UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CONSTRUCCIÓN DE FEATURES PARA IMÁGENES SATELITALES UTILIZANDO DEEP CONVOLUTIONAL NETWORKS PARA LA ESTIMACIÓN DEL VALOR DE VENTA DE PROPIEDADES EN CHILE

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

MARIO WILFREDO GARRIDO FERNÁNDEZ

PROFESOR GUÍA:
PABLO GUERRERO PÉREZ

MIEMBROS DE LA COMISIÓN:
BARBARA POBLETE LABRA
JOSÉ SAAVEDRA RONDO

SANTIAGO DE CHILE
2019

Resumen

El presente trabajo tiene como objetivo la construcción de un modelo de regresión que permita predecir precios de venta de casas en la ciudad de Santiago. Para lograr esto se explota la información disponible en internet correspondiente a avisos de venta de propiedades.

Reconociendo que muchos anuncios de venta publicados en internet cuentan con información de la ubicación de la propiedad, ya sea en forma de coordenadas o, más comúnmente, como una marca en Google Maps o algún otro servicio de mapas *online*, existe una rica fuente de información a explotar: El entorno de la propiedad descrito en formato de imágenes.

Un buen modelo necesita una buena fuente de datos para aprender, por lo que se procede a explotar la información extra de los anuncios utilizando redes convolucionales profundas, entrenadas para resolver un problema de clasificación llamado *Imagenet*. Estas redes se re-entrenan parcialmente para generar *deep features* de imágenes satelitales centradas en la propiedad a distintos niveles de *zoom*. Cada una de estas imágenes entrega información respecto a distintos aspectos de la propiedad: Características del terreno, aspecto del vecindario, presencia de masas de agua, cercanía a montañas, entre otros.

Para re-entrenar las redes y generar los *features* se construye un *dataset* de anuncios de venta de propiedades ubicadas en las comunas de Santiago, Estación Central, Las Condes, La Florida, Puente Alto y Maipú.

Se experimenta con las métricas *RMSE*, R^2 y *MAE*, para determinar el desempeño de los modelos generados.

Los experimentos revelan que, para *Random Forest*, complementar la información original de los anuncios con los *deep features* generados permite superar el desempeño en regresión obtenido sin ellos, disminuyendo cerca de un 8 % el *RMSE*.

Never surrender.

Tabla de Contenido

Introducción

El mercado inmobiliario se encuentra siempre en constante cambio y puede ser difícil determinar el valor de una propiedad. Este valor se encuentra ampliamente, pero no completamente, determinado por las características de la propiedad, ya sea su tamaño, ubicación, año de construcción, entre otros. Es la principal motivación del presente trabajo crear una herramienta que permita, dada la información contenida en un aviso de venta de propiedad, estimar el valor de esta, en función de los valores manejados en el mercado.

Páginas tales como Portal Inmobiliario[1] o Goplaceit[2] constituyen uno de los puntos de partida más comúnmente utilizados actualmente para buscar una propiedad en Chile. En estas páginas se ha transformado en costumbre contar con imágenes que caracterizan a la propiedad: Fotografías de la fachada y del interior de las propiedades y una marca en un mapa (comunmente de Google Maps) para indicar la posición de la propiedad en el planeta, siendo estas imágenes y marcas una de las fuentes de información más ricas para quien busca propiedades al momento de evaluar las ofertas, fuera de visitar la propiedad.

Gracias a los avances de los últimos 30 años se ha visto un fuerte resurgimiento del campo de las redes neuronales, las que han demostrado un gran poder de expresividad para aproximar funciones que permiten resolver problemas complejos y variados. El problema de la predicción de precios de propiedades ha sido abordado a través de las redes neuronales[3], enfocándose, principalmente, en indicadores compilados a mano que describen la estructura y la calidad de las propiedades (*i.e:* Tamaño, número de baños, pisos, año de construcción, etc), como datos de entada.

Gracias a los avances en *Deep Learning*[4], particularmente el desarrollo de las redes convolucionales profundas[5], se logra, hoy en día, resolver complejos problemas de clasificación de imágenes, siendo posible construir redes que pueden reconocer entre miles de objetos distintos con alta precisión. Esto se logra construyendo representaciones vectoriales compactas de las imágenes, que son utilizadas para clasificarlas.

Entendiendo que la información contenida en las imágenes que acompañan a los avisos de venta representan una rica fuente de información, se busca, en el presente trabajo, explorar la oportunidad de utilizarlas, a través de redes convolucionales profundas, para confeccionar modelos de regresión que permitan estimar el precio de venta de casas en el mercado Chileno, utilizando como punto de partida la ciudad de Santiago.

Este documento se encuentra dividido en 8 secciones, agrupadas temáticamente y construyendo cada una en base a la anterior, descritas, brevemente, a continuación:

1. Tecnologías e ideas: Recopilación de herramientas teóricas e ideas detrás del enfoque utilizado para complementar la regresión con la información de las imágenes.
2. Objetivos y consideraciones: Descripción de lo que se persigue y los supuestos que se toman.
3. Creación del *dataset*: Descripción de los datos disponibles, datos obtenidos, estructuración y filtrado de la información y creación de los *splits*.
4. Generación de *deep features*: Descripción concreta respecto a la obtención de *deep features* para el *dataset* generado. Incluye descripción de las arquitecturas involucradas, *hardware* utilizado, consideraciones técnicas y resultados obtenidos.
5. Regresión: Modelos empleados para resolver el problema de regresión de los precios de propiedades y los resultados obtenidos.
6. Modelos adicionales: Descripción de modelos alternativos que se derivan de lo estudiado en la sección anterior.
7. Modelo vs *Baseline*: Inspección del desempeño del mejor modelo y su contraste con el *baseline*.
8. Conclusión

Un lector familiarizado con las redes neuronales convolucionales profundas, métricas de evaluación para problemas de clasificación y regresión y construcción de *datasets* para *machine learning* puede, sin problemas, comenzar desde la segunda sección: Objetivos y consideraciones.

Capítulo 1

Tecnologías e ideas

Frente a la interrogante de qué es un objeto, o de definir qué es un objeto, es difícil no filosofar. Quizás motivados por la necesidad de contar con una respuesta utilizable, tomamos por ciertos una serie de supuestos que, bajo una inspección más profunda, resultan constituir una base endeble. Tenemos, en la mente, una representación de lo que es el objeto y, en el mejor de los casos, una serie de acciones que podemos ejercer al respecto. Sin embargo, nunca experimentamos el resultado concreto de las acciones sobre el objeto, sino, más bien, actualizamos el modelo del objeto en base a nuevas observaciones capturadas luego de que la acción ha sido ejercida. Notamos, entonces, que la representación que tenemos del objeto constituye, para nosotros, casi el objeto mismo en nuestros modelos. Operamos en base a estas representaciones, en base a estos datos que obtenemos de los objetos. Discernir entre, al menos, 2 objetos es una herramienta poderosa, después de todo la lógica binaria ha permitido el desarrollo de potentes tecnologías que han revolucionado los límites de lo que se creía posible.

Es probable que un agente altamente inteligente logre discernir entre 2 objetos distintos con suficiente información disponible, a la vez que un agente con menos capacidad podría lograr lo mismo dada una representación favorable. Esto nos obliga a notar que hay 2 aspectos interesantes en esta idea: interpretar representaciones y representar. Una buena representación facilita la interpretación. Pero es aquí que vale la pena preguntarse: ¿Qué es una buena representación? Y, en un gran despliegue de razonamiento circular, caemos en cuenta que una buena representación es aquella que permite fácilmente discernir entre los objetos. Ciertamente, una representación universal no podría ser sucinta ni sencilla. Por lo que, en un despliegue de pragmatismo, nos convendría buscar buenas representaciones dada la necesidad de discernir entre objetos con un fin en específico. Podemos, entonces, tomar, desde toda la información disponible de los objetos, un subconjunto que, operado y expresado de determinada manera, se presta bien para discernir entre los objetos bajo el objetivo específico. Para determinar el objeto más liviano es suficiente concentrarse en la masa de cada objeto, ignorando sus dimensiones.

Una vez que hemos decidido un objetivo es fructífero, como ya se mencionó, poner cuidado en la representación que le daremos a los objetos involucrados, ya que esto facilitará o dificultará la realización de la tarea. La generación o selección manual de características

significativas es una tarea que consume muchos recursos, ya que requiere un conocimiento del área en la que se está trabajando y el objetivo que se está persiguiendo (*e.g:* escoger los indicadores importantes para describir la potencia de una nación de forma que se pueda determinar la más fuerte). Es natural querer, entonces, diseñar ciertos elementos que, pese a no ser idóneos para toda tarea, se presten bien para representar en varias tareas. En el área de análisis de imágenes se ha perseguido esta idea, desarrollando, con esto en mente, ciertos filtros, considerados de bajo nivel, que pueden ser aplicados a cualquier imagen para obtener información básica de esta. Un ejemplo de esto son los filtros de Sobel, que se pueden aplicar a una imagen para resaltar sus bordes. Los resultados de estos filtros de bajo nivel se pueden componer para generar representaciones vectoriales, más complejas y densas en información, de las imágenes con ciertas propiedades, como invarianza a la escala. Un ejemplo de esto es el descriptor *Sift*[6]. Como era de esperarse, estos descriptores no son lo suficientemente efectivos para todas las tareas de reconocimiento de imágenes, por lo que el siguiente paso lógico es considerar la posibilidad de generar representaciones vectoriales pertinentes a cada tarea de forma automática. Liberando así al humano de la tarea de determinar buenas representaciones y facilitando el trabajo de diferenciar entre representaciones de objetos, ya que estas están diseñadas con el objetivo en mente.

El campo del *machine learning* ha evolucionado mucho persiguiendo la idea de mejorar las representaciones, a la vez que se mejoran los sistemas que aprenden a diferenciar. Particularmente, en el campo del procesamiento de imágenes, el desarrollo de las redes neuronales convolucionales profundas, que pueden interpretarse como grandes aparatos encargados de generar representaciones sucintas y densas en información, ha permitido resolver problemas muy difíciles, siempre que estos puedan representarse, originalmente, en lo que entendemos como una imagen. Se puede representar un conjunto de series de tiempo multi-canal como una imagen[7] y extraer, a través de una red convolucional, una descripción densa que permite representarlas.

Las redes convolucionales logran generar estos *features* específicos para cada tarea de forma automática a través de la agregación sistemática de la información de filtros de bajo nivel (similares en naturaleza al filtro de Sobel) que se aprenden en sus primeras capas. A medida que se va avanzando en la red se van combinando filtros de niveles progresivamente superiores, por lo que la densidad de información de las capas finales, respecto a la tarea específica que se busca resolver, es muy alta.

Planteado lo anterior, con el objetivo de sentar bases más concretas y suficientes para entender la terminología y el razonamiento detrás de las siguientes secciones, se definirán algunos rudimentos concernientes a las tecnologías a utilizar y las ideas a discutir.

1.0.1. Vector de características o *feature vector*

Un vector de características $\vec{X}$ es, simplemente, una representación numérica de algún objeto. Por ejemplo, una fotografía en formato de imagen RGB (que corresponde a un arreglo tres-dimensional) es una representación vectorial del objeto fotografiado. Se podría representar el objeto fotografiado a través de una descripción textual, o de una canción, pero cada

tipo de representación se prestará mejor para describir ciertas características del objeto y, por tanto, para discernir entre objetos de mejor forma en base a distintos objetivos.

1.0.2. Problemas de clasificación y regresión

Dada una representación vectorial n dimensional $\vec{X}$ de un objeto, un problema de clasificación consiste en asignar una etiqueta correspondiente a la clase del objeto, teniendo $\vec{X}$ como entrada. Por otro lado, un problema de regresión busca obtener una cantidad como resultado, por ejemplo, dadas las características de una casa obtener una aproximación de su precio de venta.

1.0.3. Neurona Artificial

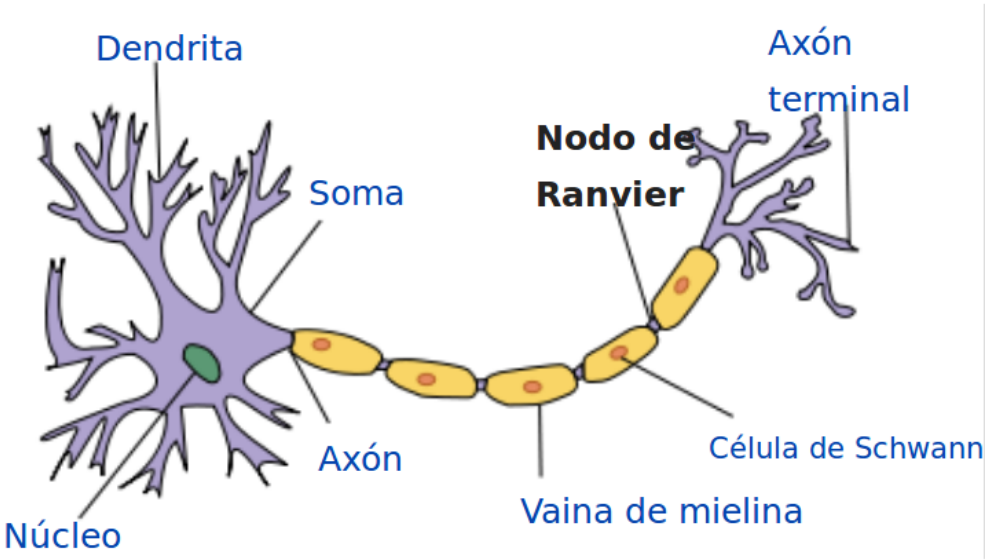


Figura 1.1: Dibujo de una neurona humana.

Una neurona humana (como se aprecia en la Figura 1.1) cuenta con un núcleo y un axón. En el núcleo tiene dendritas, que son pequeño filamentos que le permiten conectarse a los axones de otras neuronas, con el fin de transmitir impulsos eléctricos. Otras neuronas se conectan a su axón utilizando las dendritas de sus respectivos núcleos. Es así que las neuronas forman redes de conexiones. Las conexiones son a través de químicos llamados neurotransmisores, que se liberan desde los extremos del axón hacia las dendritas. Estos impulsos se transmiten desde el núcleo en forma de saltos químicos de un nodo de Ranvier hacia el siguiente. Un punto importante, y quizás clave, del funcionamiento de una neurona es que posee un umbral de estímulo, bajo el que no se produce impulso y sobre el que se produce impulso: Si la neurona no es excitada lo suficiente entonces no se produce respuesta.

Para resumir, la neurona tiene entradas, en las dendritas, que combina para producir una respuesta, siempre y cuando se haya superado el umbral de activación.

El modelo de neurona artificial[8] recoge los elementos clave del funcionamiento de la neurona, en forma matemática.

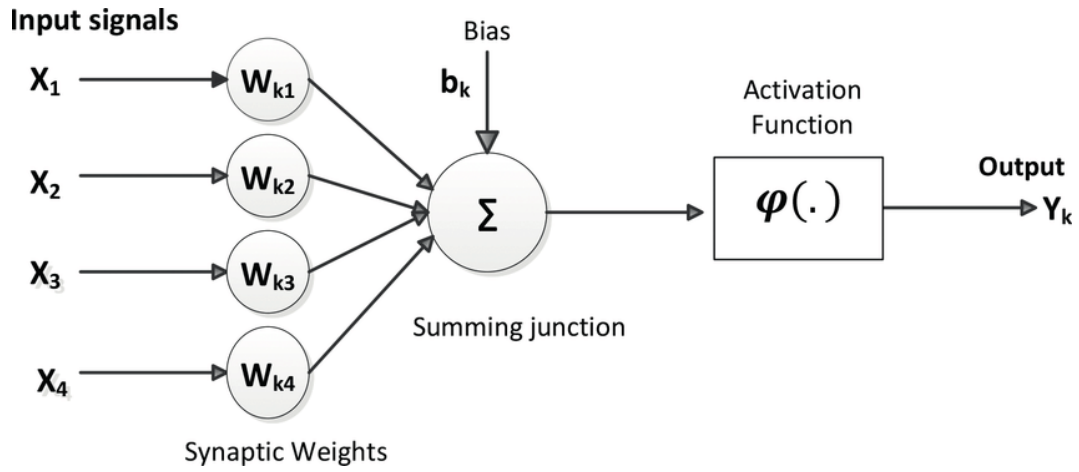


Figura 1.2: Modelo de neurona artificial.

Cuenta con un vector $\vec{X}$ de entrada y un vector $\vec{W}$ de pesos sinápticos. El producto $\vec{W} \cdot \vec{X}$ es la operación clave que realiza la neurona para ponderar los valores de la entrada, y se codifica en el dibujo de la Figura 1.2 con Σ, vale la pena notar que en Σ existe otro término b llamado *bias*, que corresponde al valor base de Σ, incluso en ausencia de estímulo. Este valor b pertenece a $\mathbb{R}$, por lo que puede facilitar o complicar la activación de la neurona. Finalmente, es necesario incorporar al modelo el umbral de activación, lo que se logra a través de la función $\phi()$, también llamada función de activación. Esta función recibe $\vec{W} \cdot \vec{X} + b$ y determina el valor de salida de la neurona. Algunos ejemplos de funciones de activación son:

- Función logística: $f(x) = \frac{1}{1+e^{-x}}$
- Tangente hiperbólica: $f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
- ReLU: $f(x) = max(0, x)$

Con lo anterior tenemos que para una neurona k su valor de salida será $y_k = \phi(\vec{W} \cdot \vec{X} + b)$. El valor de salida de una neurona artificial puede ser alimentado a otra neurona, tal como en el caso de las neuronas humanas, por lo que este modelo permite formar redes de neuronas artificiales. Es así que la salida y_k de una neurona k puede ser un elemento del vector $\vec{X}$ de entrada de otra neurona j, la que tendrá su propio vector de pesos $\vec{W_j}$.

1.0.4. Red neuronal

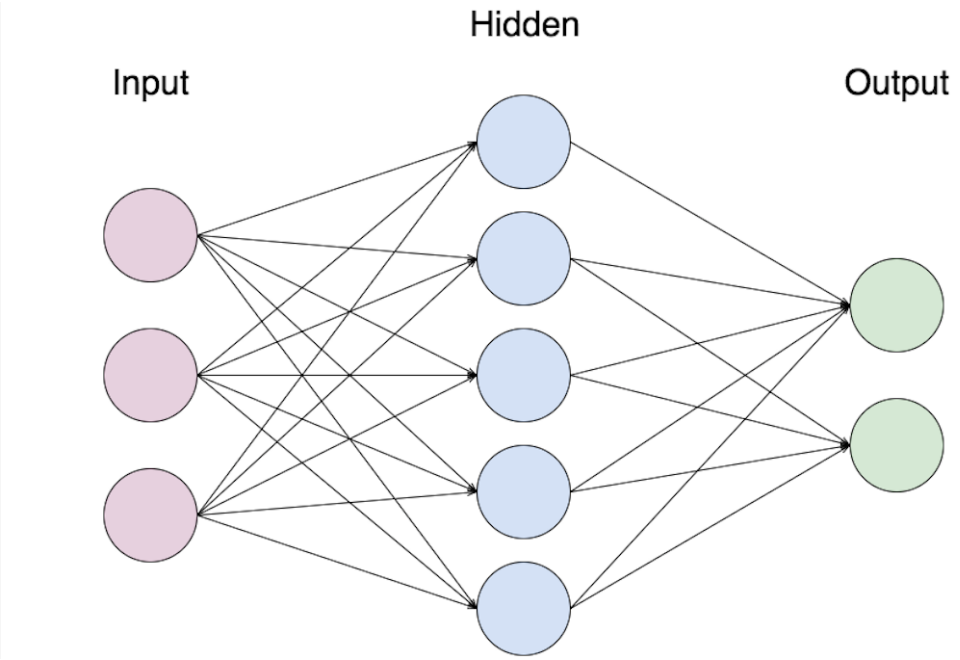


Figura 1.3: Arquitectura de red neuronal *feed-forward fully connected*.

Las neuronas artificiales pueden combinarse para producir redes neuronales artificiales. La forma en que estas neuronas se combinan se denomina arquitectura de la red. En la Figura 1.3 se aprecia un ejemplo de red neuronal artificial. Cada círculo representa una neurona y cada arco representa la alimentación de la salida de una neurona a una neurona que se encuentra más a la derecha. Existen varias arquitecturas de red neuronal artificial, siendo una de las más sencillas de describir la *feedforward*. En esta arquitectura de red, que corresponde a aquella descrita en la Figura 1.3, se pueden ordenar las capas desde la entrada a la salida en una secuencia que va de izquierda a derecha. La capa de entrada de una red con esta arquitectura corresponde a los valores originales de entrada, particularmente corresponde al vector de características $\vec{X}$ que representa una muestra. Otro detalle de la arquitectura de la Figura 1.3 es que todas las neuronas de la capa i se encuentran conectadas a las neuronas de la capa i$+1$, si existe, de esta forma cada neurona correspondiente a la capa *Hidden* recibe el vector $\vec{X}$. Cada una de las neuronas de la capa de entrada (*Input*) alimenta una característica del vector $\vec{X}$ a todas las neuronas de la capa *Hidden*. Una red neuronal artificial puede tener un número arbitrario de capas ocultas (*Hidden*). Cuando todas las neuronas de una capa se encuentran conectadas a todas las neuronas de la capa siguiente se le llama *fully connected*. En el caso de la Figura 1.3, cada capa está totalmente conectada a la capa siguiente, por lo que la red completa es *fully connected*. Es así que esta arquitectura de red se describe como una red neuronal artificial *feedforward, fully connected* con una capa oculta, con 3 neuronas en la capa de entrada, 5 neuronas en la capa oculta y 2 neuronas en la capa de salida. Cada una de estas neuronas tiene su propio set de pesos sinápticos W y una función de activación $\phi()$. No es necesario que todas tengan la misma función de activación, aunque los modelos comúnmente utilizados suelen tener la misma función de activación para las neuronas de una misma capa.

Una estrategia que suele utilizarse para la capa de salida (*Output*) es que exista una neurona artificial de salida por cada una de las clases posibles del problema de clasificación y que al momento de producir las etiquetas del problema estas se expresen en *one-hot-encoding*, que quiere decir que se utiliza una codificación de n bits para representar n clases, y cuando la muestra corresponde a la clase i entonces el bit i es 1 y el resto 0. Lo anterior permite utilizar una función de activación como *softmax* para que cada neurona artificial de salida aproxime la probabilidad de que la muestra pertenezca a la clase que codifica esa neurona.

Con lo anterior, notamos que una vez que se selecciona una arquitectura para la red neuronal artificial, lo que comprende número de capas, funciones de activación, número de neuronas por capa y conexiones entre neuronas, se pueden modificar los pesos $\{\vec{W}_{\mathrm{i}}\}_{\mathrm{i}}$ para acercar las salidas de la red artificial a las salidas esperadas. Para lograr esto se puede calcular un error de clasificación ε, lo que permite, desde la capa de salida hacia la capa de entrada, calcular un gradiente con el que actualizar los pesos $\{\vec{W}_{\mathrm{i}}\}_{\mathrm{i}}$, utilizando el algoritmo de *backpropagation*[9], que utiliza la regla de la cadena de forma agresiva para determinar el factor de cambio de los pesos de cada neurona artificial. Con estas técnicas es posible ajustar los pesos de las neuronas artificiales a medida que se le alimentan ejemplos a la red, a lo que se le llama aprendizaje de la red.

Las redes neuronales son útiles ya que tienen la capacidad de aproximar funciones arbitrariamente complejas cuando su arquitectura tiene la forma adecuada[10]. Dado lo anterior, partiendo desde una arquitectura con suficiente expresividad como para poder representar una función que resuelva un problema de clasificación o regresión se logra, en la práctica, utilizando *backpropagation*, encontrar pesos sinápticos que emulan, con cierto error, el comportamiento de esta función ideal que resuelve el problema. Si bien, no existe una garantía de alcanzar los valores ideales en los pesos sinápticos utilizando *backpropagation*, en la práctica el error alcanzado en las tareas de clasificación y regresión es lo suficientemente bajo como para utilizar las redes en aplicaciones.

