprendizaje Profundo para la clasificación de escenas pero que la capacidad de predicción estará sujeta a la calidad de las imágenes con las que se hayan entreado.

El presente documento, el código utilizado y los modelos entrenados quedan disponibles para uso bajo la Licencia MIT, en el [repositorio del trabajo](#).

Keywords: Scene Classification, Deep Learning, Real Estate, Convolutional Neural Network, Transfer Learning.

Palabras clave: Clasificación de Escenas, Aprendizaje Profundo, Propiedades Inmuebles, Redes Neuronales Convolucionales, Aprendizaje por Transferencia.

*Dedicado a mi padre, que me guía
y acompaña en cada paso que doy.*

Agradecimientos

A mi familia por brindarme su apoyo y amor incondicional, a mi director por ayudarme en cada ocasión y mantenerse atento a mis necesidades y al presente director de la carrera de Ingeniería en Informática por darme las herramientas necesarias para desarrollarme durante la carrera.

Índice

1. Introducción	6
1.1. Contexto	6
1.2. Motivación	7
1.3. Estructura del documento	11
2. Revisión de antecedentes	13
2.1. Clasificación de escenas mediante Redes Neuronales Recurrentes	13
2.2. Clasificación de escenas mediante Redes Neuronales Convolucionales	16
2.3. Clasificación de escenas mediante Aprendizaje por Transferencia	26
2.4. Otros trabajos relacionados a las propiedades inmunes que toman provecho de las imágenes de los mismos	29
2.5. Antecedentes no académicos	31
3. Marco teórico	33
3.1. Aprendizaje automático	33
3.2. Representación de una neurona	34
3.3. Red Neuronal	35
3.4. Red Neuronal Convolutional	41
3.5. Redes preentrenadas y Aprendizaje por transferencia	44
4. Diseño experimental	47
4.1. Asunciones	47
4.2. Limitaciones	47
4.3. Hardware a utilizar	48
4.4. Hipótesis	48
4.4.1. Hipótesis 1	48
4.4.2. Hipótesis 2	48
4.4.3. Hipótesis 3	49
4.4.4. Hipótesis 4	49
4.4.5. Hipótesis 5	49

4.4.6. Hipótesis 6	50
4.5. Experimentos	50
4.5.1. Experimento 1	50
4.5.2. Experimento 2	52
4.5.3. Experimento 3	53
4.5.4. Experimento 4	56
4.5.5. Experimento 5	58
4.5.6. Experimento 6	59
5. Análisis de los resultados obtenidos	62
5.1. Error por categoría modelos obtenidos en los experimentos 4.5.2 y 4.5.6	62
5.2. Distribución de probabilidades de salida para clasificaciones erróneas	62
5.3. Exactitud categórica casos que deberían poder predecirse correctamente	65
6. Conclusiones	69
6.1. Conclusión general	69
6.2. Lineamientos Futuros	71
7. Anexo	73
7.1. Anexo 1: Repositorio abierto del trabajo	73
7.2. Anexo 2: Licencia del proyecto	74
7.3. Anexo 3: Mapeo de clases realizado para el experimento 4.5.3	75
7.4. Anexo 4: Arquitectura del Perceptrón Multicapa utilizado en experimento 4.5.1	75
7.5. Anexo 5: Arquitectura de la Red Neuronal Convolucional utilizada en los experimentos 4.5.1 y 4.5.2 . . .	77
7.6. Anexo 6: Arquitectura de la red PlacesCNN utilizada en experimento 4.5.3	77
7.7. Anexo 7: Arquitectura utilizada en experimentos 4.5.4 y 4.5.6	77
7.8. Anexo 8: Arquitectura utilizada en experimento 4.5.5	77

1. Introducción

1.1. Contexto

De la mano del avance tecnológico tanto en materia de hardware como de software, en los últimos años ha sido posible explotar de forma más efectiva y eficiente una rama algo olvidada de la inteligencia artificial: las redes neuronales.

Esta rama ha demostrado en múltiples ocasiones ser capaz de obtener resultados significativos en tareas de detección y localización de objetos, clasificación de escenas, segmentación de imágenes y detección de rostros (entre otras).

Estas prácticas tienen una gran aplicación en la industria como la detección y localización de objetos como semáforos, transeúntes, automóviles y otros relacionados al tráfico para autos que se conducen por sí mismos, detección de rostros para controles de ingreso de personas a aeropuertos como también a empresas, detección de elementos aplicado a imágenes médicas (posibles tumores o malformaciones).

En este trabajo se hará frente a uno de los ejes inicialmente mencionados: la clasificación de escenas. Esta asignatura que resulta prácticamente trivial para una persona incluye un conjunto de actividades complejas por sí mismas: detección de objetos locales y su disposición dentro de la escena, entorno de fondo, distinción de características entre escenas parecidas, cultura de la que proviene la escena, calidad de reconocimiento de una escena y muchas más. Actualmente, mediante actividades de investigación, competiciones y necesidades del sector privado se han logrado significativos resultados en clasificación de escenas relacionadas a diferentes contextos: distinción entre lugares de una ciudad, clasificación de zonas de la misma a partir de imágenes satelitales, ambientes de una propiedad, escenas relacionadas al tráfico de una ciudad, etc.

A priori, en una fugaz interpretación de la tarea a realizar, se la podría definir como la clasificación de imágenes tradicional, aunque no sea así. En esta actividad se puede destacar tanto detección

de objetos como, en algunos casos, la definición de su contorno, sin importar el tamaño del mismo o su posicionamiento dentro de la imagen. La tarea de reconocer escenas acapara varias otras aristas, como son la disposición de los objetos en la imagen, los elementos que se encuentren en la misma, el ambiente en el que se encuentren, el fondo, entre otras. En una escena existen múltiples objetos en diferentes escalas, enfocados desde diferentes ángulos y disposiciones, mientras que en la clasificación de imágenes se suele tratar con un único objeto centrado. Éstas son, entre muchas otras, algunas de las principales diferencias entre clasificación de escenas e imágenes, dos tareas que pertenecen a un mismo tópico pero que no es posible solucionarlas totalmente utilizando el mismo enfoque para el problema.

Dado que hasta hace no muchos años la cantidad de imágenes a clasificar no alcanzaba el orden de magnitud que se tiene actualmente, queda claro que era posible de abordar la necesidad mediante tareas realizadas por individuos. Siendo que actualmente el flujo de información es mucho mayor, resulta que la automatización del etiquetado imágenes pasó a ser un requerimiento para determinados entornos como empresas de venta y/o alquiler de propiedades, intermediarios dentro de la misma actividad o sitios en los que se suben imágenes de este tipo y se las quiere mantener etiquetadas de forma inmediata.

Dentro de los posibles enfoques a utilizar descriptos en [1] se encuentran las representaciones esparsas, máquinas de soporte vectorial, redes neuronales artificiales y redes neuronales convolucionales. Dentro de las redes neuronales se suelen usar diferentes arquitecturas en calidad de obtener los mejores resultados, dependiendo de en qué manera se estructure la información. Estas arquitecturas son las Redes Neuronales Convolucionales, las Redes Neuronales Recurrentes y el Aprendizaje por Transferencia.

1.2. Motivación

La clasificación de imágenes relacionadas a propiedades inmuebles hace referencia a la capacidad de etiquetar automática y correcta-

mente imágenes de escenas relacionadas a diferentes partes de los mismos de manera tal que luego sea posible consumir la información de cada sector de manera individual por cada bien. Una actividad que resulta altamente atractiva y beneficiosa cuando se trata con cientos o miles de imágenes de propiedades y se quiere explotar esta información para otro tipo de tareas. Dentro de los beneficios más destacables se pueden mencionar la automatización de tarea de etiquetado, las mejoras en sistemas que requieran este tipo de tareas, el ahorro de tiempo para clasificar imágenes de este dominio, entre otras.

En el contexto actual existen empresas que brindan una larga lista de servicios relacionados a las propiedades inmuebles y que la resolución de este problema les sería de gran ayuda tanto en las tareas del día a día como para explotar de mejor manera la información de los inmuebles que ya tienen almacenada internamente. Este tipo de empresas u organizaciones son las que se dedican a actividades como: la venta y alquiler de bienes propios, la intermediación entre residentes y dueños para alquileres temporales, la valuación y control del estado de las propiedades, entre otras. Dentro de las posibles aplicaciones y ventajas que puede otorgar el adjudicarse con un modelo que se dedique a realizar esta actividad se destacan:

- Clasificación automática de las escenas: el hecho de poder clasificar las escenas de cada propiedad de la que se tiene conocimiento permite no sólo un mejoramiento de calidad de información sino también la posibilidad de poder aprovecharla para ser explotada a posteriori.
- Validación de escenas requeridas: un claro ejemplo de mejora en el proceso de posteo de propiedades, ya sea para alquiler o para venta, es solicitar imágenes de los diferentes ambientes de la propiedad, validando mediante un modelo de este tipo que se provean imágenes de los ambientes que se consideren más importantes (por ejemplo, si se quiere postear un departamento en un sitio de alquileres, que se valide la existencia de imágenes

de la cocina, el comedor, el dormitorio y el living, con el fin de mejorar la calidad de información que luego se brindará a quienes buscan alquilar). Sin un modelo encargado de esta tarea, esta mejora no es posible de obtenerse a gran escala.

- Extracción de características relevantes: a partir de un modelo de aprendizaje profundo, como se verá posteriormente, es posible extraer conocimiento para resolver otro tipo de problemas. Esta técnica se conoce Aprendizaje por Transferencia.
- Recomendación de escenas: un posible enfoque luego de contar con las imágenes etiquetadas y puesto en producción un modelo que valide qué imágenes se postean de las propiedades, es posible realizar un análisis de qué relación existe entre las imágenes que la persona que alquila ve de la propiedad y la conversión del poste (ya sea una venta, un contacto para visitarla o un alquiler temporal). De esta manera se podría brindar un mejor feedback a los propietarios para que aumenten sus probabilidades de convertir recomendando escenas faltantes o aquellas que mejores resultados podrían brindarle.
- Valoración automática o refinamiento de valoraciones de propiedades a partir de imágenes: en [2] se demuestra que es posible mejorar los resultados obtenidos en tareas de valuación automática de inmuebles utilizando las imágenes de los ambientes de los mismos.
- Otras aplicaciones que se pueden encontrar o se encontrarán en un futuro próximo en el mercado actual.

Dentro de la tarea previamente mencionada de valuación de inmuebles existen cientos de factores que componen el valor final de los mismos, tales como el terreno total ocupado y el construido, los años desde su construcción, la cantidad de habitaciones de cada tipo, los precios de los inmuebles circundantes y los precios de los inmuebles similares, el estado del inmueble, entre otros. Dada la complejidad

de la cantidad de información a tener en cuenta y las diferentes formas en que se presenta (variables numéricas y categóricas, datos no estructurados como imágenes, etc), en conjunto con el alto número de inmuebles que se necesitan tasar en algunos casos se han implementado diferentes enfoques de valoración automática.

En [2] se demuestra que a partir de la información que brindan las imágenes de los inmuebles y mediante aprendizaje profundo es posible mejorar los resultados de los enfoques que no cuentan con esta información y dedican su esfuerzo a realizar las tasaciones sólo con la información tabular y estructurada. Teniendo en cuenta este último punto, queda claro que poder absorber la mayor cantidad de información sobre estas imágenes resulta una tarea de alta significancia. Uno de los enfoques utilizados para hacerlo es basar la estimación del precio únicamente a partir de las imágenes que se tienen de la propiedad, y luego agregar los datos estructurados para mejorar los resultados.

Por otro lado, en actividades como el control o chequeo de imágenes que se suben al hacer publicaciones de propiedades o las sugerencias de escenas a subir dentro de las mismas, reconocer de qué habitación o vista se trata en cada caso resulta la única manera de brindar esta información al usuario.

Si de la agrupación de imágenes similares se trata, entonces luego de obtener la relación entre precio con el que se venden las propiedades y las características de las diferentes partes de las mismas sería posible conocer qué particularidades tienen aquellas propiedades que logran mayor margen de ganancia al momento de venderlas, qué se puede hacer con ciertas partes del inmueble para que su valor suba en relación a las cualidades de sus diferentes sectores y otras tareas relacionadas que brindarían un alto valor al momento de postular a la venta una propiedad.

Por último, es importante destacar la posibilidad de extraer el conocimiento adquirido por un modelo encargado de hacer estas tareas para hacer frente o refinar resultados de otros modelos, que no necesariamente estén fuertemente relacionados con la clasificación

de escenas de propiedades inmuebles.

Como se mostrará a partir de la revisión de antecedentes, la utilización de aprendizaje profundo para clasificar escenas es una opción altamente viable por los resultados que se obtienen. De igual manera, es cierto que aún existe un margen de error en estos modelos, que por un lado puede provenir por los datos que se utilizan para entrenarse y por otro por la gran cantidad de cuestiones a tener en cuenta en relación a las configuraciones de la red a utilizar (arquitectura, funciones de pérdida, optimizadores, métrica tener en cuenta, etc), lo que da lugar a continuar investigando con el fin de seguir refinando los modelos y eventualmente disminuir el margen de error en los resultados. Este trabajo hará foco en esta problemática, a través de la investigación de diferentes métodos de aprendizaje profundo que actualmente alcanzan el estado del arte, utilizando métricas que ya se utilicen en este tópico y conjuntos de datos para que tanto la diversidad como la densidad de las escenas sea la suficiente para brindar conclusiones correctas con respecto a esta tarea.

1.3. Estructura del documento

En este trabajo se investigarán diferentes métodos de clasificación de escenas aplicado a propiedades inmuebles. Con el fin de alcanzar el estado del arte en esta tarea y eventualmente intentar mejorar estos resultados, la composición del trabajo será la siguiente: en el capítulo segundo se realizará una revisión de antecedentes en materia de utilización de técnicas de aprendizaje profundo para clasificar escenas. En el capítulo tercero se definirá el marco teórico y otros conceptos necesarios para entender el funcionamiento de este tipo de técnicas. En el capítulo cuarto se enunciarán las limitaciones e hipótesis del proyecto, también se realizarán los experimentos para contrastar estas hipótesis. En el capítulo quinto se realizará un análisis de los resultados obtenidos por aquellos modelos que mejor funcionaron. En los capítulos sexto y séptimo de definirán los experimentos a realizar y sus resultados, respectivamente. Para finalizar, en el capítulo séptimo, se declararán las conclusiones del trabajo y

posibles lineamientos futuros.

2. Revisión de antecedentes

2.1. Clasificación de escenas mediante Redes Neuronales Recurrentes

En [3] J. H. Bappy y otros introducen un framework para aprender la información de las escenas de manera secuencial. Además, generan y hacen público una base de datos de aproximadamente 6000 imágenes etiquetadas pertenecientes a las seis partes que consideran más importantes a clasificar de una propiedad inmobiliaria: frente, patio, dormitorio, baño, living y cocina.

Para empezar con la estructura que proponen, es necesario explicar el algoritmo de preprocessamiento que aplican a cada imagen con el fin de intensificar los contrastes locales de la misma. Se trata de una de las variantes de ecualización de histograma de las imágenes llamada Ecualización de Histograma Adaptativo con Limitación de Contraste (o por sus siglas en inglés C.L.A.H.E., Contrast Limited Adaptive Histogram Equalization). Este algoritmo es la evolución de HE (Histogram Equalization) y de AHE (Adaptive Histogram Equalization). La Ecualización del Histograma de la imagen incrementa el contraste global de las mismas, sobretodo cuando los datos representativos de la imagen están representados por valores de contraste cercanos. Por otro lado, la Ecualización del Histograma Adaptativo computa múltiples histogramas correspondientes a diferentes secciones de la imagen, utilizándolos para redistribuir la luminosidad de la misma. La principal mejora es que obtiene mejores contrastes en sectores locales de la imagen, y define mejor límites dentro de cada región. Este método tiende a amplificar por demás el ruido en regiones relativamente homogéneas de la imagen, la Ecualización del Histograma Adaptativo con Limitación de Contraste se encarga de prevenir esta situación limitando la amplificación, podemos observar un ejemplo de aplicación de esta técnica en la Fig. 1.

La propuesta de estos investigadores se basa en aprender la información de la escena secuencialmente, tanto de forma vertical como

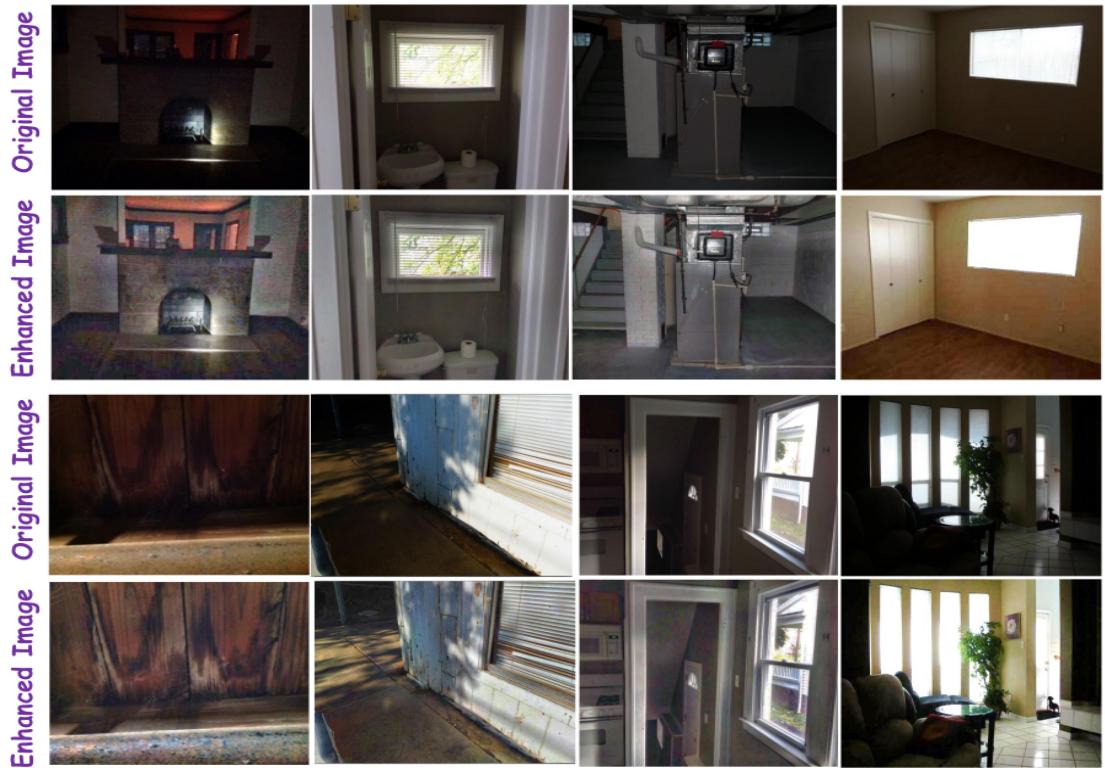


Figura 1: Ejemplo aplicación del algoritmo C.L.A.H.E.

horizontal. Para hacerlo, crearon dos redes recurrentes LSTM (Long Short Term Memory) con cuatro (4) capas ocultas de ciento veintiocho (128) unidades cada una. Como segundo punto, alimentan a la red con los datos de las imágenes de forma secuencial, es decir, transforman las imágenes a un tamaño de 128x128 y luego alimentan la red con la información secuencial de cada pixel de forma vertical para una de las redes y de forma horizontal para la otra. Para finalizar, introducen una capa densa totalmente conectada (fully connected layer, en inglés) con las salidas de ambas LSTM, seguida de otra capa densa totalmente conectada que concluye con una capa densa final con activación Softmax.

