Máster en Investigación en Inteligencia Artificial

Visión Artificial

Memoria M0: Bloque II: Deep Learning

Autora:
Marta Rodríguez Sampayo

27 de enero de 2020

Índice

1. Materiales y métodos **3**
1.1. Materiales . . . 3
1.2. Tecnologías . . . 3
1.2.1. Creación de una CNN con Keras . . . 4
1.2.2. Monitorización del entrenamiento . . . 5
1.3. Redes Neuronales Convolucionales . . . 5
1.3.1. MobileNetv2 . . . 5
1.3.2. Modelo creado íntegramente . . . 6
1.3.3. VGG16 . . . 6
1.4. Implementación . . . 7
1.4.1. Localización de regiones de interés . . . 7
1.4.2. Reorientación de las figuras . . . 8
1.4.3. Localización de elementos concretos . . . 8

2. Resultados experimentales **9**
2.1. Localización de regiones de interés . . . 9
2.1.1. MobileNetv2 . . . 9
2.1.2. Modelo de CNN . . . 10
2.1.3. VGG16 . . . 11
2.1.4. Visión artificial tradicional . . . 12
2.2. Reorientación de las figuras . . . 13
2.3. Localización de elementos concretos . . . 14
2.3.1. Cruz exterior 1 - CNN modelada íntegramente . . . 14
2.3.2. Cruz exterior 1 - VGG16 . . . 15
2.3.3. Línea horizontal central - CNN modelada íntegramente . . . 17
2.3.4. Línea horizontal central - VGG16 . . . 18

3. Discusión **20**
3.1. Localización de regiones de interés . . . 20
3.2. Reorientación de las figuras . . . 20
3.3. Localización de elementos concretos . . . 21

4. Conclusiones **21**

Índice de figuras

1. Arquitectura de la red MobileNetv2 . . . 5
2. Arquitectura de la red . . . 6
3. Arquitectura de la red VGG16 . . . 6
4. Gráficas de la métrica y función de pérdidas (MobileNetv2) . . . 9
5. Representación de las *bounding boxes* estimadas (MobileNetv2) . . . 10
6. Gráficas de la métrica y función de pérdidas (Modelo de CNN) . . . 10
7. Representación de las *bounding boxes* estimadas (Modelo de CNN) . . . 11
8. Gráficas de la métrica y función de pérdidas (VGG16) . . . 11
9. Representación de las *bounding boxes* estimadas (VGG16) . . . 12
10. Representación de las *bounding boxes* estimadas (Tradicional) . . . 12
11. Gráficas de la métrica y función de pérdidas (VGG16) . . . 13
12. Gráficas de las métricas y función de pérdidas (Modelo CNN) . . . 14
13. Representación del punto central de la cruz estimado . . . 15
14. Gráficas de las métricas y función de pérdidas (Modelo CNN) . . . 16
15. Representación del punto central de la cruz estimado (VGG16) . . . 16
16. Gráficas de las métricas y función de pérdidas (Modelo CNN) . . . 17

17. Representación de la línea horizontal central estimada 18
18. Gráficas de las métricas y función de pérdidas (VGG16) 19
19. Representación de la línea horizontal central estimada (VGG16) 19

Índice de tablas

1. Informe de clasificación (VGG16) 13

1. Materiales y métodos

Las redes neuronales convolucionales (CNNs) se emplean para abordar problemas de clasificación y predicción aplicados a imágenes. A grandes rasgos, una CNN está formada por múltiples capas, cada una de las cuales se compone de neuronas que reciben datos de entrada y ejecutan ciertas funciones sobre ellos, obteniendo al final una salida dependiente del tipo de problema a resolver (distribución de probabilidad, valores numéricos relacionados con el problema, etc.). Durante su desarrollo se especifican un conjunto de hiperparámetros que determinan el comportamiento de la red. Los conceptos básicos y definiciones sobre redes neuronales convolucionales aplicados en la implementación que es objeto de esta memoria se han obtenido de la literatura recomendada para esta asignatura, más concretamente en [1] y [2].

En esta sección se explicarán en primer lugar los datos empleados, seguidos por las tecnologías utilizadas para el desarrollo de las soluciones, así como sus características más destacables y los métodos que se han empleado. Tras esta explicación general, se profundizará individualmente en la solución implementada para cada tarea, describiendo los modelos de redes convolucionales utilizados y su configuración específica.

1.1. Materiales

Los conjuntos de imágenes sobre los que se van a realizar las tareas propuestas son copias a mano de la figura de Rey (test neuropsicológico gráfico utilizado para la detección temprana y evaluación del deterioro cognitivo) y se dividen en tres directorios, cada uno de ellos con 887 muestras.

- REY_scan_anonim: copias de la figura, utilizadas en la primera tarea (extracción de regiones de interés).
- REY_roi_rot0: regiones de interés extraídas de las copias, utilizadas en la segunda tarea (clasificación según la orientación).
- REY_roi_manualselection1: regiones de interés rotadas para acercarse a la orientación del modelo, utilizadas en la tercera tarea (localización de elementos concretos).

En la sección 2 se especifica para cada una de las implementaciones la división de estos conjuntos para entrenar las redes y probarlas posteriormente.

Además de las imágenes, se dispone de dos archivos con extensión csv con las anotaciones necesarias para la implementación de las soluciones.

- traza_REY.csv: nombres de los archivos de los dos primeros directorios de imágenes con las coordenadas del rectángulo que encierra la región de interés y los ángulos de rotación de los dibujos.
- anotaciones1.csv: nombres de los archivos del tercer directorio y las coordenadas de los puntos representativos de cada elemento perteneciente a la figura.

La creación del conjunto de datos de entrada y las anotaciones es un paso fundamental en el desarrollo de cualquier sistema de *deep learning*, puesto que a partir de esta información se entrena el modelo y se realizan las predicciones. Para disponer de un número elevado de muestras y probar los modelos implementados para este proyecto, se decide emplear todas las imágenes con sus correspondientes anotaciones para cada tarea respectivamente. Aunque existe la posibilidad de que se hayan cometido errores en las anotaciones o que algunas imágenes no dispongan de la calidad suficiente, esto servirá para probar la robustez de las soluciones implementadas.

1.2. Tecnologías

La librería de *deep learning* más utilizada en Python es Tensorflow (https://www.tensorflow.org/), que permite implementar soluciones con distintos grados de abstracción. Existe mucho material disponible online sobre esta librería y dispone de una API de alto nivel llamada Keras (https://www.tensorflow.org/guide/keras?hl=es), cuyo uso es recomendable para una primera toma de contacto con redes neuronales. Por estos motivos, se utilizan estas tecnologías en la implementación de las distintas redes neuronales

convolucionales usadas para desarrollar las soluciones a las diferentes tareas propuestas.

Keras permite crear modelos de redes de forma intuitiva y trabajar (crear, modificar, congelar, etc.) con las distintas capas de forma sencilla. Además, dispone de varios modelos predeterminados [3], para realizar implementaciones sin tener que crear una arquitectura de red propia. También ofrece distintas opciones de funciones de pérdidas y métricas, pudiendo realizar una evaluación personalizada del entrenamiento de la red.

A mayores se emplea Scikit-learn, una librería de *machine learning* para Python, para realizar la evaluación (incluye varios métodos con este propósito) de las distintas redes neuronales implementadas y para tratar los datos de entrada (división de los sets, binarización de las etiquetas, etc.).

1.2.1. Creación de una CNN con Keras

Existen una serie de métodos y parámetros comunes a la elaboración de un modelo de red neuronal convolucional, independientes de la tarea a realizar. Se definen a continuación aquellos que resultan relevantes.

- Función de pérdidas: utilizada para optimizar el valor de los pesos calculados durante el entrenamiento [4]. Esta función se calcula entre el valor objetivo (datos de entrada) y el valor predicho y debe ser minimizada en cada iteración.
- Métrica: el objetivo de esta función es evaluar la precisión de las predicciones realizadas durante el entrenamiento.
- Algoritmo de optimización: en este contexto, los optimizadores son utilizados para encontrar un equilibrio entre tiempo de ejecución y calidad de los resultados obtenidos [5].
- Épocas: número de iteraciones durante el entrenamiento de la red.
- *Batch size*: el número de muestras empleado para entrenar la red en cada iteración.
- Forma de los datos de entrada: el tamaño de las imágenes de entrada a la red debe ser constante.
- Salida de la red: la última capa de la red debe adaptarse al tipo de salida deseada. Para problemas de clasificación, la salida serán las probabilidades correspondientes a cada una de las clases posibles; en cambio, en problemas de regresión, los datos de salida serán valores numéricos predichos en función de los datos de entrada.
- Función de activación: los detalles sobre la activación de las neuronas en este tipo de redes se encuentran de forma resumida en [6]. En lo referente a esta memoria, es suficiente con mencionar que se utiliza la función *softmax* si se desea un vector de probabilidades como salida (problemas de clasificación) y la función $ReLU$ en el resto de casos, puesto que son las más populares para este tipo de implementaciones.