Debido a que el entrenamiento de las redes suele basarse en la minimización o maximización de una cantidad (comúnmente el error ε) se transforma en un problema de optimización, que es susceptible al problema de los extremos locales, dado su método de convergencia basado en gradientes: No alcanzar el estado óptimo por quedarse atrapado cerca de un mínimo o un máximo local.

1.0.5. Capa convolucional

En una capa *fully connected* cada una de las neuronas que la componen recibe la totalidad de la salida de la capa anterior. A la entrada de cada neurona se le llama, también, campo receptivo. Con esto, en una capa *fully connected* el campo receptivo de cada neurona es la salida completa de la capa anterior (la colección de todas las salidas de las neuronas de la capa anterior, asumiendo que la red es *feedforward*). Perfectamente podría darse que, a través del entrenamiento, algunas de estas entradas terminen siendo multiplicadas por un peso sináptico de valor 0, efectivamente cortando la conexión entre esas neuronas, pero, obviamente, sería

computacionalmente más barato eliminar conexiones desde el comienzo. Resulta ser, además, que para algunas tareas es más benéfico quitar conexiones (además de más barato). Para el procesamiento de imágenes se popularizó[11] un esquema de conexiones en que una neurona tiene un campo receptivo de $n \cdot m$ elementos que se va deslizando por toda la imagen. Los nm pesos que aprende la neurona, a los que se les suele llamar *kernel*, se aplican a toda la imagen, y los valores de salida de la neurona producto de su aplicación se compilan en una nueva imagen llamada *feature map*, se dice que esta neurona actua como un filtro sobre la imagen de entrada.

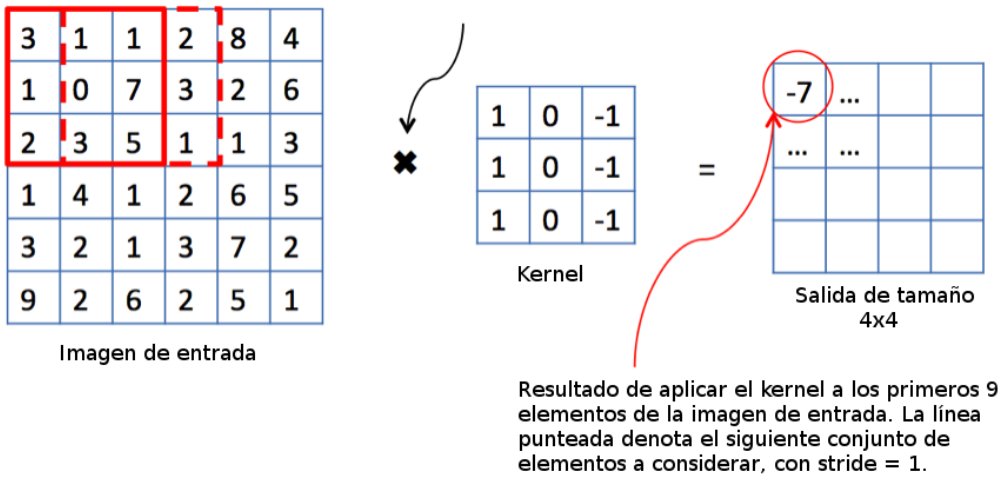


Figura 1.4: Visualización de operaciones de una neurona en una capa convolucional.

En la Figura 1.4 se muestra un ejemplo de la aplicación del *kernel* de tamaño $3 \cdot 3$ de una neurona a una imagen de entrada de 1 canal (o 1 color). Otro punto relevante es que la ventana de aplicación sobre la imagen de entrada puede moverse en distintos rangos. En la Figura 1.4 esta ventana se mueve una distancia 1 para obtener la siguiente ventana. A este valor del salto o del paso se le llama *stride*.

Una capa convolucional, como suele utilizarse, queda definida por el número de neuronas (lo que se conoce como número de filtros), el valor del *stride* y la función de activación que utilizarán las neuronas. Notamos que, pese a que, en general, la dimensionalidad de los *feature maps* producidos es menor a la dimensionalidad de la imagen original, las capas convolucionales suelen tener varias neuronas (o filtros), por lo que la dimensionalidad de salida de la capa completa suele aumentar, aunque lo anterior no implica que no se vaya perdiendo información. Se espera, sin embargo, que la información que se va perdiendo, o representando de formas menos favorables para su interpretación, sea menos útil para el problema que se busca resolver, en comparación a la información que resulta ser expresada de forma más directa luego de la aplicación de los filtros (*e.g:* Filtros de Sobel para el problema detección de bordes).

1.0.6. Capa de *pooling*

Una capa de *pooling* aplica una función a un campo receptivo de tamaño $n \cdot m$, con un *stride* determinado. Las funciones típicamente usadas son máximo, promedio y mínimo, pero, en principio, puede ser cualquier función aplicada a los nm elementos del campo receptivo.

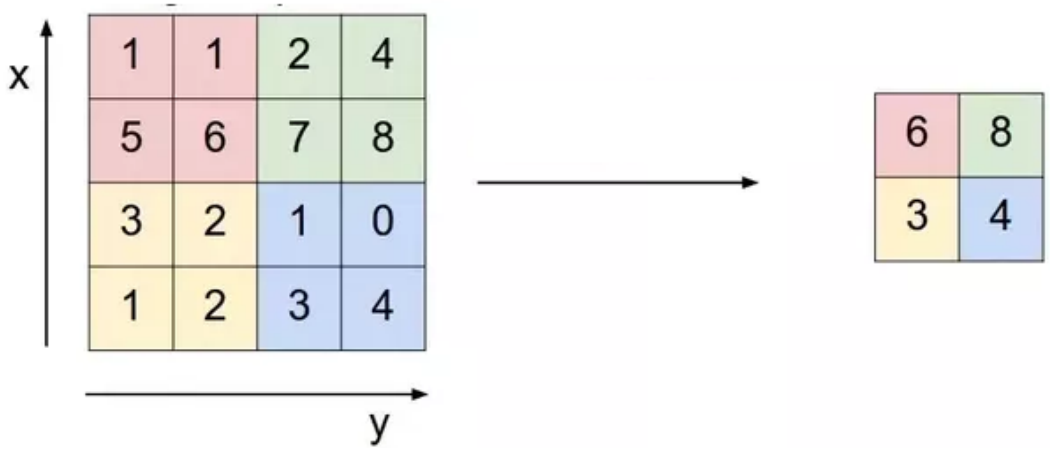


Figura 1.5: Ejemplo de aplicación de máx *pooling* con un campo receptivo de $2 \cdot 2$ y *stride* 2.

1.0.7. Red convolucional

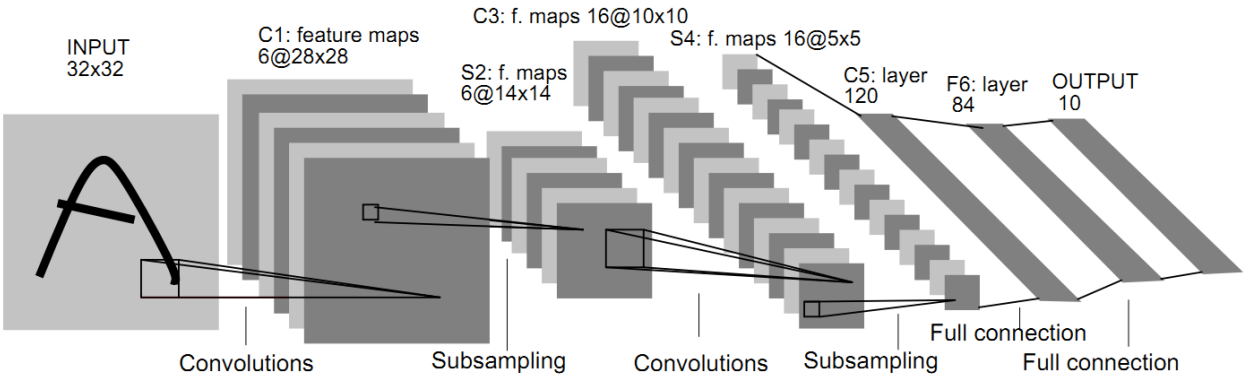


Figura 1.6: Red convolucional LeNet 5 para resolver el problema de clasificación de dígitos manuscritos.

Una red convolucional es un tipo de arquitectura de red neuronal artificial que utiliza capas convolucionales en su estructura. En la Figura 1.6 se aprecia un ejemplo[11] de red convolucional en que la primera capa es una capa convolucional, seguida de una capa de *pooling*, otra capa convolucional, otra capa de *pooling* y, finalmente, 3 capas *fully connected*.

Para entender esta estructura notamos que la arquitectura de la Figura 1.6 busca resolver un problema de clasificación de dígitos manuscritos: Dado un *feature vector* $\vec{X}$ se debe predecir el dígito manuscrito representado por $\vec{X}$. Al momento del entrenamiento se le proporciona a la red las imágenes de tamaño $32 \cdot 32$ que representan a los dígitos y una codificación *one-hot* de su clase. Por ejemplo, si la clase del dígito es 2 entonces su codificación *one-hot* será 0010000000, por eso la capa de salida tiene tamaño 10. Ahora bien, notamos una segunda idea más importante: Lo que hace la primera sección de la red es transformar una representación vectorial del dígito (la imagen de $32 \cdot 32$) a otra representación vectorial del dígito (la concatenación de la salida de la última capa de *pooling*, o la salida de la primera capa *fully connected*). Lo importante aquí es que la última sección de capas *fully connected* corresponden al clasificador que utiliza los nuevos *feature vectors* para resolver el problema de clasificación original. Esta idea es relevante ya que, en principio, se podría utilizar los nuevos *feature vectors* para entrenar otro clasificador distinto para resolver el problema. Esto se suele hacer bastante en la práctica.

1.0.8. Redes residuales

Debido a los problemas de convergencia presentados por redes convolucionales demasiado profundas fue necesario[12] idear un mecanismo que facilitara su entrenamiento, por lo que se generó el bloque residual, que corresponde a sumar a la salida de una capa los valores de salida de una capa anterior. Las redes se construyen como secuencias de bloques residuales.

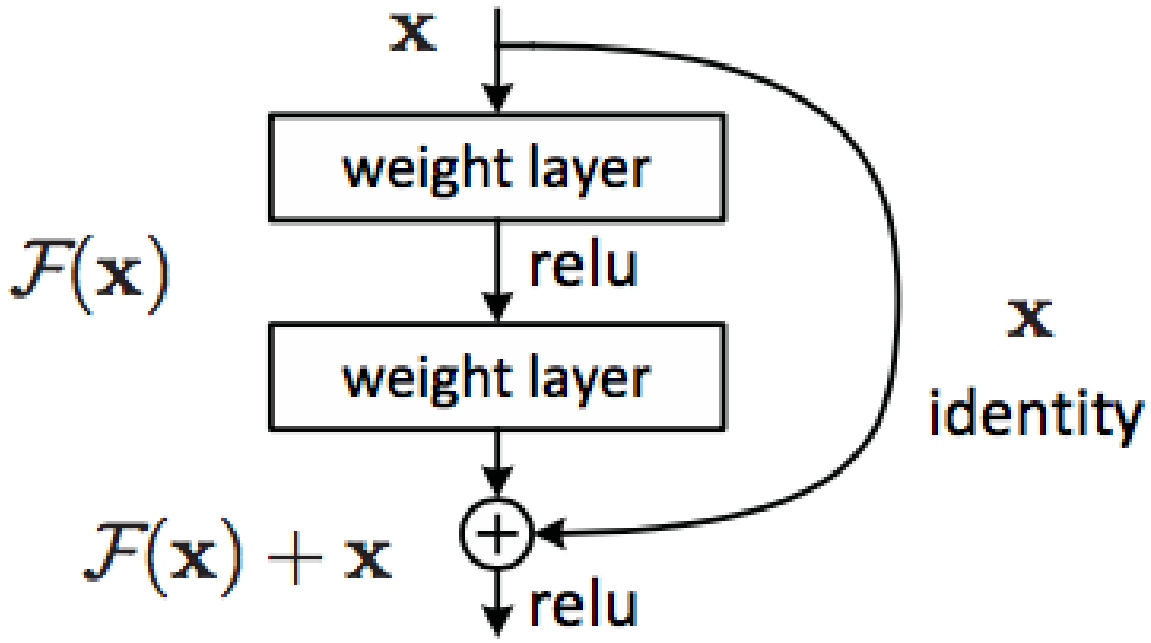


Figura 1.7: Componente fundamental de una red residual: bloque residual.

1.0.9. Transfer learning

Cuando una red neuronal es entrenada para resolver un problema (aproximar la función que entrega las salidas útiles para la aplicación en mente) puede pasar que, desde ese estado de valores de los pesos sinápticos aprendidos, sea más fácil entrenar la red para resolver otro problema, ya sea porque se entrena desde ese punto la red completa utilizando las muestras del nuevo problema o porque se entrena una porción de la red (normalmente las últimas capas, dejando los pesos del resto estáticos) para escoger, desde los *features* generados para el primer problema, un subconjunto o combinación no-lineal que se presta mejor para el nuevo problema.

Una competencia de clasificación de imágenes sobre el dataset Imagenet[13] ha propiciado la generación de varias arquitecturas de redes convolucionales profundas para su resolución[14][15][12][16][17][5], siendo interesante el hecho de que estas arquitecturas entrenadas para resolver este problema se han utilizado como punto de partida para realizar *transfer learning*[18] hacia otros problemas que se benefician de los filtros que permiten clasificar objetos.

También se le denomina *transfer learning* a la estrategia de utilizar los *features* generados por una red entrenada para resolver un problema en la resolución de otro problema, sin necesidad de entrenar nuevamente ninguna porción de la red. Para esto, simplemente se extraen *features* generados en alguna capa, comúnmente cercanas al final de la red, en base a los elementos del nuevo problema. Estos *features* son, luego, utilizados para entrenar nuevos modelos.

Al proceso de re-entrenar una red desde un conjunto de pesos se le denomina *fine-tuning*.

1.0.10. Accuracy

Accuracy es una métrica que puede ser utilizada para cuantificar la eficacia de un sistema de clasificación (que resuelve un problema de clasificación). Para una clasificación de N muestras, en que se clasificaron correctamente n muestras, se puede calcular el *accuracy* como $\frac{n}{N}$.

1.0.11. *RMSE*

RMSE o *Root mean square error* es una métrica que se puede utilizar para cuantificar el error entre 2 cantidades o conjuntos de cantidades. Aplicado a un problema de regresión en que y_i corresponde a uno de los valores a aproximar y $\hat{y}_i$ corresponde a la aproximación para dicho valor, entonces el *RMSE* de todo el conjunto de N elementos corresponde a $\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$.

1.0.12. R^2

El coeficiente de determinación, también llamado R^2, se define $1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$ y corresponde a una medida de qué tan bueno es un modelo prediciendo resultados, por lo que puede ser utilizado para medir el desempeño de un sistema para resolver un problema de regresión, siendo un valor de 1 una solución óptima sobre el conjunto de datos conocidos.

1.0.13. *Train/test split*

Comúnmente, cuando se entrenan clasificadores (sistemas que resuelven problemas de clasificación) o regresores (sistemas que resuelven problemas de regresión) no se utiliza la totalidad de los datos para crear el sistema, ya que el objetivo suele ser la clasificación o regresión de elementos cuya clase o valor asociado se desconoce: Realizar predicciones en base al sistema construido. Para intentar tener una medida del poder de generalización del sistema (o, dicho de otra forma, qué tan bien logra predecir para nuevas muestras) se dividen los datos disponibles en un conjunto de entrenamiento (*train set*) y un conjunto de prueba (*test set*), ejecutando todas las operaciones necesarias para confeccionar el modelo sobre el *train set* y todas las pruebas y evaluaciones con métricas sobre el *test set*, que no fue utilizado para la construcción. En el caso particular de las redes neuronales se suele sub-dividir el *train set* en un *train set* y un *validation set*, siendo este último utilizado para pulir ciertos parámetros de la arquitectura. Esto se hace para evitar sobreajustar el modelo a los datos del conjunto de prueba, de modo que sea una buena aproximación frente al comportamiento que tendrá el modelo al operar sobre datos nuevos.

1.0.14. Imágenes satelitales o de vuelo aéreo

Son imágenes tomadas por cámaras de satélites o de aeronaves a alta altura. Para efectos del presente trabajo consideraremos las imágenes disponibles en Google Maps, cuyos niveles de *zoom* van de 1 a 21. Estas imágenes se encuentran codificadas en formato RGB, por lo que constituyen arreglos 3-dimensionales.

1.0.15. Imágenes *street view*

Una imagen *street view* es una imagen a nivel de calle, o de fachada, de un lugar. Nuevamente, consideraremos como *street view* las imágenes *street view* de Google Maps. Estas imágenes también se encuentran codificadas en formato RGB y son, por tanto, arreglos 3-dimensionales. Esto puede interpretarse como que cada elemento de entrada es una colección de 3 *feature maps*.

Figura 1.8: Imagen satelital, zoom 16.

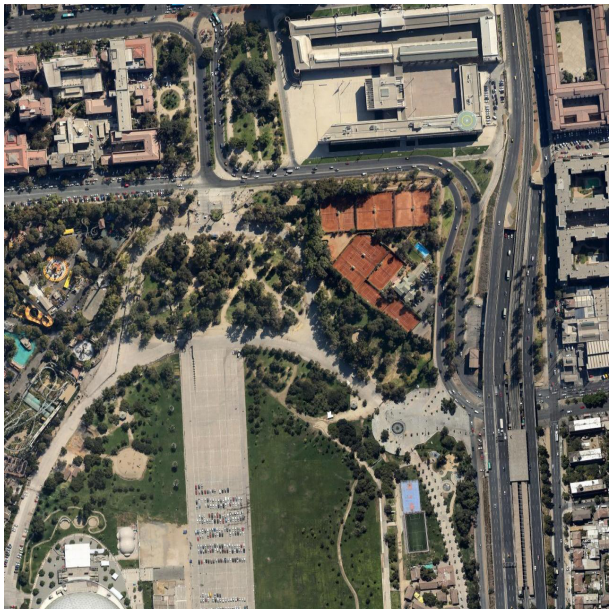
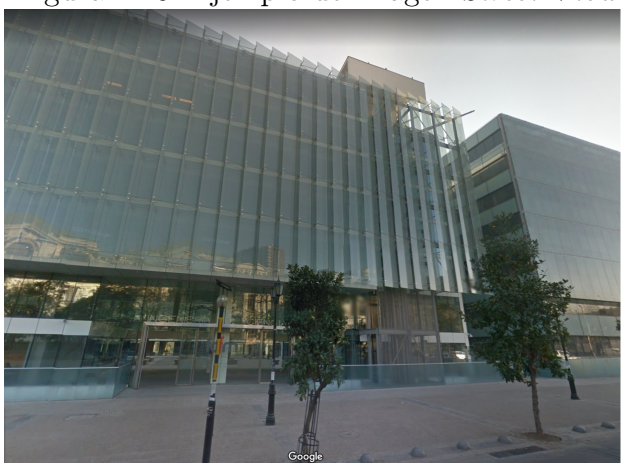


Figura 1.9: Imagen satelital, zoom 18.

Figura 1.10: Ejemplo de imagen *Street View*



1.1. Información de las imágenes: Enfoques populares

Tras consultar las opciones disponibles, en la literatura, para abordar el problema de regresión de precios de propiedades utilizando imágenes, se concluyó que las 2 corrientes con más tracción corresponden a utilizar imágenes satelitales (o de vista aérea) o imágenes de tipo *street view*.

1.1.1. Imágenes Satelitales

La primera idea busca utilizar la información contenida en una imagen satelital (o aérea) de la propiedad y de su vecindario para construir un *feature vector* que permita entrenar un regresor que estime el precio. Para utilizar exitosamente las imágenes satelitales es necesario considerar distintos niveles de *zoom* en el entorno de la propiedad. Para ver por qué esto es útil: tiende a darse que los vecindarios en que se ubican las propiedades más caras tienen mejor acceso a áreas verdes y masas de agua, existiendo, además, mayor espacio entre las propiedades mismas, por lo que, para un observador humano, es posible diferenciar un barrio de propiedades baratas y un barrio de propiedades caras. Para lograr combinar la información de los distintos niveles de *zoom* es necesario entrenar una red convolucional profunda para extraer *features* en cada uno de los niveles (*i.e:* Si se utilizan 4 niveles de *zoom* se deben entrenar 4 redes convolucionales profundas distintas) y luego agregar esta información para producir la predicción. Se toma como principal referencia [19] un trabajo en que se utiliza *transfer learning* sobre una red convolucional Inception v3[14], la que tiene buen desempeño en el problema de clasificación de *Imagenet*[13]. Se toma un modelo Inception v3 pre-entrenado para resolver el problema de clasificación de *Imagenet* y se le ajusta para resolver el problema de clasificación de propiedades caras y propiedades baratas (10 % superior y 10 % inferior, en precio, respectivamente). Los autores utilizan un *dataset* de 43037 muestras de propiedades de Londres (90/10 train/test split), correspondientes a imágenes satelitales obtenidas de Google Maps, utilizando niveles de zoom 15, 16, 17, 18, 19 y 20. Con este nuevo adiestramiento se producen *features* que son agregados para ser utilizados como información de entrada para 3 regresores (o estimadores): Regresor lineal, *Random Forest*[20] y *MLP* (*Multi-layer perceptron o red neuronal feedforward fully connected de varias capas*).

1.1.2. Imágenes estilo *Street View*

El otro enfoque es generar los *features* sobre imágenes estilo *street view* para resolver el mismo problema de predicción. En este trabajo[21] los autores proponen transformar el problema en un problema secuencial, para poder utilizar una arquitectura de red recurrente como LSTM[22] para la regresión. Para representar una propiedad se crea una secuencia (que parte con la propiedad en cuestión) de propiedades aledañas, escogidas con probabilidad inversamente proporcional a la distancia que tienen con la propiedad en cuestión. Cada una de las propiedades se encuentra representada por el *feature vector* producido por la red GoogleNet (entrenada para resolver la clasificación de Imagenet) en base a la imagen (o imágenes, en cuyo caso se usa *average pooling* para agregar la información) *street view* de la

propiedad. Vale la pena notar que, en esta formulación, una propiedad cuenta con muchas representaciones válidas y que no se realiza *transfer learning*. Se entrena la red recurrente LSTM alimentándole estas secuencias de *features* para producir las regresiones.

1.1.3. Discusión

Ambos enfoques se complementan con la información adicional disponible de la propiedad, como número de baños, superficie, etc. En el corazón de ambos enfoques se encuentra la construcción de *features* en base a redes convolucionales profundas, por lo que resulta natural explorar arquitecturas más modernas que han tenido éxito en resolver el problema de clasificación de Imagenet. Otro componente que se puede agregar al enfoque basado en *street view* (dada su transformación del problema en secuencias) es el *attention* con *sentinel gate*[23]. Un supuesto es que la información contenida en las imágenes satelitales de mayor *zoom* (en que se incluye casi únicamente la propiedad) y las imágenes *street view* tienen un grado de redundancia que no llega a ser total, por lo que debe existir una forma de complementar ambas fuentes.

De ambos enfoques pareciera que el de imágenes satelitales o vuelo aéreo es más prometedor, principalmente por el grado de plasticidad mostrado por los autores de [19], quienes lograron utilizar las redes entrenadas sobre datos de Londres en muestras de Birmingham y Liverpool, con una degradación baja de los resultados. Lo anterior, junto a la uniformidad de las imágenes satelitales y la clara determinación de la cantidad de imágenes necesarias por cada muestra, plantan esta estrategia como la opción preferible.

Capítulo 2

Objetivos y consideraciones

El objetivo principal es crear un modelo de regresión que permita predecir el precio de venta de una casa, en la ciudad de Santiago, contando con la siguiente información: Número de habitaciones, número de baños, superficie construida, superficie del terreno, latitud y longitud. Esto se da ya que esta información existe, actualmente, en internet, a través de los avisos de venta de propiedades, en los que se indica el precio, por lo que es posible construir un *dataset*.