La arquitectura final propuesta como se puede observar en la Fig. 2, queda de la siguiente manera: dos redes LSTM que reciben los pixeles orientados de forma vertical y horizontal, respectivamente;

una capa densa totalmente conectada que recibe las salidas de cada celda de la red LSTM, otra capa densa totalmente conectada que se concatena a la anterior y una capa densa final con activación Softmax que brinda las salidas. Vale mencionar que es necesaria la aplicación del algoritmo CLAHE a cada imagen para el posterior entrenamiento y clasificación mediante la red propuesta.

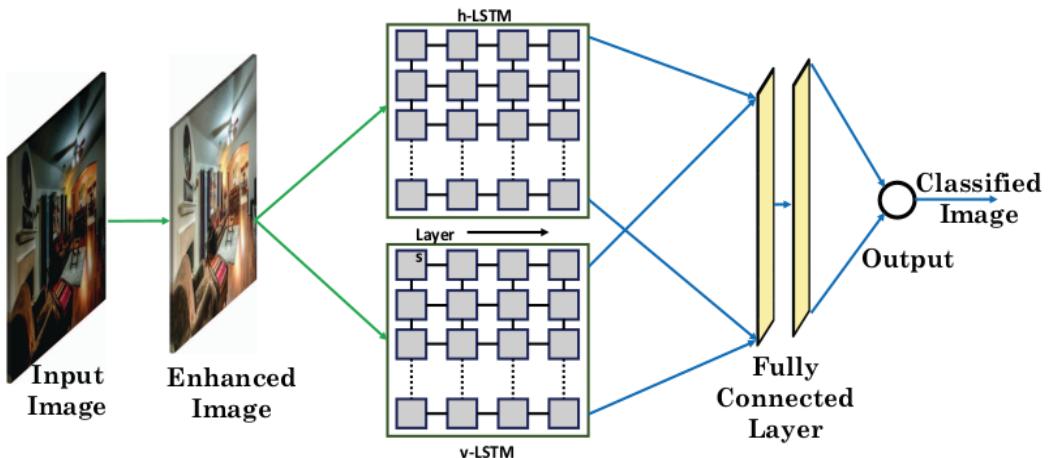


Figura 2: Arquitectura de la red presentada

Las comparaciones realizadas por los investigadores se basan en la métrica Exactitud (Accuracy, en inglés), se emplean utilizando diferentes configuraciones de esta red, y en relación al conjunto de datos propuesto por ellos (Real Estate Images, en inglés) además del conjunto de datos abierto *SUN*. Los investigadores obtienen como mejores valores un 96.92 % de exactitud en el conjunto de datos propuesto *REI*, y un 90.24 % en el la base de datos de imágenes *SUN*, utilizando la configuación descripta anteriormente. De esta manera quedan por encima de los resultados de utilizar extracción de características de redes preentrenadas con el conjunto de datos abiertos *ImageNet* con las arquitecturas AlexNet y VGGNet.

2.2. Clasificación de escenas mediante Redes Neuronales Convolucionales

En [4] se creó una nueva base de datos con siete (7) millones de imágenes de escenas etiquetadas. Zhou y otros utilizaron Redes Neuronales Convolucionales para aprender características profundas sobre las escenas y alcanzar un nuevo estado del arte. Ellos se encargaron de mostrar que las características de más alto nivel aprendidas por redes neuronales profundas en conjuntos de datos centrados en objetos y a escenas son diferentes: imágenes de objetos no contienen la riqueza y diversidad de información visual que brindan las imágenes de escenas y ambientes para poder identificarlos.

Su trabajo comienza por la construcción del banco de imágenes: creación de URLs a partir de sustantivos y adjetivos relacionados a las escenas, eliminación de los links duplicados y luego de realizada la descarga, la eliminación de imágenes que se encuentren ya en el SUN database. Comparar la calidad del conjunto de datos generado en relación a otros de similar magnitud depende no sólo de las imágenes que contengan y sus categorías, sino también de múltiples factores como la variabilidad de las posiciones de la cámara, los estilos de decorado, la ubicación y el tamaño que los objetos ocupan dentro de la imagen, etc. Es razonable asumir que una buena base de datos de imágenes debería ser densa y diversa. La densidad de una medida de concentración de los datos, y una base de datos de imágenes debe serlo ya que para aprender sobre un elemento (en este caso escenas) es necesario que haya alto grado de concentración de ese elemento. La realidad es que la densidad de un conjunto de datos no alcanza, ya que si se cuenta con todas imágenes de la misma habitación tendrá una densidad muy alta pero una diversidad muy escasa. La diversidad es una medida relacionada a la cantidad de clases en un conjunto de datos; una base de datos de imágenes debe ser diversa porque es necesario que haya tanto múltiples elementos dentro de la misma como también variabilidad de enfoques para sus imágenes. Ambas medidas son difíciles de medir en conjuntos de datos de imágenes. En este caso, los autores proponen dos medidas: densidad relativa

y diversidad relativa. Para la primera los autores asumen que, en el dominio de las bases de datos de imágenes, una alta densidad es equivalente a que en general las imágenes tienen vecinos similares. Por esta razón para realizar la medición toman una imagen aleatoria de un conjunto de datos A (llámese a_1) y una imagen aleatoria del segundo conjunto de datos B (llámese b_1). Si el conjunto A es más denso que el conjunto B , entonces es más probable que a_1 tenga menor distancia d a su vecino más cercano que b_1 , siendo d una medida de distancia entre dos imágenes en la que menores valores significa mayor similaridad. Con esta definición, se tiene que A es más denso que B si y sólo si la densidad de A dado B es mayor que la densidad de B dado A .

$$Den_B(A) = p(d(a_1, a_2) < d(b_1, b_2)) \quad (1)$$

Esta noción de densidad también puede aplicarse a múltiples conjuntos de datos A_1, \dots, A_N :

$$Den_{A_2, \dots, A_N}(A_1) = p\left(d(a_{11}, a_{12}) < \min_{i=2:N} d(a_{i1}, a_{i2})\right) \quad (2)$$

Si de diversidad se trata, existen varias formas de medirla que mayormente se utilizan en biología para conocer la riqueza de un ecosistema. Para este trabajo los investigadores se basaron en el índice Simpson de diversidad que es una medida de qué tan bien están distribuidos los individuos de las diferentes especies en un ecosistema, y está relacionado a la entropía de la distribución de los mismos. Ellos proponen medir la diversidad relativa de dos conjuntos de datos A y B basándose en la idea de que A resultará más diverso si al seleccionar aleatoriamente dos imágenes de B resultan más similares visualmente que seleccionar aleatoriamente dos imágenes de A . De esta manera, la diversidad de A con respecto a B puede ser definida como:

$$Div_B(A) = 1 - p(d(a_1, a_2) < d(b_1, b_2)) \quad (3)$$

donde a_1 y a_2 pertenecen a A y b_1 y b_2 pertenecen a B y fueron todas las imágenes seleccionadas aleatoriamente. De igual manera a

la medida anterior, es posible de ser calculada entre más bases de datos de imágenes A_1, \dots, A_N :

$$Div_{A_2, \dots, A_N}(A_1) = 1 - p \left(d(a_{11}, a_{12}) < \min_{i=2:N} d(a_{i1}, a_{i2}) \right) \quad (4)$$

siendo $a_{i1}, a_{i2} \in A_i$ seleccionados aleatoriamente.

En el marco de la experimentación realizada para demostrar que las redes aprenden características diferentes según el tipo de conjunto de datos que se utilice para entrenarlas los investigadores se quedaron con aproximadamente 2.48 millones de imágenes correspondientes a 205 categorías con un mínimo de cinco mil y un máximo de quince mil imágenes por cada una como set de entrenamiento (al cual se refieren como "Places205"). El set de validación se seleccionó con cien imágenes por escena y el set de test doscientas, alcanzando un total de cuarenta y un mil imágenes entre estas dos últimas particiones. Finalizado el entrenamiento, los investigadores comparan las respuestas de unidades de varias capas de la red para entender mejor las diferencias entre ImageNet-CNN y Places-CNN, dos redes que tienen idéntica arquitectura pero que fueron entrenadas con conjuntos de datos de datos creados para diferentes propósitos (objetos y escenas, respectivamente). Las diferencias mayores entre las activaciones se dan a partir de la capa de pooling número dos gradualmente hasta la número cinco y también en la capa totalmente conectada número siete, en las que para ImageNet-CNN los campos receptivos se asemejan más a partes de objetos mientras que para Places-CNN en las mismas capas los campos receptivos parecen ser paisajes o estructuras relacionadas a un espacio. En Fig. 3 los investigadores hacen una comparación sobre conjuntos de imágenes tanto centradas en objetos como en escenas. La métrica utilizada es exactitud y las bases de datos de imágenes con que compararon fueron: SUN397, MIT INDOOR67, SCENE15, SUN Attribute, CALTECH101, CALTECH256, ACTION40, EVENT8. En los primeros cuatro, centrados a imágenes, los resultados obtenidos por Places-CNN son mayores a los de ImageNet-CNN. En los segundos cuatro bases de datos de

	SUN397	MIT Indoor67	Scene15	SUN Attribute
Places-CNN feature	54.32±0.14	68.24	90.19±0.34	91.29
ImageNet-CNN feature	42.61±0.16	56.79	84.23±0.37	89.85
	Caltech101	Caltech256	Action40	Event8
Places-CNN feature	65.18±0.88	45.59±0.31	42.86±0.25	94.12±0.99
ImageNet-CNN feature	87.22±0.92	67.23±0.27	54.92±0.33	94.42±0.76

Figura 3: Métricas por conjunto de datos y red

imágenes, que son centradas en objetos, ImageNet-CNN obtiene mejores resultados.

Para finalizar, entrenaron una red híbrida combinando el set de datos de entrenamiento de Places-CNN y de ImageNet-CNN. La llamaron Hybrid-CNN y luego de remover categorías solapadas alcanzó los 3.5 millones de imágenes pertenecientes a 1183 etiquetas diferentes. Esta red logró pequeñas mejoras en algunos de los conjuntos de datos utilizados en la comparación entre Places-CNN e ImageNet-CNN. Los resultados se muestran en Fig. 4.

SUN397	MIT Indoor67	Scene15	SUN Attribute	Caltech101	Caltech256	Action40	Event8
53.86±0.21	70.80	91.59±0.48	91.56	84.79±0.66	65.06±0.25	55.28±0.64	94.22±0.78

Figura 4: Métricas por conjunto de datos - Hybrid-CNN

En [5] demostraron que no es necesario entrenar múltiples redes para realizar las tareas de clasificación de escenas y detección de objetos de una sola vez, ya que los detectores de objetos emergen por sí mismos en redes neuronales convolucionales entrenadas con bases de datos de escenas. Entender las representaciones aprendidas en las capas intermedias de arquitecturas profundas es un factor importante y del cual se podría sacar más provecho. Como las escenas están, en parte, compuestas por objetos, las redes neuronales convolucionales entrenadas para esta tarea aprenden a identificarlos internamente para definir de qué escena se trata, ergo la clasificación de escenas y la detección de objetos puede ser realizada en un mismo recorrido hacia adelante de la red, sin la necesidad de dar a la misma explícitamente la noción de objetos. La contribución más importante en

este trabajo fue demostrar que las redes entrenadas para detección de escenas, internamente aprenden a detectar los objetos relacionados a estas escenas; característica que hace a estas redes explotables para realizar otros propósitos sin la necesidad de tomarse todo el trabajo de crear, entrenar y refinar una red más con la que detectar los objetos que se contienen en estas escenas. En mayor medida, si la red fue entrenada con un conjunto de datos centrado en objetos. Para esta tarea los investigadores buscaron simplificar las imágenes de entrada para poder conocer cuáles eran las características de éstas que concentraban la mayor parte de la información que es utilizada por la red, es decir, aquellas en las que luego de ir quitando resto de características de la escena, la exactitud con la que se predecía se mantenía similar. En el primer intento de realizar esta tarea, para cada imagen, ellos realizaron una segmentación a partir de los bordes y regiones, removiendo subregiones diferentes de la siguiente manera: en cada iteración se remueve aquel segmento que produce el menor decrecimiento en la puntuación de clasificación, hasta que la escena sea clasificada incorrectamente. Al finalizar este primer enfoque obtuvieron una representación de la imagen que contiene la información mínima necesaria para que la red clasifique correctamente la escena. En un segundo intento basado en la hipótesis de que para la red Places-CNN existían objetos cruciales en el reconocimiento de escenas, generaron representaciones mínimas de las imágenes utilizando el conjunto de datos totalmente anotado SUN Database en cambio de realizar una segmentación automática. Para hacerlo realizaron el mismo procedimiento que en el primer enfoque para obtener estas representaciones con la diferencia que tomaron como verdaderos los segmentos provistos por la base de datos SUN. Vale denotar que para cada escena, son objetos los que usualmente forman parte de la representación mínima necesaria por la red, como es posible observar en la Fig. 5.

Para continuar con el trabajo, realizaron un análisis de los campos receptivos de las unidades y sus patrones de activación. Esto llevó a la observación de que las activaciones de las regiones tienden

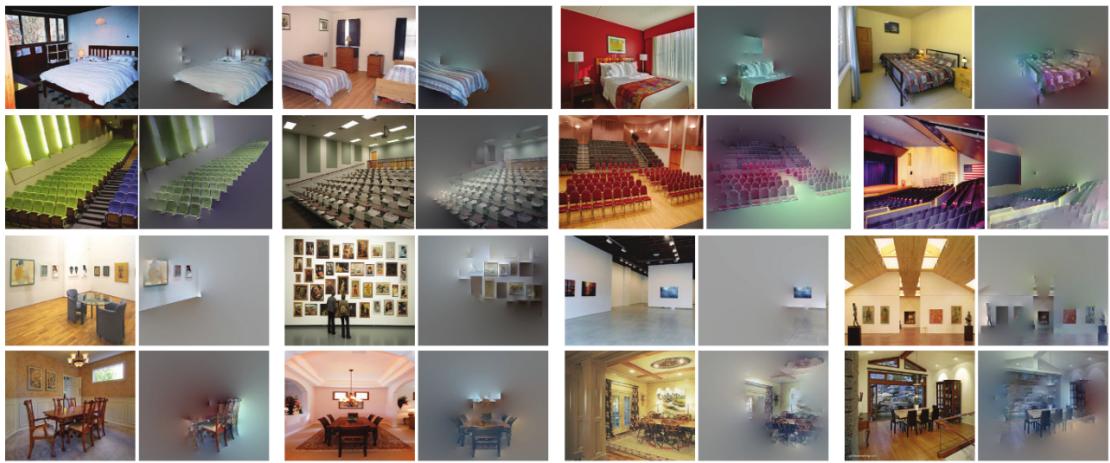


Figura 5: Ejemplos de representaciones mínimas encontradas

a tomar en mayor significado semántico a medida que se incrementa en la profundidad de las capas. Finalmente, los investigadores brindan algunas razones de porqué emergen estos objetos en las tareas de clasificación de escenas: por un lado, que los objetos que emergen son los que más se encontraron en la base de datos SUN, por otro que éstos objetos que emergen son los que permiten discriminar mejor entre diferentes escenas, como es posible de observar en la Fig. 6. Gracias a este trabajo es posible alcanzar el estado del arte en clasificación de escenas utilizando la red Places-CNN, pero también aprovechar las capas intermedias para encargarse de detectar los objetos que aparecen en estas escenas a partir de los campos receptivos de las mismas.

En [6] Herranz y otros, basándose en la idea de que el reconocimiento de escenas requiere comprender tanto sobre escenas como de objetos en la escena, se dedicaron a la tarea de resolver dos problemas relacionados; por un lado el sesgo inducido por conjunto de datos relacionados a objetos de una escala en redes neuronales convolucionales entrenadas con objetos en múltiples escalas y por el otro, cómo combinar eficientemente el conocimiento obtenido de conjuntos de datos centrados en escenas y centrados en objetos.