Keras permite configurar los parámetros utilizados en la CNN de forma personalizada, pudiendo así adaptar estos valores al problema planteado.

En todas las redes desarrolladas se emplea como optimizador AdaM [7], ampliamente utilizado debido a que aumenta considerablemente la eficiencia del sistema. Una vez escogido el optimizador, se debe enviar como parámetro el factor de aprendizaje o *learning rate*. Básicamente, este parámetro controla cuánto varía el modelo en respuesta al error estimado cada vez que se actualizan los pesos. Es decir, con un valor de lr pequeño, los cambios efectuados en los pesos son menores, aumentando el tiempo de ejecución del entrenamiento y pudiendo llevar a un estancamiento en el aprendizaje; mientras que un valor demasiado grande, puede resultar en un aprendizaje subóptimo, de forma que convergerá a una mayor velocidad a un conjunto de pesos que puede no ser adecuado. La elección de este parámetro es crucial para el desarrollo de una red neuronal. El factor de aprendizaje es controlado por el optimizador y su valor inicial es de 0.001 sugerido por los autores en [7] para obtener buenos resultados utilizando AdaM.

1.2.2. Monitorización del entrenamiento

En este tipo de implementaciones es posible simplemente ejecutar el entrenamiento y esperar a que el modelo termine en el número de épocas especificado. Sin embargo, un número mayor de iteraciones no garantiza una mejora en los resultados, por tanto resulta deseable controlar los valores obtenidos en la métrica y función de pérdidas para finalizar el entrenamiento en condiciones óptimas. Para ello se han utilizado las siguientes clases proporcionadas por Keras:

- ModelCheckpoint: se guardan los pesos obtenidos durante el entrenamiento cuando el valor de una métrica seleccionada es máximo (o mínimo, dependiendo de la función escogida).
- EarlyStopping: si durante un número previamente especificado de épocas el valor de la métrica no mejora se detiene el entrenamiento.
- ReduceLROnPlateau: habitualmente los modelos se benefician del uso de un factor de aprendizaje adaptativo [8], por tanto si el aprendizaje no mejora durante un cierto número de épocas, se reduce el valor del lr y se continúa hasta un umbral mínimo especificado

Estos tres métodos se emplean en todas las implementaciones realizadas, variando la función de control según la solución desarrollada.

1.3. Redes Neuronales Convolucionales

Se han empleado tres modelos de red diferentes para el desarrollo de las soluciones de cada apartado. En esta sección se presentan las arquitecturas de cada uno de ellos.

1.3.1. MobileNetv2

La segunda versión de MobileNet [9] de Google, es un modelo de red sencillo, de bajo coste y reducido tamaño, adecuado para dispositivos con baja potencia computacional (p.ej.: móviles). La idea de utilizar este modelo se basa en el código del primer ejemplo del repositorio `https://github.com/lars76/object-localization`, adaptándolo al conjunto de datos proporcionado. La arquitectura de red se muestra en la Figura 1.

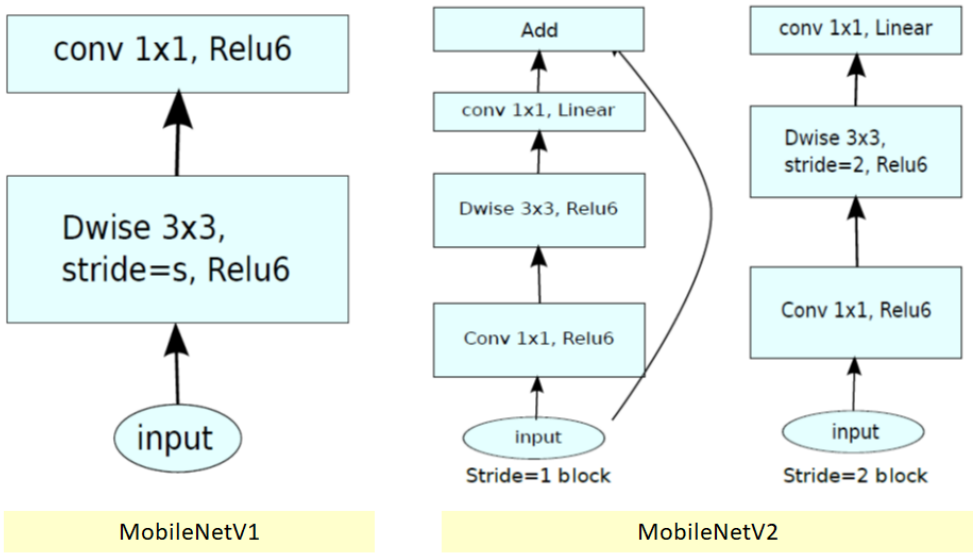


Figura 1: Arquitectura de la red MobileNetv2

- Configuración empleada en la implementación:
  - Forma de los datos de entrada: imágenes de tamaño 96x96.
  - Parámetro alfa: controla el ancho de la red. Si es menor que uno, reduce proporcionalmente el número de filtros en cada capa; si es mayor que uno, aumenta este número y, si es igual a 1 utiliza un número predeterminado de filtros. En este caso, se utiliza un valor de alfa de 1.0
  - *Batch size*: 32 muestras.
  - Épocas: valor inicial de 200 épocas.

1.3.2. Modelo creado íntegramente

En este caso se crea el modelo de red desde cero, concretamente el utilizado en [10] para la predicción de *bounding boxes*. Por cada capa convolucional, en lugar de especificar la función de activación, se añade una capa de activación LeakyReLU. Se muestra en la figura 2 un esquema sencillo de los bloques que conforman esta red.

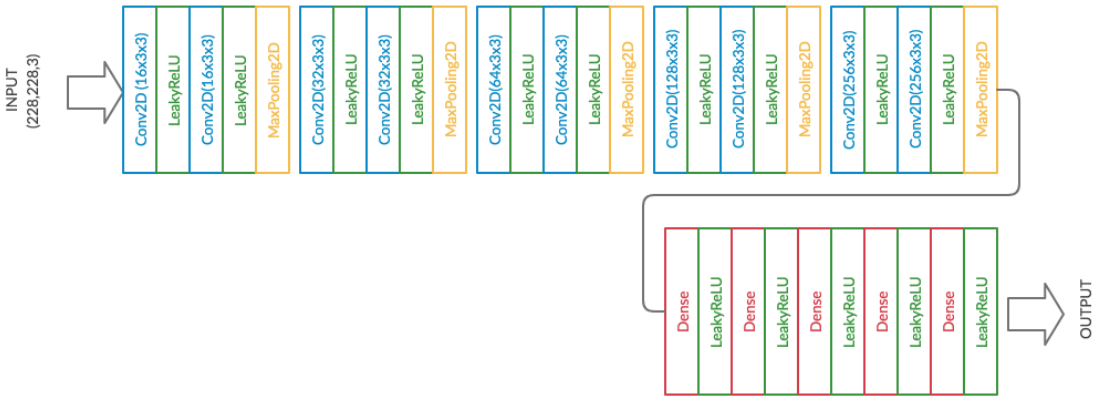


Figura 2: Arquitectura de la red

- Configuración empleada en la implementación:
  - Forma de los datos de entrada: imágenes de tamaño 228x228.
  - Parámetro alfa: 0.2
  - *Batch size*: 32 muestras.
  - Épocas: valor inicial de 100 épocas.

1.3.3. VGG16

VGG16 [11] es un modelo de red neuronal convolucional que alcanza el 92.7 % de precisión en el dataset de ImageNet (14 millones de imágenes pertenecientes a 1000 clases). Mejora el modelo AlexNet reemplazando filtros con un gran tamaño de kernel con múltiples filtros 3x3 uno después de otro. Las diferentes capas se pueden apreciar en la figura 3.

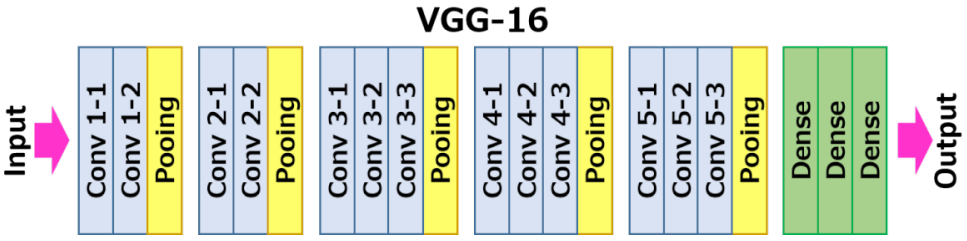


Figura 3: Arquitectura de la red VGG16

- Configuración empleada en la implementación:
  - Forma de los datos de entrada: imágenes de tamaño 224x224.
  - Número de etapas por época: 10.
  - Épocas: valor inicial de 100 épocas.

