

UNIVERSIDAD DE ALMERIA

ESCUELA SUPERIOR DE INGENIERÍA

Aplicación de técnicas de aprendizaje automático y visión artificial en seguidores solares.

Curso 2020/2021

Alumno/a:

Pablo Dahl Cruz

Director/es:

José Antonio Carballo López
Manuel Berenguel Soria



UNIVERSIDAD DE ALMERÍA
ESCUELA SUPERIOR DE INGENIERÍA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL



TRABAJO FIN DE GRADO

Aplicación de técnicas de aprendizaje automático y visión artificial en
seguidores solares.

Alumno: Pablo Dahl Cruz

Director: Manuel Berenguel Soria

Codirector: José Antonio Carballo López

Fecha: Diciembre 2020

ÍNDICE

AGRADECIMIENTOS.....	6
ACRÓNIMOS.....	8
ÍNDICE DE FIGURAS	10
ÍNDICE DE TABLAS	12
RESUMEN	14
ABSTRACT.....	16
1. INTRODUCCIÓN.....	18
1.1 Motivación del trabajo.....	18
1.2 Objetivos	19
1.3 Contexto.....	20
1.4 Resultados.....	20
1.5 Estructura del trabajo.....	20
1.6 Competencias utilizadas en el TFG.....	21
2. REVISIÓN BIBLIOGRÁFICA.....	22
2.1 Energía solar de concentración (CSP).....	22
2.2 Sistemas de receptor central	25
2.3 Heliostatos.....	30
2.3.1 Partes de un heliostato	30
2.3.2 Seguimiento	32
2.3.3 Métodos de calibración y sistemas de seguimiento	33
2.3.4 Evolución	34
2.3.5 Visión artificial en CSP	35
2.4 Sistemas de seguimiento solar.....	37
2.4.1 Requerimientos de seguidores basados en visión artificial.	39
2.5 Inteligencia artificial en energía solar.	39
2.5.1 Redes neuronales.....	40
2.5.2 Entrenamiento o aprendizaje.....	42
2.5.3 Redes Neuronales Convolucionales	45
2.6 Visión artificial, inteligencia artificial y seguimiento solar en energía solar.	48
2.7 Software y librerías disponibles de IA.....	49

2.8	Detección de objetos y segmentación semántica.....	50
2.8.1	SSD (Single Shot MultiBox Detector).....	52
2.8.2	Arquitecturas de redes neuronales.....	54
2.9	Software para etiquetado	55
2.10	Configuración del hardware, software y datos	56
3.	MÉTODOS.....	58
3.1	Configuración de parámetros	58
3.1.1	Parámetros SSD	59
3.1.2	Técnicas de optimización de parámetros.....	61
3.2	Visualización de los resultados y Tensorboard:	62
4.	RESULTADOS Y DISCUSIÓN.....	68
4.1	Elección de modelos.....	68
4.2	Entrenamiento de los modelos	70
4.2.1	Entrenamientos <code>ssd_inception_v2_coco</code>	71
4.2.2	Entrenamientos <code>ssd_mobilenet_v1_coco</code>	89
4.3	Validación de los modelos.....	107
4.4	Estudio del modelo óptimo	115
5.	CONCLUSIONES Y FUTUROS TRABAJOS	118
	REFERENCIAS.....	120

AGRADECIMIENTOS

En primer lugar agradecer toda la ayuda que me han brindado mis tutores Manuel Berenguel, Javier Bonilla; y en especial, a José Antonio Carballo por toda su ayuda y todas las dudas que me ha aclarado.

A mis padres y a la gente con la que he pasado esta maravillosa etapa, que con la entrega de este trabajo, llega a su fin.

A la Universidad de Almería y a la Escuela Superior de Ingeniería.

ACRÓNIMOS

AP – *Average Precision*

CETA – Centro Extremeño de Tecnologías Avanzadas

CIEMAT – Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas

CNN – *Convolutional Neural Network*

CSP – *Concentrated Solar Power*

COCO – *Common Objects in Context*

FOV – *Field Of View*

GPU – *Graphics Processing Unit*

IA- Inteligencia Artificial

IEA – *International Energy Agency*

IoU – *Intersection over Union*

LCOE – *Levelized Cost Of Energy*

MAPE - *Mean Absolute Percentage Error*

MSE - *Mean Square Error*

MWe – MegaWatio eléctrico

PNIEC - Plan Nacional Integrado de Energía y Clima

PSA – Plataforma Solar de Almería

R-CNN – *Region Convolutional Neural Network*

ReLu – *Rectified Linear Unit*

SAM – *System Advisor Model*

SGD – *Stochastic Gradient Descendent*

SSD – *Single Shot Detection*

UAL – Universidad de Almería

USD – *United States Dollar*

ÍNDICE DE FIGURAS

Figura 2.1. Planta de tecnología de receptor central. CESA-PSA	22
Figura 2.2. Principales tecnologías CSP	23
Figura 2.3. Proyectos CSP a nivel mundial.....	25
Figura 2.4. Ejemplo de la tecnología CSP de torre central	26
Figura 2.5. Esquema receptor de cavidad.....	27
Figura 2.6. Configuraciones típicas del receptor solar.....	27
Figura 2.7. Esquema de receptor volumétrico.....	28
Figura 2.8. Ilustración del ensombrecimiento en heliostatos	28
Figura 2.9. Ilustración de bloqueo de la radiación	29
Figura 2.10. Ilustración de radiación atenuada por agentes atmosféricos	29
Figura 2.11. Pérdidas de energía.....	30
Figura 2.12. Elementos de un heliostato	31
Figura 2.13. Facetas y estructura soporte de un heliostato de la PSA	31
Figura 2.14. Representación gráfica de los vectores para el alineamiento del heliostato	32
Figura 2.15. Esquema de método de calibración mediante detección de diferencia de claridad.....	33
Figura 2.16. Distintos métodos de calibración de heliostatos.....	34
Figura 2.17. Esquema cronológico de la evolución de los heliostatos.....	35
Figura 2.18. Captador cilindro - parabólico a analizar y resultado de la detección de la desviación..	36
Figura 2.19. Ejemplo de un patrón regular de líneas horizontales reflejado en un heliostato	37
Figura 2.20. Estudios de sistemas de seguidores solares a lo largo de los años	38
Figura 2.21. Esquema de una red neuronal.....	40
Figura 2.22. Estructura de la red neuronal	41
Figura 2.23. Aprendizaje supervisado por regresión y clasificación.....	43
Figura 2.25. Comparativa aprendizaje tradicional y aprendizaje por transferencia.....	44
Figura 2.26. Esquema de una red neuronal convolucional.....	45
Figura 2.27. Operación de convolución y resultado	46
Figura 2.28. Resultado de aplicar distintos detectores de rasgos en una imagen	46
Figura 2.29. Imagen original.....	47
Figura 2.30. Imagen tras convolución e imagen tras función ReLu.....	47
Figura 2.31. Imagen analizada por una red neuronal convolucional en la PSA	48
Figura 2.32. Cuadros delimitadores para figuras geométricas	50

Figura 2.33. Comparativa entre detección de objetos y segmentación semántica	51
Figura 2.34. Estructura del modelo SSD	52
Figura 2.35. Ejemplo de cuadros delimitadores	53
Figura 2.36. Representación de la intersección sobre la unión (IoU)	53
Figura 2.37. Resultados de una serie de convoluciones.....	54
Figura 2.38. Ejemplo herramienta LabelBox para etiquetado de una imagen.....	56
Figura 2.39. Interfaz de bienvenida del supercomputador perteneciente a CETA-Ciemat	57
Figura 3.1. Esquema de varios ratios de aprendizaje	59
Figura 3.2. Distribución normal a la izquierda, y distribución truncada en la derecha	60
Figura 3.3. Gráfica de pérdida de set de evaluación.....	63
Figura 3.4. Gráfica de pérdida de set de entrenamiento.....	63
Figura 3.5. Ejemplo de <i>overfitting</i>	64
Figura 3.6. Ejemplo de un buen entrenamiento	64
Figura 3.7. Ejemplo de intersección sobre la unión.....	65
Figura 3.8. Curva precisión – exhaustividad	66
Figura 3.9. Ejemplo de gráfica de precisión media en Tensorboard.....	66
Figura 4.2. Distribución de las fronteras para métrica mAP	107
Figura 4.3. Detecciones del modelo 13 <i>Inception</i> sobre imagen n° 1	108
Figura 4.4. Etiquetas de la imagen n° 1	108
Figura 4.5. Detecciones del modelo 13 <i>Inception</i> sobre imagen n° 2	109
Figura 4.6. Etiquetas de la imagen n° 2.....	109
Figura 4.7. Detecciones del modelo 13 <i>Inception</i> sobre imagen n° 3.....	110
Figura 4.8. Etiquetas de la imagen n° 3.....	110
Figura 4.9. Detecciones del modelo 11 <i>MobileNet</i> sobre imagen n° 1	111
Figura 4.10. Etiquetas de la imagen n° 1.....	112
Figura 4.11. Detecciones del modelo 11 <i>MobileNet</i> sobre imagen n° 2.....	112
Figura 4.12. Etiquetas de la imagen n° 2.....	113
Figura 4.13. Detecciones del modelo 11 <i>MobileNet</i> sobre imagen n° 3.....	114
Figura 4.14. Etiquetas de la imagen n° 3.....	114

ÍNDICE DE TABLAS

Tabla 2.1. Comparativa de las distintas tecnologías CSP	24
Tabla 3.1. Distintos valores de predicciones para cálculo de AP	65
Tabla 4.1. Distintos modelos para utilizar en aprendizaje por transferencia	68
Tabla 4.2. Modelos SSD con arquitectura <i>MobileNet</i>	69
Tabla 4.3. Modelos SSD con diferentes arquitecturas	70
Tabla 4.4. Modelos seleccionados para entrenamiento	70
Tabla 4.5. Configuración modelo 1 <i>Inception</i>	71
Tabla 4.6. Configuración modelo 2 <i>Inception</i>	72
Tabla 4.7. Configuración modelo 3 <i>Inception</i>	74
Tabla 4.8. Configuración modelo 4 <i>Inception</i>	75
Tabla 4.9. Valores de las métricas de los primeros cuatro modelos <i>Inception</i>	77
Tabla 4.10. Configuración modelo 5 <i>Inception</i>	77
Tabla 4.11. Configuración modelo 6 <i>Inception</i>	79
Tabla 4.12. Modelos con imágenes	80
Tabla 4.13. Configuración modelos 11 y 12 <i>Inception</i>	81
Tabla 4.14. Configuración modelo 14 <i>Inception</i>	83
Tabla 4.15. Configuración modelo 14 <i>Inception</i>	84
Tabla 4.16. Configuración modelo 15 y 16 <i>Inception</i>	85
Tabla 4.17. Comparación modelos prometedores <i>Inception</i>	86
Tabla 4.18. Entrenamientos <i>Inception</i> evaluados	88
Tabla 4.19. Configuración modelo 1 <i>MobileNet</i>	89
Tabla 4.20. Configuración modelo 2 <i>MobileNet</i>	90
Tabla 4.21. Configuración modelo 3 <i>MobileNet</i>	92
Tabla 4.23. Comparación resultados de los primeros cuatro modelos <i>MobileNet</i>	94
Tabla 4.24. Configuración modelo 5 <i>MobileNet</i>	95
Tabla 4.25. Configuraciones y resultados de modelos	97
Tabla 4.26. Configuración modelos 9, 10 y 11 <i>MobileNet</i>	98
Tabla 4.27. Configuración modelo 12 <i>MobileNet</i>	99
Tabla 4.28. Configuración modelo 13 <i>MobileNet</i>	101
Tabla 4.29. Configuración modelos 14 y 15 <i>MobileNet</i>	102
Tabla 4.30. Resultados modelos 12, 13, 14 y 15 <i>MobileNet</i>	104

Tabla 4.32. Modelos prometedores <i>MobileNet</i>	104
Tabla 4.33. Entrenamientos <i>MobileNet</i> evaluados	106
Tabla 4.34. Modelos óptimos de cada arquitectura.....	115

RESUMEN

El objetivo fundamental de este Trabajo Fin de Grado (TFG) consiste en la aplicación de técnicas de aprendizaje automático y visión artificial en sistemas de seguimiento solar aplicados a heliostatos. En primer lugar, se realiza un estudio teórico previo de las áreas del trabajo; seguido de la preparación de los datos y el software necesario. Se seleccionaron los modelos más adecuados para esta aplicación, siendo la velocidad y la precisión los criterios fundamentales, los cuales se usaron para la técnica de *transfer learning* como base para el aprendizaje de nuestros modelos. Posteriormente, se llevó a cabo una sintonización manual de los hiperparámetros, con el fin de encontrar los modelos óptimos para esta aplicación, discutiendo y analizando finalmente los resultados obtenidos.

El trabajo nace como respuesta a la demanda de mejora del sistema de seguimiento de los captadores solares, en especial los de tipo heliostato. Se busca la optimización de un algoritmo de control que sea eficaz y preciso. Para ello, se propone un sistema basado en redes neuronales artificiales aplicados a la visión artificial, el cual será capaz de reconocer y detectar los objetos que se encuentran a su alrededor, mediante el aprendizaje a través de un conjunto de datos formado por imágenes tomadas en la planta solar de heliostatos CESA de la Plataforma Solar de Almería (PSA). De esta forma, el heliostato será capaz de predecir su posición óptima, mejorando la eficiencia del sistema.

Finalmente, se obtuvo un modelo basado en la arquitectura *Inception*, el cual presentó un desempeño realmente bueno en cuanto a velocidad y precisión, por lo que sería adecuado para la implementación de este en el sistema de seguimiento solar.

Palabras clave: calibración de heliostatos, aprendizaje automático, visión artificial, optimización, *transfer learning*.

ABSTRACT

The main aim of this thesis consists in the application of machine learning and computer vision techniques in solar tracking systems applied to heliostats. For this purpose, every area of interest has been theoretically studied; followed by the data and software preparation. Then, models that seemed more suitable for this application were chosen, based on two fundamental parameters of this real time detection technology: velocity and accuracy. *Transfer learning* is used in order to have a model that already is trained. Afterwards, fine-tuning of some models has been done to optimize them as much as possible, and check which one of them presents a better overall efficiency in the solar system tracking.

This thesis arises from the demand of the improvement in solar systems trackers, especially in heliostats. It is looked for the most efficient control algorithm optimization. To this purpose, it is proposed an artificial neural network system applied to computer vision, which will be able to recognise and detect objects around heliostats, through learning by a dataset based on images of the solar plant CESA in Plataforma Solar de Almeria (PSA). Thus, heliostats will be able to predict its optimum position.

Finally, a model based on *Inception* architecture was achieved. It showed a really good performance in terms of velocity and accuracy, therefore, being suitable for implementing it in a solar tracking system.

Key words: heliostat calibration, machine learning, computer vision, optimization, transfer learning.

1. INTRODUCCIÓN

1.1 Motivación del trabajo

La demanda de la energía crece a nivel mundial debido al rápido incremento de población y la considerable evolución en industria. Los recursos de energía renovables, los cuales dependen de recursos naturales para generar un suministro infinito de energía que es sostenible y no contaminante, son una prometedora alternativa a los recursos de energía convencionales y han ganado una importancia considerable en los años recientes [1].

Combustibles fósiles tales como el petróleo crudo, la antracita y el gas natural, usados en procesos de combustión, causan un gran daño al medio ambiente, principalmente debido a las emisiones de gases de efecto invernadero; emisiones que conducen al calentamiento global del planeta [2].

Según la Organización de las Naciones Unidas, si seguimos con la tendencia actual del modelo energético, se incrementará la temperatura más de 1.5°C entre 2030 y 2050; lo que provocará cambios en el planeta como temperaturas extremadamente altas, alteración del ciclo de agua, aumento de la frecuencia de eventos climatológicos extremos, aumento del nivel del mar que tendrá especial repercusión en los ecosistemas y recursos hídricos de zonas costeras, consecuencias negativas directas sobre la ganadería, agricultura y economía, aumento de la pobreza y desigualdad [3].

Además, los recursos de energía predominantes hoy día pronto estarán agotados; por todo esto nace la necesidad de alternativas sostenibles. Existen diferentes tecnologías capaces de aprovechar fuentes renovables de energía, aunque la mayor parte de estas fuentes son una forma de energía solar transformada por procesos naturales, se pueden clasificar en eólica, solar, geotérmica, oceánica, hidráulica, biomasa y residuos.

Son numerosas las ventajas ambientales, estratégicas y socioeconómicas del uso de las energías renovables frente a las energías fósiles. En la actualidad, en la mayor parte del mundo las tecnologías renovables son la fuente más económica de producción de energía eléctrica en función del coste nivelado de la energía (LCOE). El 77% de la capacidad de los proyectos eólicos en tierra y el 83% de la capacidad de los proyectos basados en tecnología fotovoltaica ofertada en subastas públicas para 2020, pujan por debajo de la opción más barata de generación con combustibles fósiles [4].

Debido a esto, la inversión en energías renovables en países avanzados está aumentando considerablemente. En Europa, por ejemplo, durante el periodo 2007-2017, se aumentó de 258 a 512 GW la capacidad de generación de origen renovable; al igual que aumento exponencialmente el número de publicaciones científicas relacionadas [4].

España por su parte publicó el Plan Nacional Integrado de Energía y Clima de 2012-2030 (PNIEC), donde fija un marco estratégico para la transición ecológica de la economía española que pretende reducir a corto plazo un 21% las emisiones de gases de efecto invernadero y aumentar hasta un 74% sobre el total de consumo el aporte de energía eléctrica. Con este plan, se prevé reducir más de un 90% las emisiones de gases de efecto invernadero y alcanzar el 100% renovable del mix energético para 2050 [5].

Esta transición ecológica previsiblemente tendrá un impacto positivo sobre el empleo y la economía, puesto que las energías renovables y la industria sostenible ayudan a un mejor reparto de riqueza.

Según el PNIIEC, España podrá ahorrar más de 75.000 millones de euros en importaciones de combustibles fósiles respecto del escenario actual. Además, en España el PNIIEC estima que se generaran entre 250.000 y 364.000 empleos netos hasta 2030 gracias a esta transición a un modelo energético renovable [4].

Uno de los recursos naturales disponibles más abundantes es la energía solar. Es uno de los recursos más limpios y sostenibles con el medio ambiente, comparado con otras fuentes de energía renovables. Entre las diferentes tecnologías que aprovechan el sol como fuente de energía, se encuentra un tipo de tecnología llamada energía termosolar de concentración, que refleja y concentra la radiación solar sobre un receptor que está refrigerado por un fluido, capaz de transportar la energía térmica captada [4].