En este trabajo ellos se encargaron de tener en cuenta la escala

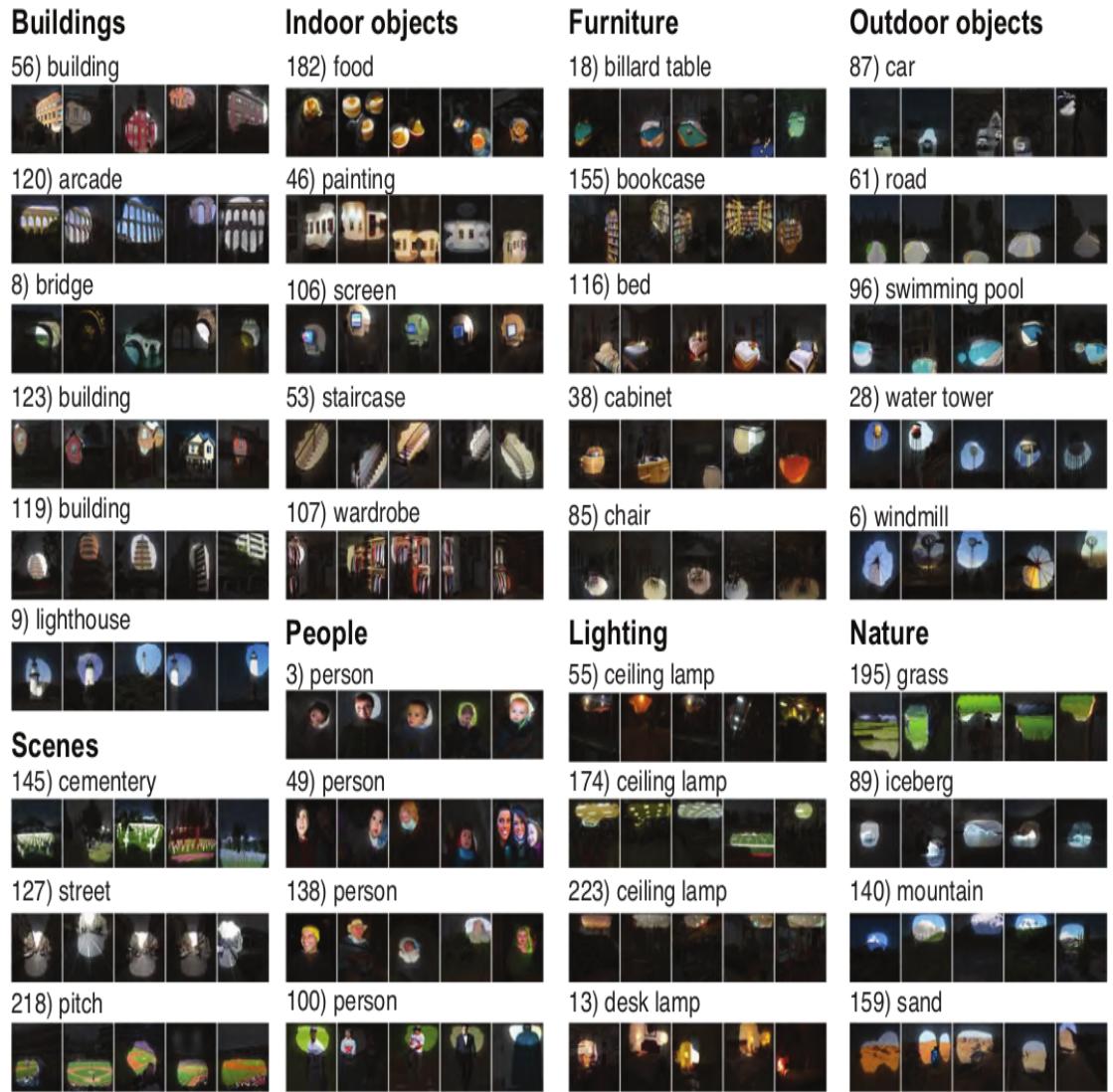


Figura 6: Ejemplos de objetos detectados a partir de la capa de pooling número 5

de los objetos para hacer que la red gane en razón de reconocimiento de las escenas. Para hacerlo eligieron centrarse en dos de los aspectos más importantes de las bases de datos de imágenes: escala y densidad. Por el lado de la escala, en el conjunto de datos ImageNet cada objeto ocupa casi el total de cada imagen, mientras que por el lado del conjunto de datos Sun397 los objetos son mucho más pequeños. Por

parte de la densidad, en el conjunto de datos centrado en objetos, cada imagen contiene un gran objeto, mientras que en el centrado en escenas cada imagen contiene muchos objetos pequeños.

Gracias a la segmentación de objetos dentro de las escenas que lograron, fueron capaces de crear variaciones de un mismo elemento. De esta manera definieron y generaron los mismos objetos en dos escalas diferentes: escala original (la escala del objeto en la escena original) y escala canónica (se centra el objeto en la imagen y es reescalado para ocupar el tamaño de la imagen, manteniendo aspecto de radio).

La arquitectura propuesta por estos investigadores está enfocada a atender múltiples escalas mediante redes de escalas específicas. Esta gran red es una combinación de varias redes que operan en paralelo sobre sectores extraídos de la versión original de la imagen. Para cada una de las escalas utilizaron una red con una variante de la arquitectura AlexNetCNN llamada Caffe-Net. Para la extracción de los sectores, con el fin de acelerar el procesamiento, utilizaron una red totalmente convolucional (Fully convolutional network), con una capa de agrupamiento mediante máximos para agregar las características de los sectores en características de la imagen en sí.

Ellos detallaron sus dos diferencias principales con la arquitectura híbrida base. Primeramente, ellos utilizan el modelo más adecuado para cada escala de las imágenes. Luego, se encargaron de refinar individualmente los parámetros de cada uno de los modelos generados para adaptarse de la mejor manera a la escala. Además, remarcan que el punto de principal inflexión con el resto de trabajos similares es que ellos le dan importancia a la escala de la imagen, usando varias redes neuronales convolucionales en cambio de sólo una, alegando que las diferencias de escalas de los objetos entre las bases de datos ImageNet y Places dan lugar a la principal limitante de rendimiento.

La idea final que otorgan es que la información local obtenida de la capa totalmente conectada número 7 de la red ImageNet-CNN más la información global obtenida de la capa totalmente conectada

número 7 de la red Places-CNN funciona mejor que la implementación híbrida base. Esto es así debido a que la red ImageNet-CNN aprende características sobre los objetos en sí (aprendizaje local), mientras que la red Places-CNN aprende características sobre las escenas completas (aprendizaje global).

En [7] Zhang y otros dieron a conocer una nueva estructura para realizar esta tarea, con el cual sobrepasaron el estado del arte en los conjuntos de datos utilizados. Se trata de Redes Neuronales Convolucionales Aleatorias Potenciadas por el Gradiente (Gradient Boosting Random Convolutional Neural Network, su nomenclatura en inglés), una forma de aprendizaje conjunto (ensemble) que combina varias redes neuronales profundas. Dentro de los aportes más significativos del trabajo se encuentran: la introducción de la red mencionada anteriormente en sí, una nueva función de pérdida multi-clase para poder combinar la potenciación por el gradiente con redes convolucionales y finalmente una variante a la red convolucional llamada red convolucional aleatoria (Random Convolutional Neural Network) que sirve como aprendiz base en tareas de aprendizaje conjunto profundo. La red está diseñada para generar un ensamblado de redes convolucionales aleatorias (RCNet) de manera de poder combinarlas usando una función de pérdida que se ajusta a la red base. La red propuesta como bien se mencionó anteriormente es un ensamblado, es decir, de múltiples redes intentando minimizar la función de costo y mapear datos de entrada con una salida a través de la estimación de una función que sea capaz de realizar este mapeo, en este caso, esta función estará formada por un conjunto de M funciones agregadas de forma aditiva.

$$\hat{f}(x) = \hat{f}^M(x) = \sum_{t=0}^M \hat{f}_t(x) \quad (5)$$

Los autores proponen tanto una nueva red base como una función de

pérdida para éstas. La función de costo está dada por:

$$\Psi(y, \hat{f}(x)) = - \sum_{k=1}^K y_k \log p_k(x) \quad (6)$$

donde la etiqueta a predecir está representada como 1 de los K vectores, siendo K igual al número de clases. $f(x)$ es la estimación general de la función de ensamblado y $p_k(x)$ es:

$$p_k(x) = \frac{\exp(f_k(x))}{\sum_{l=1}^K \exp(f_l(x))} \quad (7)$$

Basándose en la aleatoriedad de los Bosques Aleatorios, con el fin de evitar el sobreentrenamiento por compartir todas las características entre todas las redes del ensamblado, los autores propusieron una variante a las funciones CNet: RCNet, que aleatoriamente comparte algunos de los parámetros con otras de las redes del ensamblado mediante un banco de filtros.

Durante cada función RCNet se muestrea aleatoriamente un set de filtros del banco de filtros compartidos para construir la etapa de extracción de características de la RCNet y simultáneamente actualizar los parámetros de la RCNet y el banco de filtros compartido. El tamaño del filtro del banco de filtros compartidos es mayor al del filtro en la función RCNet de manera que diferentes redes compartirán algunos parámetros.

Para concluir, los investigadores pusieron a prueba la arquitectura (ver Fig. 7) con dos conjuntos de datos centrados en imágenes satelitales, comparando los resultados con arquitecturas clásicas de clasificación de imágenes y demostrando no sólo que la red propuesta es apropiada como aprendiz base en arquitecturas de ensamblado, sino que también es posible alcanzar el estado del arte y superarlos en relación a los métodos tradicionales.

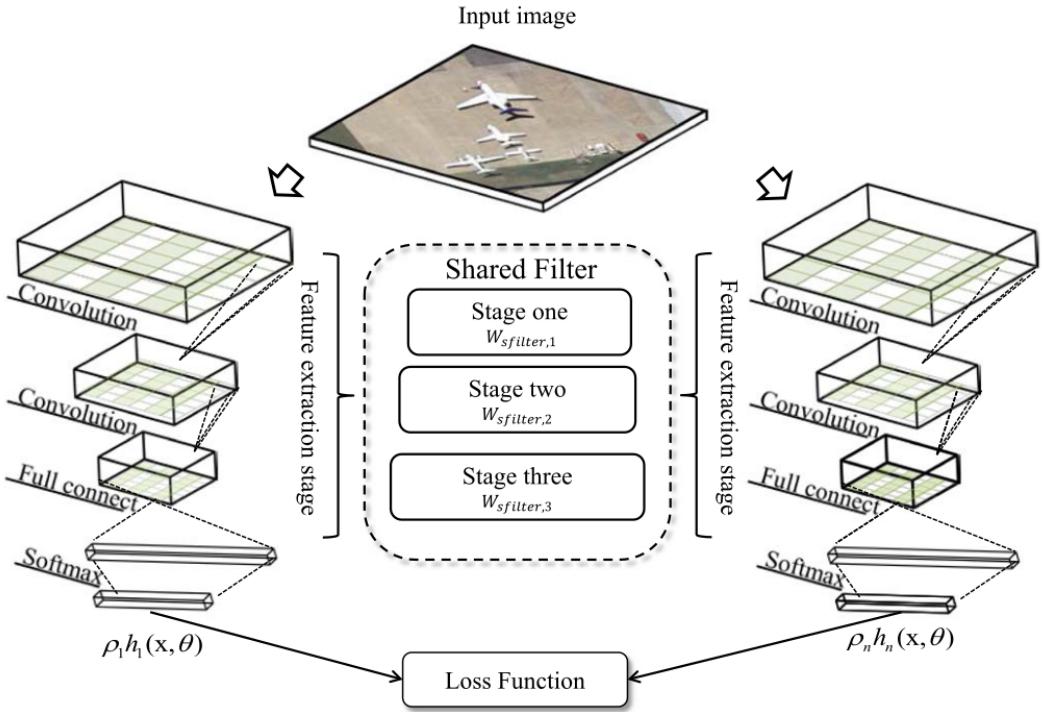


Figura 7: Arquitectura de la red

2.3. Clasificación de escenas mediante Aprendizaje por Transferencia

En [?], un trabajo realizado por Råhlén Oskar y Sjöqvist Sacharias en 2019, se trabajó específicamente con los fines de utilizar el aprendizaje mediante transferencia para clasificar imágenes de propiedades. Resumidamente, ya que luego se abordará debidamente el tópico en el marco teórico, el aprendizaje mediante transferencia se trata de utilizar una red preentrenada R con un conjunto de datos A para resolver un problema relacionado a un conjunto de datos B , y la manera para hacerlo es reentrenar la red R con los datos del conjunto de datos B . Para este proyecto utilizaron como redes preentrenadas ResNet18, AlexNet, VGG-11, DenseNet-121 e Inception-v3. Los datos con los que cuentan en esta investigación son imágenes extraídas de Google Image search y [8], distribuidos de la siguiente manera

para cada etiqueta como se observa en la Fig. 8.

Category	Training	Validation	Total
balcony	558	139	697
Indoor	486	122	608
Fireplace	382	96	478
no_fireplace	205	51	256
Kitchen	256	64	320
bathroom	208	52	260
bedroom	291	73	364

Figura 8: Distribución de las imágenes por categoría etiquetada

En el trabajo se realizaron tres experimentos, y en cada uno de ellos se testearon todas las redes descriptas previamente, con y sin refinamiento de las mismas. El primero se trata de un clasificador binario para predecir si una imagen contiene o no un balcón; el segundo es igualmente un clasificador binario pero que predice si en la imagen se encuentra un hogar (por ejemplo, hogar a leña), y en el tercer experimento plantean una clasificación multiclase en la que intentan etiquetar las imágenes según si se trata de una cocina, un dormitorio o un baño. Es sobre éste último en el cual se hará énfasis. Los primeros dos experimentos con clasificadores binarios no resultan de gran interés para este problema, aunque es importante remarcar que para el primero se alcanza una exactitud de un 98 % luego de refinamientos utilizando la red Inception-v3, mientras que para el segundo el mayor porcentaje se alcanza con la red DenseNet y alcanza un 85.5 %. El experimento realizado que resulta de mayor importancia para este proyecto es el tercero, un clasificador de múltiples etiquetas que intenta determinar si la imagen se trata de una cocina, un dormitorio o un baño. Las métricas obtenidas antes de realizar refinamiento de las redes testeadas se dan a conocer en la Fig. 9 Luego de hacer refinamiento de las redes entrenando con las imágenes

Model	Time	Max. accuracy
RESNET	02m 33s	93.75
Alexnet	02m 18s	93.22
VGG-11	04m 31s	93.75
Densenet	04m 20s	96.87
inception V3	06m 28s	93.75

Figura 9: Métricas usando extracción de características de los modelos preentrenados

del conjunto de datos generado por los investigadores se alcanzan los resultados expuestos en la Fig 10.

Model	Time	Max. accuracy
RESNET	04m 16s	96.35
Alexnet	02m 34s	94.79
VGG-11	11m 02s	97.91
Densenet	09m 53s	97.91
Inception V3	16m 10s	97.39

Figura 10: Métricas luego de realizar refinamiento de las redes

Como se puede observar, luego de reentrenar los modelos se obtienen mejoras representativas, dado el percentil de exactitud en que se encuentran los resultados. Otro punto a denotar es que en ambos casos es la red DenseNet la que obtiene la mejor performance, aunque toma aproximadamente el doble de tiempo que el resto de las redes en entrenarse. Los autores concluyen su trabajo explicando que a partir de un bajo número de imágenes en su conjunto de datos, y a través de aprendizaje por transferencia, es posible agregar palabras claves a las imágenes, como ser: balcón, hogar, baño, cocina y habitación.

2.4. Otros trabajos relacionados a las propiedades inmuebles que toman provecho de las imágenes de los mismos

En [2] Poursaeed y otros se dedican a la tarea de la estimación de precios de inmuebles basándose en las características visuales de las propiedades. El trabajo incluye una evaluación del impacto visual de las características de una casa en su valor de mercado, la estimación de lujosidad mediante redes neuronales convolucionales, un armazón para la automatización de la valuación utilizando tanto imágenes de las propiedades como metadatos de las mismas y experimentos en los que aplican su trabajo a un nuevo conjunto de datos. Para comenzar se encargaron de obtener alrededor de doscientas mil imágenes correspondientes a diferentes ambientes de casas a partir del conjunto de datos Places, Houzz (una empresa de alquiler y venta de viviendas) y búsquedas en Google Imágenes. Luego, entrenaron una red con arquitectura DenseNet para la tarea de predecir las etiquetas baño, dormitorio, cocina, living, comedor, interior y exterior. Con este clasificador, alcanzaron un 91 % de exactitud en el conjunto de validación.

A partir de las imágenes etiquetadas, en esta investigación propusieron segmentar cada habitación en ocho niveles de lujosidad utilizando la herramienta de crowdsourcing Amazon Mechanical Turk con el fin de obtener estas etiquetas de cada sector. De esta manera, se hicieron de un embedding (descripto en ??) de baja dimensión en el cual las imágenes con el mismo nivel de lujosidad se encuentran cercanas entre sí, como se puede ver en la figura ??, extraída de la figura 3.c de [2]. Mediante el algoritmo t-STE los investigadores obtuvieron un embedding bidimensional de las imágenes, que a partir de visualizaciones de los clusters se determinó que las imágenes con mayor nivel de lujosidad quedan en el centro, mientras que las menos lujosas se ubican alrededor. Para aquellas casas que no presentaban imagen de alguno de los ambientes, lo que hicieron fue imputar el promedio de las otras categorías para representar el nivel

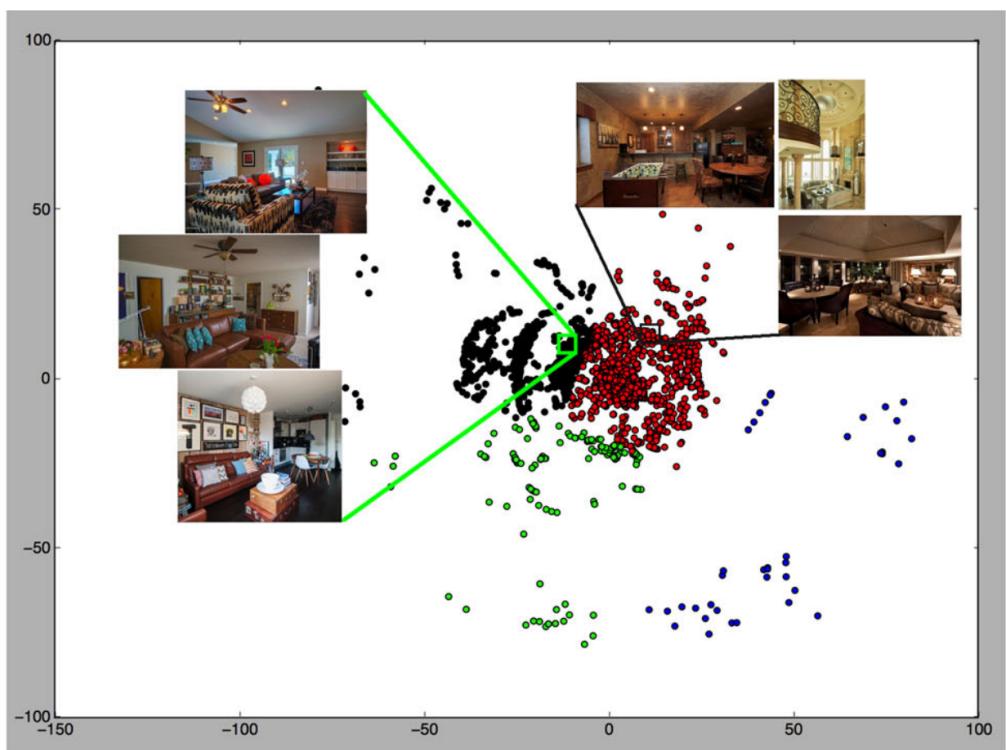


Figura 11: Figura 3.c de Vision Based Real Estate Price Estimation

de lujosidad de ese ambiente. Una actividad aparte para ellos fue la estimación de precios, para la cual implementaron una regresión que absorve tanto la salida de los modelos que estiman la lujosidad de las habitaciones previamente clasificadas por la DenseNet, como metadatos al respecto de la propiedad (precio de oferta, tamaño, años desde su construcción, cantidad de habitaciones de cada tipo, etc). Vale aclarar que la etiqueta a predecir en cada propiedad es el valor final al que fue adquirida.