1.4. Implementación

A la hora de utilizar un modelo de red neuronal para una tarea en concreto, se deben adaptar sus parámetros de configuración a dicho problema. En las siguientes subsecciones se definen las funciones de pérdidas, métricas y tipos de salida utilizados para cada una de las arquitecturas explicadas anteriormente

A nivel de implementación, una diferencia fundamental entre utilizar *Transfer Learning* (uso de una red neuronal entrenada con unos datos y aplicada a otros [12]) o realizar el modelado y entrenamiento de la red desde cero, es la forma de enviar los datos de entrada a la primera capa de la red. En ambos casos es posible introducir directamente el conjunto de datos de entrenamiento, pero cuando se trata de un modelo pre-entrenado los tiempos de ejecución aumentan considerablemente (de 3 minutos por época a 56 minutos). Keras ofrece como solución a este problema la creación de un generador de datos que envía progresivamente los datos a la red y este es el método empleado con los modelos MobileNetv2 y VGG16.

1.4.1. Localización de regiones de interés

En la clasificación de objetos en imágenes, el primer paso es detectar en qué posición dentro de una imagen se encuentra el objeto en cuestión, para poder asignarle una etiqueta posteriormente. Este procedimiento se lleva a cabo calculando la sección rectangular mínima que engloba al objeto, conocida como *bounding box*.

Por tanto, el conjunto de imágenes con los que se entrena la red neuronal debe acompañarse de las coordenadas de dicho rectángulo (usualmente las coordenadas del punto superior izquierdo y el punto inferior derecho o el punto superior izquierdo, la altura y la anchura). Una vez finalizado el entrenamiento, el modelo debe ser capaz de localizar los objetos en nuevas imágenes.

La métrica más utilizada para evaluar modelos de predicción de *bounding boxes* es *Intersection Over Union* (IOU) [13]. Esta medida se calcula siguiendo la siguiente fórmula:

$$\text{IOU} = \frac{\text{Área de superposición}}{\text{Área de unión}}$$

donde dichas áreas se calculan entre las cajas predichas y las cajas anotadas manualmente. De esta forma es posible medir la calidad de los resultados obtenidos.

Puesto que el objetivo de este apartado es el cálculo de las cajas que encierran a los objetos, se trata de un problema de regresión, no de clasificación y, por ello, se ha utilizado como función de pérdidas el error cuadrático medio o MSE (*Mean Squared Error*), típicamente usado en este tipo de problemas. Se trata por tanto de reducir el valor de la siguiente función

$$MSE = \frac{1}{n}\sum_{t=1}^{n} e_t^2$$

donde e es el error entre la predicción y los datos de entrada.

Se han utilizado los tres modelos de red definidos en la sección 1.3. La última capa de las redes produce cuatro datos de salida, correspondientes a las coordenadas de la esquina superior izquierda y la esquina inferior derecha de la *bounding box* predicha.

Con el objetivo de realizar una comparación entre los métodos de *deep learning* y visión artificial tradicional, se ha realizado una implementación con la librería OpenCV para extrar las regiones de interés de las imágenes. El código está basado en [14] y consiste en binarizar una imagen en escala de grises, dilatarla utilizando un kernel (dependiente de si se desea afinar más o menos la detección, p.ej.: para encontrar regiones de interés de menor tamaño) y extraer los contornos y las *bounding boxes* de estos. Finalmente, se escoge la región de interés de mayor tamaño.

1.4.2. Reorientación de las figuras

La tarea que se propone resolver en este apartado es un problema de clasificación, es decir, la solución propuesta debe identificar la rotación a la que ha sido sometida cada figura. En este caso, las posibles clases son 0^{o}, 90^{o}, -90^{o} y 180^{o}.

Al tratarse de un problema con múltiples clases, la función de pérdidas escogida es la Entropía Cruzada Categórica (*Categorical Cross Entropy*). Cuando se incluye en los datos de entrada más de una clase, la salida de la red es un vector de probabilidades, donde la más alta se corresponde con la clase predicha. La entropía cruzada (https://en.wikipedia.org/wiki/Cross_entropy) es un método conocido para comparar distribuciones de probabilidad, siendo la entropía cruzada categórica un caso particular de aplicación de la misma. La CCE se define según la siguiente fórmula

$$CCE = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} 1_{y_i \in C_c} \log p_{model}[y_i \in C_c]$$

donde para un número N de observaciones, i representa cada una de ellas, c es una categoría de un conjunto de tamaño C y p_{model} es la probabilidad predicha para la observación i de pertenecer a la categoría c.

En cuanto a la métrica, se utiliza la precisión categórica o *categorical accuracy* que comprueba si el índice del máximo valor verdadero es igual al índice del máximo valor predicho. A esta se añaden varias funciones conocidas:

- Precisión y exhaustividad (*Precision and recall*): la precisión es la fracción de instancias relevantes entre las instancias recuperadas y la exhaustividad es la fracción de la cantidad total de instancias relevantes que han sido recuperadas.
- Valor F1 (*F1-Score*): se emplea para determinar un valor único ponderado de la precisión y la exhaustividad y se define como

$$F_1 = 2 \cdot \frac{\text{Precisión} \cdot \text{Exhaustividad}}{\text{Precisión} + \text{Exhaustividad}}$$

Para esta tarea se utiliza el modelo VGG16, puesto que los resultados obtenidos en la anterior son de mayor calidad, esta diferencia se mostrará en la sección 2. La última capa de la red debe utilizar como función de activación *softmax*, produciendo un vector de tamaño cuatro con las probabilidades calculadas para cada una de las clases (posibles ángulos de rotación).

1.4.3. Localización de elementos concretos

En este apartado, de nuevo nos encontramos con un problema de regresión, en el que la salida de la red neuronal no son las probabilidades de cada clase si no uno o varios valores numéricos, en este caso, representando los puntos que definen cada elemento.

Del mismo modo que en el apartado 1.4.1, se utiliza como función de pérdidas el MSE. Para evaluar los resultados se aplican las siguientes métricas:

- Error absoluto medio (*Mean Absolute Error*): sirve para cuantificar la precisión de una técnica de predicción comparando los valores predichos frente a los observados. Su definición es la siguiente:

$$MAE = \frac{1}{n}\sum_{t=1}^{n} |e_t|$$

- Coeficiente de determinación (*R squared*): determina la calidad del modelo para replicar los resultados, y la proporción de variación de los resultados que puede explicarse por el modelo. Se calcula según la siguiente fórmula:

$$R^2 = \frac{\sigma_{XY}^2}{\sigma_X^2 \sigma_Y^2}$$

Para esta tarea, se utilizan la red VGG16 y la CNN modelada para la tarea de extracción de las regiones de interés. La última capa de la red se configura con función de activación *ReLU* y predice las coordenadas del elemento a localizar, dependiendo su tamaño de cómo se defina este elemento (un punto, una línea, etc.).

2. Resultados experimentales

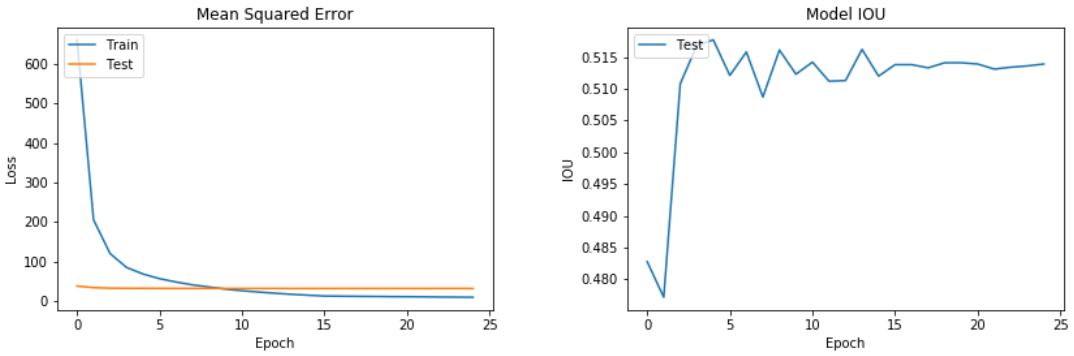
Para las CNNs propuestas en cada apartado se presentan los siguientes resultados:

- Gráfica de los valores de las funciones escogidas como métrica, para el set de entrenamiento y el set de validación.
- Gráfica de los valores de las funciones de pérdidas escogidas, también para ambos sets.
- Valores de las métricas adicionales empleadas.
- Imágenes con los elementos correspondientes (*bounding box*, punto o línea) estimados.