Esta se trata de una tecnología de nueva generación, ya que se encuentra todavía en una fase temprana de desarrollo, pero está experimentando un fuerte crecimiento [5].

La tecnología termosolar presenta un alto potencial de reducción de coste y una solución a la gestión eficiente de la producción gracias a su capacidad de almacenamiento de energía térmica. El pronóstico de la Agencia Internacional de la Energía (IEA) para 2023 sobre energías renovables predice un crecimiento a nivel mundial de la CSP de un 87% respecto de la capacidad en 2018 [3].

Entre las posibles tecnologías para la optimización de plantas termosolares está la inteligencia artificial (IA). Este término, en su sentido más amplio, indica la habilidad de una máquina de realizar el mismo tipo de funciones que caracterizan al pensamiento humano. El término abarca la tecnología que se aplica a una máquina desarrollada mediante algoritmos que le proporcionan la capacidad de interpretación, decisión y resolución de forma autónoma ante las señales que recibe.

Dentro de la IA hay varias ramas, siendo el aprendizaje automático una de ellas. Es una aplicación que dota al sistema de la habilidad de aprender por sí mismo, y mejorar a través de la experiencia sin ser programado específicamente para ello. Se centra en el desarrollo de algoritmos que pueden manejar gran cantidad de datos y usarlos para encontrar patrones de una manera mucho más rápida y eficiente de lo que un humano sería capaz [6].

1.2 Objetivos

El trabajo que se propone a continuación, tiene como objetivo la aplicación de técnicas de aprendizaje automático para optimizar el funcionamiento de los sistemas de seguimiento solar aplicados a producción energética, que son pieza fundamental en sistemas CSP y PV. Concretamente se pretende optimizar el uso de modelos basados en redes neuronales para la detección de objetos en imágenes, aplicados a sistemas de seguimiento solar para sistemas de torre central.

Para ello se realiza un estudio teórico y una comprensión de todas las bases de éstas áreas, para posteriormente ponerlas en práctica. Será fundamental la etapa de entrenamiento de la red neuronal, que se basará en un conjunto de datos de entrada formado por una serie de imágenes etiquetadas, a partir de las cuales el algoritmo aprenderá a identificar los distintos objetos fundamentales que se encontrará durante su recorrido diario, como son el sol, los heliostatos vecinos, la torre receptora y las nubes. A continuación se realizará una optimización manual de todos los modelos seleccionados modificando los hiperparámetros que los configuran, fijando como criterios la precisión y la velocidad. Se empleará una técnica de entrenamiento llamada *transfer learning*. Finalmente, se detallarán los resultados y expondrán las conclusiones del estudio.

Se persigue de este modo aumentar reducir costes y aumentar la eficiencia global de la planta.

1.3 Contexto

Este trabajo se encuentra en el marco del proyecto llevado a cabo por la colaboración que la Plataforma Solar de Almería (PSA) y la UAL mantienen a través de su centro mixto CIESOL, encuadradas en el proyecto de investigación SOLTERMIN, “Soluciones termosolares para integración en procesos industriales”, financiado por el Ministerio de Ciencia e Innovación; cuyo objetivo es avanzar en el desarrollo de soluciones compactas y optimizadas de las tecnologías energéticas termosolares de concentración, adecuadas para el suministro de calor en procesos industriales [12].

Este trabajo es presentado con el propósito de la finalización del grado de ingeniería electrónica industrial.

1.4 Resultados

Los resultados obtenidos se clasificarán en función de tres parámetros: pérdidas, velocidad y precisión; siendo estos dos últimos fundamentales, ya que la idea sería implementar estos algoritmos en un sistema de seguimiento solar a tiempo real. Estas son unas métricas que nos indicarán la bondad del modelo; siendo las pérdidas un valor que indica como de acertadas son las predicciones realizadas; la velocidad indica la rapidez con la que se realizan las predicciones; mientras que la precisión servirá para mostrar la fiabilidad de esas predicciones.

Se entrenará un método de detección de objetos, SSD, mediante aprendizaje por transferencia, utilizando modelos preentrenados con el conjunto de datos COCO, los cuales se basan en distintas arquitecturas de redes neuronales. Tras hacer una selección de los modelos más adecuados, escogieron dos modelos basados en las arquitecturas *Inception* y *MobileNet*.

Tras realizar una optimización manual de los parámetros de configuración de modelos seleccionados, para lo cual es necesario volver a entrenar los modelos cada vez que se modifican sus parámetros, se llega a la conclusión de que una configuración de la arquitectura *Inception* es la más óptima entre todas las probadas. Esta se caracteriza por tener un tamaño de imagen de entrada de 400 x 400; utilizar la técnica *dropout*; y tener un ratio de transferencia de 0.01, el cual decrece exponencialmente cada 3.000 pasos. Este modelo es validado mediante la comparación de sus detecciones junto con las etiquetas reales de la imagen, y se puede afirmar que tiene un comportamiento bastante aceptable por los usuarios finales.

1.5 Estructura del trabajo

La planificación de este trabajo se ha llevado a cabo en 6 etapas. A continuación se muestra el orden y el tiempo empleado en cada una de ellas:

1. Revisión bibliográfica: estudio previo sobre cada una de las áreas que abarca este trabajo. Esto se llevó a cabo durante los meses de mayo, parte de junio y parte de julio
2. Familiarizarse con las librerías de visión artificial e inteligencia artificial, que se llevó a cabo durante junio y julio.
3. Análisis de las demandas y limitaciones de los diferentes modelos basados en redes neuronales para detección de objetos aplicados a sistemas de seguimiento solar. Se basó en realización de un estudio para detectar y limitar el número de modelos a emplear; lo cual se hizo durante los meses de septiembre y parte de octubre.

También durante este tiempo se realizó el etiquetado de una pequeña parte del conjunto de datos para aprender esta parte del proyecto; ya que la idea principal era tomar imágenes en la Plataforma Solar de Almería, pero con el problema del COVID-19, no se pudieron adquirir nuevas imágenes hasta el mes de diciembre de 2020.

4. Entrenamiento y optimización de los parámetros que definen los diferentes modelos y los de entrenamiento en base a diferentes criterios como velocidad y precisión; que abarcó finales de octubre y noviembre.
5. Validación de los modelos elegidos, que se hizo a finales de noviembre y principios de diciembre.
6. Redacción de la memoria que se hizo a la par que la etapa anterior, a finales de noviembre y principios de diciembre.

Por tanto, se estima que la duración del proyecto ha sido de 180 días que, trabajando una media de 3 horas diarias, resulta en un total de 540 horas.

1.6 Competencias utilizadas en el TFG

El Trabajo Fin de Grado (TFG) perteneciente al Grado en Ingeniería Electrónica Industrial es requisito indispensable para la obtención del título, y en él se espera apreciar un conjunto de competencias que se han ido adquiriendo a lo largo del grado universitario. A continuación, se presentan las competencias específicas del grado desarrolladas durante la realización de este proyecto:

CT1. Capacidad para la redacción, firma y desarrollo de proyectos en el ámbito de la ingeniería industrial que tengan por objeto la construcción, reforma, reparación, conservación, demolición, fabricación, instalación, montaje o explotación de: estructuras, equipos mecánicos, instalaciones energéticas, instalaciones eléctricas y electrónicas, instalaciones y plantas industriales y procesos de fabricación y automatización.

CT3. Conocimiento en materias básicas y tecnológicas, que les capacite para el aprendizaje de nuevos métodos y teorías, y les dote de versatilidad para adaptarse a nuevas situaciones.

CT4. Capacidad de resolver problemas con iniciativa, toma de decisiones, creatividad, razonamiento crítico y de comunicar y transmitir conocimientos, habilidades y destrezas en el campo de la Ingeniería Industrial.

CT10. Capacidad de trabajar en un entorno multilingüe y multidisciplinar.

CB3. Conocimientos básicos sobre el uso y programación de los ordenadores, sistemas operativos, bases de datos y programas informáticos con aplicación en ingeniería.

CTEE11. Capacidad para diseñar sistemas de control y automatización industrial.

CTEQ2. Capacidad para el análisis, diseño, simulación y optimización de procesos y productos.

2. REVISIÓN BIBLIOGRÁFICA

2.1 Energía solar de concentración (CSP)

En la actualidad estamos experimentando un creciente interés y desarrollo en el campo de las tecnologías solares; entre las cuales encontramos la energía termosolar de concentración, también conocida como CSP (*Concentrated Solar Power*). Esta tecnología de generación de energía eléctrica utiliza reflectores o lentes que concentran la radiación solar en una superficie pequeña para convertirla en energía térmica.

Las principales tecnologías CSP son:

- i) Colector cilindro-parabólico, también conocido como PTC, del inglés *Parabolic Through Collector*: una superficie reflectante parabólica refleja y concentra la radiación solar sobre un tubo receptor (foco lineal) que contiene un fluido caloportador, típicamente un aceite sintético. Tienen una configuración de seguidor solar de un eje. Es la tecnología CSP más desarrollada y extendida actualmente, constituyendo en 2017 un 80% de plantas operativas y en construcción. Tiene una temperatura de trabajo de hasta 400 °C aproximadamente [7].
- ii) Sistemas de torre o receptor central, también conocido como SPT, del inglés *Solar Power Tower*: un gran número de superficies reflectantes, llamados heliostatos, reflejan y concentran la radiación en un punto (receptor). Tiene una temperatura de trabajo que ronda los 600 °C, aunque existen de mayor temperatura. Los materiales del receptor son generalmente cerámicos o metales que son relativamente estables a temperaturas elevadas. El fluido de trabajo puede ser agua/vapor, sal fundida, sodio líquido o aire [8]. Los seguidores son de dos ejes, foco puntual. Esta es la tecnología en la que nos vamos a centrar en este trabajo. Se muestra una imagen de ella en la Figura 2.1.



Figura 2.1. Planta de tecnología de receptor central. CESA-PSA [Cortesía de PSA].

- iii) Reflector Fresnel, también conocido como LFR, del inglés *Linear Fresnel Reflector*: Se basa en las configuraciones previas. Los rayos solares son reflejados a un receptor fijo situado a lo largo de la línea focal de los reflectores. El receptor contiene tubos llenos de agua o aceite. A diferencia de PTC, el receptor no está localizado en los espejos, sino en un espacio separado. Tiene una temperatura de trabajo de 50 a 300°C.
- iv) Sistema de disco parabólico, también conocido como PDC, del inglés *Parabolic Dish Collector*: un conjunto de espejos forman una estructura parabólica que concentran la luz solar un punto. Las superficies reflectantes se colocan en un montaje con un sistema de seguimiento solar de dos ejes. En la zona focal se coloca un motor Stirling para mejorar la eficiencia de la conversión energética. Tiene una temperatura de trabajo de 120 a 1500°C, y el fluido de trabajo es helio. La eficiencia del sistema PDC es de las más altas de las tecnologías solares [8].

Se presenta en Figura 2.2 una imagen ilustrativa de estos tipos de tecnologías CSP:

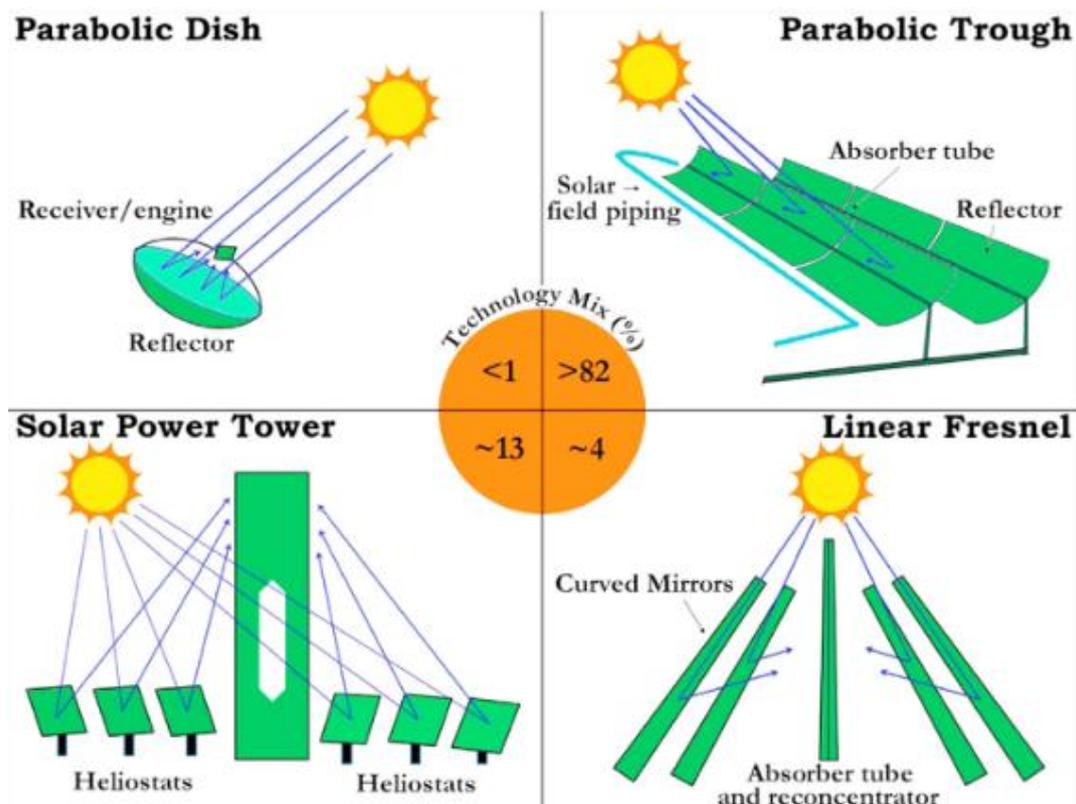


Figura 2.2. Principales tecnologías CSP [9].

A continuación, se muestra en Tabla 2.1 la comparativa de algunas características de estas tecnologías CSP, como son la eficiencia termodinámica, el coste, o el potencial de mejora.

	Coste relativo	Eficiencia termodinamica	Rango de operación de T (°C)	Ratio de concentración solar	Potencial de mejora
PTC	Bajo	Baja	20-400	15 - 45	Limitado
LFR	Muy bajo	Baja	50-300	10 - 40	Significante
SPT	Alto	Alta	300-565	150 - 1500	Muy significativa
PDC	Muy alto	Alta	120-1500	100 - 1000	Alto potencial

Tabla 2.1. Comparativa de las distintas tecnologías CSP [10].

De acuerdo con la Tabla 2.1, la tecnología PTC es la más desarrollada, tiene un potencial de mejora limitado; mientras que otras como SPT o PDC, tienen un significativo potencial de mejora, lo que las hace muy atractivos para proyectos de investigación [10].

En Tabla 2.1, también se encuentra el radio de concentración solar [11], que representa el ratio entre la densidad de la radiación solar en el receptor dividido entre la densidad de la radiación solar sin concentración (aproximadamente 1kW/m²). Como se puede observar, la tecnología de disco parabólico es la que mayor radio de concentración tiene, ya que la luz reflejada se concentra en un único punto; mientras que las tecnologías con menor ratio son la tecnología Fresnel y el captador cilindro - parabólico, ya que estas reflejan la luz a un tubo que tiene una gran extensión. Se observa que el ratio es directamente proporcional a la temperatura de trabajo, siendo el disco parabólico el que alcanza una mayor temperatura, que puede alcanzar hasta los 1500°C. Para una planta de torre central, el coste de inversión estimado, es decir, los costes que se incurren en la adquisición de los activos necesarios para poner el proyecto en funcionamiento, es de 4000 – 6000 €/ kW [20].

Se observa que la tecnología PTC tiene un menor riesgo técnico y financiero para los inversores por su longeva experiencia comercial; sin embargo, la tecnología SPT, a largo plazo, reduce estos riesgos de manera más significativa [5].

Las plantas CSP han evolucionado considerablemente desde su inicio en 1878 y están ganando un interés creciente debido a su bajo coste de operación y su independencia respecto al combustible. Esta tecnología presenta el mayor potencial de reducción de coste y la solución a la gestión eficiente de la producción gracias a su capacidad de almacenamiento de energía térmica. En el periodo 2010 – 2013, la tecnología CSP batió record, ya que la capacidad global aumentó desde 700 MWe (MegaWatio Eléctrico) a finales de 2009, hasta 2205 MWe a finales del 2013, lo que representa un crecimiento del 215%. [12] Además, el pronóstico de la Agencia Internacional de Energía (IEA) para 2023 sobre energías renovables predice un crecimiento a nivel mundial de la CSP de un 87% respecto de la capacidad en 2018 [13].

Actualmente, España y EEUU son los líderes mundiales en generación de electricidad CSP. De hecho, la capacidad instalada total en ambos países abarca el 73% de la capacidad global. Aunque si tenemos en cuenta los proyectos de energía termosolar de concentración que se encuentran bajo desarrollo, China debe ser considerada uno de los mayores inversores en esta tecnología, ya que cuenta con una visión política muy comprometida con el acuerdo de la COP21, donde se estableció el límite del cambio climático a menos de 2°C [14].

Se muestra a continuación, Figura 2.3, los proyectos CSP alrededor del mundo, en azul las plantas que están operativas; en verde las que están en desarrollo y en rojo las que están bajo construcción [26].

El receptor absorbe la radiación y suministra energía térmica mediante un fluido a una temperatura típicamente de 300-565 °C. Los SPT pueden incorporar un sistema de almacenamiento de energía térmica, típicamente sal fundida, que se almacena en tanques para permitir la generación de energía eléctrica bajo demanda. Se observa en la Figura 2.4 una planta solar de esta tecnología.

Una planta de energía fotovoltaica actualmente proporciona energía eléctrica a un coste más bajo que una planta de energía termosolar de concentración; sin embargo, el almacenamiento de la energía eléctrica es en general más caro que el almacenamiento de energía térmica. De esta manera, las plantas fotovoltaicas son más adecuadas para suministrar energía durante las horas de sol, mientras que las plantas CSP, durante la noche y en condiciones meteorológicas poco adecuadas [9].

Una combinación de ambas está vista como una solución comprometedora para el futuro del suministro energético debido a la rápida caída de las tecnologías solares, que provoca que sean capaces de competir directamente con los suministros de energías fósiles [9].



Figura 2.4. Ejemplo de la tecnología CSP de torre central [18].

Como se comenta anteriormente, en CSP, el receptor es un intercambiador de calor en donde la radiación solar es absorbida y transformada en energía térmica útil para los sistemas de conversión. Uno de los componentes más críticos de las centrales de torre es el receptor, situado en lo alto de la misma. Debido a las altas temperaturas y gradientes que se pueden alcanzar, ligado al rendimiento del ciclo termodinámico; los estudios actuales se centran en la elección de materiales y la disposición de estos, de forma que existan las menores pérdidas de calor posibles, incrementando así no sólo la eficiencia global del ciclo sino la del receptor en sí [19].