Contar con la información de latitud y longitud hace posible obtener imágenes que describen la propiedad y su entorno, tales como las imágenes satelitales o imágenes *street view*. Un supuesto que se adopta es que estas imágenes pueden resultar ricas en información que puede ser útil para la regresión del precio, en particular la información del barrio en que se encuentra emplazada la propiedad, ya que el precio de una casa, en Santiago, tiende a variar fuertemente dependiendo de su ubicación dentro de la ciudad. Es posible para un humano notar ciertas características que permiten distinguir si un barrio tiene casas de alto o bajo valor, y, ciertamente, es posible para un humano distinguir, a un alto nivel de *zoom*, el tamaño y calidad del techo de una casa y los elementos que hay en su patio (*e.g:* si hay piscina o no). Un conjunto de imágenes satelitales de una casa, a distintos niveles de *zoom*, le entregan a un humano información adicional para estimar el valor de venta.

El siguiente supuesto es, entonces, que se puede capturar parte de esta información útil de las imágenes satelitales para ser usada en la regresión de precio de venta de casas. Para capturar esta información se asume que una red convolucional profunda puede producir *features* significativos para el problema de regresión. Basándose principalmente en un trabajo anterior[19], es que se decide hacer *fine tuning* a redes convolucionales profundas, pre-entrenadas para resolver *Imagenet*, con el objetivo de resolver un problema de clasificación que consiste en distinguir si las casas son baratas o caras. De acuerdo al trabajo anterior citado, los *features* que se extraen desde estas redes re-entrenadas mejoran el desempeño de regresores como *Random Forest*.

Se determina que los principales pasos a seguir para conseguir el objetivo son los siguientes:

1. Confeccionar un *dataset* que cuente, para cada muestra, con imágenes satelitales a 7

niveles distintos de *zoom*.

2. Hacer *fine-tuning* de redes convolucionales profundas, pre-entrenadas para resolver *Imagenet*, de forma que puedan identificar si una imagen satelital está centrada en una propiedad barata o una cara.
3. Extraer *deep features* desde las redes re-entrenadas que describan las imágenes satelitales centradas en las propiedades del *dataset*.
4. Entrenar un regresor que utilice los *features* generados por las redes junto con los *House Attributes* (*i.e:* Número de baños, número de habitaciones, superficie construida, superficie del terreno, latitud y longitud) para predecir los precios de venta de las propiedades.
5. Evaluar el desempeño de las redes convolucionales profundas re-entrenadas frente al problema de clasificación de casas, utilizando *accuracy* como la principal métrica.
6. Evaluar el desempeño del regresor frente al problema de predicción de precios de venta de propiedades, utilizando *RMSE* y R^2 como principales métricas.
7. Evaluar el estado de la solución, considerar nuevas posibilidades e iterar.

Capítulo 3

Creación del *dataset*

Los datos utilizados corresponden a una recopilación de anuncios de venta, mayoritariamente de casas, dentro de la ciudad de Santiago, desde varios portales inmobiliarios dedicados al mercado chileno. Los datos originales corresponden a 173640 anuncios de las comunas Estación Central, La Florida, Las Condes, Maipú, Puente Alto y Santiago, con distintos grados de información. Dado que el objetivo de la construcción de este *dataset* es el entrenamiento de un modelo de regresión que permita estimar precios de venta de casas y que, para la construcción de este modelo, se utilizan *deep features* generados en base a imágenes satelitales de las propiedades, se determinó que los atributos necesarios para generar una muestra del *dataset* son los siguientes: (id, habitaciones, baños, construidos, terreno, latitud, longitud, fecha, comuna, precio). Siendo:

- id: Número único que identifica la muestra en el *dataset*.
- habitaciones: Número de habitaciones disponibles en la propiedad.
- baños: Número de baños disponibles en la propiedad.
- construidos: Número de m^2 construidos.
- terreno: Número de m^2 de terreno.
- latitud: Latitud expresada en 1 número.
- longitud: Longitud expresada en 1 número.
- fecha: Fecha de publicación del anuncio, en formato año/mes/día.
- comuna: Comuna a la que corresponde el anuncio.

De los 173640 anuncios originales es posible construir 72164 muestras con toda la información necesaria. Debido a ciertas particularidades de las muestras generadas es necesario refinar esta selección.

3.1. Consistencia de información

Al estudiar la distribución de los valores de los atributos de las muestras, y los contenidos de los anuncios, se hacen evidentes ciertos detalles frente a los que es necesario tomar acciones:

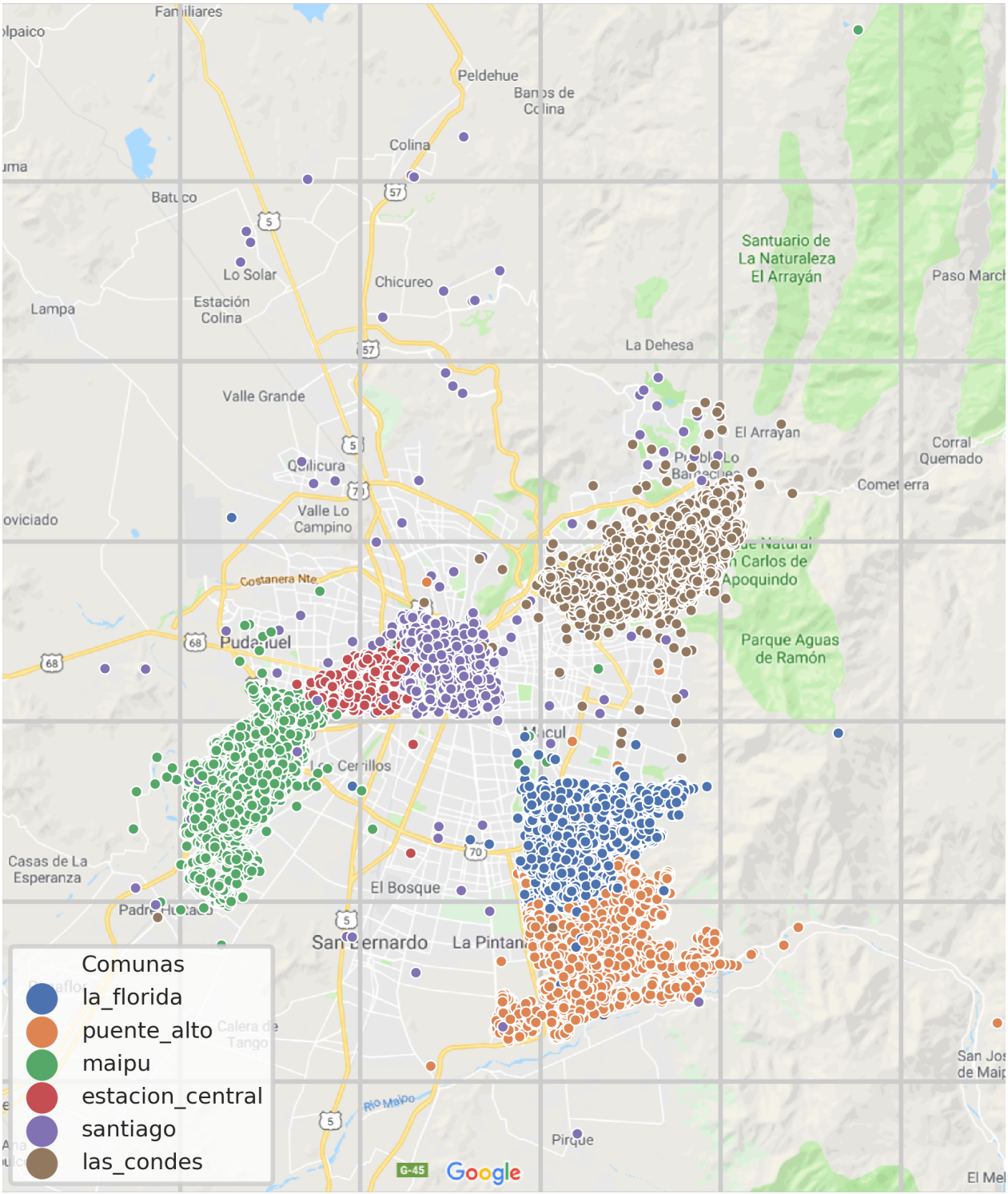


Figura 3.1: Muestras sin filtrar.

- Corregir, de ser posible, muestras que presentan una discrepancia entre m^2 construidos y m^2 de terreno (*e.g:* 200 m^2 construidos en un terreno de 1 m^2).
- Corregir, de ser posible, precios muy distintos al tipo de propiedad en su sector. Notando que esto puede deberse, entre otras cosas, a que el precio está expresado en U.F. o en alguna moneda internacional (*e.g:* USD, EUR.).
- Eliminar muestras que corresponden a anuncios de arriendo, con precios mensuales o anuales, lo que dificulta su separación de la venta de propiedades más baratas, particularmente en el caso de arriendo de propiedades caras. Estos anuncios pueden ser conservados en caso de que se cuente con un precio de venta.
- Eliminar muestras que corresponden a ofertas de otros servicios como construcción de casas o financiamiento de créditos hipotecarios.
- Eliminar muestras que corresponden a remates, subastas o ventas por apuro. Se considera que esto ensucia la predicción de los precios de venta.
- Eliminar muestras que corresponden a ofertas de compra de casas.
- Eliminar muestras que corresponden a venta de promesas de compra.
- Corregir, de ser posible, los *outliers* en cada uno de los atributos.

Para regularizar estas situaciones, se consulta nuevamente la información del anuncio original y, de ser insuficiente, se consulta la información de los 173640 anuncios originales con el fin de identificar anuncios incompletos correspondientes a la misma oferta, ya que, pese a que se excluyen, es posible combinar varios anuncios insuficientes para corregir un candidato a anuncio suficiente. En caso de que lo anterior falle, se procede a obtener información nueva desde internet o, agotando todas las instancias, a eliminar la muestra.

3.2. Filtrado de muestras

Para reducir la cantidad de muestras problemáticas (*e.g:* Las muestras al norte del Santuario El Arrayán y sur del Parque Aguas de Ramón) se filtran las muestras de cada comuna en base a los límites de cada una, de forma que las muestras que aporta cada comuna están contenidas en las coordenadas de la comuna.

Los anuncios originales fueron recopilados desde varios portales inmobiliarios en internet, por lo que existen, en las muestras generadas desde los anuncios, grupos correspondientes al mismo anuncio. Además, existen muestras que, pese a no corresponder al mismo anuncio, cuentan con la misma longitud y latitud. Esto es problemático desde el punto de vista de la regresión, ya que se asignan valores distintos a un mismo conjunto de atributos. Para solucionar estos problemas se establecen 2 relaciones de equivalencia que, aplicadas de forma sucesiva, permiten extraer representantes útiles para perseguir el objetivo.

Diremos que la relación de equivalencia R entre 2 muestras a y b se cumple cuando a y b comparten los siguientes atributos: habitaciones, baños, construidos, terreno, latitud, longitud. En base a esto se escoge un representante para cada clase de equivalencia tomando la muestra con la fecha de publicación más reciente.

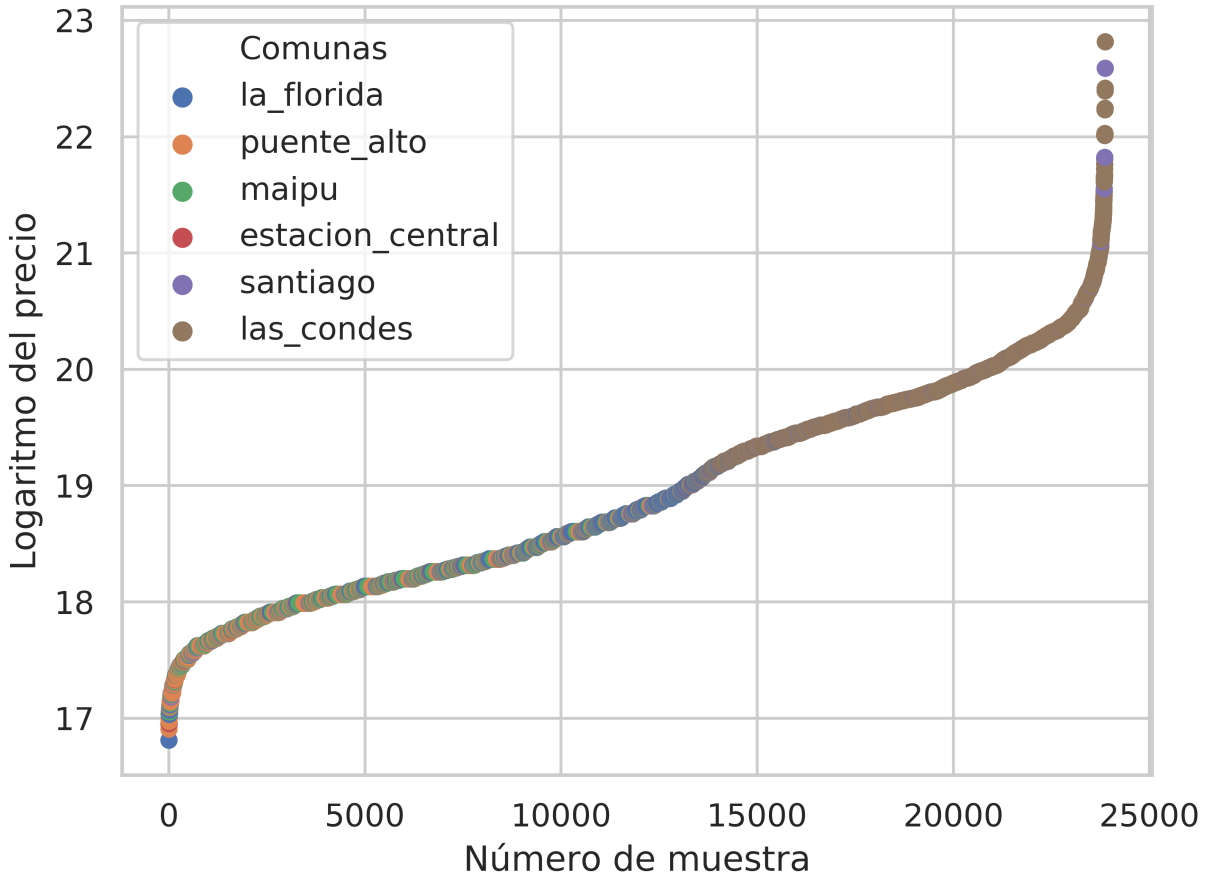


Figura 3.2: Logaritmo de precio de muestras filtradas.

Es contraproducente, al considerar los *deep features*, tener muestras con las mismas imágenes y distintos atributos, por lo que se considera una segunda relación de equivalencia R' entre muestras a' y b' que se cumple cuando a' y b' cuentan con la misma latitud y longitud. Esta nueva relación crea clases de equivalencia en el conjunto de los representantes de las clases de equivalencia de R, desde las que se escogen los representantes de las clases de equivalencia de R' de forma aleatoria.

De este proceso resultan 23877 muestras válidas.

3.3. Imágenes satelitales

El *dataset* contempla una colección de imágenes satelitales con distintos niveles de *zoom* para cada una de las muestras válidas. Uno de los supuestos clave, realizados al momento de decidir utilizar imágenes satelitales en el entrenamiento de los modelos de regresión, es que

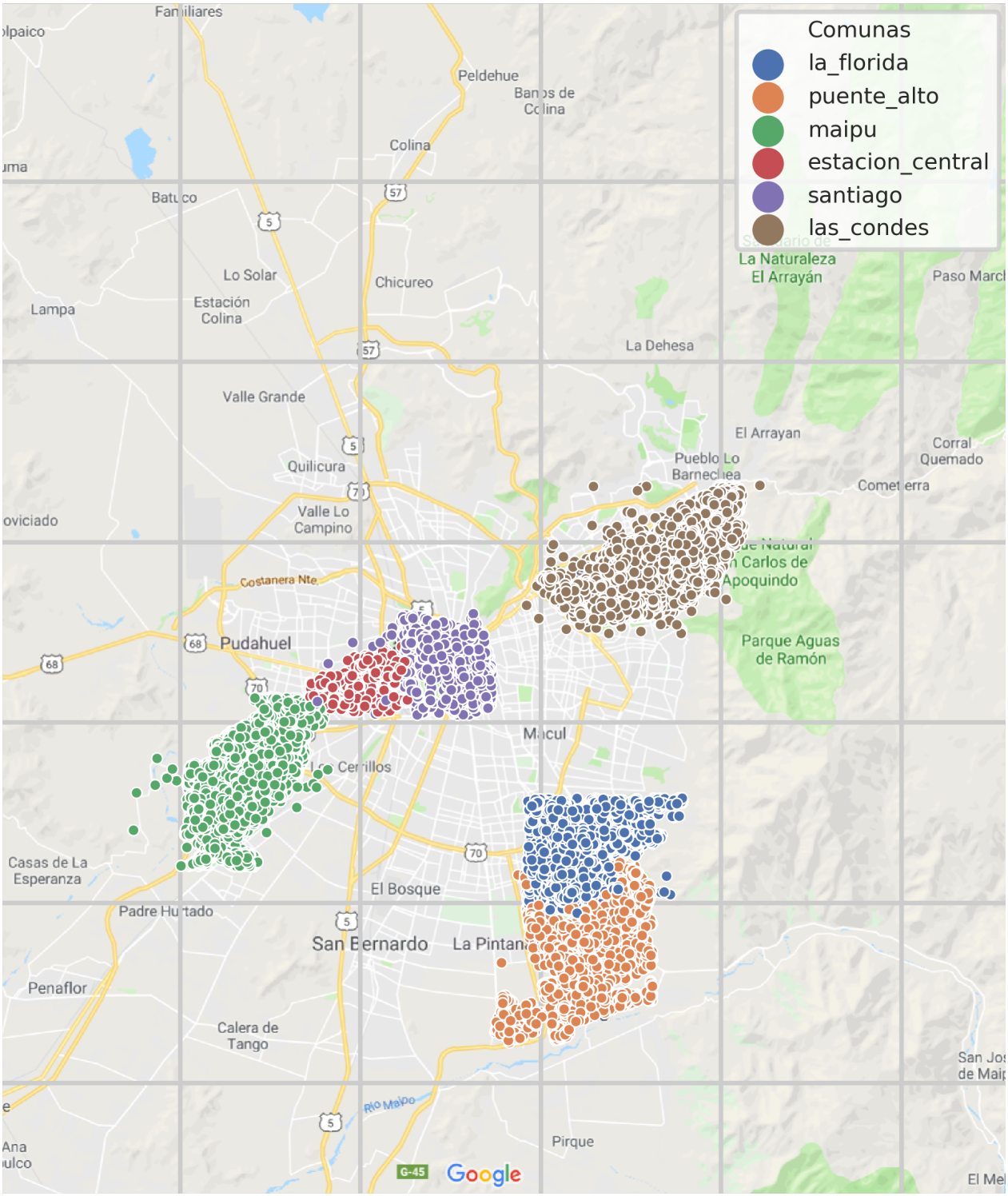


Figura 3.3: Muestras filtradas.

a menor nivel de *zoom* se captura información del barrio en que se encuentra la propiedad, y a mayor nivel de *zoom* se describen aspectos específicos de la propiedad misma tales como calidad de los materiales del techo, la presencia de áreas verdes o de masas de agua, por nombrar algunos.

Figura 3.4: *Zoom* 15 para muestra 1543999.

Figura 3.5: *Zoom* 16 para muestra 1543999.

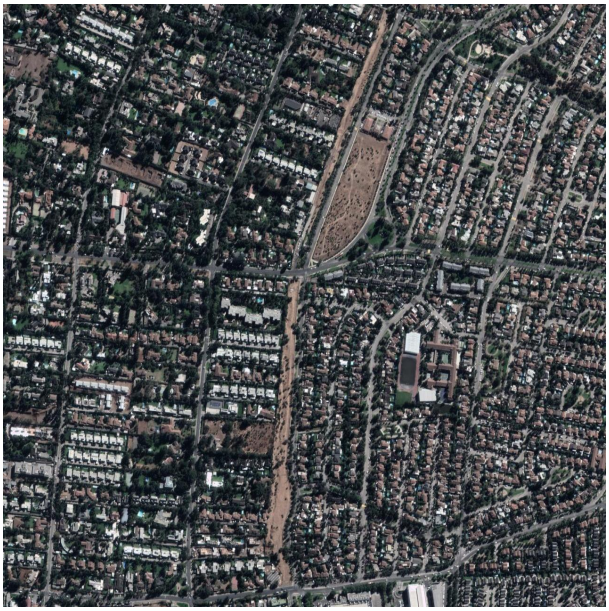


Figura 3.6: *Zoom* 17 para muestra 1543999.

Figura 3.7: *Zoom* 18 para muestra 1543999.

La resolución de cada una de estas imágenes es de 1280 x 1280 píxeles, contando con un nivel de detalle muy superior a las imágenes de 299 x 299 píxeles utilizadas normalmente en los modelos de redes convolucionales entrenados para resolver Imagenet, esto con el fin de ayudar a la generación de *features* con información significativa.

Figura 3.8: *Zoom* 19 para muestra 1543999.

Figura 3.9: *Zoom* 20 para muestra 1543999.

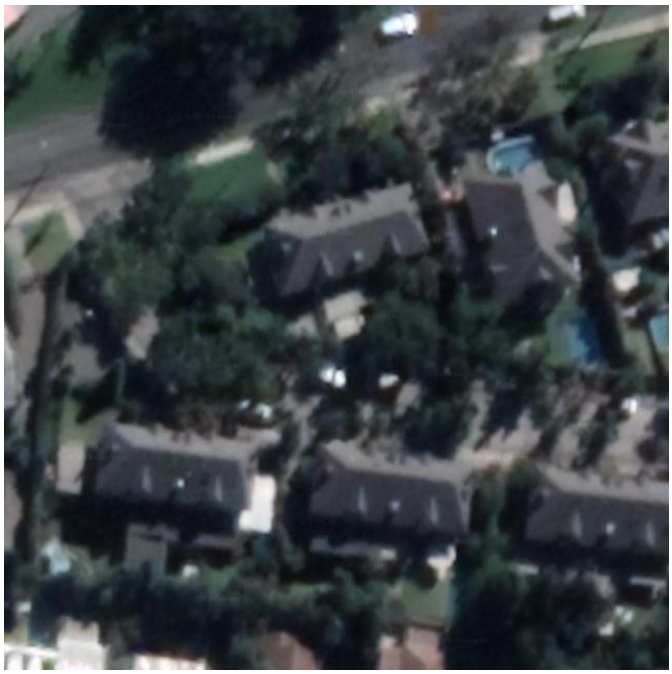


Figura 3.10: *Zoom* 21 para muestra 1543999.

3.4. *Splits*

Para construir y probar los modelos se separa el *dataset* de 23877 muestras utilizando 3 esquemas.

3.4.1. Clasificación binaria de casas

Se separa el *dataset* en 8000 muestras de entrenamiento y 4000 muestras para validación y *test*. Para hacer esto se utilizan muestras de 2 tipos: Las muestras más baratas y las muestras más caras. Las 4000 muestras más baratas y las 4000 muestras más caras conforman los datos de entrenamiento. Los datos de validación y prueba constan de 2000 muestras cada uno, siendo las 2000 muestras de validación las siguientes 1000 menos baratas y las siguientes 1000 menos caras (*i.e:* Si, al ordenar las muestras en base al precio, las muestras 0 a 3999 son las 4000 muestras más baratas, entonces las muestras 4000 a 4999, serán las 1000 muestras baratas del conjunto de validación y las muestras 5000 a 5999 serán las muestras baratas del conjunto de prueba) y las siguientes 1000 menos baratas y 1000 menos caras conformarán el conjunto de prueba.

3.4.2. Clasificación con más clases de casas

Considerando que los *features* generados para resolver la clasificación binaria de casas caras y baratas podrían resultar sobreajustados a este objetivo, resultando poco útiles como datos de entrada para resolver el problema de regresión de precios, se considera dividir la totalidad del *dataset* en una mayor cantidad de clases, aumentando la granularidad del problema de clasificación.