2.1. Localización de regiones de interés

2.1.1. MobileNetv2

- División manual del conjunto de datos de entrada:
  - Número de imágenes para el entrenamiento: 708
  - Número de imágenes para la validación: 179
- Valor máximo de IOU: 0.5177
- Número de épocas hasta finalización del entrenamiento: 25
- Tiempo de ejecución por época: 16 segundos
- Valor medio de IOU: 0.5014



(a) Función de pérdidas (MSE)

(b) Métrica (IOU)

Figura 4: Gráficas de la métrica y función de pérdidas (MobileNetv2)

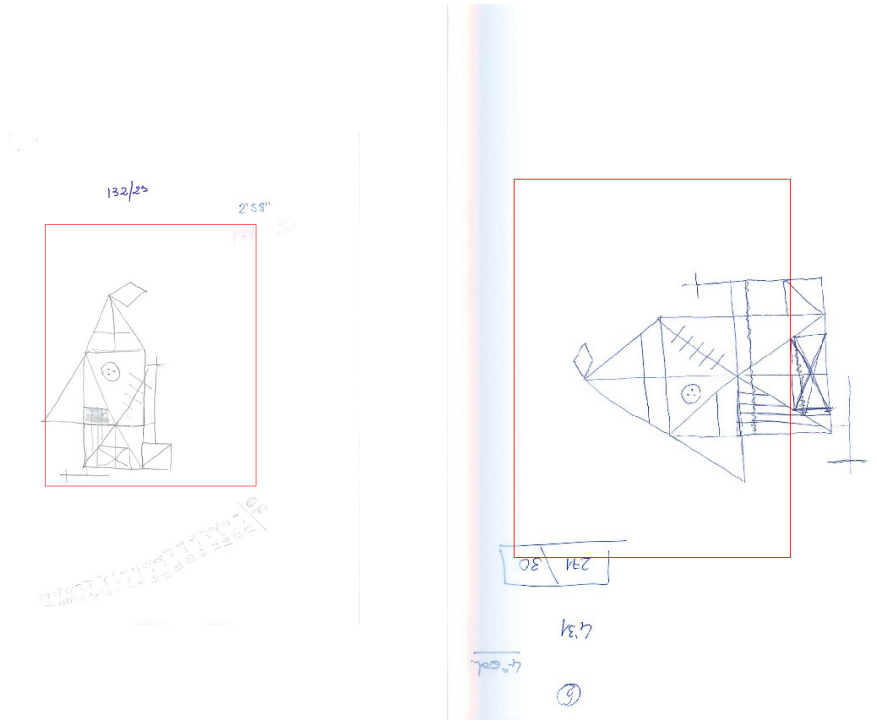
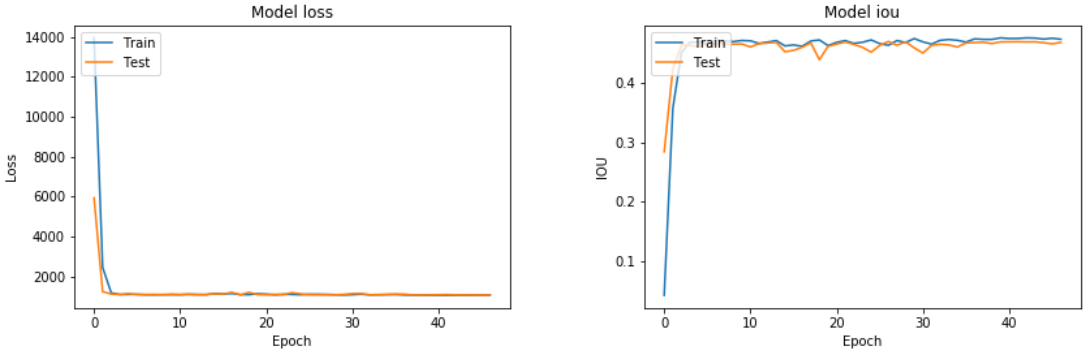


Figura 5: Representación de las *bounding boxes* estimadas (MobileNetv2)

2.1.2. Modelo de CNN

- División del conjunto de datos de entrada (25-75 %):
  - Número de imágenes para el entrenamiento: 665
  - Número de imágenes para la validación: 222
- Valor máximo de IOU: 0.4704
- Número de épocas hasta finalización del entrenamiento: 47
- Tiempo de ejecución por época: 45 segundos
- Valor medio de IOU: 0.4688



(a) Función de pérdidas (MSE)

(b) Métrica (IOU)

Figura 6: Gráficas de la métrica y función de pérdidas (Modelo de CNN)

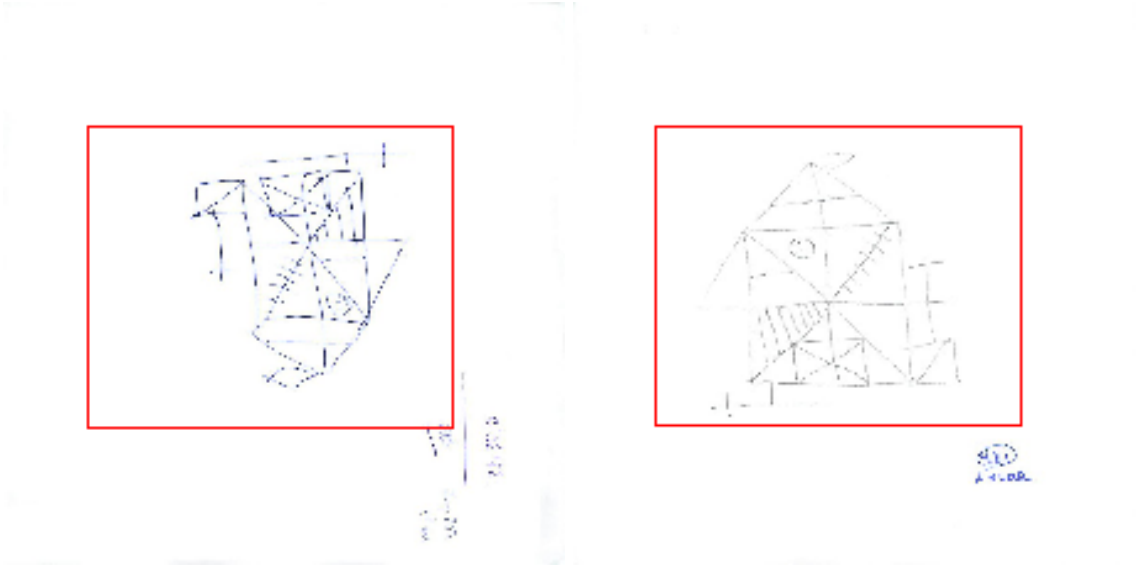
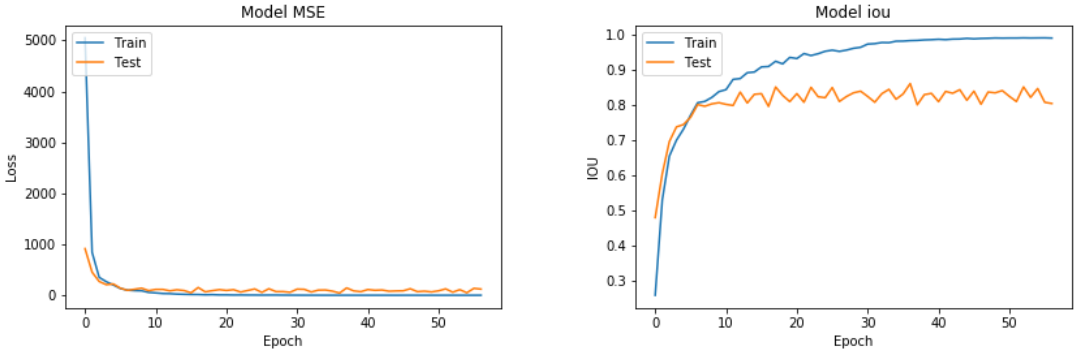


Figura 7: Representación de las *bounding boxes* estimadas (Modelo de CNN)

2.1.3. VGG16

- División del conjunto de datos de entrada (25-75 %):
  - Número de imágenes para el entrenamiento: 665
  - Número de imágenes para la validación: 222
- Valor máximo de IOU: 0.8624
- Número de épocas hasta finalización del entrenamiento: 57
- Tiempo de ejecución por época: 205 segundos
- Valor medio de IOU: 0.8355



(a) Función de pérdidas (MSE)

(b) Métrica (IOU)