Hay diferentes criterios para clasificación de tipos de receptores, dependiendo de la configuración geométrica o de los materiales absorbentes usados para transferir la energía al fluido de transferencia. Para ejemplificar, se puede encontrar receptores de cavidad (Figura 2.5), en los que haces tubulares se encuentran en el interior de un recinto, entrando los rayos solares por una de sus paredes. Esto provoca que las pérdidas por fenómenos de conducción-convección se minimicen, pero aumentan las pérdidas por desbordamiento [20].

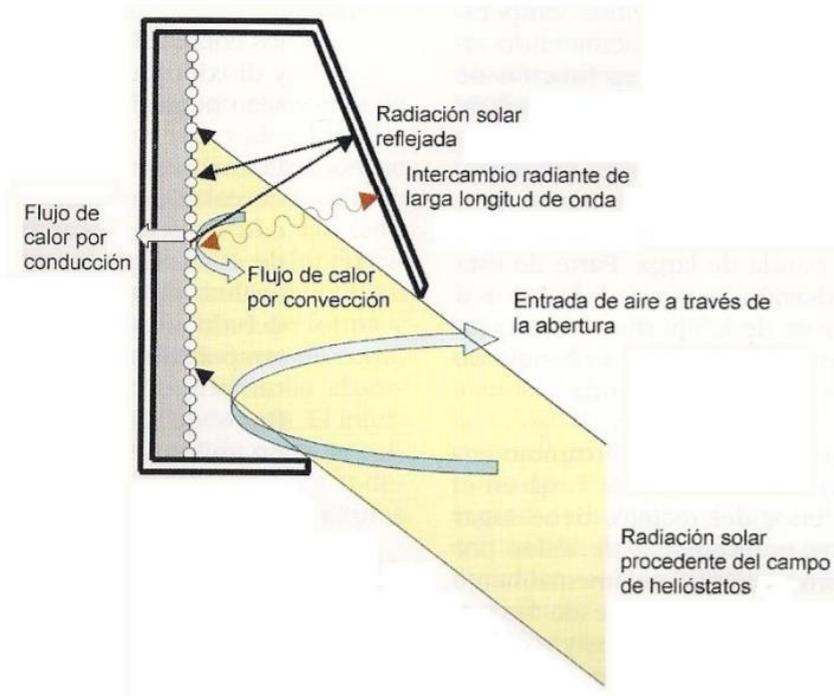


Figura 2.5. Esquema de receptor de cavidad [20].

Se encuentran también otras configuraciones geométricas típicas de receptor solar como son el cilíndrico externo y el plano externo (Figura 2.6). Aquí la radiación solar reflejada por el campo de radiación solar incide directamente sobre la superficie absorbente, mientras que en los receptores de cavidad, la radiación pasa a través de una apertura a una zona hueca en forma de caja, antes de llegar a las superficies absorbentes [16].

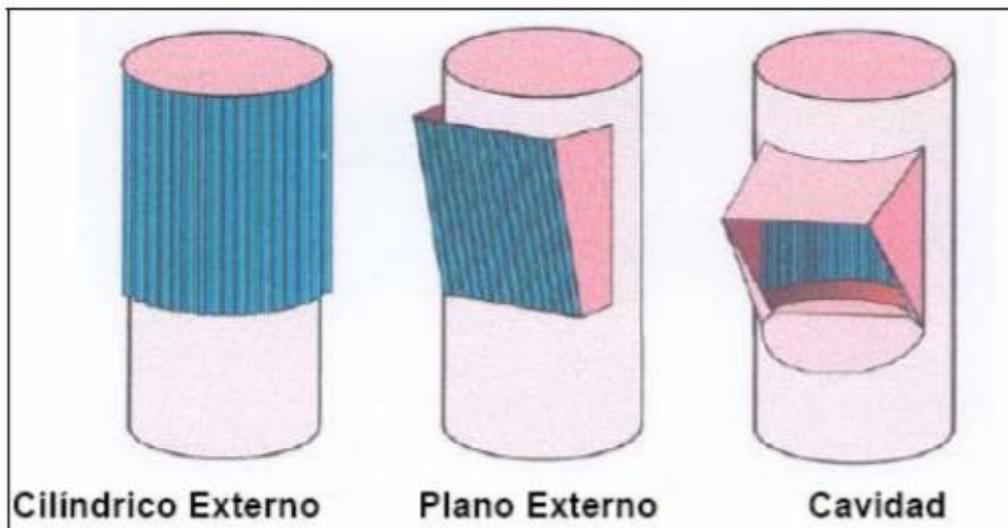


Figura 2.6. Configuraciones típicas del receptor solar [20].

También se encuentra otra configuración muy usada en diferentes proyectos, es el receptor volumétrico, esquema en la Figura 2.7, usando como fluido de trabajo el aire. Los receptores volumétricos están formados por una estructura metálica o cerámica de diversas formas con una matriz volumétrica sobre la que incide la radiación reflejada, calentando de esta forma el aire que pasa por su interior [21].

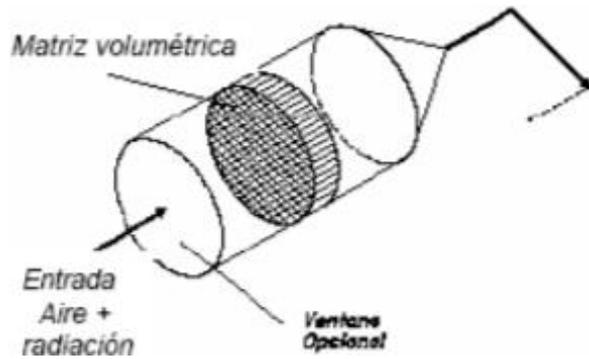


Figura 2.7. Esquema de receptor volumétrico [21].

Por otro lado, los ciclos termodinámicos más usados son el ciclo de Brayton, fluido de trabajo aire-gas, y el ciclo de Rankine, fluido de trabajo aire-vapor; aunque cada vez más se apuesta por un híbrido para mejorar los resultados. También se encuentra el ciclo combinado, el cual en una planta se utiliza la energía solar como energía auxiliar que permitirá incrementar el rendimiento del ciclo, y disminuir las emisiones [22].

Las plantas CSP también se emplean en cogeneración que consiste en producción simultánea de dos tipos energías que, en este caso, son energía eléctrica y energía térmica. El objetivo de la cogeneración es que no se pierda una gran cantidad de energía térmica en el condensador, que en muchos casos es mayor en cantidad que la energía eléctrica aprovechable en el ciclo. Esto hace que la planta tenga un potencial de rendimiento mayor que una central convencional [23].

El diseño de una planta de torre central, es decir, la especificación del número y la posición de los heliostatos; el tamaño de los espejo y de la torre central; la posición del receptor en la parte superior de la torre... es un problema con un número indefinido de grados de libertad. La mitad de la inversión total y el 40% de las pérdidas de energía se atribuyen al campo de heliostatos. Por tanto, es esencial optimizar su diseño para reducir el coste y para mejorar la eficiencia global de la planta solar [6].

A continuación se mencionan, y se muestran, algunos de los mecanismos básicos de pérdida de posible energía que ocurren en un STC debido a la acción de heliostatos y un esquema de estas pérdidas:

- Ensombrecimiento por heliostatos vecinos.

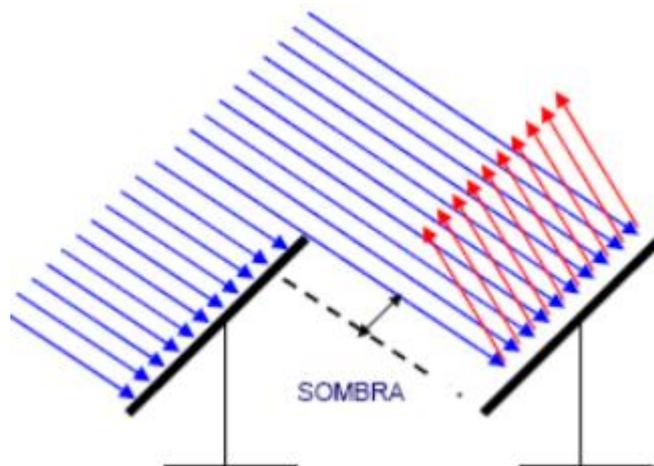


Figura 2.8. Ilustración del ensombrecimiento en heliostatos [16].

- Bloqueo de la radiación reflejada por heliostatos vecinos

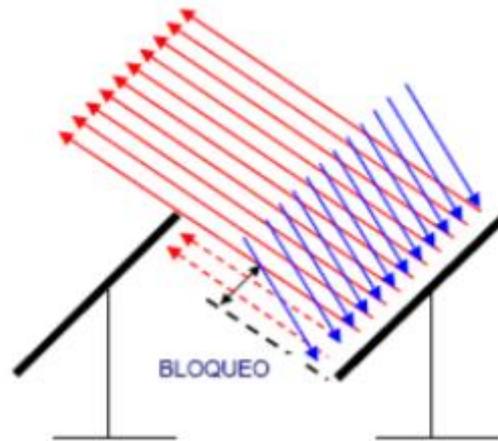


Figura 2.9. Ilustración de bloqueo de la radiación [16].

- Atenuación de la radiación reflejada debido a los procesos de absorción y dispersión de la atmosfera

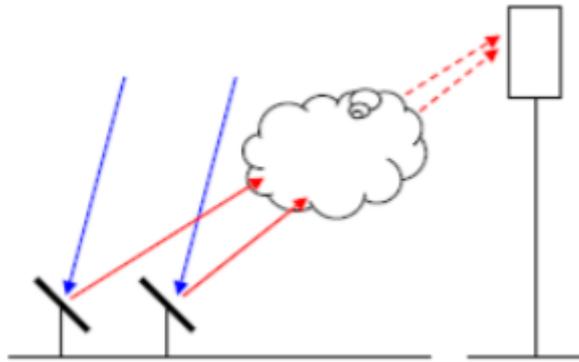


Figura 2.10. Ilustración de radiación atenuada por agentes atmosféricos [16].

- Radiación solar reflejada que alcanza el plano del receptor pero no apunta a la apertura debido a imperfecciones de alineamiento.
- Pérdidas geométricas como el factor coseno, que son aquellas causadas por la inclinación del eje óptico del heliostato con respecto a la trayectoria de los rayos solares. Estas son las mayores pérdidas que se producen en el campo solar, siendo su valor medio del orden del 20% de la potencia reflejada por el campo de heliostatos [16].

Para minimizar estas pérdidas, existen distintos métodos computacionales que permiten simular los campos de heliostatos. [9] Un software de uso extendido es SAM (*System Advisor Model*), es un modelo de software gratuito de carácter tecnológico-económico que ayuda a la toma de decisiones en el sector de las energías renovable; o NSPOC, un programa de optimización de plantas solares tipo torre. Optimiza plantas con receptores de cavidad, o cilíndricos, dando como resultado el diseño del campo, altura de torre, dimensiones del receptor... [24]. También destaca Tonatiuh que es un programa de fuentes abiertas para la simulación óptico-energética de sistemas solares de concentración [15].

A continuación, en la Figura 2.11 se muestran los principales mecanismos de pérdidas y el porcentaje que representan.

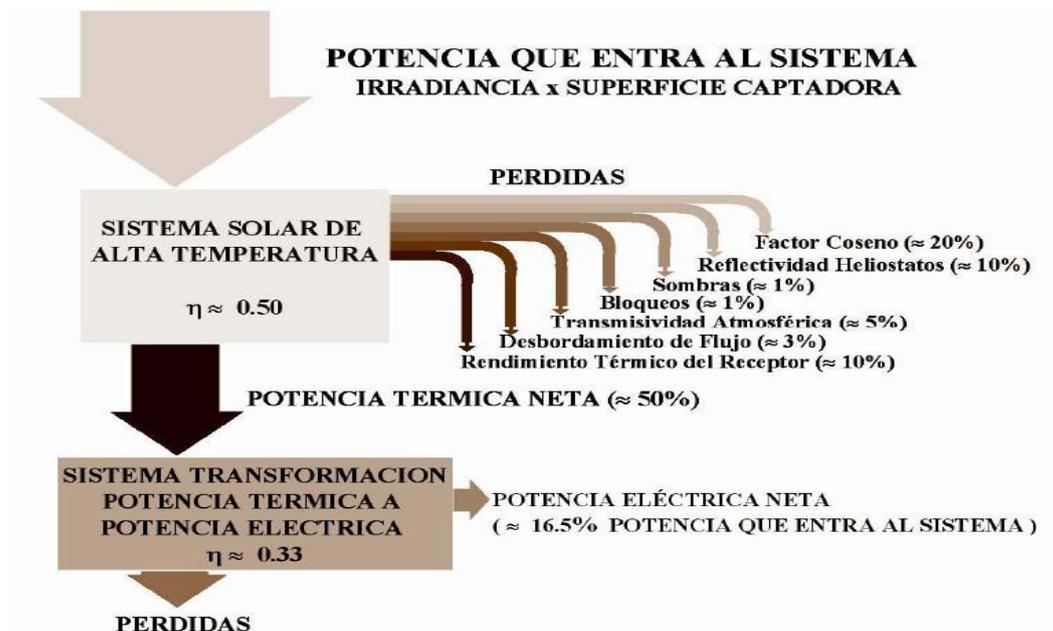


Figura 2.11. Pérdidas de energía [16].

2.3 Heliostatos

Como se comentó anteriormente, la función de un heliostato es reflejar y concentrar la radiación solar en un punto fijo sobre el receptor de la torre central. Debido al movimiento constante de la tierra respecto del sol, la posición aparente de este cambia constantemente y los heliostatos deben ajustar su configuración continuamente para seguir el movimiento solar, ajuste que se rige por la exactitud y es del orden de miliradianes [8].

2.3.1 Partes de un heliostato

Un heliostato está formado, como se ve en la Figura 2.12, por una superficie reflectante (espejos), una estructura soporte (formada por cerchas metálicas normalmente), un pedestal de cimentación, unos mecanismos de movimiento (servomecanismos) y un sistema de control.

El sistema de control alinea, actuando sobre el servomecanismo de elevación y azimut, la superficie reflectante de forma que la reflexión de la radiación solar directa que incide sobre ella sea dirigida al receptor solar [16]. La ley de la reflexión especular, Ecuación 2.2 y Ecuación 2.3, que permite esta forma de actuar, se puede describir en la siguiente forma:

$$1) \quad \vec{i} \cdot \vec{n} = \vec{r} \cdot \vec{n} \quad (2.2)$$

$$2) \quad \vec{i}, \vec{r} \text{ y } \vec{n} \text{ pertenecen al mismo plano: } (\vec{i} \times \vec{r}) \cdot \vec{n} = 0 \quad (2.3)$$

Donde,

\vec{i} : Dirección del rayo incidente.

\vec{r} : Dirección del rayo reflejado.

\vec{n} : Dirección de la normal a la superficie reflectante.

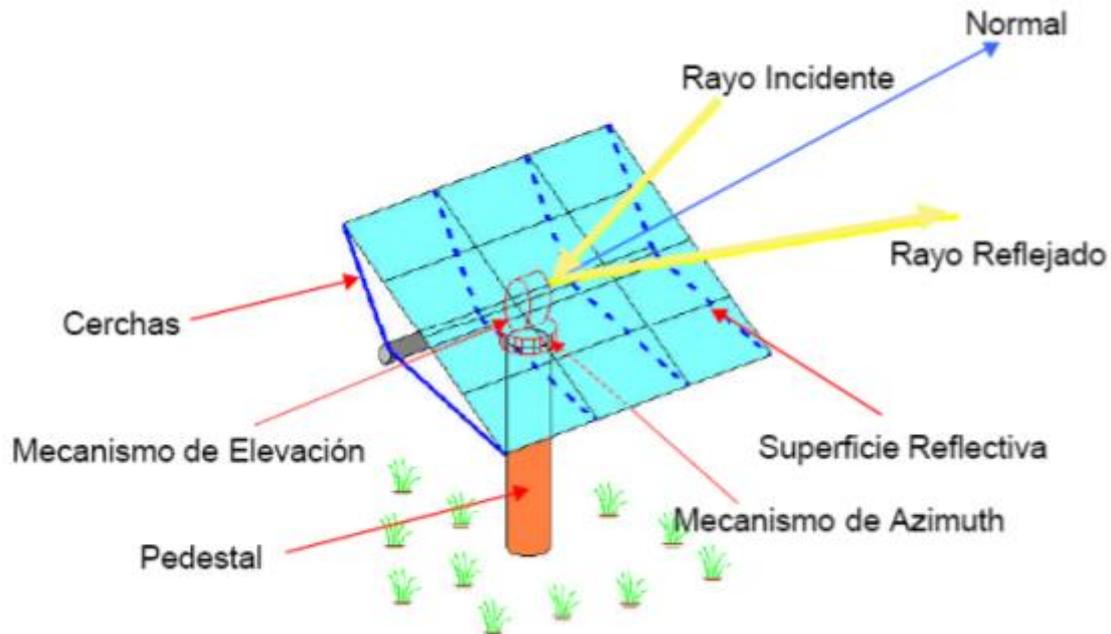


Figura 2.12. Elementos de un heliostato [16].

Cada heliostato convencional está formado por múltiples módulos de espejos, llamados facetas, Figura 2.13. Cada faceta tiene, normalmente, una ligera curvatura cóncava y también se inclinan respecto al plano de la estructura de soporte.

El correcto alineamiento del sistema óptico del heliostato implica dos acciones: enfoque o canteo y seguimiento. El canteo del heliostato consiste en curvar ligeramente la superficie reflectante que forma el sistema óptico de manera cóncava (parabólica o esférica). El seguimiento solar del heliostato consiste en alinear el sistema óptico completo para dirigir la radiación solar a un único punto en el receptor. Un buen enfoque y seguimiento del heliostato implica un mayor factor de concentración y una reducción de las pérdidas; como resultado, la potencia anual interceptada por el receptor se maximiza [25].



Figura 2.13. Facetas y estructura soporte de un heliostato de la PSA [Cortesía de la PSA].

El error de seguimiento se define como la desviación estándar del error entre la orientación real y la ideal del eje óptico del heliostato a lo largo del año.