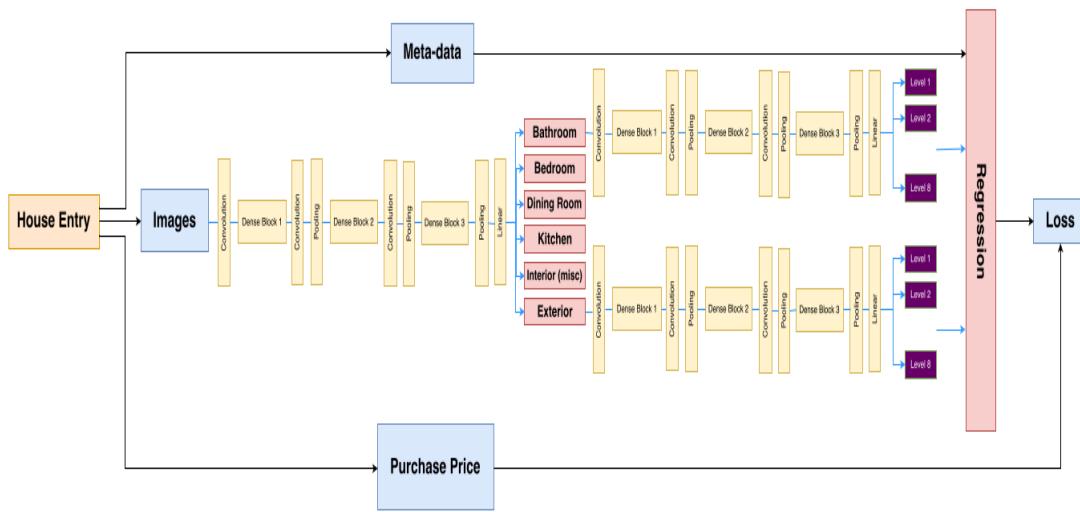


Figura 12: Arquitectura de la red descriptiva

Finalmente, con la arquitectura utilizada (ver Fig. 12) demostraron que es posible mejorar los resultados de estimaciones que actualmente se utilizan en el mercado (Índice Zestimate de Zillow) disminuyendo la mediana del error de un 7.9 % a un 5.6 % logrado a partir de la arquitectura presentada.

2.5. Antecedentes no académicos

Dentro de los antecedentes que no califican como investigaciones, nos encontramos con *RestbAI* [9], una empresa que comenzó en 2016 de la mano de Angel Esteban. ”Buscando entre docenas de propie-

dades, él estaba shockeado por la inconsistencia en la calidad de las mismas y la dificultad general para encontrar la casa de ensueños para su familia. Tenía que haber una mejor manera..^{es} lo que muestran en la historia de la empresa, que actualmente cuenta con oficinas en Estados Unidos y Europa. Dentro de las capacidades de visión por computadora que listan relacionadas a las propiedades se encuentran la clasificación de escenas, la detección de características dentro de una imagen, análisis de estado de las habitaciones. Además, gracias a las capacidades mencionadas anteriormente en su sitio, demuestran cómo se aprovecharon del etiquetado de las imágenes para generar sus productos. Éstos son avocados a la experiencia de usuario, modelos de datos y moderación de contenido. En relación a la experiencia de usuario detallan tres productos fundamentales: la experiencia en búsquedas, el análisis comparativo del mercado y la conversión de imagen a discurso. Por el lado de los modelos de datos se encuentran la valoración automática de propiedades, completar información al respecto de una propiedad que pueda no haber sido detallada, es decir, la integridad de datos, las publicidades dirigidas y la analítica de datos relacionados a este contexto. Por parte de la moderación de contenidos se encuentra la detección de marcas de agua, detección de imágenes duplicadas y la detección de información sensible dentro de la imagen (patentes, números telefónicos, rostros). Como se puede ver, algunos de estos productos resultan ya definidos como posibles formas de aprovechar el etiquetado de imágenes en la motivación del proyecto, aunque otros no fueron mencionados y también continúan agregando valor y razón de ser a esta investigación.

3. Marco teórico

3.1. Aprendizaje automático

Dentro de las diferentes ramas de la inteligencia artificial, los algoritmos utilizados se pueden clasificar en tres grandes grupos según la forma en que son entrenados: algoritmos de aprendizaje supervisado, algoritmos de aprendizaje semi-supervisado y algoritmos de aprendizaje no supervisado. El trabajo [10] detalla la definición de cada uno de estos conceptos.

Los primeros se utilizan cuando para un conjunto de entrenamiento X se tiene un conjunto de etiquetas y relacionadas a cada elemento; de esta manera, los algoritmos encuadrados dentro de esta rama se encargan de estimar una función que mapee los datos del conjunto X al conjunto y , con el fin de luego poder realizar inferencias a partir de sólo nuevos casos de entrada.

Los algoritmos de aprendizaje no supervisado se utilizan en situaciones en las que se tiene un conjunto de datos de entrada X pero no un conjunto de etiquetas asignadas para los mismos; estos algoritmos se emplean para realizar sobre los datos tareas tales como agrupamientos, detección de anomalías, reducción de dimensionalidad, entre otras.

Por último, el aprendizaje semi-supervisado entra en juego en aquellos casos en los que se tiene un pequeño conjunto de datos etiquetado, y un conjunto de datos no etiquetados. Existen modelos en los que agregar datos no etiquetados al entrenamiento hace que mejoren sus predicciones, como otros en los que hace que empeoren. Las ideas principales detrás del aprendizaje semi-supervisado son: ayudar a ganar representación general de la población de los datos, y ganar conocimiento sin recurrir a los costos de tiempo y dinero que suponen etiquetar datos que se tienen.

En este trabajo se contará tanto con un conjunto de entrenamiento (que serán bases de datos de escenas de propiedades) como con sus pertenecientes etiquetas, por lo que se intentará mapear mediante técnicas de aprendizaje profundo cada imagen con sus etiquetas,

es decir, se trabajará en un problema de aprendizaje supervisado. Dentro de esta rama existen otros algoritmos como los expuestos en [11] que pueden servir para este problema o para otros de dominio diferente, algunos de estos son: árboles de decisión, regresiones lineales o logísticas, ensembles, etc.

3.2. Representación de una neurona

En este trabajo, como bien se explicó anteriormente, se abordará la solución mediante técnicas de aprendizaje profundo, por lo que a continuación se dará lugar a las explicaciones pertinentes relacionadas a este tipo de técnica. Para empezar es necesario comentar que el término "profundorecae en la profundidad de las redes neuronales que se utilizan en esta técnica. Estas redes pueden estar compuestas hasta por millones de neuronas ([12]) interconectadas entre si; neuronas que se representan como se observa en la figura 13 donde se muestra un ejemplo con n entradas, conformada por:

- Entradas: conjunto de datos de entrada $x_1..x_n$
- Pesos: conjunto de pesos $w_1..w_n$ correspondientes a cada entrada
- Función de agregación: función que sumariza la multiplicación pesada entre cada entrada x y su correspondiente peso w .
- Función de activación: función no lineal responsable de mapear el resultado de la función de agregación en salidas (según que tipo de función sea, los resultados suelen variar entre $[0, 1]$ o $[-1, 1]$).
- Salida: resultado de la función de activación.

De esta manera, se puede obtener la salida de la neurona Y , a partir de aplicar la función de activación σ a la suma entre el sesgo b y la multiplicación matricial del conjunto de entradas X y los pesos W .

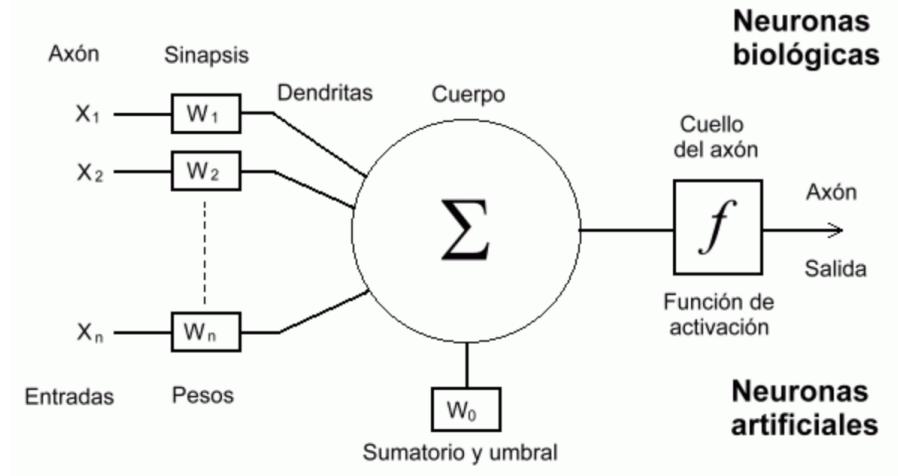


Figura 13: Representación de una neurona [13]

$$Y = \sigma(W^T X + b) \quad (8)$$

3.3. Red Neuronal

Una red neuronal, como su nombre lo enuncia, es una consecución de capas de N neuronas en cada una, conectadas entre sí como se puede ver en el ejemplo de la figura 14. Es necesario que se cuente con capas de entrada, ocultas y de salida; para el ejemplo enunciado se cuenta con una capa de entrada, dos capas ocultas y una capa de salida.

El fin de las redes neuronales es aprender representaciones del contenido de la información en relación a las salidas esperadas para luego poder hacer inferencias en nuevo contenido, a modo de ejemplo, si se tienen imágenes de perros y gatos, y el fin es detectar si se trata de un perro o un gato la red posiblemente no aprenda las mismas representaciones que si el fin de la misma es detectar presencia o ausencia de animales.

Para explicar el funcionamiento de las redes neuronales es necesario también introducir algunos conceptos como son la propagación hacia adelante y hacia atrás, función de pérdida y optimización.

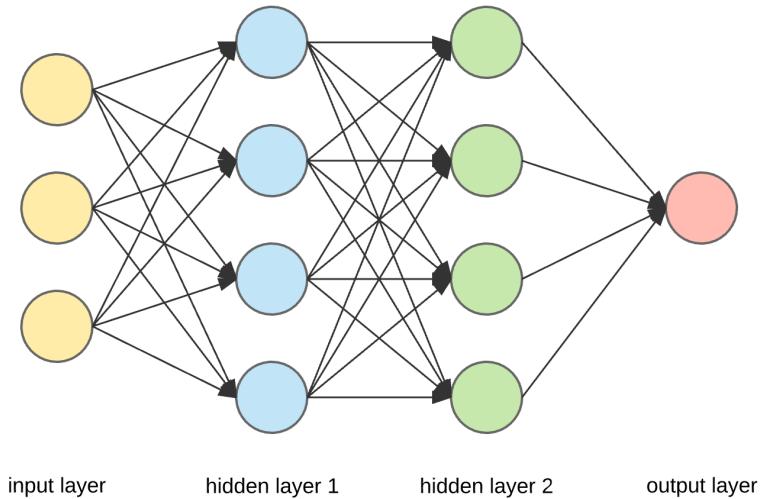


Figura 14: Perceptrón Multi Capa

Durante la fase de entrenamiento, los pasos que se siguen son: (i) inicialización de pesos y sesgo, (ii) propagación hacia adelante, (iii) evaluación de función de pérdida y (iv) propagación hacia atrás.

Antes de comenzar a entrenar una red es necesario inicializar los pesos que conectan las capas, suele hacerse de manera aleatoria a partir de una distribución normal, aunque existen otros métodos como hacerlo a partir de una distribución uniforme, la inicialización de Xavier [14], entre otras.

La propagación hacia adelante [13] es la actividad en que se le brindan los datos a la capa de entrada de la red para que se evalúen consecutivamente todas las capas siguientes en esa dirección aplicando f en cada una, tomando como entrada la capa inmediata anterior. De esta manera, se obtiene como salida de la red sus predicciones.

$$z = W^T X + b \quad (9)$$

Luego, a partir de las salidas \hat{y} de la red es posible medir la función de pérdida [13], [15] (detallado en 10) que brinda la información de cuán cerca estuvo la red de predecir las salidas correctas y con la que se puede computar la propagación hacia atrás del error, es decir, optimizar la función de pérdida actualizando los pesos desde atrás

hacia adelante teniendo en cuenta el error en cada capa. La manera de hacerlo es mediante la derivada parcial de la función de costo con respecto tanto a los pesos ω (fórmula 11) como a los sesgos b (fórmula 12) en cada capa con el fin de obtener un nuevo valor de los pesos y sesgos W y b en cada una de éstas.

$$\mathcal{L}(y, \hat{y}) = - \sum_{i=1}^n y_i \log \hat{y}_i \quad (10)$$

$$\omega = \omega - \alpha \frac{\partial J(\omega, b)}{d\omega} \quad (11)$$

$$b = b - \alpha \frac{dJ(\omega, b)}{db} \quad (12)$$

La función de pérdida se define de diferentes maneras según el problema que se esté tratando, ya sea error cuadrático medio, error absoluto medio, proximidad del coseno, etc. La propuesta general para problemas de clasificación es usar *entropía–cruzada* (o *cross–entropy* en inglés), una función derivable que mide qué tan lejos del resultado esperado están las salidas de la red, como se observa en la ecuación 10. Como se mencionó anteriormente, el objetivo de la red es minimizar la función de pérdida y para esto se requiere de una función de optimización. *Adam*, introducido en [16] es un optimizador en que se puede tener ciertas certezas de su correcto funcionamiento ya que su implementación tiene en cuenta las principales mejoras de otros como son el Descenso por el Gradiente con Momentum y la Propagación de la Raíz Cuadrada de la Media de los Cuadrados, también expuestos en [17].

En este trabajo se utilizará también una función de pérdida personalizada, que llamaremos de ahora en más Entropía Cruzada Categórica pesada por clase. Esta función es una variante que multiplica el resultado del valor de la función en su estado puro por la inversa de la distribución de probabilidad de las clases en el conjunto de datos. Esto hace que se le de mayor importancia a aquellas clases que en

el conjunto de entrenamiento tienen menor cantidad de imágenes y menor importancia a las que tienen mayor cantidad.

$$\mathcal{L}(y, \hat{y}) = - \sum_{i=1}^n y_i \log \hat{y}_i \quad (13)$$

Un problema común al entrenar una red es el sobreentrenamiento, que se da cuando la red sobreajusta sus pesos para el conjunto de entrenamiento y pierde capacidad de generalizar. Una forma fácil de controlarlo es mediante la verificación de la función de pérdida y las métricas elegidas sobre un subconjunto de validación, que son imágenes que no se utilizan para entrenar la red sino que para medir cómo está funcionando la misma (si la red está sobreentrenando, entonces la diferencia entre las métricas de entrenamiento y validación comenzarán a incrementarse, siendo las de entrenamiento mayores a las otras). Para contrarrestar estas situaciones existen capas regularizadoras, es decir, que evitan la concentración de gran porcentaje del conocimiento en un subconjunto de unidades en las capas y por ende el aprendizaje de la representación del contenido sea compartido entre las neuronas de las capas. Aunque existen otras, las más utilizadas son *dropout* [18] y *batch – normalization* [19]. La capa *dropout* se encarga de apagar conexiones aleatoriamente entre capas durante el entrenamiento, esto hace que no se entrene siempre con las mismas unidades dentro de cada capa, y así, que no sean siempre las mismas neuronas las que se activen frente a cada situación. La capa de *batch – normalization* aplica una función de regularización que sale de la media y la varianza de cada iteración durante el entrenamiento, agregando ruido en las activaciones y evitando que las mismas no sean demasiado altas ni bajas, reduciendo de esta manera el desplazamiento covariante interno (*internal covariance shift*, en inglés), y la manera de hacerlo es mediante la aplicación de las ecuaciones 14, 15, 16, 17, como se explica en el Algoritmo 1 de [19].

$$\mu_B = \frac{1}{m} \sum_{i=1}^m x_i \quad //media del mini – batch \quad (14)$$

$$\sigma_{\mathcal{B}}^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // varianza del mini-batch \quad (15)$$

$$\hat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // normalizacion \quad (16)$$

$$y_i = \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // escalacion y desplazamiento \quad (17)$$

Las neuronas de las capas convolutivas y de las capas totalmente conectadas requieren de una función de activación para determinar su salida, y aunque existe gran variedad de estas, hay algunas funciones que se utilizan mucho más que otras tanto en el campo de investigación como en la práctica, como se puede ver en el análisis en detalle realizado en [20]. A continuación se detallarán algunas de las funciones de activación que posiblemente se utilicen en las etapas de experimentación:

- Sigmoide (*Sigmoid*): Está definida como 18 y la razón principal por la que se la utiliza es debido a que sólo devuelve valores entre [0, 1], por lo que es muy útil en tareas en las que se necesita predecir la probabilidad de una clase.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (18)$$

- Tangente hiperbólica (*TanH*): Su formulación se explica en 19. Como se puede observar en la figura 16, mantiene la forma S de la función Sigmoide pero en este caso los resultados posibles van desde [-1, 1].

$$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \quad (19)$$

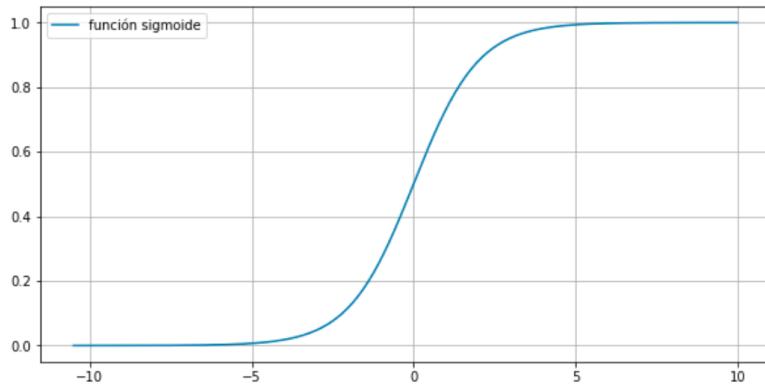


Figura 15: Función Sigmoide

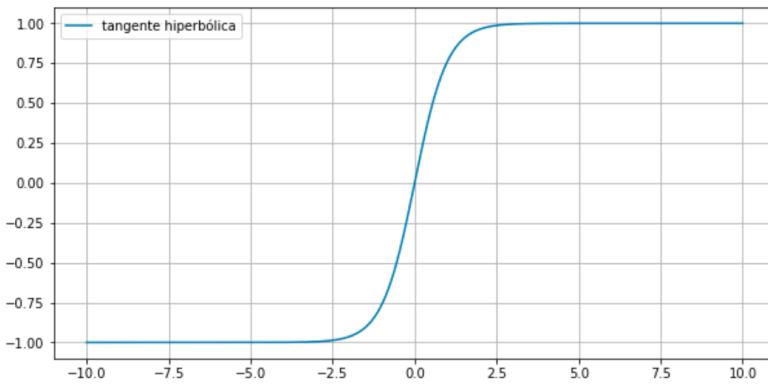


Figura 16: Función Tangente Hiperbólica

- Unidad lineal rectificada (*ReLU*): En la figura 17 se puede observar el comportamiento de esta función (definida en la fórmula 20), en donde para valores negativos el resultado es cero, mientras que para el resto, el resultado es el mismo valor. Por este comportamiento es una de las funciones de activación más utilizadas (junto a sus otras variantes eLu, LeakyReLu, etc) en las capas ocultas de las redes neuronales.

$$ReLU(x) = \max(0, x) \quad (20)$$

- Exponencial normalizada (*Softmax*): es una variante de la función Sigmoide pero definida para problemas de clasificación

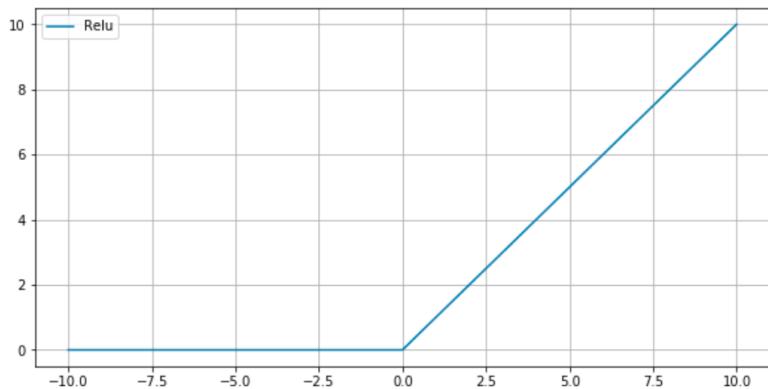


Figura 17: Función ReLu

multiclas; como se observa en la fórmula 21, normaliza las entradas en una salida que es en un vector de K valores de probabilidad (uno por cada clase definida) entre $[0, 1]$ que sumados dan uno.