Uno de los principales cuidados que se tiene al momento de realizar estas divisiones es que estas nuevas clases se encuentren adecuadamente representadas, contando con un número de muestras similar. Estableciendo límites de precios para cada categoría, de forma que estas clases sean balanceadas, se llega a que las divisiones óptimas, en términos del balance y de aumentar la granularidad del problema, son 6, 11 y 22 clases. Se divide el *dataset* de forma balanceada en 6 clases, dejando un 10 % de las muestras para validación y un 10 % para prueba. Esto corresponde a 19101 muestras de entrenamiento, 2388 muestras de validación y 2388 muestras de prueba. Esta divisón es fija.

3.4.3. *Dataset* de regresión

Aprovechando la flexibilidad esperada de esta etapa, se divide, a medida que se estima necesario, el *dataset* completo de 23877 muestras en un *split* aleatorio de 80/10/10, para entrenamiento/validación/prueba. Notando que no hay ningún tipo de garantía respecto a la naturaleza de los precios de las muestras elegidas para conformar cada división, por lo que pueden estar desbalanceadas. Esto es altamente conveniente para realizar experimentos de desempeño de regresión, ya que se puede obtener una mejor estimación de la capacidad de generalización.

Capítulo 4

Generación de *deep features*

Con el fin de crear *features* que ayuden a resolver de mejor forma el problema de regresión de precios de casas en venta, se procede a re-entrenar 2 arquitecturas de redes convolucionales profundas para el problema de clasificación de casas entre caras y baratas, y posteriormente, para resolver un problema de clasificación de 6 clases, en base al precio de las propiedades. Las arquitecturas a entrenar corresponden a *Inception Resnet V2*[15] y *PNasNet Large 331*[17].

Es posible, gracias a las arquitecturas y *datasets* modernos, entrenar redes convolucionales profundas que logran distinguir entre 1000 clases de objetos distintos, utilizando imágenes. El *dataset* de entrenamiento para una tarea tan compleja cuenta con millones de imágenes, siendo tal cantidad de imágenes regularmente necesaria para entrenar redes convolucionales profundas desde el comienzo. Dado el tamaño de 23877 muestras del *dataset* es poco probable que se pueda entrenar una arquitectura de red convolucional profunda desde el comienzo para resolver el presente problema de clasificación, siendo el principal aspecto problemático la alta capacidad de la red: Se espera que, una red con tanta capacidad, frente a un problema binario, comparativamente mucho más sencillo que el que motivó el diseño de la arquitectura, se sobreajuste y, por tanto, produzca *features* cuya información sea pobre.

Dada la restricción anterior es posible aprovechar las arquitecturas mencionadas a través del *transfer learning*: Tomar como punto inicial de la optimización los pesos sinápticos aprendidos para resolver el problema de *Imagenet* y entrenar una porción menor de la red manteniendo el resto de los pesos ya aprendidos.

4.0.1. Esquema de arquitectura de redes

El esquema de la red *Inception Resnet V2* utilizada es el siguiente:

- Entrada
- conv2d_1a_3x3
- conv2d_2a_3x3
- conv2d_2b_3x3

- MaxPool_3a_3x3
- conv2d_3b_1x1
- conv2d_4a_3x3
- MaxPool_5a_3x3
- Mixed_5b (4 aristas)
- 10 x Block35 (3 aristas)
- Mixed_6a (3 aristas)
- 20 x Block17 (2 aristas)
- Mixed_7a (4 aristas)
- 9 x Block8 (2 aristas)
- Block8, sin activación (2 aristas)
- conv2d_7b_1x1
- Capa agregada: *Fully connected* de 256 neuronas, sin activación.
- *Logits*

Los bloques de tamaño 35x35, 17x17 y 8x8 se repiten 10, 20 y 10 veces, respectivamente. El número de aristas indica la cantidad de caminos que toman los datos dentro del *scope*, uniéndose los resultados en una capa de concatenación.

Para *PNasNet* el esquema utilizado es el siguiente:

- Entrada
- *Stem*
- 12 x *Cell*
- Global_pool
- Capa agregada: *Fully connected* de 256 neuronas, sin activación.
- *Logits*

4.0.2. Problemas y capacidad

Como se mencionó anteriormente, el objetivo es generar *deep features* que puedan ser utilizados para complementar la información al momento de realizar regresión de precio de venta de casas. Para esto se plantean 2 problemas para que las redes convolucionales profundas re-ajusten sus pesos: Clasificación binaria de casas entre baratas y caras, y clasificación de 6 clases de casas de todo el *dataset*. Las 6 clases se determinan en base al precio de la propiedad, y los límites de estas categorías están determinados de forma que las clases tengan una representación balanceada.

Ya que se estima que el problema de clasificación binaria es un problema sencillo para la capacidad de las redes convolucionales profundas, se procede a re-entrar simplemente una pequeña porción de ambas redes. En el caso de *Inception* se re-entrena el último *block8* y la capa de *flatten* de 1536 neuronas de nombre *conv2d_7b_1x1* que le sigue, agregando,

además, una capa adicional, *fully connected* de 256 neuronas, para extraer los *deep features* con dimensiones más reducidas. Para *PNasNet* se re-entrena la última capa de *global pooling* y la capa adicional de 256 neuronas.

Para el problema más complejo de clasificación en 6 clases se aumenta la capacidad de re-entrenamiento de las redes. Se agrega a la lista de entrenamiento de *Inception* los otros 9 *block8* anteriores, y para *PNasNet* se agrega *Cell 11* (último).

4.0.3. Hiperparámetros y otras consideraciones

- Debido a las limitaciones de memoria del *hardware* utilizado, y las restricciones de tiempo, se utiliza un tamaño de *batch* de 4 imágenes de 1280 x 1280 píxeles.
- Para compensar el tamaño del *dataset* se utiliza el pre-procesamiento de imágenes *standard* de *Inception Resnet V2*, que, de forma aleatoria, corta, distoriona los colores y voltea horizontalmente las imágenes.
- En el caso de *Inception* no se actualiza la información de *batch normalization*, ya que el *dataset* es muy pequeño para generar cambios significativos.
- Antes de las salidas de las redes se agrega una capa *fully connected* de 256 neuronas sin función de activación, por lo que en la práctica esto corresponde simplemente a una transformación lineal entre las capas que conecta, lo que se presta bien para los fines de reducir dimensionalidad perdiendo menos información que con una función de activación agresiva como $ReLU$. Esta capa será utilizada para extraer los *features* que serán usados en el problema de regresión.
- Dentro de las configuraciones exploradas, finalmente se utiliza un *learning rate* de 0.0001 en el problema de clasificación binaria, mientras que en el de 6 clases se escalonan *learning rates* de 0.001, 0.0001 y, finalmente, 0.00001.
- El optimizador con mejores resultados es *Adam*.
- Para definir cuando se detiene el entrenamiento se adopta la estrategia de *early stopping* con paciencia de 5 epochs sobre *accuracy* de validación: Si transcurren 5 *epochs* en que no ha mejorado la clasificación del conjunto de validación, entonces se da por terminado el entrenamiento.
- Re-entrenar toda la red no es recomendable por varias razones: No hay suficientes datos para converger bien, el problema es demasiado sencillo para la red, por lo que la probabilidad de que se sobreajuste es muy alta, la probabilidad de que los pesos tengan *spikes* o se indefinan es alta si no se usa un *learning rate* muy bajo, lo que es prohibitivo en tiempo, entre otras.
- Una de las razones de por qué se decide utilizar imágenes de 1280 x 1280, que constituye una resolución alta para lo que se suele utilizar en estas redes, es para compensar el pequeño tamaño del *dataset*, entregándole a la red mucha información por cada muestra.
- Para *Inception* se utiliza *weight decay* de 0.00004, y funciones de activación $ReLU$, exceptuando la capa de 256 neuronas sin función de activación.

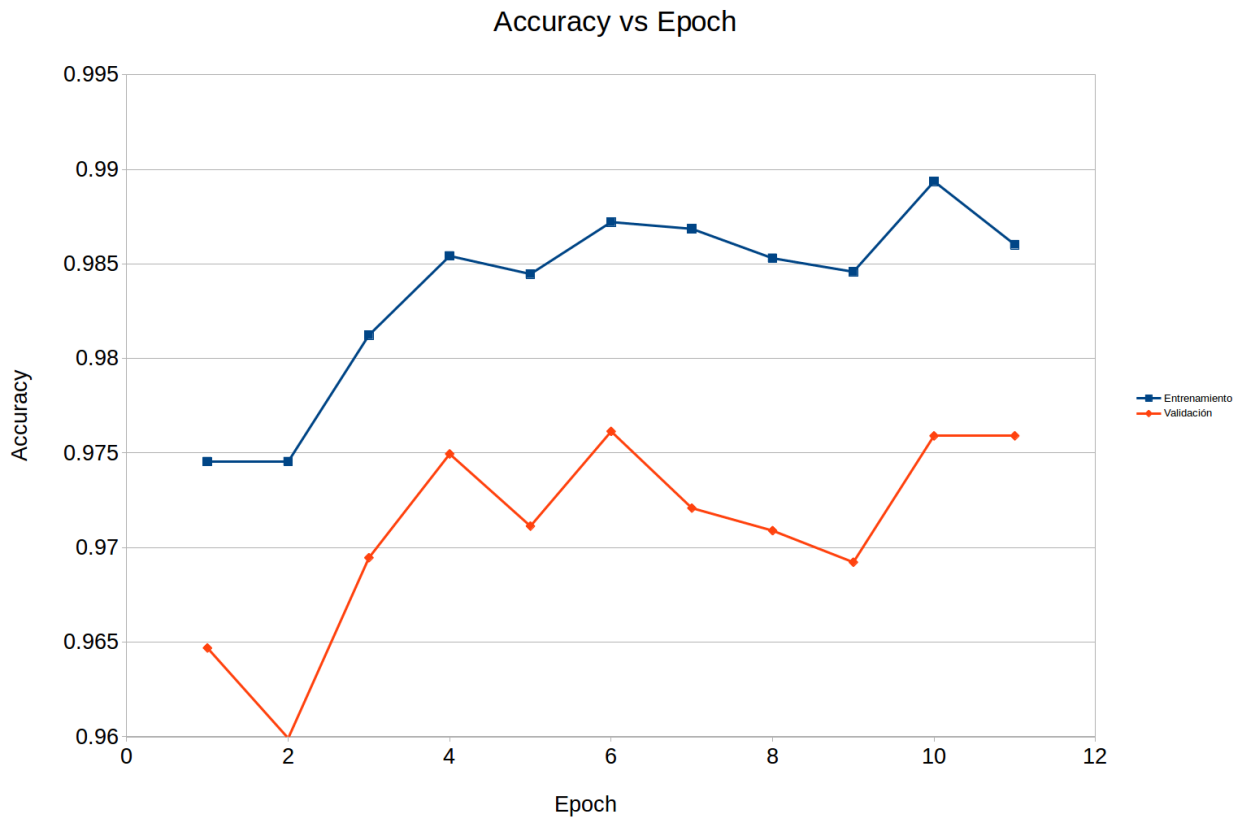


Figura 4.1: *Accuracy vs epoch* para *Inception* en el problema de clasificación de casas baratas y caras. Corresponde al nivel de *zoom* 19.

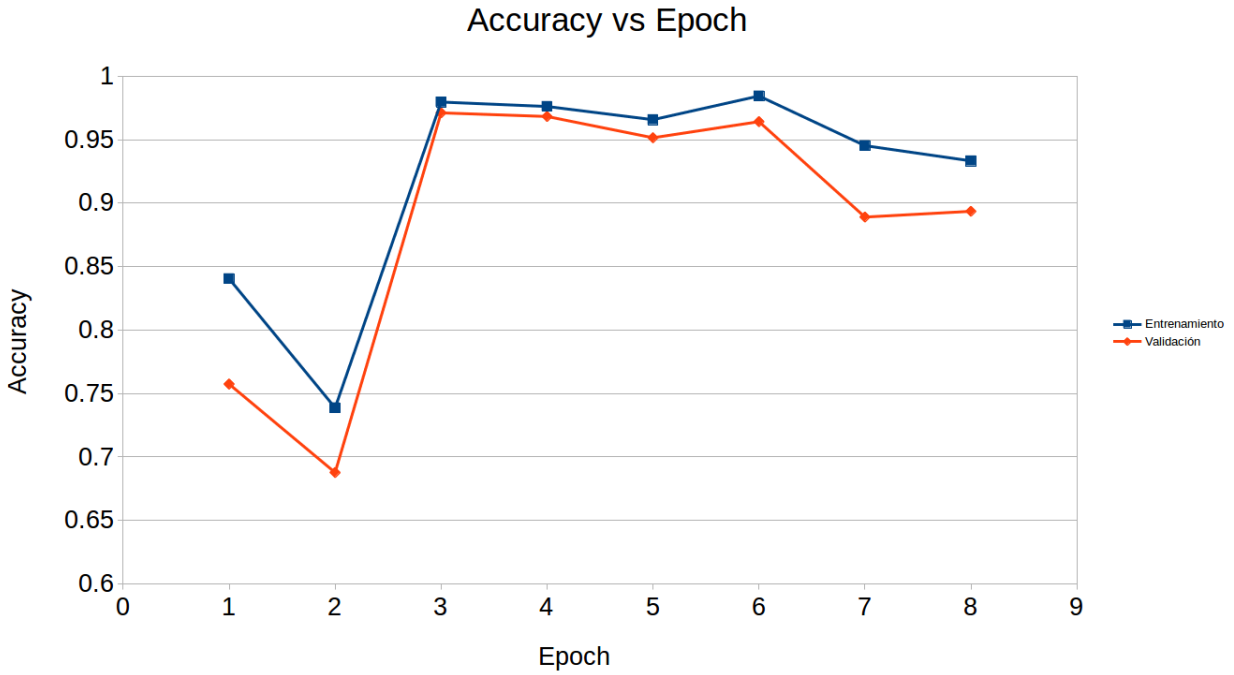


Figura 4.2: *Accuracy vs epoch* para *PNasNet* en el problema de clasificación de casas baratas y caras. Corresponde al nivel de *zoom* 18.

4.0.4. Resultados de clasificación

El problema de clasificación binaria no presenta mayores desafíos para las redes, logrando *accuracies* superiores a 0.96 tanto en validación como en entrenamiento, de hecho, como se aprecia en las Figuras 4.1 y 4.2, todo indica que es un problema fácil para las redes. Una vez que se han determinado los hiperparámetros correctos la única dificultad del problema es el tiempo que toma a las redes converger, el que es significativo. Sin duda, este tiempo es mucho menor que entrenar una red de este tamaño desde el comienzo, pero no debe tomarse a la ligera.

No debemos olvidar que el objetivo de este paso es mejorar la regresión a través de *features* significativos, por lo que esta clasificación exitosa no necesariamente se traducirá en *features* útiles.

Para el problema de clasificación de 6 clases se alcanzan *accuracies* cercanos a 0.5 en validación, por lo que no se considera resuelto, incluso después de más de 50 *epochs*. Nuevamente, resolver estos problemas planteados no implica que los *features* sean necesariamente buenos o malos.

4.0.5. Consideraciones técnicas

Las redes se entrenan usando la interfaz de *Tensorflow*[24] *slim*, utilizando los *checkpoint*[25] disponibles de modelos ya entrenados para resolver *Imagenet*. Para evitar problemas de compatibilidad se utiliza *Tensorflow* 1.5.0, corriendo bajo *CUDA* 9.0.176, con *CuDNN* 7.0.5, driver, 384.130 y *Python* 3.6.8. Para entrenar las redes se utiliza una *GPU Nvidia GTX* 1070. Debido a la restricción de memoria de la tarjeta el máximo tamaño de *batch* utilizable, sin experimentar degradación, es de tamaño 4, por lo que un *epoch* toma entre 50 y 60 minutos, para *Inception* y cerca del doble para *PNasNet*.

4.0.6. Acerca de la elección de arquitecturas

Si bien, el objetivo de entrenar una red convolucional para la resolución de un problema de clasificación de casas baratas y caras para extraer *features* puede lograrse de varias formas (incluso no re-entrenando ciertas arquitecturas y simplemente extrayendo *features* directamente) un trabajo anterior[19] hace uso de la arquitectura *Inception V3*, reportando resultados satisfactorios en el uso de *features* de clasificación para mejorar la regresión de precios. Tomando esto como punto de referencia se procede a investigar el comportamiento de la arquitectura que puede considerarse sucesora de *Inception V3*: *Inception Resnet V2*. Como contraparte, *PNasNet Large 331* cuenta con un número comparable de parámetros (lo que permite entrenar en el *hardware* utilizado), resuelve *Imagenet* un poco mejor que *Inception Resnet V2* y cuenta con modelos pre-entrenados disponibles para *Tensorflow* en la plataforma *slim*, es por esto que se presenta como un punto de comparación asequible.

Capítulo 5

Regresión

Una vez que se han fabricado *deep features*, en base a ambas arquitecturas de red (*i.e: Inception Resnet V2 y PNasNet Large 331*) para cada una de las 23877 muestras del *dataset* se procede a entrenar regresores que resuelvan el problema de estimación de los precios de venta de las propiedades, utilizando los *features* generados como información de entrada.

El primer modelo corresponde a una red neuronal *fully connected* de 2 capas ocultas, que toma como información de entrada los 6 *features* originales (*i.e*: Número de habitaciones, número de baños, m^2 construidos, m^2 de terreno, latitud y longitud), también llamados *house attributes* o HA, y los *deep features*, correspondientes a 1792 *floats* (256 por cada nivel de *zoom*). Los resultados obtenidos por esta arquitectura, variando el optimizador, *learning rate*, funciones de activación, cantidad de capas ocultas, normalización de los datos de entrada, reducción de dimensionalidad, y menor cantidad de niveles de *zoom*, son, en general, deficientes. Es claro que esta arquitectura de red no tiene suficiente capacidad para resolver el problema de forma satisfactoria, obteniendo para una entrada de 1798 *features* (*i.e:* todos los niveles de *zoom* y los HA) un R^2 cercano a 0.72 para el conjunto de entrenamiento en los experimentos realizados.
Otro detalle relevante es que, en comparación al siguiente modelo, todas estas arquitecturas de red toman, al menos, un orden de magnitud más tiempo en converger. Debido a los resultados deficientes encontrados, y al alto costo enfrentado, se descartan arquitecturas de redes neuronales *fully connected* como regresores.

Debido a su buena capacidad de aprender datos de entrada, relativo bajo tiempo de entrenamiento y comportamiento altamente replicable, se decide utilizar *Random Forest* como regresor. En los experimentos realizados, el modelo de regresión de *Random Forest* alcanza valores de R^2 cercanos a 0.98 en los conjuntos de entrenamiento de los experimentos realizados, por lo que es claro que logra, al menos, memorizar los datos de entrada. Al probar, sin embargo, su capacidad para generalizar frente a nuevos datos, los experimentos revelan que la combinación de 1798 *features* genera resultados aceptables con valores de R^2 mayores a 0.8, pero es necesario establecer un *baseline* de referencia. El *baseline* de estos experimentos serán los resultados obtenidos por el mismo regresor utilizando los 6 *house attributes* originales. Al establecer el *baseline*, se desprende de los experimentos que la regresión con 1798 *features*

obtiene peores resultados, comparativamente.

Se plantean, en base a esto, las siguientes interrogantes:

- ¿Por qué empeora el resultado al utilizar todos los *features*?
- ¿Hay algún aspecto deseable?
- ¿Existe alguna combinación de niveles de *zoom* que entregue información útil al regresor?
- ¿Serán características propias de las propiedades de Santiago las que dificultan el uso de estos *deep features*?

5.1. Experimentos

Contando con un modelo de regresión que tiene suficiente capacidad para aprender los datos de entrada, y con estas interrogantes en mente, se procede a realizar experimentos que permitan determinar el nivel de riqueza de información que aporta cada nivel de *zoom*. Vale la pena recordar que estos experimentos se realizan con un *split* de 80/10/10, para entrenamiento, validación y prueba, respectivamente. Cada uno de los experimentos opera con un *split* aleatorio distinto, lo que, gracias a que las particiones no son necesariamente balanceadas respecto a los precios de las propiedades, permite enfrentar al regresor a distintas distribuciones de los datos.

5.1.1. Exploración

Utilizando un regresor de *Random Forest* con 50 estimadores, se procede a comparar el resultado del *baseline* con una serie de subconjuntos de los *features* generados a distintos niveles de *zoom*, por ambas redes, frente al problema de clasificación de casas caras y baratas.

Nomenclatura necesaria para apreciar las figuras que resumen los resultados de los experimentos:

- HA: Resultados obtenidos por el regresor utilizando los *house attributes* como información de entrada.
- X: Resultados obtenidos por el regresor utilizando los *deep features* generados para el nivel X de *zoom*.
- HA + X: Resultados obtenidos por el regresor utilizando los *deep features* generados para el nivel X de *zoom* y los *house attributes*.
- X $\rightarrow$ Y: Resultados obtenidos por el regresor utilizando los *deep features* generados para los niveles desde X hasta Y, inclusive. Comenzando desde X.
- X $\leftarrow$ Y: Análogo a lo anterior, pero comenzando desde Y.
- HA + X $\rightarrow$ Y o HA + X $\leftarrow$ Y: Análogo a los anteriores, pero incluyendo los *house attributes*.
- X,Y: Resultados obtenidos por el regresor utilizando los *deep features* generados para los niveles X e Y.
- HA + X,Y: Análogo a lo anterior, pero incluyendo los *house attributes*.
- HA + X,Y,Z: Análogo a lo anterior, pero incluyendo un nivel de *zoom* adicional.

En las Figura 5.1 y 5.2 se muestran los valores de R^2 obtenidos en un ejercicio exploratorio en que se busca establecer una tendencia en cuanto a qué tan informativos son los distintos niveles de *zoom*.

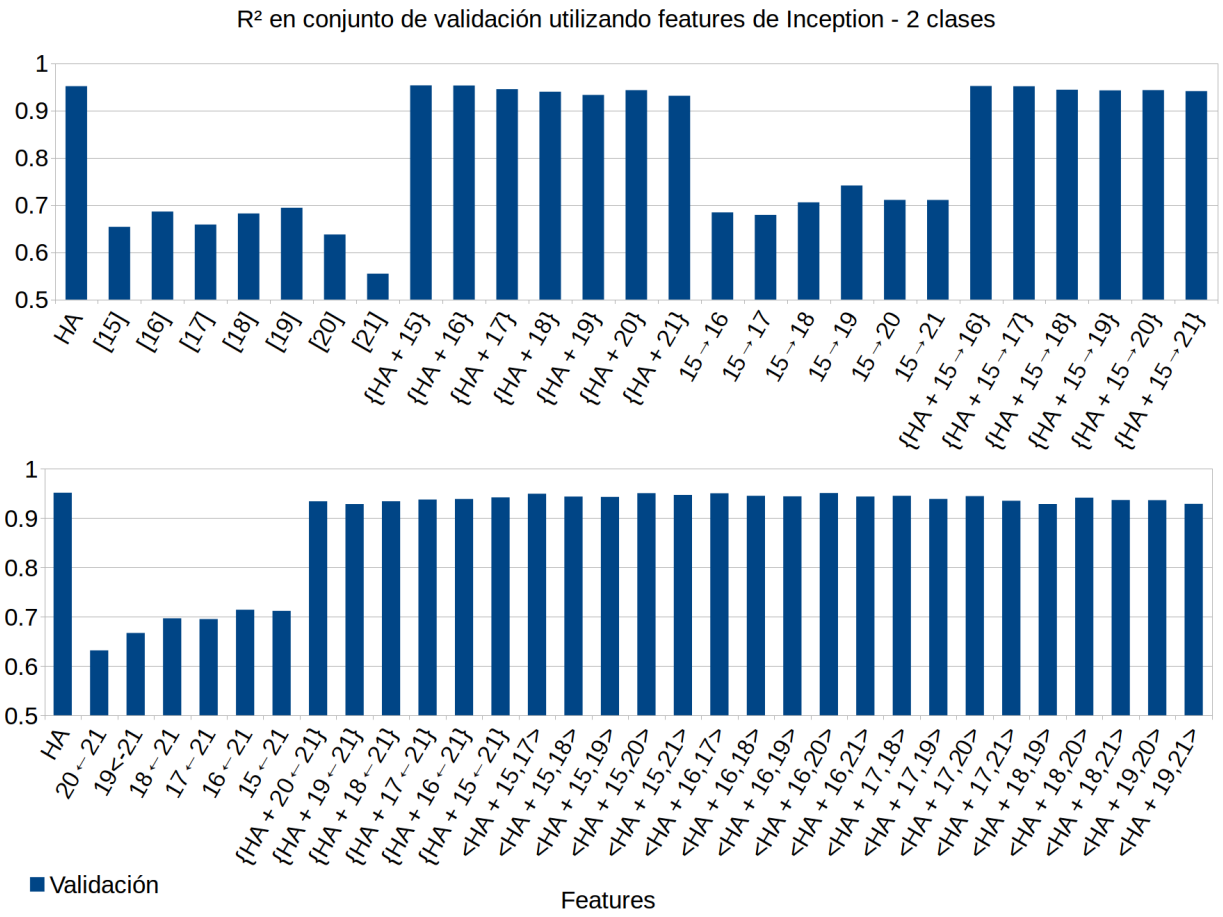


Figura 5.1: R^2 para todas las configuraciones de *features* de *Inception* estudiadas, sobre el conjunto de validación de un experimento.