Esta desviación puede ser causada por errores sistemáticos como la orientación incorrecta de los ejes de rotación o errores en los cálculos de la posición solar. Otros problemas de alineamiento pueden ser debidos a la exposición de los heliostatos a las condiciones atmosféricas que prevalecen en el campo. Las fuerzas aerodinámicas causadas por el viento deben ser tomadas en cuenta a la hora del diseño de los heliostatos, de modo que si es una zona con frecuentes ráfagas de viento de elevada velocidad, menor será el tamaño del heliostato. También hay que tener en cuenta los efectos dinámicos que causa el viento en el heliostato, como la inercia, elasticidad y pequeñas vibraciones. Por otro lado, se debe tener en cuenta el efecto de la gravedad cuando el heliostato está operando, ya que se ha demostrado que puede influir en el seguimiento y el enfoque [28].

2.3.2 Seguimiento

El procedimiento general de seguimiento solar de un heliostato es el siguiente: el primer paso es determinar el vector heliostato-sol (V_S), es decir, el vector desde el punto central de la superficie reflectante del heliostato (O') hasta el punto central del sol. Luego se determina el vector heliostato-receptor (V_T), es decir, el vector desde el punto central de la superficie reflectante del heliostato (O') hasta el punto central del receptor. Finalmente, el vector normal del heliostato (V_A) es dirigido a la bisectriz del ángulo entre el vector heliostato-sol (V_S) y el vector heliostato-receptor (V_T). Se puede ver una representación gráfica en la Figura 2.14. [51].

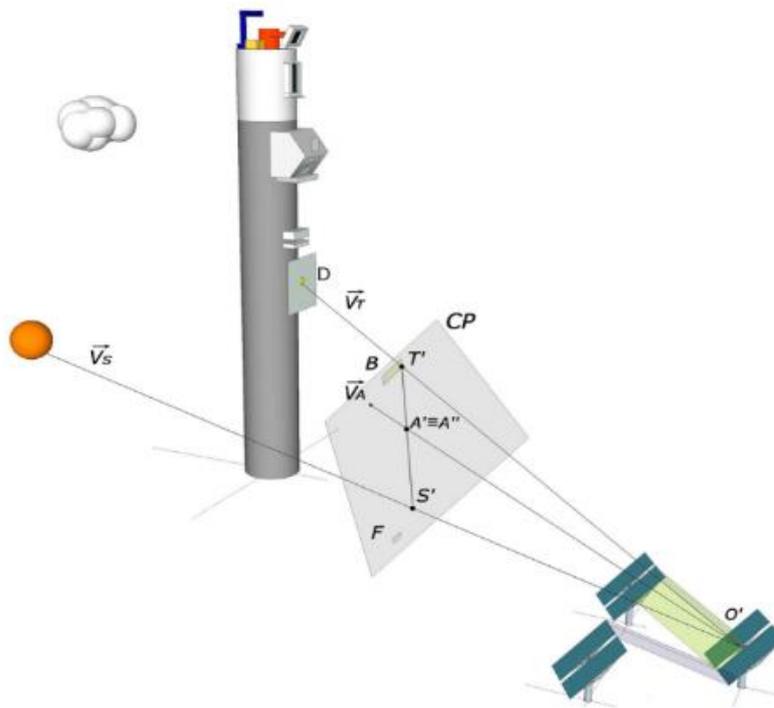


Figura 2.14. Representación gráfica de los vectores para el alineamiento del heliostato [14].

En este trabajo, para el correcto seguimiento se hace uso de una cámara colocada en el punto central del heliostato, O' . A partir de las fotos tomadas con esta cámara, se puede detectar S' , A' y T' , los cuales representan la intersección entre el plano de la cámara y los vectores V_S , V_A y V_T , respectivamente. Para un posicionamiento adecuado del heliostato, el vector V_A debería cortar al punto al punto medio del segmento formado por S' y T' , el cual se llamara A'' . La diferencia entre los puntos A' y A'' se conoce como error de seguimiento y es tratado como la principal entrada del sistema de control.

2.3.3 Métodos de calibración y sistemas de seguimiento

A continuación se define y se muestra, Figura 2.16, una clasificación para sistemas de seguimiento solar o métodos de calibración de heliostatos basados en criterios de localización, tipo y número de dispositivos de medida o sensores. Son cuatro clases que dependen de la localización del principal dispositivo utilizado para la calibración[19]:

- Clase A: Se basa en una cámara central o sensor. Este dispositivo se encuentra en el heliostato y utiliza el sol o un haz de luz que, mediante el conocimiento del punto objetivo deseado y el punto al que apunta el heliostato, corrige esta desviación. También la cámara podría estar situada en la torre, y de esta manera no se obtiene una imagen del haz de luz reflejado por el heliostato, sino una foto del heliostato en sí. De esta manera, se podría detectar los bordes del heliostato y su orientación respecto a la posición de la cámara y así estudiar su precisión.
- Clase B: En esta clase se encuentran sistemas de calibración con un láser central y cámaras, en el que el láser es orientado automáticamente hacia la cara del heliostato. Cuando este se encuentra alineado correctamente, un corto pulso de láser es emitido, y este haz es reflejado por el espejo hacia el cielo, en donde dos, o más, cámaras digitales son capaces de recrear una imagen 3D de modo que el haz se hace visible, y de esta información se puede obtener el vector normal del espejo, sabiendo así si tiene una orientación adecuada.
- Clase C: Se basa en la detección de posición del foco solar con cámaras en la torre, como por ejemplo, un método que requiere la instalación de cuatro cámaras alrededor del receptor para medir la orientación del heliostato, en términos de comparar las diferencias de claridad en la imagen con un software de procesamiento para cada uno de los espejos vistos por las cuatro cámaras. Se observa una ilustración simplificada del proceso a continuación, Figura 2.15:

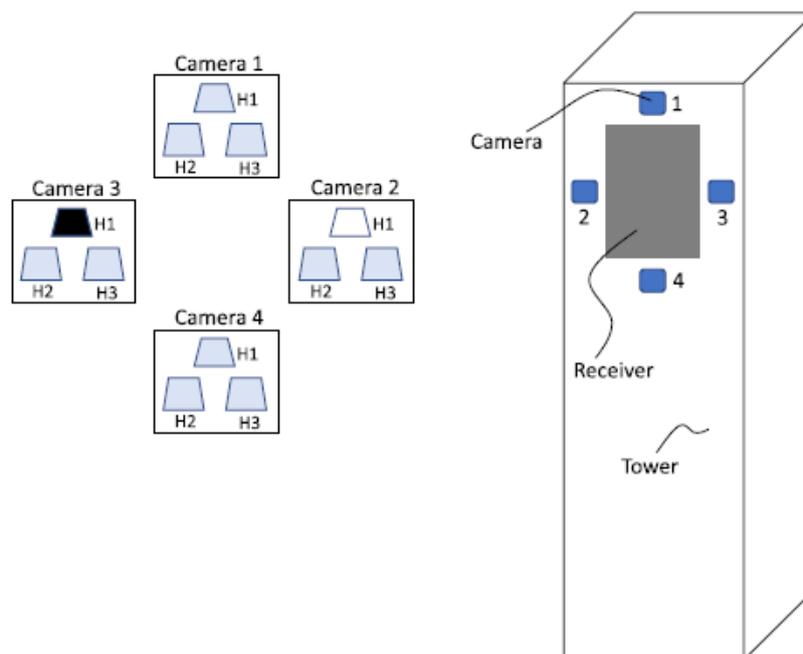


Figura 2.15. Esquema de método de calibración mediante detección de diferencia de claridad [19].

- Clase D: Se basa en la localización de cámaras o sensores en la parte móvil reflectante de cada heliostato, y trata sobre el reconocimiento de objetos conocidos en el espacio. Aunque la cámara puede estar orientada de cualquier manera, destacan dos enfoques principales: la cámara dispuesta en la parte frontal, de modo que puede detectar la posición del sol, la posición del receptor, de la torre... o la cámara en la parte trasera del reflector, por el cual puede obtener una evaluación de la orientación basada en objetos conocidos en el campo de visión. Ambos enfoques requieren tanto una calibración inicial como periódica.

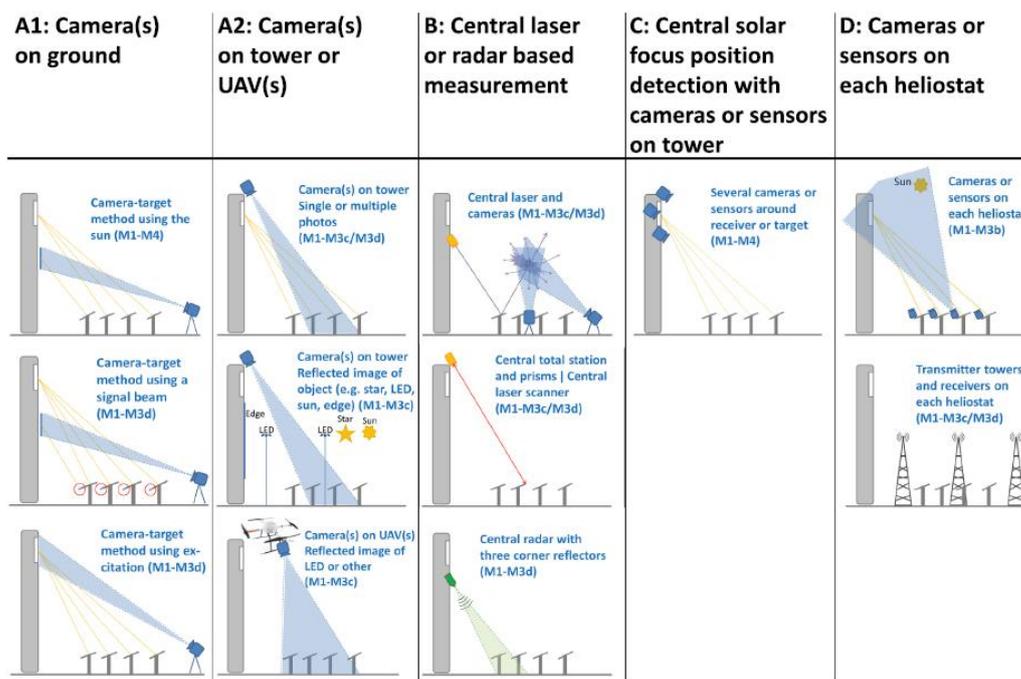


Figura 2.16. Distintos métodos de calibración de heliostatos [19].

2.3.4 Evolución

La evolución de los heliostatos se basa en la reducción de LCOE y de mejorar la eficiencia de estos.

Los heliostatos representan el 40-50% del coste de una planta de torre central, por lo tanto deben ser relativamente de bajo precio para que el coste de la energía de la planta sea competitiva con los combustibles fósiles [25]. Por ejemplo, para lograr un coste de energía de 10 USD/kWh, los heliostatos debían costar como máximo 120 USD/m²; o para un LCOE de 0.06USD/kWh, los heliostatos deberían costar alrededor de 75USD/m². Para alcanzar estos objetivos, se requieren diseños y soluciones innovadoras. Además, las dimensiones de los heliostatos deben ser seleccionadas para minimizar manufactura y costes de instalación. Esto requiere una estimación precisa de la carga de viento en heliostatos operativos y estacionados para permitir que diseños estructurales sean desarrollados con una buena actuación óptica, a la vez que evitando fallos estructurales [25]. Como se observa en la Figura 2.17, en el pasado la tendencia era aumentar el tamaño de los heliostatos para reducir costes; pero en la década del 2000 gracias al avance en la tecnología y en los materiales, esta tendencia se invirtió y los heliostatos disminuyeron su tamaño.

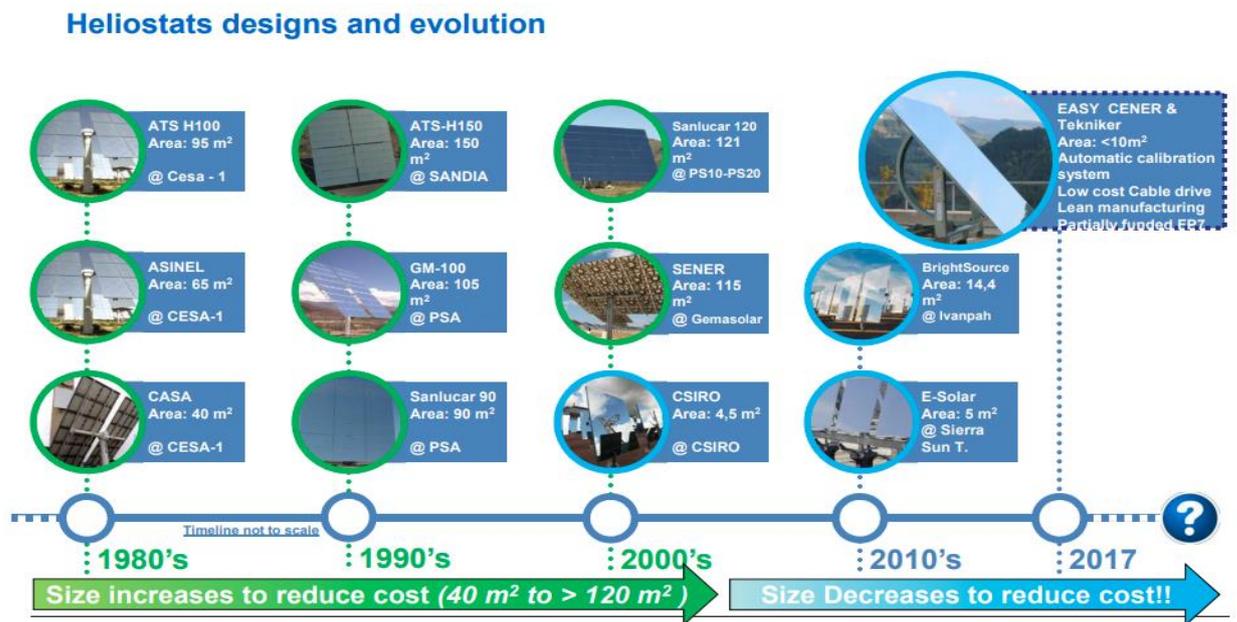


Figura 2.17. Esquema cronológico de la evolución de los heliostatos [16].

Una reciente aproximación interesante se basa en el concepto de ‘SmartHeliostats’ y ‘SmartCSP’, conceptos que se replican de las realidades que se ven en otras industrias como por ejemplo en automoción.

La idea de ‘SmartHeliostats’ tiene las siguientes proposiciones:

- Presencia de sensores integrados para que se vuelvan ‘conscientes’ y adquieran información relevante de su entorno; como proveyendo una orientación adecuada del reflector, identificando elementos vecinos dañados o indicando la suciedad de la superficie reflectora.
- Análisis integrado y capacidad de procesar los datos obtenidos a partir de los sensores, generando respuestas adecuadas; como ‘calibrar heliostato’ o ‘limpiar superficie reflectante’
- Incorporar capacidad de comunicación para que se convierta en una verdadera ‘SmartCSP’, de modo que sea una planta conectada globalmente y localmente, funcionando de manera cooperativa y coordinada [29].

2.3.5 Visión artificial en CSP

La visión artificial es una rama de la informática que trata de extraer y analizar la información de interés contenida en una imagen o secuencia de imágenes. Los campos de aplicación crecen cada día y van desde el reconocimiento de caras al diagnóstico precoz de enfermedades, pasando por la detección y localización de objetos y personas, la interacción gestual con sistemas, el guiado de robots o la conducción automática.

La visión por computador en plantas de energía solar, y concretamente en tecnología CSP, se ha utilizado tradicionalmente en métodos que permiten el conteo de las superficies reflectantes para corregir esos errores sistemáticos que producen desviaciones de la proyección de los rayos solares [34].

Uno de estos métodos es la fotogrametría, que consiste en una técnica de medida que usa la información de un conjunto de imágenes del mismo objeto desde distinta perspectiva para evaluar su estructura tridimensional. Los heliostatos pueden ser medidos con una desviación estándar menor a 0.2mm, proporcionando así la posibilidad de mostrar deformaciones y predecir su eficiencia. Se ve un ejemplo de esta técnica en la Figura 2.18 [35].

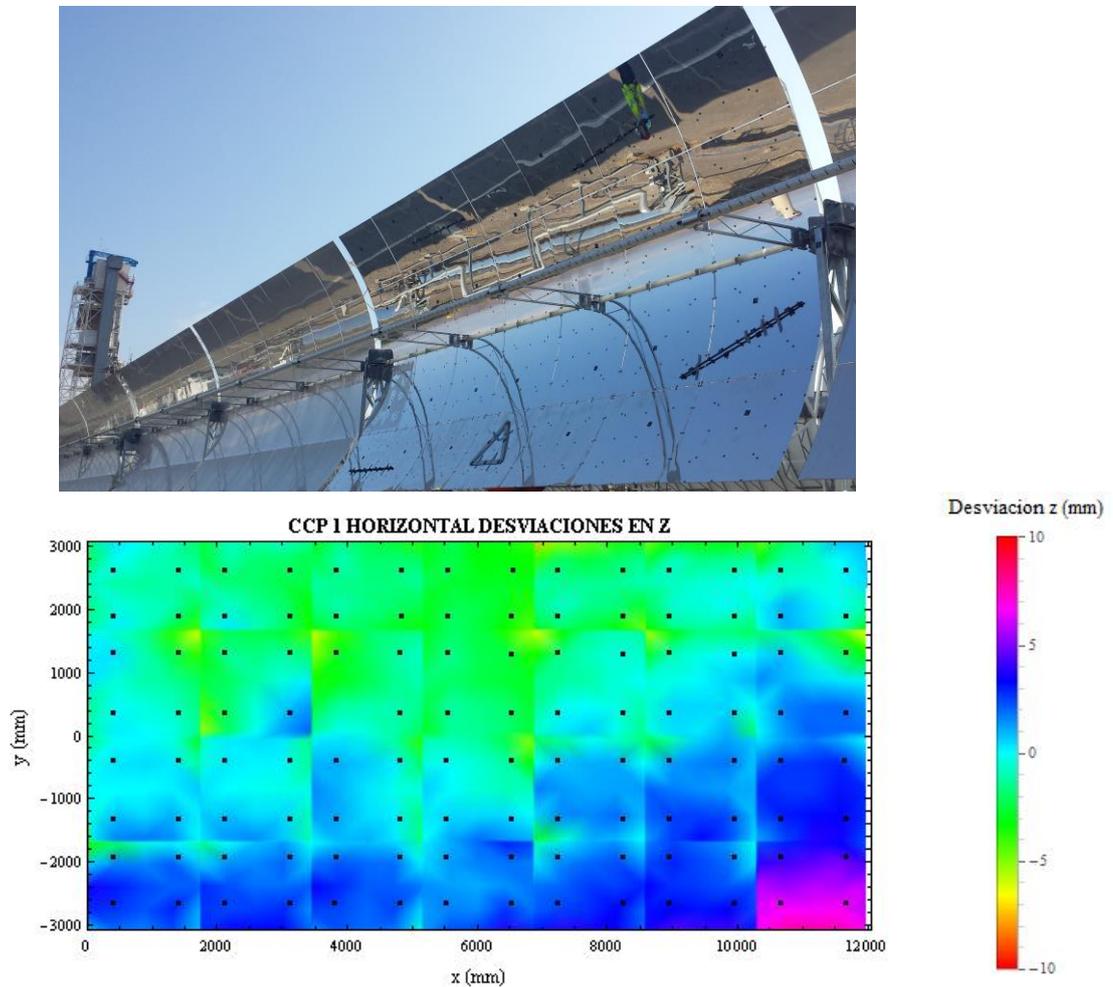


Figura 2.18. Captador cilindro - parabólico a analizar y resultado de la detección de la desviación [16].