Figura 8: Gráficas de la métrica y función de pérdidas (VGG16)

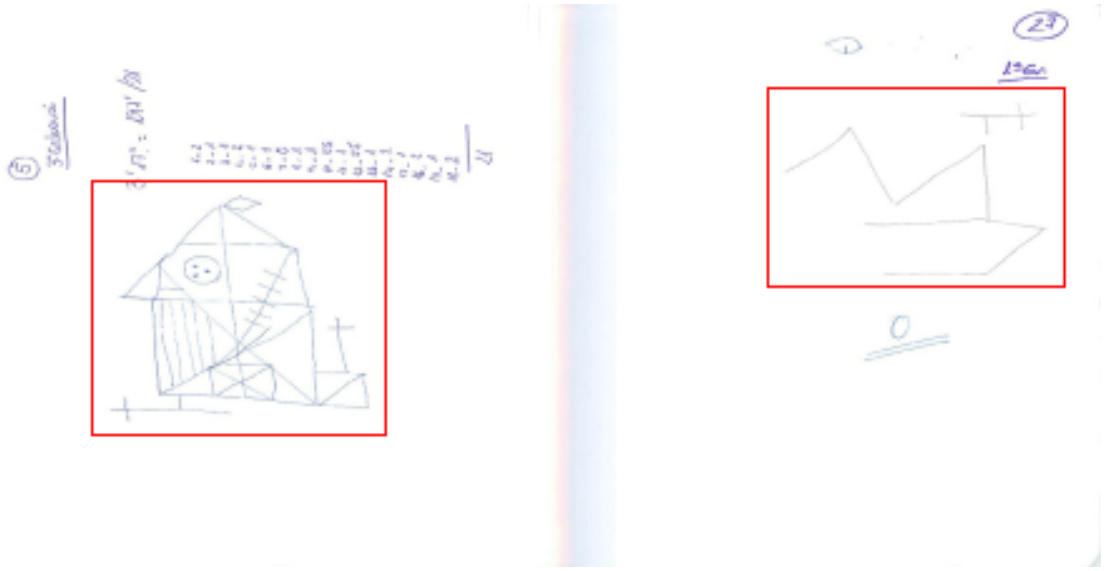


Figura 9: Representación de las *bounding boxes* estimadas (VGG16)

2.1.4. Visión artificial tradicional

- Se aplica sobre todo el conjunto de datos
- Valor medio de IOU: 0.8264

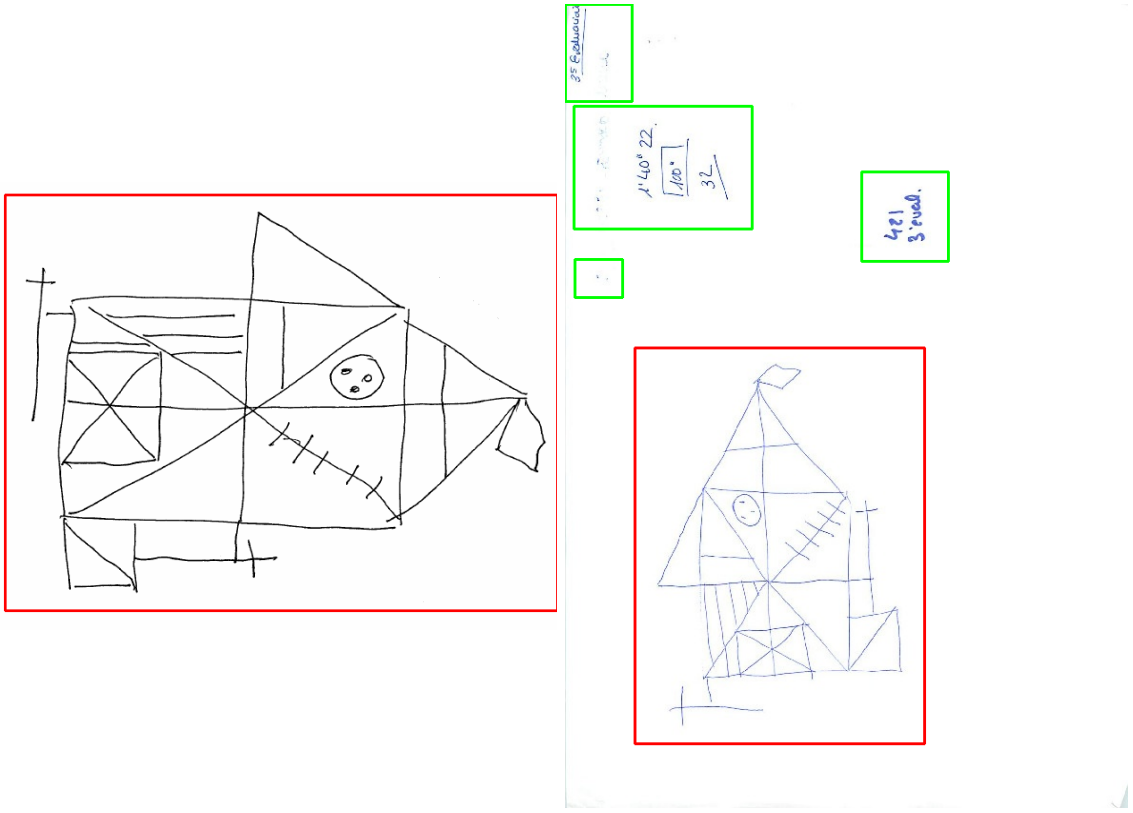
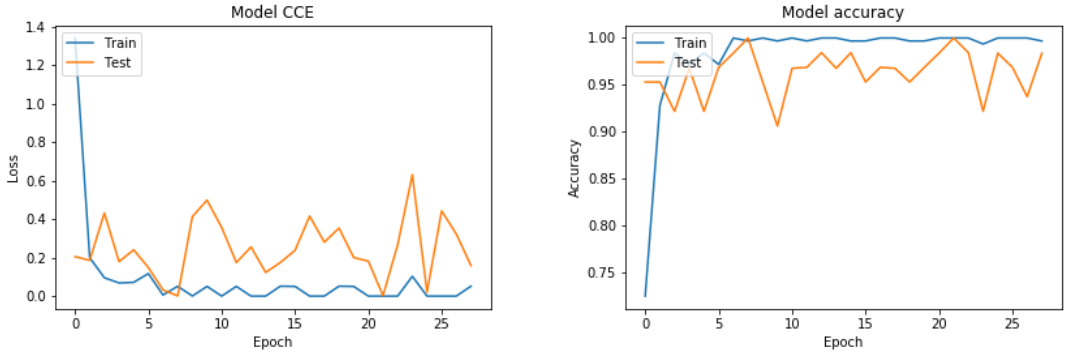


Figura 10: Representación de las *bounding boxes* estimadas (Tradicional)

2.2. Reorientación de las figuras

- División del conjunto de datos de entrada (25-75 %):
  - Número de imágenes para el entrenamiento: 665
  - Número de imágenes para la validación: 222
- Valor máximo de precisión categórica: 1.0
- Número de épocas hasta finalización del entrenamiento: 28
- Tiempo de ejecución por época: 205 segundos



(a) Función de pérdidas (CCE)

(b) Métrica (Precisión categórica)

Figura 11: Gráficas de la métrica y función de pérdidas (VGG16)

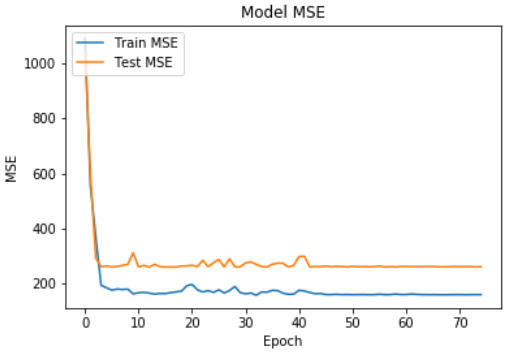
Clase	Precisión	Exhaustividad	Valor-F	Número de muestras
0	0.97	0.97	0.97	76
90	0.96	0.98	0.97	118
-90	1.00	0.95	0.97	20
180	1.00	0.75	0.86	8
Exactitud			0.97	222
Media	0.98	0.91	0.94	222
Media ponderada	0.97	0.97	0.97	222

Tabla 1: Informe de clasificación (VGG16)

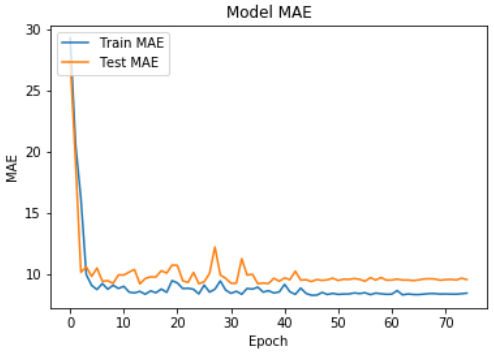
2.3. Localización de elementos concretos