Es directo notar, desde estas figuras, que la información que entregan los *deep features* no es suficiente para caracterizar los objetos por sí sola. Todas las combinaciones probadas de información exclusivamente de los *deep features* generalizan muy mal. En la Figura 5.3 se incluyen los valores de R^2 para los conjuntos de validación y entrenamiento, utilizando *deep features* generados con la red *Inception*, correspondientes al mismo experimento. Es posible apreciar que los niveles de *zoom* más grandes aportan suficiente información como para que el regresor pueda aprender de memoria los objetos de entrenamiento. Al combinar mayor número de niveles de *zoom* esta caracterización aumenta, pero sigue siendo insuficiente para generalizar satisfactoriamente.

Es importante notar que *Random Forest* logra hacer uso satisfactorio de esta información ya que tiene la capacidad de separar ciertas porciones de los *features* para tomar decisiones, lo que le permite evitar que los *House Attributes* se pierdan en la concatenación de *features*. Por otro lado, es un regresor que se beneficia de contar con una diferencia de escala entre *features*, a diferencia de, por ejemplo, las redes neuronales *fully connected*.

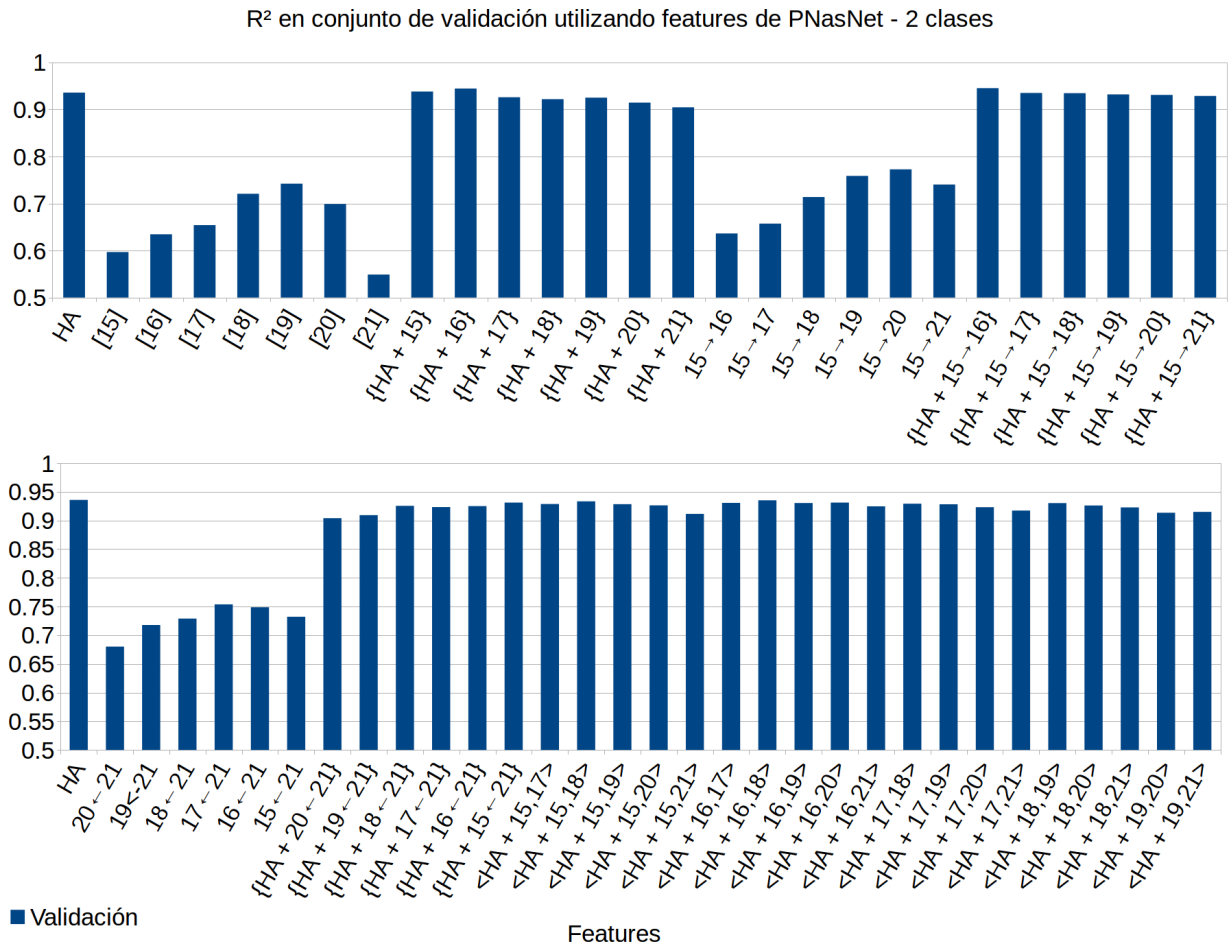


Figura 5.2: R^2 para todas las configuraciones de *features* de *PNasNet* estudiadas, sobre el conjunto de validación de un experimento.

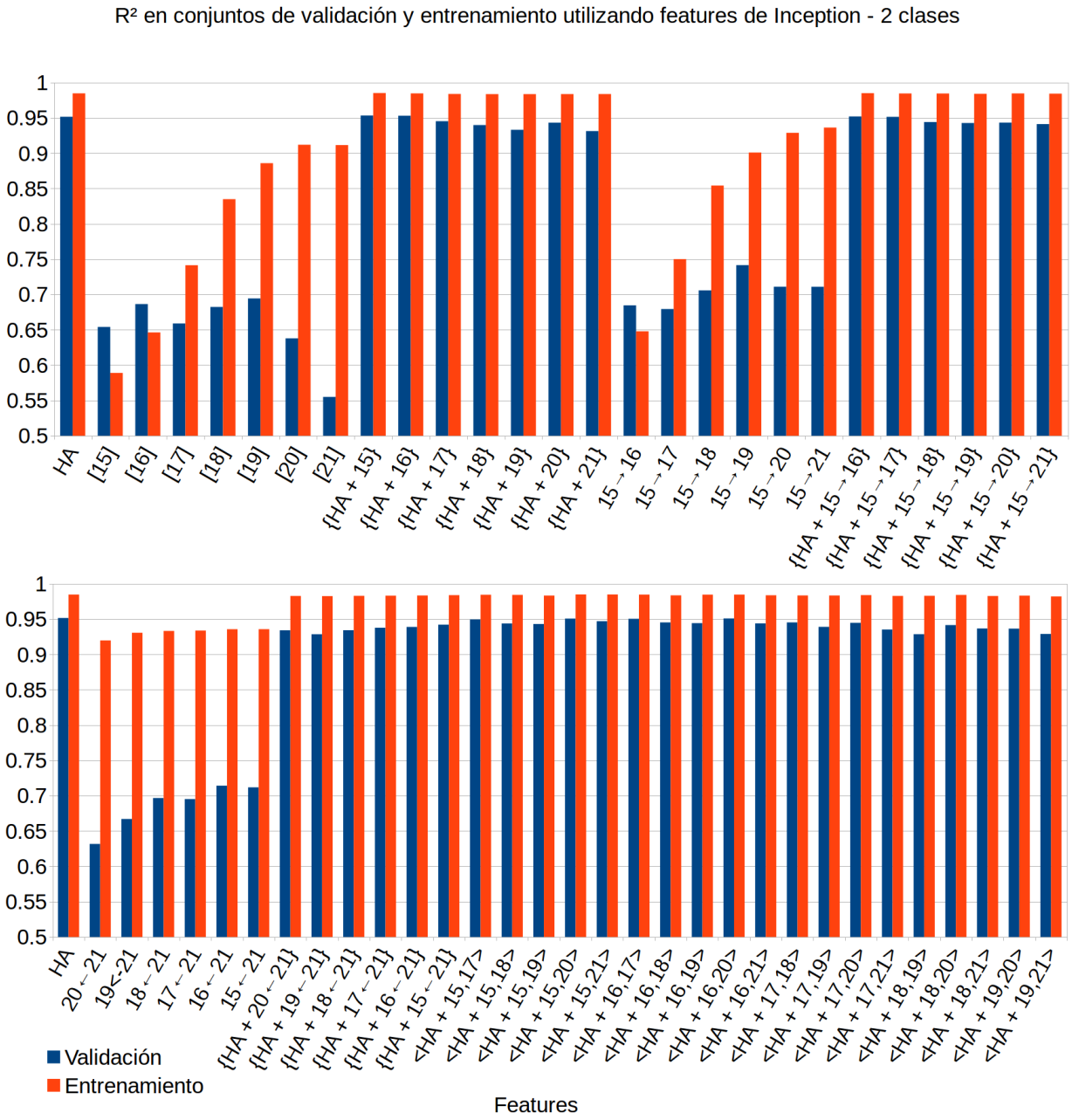


Figura 5.3: R^2 para todas las configuraciones de *features* de *Inception* estudiadas, sobre los conjuntos de validación y entrenamiento de un experimento.

Al incluir los *house attributes* los resultados mejoran considerablemente, lo que es de esperarse dados los resultados obtenidos por estos *features* de forma independiente. La mayoría de las combinaciones de niveles de *zoom* empeoran los resultados obtenidos utilizando HA, en especial al aumentar el nivel de *zoom*. Pero los niveles más bajos de *zoom*, aquellos que nisiquiera permiten memorizar satisfactoriamente el conjunto de entrenamiento (*e.g: zoom* 15), aportan información útil al momento de combinarlos con los HA. Estos niveles más bajos de *zoom* capturan información que va desde la comuna en la que se ubica la propiedad (*zoom* 15) hasta el barrio en que se ubica la propiedad (*zoom* 17). La información contenida en las

imágenes de niveles de *zoom* cercanos tiene un grado no despreciable de redundancia, pero, dados estos resultados, parecieran complementarse bien para el caso de *Inception*.

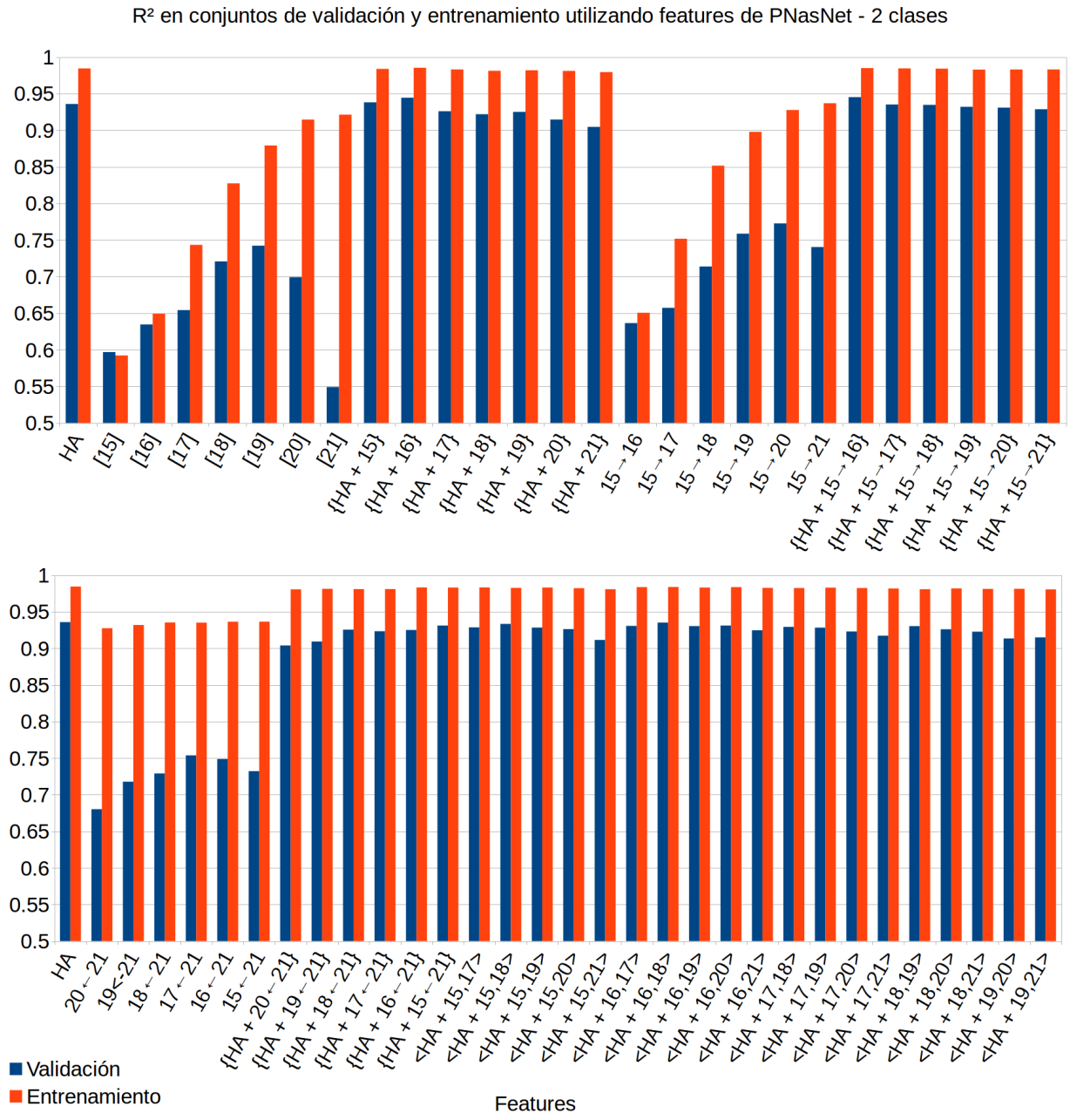


Figura 5.4: R^2 para todas las configuraciones de *features* de *PNasNet* estudiadas, sobre los conjuntos de validación y entrenamiento de un experimento.

Es posible obtener resultados mejores que los del *baseline* combinando la información de HA con los *features* de niveles de *zoom* bajos. En la Figura 5.5 se detalla, con mayor resolución, los valores de R^2 obtenidos por el regresor utilizando los *features* de *Inception* para las combinaciones competitivas. Todas las configuraciones, salvo el *baseline*, corresponden

a *deep features* de bajos niveles de *zoom*. Los $RMSE$ obtenidos por estas configuraciones son, también, menores que los del *baseline*, excepto para la combinación 15,16,17. Esta combinación no supera el valor de R^2 del *baseline* en este experimento, por lo que hay consistencia.

En el caso de *PNasNet*, el nivel de *zoom* 18 pareciera aportar un poco de información útil, pero no es suficiente como para mejorar el resultado del *baseline*. Otra característica interesante de los *deep features* generados en base a *PNasNet* es que funcionan mejor de forma desacoplada que en combinaciones.

Esta etapa de exploración permite notar una marcada tendencia hacia los niveles de *zoom* bajos. Pese a que la información contenida en los *deep features* de bajo nivel es repetitiva, resulta ser, como es esperable, información que los *house attributes* no pueden inferir: características del entorno en que se encuentra ubicada la propiedad, es por esto que constituyen un complemento útil para el regresor.

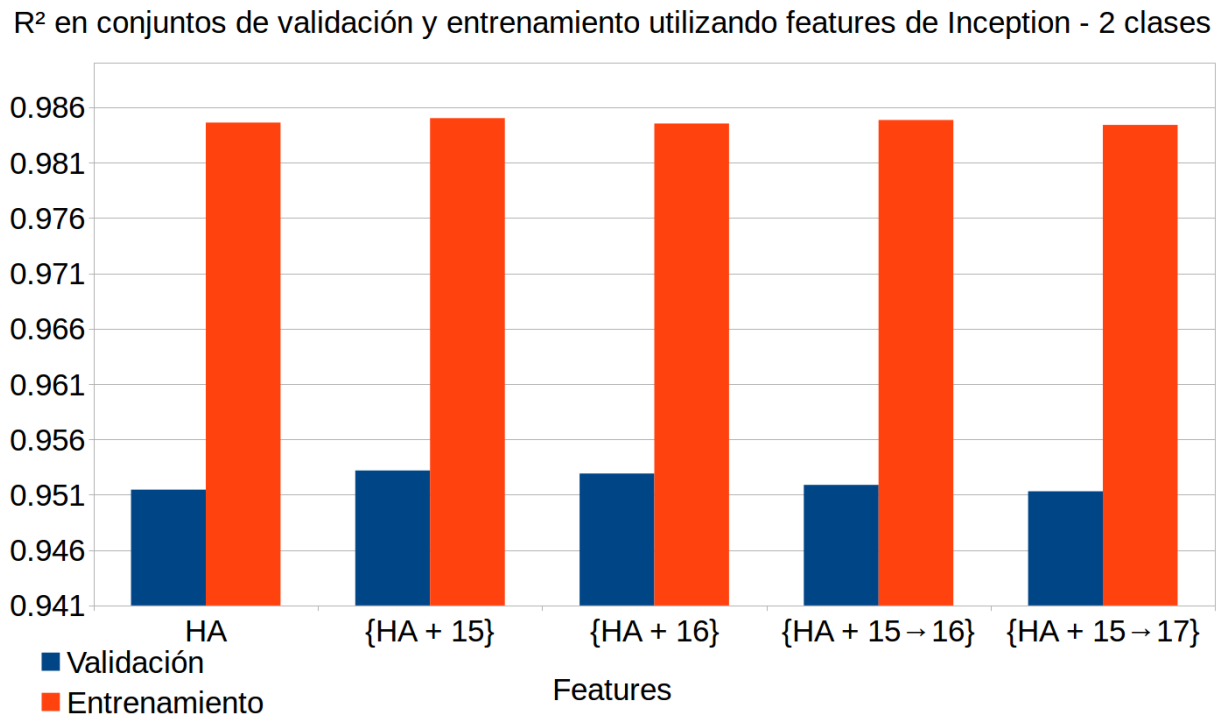


Figura 5.5: R^2 para configuraciones de *features* de *Inception* que son competitivas con los resultados obtenidos por los *HA*, sobre los conjuntos de validación y entrenamiento de un experimento.

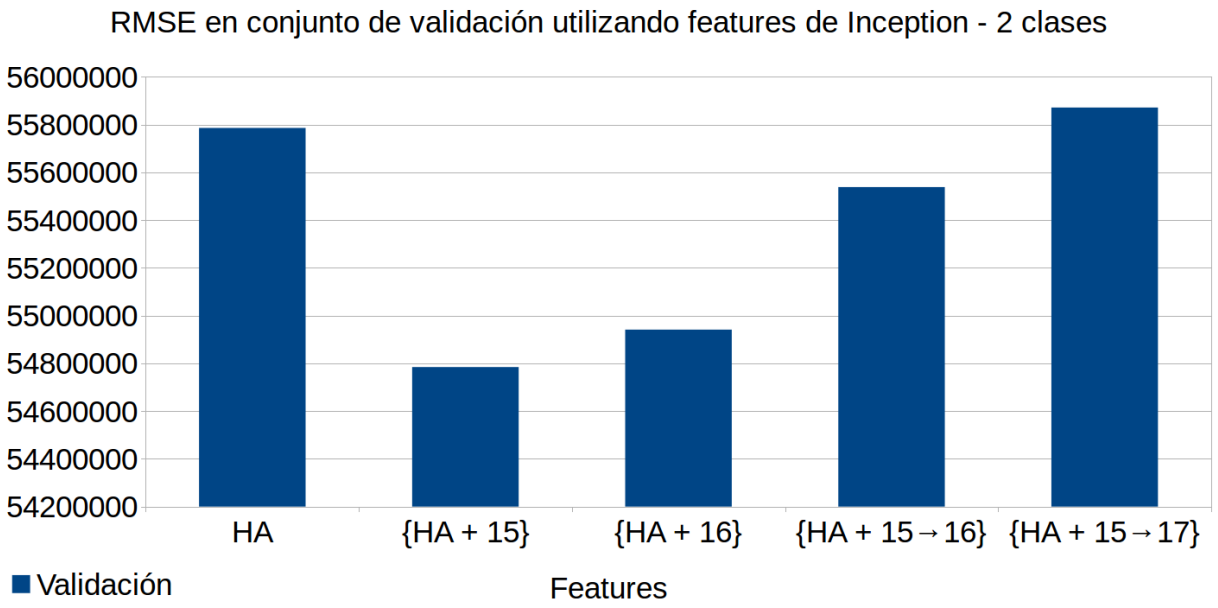


Figura 5.6: *RMSE* para configuraciones de *features* de *Inception* que son competitivas con los resultados obtenidos los *HA*, sobre los conjuntos de validación y entrenamiento de un experimento.

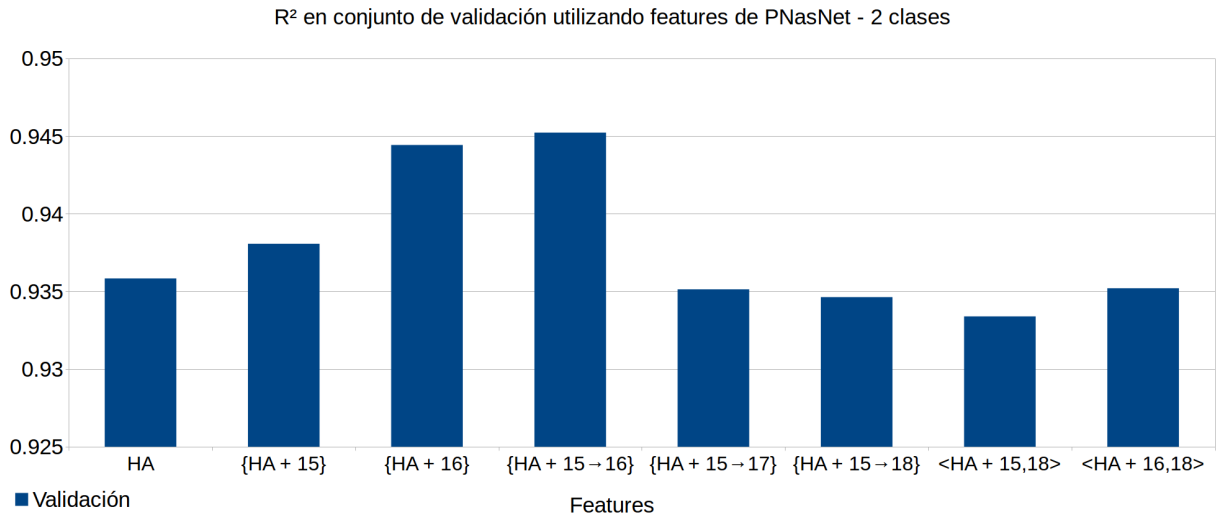


Figura 5.7: R^2 para configuraciones de *features* de *PNasNet* que son competitivas con los resultados obtenidos por los *HA*, sobre los conjuntos de validación y entrenamiento de un experimento.