Otra técnica de gran precisión y rapidez es la llamada deflectometría, que consiste en la detección de las más leves desviaciones con respecto a una superficie perfecta [36]. Este método utiliza patrones conocidos de líneas regulares cuyo reflejo en el espejo es observado por una cámara. Las deformaciones del patrón de rayas en su reflejo se utilizan para evaluar las imperfecciones locales del espejo, como en la Figura 2.19 [37].

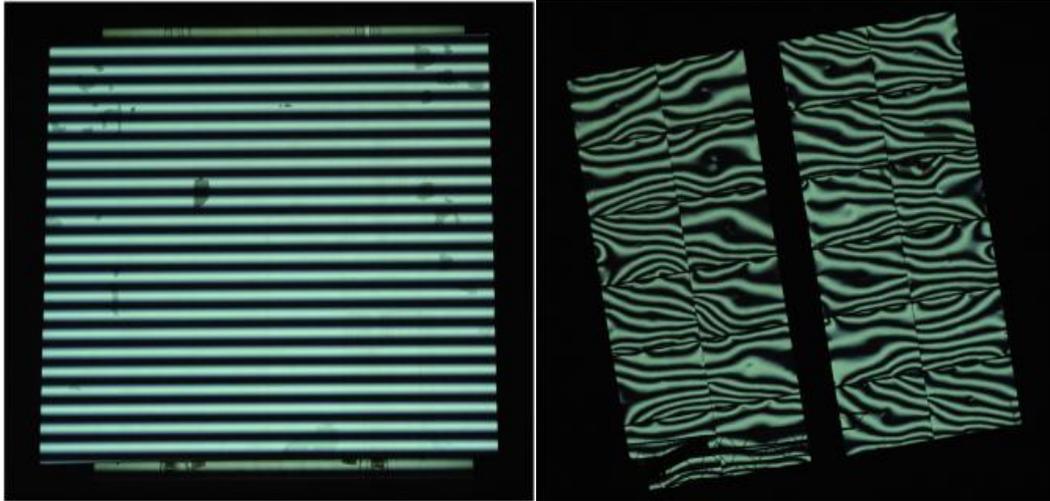


Figura 2.19. Ejemplo de un patrón regular de líneas horizontales reflejado en un heliostato [38].

Berenguel et al. [34] presentó un desarrollo de un sistema de control de posicionamiento simplificado y automático, de heliostatos mediante técnicas de visión artificial y dispositivos CCD [34]. Con este sistema de control basado en una cámara B/W en tiempo real, se corrigen las desviaciones de una manera automática realizando un trabajo similar al que hacen los operarios, que se encargarán de la supervisión del proceso en lugar de accionar el movimiento de los heliostatos. Las imágenes obtenidas se utilizan para estimar la distancia entre el centroide de los rayos de sol proyectados por los heliostatos y un blanco situado en la torre, esta distancia por lo tanto se utiliza para fines de baja precisión de corrección de desviaciones o *offsets*. Para la corrección automática se usan técnicas básicas de procesamiento de imágenes [34].

Para terminar, mencionar una novedosa técnica de calibración de los heliostatos que también se basa en visión artificial, hace uso de un dron el cual evalúa la orientación del heliostato. Las desviaciones entre la posición actual, obtenida por la cámara del dron, y la posición teórica, obtenida por un modelo óptico, permite detectar errores de alineamiento en el heliostato. Esto permite que los heliostatos puedan ser evaluados mientras están operando [28].

2.4 Sistemas de seguimiento solar.

El movimiento del sol a lo largo del cielo, lento pero continuo, necesita un sistema de control estable y resistente a oscilaciones. Para ello, la atención principal de la determinación de estos sistemas debe ser puesta en la configuración de los ejes y del sistema de control. Cada una de estas partes presenta unas ciertas opciones, con sus ventajas y desventajas, las cuales pueden ser manipuladas para incrementar la cantidad total de energía colectada por el sistema.

Se puede clasificar los seguidores solares según el número de ejes:

- De un único eje: poseen un único eje de rotación. Estos seguidores son comúnmente usados en todo el mundo debido a su simple estructura y fácil implementación de sus sistemas de control.
- De dos ejes: los sistemas de doble eje necesitan al menos un par de motores, uno para el azimut solar, y otro para el seguimiento de ángulos de elevación solar. Aunque este sistema es más caro y precisa de más trabajo para la implementación, posee un mejor resultado comparado con los de un solo eje.

La cantidad de energía solar recogida por un seguidor solar es directamente proporcional a la cantidad de radiación solar que recibe; en este caso, esto es logrado por este tipo de seguidor solar de forma más exitosa en comparación con otros tipos [30].

También se pueden clasificar los seguidores solares como activos o pasivos. Un sistema de seguimiento solar activo es aquel que determina el camino del sol en el cielo mediante sensores; mientras que un sistema pasivo es aquel que depende de la expansión térmica en materiales, normalmente un fluido, o un desajuste de presión entre dos puntos en los extremos del seguidor. El primer sistema pasivo fue desarrollado por una compañía americana en 1969; mientras que el primer sistema activo apareció en 1975 presentado por McFee. Se trataba de un algoritmo que calculaba la energía total en el receptor central de un sistema de energía solar y determinaba la distribución del flujo de densidad en él. La tolerancia de error de la posición del sol se situaba entre 0.5° y 1° .

Desde entonces, el interés por los seguidores solares ha ido en aumento, siendo especialmente notorio en los últimos cinco años como se puede comprobar en la Figura 2.20.

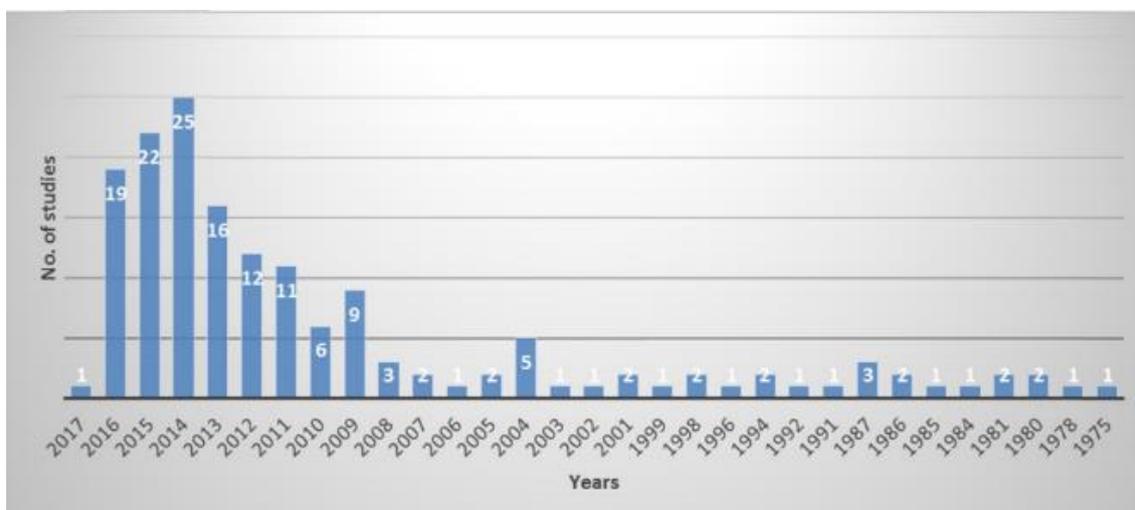


Figura 2.20. Estudios de sistemas de seguidores solares a lo largo de los años [31].

Según la estrategia de control, existen seguidores en bucle abierto y cerrado:

- Seguidores solares de bucle cerrado: Se basan principalmente en un control con realimentación, actuando en base a las señales de entradas recibidas por los sensores que detectan los parámetros de la posición del sol. Por ejemplo, recientemente se ha desarrollado un algoritmo basado en visión artificial para corregir el *offset* de un heliostato [32].
- Seguidores solares de bucle abierto: este usa un controlador que calcula la entrada desde los datos del estado actual, y el algoritmo del sistema decide si esta entrada cumple los requisitos o no. Este tipo de sistema es más simple y económico de implementar, pero requiere mucho trabajo preliminar para adaptar el algoritmo y el sistema óptico correctamente lo que se traduce en mayores costes de instalación y mantenimiento. Además, tampoco observa los datos de salida del proceso que controla por lo que no puede corregir ningún error si hubiera alguna disfuncionalidad en el sistema, provocando un menor rendimiento del sistema. Muchos algoritmos de este tipo poseen errores de seguimiento entre 0.0003° y 0.0027° [32].

Por otro lado, también podemos clasificar los sistemas de seguimiento solar según determinen la posición solar:

- Basado en fecha y hora: estos sistemas están caracterizados por un sistema de control con un procesador, algoritmos, sensores, localización geográfica, así como la fecha y hora. Emplean ecuaciones para determinar la posición relativa del sol y corregir su posición.
- Basados en microprocesadores y sensores electro-ópticos: estos sistemas se basan en principios de diferencia de luminosidad. El microprocesador recibe información de los sensores y genera una señal diferencial usada para dirigir los actuadores así reorientando el sistema en dirección del rayo solar más óptimo. Generalmente no se pueden emplear cuando hay reflexión de la radiación solar.
- Basados en una combinación de sensores, fecha y hora.

2.4.1 Requerimientos de seguidores basados en visión artificial.

En sistemas de seguimiento solar basados en visión por computador, como es el caso que ocupa este trabajo, diferentes ajustes y parámetros de la cámara deben ser tomados en cuenta ya que afectan en gran medida al resultado. El campo de visión, FOV , en mrad, es el parámetro geométrico requerido y está relacionado con la aplicación del sistema de seguimiento a diseñar, es decir, un sistema de torre central necesita un FOV más ancho que un seguidor fotovoltaico debido a que el ángulo entre V_s y V_T es mayor.

Otros parámetros geométricos relevantes son la distancia focal, f , medida en mm; y el tamaño del sensor, ambos relacionados con el campo de visión. La resolución de la cámara $H \cdot W$ (altura por anchura de la imagen en píxeles) junto con el tamaño del sensor determinan la altura y anchura del píxel (pW y pH), que suelen tener el mismo valor p .

La incertidumbre del ángulo del punto objetivo (U), Ecuación 2.4, para los sistemas de seguimiento solar basados en visión por computador pueden ser estimados como la proyección del tamaño del píxel en mrad sobre A' en el plano CP. Esta puede ser calculada a partir de f y p , como se ve en la siguiente ecuación. La resolución mínima de la cámara puede ser determinada, por tanto, a partir de una incertidumbre particular [33].

$$U = \arctg\left(\frac{p}{f}\right) \quad (2.4)$$

2.5 Inteligencia artificial en energía solar.

El uso de inteligencia artificial tiene diversas aplicaciones en el campo de la energía solar, como pueden ser la predicción de la radiación solar o la producción, modelado de una planta, el control de sistemas, entre otros [39].

2.5.1 Redes neuronales

El aprendizaje profundo o *deep learning*, es un subcampo del aprendizaje automático, en la cual se utilizan estructuras lógicas basadas en las neuronas artificiales, cuya función es recibir, procesar y transmitir información.

A estas arquitecturas de modelos computacionales basadas en neuronas se denominan redes neuronales artificiales y han mejorado significativamente el estado del arte en el reconocimiento de voz, la detección de objetos, descubrimiento de nuevos fármacos, entre otras muchas áreas [41].

La estructura básica de una red neuronal, Figura 2.21, consiste en neuronas artificiales que son agrupadas formando capas. La disposición básica de una red neuronal es una capa de entrada, una o más capas ocultas y una capa de salida [14].

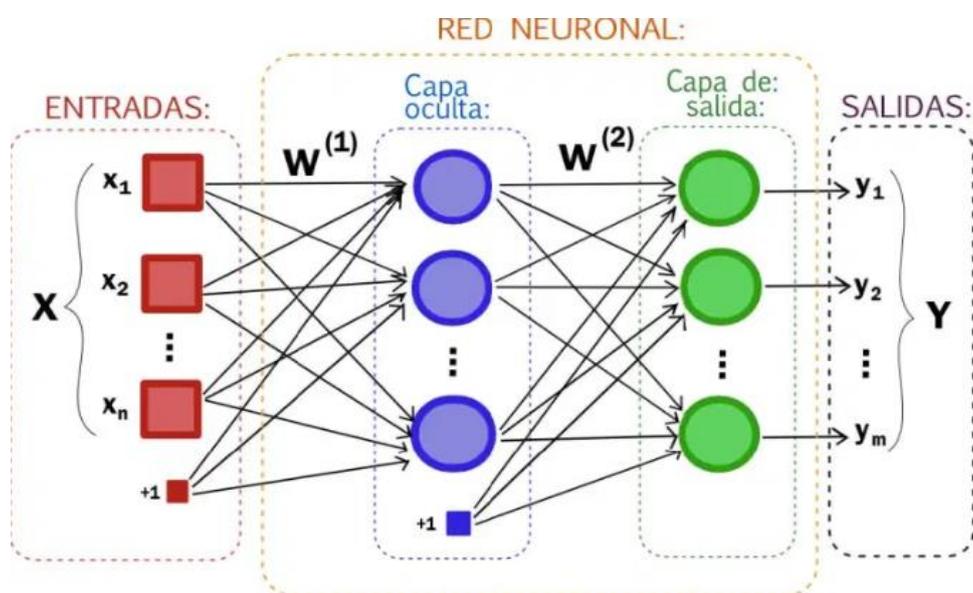


Figura 2.21. Esquema de una red neuronal [42].

Reciben el nombre de neuronas porque intentan simular el modo en el que el cerebro humano aprende mediante señales que se mandan entre las distintas neuronas. Cada conexión de estas neuronas artificiales tiene un peso asociado. La Figura 2.22 representa una estructura de red con entradas, x_m , conectadas a una neurona cuyas conexiones tienen distintos pesos, w_m . Todas estas señales recibidas son sumadas por la neurona, con cada señal multiplicada por su peso asociado en la conexión [43].

La suma ponderada de estos productos resultará en un único número que corresponde a una neurona de la capa oculta. A este número resultante se le aplica una función de activación, que permite el paso, o no, del valor resultante de la ponderación hacia la siguiente capa. Si se permite el paso, dicha neurona se activará y estará preparada para enviar información a la siguiente capa. De este modo, los valores de salida son las predicciones realizadas por la red neuronal [11].

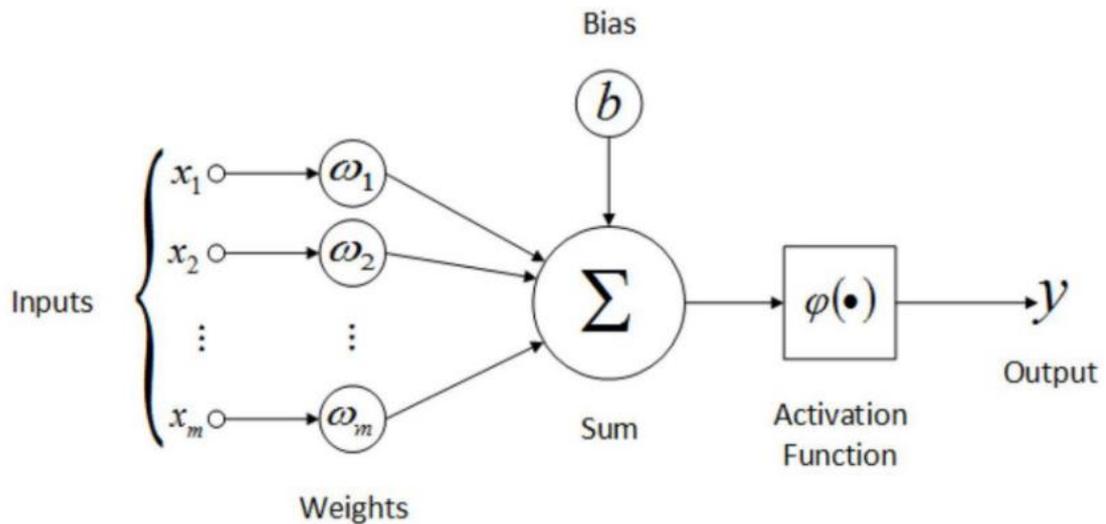
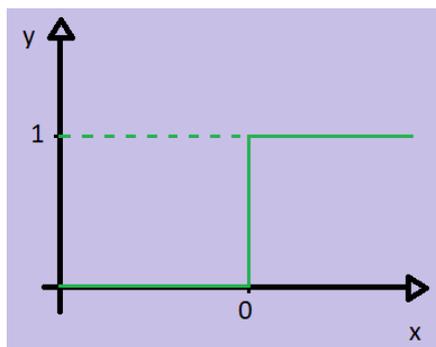


Figura 2.22. Estructura de la red neuronal [43].