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \quad \text{por cada } j = 1, \dots, K \quad (21)$$

Siendo K la cantidad de clases.

3.4. Red Neuronal Convolucional

Las redes neuronales convolucionales son una variante de las redes neuronales en las que la extracción de características se realiza mediante capas convolucionales. Fueron introducidas en [21] por LeCun y Bengio, investigadores referentes en el mundo del aprendizaje automático y las redes neuronales, y han transformado la forma de resolver problemas de clasificación de imágenes (y otros) desde entonces.

Como es posible observar en la figura 18, una convolución es una operación matemática. Se puede entender como el resultado de aplicar un filtro (una matriz con forma $N \times N$) en todas las regiones de la imagen con el fin de obtener una nueva.

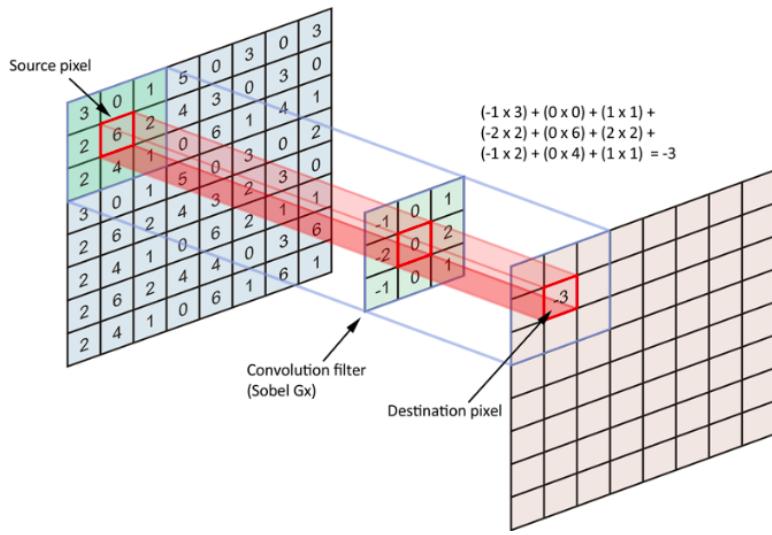


Figura 18: Convolución

$$\begin{aligned}
 f = & (-1 * 3) + (0 * 0) + (1 * 1) \\
 & + (-2 * 2) + (0 * -6) + (2 * 2) \\
 & + (-1 * 2) + (0 * 4) + (1 * 1) = -3
 \end{aligned} \tag{22}$$

En el ejemplo se aprecia que el resultado de la aplicación del filtro en la esquina superior izquierda de la imagen se obtiene a partir de multiplicar cada valor del pixel en la imagen de entrada con su par en la misma posición en el filtro, finalmente en el ejemplo se tiene que el valor del nuevo pixel en la imagen de salida se obtiene como se observa en la formulación 22. La operación detallada previamente se aplica a toda la imagen para obtener la imagen resultado, teniendo en cuenta tanto los valores del *stride* como del *padding* definidos para la capa.

El parámetro *stride* define cada cuántos píxeles de la imagen de entrada se aplicará el filtro, mientras que mediante el parámetro *padding* se decide cómo se tratarán los bordes, esto es debido a que existen ocasiones en las que se desea obtener una imagen en la que se le apliquen convoluciones también a los bordes de la misma; las formas más comunes de hacerlo son agregando a los bordes N píxeles

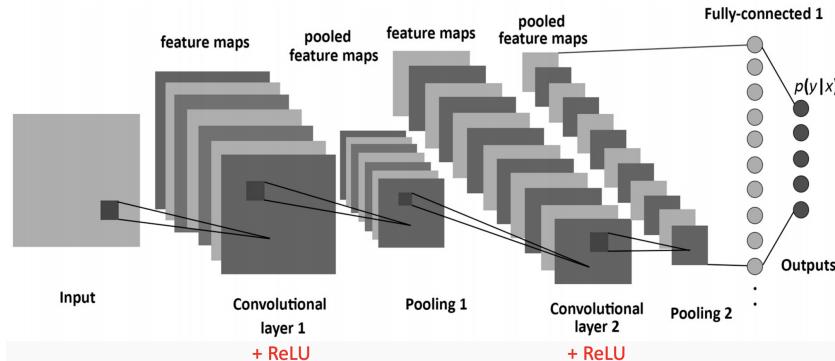


Figura 19: Ejemplo Red Neuronal Convolucionarial

más (dependiendo del *stride*), ya sea imputándolos como ceros o con mismo valor de sus píxeles más cercanos.

Gracias a estas capas las redes convolucionales son capaces de aprender representaciones del contenido invariantes, ya que cada filtro se aplica a todas las subregiones de la imagen haciendo que el aprendizaje obtenido en las capas más profundas se base en las imágenes completas. A continuación se detallará cómo funciona una red convolucionarial, con el fin de mostrar cómo estas las capas convolutivas se interconectan entre sí, reducen las representaciones computadas y finalmente se conectan con capas totalmente conectadas para realizar inferencia a partir de lo aprendido.

Existen muchas configuraciones de redes neuronales convolucionales, aunque para simplificar la explicación se utilizará una red compuesta de la siguiente manera: capa convolucionarial, capa de pooling, capa convolucionarial, capa de pooling, capa totalmente conectada y capa de salida (ilustrada en la figura 19). La capa de entrada en este caso (al igual que en el presente trabajo) serán imágenes en formato RGB normalizadas, es decir, una matriz de [*alto x ancho x n. canales*] por cada imagen; luego esta entrada será convolucionada aplicando la operación matemática previamente explicada y sus salidas serán agrupadas por una capa llamada *pooling*. En esta capa se buscará reducir tanto las representaciones aprendidas como la cantidad de cálculos a realizar: se agruparán mediante la aplicación de un filtro f y un método M los resultados de la convolución, siendo f una matriz

(usualmente de $[2 \times 2]$) y M la utilización de la media o el máximo de los valores de la imagen en cada sector en que se aplique el filtro; al igual que en la convolución, la operación de pooling se aplica en toda la imagen, aunque tiene dos diferencias fundamentales: no tiene parámetros que aprender o ajustar y busca reducir las representaciones a matrices de menor orden. Continuando con la red, se aplica nuevamente una convolución seguida de una capa de pooling que se conecta a una capa totalmente conectada (la misma que se utiliza en el perceptrón multicapa). Finalmente, la capa de totalmente conectada se conecta con una capa de salida que se encargará de computar las probabilidades con las que se predice cada clase (básicamente, otra capa totalmente conectada con función de activación Softmax, mencionada en 3.3).

3.5. Redes preentrenadas y Aprendizaje por transferencia

Como entrenar redes neuronales profundas resulta muy costoso computacionalmente tanto por la profundidad de las mismas como por la cantidad de datos con las que se las necesita entrenar, con el pasar del tiempo quienes tienen acceso a capacidad de cómputo comenzaron a liberar redes entrenadas con diferentes bases de datos. Posiblemente, las redes liberadas más conocidas sean las que se entrenaron utilizando ImageNet [22], una base de datos de 3.2 millones imágenes centradas en objetos, aunque también existen otras redes preentrenadas con otras bases de datos de imágenes abiertas como son COCO Dataset [23], Places Dataset [4], etc.

En la actualidad existen redes con diferentes arquitecturas preentrenadas utilizando diversos conjuntos de datos como son VGGNet [24] (figura 20), InceptionNet [25] (figura 21), DenseNet [26], ResNet [27], Bert [28], GPT2 [29]. Dependiendo el tipo de problema y la base de datos con la que fue entrenada cada red, es posible tomar provecho de las mismas mediante un método llamado *Aprendizaje por transferencia* (o *Transfer Learning* en inglés).

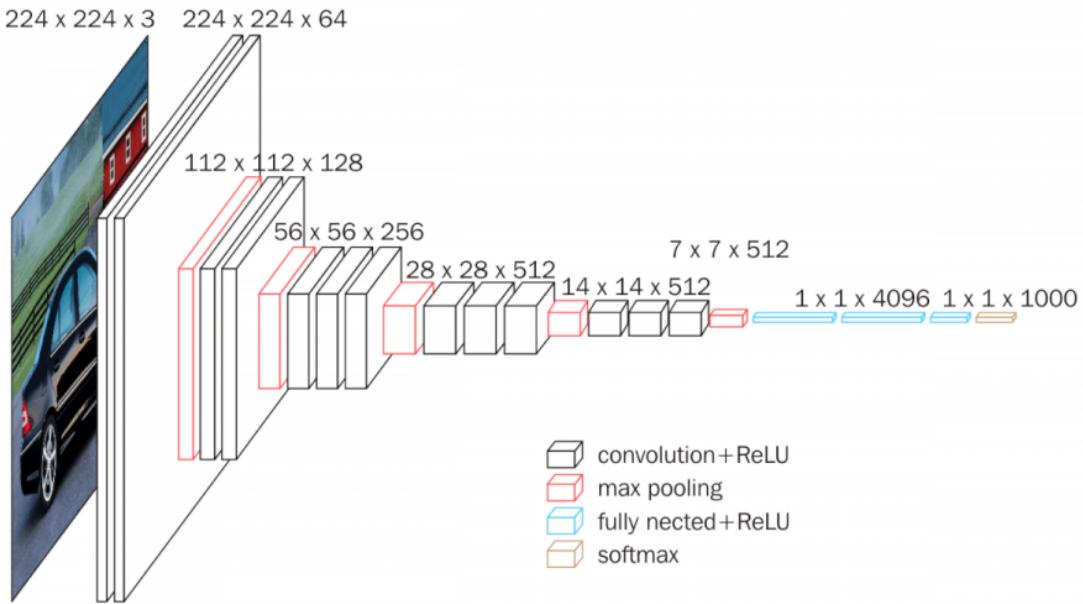


Figura 20: Arquitectura VGG16

La forma más simple de utilizar estas redes para hacer transferencia de aprendizaje es haciendo inferencia directamente a partir de ellas, es decir, luego de descargarlas, brindarles datos de entrada y realizar una pasada hacia adelante con el fin de obtener las salidas, aunque la única situación para hacerlo es si la red fue preentrenada con el mismo objetivo que el problema en cuestión (no se puede cambiar de un problema de clasificación binaria a uno multiclase directamente). Otra opción es reentrenar a la red con un conjunto de datos propio, de manera de extraer las características aprendidas por la misma y poder refinar sus pesos con un conjunto de entrenamiento que se entiende será más similar a lo que luego se utilizará para realizar predicciones. Esto permite tanto agregar capas extra sin entrenar a la red como cambiar las predicciones esperadas utilizando una capa de salida diferente a la inicial.

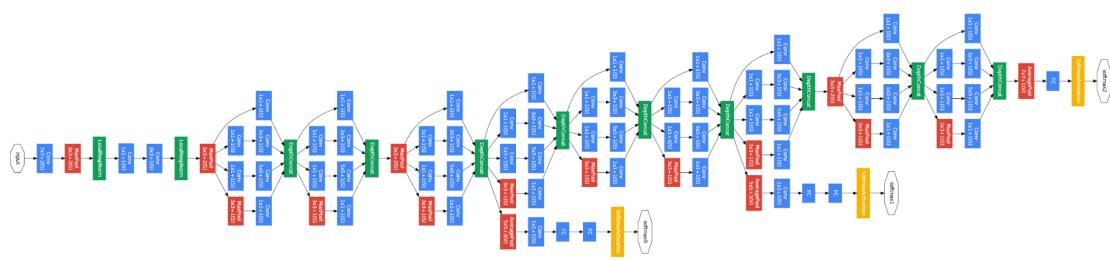


Figura 21: Arquitectura InceptionNet

4. Diseño experimental

4.1. Asunciones

En este trabajo se tomarán como ciertos los siguientes puntos:

1. Los conjuntos de datos elegidos se encuentran etiquetados correctamente
2. Cada imagen de los conjuntos de datos con los que se entrenarán los modelos resultan representativas a la escena a la que pertenecen.
3. El hardware con el que se llevará a cabo el proyecto funcionará tal y como se establece en sus respectivo manual.
4. Las redes preentrenadas a descargar, PlacesCNN e Imagenet-CNN, fueron entrenadas sólo con las imágenes de los conjuntos de datos Places365 e ImageNet, respectivamente.
5. Para ambientes productivos la métrica exactitud categórica [30] es la más representativa.

4.2. Limitaciones

El proyecto en curso contará con las siguientes limitaciones:

1. El análisis se centrará en imágenes de las siguientes escenas de propiedades: cocina, comedor, baño, dormitorio, exterior y living.
2. Tanto los tiempos de entrenamiento y predicción como el tamaño de las redes, estarán restringidos al hardware con el que se cuenta.
3. El trabajo intentará aceptar o refutar las hipótesis enumeradas a continuación, quedando excluidas del alcance del proyecto las posibles investigaciones que surjan a partir del mismo.

4.3. Hardware a utilizar

Para llevar adelante el trabajo se utilizarán los siguientes componentes:

- Procesador Intel i7 de séptima generación versión U.
- Memoria RAM de 16gb.
- Disco sólido 256GB NVMeTM M.2 SSD.
- Tarjeta gráfica Gigabyte GTX 1080 de 8gb.
- eGPU HP Omen Accelerator.

4.4. Hipótesis

Teniendo en cuenta tanto las limitaciones como las asunciones definidas para el trabajo, se comprobarán las hipótesis declaradas a continuación.

4.4.1. Hipótesis 1

Como se pudo observar en la revisión de antecedentes, existen múltiples formas de hacer frente al problema. El enfoque más simple podría ser mediante perceptrón multicapa, pero por lo que se pudo observar, la mayoría de las soluciones utilizadas son redes neuronales convolucionales.

Hipótesis: “Una red convolucional es capaz de obtener mejores resultados que un perceptrón multicapa en materia de clasificación de escenas”.

4.4.2. Hipótesis 2

Las redes convolucionales tienen la capacidad de aprender características de las imágenes con las que se entrenan, aunque a veces resulta muy costoso hacerlo por las diferencias entre imágenes de la

misma categoría o bien no se cuenta con la cantidad de imágenes que aporten la densidad y diversidad necesaria para alcanzar el tope máximo en la métrica elegida.

Hipótesis: Una red convolucional obtendrá mejores resultados sobre un conjunto de test *y* siendo entrenada con conjunto de entrenamiento *X* que si es entrenada sólo con subconjuntos de 10% o 50% del conjunto de entrenamiento *X*, respectivamente”.

4.4.3. Hipótesis 3

Como se pudo ver en [4], Zhou y otros crearon una red entrenada con millones de imágenes de escenas (Places Dataset) que debería ser capaz de predecir las imágenes de las escenas con las que fue entrenada. En términos de cantidad de imágenes, esta red está mucho más entrenada que las utilizadas en este trabajo.

Hipótesis: Realizar inferencia directamente a partir de esta red preentrenada alcanza mejores resultados que una red convolucional entrenada con el conjunto de datos del trabajo [2] (el de mayor cantidad de imágenes que se tiene) para los conjuntos de validación y verificación definidos al contrastar la hipótesis 4.4.2”.

4.4.4. Hipótesis 4

Una de las técnicas revisadas antes de comenzar con el trabajo fue Aprendizaje por Transferencia (2.3), en el cual se parte de redes preentrenadas y se las reentrena con el conjunto de imágenes propio para ajustar los pesos de la red al mismo.

Hipótesis: ”Haciendo Aprendizaje por Transferencia a partir de la red PlacesCNN es posible mejorar los resultados obtenidos al contrastar la hipótesis 4.4.2”.

4.4.5. Hipótesis 5

Otra red pre-entrenada existente es ImageNetCNN, la cual fue entrenada con el dataset ImageNet centrado en objetos. Aunque no

fue entrenada para escenas como PlacesCNN, se puede interpretar que tiene cierta capacidad de detectarlas a partir de los objetos que se encuentren en ella.

Hipótesis: ”La realización de Aprendizaje por Transferencia a partir de la red ImageNetCNN posibilita mejorar los resultados obtenidos al contrastar la hipótesis 4.4.2” .