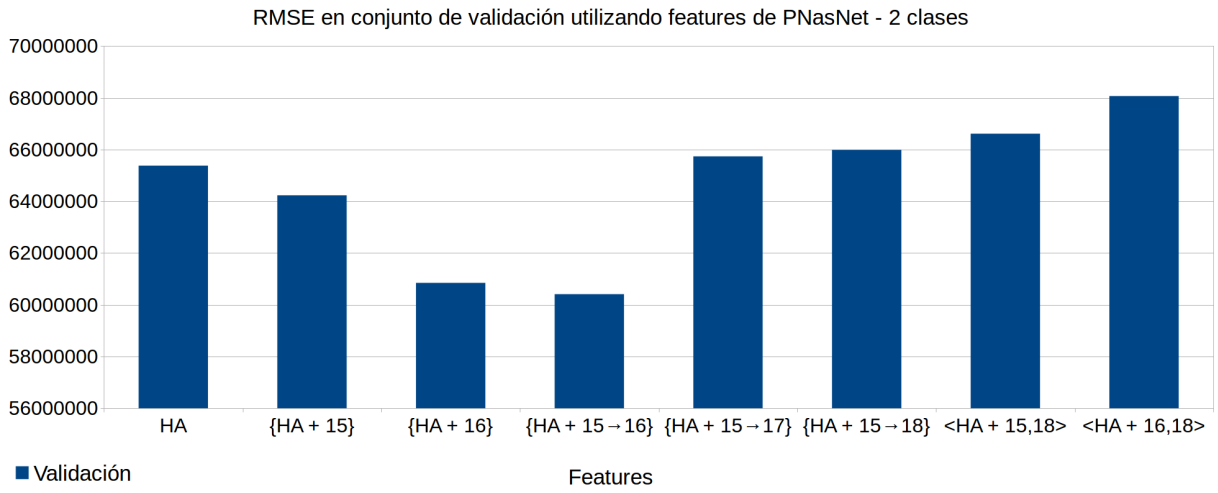


Figura 5.8: *RMSE* para configuraciones de *features* de *PNasNet* que son competitivas con los resultados obtenidos por los *HA*, sobre los conjuntos de validación y entrenamiento de un experimento.

5.1.2. Experimentos exhaustivos

Con el fin de determinar si la información aportada por los bajos niveles de *zoom* logra generar un verdadero impacto en el desempeño de los modelos de regresión, se procede a experimentar con 20 *splits* del *dataset*. Para cada uno de estos *splits* se entrena un regresor *Random Forest* de 50 estimadores, utilizando distintas configuraciones de *features* de *zoom* bajo.

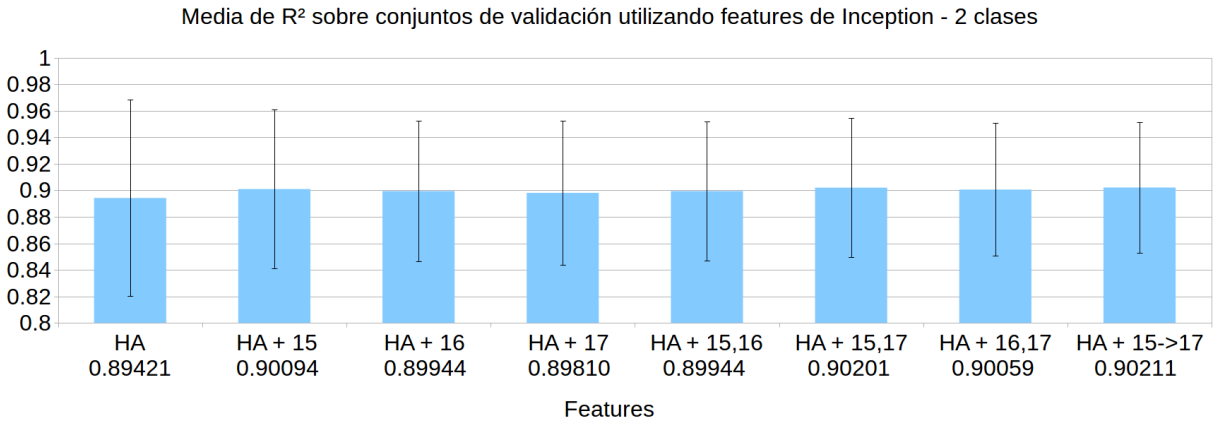


Figura 5.9: Media de R^2, con desviación estándar, para configuraciones de *features* de *Inception* competitivas, calculada sobre los resultados de todos los conjuntos de validación del experimento.

En la Figura 5.9 se detalla el valor obtenido de promediar los valores de R^2 obtenidos sobre el conjunto de validación de cada uno de los *splits*, utilizando los *features* generados por *Inception*. Todas las configuraciones de bajo nivel de *zoom* logran superar el rendimiento obtenido utilizando los HA. Además, la desviación estándar disminuye a medida que se utilizan más *deep features*. Pese a que HA + 17 obtiene una mejor media de R^2 que el *baseline*, obtiene una peor media de $RMSE$. Esto se puede deber a que falla un poco más en precios de propiedades más caras, contribuyendo un valor mayor al error.

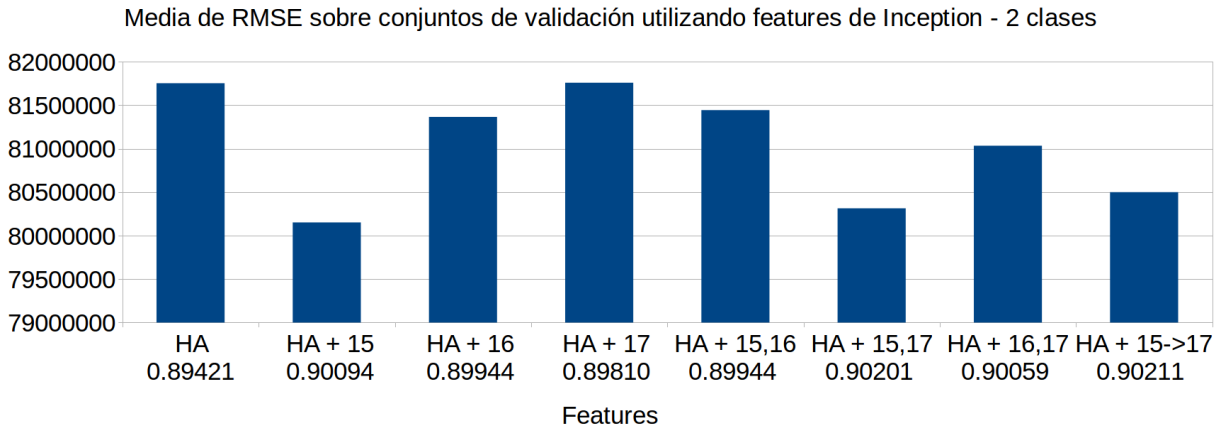


Figura 5.10: Media de $RMSE$ para configuraciones de *features* de *Inception* competitivas, calculada sobre los resultados de todos los conjuntos de validación del experimento.

En el caso de los *features* de *PNasNet*, no todas las configuraciones lograron superar el desempeño del *baseline*. Se va haciendo evidente una superioridad de los resultados obtenidos utilizando los *features* de *Inception*.

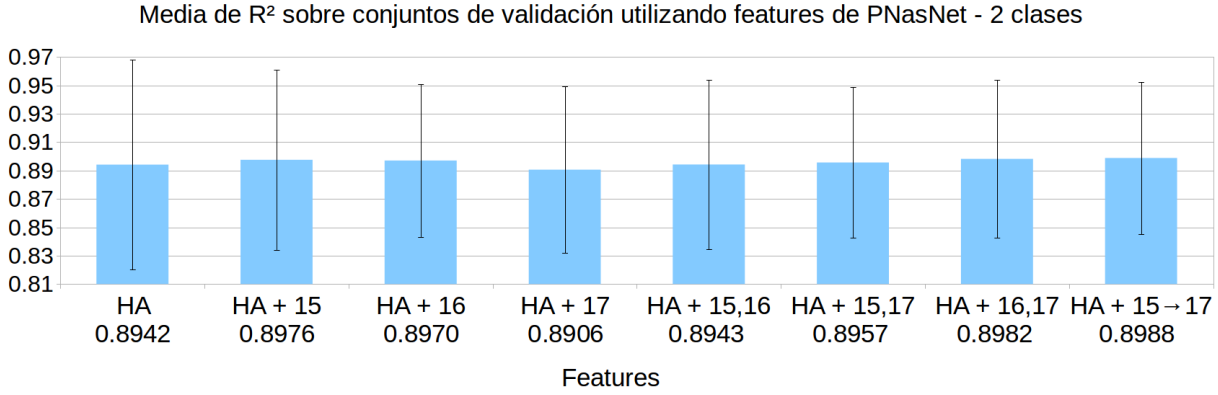


Figura 5.11: Media de R^2, con desviación estándar, para configuraciones de *features* de *PNasNet* competitivas, calculada sobre los resultados de todos los conjuntos de validación del experimento.

Las estimaciones realizadas en base a los *features* de *PNasNet* tienen mal desempeño respecto a los valores de $RMSE$, como se muestra en la Figura 5.12.

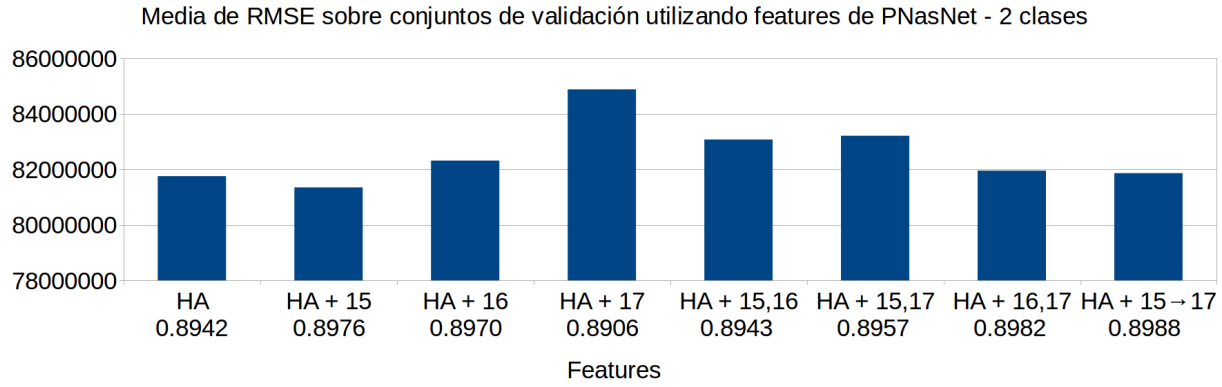


Figura 5.12: Media de $RMSE$ para configuraciones de *features* de *PNasNet* competitivas, calculada sobre los resultados de todos los conjuntos de validación del experimento.

De esta experimentación más profunda se hace evidente que el nivel de *zoom* que, consistentemente, entrega buena información complementaria a HA de forma independiente es el *zoom* 15. La baja varianza presentada por HA + 15,16,17 hace destacar esta configuración, que tanto en *Inception* como en *PNasNet*, logra obtener el mejor resultado de R^2 promedio sobre los conjuntos de validación.

Los resultados de validación dictan que los mejores modelos se obtienen con HA + 15 → 17, HA + 15,17 y HA + 15, en el caso de los *features* provenientes de *Inception* y HA + 15 → 17, HA + 16,17 y HA + 15, en el caso de los *features* provenientes de *PNasNet*. A continuación se describen los resultados de estos mejores candidatos sobre los conjuntos de prueba y se estudian otras configuraciones.

5.1.3. Resultados sobre conjuntos de prueba

Todos los candidatos logran superar el desempeño del *baseline* si se considera la media de R^2 sobre los conjuntos de prueba (Figuras 5.13 y 5.15), sin embargo, para el caso de los *features* generados por *PNasNet*, el desempeño disminuye a medida que se incluye más información, al contrario de los *features* generados por *Inception*. En la Figura 5.16 queda en evidencia que, incluso, la media de $RMSE$ aumenta al contar con mayor información, cuando se trata de los *features* de *PNasNet*, por lo que es claro que trabajan mejor de forma separada.

Gracias a la mayor cantidad de información desde la que extrapolar, las configuraciones con *deep features* de *Inception* sufren menos al entrenar en base a *splits* desbalanceados. En algunos *splits* desfavorables puede existir una diferencia de más de 0.1 R^2 entre el *baseline* y los modelos basados en *deep features* de *zoom* bajo. Cuando el *baseline* se comporta bien,

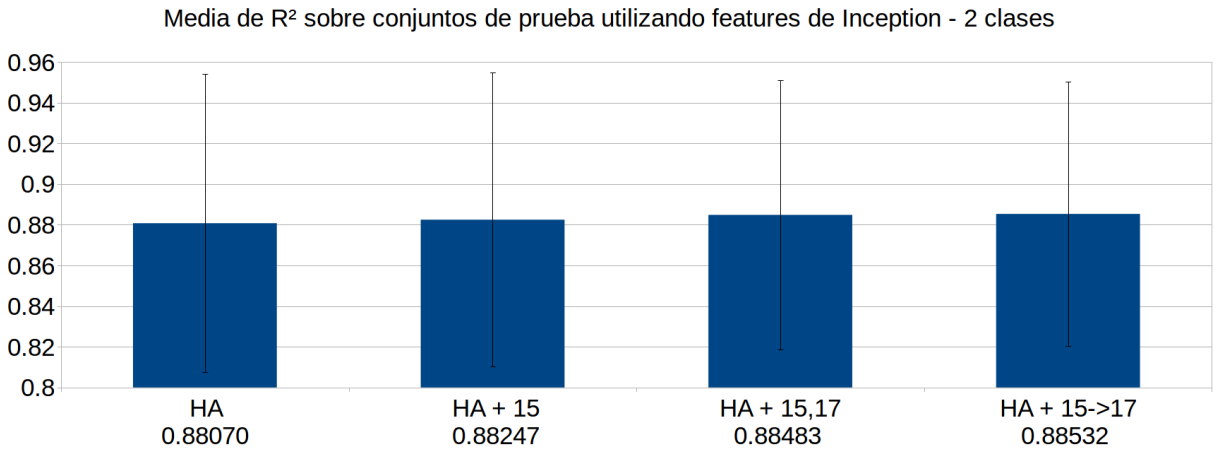


Figura 5.13: Media de R^2, con desviación estándar, para configuraciones de *features* de *Inception* competitivas, calculada sobre los resultados de todos los conjuntos de prueba del experimento.

el modelo de *deep features* lo sigue de cerca, pero cuando el *baseline* cae el modelo de *deep features* es más resiliente.

Al aumentar el número de estimadores, y, por tanto, la capacidad del modelo de regresión, se aprovecha de mejor manera la información contenida en los *features*.

Repitiendo los experimentos anteriores con un regresor con mayor capacidad se obtienen resultados consistentes. Los mejores candidatos siguen produciendo mejores estimaciones.

En la Figura 5.18, se aprecia que *Inception* sigue presentando la característica de tener *features* de bajo nivel de *zoom* que son complementarios, alcanzando cada vez mejores resultados.

En el caso de *PNasNet* se reafirma la tendencia a empeorar la estimación, desde el *zoom* 15, al incorporar la información de los niveles de *zoom* posteriores.

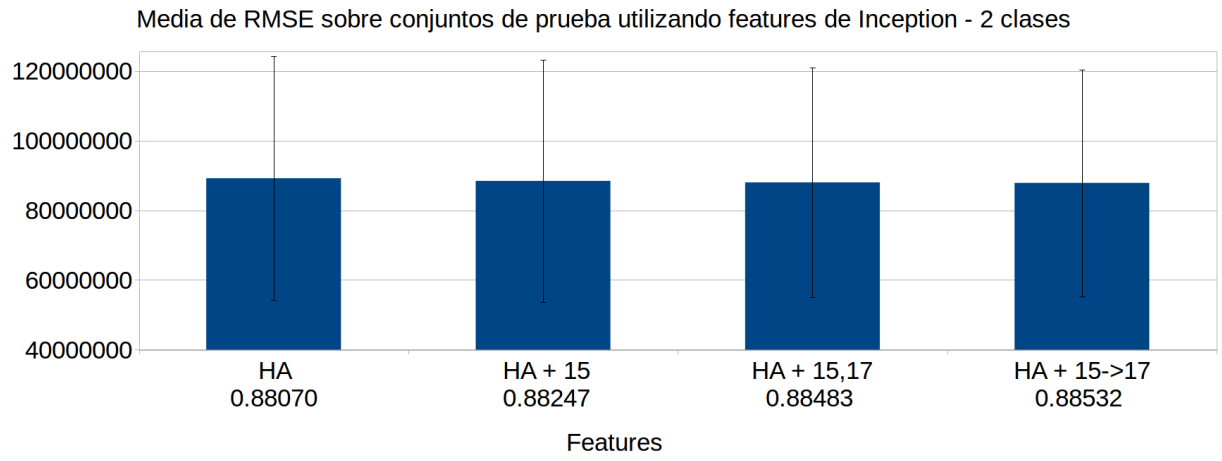


Figura 5.14: Media de $RMSE$ para configuraciones de *features* de *Inception* competitivas, calculada sobre los resultados de todos los conjuntos de prueba del experimento.

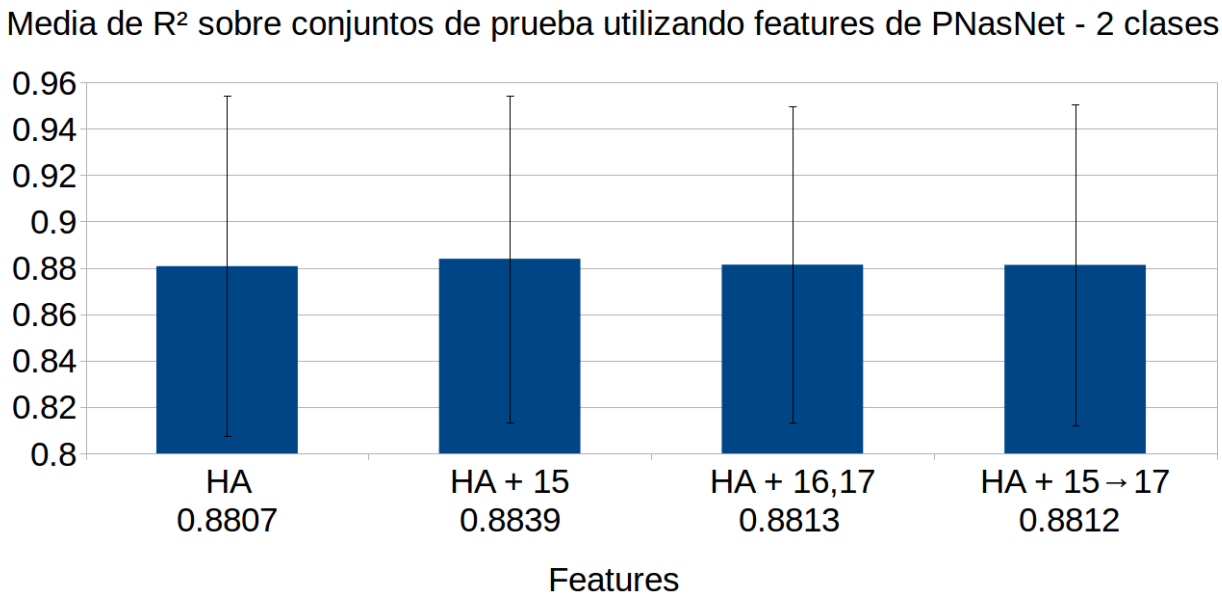


Figura 5.15: Media de R^2, con desviación estándar, para configuraciones de *features* de *PNasNet* competitivas, calculada sobre los resultados de todos los conjuntos de prueba del experimento.

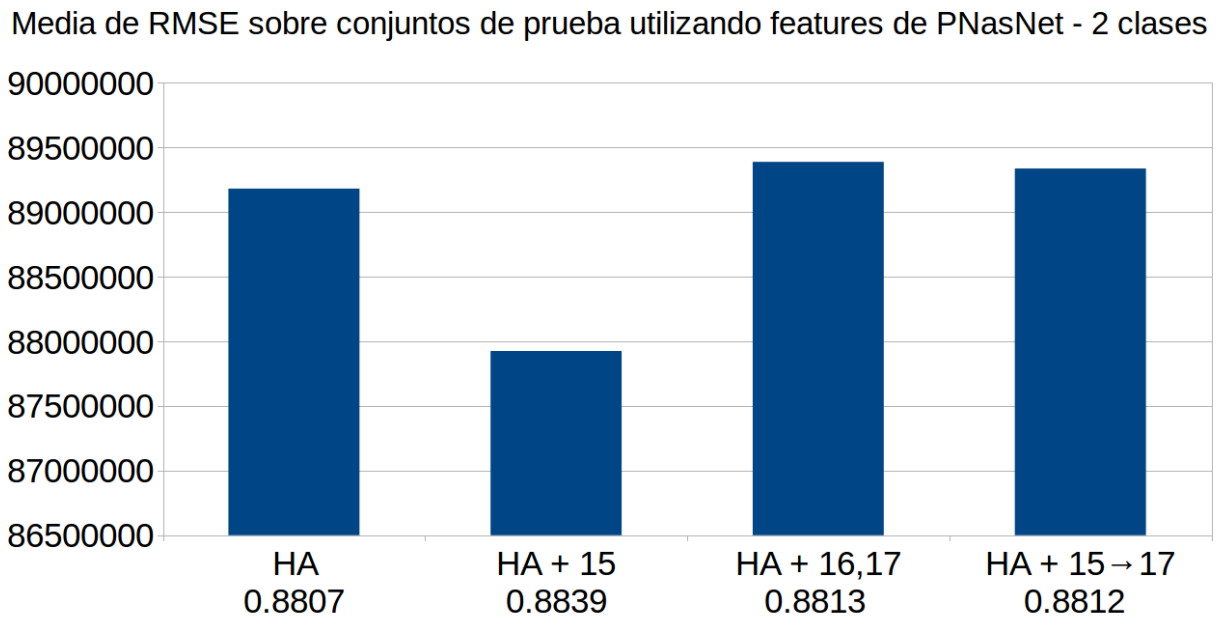


Figura 5.16: Media de *RMSE* para configuraciones de *features* de *PNasNet* competitivas, calculada sobre los resultados de todos los conjuntos de prueba del experimento.