Hay distintos tipos de funciones de activación. A continuación se presentan cuatro de las más conocidas:

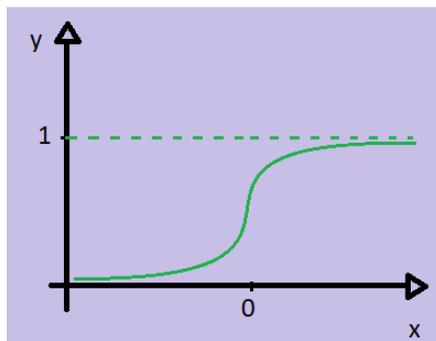
- Función escalón:



Si un número es negativo, la función lo convierte en cero; y en cambio, si es positivo, lo convierte en uno; como se ve en la siguiente figura.

$$\varphi(x) = \begin{cases} 1 & \geq 0 \\ 0 & < 0 \end{cases} \quad (2.5)$$

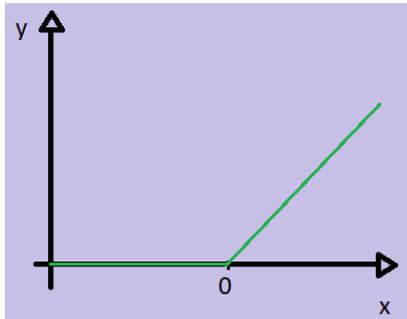
- Función sigmoideal o logística:



En esta función, cuando \$x\$ es un número negativo, tiende a 0; y cuando \$x\$ es positivo, tenderá a 1.

$$\varphi(x) = \frac{1}{1+e^{-x}} \quad (2.6)$$

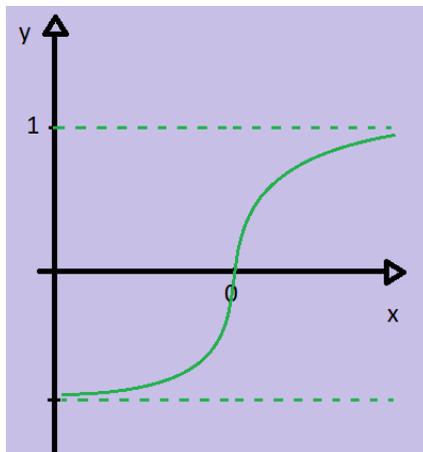
- Función Rectificador Lineal Unitario (ReLU):



En esta función, todo lo que sea negativo se convierte en cero; y lo positivo se considera como tal.

$$\phi(x) = \max(x, 0) \quad (2.7)$$

- Función Tangente-hiperbólica:



Esta función es parecida a la sigmoideal, siendo útil donde se necesita valores negativos.

$$\phi(x) = \frac{1-e^{-2x}}{1+e^{-2x}} \quad (2.8)$$

2.5.2 Entrenamiento o aprendizaje

Para obtener una red neuronal eficiente, la fase de entrenamiento es fundamental; ya que lo que se busca aquí es que las predicciones que realice sean lo más parecidas al valor real.

Durante el entrenamiento, la predicción de un valor realizada por la red, \hat{y}_i , se compara con el valor real, y , mediante funciones de coste, como pueden ser el error cuadrático medio (MSE), Ecuación 2.9; o el error porcentual absoluto medio (MAPE), Ecuación 2.10 [44].

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \quad (2.9)$$

$$MAPE = \frac{100}{N} \sum_{k=1}^N \left| \frac{\hat{y}_k - y_k}{y_k} \right| \quad (2.10)$$

El problema del aprendizaje de las redes neuronales consiste en la minimización de la función de coste asociada. Para ello, se utilizan métodos de optimización como el del gradiente descendente estocástico que tratan de encontrar el mínimo de la función de coste asociada, modificando los parámetros de la red neuronal, es decir, los valores y pesos asignados a las neuronas.

Una vez que se ha determinado un mínimo de la función, se realiza la propagación hacia atrás, donde se ajustan los nuevos parámetros de las neuronas para así volver a realizar una nueva predicción, y obtener un nuevo valor para la función de coste, idealmente, más pequeño [45].

Una vez entendido como funciona el entrenamiento, se pasa a explicar las distintas modalidades que existen dentro del entrenamiento en el aprendizaje profundo:

- **Aprendizaje supervisado:** el algoritmo es alimentado con unos datos de entrada etiquetados y unos resultados esperados. En este caso, se le enseña específicamente qué debe buscar y de esta manera el modelo es entrenado hasta que es capaz de detectar patrones característicos de ese conjunto de datos. Esto ocurre hasta que el modelo presente unos buenos resultados con datos que nunca antes había analizado. Este tipo de aprendizaje generalmente resulta en dos tipos de tareas, clasificación y regresión. El algoritmo de clasificación se centra en determinar la clase de los datos que se le presentan, como puede ser un algoritmo que realiza un análisis de sentimientos en redes sociales; mientras que el algoritmo de regresión proporcionará un valor numérico, como puede ser el precio que un cliente estaría dispuesto a pagar por un determinado producto. A continuación se muestra una representación gráfica, Figura 2.23, de estas dos modalidades [43]:

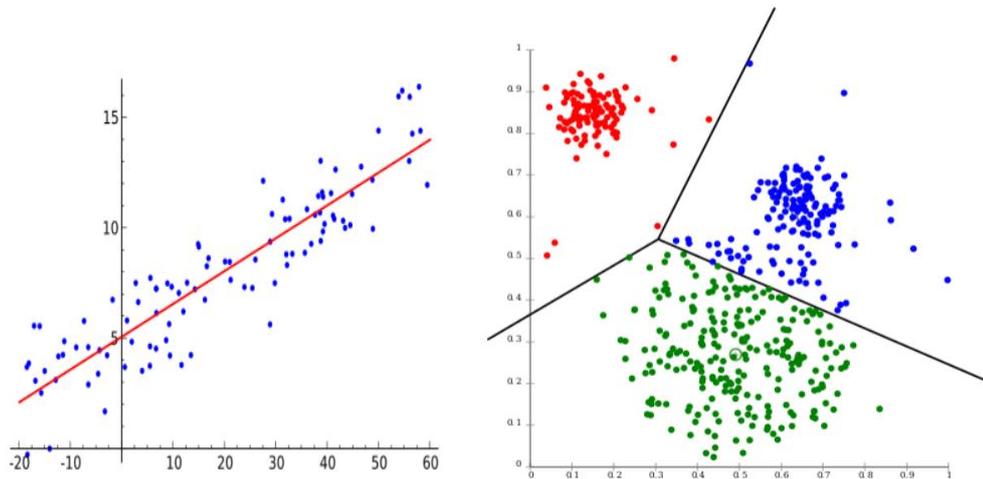


Figura 2.23. Aprendizaje supervisado por regresión y clasificación [59].

- **Aprendizaje no supervisado:** este tipo de aprendizaje se caracteriza porque solo se alimenta al modelo con un conjunto de datos de entrada, sin los valores correspondientes de salida. De este modo, el algoritmo funciona de forma independiente siendo capaz de mostrar y presentar relaciones interesantes entre los datos así descubriendo patrones que se escapan al ojo humano. Esta modalidad es popular en aplicaciones de *clustering*, el acto de descubrir grupos dentro del conjunto de datos; y de asociación, prediciendo patrones que se encuentran en el set de datos. Un ejemplo de esto es descubrir patrones dentro del ADN para analizar la biología evolutiva. En Figura 2.24 se muestra un esquema visual.

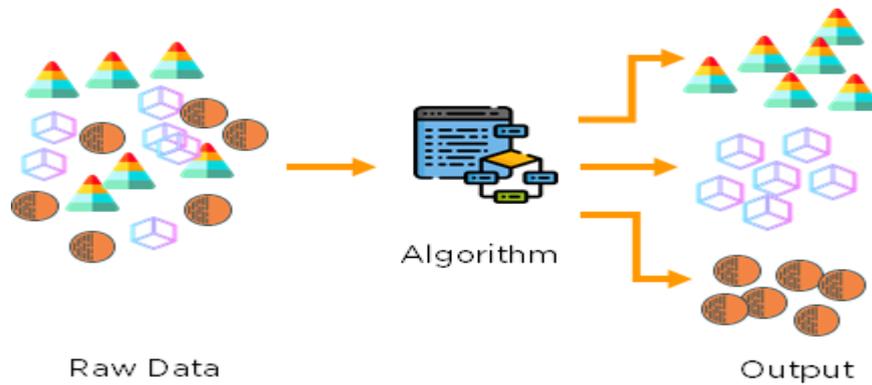


Figura 2.24. Aprendizaje no supervisado [59].

- Aprendizaje por refuerzo: es aquel que enseña al modelo a conseguir un objetivo mediante recompensas o faltas. De modo que el algoritmo tiene que aprender a conseguir un objetivo en un entorno totalmente incierto, en el cual va aprendiendo a través de estas recompensas o faltas; y el objetivo de esto es conseguir el número máximo de recompensas. El desarrollador designa la política de recompensas y el modelo es el que tiene que aprender mediante prueba y error como maximizar los buenos resultados, empezando por intentos totalmente aleatorios, hasta acabar con tácticas muy complejas. [59].

Aprendizaje por transferencia, o *transfer learning*. En este tipo de aprendizaje se aprovecha el conocimiento (características, pesos, etc) de modelos previamente entrenados para entrenar modelos nuevos, y debido a esto, incluso se solucionan problemas como la necesidad de tener un conjunto de datos de gran calibre. Por tanto, un modelo previamente entrenado y desarrollado para una tarea es modificado para realizar una tarea semejante [49].

En Figura 2.25 se observa la diferencia entre los procesos tradicionales de aprendizaje y la técnica de aprendizaje por transferencia. Como se observa, las técnicas tradicionales intentan aprender cada tarea desde cero, mientras que las técnicas de aprendizaje por transferencia intentan transferir el conocimiento adquirido en tareas anteriores a una tarea concreta cuando esta última tiene un conjunto de datos menor [50].

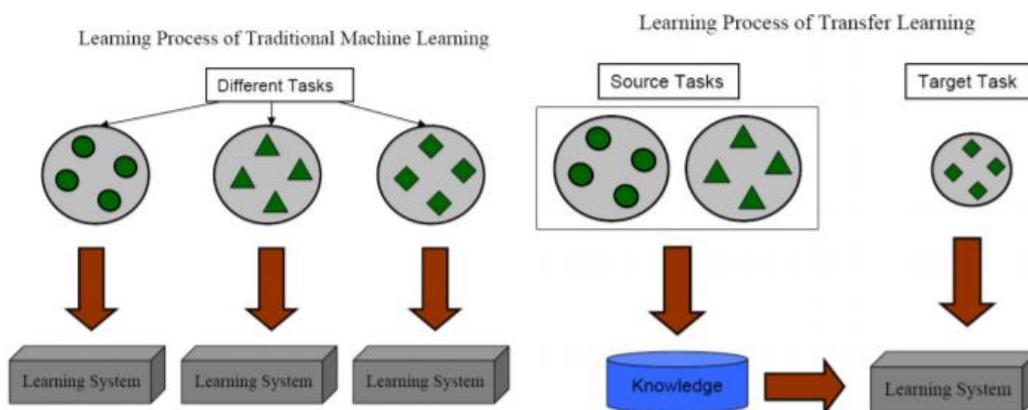


Figura 2.25. Comparativa aprendizaje tradicional y aprendizaje por transferencia [50].

Hay varias arquitecturas de redes neuronales de convolución muy usadas para el aprendizaje por transferencia debido a varios factores como son la precisión, la velocidad, la facilidad de entrenamiento o la habilidad de la red para ser capaz de adaptarse a distintas tareas, entre otros. Algunas de estas arquitecturas son *Inception*, *ResNet*, *VGGNet*, *AlexNet*; se explicarán algunas de ellas en secciones posteriores [51].

Para terminar, señalar que la mayor debilidad de las redes neuronales es la complejidad, o incluso, ausencia de explicación que está detrás de los modelos creados por ellas. Aunque las redes neuronales son fáciles de construir, el encontrar una buena estructura neuronal, como el preprocesado y procesado posterior de los datos, es una tarea que consume mucho tiempo [43].

2.5.3 Redes Neuronales Convolucionales

La red neuronal convolucional (CNN, del inglés *Convolutional Neural Network*) es un tipo de red neuronal artificial que procesa sus capas imitando al córtex visual del ojo humano para identificar distintas características en las entradas que en definitiva hacen que pueda identificar objetos y “ver”. Para ello, la CNN contiene varias capas ocultas especializadas y con una jerarquía: esto quiere decir que las primeras capas pueden detectar líneas, curvas y se van especializando hasta llegar a capas más profundas que reconocen formas complejas como un rostro o la silueta de un animal [11].

Las redes neuronales convolucionales han dominado completamente el área de visión por computador en los últimos años. Una CNN consiste en una capa de entrada, capa de salida, y múltiples capas ocultas (Figura 2.26). Las capas ocultas de una CNN consisten típicamente en capas convolucionales (*convolutional*), capas de agrupación (*pooling*), capas completamente conectadas (*fully connected*) y capas de normalización (ReLU).

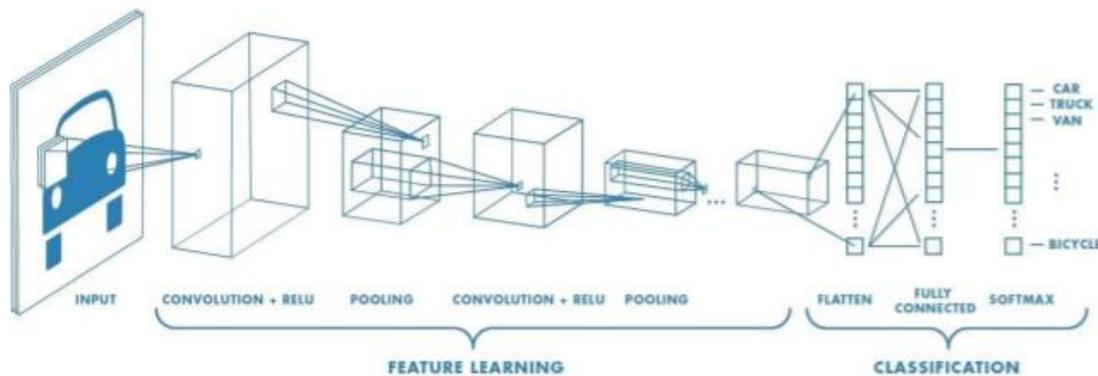


Figura 2.26. Esquema de una red neuronal convolucional [11].

Cada una de estas capas contiene un conjunto diferente de características para una imagen. Por ejemplo, si se presenta la imagen de una cara como entrada para una CNN, la red aprenderá algunas de sus características básicas como formas, zonas sombreadas, bordes... en sus capas iniciales.

El siguiente conjunto de capas consistirá en formas y objetos relativos a la imagen que son reconocibles como los ojos, nariz y boca, por ejemplo. Y de esta manera ira asociando la red esos rasgos a una cara [11].

La primera capa es la capa de convolución, ya que se aplica esta operación a la imagen de entrada junto con un filtro, generalmente una matriz 3x3, llamada detector de rasgos.

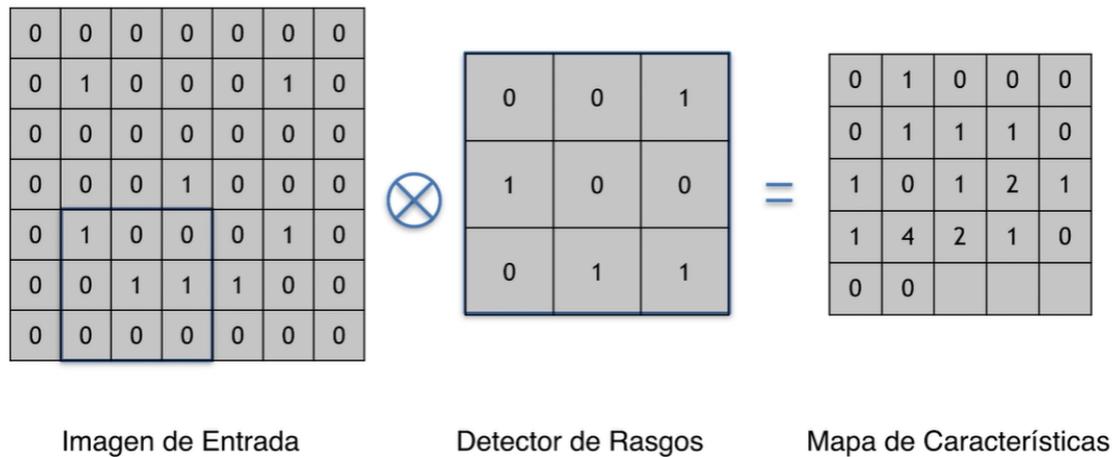


Figura 2.27. Operación de convolución y resultado [42].

Así, la red va aplicando esta operación píxel a píxel, y almacena los resultados en otra matriz llamada mapa de características. De este modo, se eligen muchos filtros que detecten diferentes rasgos, resultando en una gran cantidad de mapas de características, obteniendo versiones convolucionadas de la imagen original, que forman la capa de convolución. Se muestra en la Figura 2.27 un ejemplo de la obtención de un mapa de características, que como se ve, tiene en la parte inferior forma de media luna, lo que podría ser utilizado para detectar formas como el inicio de la comisura de la boca. Se ve también, en Figura 2.28, el resultado de aplicar distintos detectores de rasgos para una misma foto; en este caso se aplica para detectar bordes horizontales y bordes verticales:

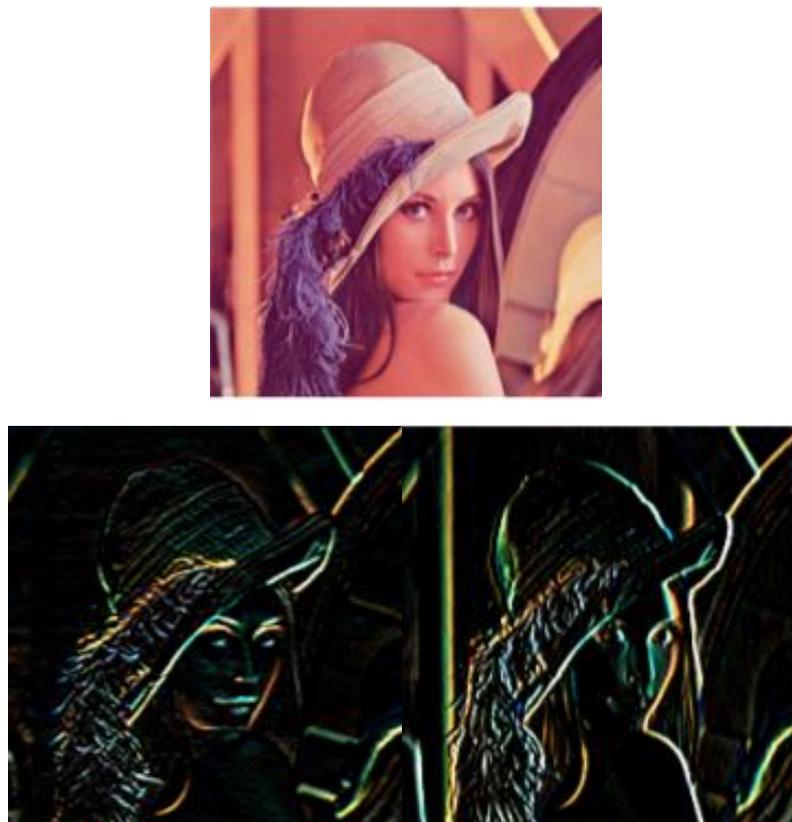


Figura 2.28. Resultado de aplicar distintos detectores de rasgos en una imagen [46].

Posteriormente a esta capa de convolución, se aplica la función ReLu. Esta se aplica ya que en la capa anterior, se llevaron a cabo operaciones lineales, y lo que buscamos es aumentar la no linealidad, ya que en las imágenes los rasgos son altamente no lineales [46].