4.4.6. Hipótesis 6

Un caso observado en la revisión de antecedentes es el de [3] en el cual se obtienen mejoras luego de aplicar un filtro de ecualización del histograma a las imágenes tanto para entrenar como para predecir.

Hipótesis: Realizar aprendizaje por transferencia utilizando PlacesCNN sobre un nuevo conjunto de datos obtenido a partir de aplicar el filtro utilizado en [3] para el conjunto de datos completo hará que los resultados sean mejores”.

4.5. Experimentos

Los conjuntos de datos a utilizar en los experimentos serán los generados en los trabajos [2] y [3], de los cuales se seleccionarán las escenas determinadas en las limitaciones del trabajo. En las tablas 1 y 2 se puede observar la distribución de escenas que contienen de los mismos. La métrica con la que se compararán los diferentes resultados será Precisión Categórica.

4.5.1. Experimento 1

Se entranará tanto una red convolucional como un perceptrón multicapa con las imágenes del conjunto de datos generado en el trabajo [2]. A continuación podremos comparar los resultados de ambos modelos de manera que será posible contrastarlos con la hipótesis 4.4.1.

El perceptrón multicapa del experimento cuenta con cinco capas ocultas totalmente conectadas de 1024, 1024, 512, 256 y 128 neuronas

Escena	Cantidad de imagenes
backyard	745
bathroom	793
bedroom	1593
frontyard	884
kitchen	992
living_room	852

Cuadro 1: Imágenes por escena - Conjunto de datos del trabajo [3]

Escena	Cantidad de imágenes
bathroom	10058
bedroom	24123
dining_room	19500
frontyard	24869
kitchen	24234
living_room	24210

Cuadro 2: Imágenes por escena - Conjunto de datos del trabajo [2]

en cada una, con una capa de dropout intercalada luego de cada una de ellas con probabilidad de 0.3, finalmente, una capa de salida con activación softmax como se observa en en anexo 7.4.

Por el lado de la red neuronal convolucional se utilizará una arquitectura que concatenará dos bloques compuestos por: capa convolucional, capa de normalización en lote, capa convolucional, capa de normalización en lote, capa de pooling y capa de dropout; en el primer bloque las capas convolutivas aplicarán 64 filtros cada una, mientras que en el segundo 128, todos ellos de tamaño $[5 \times 5]$ en todas las capas convolutivas, las capas de pooling aplicarán filtros de $[2 \times 2]$ y las capas de dropout tendrán asignada una probabilidad de 0.25. Luego de estos bloques, la red seguirá con una capa totalmente conectada de 512 neuronas con regularización L2, una capa dropout con probabilidad 0.5 y la capa de salida con activación Softmax. La

inicialización de los pesos de la red se realizará mediante inicialización de Xavier [14]. En 7.5 se puede observar la arquitectura de esta red.

En ambas redes se utilizaron de la misma manera las siguientes configuraciones:

- Tamaño del lote: la cantidad de imágenes a computar antes de realizar una actualización de pesos es de 64.
- Entradas: ambas redes se entrenaron con los tres canales de las imágenes (Red, Green, Blue), redimensionando cada una a 128×128 y luego normalizando el valor de los colores (se divide el valor de cada pixel sobre 255, el valor máximo en la escala RGB).
- Optimizador: exactitud categórica. En cada lote contabiliza las veces en las que la posición de la probabilidad más alta se condice con la posición de la predicción correcta. Vale aclarar que los resultados, al utilizar una capa de activación softmax son *one – hot encoders*, es decir, para cada imagen se tiene como resultado un vector de longitud igual a la cantidad de clases a predecir con la probabilidad para cada clase.
- Función de pérdida: entropía cruzada categórica pesada por clase definida en 13.

Se llevó a cabo el entrenamiento de ambas redes con una porción de 4671 imágenes, guardando N para validación y N para verificación. Los resultados se pueden observar en la tabla 3.

A partir de los resultados expuestos en la tabla previamente mencionada, no se cuenta con las evidencias necesarias para negar la hipótesis 4.4.1.

4.5.2. Experimento 2

Se entrenará la misma red que en 4.5.1 pero esta vez con diferentes subconjuntos del conjunto de datos del trabajo [2] y finalmente con

Modelo entrenado	Exactitud Categórica Conj. de Validación	Exactitud Categórica Conj. de Verificación
Perceptrón Multi-Capa	0.114	0.145
Red Neuronal Convolutacional	0.826	0.826

Cuadro 3: Exactitud categórica por modelo para los conjuntos de validación y verificación experimento 4.5.1

el conjunto de datos completo. A partir de los resultados obtenidos será posible contrastar la hipótesis 4.4.2.

En este experimento se utilizará la misma red neuronal convolucional detallada en el experimento 4.5.1, será entrenada con subconjuntos aleatorios tanto del diez como del cincuenta por ciento del conjunto de datos total elegido, además del entrenamiento con la totalidad del mismo. En la tabla 4 se puede observar la cantidad de imágenes de cada clase que se tienen para cada el entrenamiento en cada experimento. Además, en la tabla 5 se muestran las distribuciones de los conjuntos de datos de validación y verificación por clase, que serán fijos para hacer posible el testeo correcto de la hipótesis.

En la tabla 6 se muestran los resultados obtenidos para cada conjunto seleccionado. Como es posible apreciar, a medida que se incrementa el tamaño del conjunto de datos con el que se entrena, la red es capaz de predecir mejor las imágenes de validación y verificación. Con estos resultados no se cuenta la evidencia necesaria para declarar la hipótesis 4.4.2 como inválida.

4.5.3. Experimento 3

Se descargará la red preentrenada con el conjunto de datos Places presentada en [4] llamada PlacesCNN y se clasificarán las imágenes de los conjuntos de datos de validación y verificación previamente utilizados (en 4.5.2) con el fin constatar la hipótesis 4.4.3.

Al tratarse de una red entrenada para predecir sobre 365 categorías diferentes, sucede que algunas de las etiquetas del conjunto de

Porcentaje subconjunto	Escena	Cantidad de Imágenes
10	bathroom	877
	bedroom	2067
	dining_room	1662
	frontyard	2129
	kitchen	2074
	living_room	2084
50	bathroom	3588
	bedroom	8513
	dining_room	6897
	frontyard	8748
	kitchen	8620
	living_room	8607
100	bathroom	9088
	bedroom	21710
	dining_room	17534
	frontyard	22359
	kitchen	21848
	living_room	21822

Cuadro 4: Imágenes por conjunto de entrenamiento utilizado en el experimento 4.5.2

datos del trabajo [4] no tienen un par directo con las utilizadas para entrenar el modelo, por lo que se realizó un mapeo entre las clases que predice el modelo y las esperadas para el conjunto de datos propio. La forma de hacerlo fue asignando a cada una de las 365 etiquetas que es capaz de predecir el modelo una de las clases del conjunto de datos utilizado en el trabajo, de manera que cuando se clasifique un "baño" (en inglés *bathroom*) como "ducha" (*shower*) se considere como una predicción correcta. Los mapeos realizados se encuentran en la tabla 16 del anexo 7.3.

Escena	Cantidad de Imágenes para validación	Cantidad de Imágenes para verificación
bathroom	480	490
bedroom	1240	1173
dining_room	966	1000
frontyard	1249	1261
kitchen	1176	1210
living_room	1194	1194

Cuadro 5: Imágenes por conjunto de validación y verificación utilizado en el experimento 4.5.2

Imágenes en subconjunto de entrenamiento (%)	Exactitud Categórica Conj. de Validación	Exactitud Categórica Conj. de Verificación
10	0.606	0.595
50	0.653	0.657
100	0.752	0.739

Cuadro 6: Exactitud categórica por modelo para los conjuntos de validación y verificación experimento 4.5.2

Modelo	Exactitud Categórica Conj. de Validación	Exactitud Categórica Conj. de Verificación
Places-CNN con mapeo	0.056	0.05

Cuadro 7: Exactitud categórica por modelo para los conjuntos de validación y verificación experimento 4.5.3

Como es posible observar en la tabla 7, los resultados obtenidos al predecir directamente con la red Places-CNN preentrenada sobre los conjuntos de validación y verificación utilizados en el experimento 4.5.2 quedan por debajo del %6. Existen varias posibles razones para esto: que las imágenes del conjunto de datos con que se realizan las pruebas sean muy diferentes a las utilizadas para entrenar el

modelo de manera que las características aprendidas durante el entrenamiento no sean interpretables en estas imágenes, o que el modelo preentrenado realmente tenga una capacidad similar a la alcanzada para el subconjunto de clases comprobado, etc.

Con fundamento en los resultados obtenidos queda rechazada la hipótesis 4.4.3, dado que el modelo preentrenado no fue capaz de mejorar los resultados obtenidos a partir del experimento 4.5.2. De esta manera, el modelo obtenido en el experimento 4.5.2 continúa siendo el de mejor rendimiento para el conjunto de datos elegido.

4.5.4. Experimento 4

Se realizará aprendizaje por transferencia tomando como red base PlacesCNN y entrenando con el conjunto de datos elegido. A partir de esta red reentrenada, se podrá constatar la hipótesis 4.4.4.

Se reentrenó la red con el total de imágenes de entrenamiento definidas en el experimento 4.5.2 (114361) utilizando diferentes variantes en cuanto a arquitectura del clasificador (capas que siguen luego del último bloque convolutivo) como a la cantidad de capas preentrenadas que se vuelven a entrenar con los nuevos datos; a continuación se detallará cada una. Vale aclarar que para todas estas variantes capa de salida es la misma: una capa densa con activación *softmax* de 6 unidades.

Se realizaron variantes de entrenamiento tanto congelando (configurando como no entrenables) la red completa y entrenando un clasificador como congelando partes de la red y reentrenando el resto, también agregando un clasificador. Las pruebas realizadas se observan en los siguientes listados.

Aprendizaje por transferencia - variantes con las capas de la red preentrenada configuradas como no-entrenables:

- Variante 1: Luego de las capas convolutivas se agregó un clasificador con dos capas: una totalmente conectada de 512 neuronas con activación *Relu*, una capa de dropout con probabilidad de 0.5.

- Variante 2: Luego de las capas convolutivas se agregó un clasificador con una capa totalmente conectada de 512 neuronas con activación *Relu*.
- Variante 3: Luego de las capas convolutivas se entrenó un clasificador con una cuatro capas, compuestas por dos bloques con una capa totalmente conectada de 512 neuronas con activación *Relu* y una capa de dropout con probabilidad de 0.5 cada uno.

Aprendizaje por transferencia - variantes que reentrenan capas de la red preentrenada:

- Variante 4: Se configuraron como entrenables todas las capas de la red PlacesCNN preentrenada y se agregó un clasificador compuesto de una capa totalmente conectada de 512 neuronas con activación *Relu* y una capa de *Dropout* con probabilidad de 0.5.
- Variante 5: Se configuró como entrenable sólo el último bloque de la red PlacesCNN (de nombre *block₅* en la red), al cual se le agregó una capa de Pooling Máximo Global (llamada *GlobalMaxPooling* en inglés), que continuó con una capa totalmente conectada con activación *Relu* de 256 neuronas (también se hicieron experimentos con 128, 256 y 1024) y una capa de dropout con probabilidad de 0.5.
- Variante 6: Se configuraron como entrenables las capas 3(de nombre *block₃*), 4 (*block₄*) y 5 (*block₅*) de la red PlacesCNN, seguid de una capa totalmente conectada con activación *Relu* de 256 neuronas y una capa de dropout con probabilidad de 0.5.

Entre todos los experimentos, la variante que mejor resultó en razón de exactitud categórica fue la número 5 y sus resultados se observan en la tabla 8.

Como es posible observar en la tabla de resultados 8, en ninguno de los casos de alcanzan o sobrepasan los resultados obtenidos en el experimento 4.5.2, por lo que se refuta la hipótesis 4.4.4.

Modelo: Aprendizaje por Transferencia PlacesCNN	Exactitud Categórica Conj. de Validación	Exactitud Categórica Conj. de Verificación
Variante 1	0.45	0.44
Variante 2	0.19	0.18
Variante 3	0.42	0.41
Variante 4	0.18	0.19
Variante 5	0.657	0.639
Variante 6	0.07	0.07

Cuadro 8: Exactitud categórica por modelo para los conjuntos de validación y verificación experimento 4.5.4

4.5.5. Experimento 5

Se reentrenará la red ImageNetCNN con el mismo conjunto de datos utilizado en los experimentos 4.5.2, 4.5.3 y 4.5.4.

Al igual que con el experimento 4.5.4, se realizaron diferentes variantes de arquitecturas y capas entrenables para realizar la transferencia del aprendizaje. En todos los casos, de igual manera, la capa de salida es la misma: una capa densa con activación *softmax* de 6 unidades. Los diferentes experimentos realizados fueron:

- Variante 1: Se agregó una capa totalmente conectada de 512 neuronas, seguida de una capa de dropout con probabilidad de 0.5.
- Variante 2: Se agregó una capa *GlobalMaxPooling* luego de las capas totalmente conectadas, continuada por una capa densa de 256 neuronas, también conectada a una capa de dropout con probabilidad de 0.5.
- Variante 3: Se configuró como entrenable el último bloque convolucional de la red (*block*₅), el cual fue conectado a una capa

densa de 1024 neuronas seguida de una capa de dropout con probabilidad de 0.5.

- Variante 4: Se configuraron como entrenables las últimas dos capas convolucionales de la red ($block_5$ y $block_4$), y luego se conectaron a una capa densa de 1024 neuronas seguida por una capa de dropout con probabilidad 0.5.

Modelo: Aprendizaje por transferencia Imagenet-CNN	Exactitud Categórica Conj. de Validación	Exactitud Categórica Conj. de Verificación
Variante 1	0.51	0.50
Variante 2	0.34	0.34
Variante 3	0.627	0.623
Variante 4	0.18	0.19

Cuadro 9: Exactitud categórica por modelo para los conjuntos de validación y verificación experimento 4.5.5

En la tabla 9 se detallan los resultados de cada variante enunciada anteriormente. Como se puede observar la Variante 3 fue la que mejores resultados obtuvo, aunque igualmente no supera los resultados del experimento 4.5.2, por lo tanto la hipótesis 4.4.5 se declara inválida.

4.5.6. Experimento 6

Se aplicará ecualización del histograma a las imágenes de los conjuntos de datos elegidos, luego se procederá a los mismos pasos que en el experimento 4.5.4. De esta manera, será posible comprobar de la hipótesis 4.4.6.

A partir de la aplicación del filtro C.L.A.H.E. explicado en 2.1 se obtuvo un nuevo conjunto de datos, algunos de los ejemplos comparados se observan en la figura 22. Como se puede apreciar, existe

un mayor contraste entre los bordes y una clarificación de la imagen. Con este nuevo banco de imágenes se realizó Aprendizaje mediante Transferencia utilizando la misma arquitectura que alcanzó los mejores resultados en el experimento 4.5.4 (es decir, la Variante 5). Es importante aclarar que aunque se le haya aplicado un filtro a las imágenes, los conjuntos de entrenamiento, validación y verificación se mantuvieron iguales.

Modelo: Aprendizaje por Transferencia PlacesCNN	Exactitud Categórica Conj. de Validación	Exactitud Categórica Conj. de Verificación
Variante 5	0.656	0.642

Cuadro 10: Exactitud categórica para los conjuntos de validación y verificación experimento 4.5.6

En la tabla 10 se pueden observar los resultados del modelo, con lo cual no existen evidencias suficientes para decir que la hipótesis 4.4.6 sea inválida. El hecho de aplicar el filtro C.L.A.H.E. no hizo que los resultados mejoren en gran medida en relación a haber dejado las imágenes en crudo, pero de todos modos el modelo de este experimento obtuvo mejor exactitud categórica que el del experimento 4.5.4.

Normal



Filtro C.L.A.H.E.

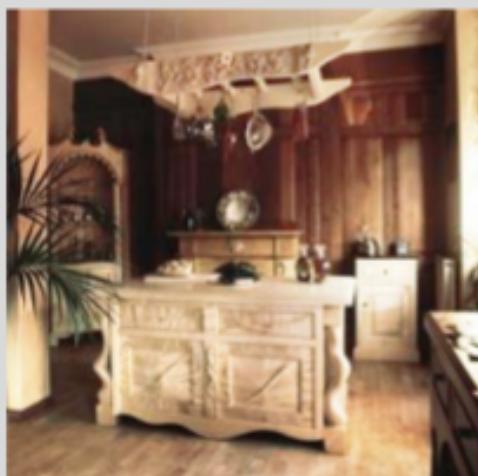
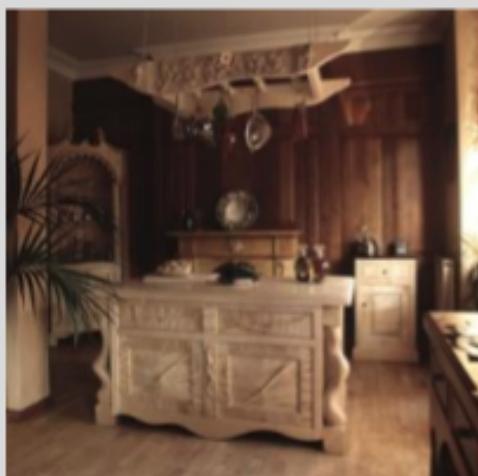


Figura 22: Ejemplo aplicación del algoritmo C.L.A.H.E. en imágenes del dataset utilizado en el experimento 4.5.6

5. Análisis de los resultados obtenidos

A partir de las hipótesis planteadas es posible dar cuenta de que a veces los resultados esperados según la teoría no se dan en la práctica, y que en algunos casos puede resultar mejor entrenar una arquitectura propia desde cero que intentar sacar provecho de redes preentrenadas. Como se pudo observar en el trabajo, ninguno de los experimentos que se hicieron utilizando aprendizaje mediante transferencia alcanzó los resultados obtenidos por la arquitectura planteada en el experimento 4.5.2 para el conjunto de datos de [2].

5.1. Error por categoría modelos obtenidos en los experimentos 4.5.2 y 4.5.6

Como se observa en las tablas 11 y 12, no todas las categorías se predicen correctamente en la misma medida, es decir, cada modelo aprende a clasificar mejor algunas categorías que otras. Llama la atención que para estos modelos se cumple que el orden de las categorías contabilizando clasificaciones incorrectas es el mismo en ambos, es decir, ambos modelos tienen menor cantidad de errores en las mismas categorías; el orden (ascendente) es: *frontyard*, *bathroom*, *diningroom*, *kitchen*, *bedroom* y *livingRoom*. Tanto la calidad de las imágenes de las categorías con mayor tasa de error como el solapamiento de posibles entidades que pueden formar parte de estas escenas son razones que podrían explicar este comportamiento.