2.3.1. Cruz exterior 1 - CNN modelada íntegramente

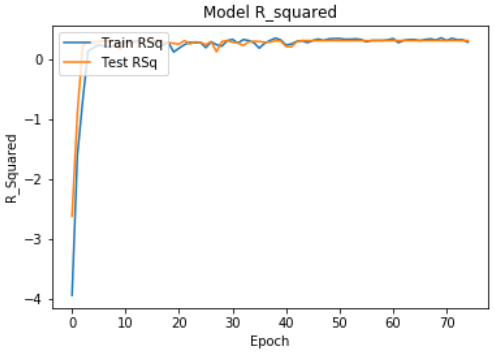
- División del conjunto de datos de entrada (30-70 %):
  - Número de imágenes para el entrenamiento: 200
  - Número de imágenes para la validación: 87
- Valor máximo del coeficiente de determinación: 0.31085
- Número de épocas hasta finalización del entrenamiento: 75
- Tiempo de ejecución por época: 15 segundos
- Valor medio del coeficiente de determinación: 0.2787
- Valor medio de MSE: 261.69
- Valor medio de MAE: 9.5629



(a) Función de pérdidas (MSE)



(b) Métrica 1 (MAE)



(c) Métrica 2 (Coeficiente de determinación)

Figura 12: Gráficas de las métricas y función de pérdidas (Modelo CNN)

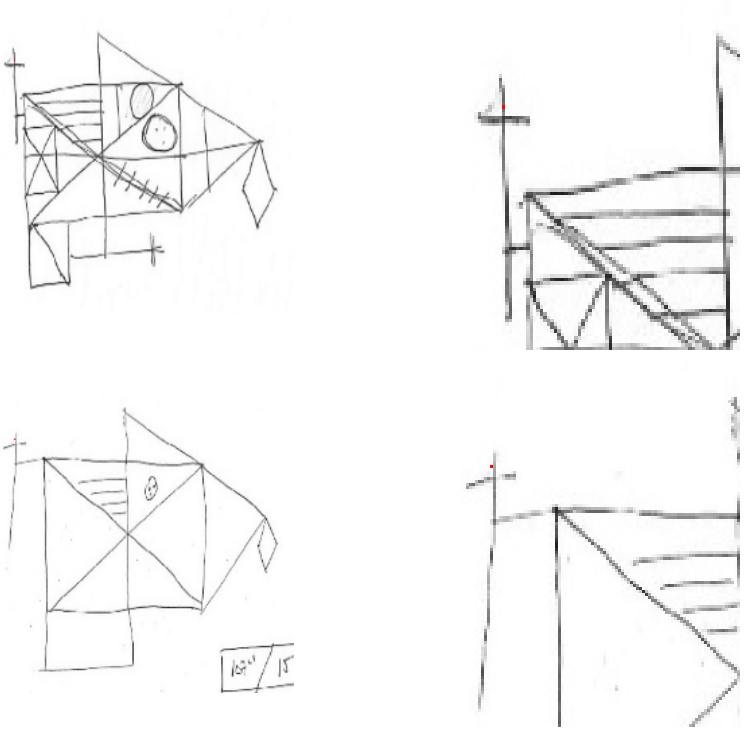
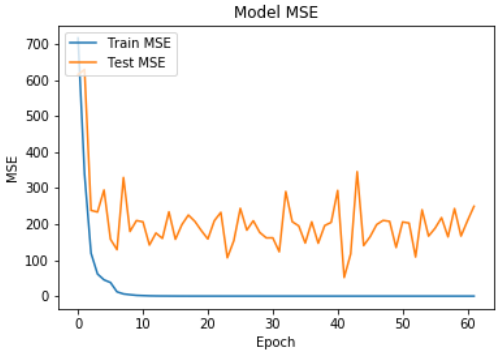


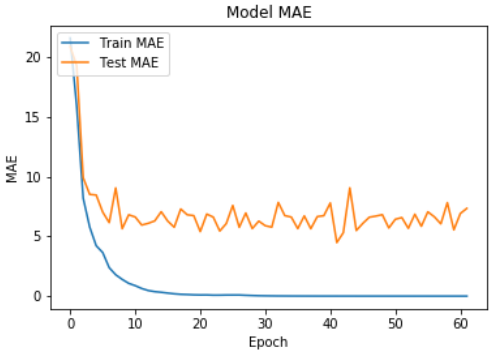
Figura 13: Representación del punto central de la cruz estimado

2.3.2. Cruz exterior 1 - VGG16

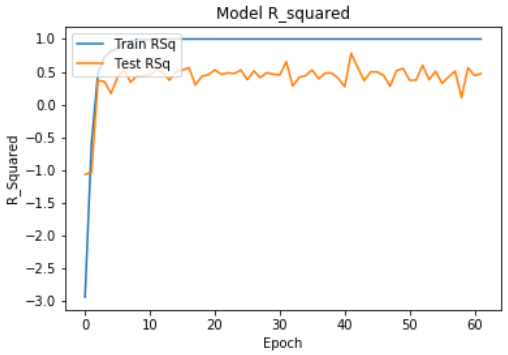
- División del conjunto de datos de entrada (75-25 %):
  - Número de imágenes para el entrenamiento: 216
  - Número de imágenes para la validación: 72
- Valor máximo del coeficiente de determinación: 0.78189
- Número de épocas hasta finalización del entrenamiento: 62
- Tiempo de ejecución por época: 205 segundos
- Valor medio del coeficiente de determinación: 0.5713468591372172
- Valor medio de MSE: 186.83
- Valor medio de MAE: 6.3976



(a) Función de pérdidas (MSE)



(b) Métrica 1 (MAE)



(c) Métrica 2 (Coeficiente de determinación)

Figura 14: Gráficas de las métricas y función de pérdidas (Modelo CNN)

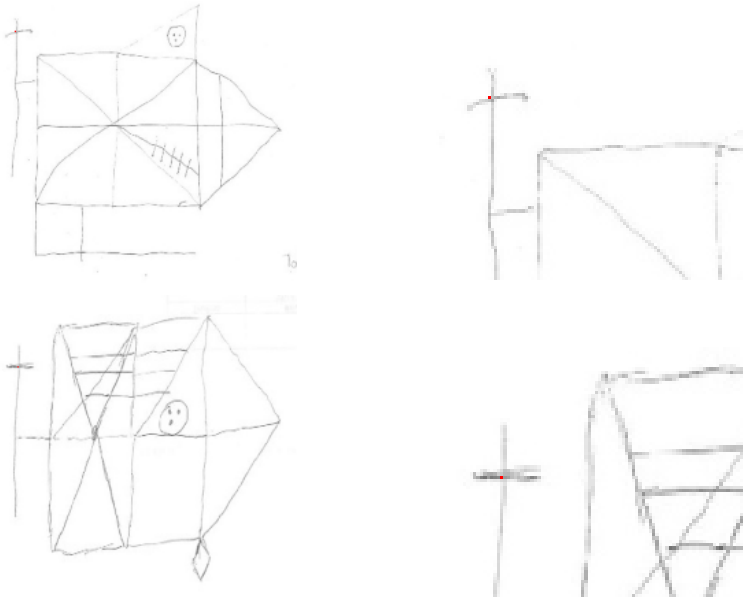
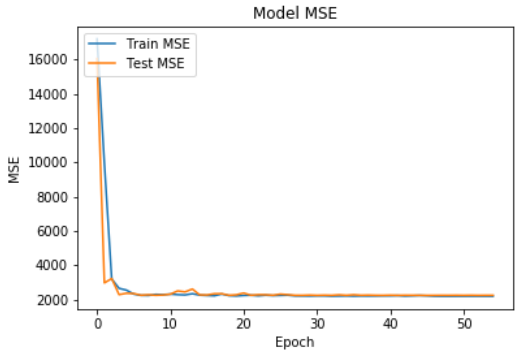


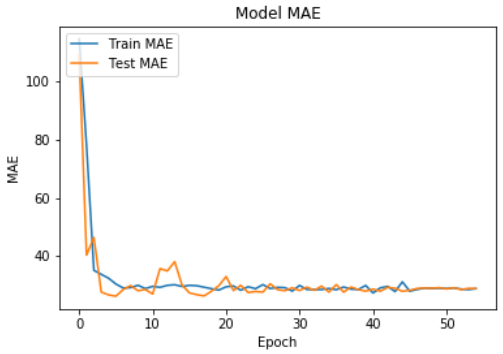
Figura 15: Representación del punto central de la cruz estimado (VGG16)