A continuación se muestra el ejemplo de una imagen original (Figura 2.29), una imagen convolucionada y una imagen tras aplicar la función ReLu (Figura 2.30).



Figura 2.29. Imagen original

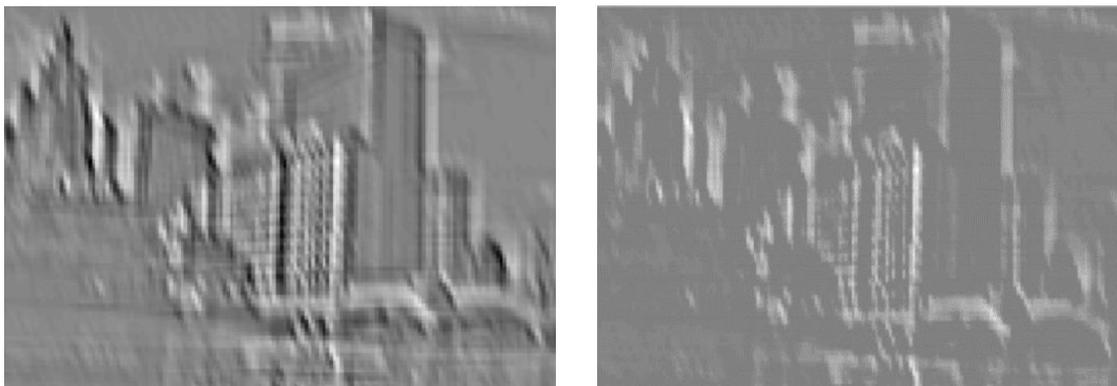


Figura 2.30. Imagen tras convolución (izquierda) e imagen tras función ReLu (derecha) [47].

Ahora se pasaría a la capa de agrupación máxima, *max pooling*, partiendo del mapa de características convolucional, con valores 0 o positivos, ya que pasó por la capa ReLu. Esta capa reduce todavía más la dimensión, obteniendo un mapa de características agrupado, agrupando la información de píxeles cercanos. Se aplica un filtro más pequeño que en la capa convolucional, y lo que se va a buscar es el valor máximo que hay dentro del filtro. Esto lo que permite es que se intenta agrupar la información lo máximo posible preservando las características, y teniendo en cuenta rotaciones espaciales, distorsiones y cambios de escala [45].

A continuación, se lleva a cabo la operación de aplanado (*flattening*), que consiste en pasar los valores de la matriz del mapa de características agrupado a un vector unidimensional, colocando los valores de las filas de la matriz uno debajo del otro, que consiste en la capa de entrada de la red neuronal artificial; donde la siguiente capa, o capas, sería la capa completamente conectada (*fully connected*), a partir de la cual la red neuronal realizará una posible predicción del valor de entrada mediante una asignación de pesos y una corrección de la función de coste, como se vio anteriormente [48].

2.6 Visión artificial, inteligencia artificial y seguimiento solar en energía solar.

Al combinar estas disciplinas de visión artificial y aprendizaje automático como ocurre en este trabajo, se puede crear un sistema de seguimiento solar inteligente, con el fin de avanzar en la aplicación de estas técnicas para beneficio de los seguidores solares. De este modo, la red es capaz de captar la posición del sol y la posición del receptor en todo momento, así orientándose de manera óptima siguiendo las suposiciones vistas anteriormente.

Estudios anteriores indican que además, este enfoque a partir del empleo de la visión artificial permite detectar algunas otras variables a tener en cuenta relacionada con el control de sistemas de seguimiento solar, que actualmente se procesan aparte debido a la dificultad de su obtención; como pueden ser la sombra solar proyectada por el colector, o el reflejo solar bloqueado por otros heliostatos [51].

La predicción del movimiento de las nubes también se puede obtener a partir de las imágenes, de manera que el sistema puede actuar de acuerdo a ello para mitigar los efectos provocados por estos cambios transitorios. Otra variable clave que se puede medir a partir de imágenes es la distribución del flujo solar concentrado, es decir, el flujo solar total reflejado que llega al receptor.

Como se analizó anteriormente, la pérdida del flujo de radiación solar debido al efecto de la atmósfera es una importante causa de pérdida de energía en plantas solares, lo que se puede medir también a partir de imágenes digitales.

Este sistema puede seguir el movimiento solar desde el amanecer hasta el atardecer mediante una cámara de lente gran angular, un procesador integrado, Raspberry Pi, y una batería solar portable para suministrar su pequeña demanda energética. Por tanto tenemos un dispositivo de bajo coste y baja demanda energética que soluciona los problemas presentados por los sistemas tradicionales (calibrado periódico, altos requerimientos instalación y mantenimiento, predicción de nubes, bloqueos y sombras) y reduce costes (bajo coste, mejora eficiencia, menor coste de instalación, mantenimiento, cableado) [51].

En la Figura 2.31, se muestra una foto captada con una cámara posicionada en un heliostato de la Plataforma Solar de Almería, donde el círculo rojo es el punto de apunte, lo que se llamó A'; y el círculo blanco es el punto objetivo, A'', punto intermedio entre el vector solar y el vector del receptor. La diferencia entre ambos puntos es el error de seguimiento.



Figura 2.31. Imagen analizada por una red neuronal convolucional en la PSA [51].

2.7 Software y librerías disponibles de IA.

El lenguaje informático que se utilizará en este trabajo es Python, el cual es un lenguaje de programación de tipado dinámico, que tiene una sintaxis que favorece la lectura del código incluso a quienes no lo han desarrollado. Además es de código abierto y hay una gran cantidad de repositorios abiertos en internet. Sus características son:

- Interpretado: Se ejecuta sin necesidad de ser procesado por el compilador y se detectan los errores en tiempo de ejecución.
- Multiparadigma: Soporta programación funcional, programación imperativa y programación orientada a objetos.
- Tipado dinámico: Las variables se comprueban en tiempo de ejecución.
- Multiplataforma: disponible para plataformas de Windows, Linux o MAC.
- Gratuito: No dispone de licencia para programar.

Este lenguaje vio la luz sobre principios de la década de los 90 y, en un inicio, fue desarrollado como un pasatiempo por Guido Van Rossum, un ingeniero holandés que trabajaba en ese momento en el Centro de Investigación de Ciencias de la Computación holandés. Como curiosidad, el nombre elegido, Python, fue tomado del grupo cómico británico Monty Python, del que Guido era un gran fan [56].

Sumado a esto cuenta con grandes compañías que hacen de este un uso intensivo. Tal es el caso de Google, Facebook o Youtube, ya que permite, entre otras de sus características, la automatización de procesos y ejecución de tareas en tanto en entorno cliente como servidor.

En cuanto a las librerías que se usarán, TensorFlow [21] será la predominante. Es una interfaz para expresar algoritmos de aprendizaje automático, que abarca desde dispositivos móviles como tablets; hasta sistemas distribuidos de larga escala de cientos de máquinas y miles de dispositivos computacionales como tarjetas GPU.

Tensorflow puede ser utilizado para expresar una amplia variedad de algoritmos, incluidos algoritmos de entrenamiento e inferencia para modelos de redes neuronales. También ha sido usado para desarrollar sistemas de aprendizaje automático a través de más de una docena de áreas en la ciencia de la computación y otros campos, incluyendo reconocimiento de voz, visión por computador, robótica, procesamiento del lenguaje natural...

Esta librería fue creada por Google bajo el proyecto de 'Google Brain', para explorar el uso del aprendizaje profundo tanto como para investigación como para implementación en los productos de Google [56].

2.8 Detección de objetos y segmentación semántica.

Las redes neuronales se han convertido en el método más popular para una eficiente detección de objetos. En esta sección se mostrará algunos de los modelos que llevan esto a cabo, y se discutirá sobre sus posibles ventajas y desventajas.

Primero se aclarará algunos términos necesarios para la posterior explicación de este tema. En primer lugar, se encuentra la clasificación de imágenes, la cual consiste en asignar una etiqueta de alguna categoría a una imagen; mientras que la localización de objetos consiste en dibujar un cuadro delimitador, o *bounding box*, alrededor de uno o más objetos en la imagen. Este cuadro delimitador es el rectángulo que encierra todos los píxeles que parecen indicar que dicha área es un objeto. Finalmente, la detección de objetos combina estas dos técnicas, y marca el cuadro delimitador sobre cada objeto en la imagen y le asigna una categoría. Se muestra en Figura 2.32 un ejemplo de cuadros delimitadores para figuras geométricas [52].

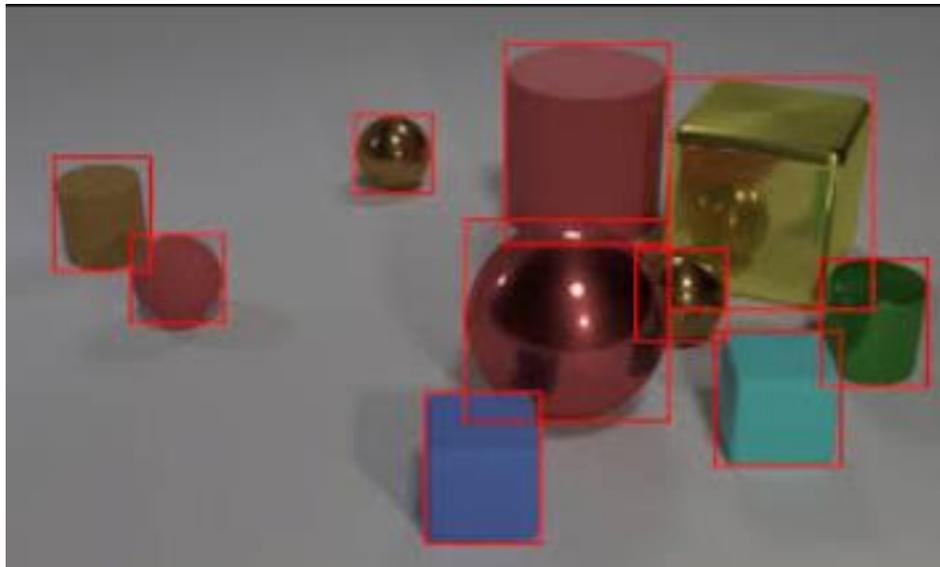


Figura 2.32. Cuadros delimitadores para figuras geométricas [65].

La segmentación semántica consiste en asociar una categoría a cada píxel presente en una imagen. Se utiliza para reconocer un conjunto de píxeles que conforman distintas categorías. Por ejemplo, un vehículo de conducción autónoma necesita identificar vehículos, peatones, señales de tráfico, aceras y otros elementos de la carretera.

La segmentación semántica puede ser una útil alternativa a la detección de objetos, pues permite que el objeto de interés abarque diferentes áreas de la imagen en el nivel de píxel. Esta técnica detecta claramente objetos que tienen una forma irregular, al contrario que la detección de objetos, en donde los objetos deben encajar en un cuadro delimitador. Debido a que la segmentación semántica etiqueta los píxeles de una imagen, es más precisa que otras formas de detección de objetos [53]. Se ejemplifica esta diferencia en Figura 2.33.

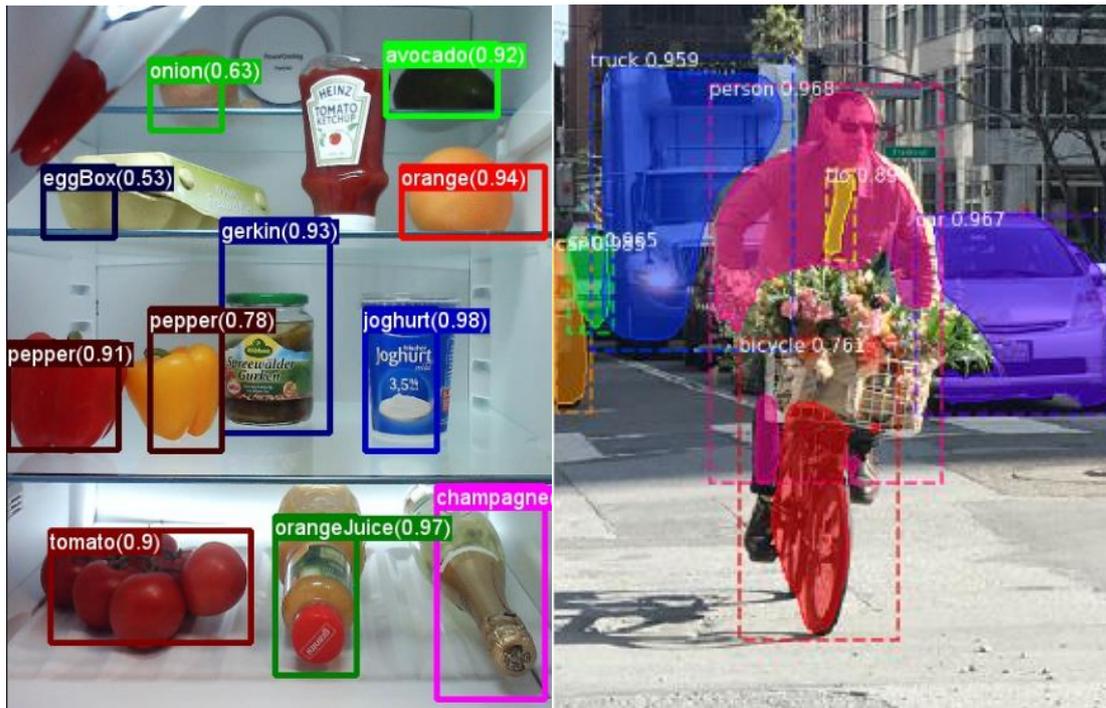


Figura 2.33. Comparativa entre detección de objetos y segmentación semántica [54].

El desarrollo del estudio de la detección de objetos y segmentación semántica se ha desarrollado muy rápidamente a lo largo de los últimos años, por lo que han ido surgiendo nuevos e innovadores métodos continuamente.

La métrica estándar de precisión, mAP, la cual se mide de 0 a 100 y cuanto más elevada, mejor; no es suficiente, sino que hay que tener en cuenta otros parámetros críticos como el tiempo de ejecución o el uso de memoria. Por ejemplo, los dispositivos móviles usualmente requieren una pequeña capacidad de memoria; mientras que los coches autónomos requieren una actuación excelente en tiempo real [54].

La detección de objetos es modelado como un problema de clasificación, en el cual se toma distintos cuadros delimitadores de un tamaño fijo sobre la imagen de entrada, y estos son posteriormente analizados por el clasificador de imágenes. Se presenta un problema principal y es que no se sabe cuántos objetos habrá. Además, se encuentra el problema del tamaño del cuadro delimitador a definir, y el ratio de aspecto de este, es decir, la relación entre la altura y la anchura del cuadro.

Para resolver estos problemas, se tendría que probar diferentes tamaños y formas de cuadros delimitadores, lo que es muy intenso computacionalmente, sobre todo en el tema de redes neuronales profundas. En la práctica, se encuentran los dos tipos de algoritmos de detección de objetos más comerciales. Los algoritmos de la familia R-CNN, que se basan en un proceso de dos pasos, primero identificar regiones donde se espera que hay objetos, y después detectar los objetos en esas regiones. Por otro lado, se encuentran algoritmos como YOLO (*You Only Look Once*) y SSD (*Single Shot Multibox Detector*), que usan técnicas en las que la red es capaz de encontrar todos los objetos en la imagen en una sola pasada.

Los algoritmos de proposición de regiones normalmente tienen una ligera mejor precisión pero son más lentos de ejecutar, mientras que los algoritmos de una pasada son más eficientes y con una buena precisión, es por eso que nos centraremos únicamente en algoritmos de SSD [55].

2.8.1 SSD (Single Shot MultiBox Detector)

Se usa este término para aludir a la familia de métodos que usan arquitecturas de redes convolucionales simples que directamente hacen la predicción de las clases y de los cuadros delimitadores de los diferentes objetos de la imagen, sin necesitar una segunda etapa de operación de clasificación, como los métodos anteriormente descritos; es por eso que se le denomina *Single Shot* [59].

SSD es diseñado para detección de objetos en tiempo real. Faster R-CNN utiliza una red de regiones de interés para crear cuadros delimitadores y utilizar estos cuadros para clasificar objetos. El proceso completo es ejecutado en 7 fotogramas por segundo, muy lejos de lo que un proceso a tiempo real precisa. SSD acelera el proceso eliminando la necesidad de la red de regiones de interés. Para recuperar la caída en la precisión, SSD aplica algunas mejoras, como las funciones de escala múltiple. Estas mejoras permiten que el SSD coincida con la precisión del Faster R-CNN, utilizando imágenes de menor resolución, lo que aumenta aún más la velocidad, llegando a alcanzar la velocidad de procesamiento necesaria para tiempo real, otro motivo por lo que se descarta Faster R-CNN [60]. En 2016 se publicó el artículo original de SSD, alcanzando nuevos records en términos de actuación y precisión para tareas de detección de objetos, con un 74% mAP a 59 fotogramas por segundo en conjuntos de datos estándares como COCO.

SSD usa VGG16, una red neuronal convolucional creada por la Universidad de Oxford [61], pero descarta las capas completamente conectadas. La razón por la cual VGG16 fue usada como la red base es debido a su robusta actuación en tareas de clasificación de imágenes de alta calidad, y su popularidad para problemas donde el aprendizaje por transferencia ayuda a mejorar los resultados. En lugar de las capas completamente conectadas de la red VGG, un conjunto de 6 capas de convolución fueron añadidas, como se aprecia en la figura, de este modo permitiendo extraer características en múltiples escalas y progresivamente reduciendo el tamaño de la entrada. De este modo se realizan un total de 8732 predicciones para una imagen de entrada 300x300, como se ve en Figura 2.34 [62].

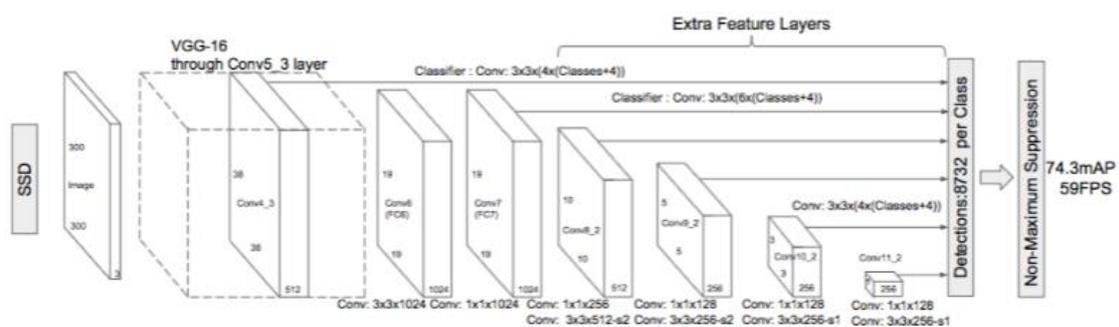


Figura 2.34. Estructura del modelo SSD [62].