5.2. Distribución de probabilidades de salida para clasificaciones erróneas

Como se mencionó en 3, la capa de salida de una red con activación *Softmax* devuelve la probabilidad de cada clase (de las que luego se elije la de mayor magnitud).

Una situación que puede estar sucediendo es que en los aquellos casos en que las redes fallan en su clasificación, la magnitud de la

Categoría	Cantidad errores	% del total
frontyard	100	0.06
bathroom	182	0.11
dining_room	220	0.13
kitchen	269	0.16
bedroom	384	0.23
livingRoom	497	0.30

Cuadro 11: Error por categoría y porcentaje sobre el total - Modelo del experimento 4.5.2

Categoría	Cantidad errores	% del total
frontyard	145	0.06
bathroom	148	0.07
dining_room	356	0.16
kitchen	504	0.22
bedroom	537	0.24
livingRoom	574	0.25

Cuadro 12: Error por categoría y porcentaje sobre el total - Modelo delexperimento 4.5.6

probabilidad que arrojan es baja. Una forma de analizar estos casos es mediante algunos estadísticos simples de las probabilidades asignadas a cada clase cuando la red falló.

A diferencia de lo que se podía esperar, las redes se equivocan prediciendo con probabilidades mayores o iguales a 0.5, lo cual es una probabilidad alta ya que queda el resto (entre 1 y esta probabilidad) para las demás 5 clases.

Conforme estos resultados se analizan, surge el interroongante de conocer qué sucedería si se define un umbral mínimo para siquiera considerar la predicción de un modelo, es decir que los casos que tengan una probabilidad por debajo del umbral directamente se configuren como no predecibles o no aceptados. En las figuras 23 y 24 se

Categoría	Probabilidad asignada				
	media	mediana	desvío	mín.	máx.
bathroom	0.63	0.61	0.20	0.26	1.00
bedroom	0.64	0.62	0.18	0.26	1.00
dining_room	0.67	0.64	0.19	0.26	1.00
frontyard	0.63	0.62	0.19	0.26	0.99
kitchen	0.66	0.63	0.19	0.31	1.00
livingRoom	0.69	0.70	0.19	0.26	1.00

Cuadro 13: Estadísticos de probabilidades asignadas en clasificaciones erróneas - Modelo de experimento 4.5.2

Categoría	Probabilidad asignada				
	media	mediana	desvío	mín.	máx.
bathroom	0.56	0.53	0.19	0.19	1.00
bedroom	0.59	0.56	0.19	0.25	1.00
dining_room	0.60	0.57	0.18	0.20	1.00
frontyard	0.55	0.50	0.20	0.19	0.98
kitchen	0.61	0.59	0.19	0.23	1.00
livingRoom	0.61	0.59	0.19	0.22	1.00

Cuadro 14: Estadísticos de probabilidades asignadas en clasificaciones erróneas - Modelo de experimento 4.5.6

puede observar tanto la cantidad de casos alcanzados utilizando diferentes umbrales, la exactitud categórica y el porcentaje de errores que se dejarían de cometer para umbrales entre cero y uno.

En la figura 23 con un umbral de 0.55 se alcanzaría a predecir 5480 de los 6328 casos totales con una exactitud categórica del %80, mientras que utilizando el mismo umbral para el modelo del experimento 4.5.6, en la figura 24 se alcanzan sólo 4625 casos, con un %71 de exactitud categórica.

En las figuras 25 y 26 podemos observar un gráfico de caja de la probabilidad con la que se determinó cada etiqueta, dividido por

Cantidad de casos, porcentaje de errores y exactitud categórica por umbral - modelo del exp. 2

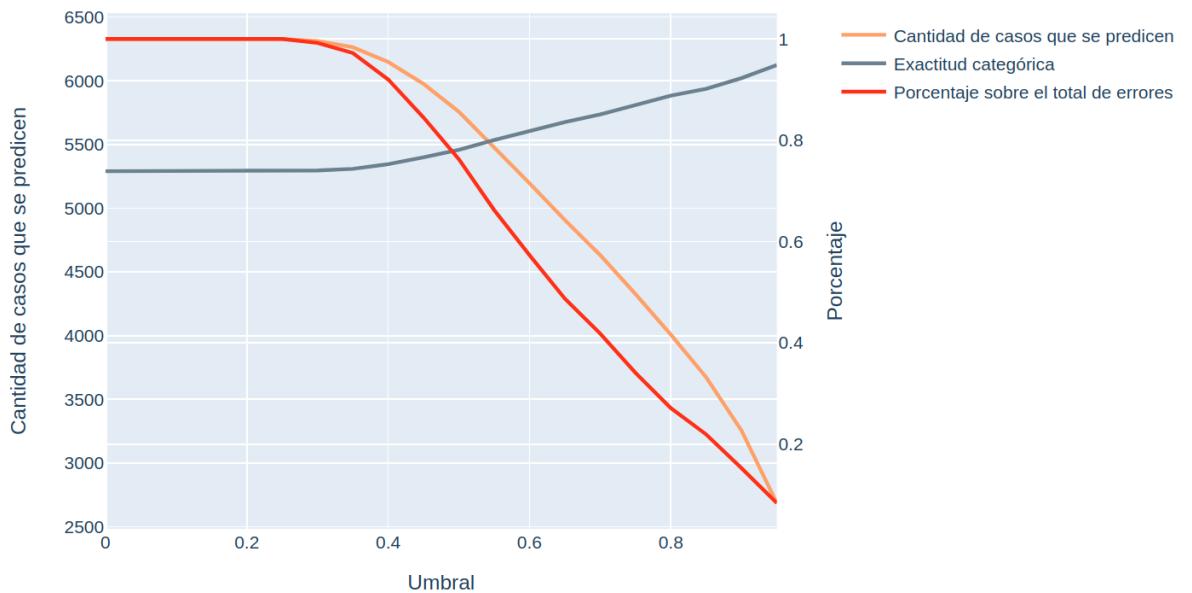


Figura 23: Cantidad de errores y exactitud categórica por umbral para el modelo del experimento 4.5.2

categoría según si fueron correcta o incorrectamente clasificados. En los mismos podemos observar que para los casos clasificados correctamente las predicciones tienen distribuciones de probabilidad muy diferentes a los casos incorrectos, por lo que cobra mayor sentido seleccionar un umbral para las predicciones. Un punto no menor a mencionar es que la etiqueta *frontyard* es la que ambos modelos mejor predicen, mientras que las mayores diferencias entre cada modelo se observan en las categorías *kitchen* y *bathroom*.

5.3. Exactitud categórica casos que deberían poder predecirse correctamente

A continuación se revisará un subconjunto de casos que deberían ser clasificados correctamente por un modelo, debido a que son claros ejemplos de la clase a la que se corresponden. Este subconjunto

Cantidad de casos, porcentaje de errores y exactitud categórica por umbral - modelo del exp. 6

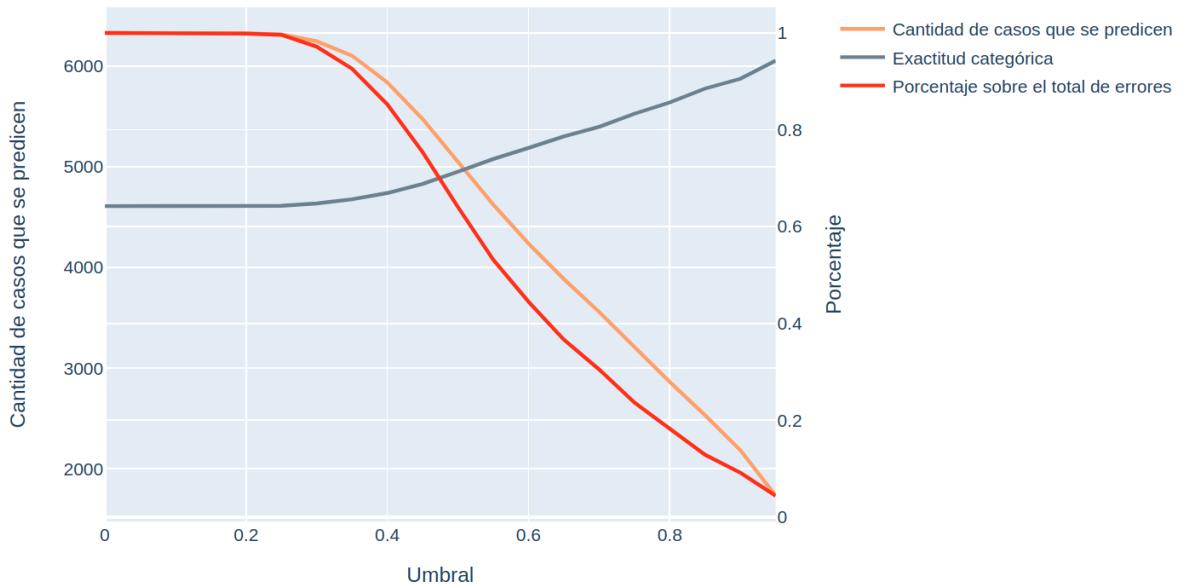


Figura 24: Cantidad de errores y exactitud categórica por umbral para el modelo del experimento 4.5.6

contiene 25 imágenes por categoría seleccionadas del conjunto de verificación utilizado para los experimentos 4.5.2 al 4.5.6.

La tabla 15 muestra que para el subconjunto elegido los modelos obtienen un gran incremento en relación a las métricas proporcionadas anteriormente. Se puede razonar que las imágenes que peor resultado tienen son aquellas que contienen demasiado ruido en relación al relacionado a la escena.

Distribución de probabilidades asignadas a la categoría elegida para casos predichos correcta e incorrectamente

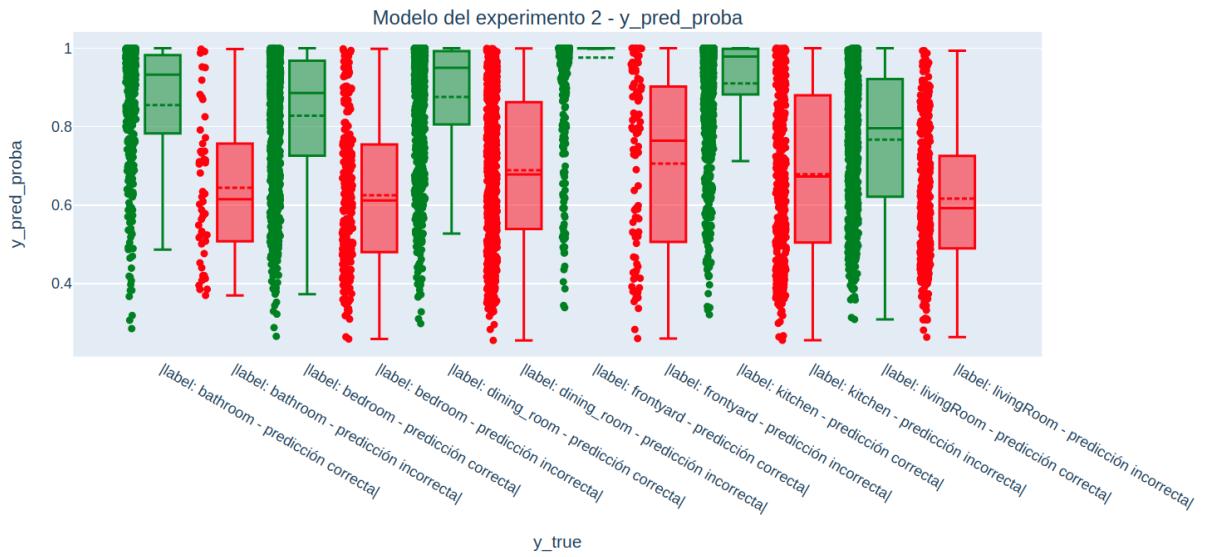


Figura 25: Distribución de probabilidades asignadas por clase para el modelo del experimento 4.5.6

Modelo obtenido en experimento	Exactitud Categórica Subconjunto elegido
Experimento 2 (4.5.2)	0.94
Experimento 6 (4.5.6)	0.78

Cuadro 15: Resultados obtenidos por los modelos en experimentos 4.5.2 y 4.5.6 para el subconjunto de imágenes elegido

Distribución de probabilidades asignadas a la categoría elegida para casos predichos correcta e incorrectamente

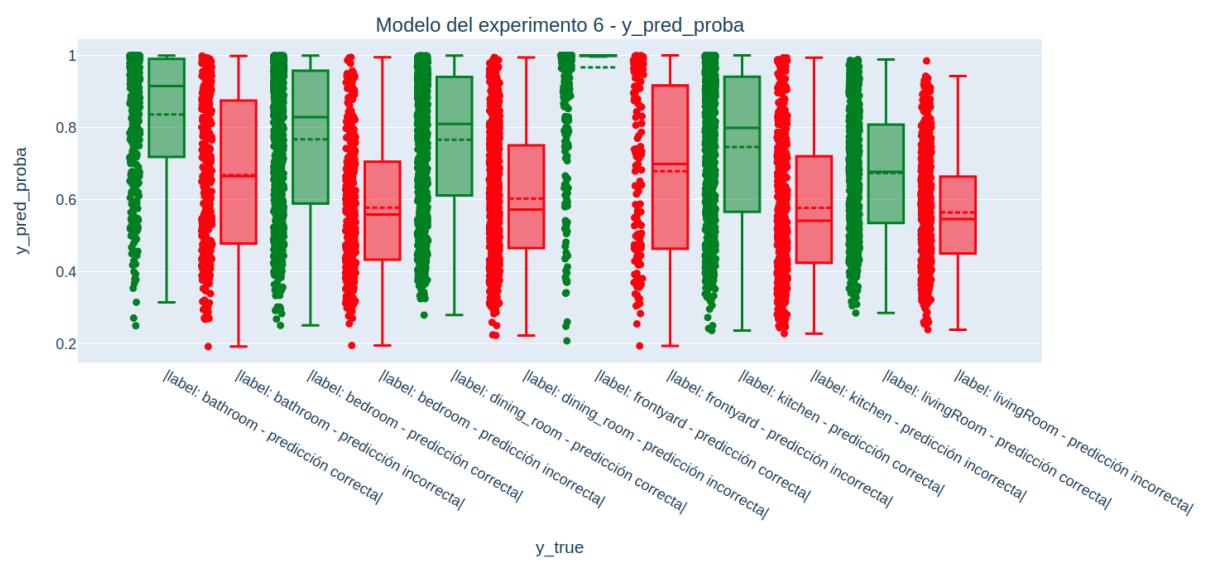


Figura 26: Distribución de probabilidades asignadas por clase para el modelo del experimento 4.5.6

6. Conclusiones

6.1. Conclusión general

En este trabajo se planteó llevar a cabo una investigación en la que mediante la aplicación de técnicas de Aprendizaje Profundo se realice la tarea de clasificar escenas relacionadas a propiedades inmuebles, se realizó la búsqueda de investigaciones similares con el fin de llegar a hipótesis

A partir de la revisión de antecedentes se obtuvieron dos conjuntos de datos con los que se plantearon diferentes experimentos, por un lado el realizado en [2] y por otro el de [3]. Ambos con diferentes cantidades de imágenes pero también con diferentes calidades en relación a las imágenes que contienen. Por un lado, el banco de datos del trabajo [3] tiene menor cantidad de imágenes, representa un 5% de las del trabajo [2], aunque de igual modo sus imágenes resultan más representativas a cada escena que componen. Mencionado este punto, los resultados de cada experimento estuvieron sujetos a la calidad de estos bancos de datos de escenas, en el experimento 4.5.1 que se utilizó el conjunto de datos de [3] se alcanzaron mejores resultados que en el mejor de los casos en los que se usó [2]; estos bancos de datos se eligieron a sabiendas de la calidad de cada uno debido a que en ambientes productivos pueden ocurrir ambos casos: que las imágenes sean más representativas a su escena o que tengan cierto ruido”, como ser fotos sacadas de lejos, desde ángulos poco vistosos, escenas que no contengan mucha información, escenas de un mismo tipo pero que por pertenecer a diferentes culturas no contienen las mismas entidades, etcétera.

En este proyecto se realizaron y validaron hipótesis que buscando abordar la resolución del problema inicial planteado: la clasificación de escenas de propiedades inmuebles mediante técnicas de aprendizaje profundo. Los resultados de los experimentos permiten dar a luz conocimiento sobre algunos posibles enfoques a la hora de enfrentar este problema como son: entrenar una red propia como también tomar provecho de redes preentrenadas.

A partir de lo demostrado en el análisis de la sección 5 es posible decir que:

- Los modelos no aciertan con el mismo ratio en todas las categorías.
- Las distribuciones de las probabilidades con las que se predicen los casos mal etiquetados son muy diferentes a las de los casos etiquetados correctamente, lo que señala que los modelos cuando predicen correctamente lo hacen con un determinado nivel de certidumbre.
- Para ambientes productivos se podría setear un *umbral* con el cual se acepten las predicciones realizadas o no, de manera que si no se aceptan las predicciones se pueda tomar acción ante cada imagen.
- Ante casos en los que se espera que los modelos funcionen correctamente queda demostrado que la exactitud categórica mejora de manera significativa, por lo que se asume que una de las posibles razones de que un modelo no alcance resultados correctos es la calidad de la escena que se busca clasificar.
- Para el modelo obtenido en el experimento 4.5.2 se puede brindar predicción de un 87 % de los casos con un 80 % de probabilidad de acierto.

Por otro lado, disponibilizar estos modelos listos para realizar predicciones sobre las clases elegidas en 4.2 resulta uno de los grandes logros de este trabajo ya que pueden servir tanto para utilizarlos en ambientes productivos como para ser el puntapié inicial a nuevas investigaciones, proyectos, modelos e incluso la aplicación de Aprendizaje Mediante Transferencia a partir de los mismos.

Otro aporte de valor del trabajo es el repositorio abierto en donde se encuentra todo el código (y libretas *Jupyter*) generado, utilizado, estructurado y modularizado como se explica en 7.1, que por el hecho

de que pueden reutilizarse y extenderse libremente para el uso que se decida ya resulta meritorio.