2.3.3. Línea horizontal central - CNN modelada íntegramente

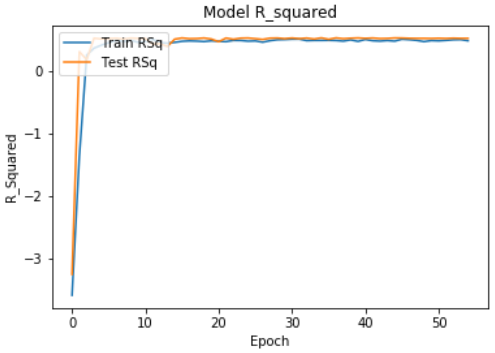
- División del conjunto de datos de entrada(30-70 %):
  - Número de imágenes para el entrenamiento: 207
  - Número de imágenes para la validación: 89
- Valor máximo del coeficiente de determinación: 0.52362
- Número de épocas hasta finalización del entrenamiento: 55
- Tiempo de ejecución por época: 15 segundos
- Valor medio del coeficiente de determinación: 0.4789
- Valor medio de MSE: 2245.82
- Valor medio de MAE: 28.694



(a) Función de pérdidas (MSE)



(b) Métrica 1 (MAE)



(c) Métrica 2 (Coeficiente de determinación)

Figura 16: Gráficas de las métricas y función de pérdidas (Modelo CNN)

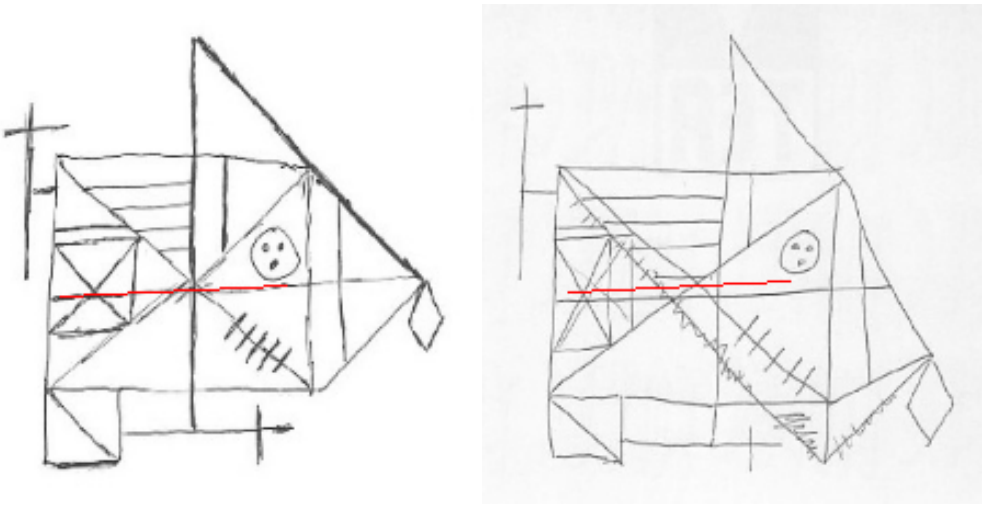
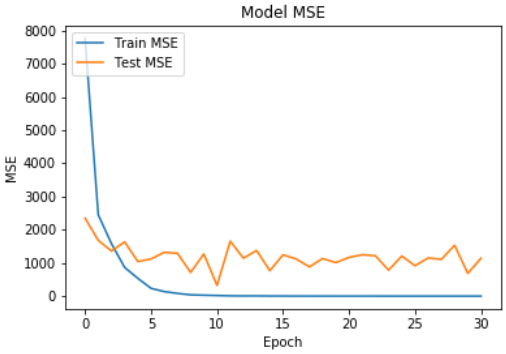


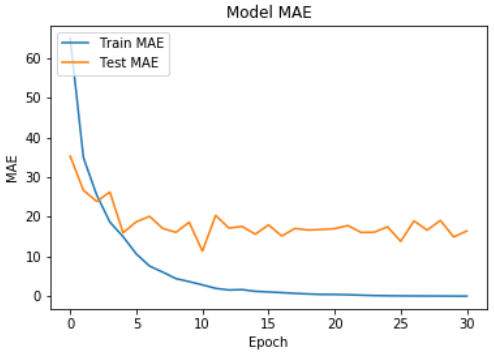
Figura 17: Representación de la línea horizontal central estimada

2.3.4. Línea horizontal central - VGG16

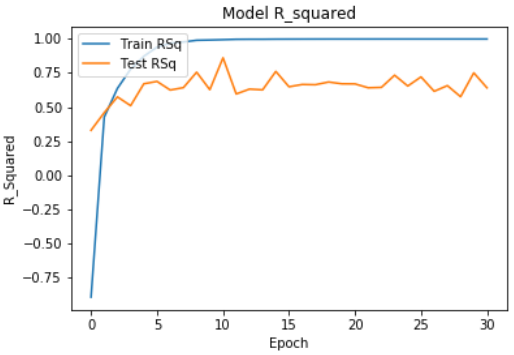
- División del conjunto de de entrada (25-75 %):
  - Número de imágenes para el entrenamiento: 222
  - Número de imágenes para la validación: 74
- Valor máximo del coeficiente de determinación: 0.8618
- Número de épocas hasta finalización del entrenamiento: 31
- Tiempo de ejecución por época: 205 segundos
- Valor medio del coeficiente de determinación: 0.6577
- Valor medio de MSE: 1103.34
- Valor medio de MAE: 16.453



(a) Función de pérdidas (MSE)



(b) Métrica 1 (MAE)



(c) Métrica 2 (Coeficiente de determinación)

Figura 18: Gráficas de las métricas y función de pérdidas (VGG16)

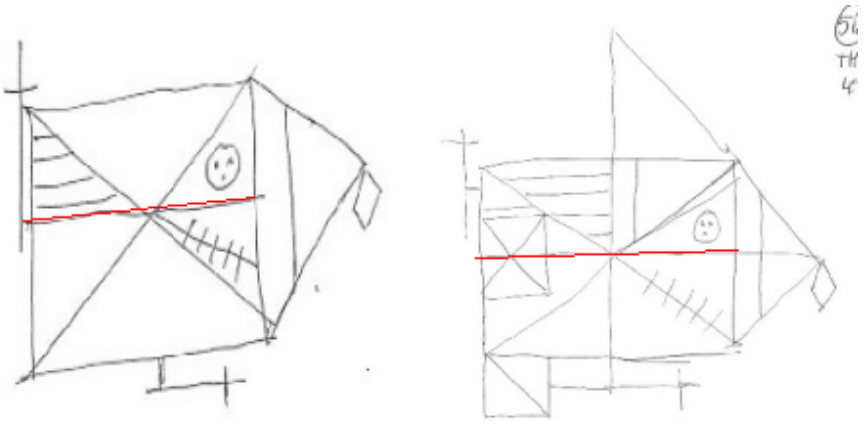


Figura 19: Representación de la línea horizontal central estimada (VGG16)

3. Discusión

3.1. Localización de regiones de interés

Además de evaluar los modelos de *deep learning*, se ha implementado un método de cálculo de regiones de interés con visión artificial tradicional, con el objetivo de poder realizar un análisis comparativo.

En cuanto a las CNN utilizadas, los mejores valores tanto de *mean squared error* como de *intersection over union* se obtienen con el modelo VGG16. Con esta red las pérdidas son prácticamente nulas y el valor medio de IOU es apróximadamente de 0.8, para el conjunto de datos de validación. La eficacia de las otras dos arquitecturas es similar, en media, la IOU obtenida se acerca a 0.5 en ambos casos, aunque las pérdidas son menores con MobileNetv2.

Los tiempos de ejecución y número de épocas de duración dependen completamente del equipo en el que se realizan los experimentos, por lo que no son indicadores reales del rendimiento y eficiencia de los modelos. Sin embargo, puesto que se ha empleado el mismo equipo para todas las ejecuciones, sí se puede realizar una comparación entre los valores obtenidos. De esta forma, el aumento de la calidad en la solución de la red VGG16 tiene como contrapartida una subida considerable del tiempo de ejecución. Mientras que con el modelo personalizado cada época del entrenamiento tarda aproximadamente 45 segundos en completarse, con VGG16 este valor aumenta hasta 205 segundos. A mayores, con el mismo tipo de monitorización, el entrenamiento con VGG16 finaliza a las 57 épocas, mientras que el modelo original finaliza en 47 épocas. Si no existe limitación temporal a la hora de resolver un problema real o se dispone de un equipo con mayor potencia (p.ej. con una GPU dedicada, ya que Keras permite la ejecución del entrenamiento en la GPU en lugar de la CPU) la mejor elección sería la arquitectura VGG16 con resultados mucho más precisos.