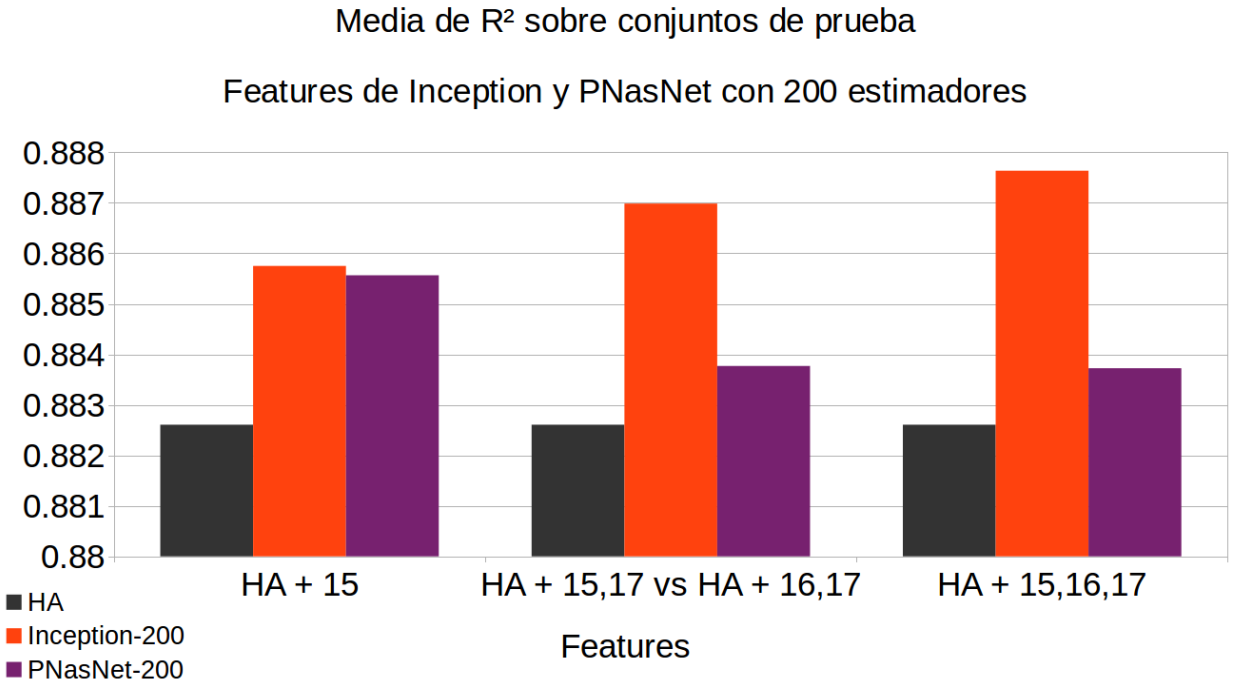


Figura 5.17: Comparación de media de R^2, sobre conjuntos de prueba, para configuraciones competitivas de *features* de *Inception* y *PNasNet*, con 200 estimadores en el regresor.

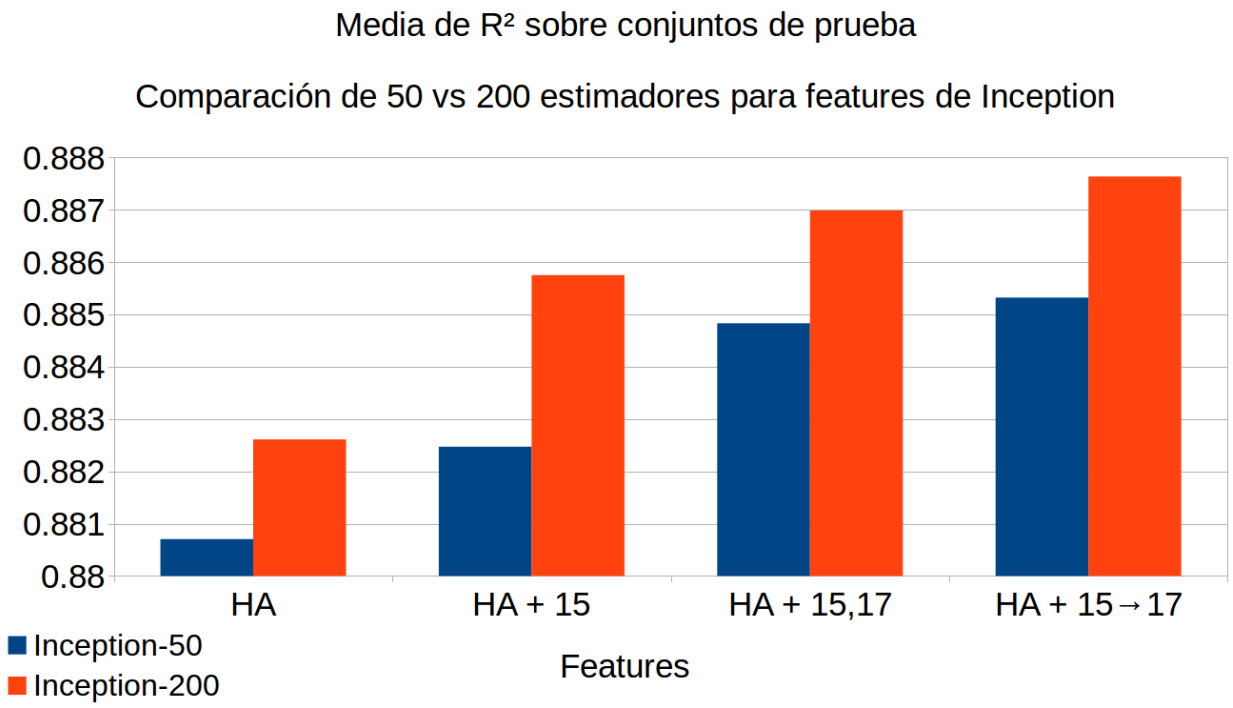


Figura 5.18: Media de R^2, para configuraciones de *features* de *Inception* competitivas, considerando 50 y 200 estimadores en el regresor, calculada sobre los resultados de todos los conjuntos de prueba del experimento.

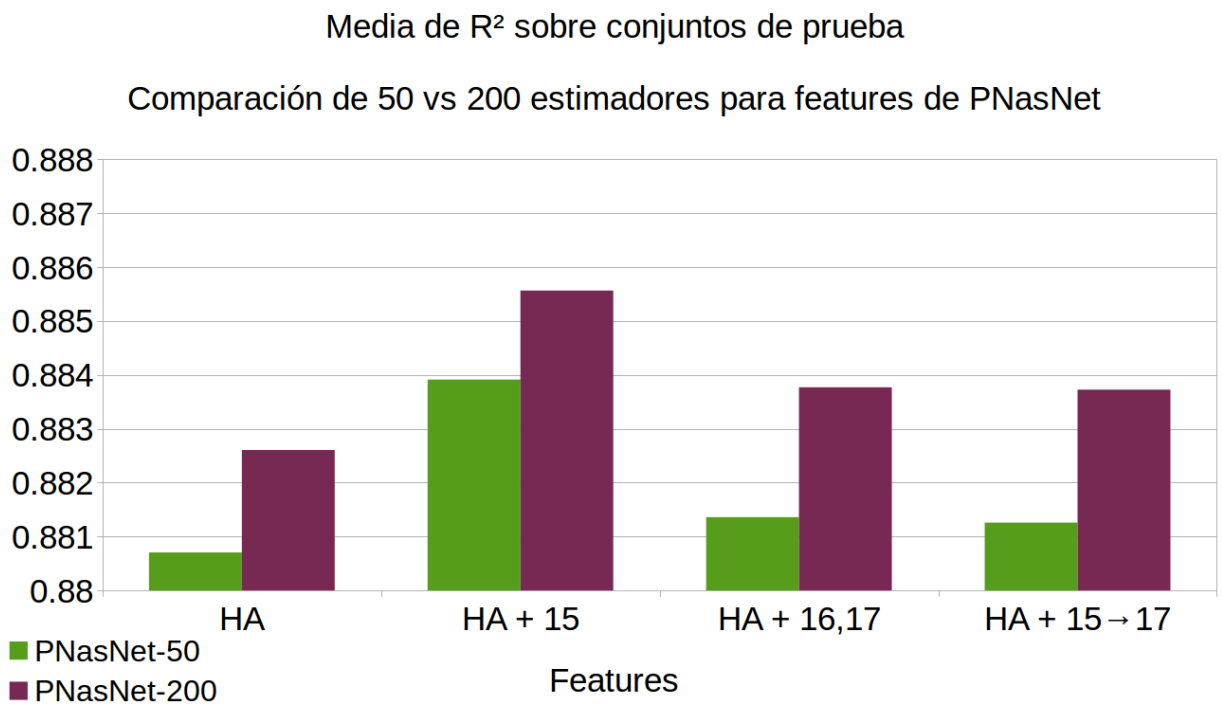


Figura 5.19: Media de R^2, para configuraciones de *features* de *PNasNet* competitivas, considerando 50 y 200 estimadores en el regresor, calculada sobre los resultados de todos los conjuntos de prueba del experimento.

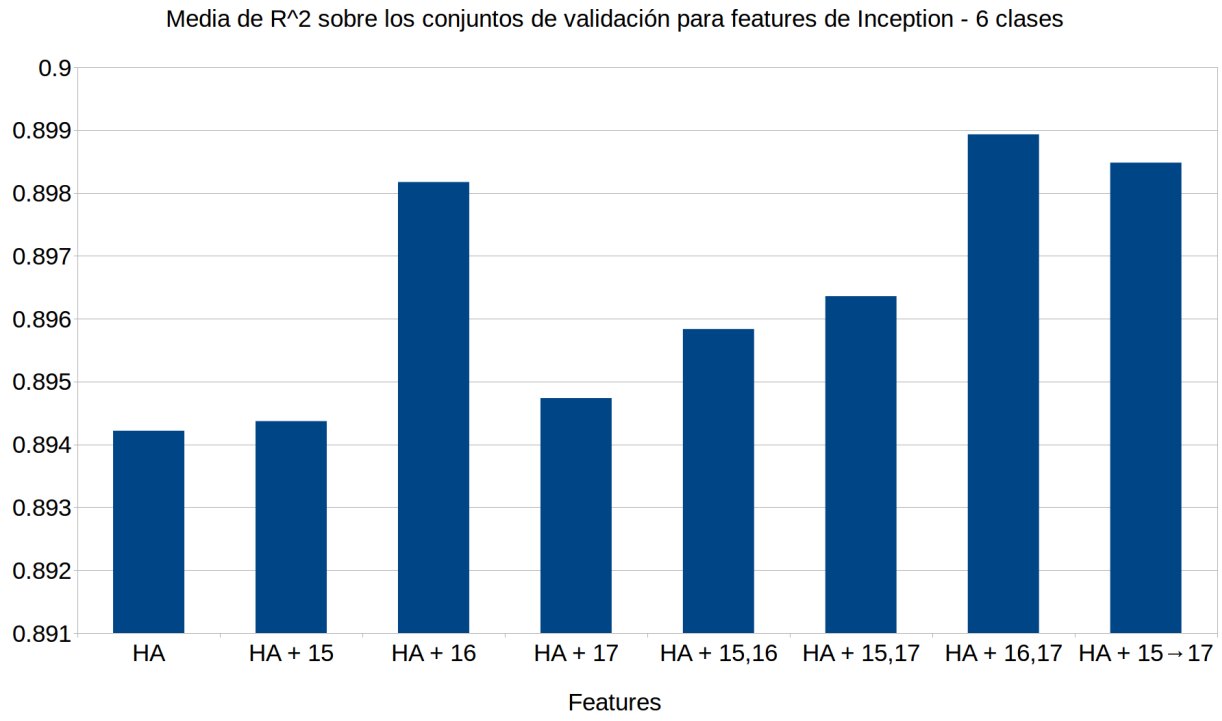


Figura 5.20: Media de R^2 sobre los conjuntos de validación, para configuraciones de *features* de *Inception* entrenada para resolver una clasificación de 6 clases.

5.1.4. *Deep features* en base a 6 clases

Considerando que aumentar la granularidad del problema de clasificación podría empujar a la red a generar *features* más ricos en información se procede a experimentar sobre los 20 *splits* utilizando los *features* generados por la red *Inception* para resolver el problema de clasificación de casas entre 6 categorías, en base a su precio, utilizando *Random Forest* como regresor con 50 estimadores.

Ya en el conjunto de validación es posible notar un deterioro en el desempeño del regresor utilizando los nuevos *features*. Recordando que la red alcanza cerca de un 0.5 de *accuracy* en este nuevo problema más complejo, pareciera ser que se introduce más ruido que información útil. Igualmente, considerando los 3 mejores candidatos en base a los resultados sobre los conjuntos de validación, se procede a evaluar sobre los conjuntos de prueba.

Claramente, el desempeño del modelo ha empeorado. Si bien, la configuración HA + 15 → 17 logra un mejor resultado que el *baseline*, este resultado es peor que lo logrado por la misma configuración con los *features* del problema binario. Se prefiere, en base a estos resultados, los primeros *features*.

Pese a que no se espera que a mayor capacidad del regresor este resultado mejore, no es posible descartar esta posibilidad con los resultados actuales.

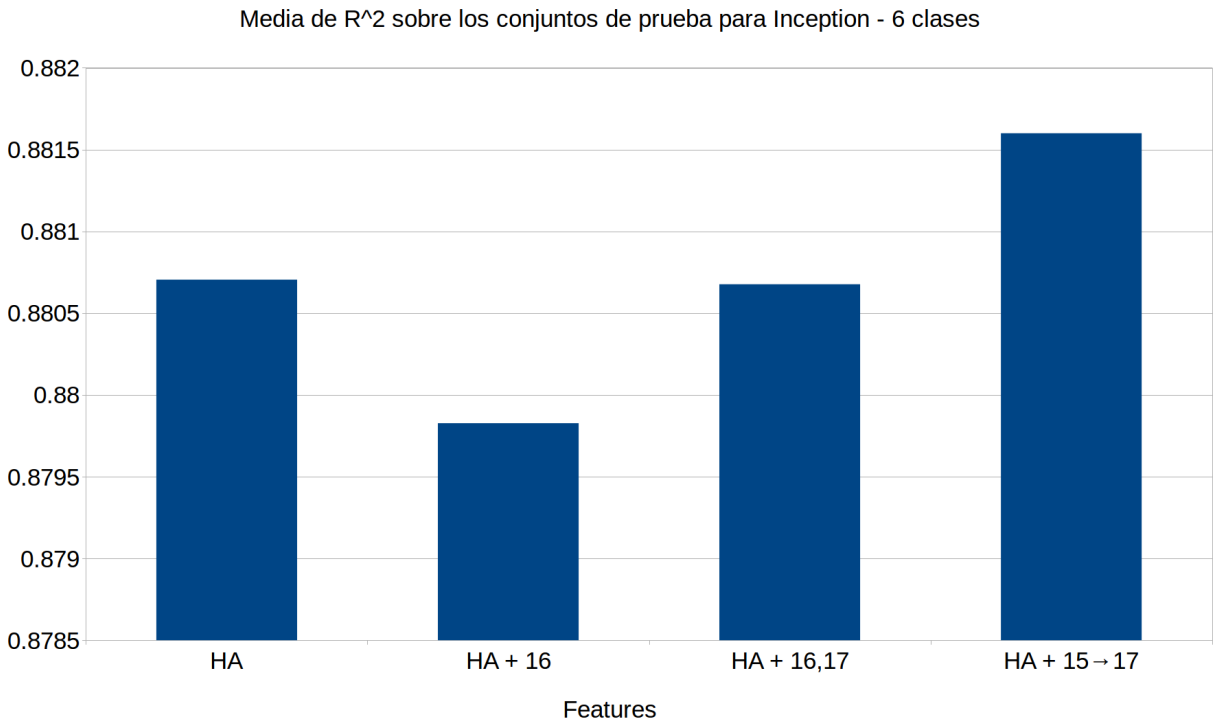


Figura 5.21: Media de R^2 sobre los conjuntos de prueba, para configuraciones de *features* de *Inception* entrenada para resolver una clasificación de 6 clases.

Capítulo 6

Modelos adicionales

En el capítulo anterior se determinó que la combinación de *House Attributes* con los *features* de tamaño 256, provenientes de *Inception*, para los niveles de *zoom* 15, 16 y 17 logran los mejores resultados entre las configuraciones estudiadas. El tamaño 256 de los *features* se obtiene a través de una capa *fully connected* sin activación, cuyo único objetivo es reducir la dimensionalidad de 1536 de la capa *conv2d_ 7b_ 1x1*.

Este capítulo explora configuraciones adicionales basadas en HA + 15,16,17 en las que se reduce de otras formas la dimensionalidad o se modifica el enfoque del problema, descritas a continuación.

6.1. Descripción

6.1.1. PCA

Los *features* de tamaño 256 obtenidos desde la red *Inception* se extraen de una capa *fully connected* agregada con el propósito de reducir la dimensionalidad. Esta capa se encuentra antes de *Logits* y luego de *conv2d_ 7b_ 1x1*, que corresponde a una capa de tamaño 1536. Vale la pena considerar si un método de reducción de dimensionalidad, como lo es *PCA*[26] (*Principal Component Analysis*), podría generar mejores *features* de tamaño 256, partiendo desde la salida de la capa de tamaño 1536.

Considerando que HA + 15,16,17 emplea 3 niveles de *zoom*, se dispone de 4608 *features* a reducir (1536 por cada nivel de *zoom* extrayendo desde *conv2d_ 7b_ 1x1*). Para realizar la reducción se consideran 2 posibilidades: Reducir los 4608 *features* a 768 o reducir cada salida de 1536 *features* a 256 y luego concatenar los resultados. La primera opción se denomina *PCA* 4608, mientras que la segunda se denomina *PCA* por nivel.

Los *features* obtenidos a través de *PCA* se concatenan con los *House Attributes*.

6.1.2. PLS

Otra opción explorada para reducir dimensionalidad es *PLS*[27] (*Partial Least Squares*). Los resultados que se obtienen desde los 4608 *features* o por nivel de *zoom* son idénticos, por lo que se considera *PLS* por nivel en los experimentos, ya que el tiempo de procesamiento es menor.

Al igual que en el caso anterior, se concatenan estos *features* con los *House Attributes*.

6.1.3. Estimación de densidad de precio por unidad de superficie

Es posible modificar el problema de regresión a la estimación de la densidad de precio por unidad de superficie de la propiedad, particularmente el tamaño del terreno. Al estimar la densidad de precio por unidad de superficie de una propiedad se puede recuperar la estimación del precio de venta utilizando el tamaño del terreno, que corresponde a uno de los *House Attributes*.

6.2. Resultados experimentales

Pese a que podría pensarse que escoger los 768 *features* más significativos desde el total de 4608 llevaría a representar de mejor forma las muestras, los experimentos revelan que reducir la dimensionalidad, utilizando la capa *fully-connected*, entrega un desempeño mayor que reducir a través de *PCA* o *PLS*.

En la Figura 6.1 se recopilan los resultados promedio de los experimentos clave para los conjuntos de validación y prueba, en los que se utiliza como regresor *Random Forest* con 200 estimadores.

Un aspecto interesante de mencionar respecto a la reducción de dimensionalidad con *PCA* es que el método es sensible a la escala de los atributos, por lo que es común estandarizar los datos antes de realizar la reducción. Esto se lleva a cabo en los experimentos. Esta transformación viene, en este caso, con penalizaciones, ya que, siendo *Random Forest* el regresor utilizado, se pierde la oportunidad de explotar la información adicional contenida en la diferencia de escala de los *features*. En otras palabras, *Random Forest* se beneficia, a diferencia de otros modelos de regresión, de *features* a escalas muy distintas. Una red neuronal *fully connected*, por ejemplo, suele funcionar peor cuando sus datos de entrada no se encuentran re-escalados.

Es posible apreciar en la Figura 6.1 que escoger, utilizando *PCA*, los *features* por nivel (*i.e:* seleccionar 256 *features* de los 1536 disponibles para cada nivel de zoom) produce mejores resultados que tomar los 768 más significativos del *pool* completo de 4608 elementos. La misma selección utilizando *PLS* otorga resultados marginalmente mejores en los conjuntos de prueba y marginalmente peores en los conjuntos de validación, pero ambos casos no

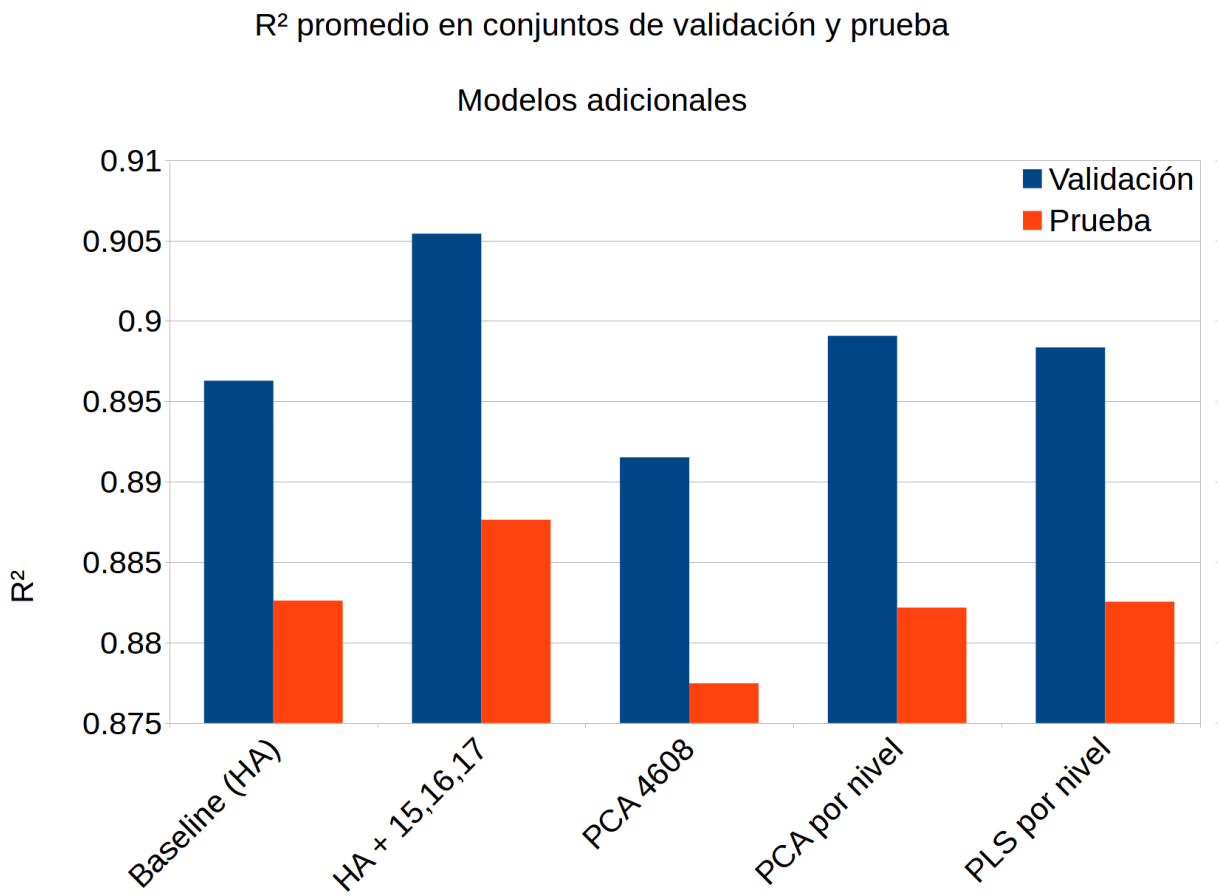


Figura 6.1: R^2 promedio en conjuntos de validación y prueba para los modelos adicionales.

logran superar el desempeño obtenido al utilizar los 768 *features* generados por la capa de reducción de dimensionalidad. Notamos, entonces, que existe un grado no despreciable de complementación entre los valores de los *features* generados.

Respecto a plantear el problema de forma distinta cambiando el *target* a la densidad de precio por unidad de terreno, los resultados, correspondientes a regresiones utilizando *Random Forest* con 200 estimadores, de la experimentación de este formato, resumidos en la Figura 6.2, muestran que la diferencia de R^2 promedio del mejor modelo y el *baseline*, en los conjuntos de validación y prueba, es despreciable.

Sin embargo, es necesario mencionar que una pequeña diferencia en los valores de densidad puede llevar a grandes diferencias de precio en los datos originales. Es posible contrastar el desempeño frente a este nuevo planteamiento del problema con lo logrado anteriormente (utilizando directamente los precios), calculando los precios en base a la densidad estimada y la información disponible de tamaño del terreno. Lo anterior muestra que este enfoque no es competitivo, siendo, por ejemplo, 0.854775 el R^2 equivalente para el *baseline HA* sobre los conjuntos de validación, lo que es menor a su valor de 0.896283 en la predicción directa del precio.

En conclusión, HA + 15,16,17 sigue siendo el modelo con mejor desempeño entre las opciones estudiadas.