La clasificación de escenas de propiedades inmuebles resulta de alta importancia para algunos sectores y un modelo de Aprendizaje Profundo entrenado para tal tarea puede utilizarse tanto para lo mencionado en 1.2 como para otras necesidades que puedan surgir y se pueda tomar provecho del mismo. En este trabajo se han investigado posibles enfoques a utilizar, se han definido hipótesis que ayuden a llegar al resultado esperado, se han entrenado modelos de Aprendizaje Profundo para realizar esta tarea y se ha logrado obtener conocimiento de algunas cuestiones que pueden servir para futuras investigaciones. El objetivo planteado inicialmente fue alcanzado, aunque siempre queda más hilo por tirar del carrete y este no es un caso aparte, pueden introducirse mejoras tanto en la red elegida, como en la calidad de los datos o el preprocesamiento que se le aplica a las imágenes.

6.2. Lineamientos Futuros

Como quedó demostrado en 2 el espectro de posibles enfoques para solucionar este problema incluye obviamente variantes de Redes Neuronales Convolucionales, pero también otros tipos de redes pudieren ser capaces de dar mejor respuesta a la clasificación de escenas como pueden ser las Redes Neuronales Recurrentes o las planteadas (pero aún no incluidas en librerías estándar) Redes Neuronales Convolucionales Aleatorias Potenciadas por el Gradiente; con esto se quiere decir que es posible que a través de la utilización de otras arquitecturas (ya existentes o no) la tarea se pueda realizar con mayor exactitud.

Algunos trabajos a futuro que se podrían plantear son:

- Investigaciones sobre el error que cometen los modelos planteados en este trabajo, desde más alto a más bajo nivel, es decir: desde las categorías que se predicen con menor exactitud entre

el resto hasta las activaciones de las neuronas en las capas convolucionales, con el fin de intentar solucionar estos problemas.

- Recolección de datos relacionada a una cultura o a un país en específico para entrenar modelos diferenciados
- En el experimento 4.5.5 se realizó Aprendizaje Mediante Transferencia utilizando la red ImageNetCNN que fue entrenada con cientos de objetos que no necesariamente pertenecen al contexto de las escenas elegidas en 4.2, por lo que tendría sentido reentrenar esta red con objetos que sean apacibles de aparecer en las escenas seleccionadas y luego a partir de esa nueva red dar lugar al Aprendizaje por Transferencia.
- Investigaciones haciendo Aprendizaje por Transferencia utilizando versiones de redes preentrenadas con otras arquitecturas base como pueden ser ResNet o InceptionV3, tanto para PlacesCNN como para ImageNetCNN.
- En el experimento 4.5.6 se observó una mejora mínima en los resultados en relación al modelo del experimento 4.5.4, siendo que la única diferencia entre ambos es que al conjunto de datos se le aplicó un filtro para mejorar la luminosidad y contraste de las entidades presentes. Esta situación es un posible indicador que la aplicación de otros filtros o realizar Aumentación de Datos podría llevar a mejores resultados.
- Es cierto que los experimentos 4.5.4, 4.5.5 y 4.5.6 obtuvieron resultados muy similares por lo que podría plantearse revisar el conjunto de datos y realizar una limpieza de aquellas imágenes que contengan demasiado ruido o que sean menos representativas de cada escena bajo algún criterio (calidad de la imagen, cantidad de objetos pertenecientes a la escena encontrados, etc) ya que estos casos podrían estar impidiendo el correcto entrenamiento de los modelos.

7. Anexo

7.1. Anexo 1: Repositorio abierto del trabajo

Para este trabajo final se utilizó la herramienta de control de versionado GIT, puntualmente alojado en la plataforma GitHub, y el tanto el código como los modelos generados por este trabajo son abiertos, es decir, cualquier persona puede acceder y hacer uso de lo realizado bajo su propia responsabilidad (ver Licencia del proyecto 7.2). Además, la estructura de los directorios elegida se detalla a continuación:

- LICENSE: Licencia del proyecto.
- README.md: Archivo de lectura inicial para desarrolladores o quien esté interesado en conocer el proyecto.
- data: directorio con los conjuntos de datos.
 - external: bancos de datos comprimidos.
 - interim: conjuntos de datos utilizados en cada experimento comprimidos.
 - processed: directorios con conjuntos de datos finales para entrenar y medir rendimiento de modelos (divididos en conjuntos de entrenamiento, validación y verificación cada uno).
 - raw: bancos de datos en crudo (sin dividir en conjuntos diferentes).
- docs: directorio con el presente documento, tanto en su versión compilable LaTex como en PDF.
- models: directorio con los modelos entrenados, instancias de abstracciones propias creadas y archivos json con los pesos de las redes entrenadas.
- notebooks: directorio con Jupyter Notebooks versionados con los que se trabajó durante el proyecto.

- requirements.txt: archivo con librerías requeridas para poder ejecutar correctamente el contenido del repositorio.
- setup.py: archivo de instalación del repositorio para poder importarlo como una librería python.
- src: directorio con el código fuente utilizado en el proyecto.
 - data: módulos utilizados para generar los diferentes conjuntos de datos.
 - models: módulos con las abstracciones utilizadas para entrenar y medir el rendimiento de los modelos.
 - visualization: módulos con visualizaciones.

7.2. Anexo 2: Licencia del proyecto

La licencia del presente proyecto es la *Licencia MIT* (en su versión X11), es para software libre de código abierto y especifica lo siguiente:

1. **Condiciones:** La condición es que la nota de copyright y la parte de los derechos se incluya en todas las copias o partes sustanciales del Software. Esta es la condición que invalidaría la licencia en caso de no cumplirse.
2. **Derechos:** sin restricciones; incluyendo usar, copiar, modificar, integrar con otro Software, publicar, sublicenciar o vender copias del Software, y además permitir a las personas a las que se les entregue el Software hacer lo mismo.
3. **Limitación de responsabilidad:** finalmente se tiene un disclaimer o nota de limitación de la responsabilidad habitual en este tipo de licencias.

7.3. Anexo 3: Mapeo de clases realizado para el experimento 4.5.3

En la tabla 16 se presentan los mapeos elegidos entre las clases con las que la red PlacesCNN fue entrenada y las clases esperadas que se predigan para el dataset utilizado en el experimento 4.5.3.

7.4. Anexo 4: Arquitectura del Perceptrón Multicapa utilizado en experimento 4.5.1

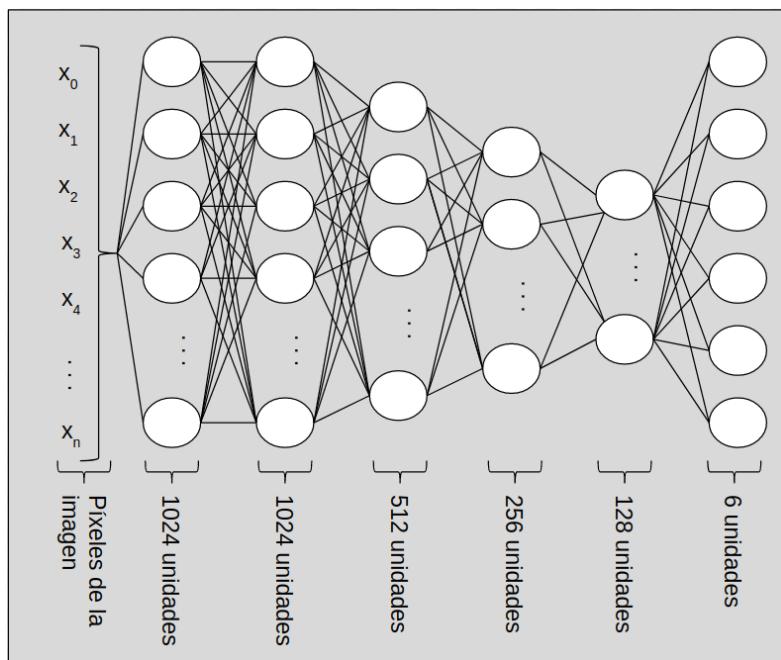


Figura 27: Arquitectura perceptrón multicapa del experimento 4.5.1

```

##### CNN_Model #####
Model: "sequential_2"

Layer (type)          Output Shape       Param #
=====                =====
conv2d_5 (Conv2D)      (None, 124, 124, 64)    4864
batch_normalization_5 (Batch Normalization) (None, 124, 124, 64)    256
conv2d_6 (Conv2D)      (None, 120, 120, 64)    102464
batch_normalization_6 (Batch Normalization) (None, 120, 120, 64)    256
max_pooling2d_3 (MaxPooling2D) (None, 60, 60, 64)    0
dropout_4 (Dropout)   (None, 60, 60, 64)    0
conv2d_7 (Conv2D)      (None, 56, 56, 128)    204928
batch_normalization_7 (Batch Normalization) (None, 56, 56, 128)    512
conv2d_8 (Conv2D)      (None, 52, 52, 128)    409728
batch_normalization_8 (Batch Normalization) (None, 52, 52, 128)    512
max_pooling2d_4 (MaxPooling2D) (None, 26, 26, 128)    0
dropout_5 (Dropout)   (None, 26, 26, 128)    0
flatten_2 (Flatten)   (None, 86528)        0
dense_3 (Dense)       (None, 512)         44302848
dropout_6 (Dropout)   (None, 512)         0
dense_4 (Dense)       (None, 6)           3078
=====
Total params: 45,029,446
Trainable params: 45,028,678
Non-trainable params: 768

```

Figura 28: Arquitectura Red Neuronal Convolucional de los experimentos 4.5.1 y 4.5.2

- 7.5. Anexo 5: Arquitectura de la Red Neuronal Convolucional utilizada en los experimentos 4.5.1 y 4.5.2**
- 7.6. Anexo 6: Arquitectura de la red PlacesCNN utilizada en experimento 4.5.3**
- 7.7. Anexo 7: Arquitectura utilizada en experimentos 4.5.4 y 4.5.6**
- 7.8. Anexo 8: Arquitectura utilizada en experimento 4.5.5**

Clase en Places365	Mapeo	Clase en Places365	Mapeo
apartment_building/outdoor	frontyard	hospital_room	bedroom
bathroom	bathroom	hotel_room	bedroom
bedchamber	bedroom	house	frontyard
bedroom	bedroom	kasbah	frontyard
building_facade	frontyard	kitchen	kitchen
chalet	frontyard	lecture_room	livingRoom
childs_room	bedroom	living_room	livingRoom
clean_room	bedroom	lobby	livingRoom
closet	bedroom	manufactured_home	frontyard
cottage	frontyard	office_cubicles	livingRoom
courthouse	frontyard	patio	frontyard
courtyard	frontyard	porch	frontyard
diner/outdoor	dining_room	recreation_room	livingRoom
dining_hall	dining_room	restaurant_kitchen	kitchen
dining_room	dining_room	shower	bathroom
doorway/outdoor	frontyard	television_room	livingRoom
dorm_room	bedroom	television_studio	livingRoom
dressing_room	bedroom	waiting_room	livingRoom
home_office	livingRoom	yard	frontyard
home_theater	livingRoom		

Cuadro 16: Mapeo entre clases con las que la red PlacesCNN fue entrenada y las clases del conjunto de datos utilizado en el experimento 4.5.3

##### ##### Model: "vgg16-places365"		
Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102764544
drop_fc1 (Dropout)	(None, 4096)	0
fc2 (Dense)	(None, 4096)	16781312
drop_fc2 (Dropout)	(None, 4096)	0
predictions (Dense)	(None, 365)	1495405

Total params: 135,755,949
Trainable params: 135,755,949
Non-trainable params: 0

Figura 29: Arquitectura Red Neuronal Convolucional del experimento 4.5.3

```

##### PlacesOnTop_Model #####
Model: "model_1"

Layer (type)          Output Shape       Param #
=====
input_1 (InputLayer)   [(None, 224, 224, 3)]   0
block1_conv1 (Conv2D)  (None, 224, 224, 64)    1792
block1_conv2 (Conv2D)  (None, 224, 224, 64)    36928
block1_pool (MaxPooling2D) (None, 112, 112, 64)  0
block2_conv1 (Conv2D)  (None, 112, 112, 128)   73856
block2_conv2 (Conv2D)  (None, 112, 112, 128)   147584
block2_pool (MaxPooling2D) (None, 56, 56, 128)  0
block3_conv1 (Conv2D)  (None, 56, 56, 256)    295168
block3_conv2 (Conv2D)  (None, 56, 56, 256)    590080
block3_conv3 (Conv2D)  (None, 56, 56, 256)    590080
block3_pool (MaxPooling2D) (None, 28, 28, 256)  0
block4_conv1 (Conv2D)  (None, 28, 28, 512)   1180160
block4_conv2 (Conv2D)  (None, 28, 28, 512)   2359808
block4_conv3 (Conv2D)  (None, 28, 28, 512)   2359808
block4_pool (MaxPooling2D) (None, 14, 14, 512)  0
block5_conv1 (Conv2D)  (None, 14, 14, 512)   2359808
block5_conv2 (Conv2D)  (None, 14, 14, 512)   2359808
block5_conv3 (Conv2D)  (None, 14, 14, 512)   2359808
block5_pool (MaxPooling2D) (None, 7, 7, 512)  0
global_max_pooling2d_1 (Glob (None, 512)      0
fc1 (Dense)           (None, 256)        131328
drop_fc1 (Dropout)    (None, 256)        0
predictions (Dense)   (None, 6)         1542
=====

Total params: 14,847,558
Trainable params: 7,212,294
Non-trainable params: 7,635,264

```

Figura 30: Arquitectura Red Neuronal Convolucional de los experimentos 4.5.4 y 4.5.6

```

##### PlacesOntop_Model #####
Model: "model_1"

Layer (type)          Output Shape       Param #
=====
input_1 (InputLayer)   [(None, 224, 224, 3)]   0
block1_conv1 (Conv2D)  (None, 224, 224, 64)    1792
block1_conv2 (Conv2D)  (None, 224, 224, 64)    36928
block1_pool (MaxPooling2D) (None, 112, 112, 64)  0
block2_conv1 (Conv2D)  (None, 112, 112, 128)   73856
block2_conv2 (Conv2D)  (None, 112, 112, 128)   147584
block2_pool (MaxPooling2D) (None, 56, 56, 128)  0
block3_conv1 (Conv2D)  (None, 56, 56, 256)    295168
block3_conv2 (Conv2D)  (None, 56, 56, 256)    590080
block3_conv3 (Conv2D)  (None, 56, 56, 256)    590080
block3_pool (MaxPooling2D) (None, 28, 28, 256)  0
block4_conv1 (Conv2D)  (None, 28, 28, 512)   1180160
block4_conv2 (Conv2D)  (None, 28, 28, 512)   2359808
block4_conv3 (Conv2D)  (None, 28, 28, 512)   2359808
block4_pool (MaxPooling2D) (None, 14, 14, 512)  0
block5_conv1 (Conv2D)  (None, 14, 14, 512)   2359808
block5_conv2 (Conv2D)  (None, 14, 14, 512)   2359808
block5_conv3 (Conv2D)  (None, 14, 14, 512)   2359808
block5_pool (MaxPooling2D) (None, 7, 7, 512)  0
global_max_pooling2d_1 (Glob (None, 512)      0
fc1 (Dense)           (None, 256)        131328
drop_fc1 (Dropout)    (None, 256)        0
predictions (Dense)   (None, 6)         1542
=====

Total params: 14,847,558
Trainable params: 7,212,294
Non-trainable params: 7,635,264

```

Figura 31: Arquitectura Red Neuronal Convolucional del experimento 4.5.5
81

Referencias

- [1] H. Patel and H. Mewada, “Analysis of machine learning based scene classification algorithms and quantitative evaluation,” *International Journal of Applied Engineering Research*, vol. 13, no. 10, pp. 7811–7819, 2018.
- [2] O. Poursaeed, T. Matera, and S. Belongie, “Vision-based real estate price estimation,” *Machine Vision and Applications*, vol. 29, no. 4, pp. 667–676, 2018.
- [3] J. H. Bappy, J. R. Barr, N. Srinivasan, and A. K. Roy-Chowdhury, “Real estate image classification,” in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 373–381, IEEE, 2017.
- [4] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, “Learning deep features for scene recognition using places database,” in *Advances in neural information processing systems*, pp. 487–495, 2014.
- [5] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, “Object detectors emerge in deep scene cnns,” *arXiv preprint arXiv:1412.6856*, 2014.
- [6] L. Herranz, S. Jiang, and X. Li, “Scene recognition with cnns: objects, scales and dataset bias,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 571–579, 2016.
- [7] F. Zhang, B. Du, and L. Zhang, “Scene classification via a gradient boosting random convolutional network framework,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 3, pp. 1793–1802, 2015.
- [8] Hemnet, *Hemnet*. <https://www.hemnet.se/>, último acceso: 2019-12-08.

- [9] A. Esteban, *Restb AI*. <https://restb.ai/>, último acceso: 2019-12-08.
- [10] X. Zhu and A. B. Goldberg, “Introduction to semi-supervised learning,” *Synthesis lectures on artificial intelligence and machine learning*, vol. 3, no. 1, pp. 1–130, 2009.
- [11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [12] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.
- [13] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.
- [14] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.
- [15] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, “Loss functions for image restoration with neural networks,” *IEEE Transactions on Computational Imaging*, vol. 3, no. 1, pp. 47–57, 2016.
- [16] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [17] D. Choi, C. J. Shallue, Z. Nado, J. Lee, C. J. Maddison, and G. E. Dahl, “On empirical comparisons of optimizers for deep learning,” *arXiv preprint arXiv:1910.05446*, 2019.

- [18] P. Baldi and P. J. Sadowski, “Understanding dropout,” in *Advances in neural information processing systems*, pp. 2814–2822, 2013.
- [19] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *ArXiv*, vol. abs/1502.03167, 2015.
- [20] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, “Activation functions: Comparison of trends in practice and research for deep learning,” *arXiv preprint arXiv:1811.03378*, 2018.
- [21] Y. LeCun, Y. Bengio, *et al.*, “Convolutional networks for images, speech, and time series,” 1995.
- [22] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge,” *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.
- [23] M. R. Ronchi and P. Perona, “Describing common human visual actions in images,” in *Proceedings of the British Machine Vision Conference (BMVC)* (M. W. J. Xianghua Xie and G. K. L. Tam, eds.), pp. 52.1–52.12, BMVA Press, September 2015.
- [24] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.
- [26] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of*

the IEEE conference on computer vision and pattern recognition, pp. 4700–4708, 2017.

- [27] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- [28] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
- [29] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language models are unsupervised multitask learners,” *OpenAI Blog*, vol. 1, no. 8, 2019.
- [30] Scikit-learn, *Balanced Accuracy Score*. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.balanced_accuracy_score.html, último acceso: 2019-12-08.