Sin embargo, el enfoque que utiliza visión artificial tradicional es el claro ganador en esta comparación. Con una IOU media de 0.83, prácticamente iguala a la red neuronal VGG16. Estos resultados demuestran que para esta tarea no es necesario utilizar técnicas más avanzadas como pueden ser las redes neuronales convolucionales, en las que se debe emplear tiempo de modelado, configuración y ejecución, ya que una aproximación más simple logra unos resultados de la misma o incluso mayor calidad, empleando un tiempo mucho menor entre implementación y ejecución.

Otra observación relevante acerca de las CNNs es el fenómeno denominado *overfitting*, consistente en la disminución de la eficacia del entrenamiento a medida que aumenta el número de épocas. Un claro ejemplo es la gráfica 8b que representa los valores de IOU de los conjuntos de entrenamiento y validación para el modelo VGG16. A partir de aproximadamente la época 30, la red deja de "aprender" y simplemente memoriza los valores correspondientes a cada imagen, por ello la IOU del set de entrenamiento sigue aumentando hasta 1, mientras que en el set de validación se estanca en el valor de 0.8. Esto demuestra por un lado que aumentar la duración del entrenamiento no es necesariamente beneficioso y la importancia de monitorizar los resultados obtenidos para detenerlo cuando los pesos obtenidos se adapten mejor a la solución.

3.2. Reorientación de las figuras

La red VGG16 está originalmente planteada para la resolución de problemas de clasificación de imágenes, por tanto, era de esperar que los resultados obtenidos fuesen altos para las métricas calculadas. En la tabla 1 se pueden ver los valores obtenidos en cada una de las métricas para cada clase. Se dispone de un número menor de muestras rotadas -90 y 180^{o} y en ambos casos la precisión obtenida es de 1.0, esta medida indica que no se han producido falsos positivos (etiquetado de una muestra como perteneciente a estas clases cuando no lo es) en la predicción, sin embargo, la exhaustividad es de 0.95 y 0.75 respectivamente para cada clase, por lo que sí hay falsos negativos (etiquetado de una muestra como no perteneciente a esta clase cuando sí lo es) entre los resultados. Sería posible aumentar el valor de exhaustividad para la clase de 180^{o} aumentando el número de muestras en el entrenamiento, sin embargo, los valores son suficientemente elevados como para no considerar necesario aumentar el dataset.

En cuanto a la función de pérdidas, la CCE cuantifica la diferencia entre las distribuciones de probabilidad, a la vista de la figura 11a se puede afirmar que este valor disminuye considerablemente en el set de entrenamiento y en ciertos picos del set de validación, coincidentes con los valores altos de precisión categórica (Figura 11b), por ejemplo en la época número 21 aproximadamente.

3.3. Localización de elementos concretos

De nuevo se utilizan la red VGG16 y la CNN modelada íntegramente para resolver estos problemas de regresión. En lugar de realizar un análisis por separado de la detección de cada elemento, se discutirán los resultados en conjunto, puesto que las características son muy similares, variando el número de coordenadas que se obtienen como salida de la red. En las figuras 15 y 13 se pueden ver dos imágenes de ejemplo con la zona ampliada en la que se encuentra el punto estimado por cada uno de los modelos. Las imágenes con la estimación de la línea horizontal central se muestran en las figuras 19 y 17.

La métrica que mejor sirve para evaluar la calidad de la predicción es el coeficiente de determinación o R cuadrado. Cuando este coeficiente vale cero, significa que el modelo proporciona los mismo resultados que un modelo que obtenga siempre como predicción la media de las muestras. Un valor negativo implica por tanto un rendimiento pésimo del modelo. En este caso, para la localización de ambos elementos, el valor de R cuadrado es positivo, indicando una eficacia aceptable. La red VGG16 obtiene nuevamente los resultados más altos, acercándose al valor ideal para este coeficiente y mostrando predicciones más acertadas.

En cuanto al MSE y MAE los valores descienden a medida que avanza el entrenamiento y se mantienen a la par entre el conjunto de datos de entrenamiento y el conjunto de datos de validación, sobretodo hacia las épocas finales, como se aprecia en las gráficas 12 y 16. Aún con este descenso, el error del modelo es significativo, con valores altos en ambas funciones. En cuanto a las gráficas 14 y 18 que muestran la ejecución del modelo VGG16, los valores de entrenamiento y validación no son tan cercanos, observándose de nuevo el efecto de *overfitting*, aún así esto no afecta al entrenamiento puesto que los pesos guardados y utilizados posteriormente en las pruebas se corresponden con los valores más altos obtenidos sobre el set de validación, gracias a una monitorización adecuada.

4. Conclusiones

Los métodos incluidos en el paradigma denominado *deep learning*, en particular las redes neuronales convolucionales que son objeto de esta memoria, están ganando popularidad en el campo de la visión artificial por la amplia gama de prestaciones que proporcionan. La comprensión del funcionamiento de las mismas, tanto el modelado de la arquitectura de red como la configuración de los hiperpárametros, es necesaria para adecuarlas al problema a resolver y obtener mejores resultados. Además de la implementación, los resultados están íntimamente relacionados con el tratamiento de los datos de entrada, por lo que crear un dataset limpio de imágenes y anotaciones es crucial para obtener un buen rendimiento del sistema. De los resultados obtenidos se puede deducir que el *transfer learning* resulta muy provechoso en algunos casos, utilizando redes pre-entrenadas sobre un nuevo conjunto de datos se pueden conseguir mejores resultados que diseñando una solución de forma específica.

Sin embargo, no siempre resulta beneficioso implementar este tipo de soluciones, que requieren un coste computacional elevado y no garantizan la calidad de los resultados. Un ejemplo de esto se presenta en la sección 3.1, donde se discuten los resultados de las distintas soluciones implementadas para la localización de regiones de interés. Para este caso, el modelo de visión artificial tradicional, de una complejidad mucho menor que las CNNs resulta mucho más eficiente, consiguiendo mejores predicciones en un tiempo de ejecución mínimo en comparación a los otros modelos.

En resumen, los experimentos recogidos en esta memoria prueban la eficacia de ciertas técnicas de *deep learning* para resolver problemas sencillos de regresión y clasificación de imágenes, utilizando métricas conocidas para la evaluación de los resultados obtenidos y cómo afectan las diferentes configuraciones de las redes neuronales convolucionales al rendimiento y eficacia de este tipo de sistemas.

Referencias

[1] "CS231n Convolutional Neural Networks for Visual Recognition." [Online]. Available: http://cs231n.github.io/convolutional-networks/

[2] A. Saxena, "Convolutional Neural Networks(CNNs): An Illustrated Explanation," 2016. [Online]. Available: https://blog.xrds.acm.org/2016/06/convolutional-neural-networks-cnns-illustrated-explanation/

[3] "Keras Applications." [Online]. Available: https://keras.io/applications/

[4] S. Verma, "Understanding different Loss Functions for Neural Networks," 2019. [Online]. Available: https://towardsdatascience.com/understanding-different-loss-functions-for-neural-networks-dd1ed0274718

[5] R. Agrawal, "Optimization Algorithms for Deep Learning," 2019. [Online]. Available: https://medium.com/analytics-vidhya/optimization-algorithms-for-deep-learning-1f1a2bd4c46b

[6] H. Sharma, "Activation Functions : Sigmoid, ReLU, Leaky ReLU and Softmax basics for Neural Networks and Deep Learning," 2019. [Online]. Available: https://medium.com/@himanshuxd/activation-functions-sigmoid-relu-leaky-relu-and-softmax-basics-for-neural-networks-and-deep-8d9c70eed91e

[7] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization," 2018. [Online]. Available: https://arxiv.org/abs/1412.6980v9

[8] J. Brownlee, "Understand the Impact of Learning Rate on Neural Network Performance," 2020. [Online]. Available: https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/

[9] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation," *CoRR*, vol. abs/1801.04381, 2018. [Online]. Available: http://arxiv.org/abs/1801.04381

[10] S. Panchal, "Getting Started With Bounding Box Regression In TensorFlow," 2019. [Online]. Available: https://towardsdatascience.com/getting-started-with-bounding-box-regression-in-tensorflow-743e22d0ccb3

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv 1409.1556*, 09 2014.

[12] J. Brownlee, "A Gentle Introduction to Transfer Learning for Deep Learning," 2017. [Online]. Available: https://machinelearningmastery.com/transfer-learning-for-deep-learning/

[13] A. Rosebrock, "Intersection over Union (IoU) for object detection," 2016. [Online]. Available: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

[14] "Extract roi from image with pyhton and opencv." [Online]. Available: https://cvisiondemy.com/extract-roi-from-image-with-python-and-opencv/