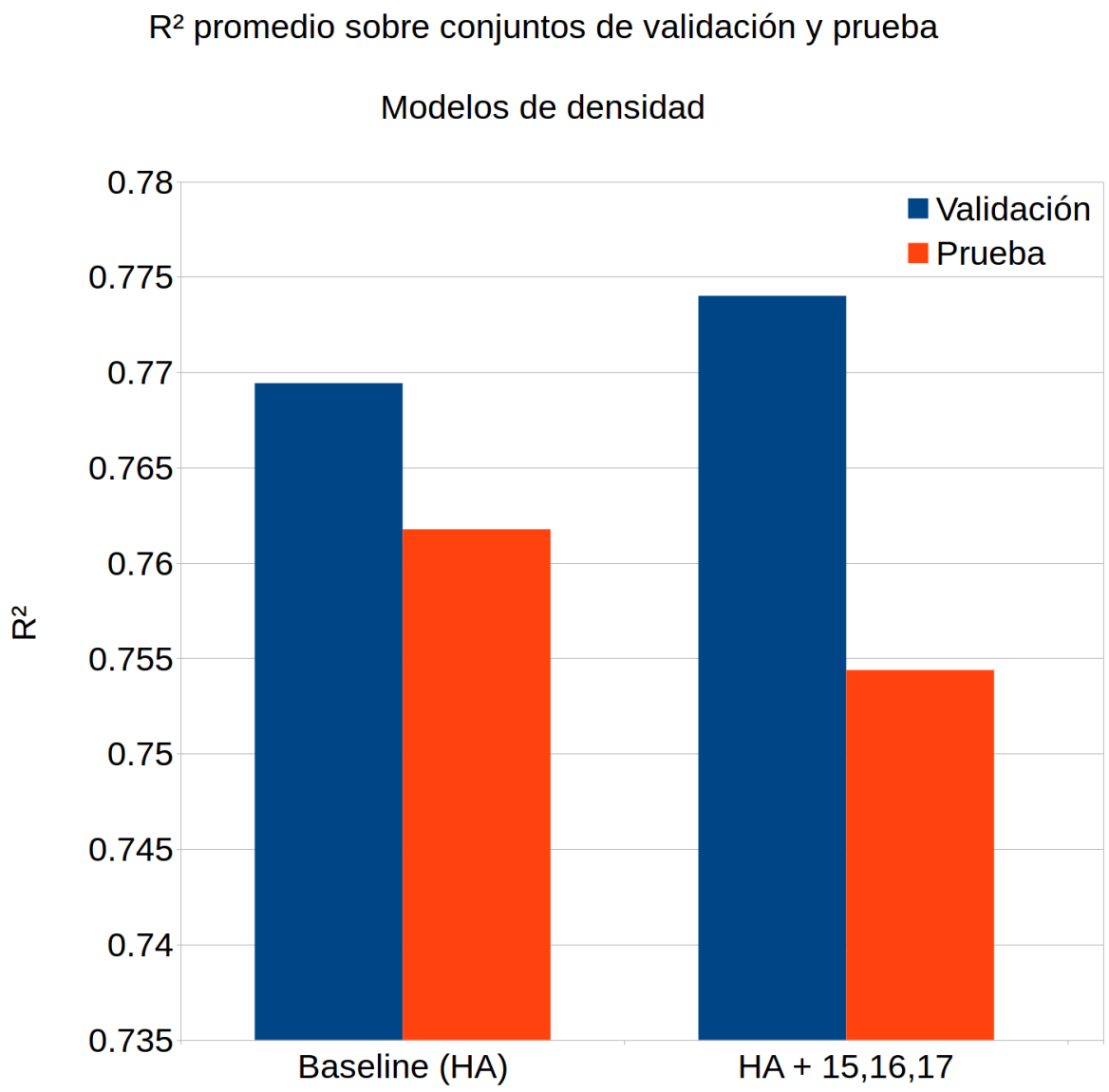


Figura 6.2: R^2 promedio para regresión de densidad de precio en conjuntos de validación y prueba. Mejor modelo vs *baseline*.

Capítulo 7

Modelo vs Baseline

Con el objetivo de contrastar más a fondo el desempeño del mejor modelo y el *baseline* se realizan pruebas adicionales sobre un nuevo conjunto de 13259 anuncios no utilizados en entrenamiento, validación o prueba tanto de las redes como de los regresores. Al contar con estos nuevos datos se procede a utilizar la totalidad del *dataset* original para entrenar los modelos de regresión y medir su desempeño sobre las 13259 nuevas muestras no utilizadas anteriormente.

Se considera la siguiente categorización en *bins* de las muestras, en base a sus precios:

1. [23,946,269 a 67,000,000): 1199 muestras.
2. [67,000,000 a 86,952,716): 1212 muestras.
3. [86,952,716 a 112,000,000): 1205 muestras.
4. [112,000,000 a 151,717,867): 1206 muestras.
5. [151,717,867 a 234,403,362): 1205 muestras.
6. [234,403,362 a 292,586,829): 1206 muestras.
7. [292,586,829 a 345,462,952): 1205 muestras.
8. [345,462,952 a 402,880,709): 1205 muestras.
9. [402,880,709 a 503,363,870): 1206 muestras.
10. [503,363,870 a 651,967,273): 1205 muestras.
11. [651,967,273 a 4,766,931,540]: 1205 muestras.

Se realizan 20 experimentos que consisten en entrenar un regresor *Random Forest* con 200 estimadores, utilizando la totalidad de las muestras originales (23877 anuncios), y evaluar su desempeño sobre los nuevos anuncios. Pese a que los datos de entrenamiento y prueba de estos experimentos no cambian, sigue siendo necesario realizar varios, ya que cambiar la semilla que utiliza el regresor produce resultados distintos.

Los experimentos realizados sobre los conjuntos de prueba en las secciones anteriores revelan que, en promedio, el mejor modelo (HA + 15,16,17) obtiene mejores resultados que el *baseline*, siendo esta diferencia menos marcada en los conjuntos de prueba que en los conjuntos de validación. Existen, sin embargo, particiones que producen resultados muy bajos

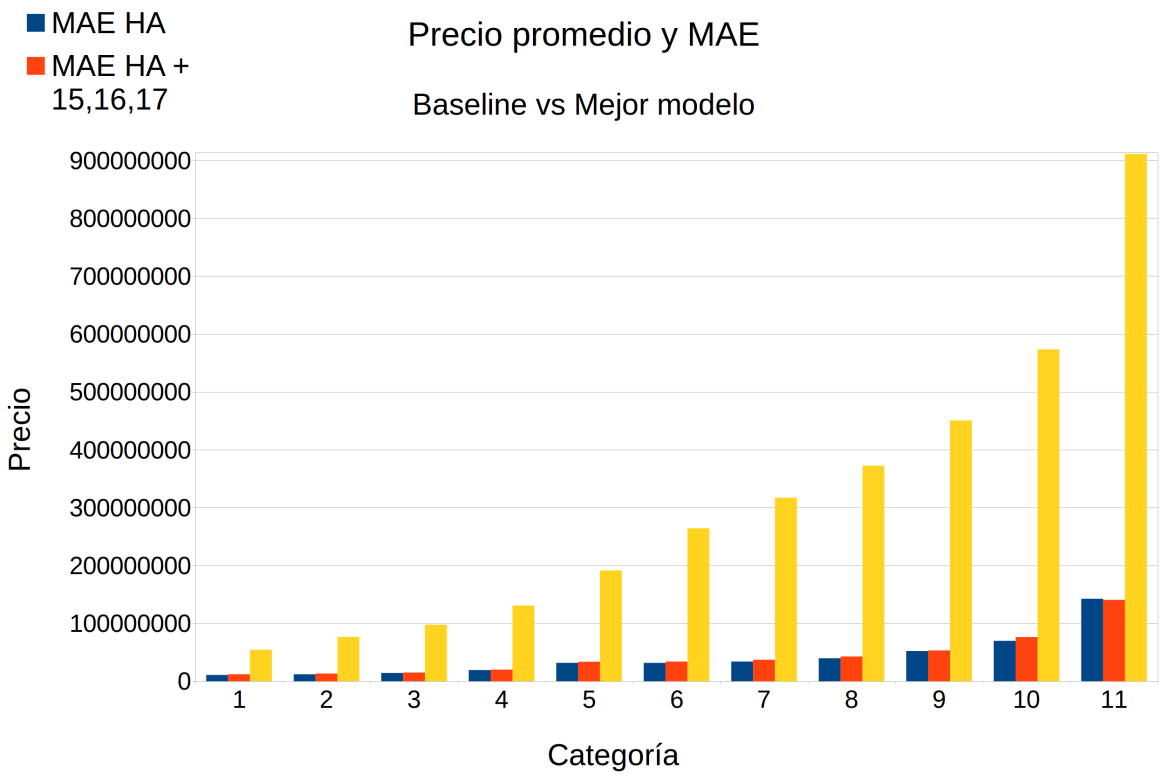


Figura 7.1: MAE por categoría, comparando *baseline* contra el mejor modelo, sobre nuevos anuncios. Incluye precio promedio por categoría.

de R^2 para el *baseline*, mientras que el mejor modelo presenta mayor estabilidad. En una de dichas particiones el valor de R^2 para HA en el conjunto es de 0.69372 y el *RMSE* es 129,391,899, mientras que para HA + 15,16,17 son 0.82557 y 97,646,913, respectivamente. Recordando que R^2 va de $-\infty$ a 1, siendo 1 una regresión perfecta, parece haber situaciones muy favorables para el mejor modelo.

Los valores promedio de *RMSE* y R^2, sobre los nuevos datos, son de 101,663,344 y 0.8535 para HA y de 94,975,402 y 0.8721 para HA + 15,16,17, respectivamente. En base a todos los resultados anteriores encontramos que el mejor modelo supera, consistentemente, al *baseline*.

Al considerar una nueva métrica llamada *Mean Absolute Error* o *MAE* surge una sutileza que vale la pena mencionar. El *MAE* se define como $MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$, con y_i el precio real de la muestra i y $\hat{y}_i$ la predicción del precio de la muestra i. El *MAE* es una métrica con una interpretación más sencilla que las anteriores, por lo que dará más claridad respecto a la diferencia entre ambos modelos. Los resultados promedio de *MAE* para HA y HA + 15,16,17, sobre los nuevos datos, son 41,617,075 y 43,437,081, respectivamente. En la Figura 7.1 se indican los valores de *MAE*, sobre los nuevos datos, para cada categoría, tanto para el *baseline* como para el mejor modelo. Las barras amarillas corresponden al precio promedio de los anuncios en cada categoría. Como se aprecia en la Figura 7.2, el *MAE* oscila entre el 10.64 % y el 22.18 % de los precios promedio de las categorías, siendo consistentemente menor en el *baseline*, excepto en la última categoría.

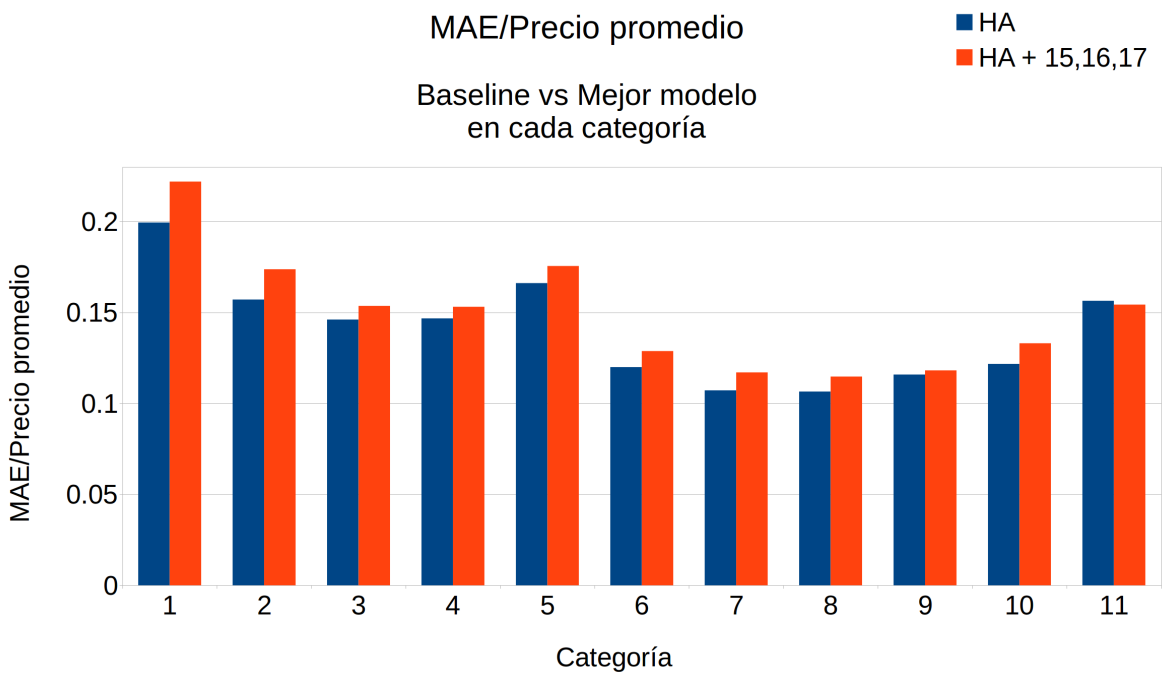


Figura 7.2: $\frac{MAE}{Precio\ promedio}$, comparando *baseline* contra el mejor modelo, para cada categoría, sobre los nuevos anuncios.

Haciendo un desglose del *RMSE* por categoría, como se aprecia en la Figura 7.3, notamos que la diferencia se va incrementando, a favor del mejor modelo, a medida que se avanza de categoría. Considerando que *RMSE* es una métrica que castiga de forma más fuerte errores con mayor magnitud (como lo son aquellos presentes en las categorías más caras) notamos que el mejor modelo toma ventaja en los valores de las casas más caras, mientras que el *baseline* tiene mejor desempeño en los valores de las casas más baratas.

Finalmente, en la Figura 7.4 es posible apreciar como las predicciones del mejor modelo (marcadas en color azul) se van acercando a la línea de precios reales (marcada en rojo) a medida que va aumentando el precio de venta.

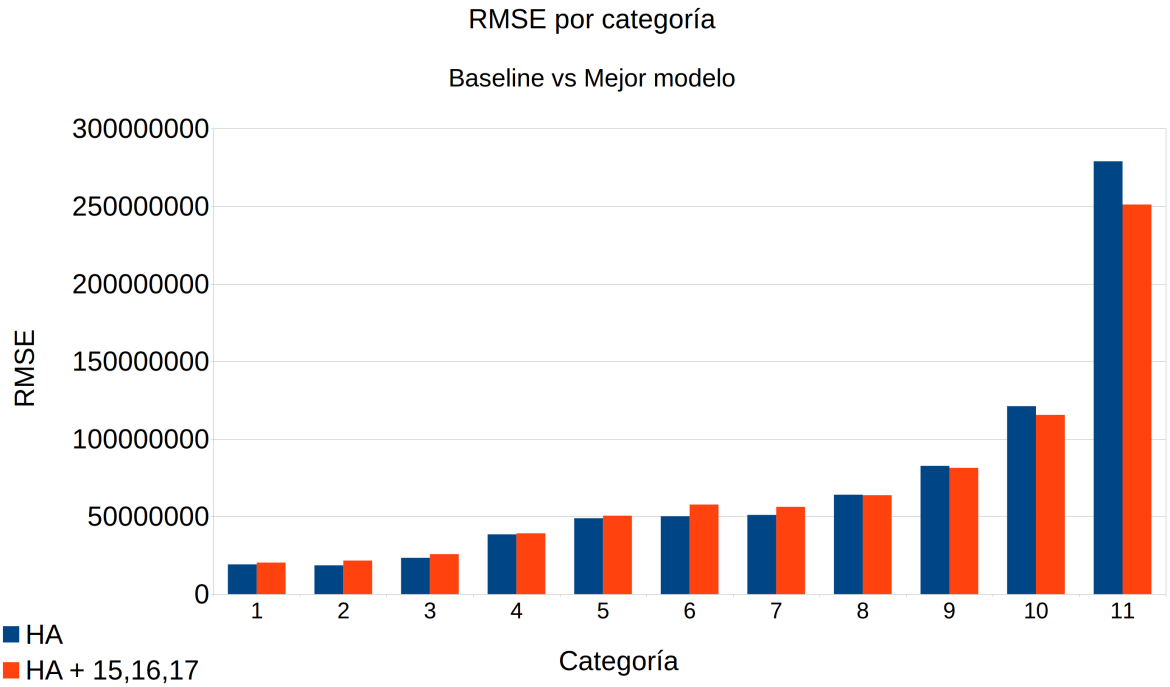


Figura 7.3: *RMSE* separado por categoría, comparando *baseline* contra el mejor modelo, sobre los nuevos anuncios.

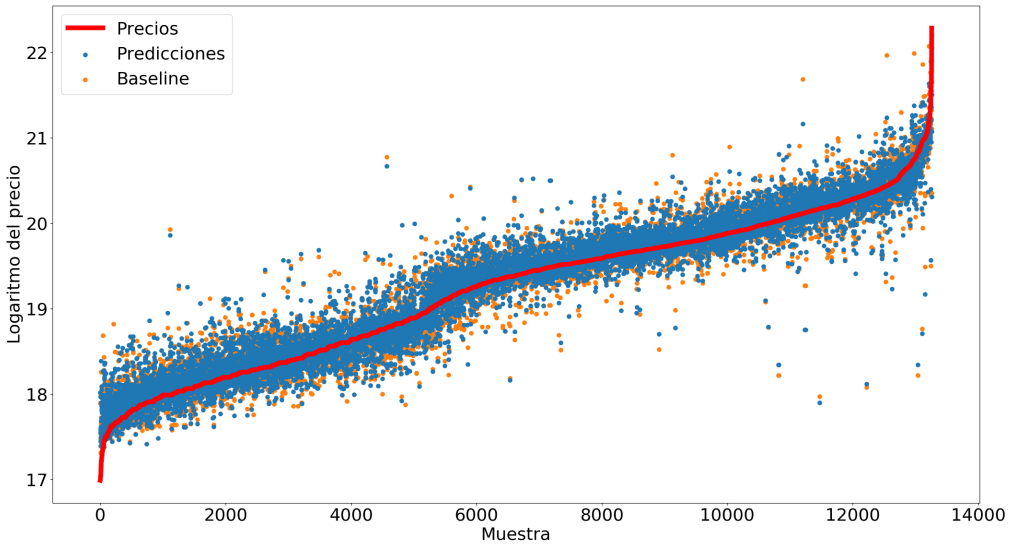


Figura 7.4: Scatter plot de logaritmo de predicción de los precios del mejor modelo, baseline y precios reales, sobre los nuevos anuncios.

Conclusión

Es posible generar *deep features*, desde imágenes satelitales, para caracterizar barrios y comunas de casas en la ciudad de Santiago, con el objetivo de utilizarlos para mejorar las capacidades de modelos de regresión que estimen el precio de venta de las propiedades. Utilizar estos *features* produce resultados cerca de un 8 % mejores, respecto al *RMSE*.

Para generar estos *deep features* se puede utilizar exitosamente la técnica del *transfer learning* sobre arquitecturas de redes convolucionales profundas diseñadas y entrenadas para resolver el problema de clasificación de *Imagenet*. No hace falta hacer *fine-tuning* de una extensa porción de las redes, la última capa de *flatten* anterior a la salida del clasificador demuestra ser suficiente. Se recomienda agregar una capa adicional con un menor número de neuronas, para reducir la dimensionalidad de los *features* producidos, cuidando que no cuente con función de activación para que represente una transformación lineal de la información de la capa de *flatten*. Una función de activación como $ReLU$ podría reducir a 0 información innecesaria para clasificar que podría servir en la regresión.

La información contenida en las imágenes satelitales, de alto nivel de *zoom*, no proporciona información complementaria útil para los regresores, a través de los *features* del problema de clasificación, y, de hecho, dificulta el problema. Esto puede deberse a que los factores que más aportan a determinar el precio de una propiedad son, en orden de importancia, el tamaño del terreno, su latitud y longitud. Una casa de características similares tendrá un precio muy diferente dependiendo de su ubicación en Santiago, por lo que en el rango de precios de 80.000.000 a 150.000.000 existe una gran cantidad de ruido por parte de los altos niveles de *zoom*: Son casas comparables que difieren mayormente en precio por su ubicación, siendo dicha ubicación mejor capturada por las imágenes más alejadas.

Un aspecto interesante sería probar el desempeño de estos *deep features* para la regresión de precios de casas de otras ciudades del país. Esto requeriría, sin embargo, expandir el *dataset* actual.

En base a los resultados experimentales obtenidos, se determina que se cumple el objetivo de construir un modelo de regresión de precios de venta de casas en la ciudad de Santiago, a través del uso de *Random Forest*, y se logra utilizar la información contenida en las imágenes satelitales centradas en la propiedad para aumentar el desempeño obtenido por el regresor, respecto a *Random Forest* entrenado utilizando, tan sólo, las características de las casas. Esta información, sin embargo, no se considera completamente explotada, existiendo mucho espacio para mejorar la detección y condensación de las características relevantes que determinan

el barrio en que se encuentran ubicadas las propiedades.

Dado lo anterior se plantea, como trabajo futuro, estudiar el desempeño de modelos de segmentación semántica sobre las imágenes satelitales para la regresión de precios. Existen, actualmente, arquitecturas como *U-Net*, que se prestan bien para el aprendizaje de segmentación de imágenes en base a pocas muestras, y pequeños *datasets* de identificación de carreteras y edificios en imágenes satelitales, tanto hiperespectrales como *RGB*, permitiendo, aquellos basados en imágenes *RGB* el entrenamiento de redes aplicables al *dataset* generado en el presente trabajo. Se espera que estos modelos puedan condensar de forma más efectiva la información relevante y permitir explotar de forma más provechosa los detalles contenidos en las imágenes con el fin de mejorar la predicción de los precios.

Bibliografía

[1] Portal inmobiliario. `https://www.portalinmobiliario.com/`.

[2] Goplaceit. `https://www.goplaceit.com`.

[3] Julia Núñez-Tabales, José Caridad, and F.J. Rey Carmona. Artificial neural networks for predicting real estate prices. In *Revista de Metodos Cuantitativos para la Economia y la Empresa*, volume 15, pages 29–44, 06 2013.

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

[6] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pages 1150–, Washington, DC, USA, 1999. IEEE Computer Society.

[7] Wenchao Jiang and Zhaozheng Yin. Human activity recognition using wearable sensors by deep convolutional neural networks. In *Proceedings of the 23rd ACM International Conference on Multimedia*, MM '15, pages 1307–1310, New York, NY, USA, 2015. ACM.

[8] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386, 1958.

[9] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–, October 1986.

[10] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Netw.*, 4(2):251–257, March 1991.

[11] Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, Contour and Grouping in Computer Vision*, pages 319–, London, UK, UK, 1999. Springer-Verlag.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for

image recognition. *CoRR*, abs/1512.03385, 2015.

[13] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014.

[14] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[15] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016.

[16] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *CoRR*, abs/1709.01507, 2017.

[17] Chenxi Liu, Barret Zoph, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan L. Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. *CoRR*, abs/1712.00559, 2017.

[18] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *CoRR*, abs/1411.1792, 2014.

[19] Archith J. Bency, Swati Rallapalli, Raghu K. Ganti, Mudhakar Srivatsa, and B. S. Manjunath. Beyond spatial auto-regressive models: Predicting housing prices with satellite imagery. *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 320–329, 2017.

[20] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001.

[21] Quanzeng You, Ran Pang, Liangliang Cao, and Jiebo Luo. Image-based appraisal of real estate properties. *IEEE Transactions on Multimedia*, 19(12):2751–2759, 2017.

[22] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[23] Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. Knowing when to look: Adaptive attention via A visual sentinel for image captioning. *CoRR*, abs/1612.01887, 2016.

[24] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on

heterogeneous systems, 2015. Software available from tensorflow.org.

[25] Modelos pre-entrenados para resolver imagenet en tensorflow. `https://github.com/tensorflow/models/tree/master/research/slim`.

[26] Ian Jolliffe and Jorge Cadima. Principal component analysis: A review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374:20150202, 04 2016.

[27] Jacob A. Wegelin. A survey of partial least squares (pls) methods, with emphasis on the two-block case. Technical report, 2000.