Por otro lado, la técnica contiene la palabra *Multibox* ya que es una técnica de regresión de cuadros delimitadores empleada que se basa en la rápida clasificación categórica de dichos cuadros. Investigadores crearon las llamadas *priors*, cuadros delimitadores prefijados de un tamaño concreto que se emparejan con la distribución de los cuadros que delimitan el objeto real, como se ve en la Figura 2.35. Estos *priors* son seleccionados de tal modo que su Intersección sobre la Unión (IoU) (Figura 2.36), sea mayor que 0.5. Un IoU de 0.5 no es suficientemente bueno pero nos proporciona un buen punto de partida para el algoritmo de regresión de cuadros delimitadores.

Por tanto, la técnica *Multibox* empieza con *priors* como predicciones y el algoritmo de regresión intenta llegar lo más cerca posible a los cuadros delimitadores reales, es decir, aquellos en los que está contenido el objeto [59].

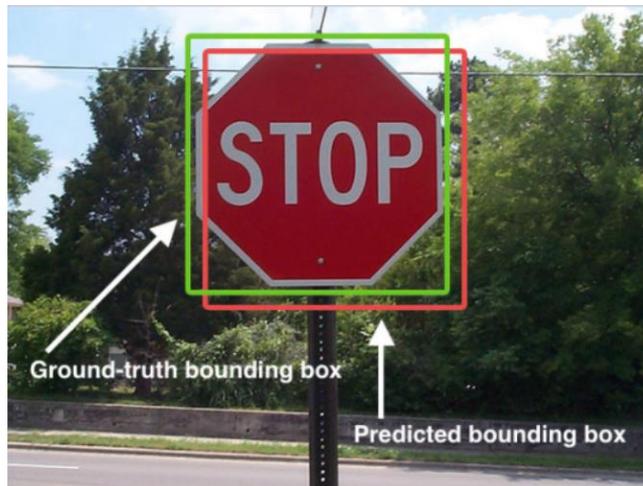


Figura 2.35. Ejemplo de cuadros delimitadores [59].

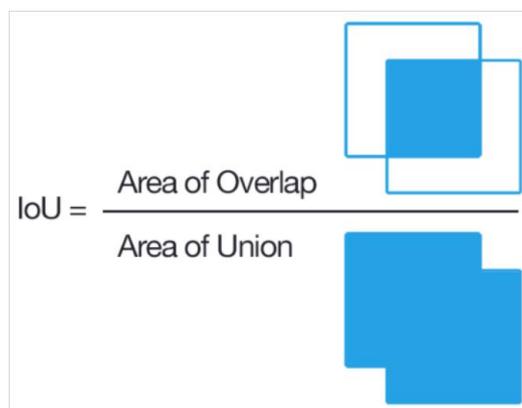


Figura 2.36. Representación de la intersección sobre la unión (IoU) [59].

La arquitectura de *Multibox* resultante contiene 11 *priors* por cada celda de mapa de características y solo uno en el mapa de características 1x1, resultando en un total de 8732 *priors* por imagen, así proporcionando una robusta solución para imágenes de entrada de múltiples escalas para detectar objetos de varios tamaños. Como 8732 posibles predicciones son demasiadas, se usan las pérdidas de localización y de confianza para solo quedarse con las predicciones más acertadas.

- La pérdida de confianza mide como de segura esta la red de que en el cuadro delimitador hay un objeto categórico.
- La pérdida de localización mide como de separadas están el cuadro delimitador predicho por la red y el cuadro delimitador real del objeto del conjunto de entrenamiento.

De este modo, se define la pérdida de *Multibox* como la suma de la pérdida de confianza más el producto de la pérdida de localización por alfa, un término que ayuda a hacer un balance de la contribución de esta pérdida al resultado total.

Así pues, el objetivo es encontrar el valor de los parámetros que reduzcan al máximo esta función de pérdida, así acercando las predicciones lo máximo posible al cuadro delimitador real [64]. En Figura 2.37, se ve como la arquitectura SSD analiza la imagen a predecir.

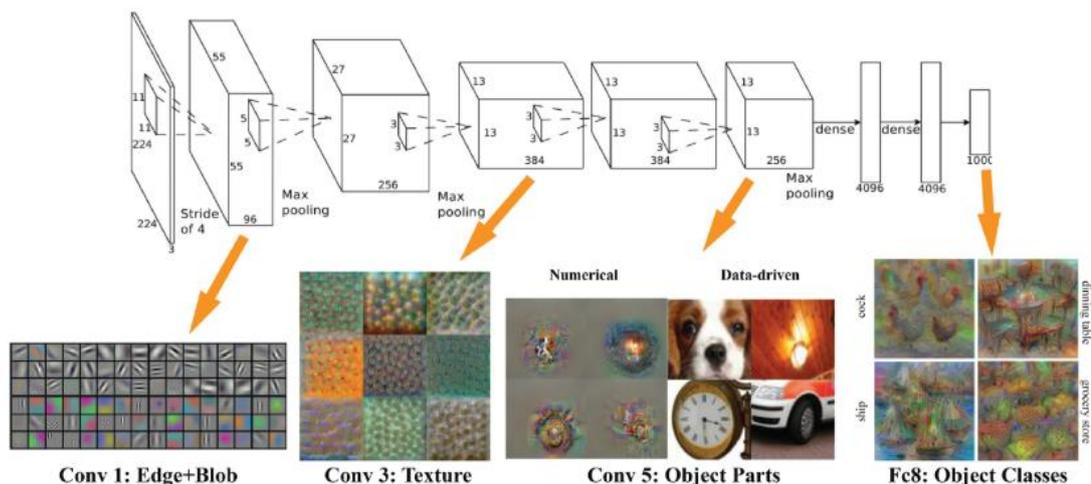


Figura 2.37. Resultados de una serie de convoluciones [62].

Finalmente, algunas observaciones sobre la arquitectura SSD [62]:

- Más cuadros delimitadores estándar definidos resulta en una detección más precisa, aunque esto conlleva un impacto negativo en la velocidad.
- Usando el conjunto de datos público Pascal VOC2007, SSD-500 (la variante SSD de mayor resolución que usa imágenes de entrada de 512x512) consigue un mejor mAP, pero a costa de la velocidad, la cual baja a 22 fotogramas por segundo. SSD-300 tiene un mejor balance con 74.2 mAP a 59 fotogramas por segundo.
- SSD tiene un resultado peor con objetos pequeños, ya que quizás no aparezcan en todos los mapas de características y no puedan ser estudiados. Esto se puede solucionar incrementando la resolución de la imagen, pero no llega a solventarse del todo dicho problema.

2.8.2 Arquitecturas de redes neuronales

Una vez definidos brevemente el método, se pasa a explicar las distintas redes neuronales convolucionales más comúnmente usadas para una tarea de detección de objetos. Estas son *Inception*, *ResNet* y *MobileNet*.

Las redes basadas en módulos *Inception* proveen bloques convolucionales apilados uno sobre el otro. Dentro de los bloques, el método usa convoluciones 1x1 para reducir el número de parámetros de la red residual ofreciendo así la posibilidad de saltar algunas capas durante el entrenamiento. En redes muy profundas, una red convolucional con conexiones residuales suele tener mejor resultados que las redes con módulos *Inception*. Esto se debe al problema de la degradación por la búsqueda de redes más y más profundas [73].

ResNet nació de una simple observación: ¿por qué las redes neuronales muy profundas tienen una peor actuación a medida que les vas añadiendo capas?

Hace unos años los investigadores de inteligencia artificial creían que a mayor profunda la red, mejor actuación tendría; ya que intuitivamente, éstas no tendrían por qué tener un desempeño peor que aquellas redes más superficiales, al menos en la fase de entrenamiento, cuando no hay riesgo de *overfitting* [74].

Esto se puede esquematizar de manera que construimos una red con n capas que consigue una cierta precisión. Como mínimo, una red con $n+1$ capas debería ser capaz de conseguir la misma precisión, aunque solo sea una copia de las primeras n capas, y reproduciendo del mismo modo que las anteriores una última capa. Similarmente podría ocurrir con redes de $n+2$, $n+3$ y $n+4$ capas; sin embargo, en la práctica, estas redes más profundas casi siempre sufren una degradación en los resultados.

Los desarrolladores de *ResNet* han reducido este problema a la hipótesis de que las asignaciones directas son difíciles de entrenar. Por lo tanto, propusieron una solución: en lugar de tratar de aprender a partir de mapeos subyacentes de x y $H(x)$, es posible aprender la diferencia entre los dos, que es el “residuo”, y posteriormente, ajustar el último a la entrada. Supongamos que el residuo es $F(x) = H(x) - x$. Ahora nuestra red intenta aprender de $F(x) + x$ [75].

Se encuentran *Resnet 50* y *101*, la cual la primera tiene 50 capas profundas y la segunda, 101 [76].

Por último, *MobileNet* es una familia de modelos de visión por computador para TensorFlow, diseñados para maximizar la precisión a la vez que siendo consciente de los recursos limitados que posee una aplicación para móvil. Cuando ésta se compara con otros modelos similares como *Inception*, *MobileNet* trabaja mejor en términos de latencia, tamaño y precisión si el modelo es desarrollado en un dispositivo móvil para detección en tiempo real [77].

2.9 Software para etiquetado

Para empezar con el entrenamiento de la red neuronal, lo primero que se debe de hacer es preparar un conjunto de datos de entrenamiento y de validación robusto. De este modo, se puede entrenar la red neuronal, y posteriormente comprobar si ha sido correctamente entrenada con el set de validación. El preprocesamiento de datos se basa en adaptar las imágenes de entrada, en este caso, para que sea entendible por nuestro algoritmo clasificador.

Seguidamente debemos etiquetar las imágenes del conjunto de datos con cuadros delimitadores sobre los objetos a detectar si hablamos de entrenamiento supervisado; para que posteriormente, la red neuronal entrenada con este set de datos sea capaz de detectar y localizar estos objetos por ella misma.

Encontramos numerosas herramientas para realizar el etiquetado como *LabelMe*, *Labelbox* o *BeaverDam*, siendo esta última la más usada para detección en videos; pero en este caso se usará *Labelbox* (<https://labelbox.com/>), se ve un ejemplo en la Figura 2.38, por ser una herramienta que se puede usar en el navegador, lo que facilita el tedioso trabajo, es gratuita y sin límites.

Se distinguirán cuatro categorías distintas:

1. *Sun*: Posición del sol.
2. *Target*: Diana blanca situada en la parte central de la torre del sistema CESA
3. *Heliostat*: Heliostatos que aparecen en la imagen
4. *Cloud*: Nube

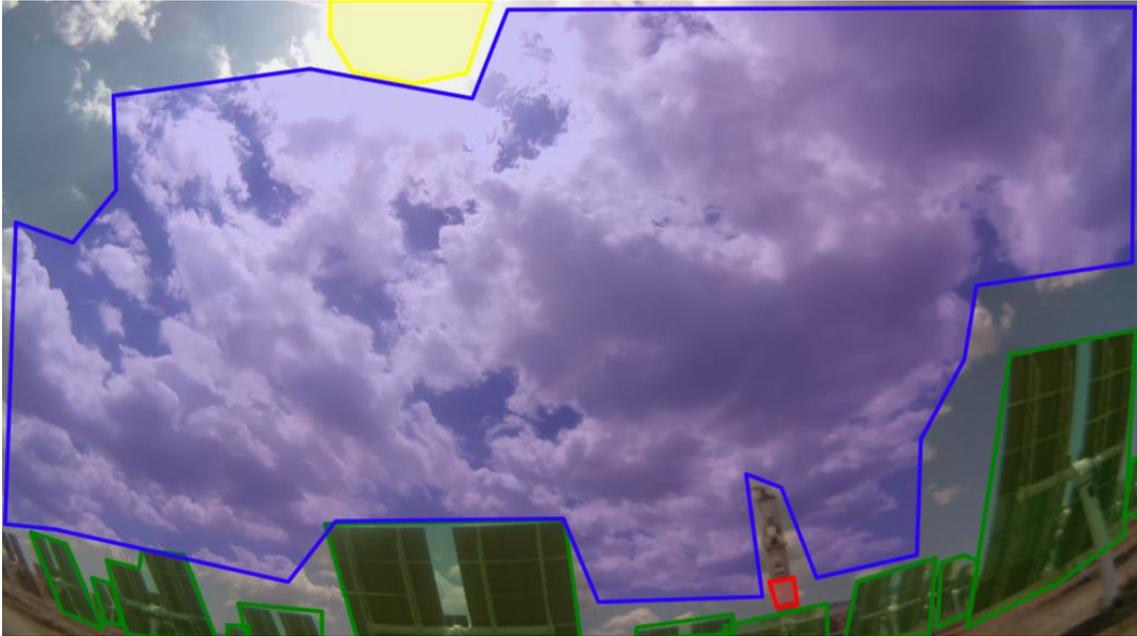


Figura 2.38. Ejemplo herramienta LabelBox para etiquetado de una imagen [40].

2.10 Configuración del hardware, software y datos

Para esta configuración, se ha habilitado el acceso a un supercomputador perteneciente al Centro Extremeño de Tecnologías Avanzadas, dependiente del Ciemat, (CETA-Ciemat), el cual gestiona un avanzado centro de procesado de datos que constituye uno de los centros de recursos para la computación científica más potentes en toda España. El centro de datos del CETA-Ciemat ofrece a sus usuarios e investigadores más de 100 Teraflops de potencia de cómputo de doble precisión (1 Teraflop = 10¹² operaciones matemáticas por segundo), 3.000 núcleos CPU de computación, 125.000 núcleos de computación GPU, y más de 700 Terabytes de almacenamiento de altas prestaciones con tasas de acceso a disco de hasta 64 Gbps [10]. En la Figura 2.39, se muestra la interfaz de inicio al acceder al supercomputador:

Para lanzar entrenamientos, se creó un entorno virtual dentro de una carpeta remota, en donde se copian los modelos entrenados previamente en el set de datos COCO, cambiándoles la configuración, o *pipeline*, adecuada para el proyecto a seguir, es decir, para un sistema de seguimiento solar. Indicar que el conjunto de datos con el que se van a reentrenar, tiene 1263 imágenes; de las cuales se han escogido 963 para el set de entrenamiento, y 300 para el set de validación; siguiendo la norma general en aprendizaje profundo de 20% test - 80 % entrenamiento.

3. MÉTODOS

3.1 Configuración de parámetros

A continuación se pasa a describir la enumeración y definición de los diferentes parámetros de configuración que constituyen los diferentes modelos que se entrenarán; con el fin de entender cada uno de ellos y así poder modificarlos con criterio para optimizar los modelos lo máximo posible. En redes neuronales, los parámetros son usados para entrenar al modelo y hacer predicciones. Hay dos tipos de parámetros [66]:

- Los parámetros del modelo son internos a la red neuronal, por ejemplo, los pesos de la neurona. Son estimados o aprendidos automáticamente mediante entrenamiento.
- Los hiperparámetros son parámetros externos definidos por el operador de la red neuronal, por ejemplo, función de activación o el tamaño del *batch* usado en el entrenamiento. Estos tienen un impacto enorme en la precisión de la red neuronal, puede haber diferentes valores óptimos para diferentes parámetros.

Se presenta una lista de hiperparámetros comunes:

- Hiperparámetros relacionados con la estructura de la red neuronal:
 1. Número de capas ocultas: añadir más capas ocultas de neuronas generalmente mejora el resultado, hasta un cierto límite donde empezaría a degradarse.
 2. *Dropout*: qué porcentaje de neuronas deberían ser aleatoriamente ‘sacrificadas’ durante cada época para prevenir el problema de *overfitting*.
 3. Función de activación: es la función que debería ser usada para procesar las entradas de cada neurona. Esta función puede impactar la habilidad de la neurona para converger y aprender para diferentes rangos de valores de entrada, así como la velocidad del entrenamiento.
 4. Inicialización de pesos: es necesario establecer unos pesos iniciales para la primera pasada, siempre que no estemos hablando de un proceso de *transfer learning*. Las dos opciones básicas son establecerlos en cero o seleccionarlos aleatoriamente.
- Hiperparámetros relacionados con el algoritmo de entrenamiento:
 1. Ratio de aprendizaje o *learning rate*: como de rápido el algoritmo de propagación hacia atrás actúa, es decir, cuanto modifica los parámetros internos de la red. Un bajo valor puede hacer que la red aprenda rápidamente pero puede resultar en que pierda el mínimo de la función de pérdida. el ratio de aprendizaje tiene un valor positivo pequeño, usualmente en el rango entre 0.0 y 1.0.
El entrenamiento debería empezar desde un ratio grande porque en el principio, los pesos aleatorios están lejos de ser los óptimos, y después el ratio puede decrecer durante el entrenamiento para permitir una actualización de pesos más ajustada.

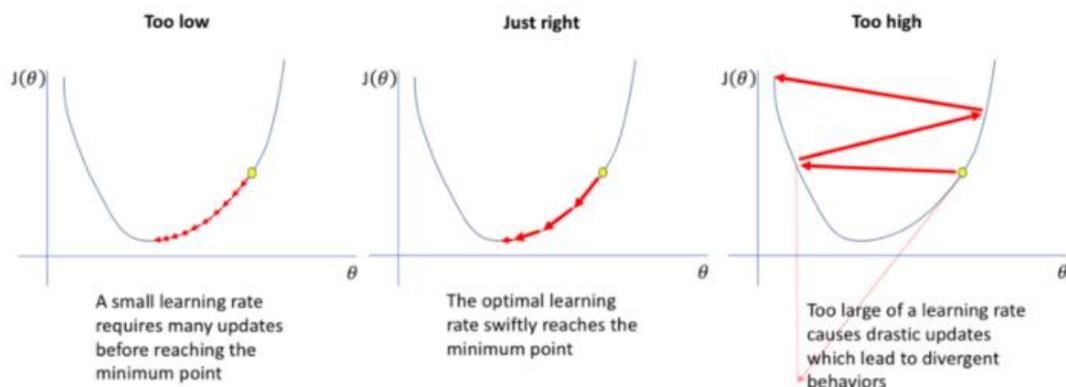


Figura 3.1. Esquema de varios ratios de aprendizaje [66].

2. **Épocas, iteraciones y tamaño del *batch*:** Una época es un grupo de muestras las cuales pasan por el modelo juntas y a continuación se realiza la propagación hacia atrás para determinar los pesos óptimos. Si la época no puede ejecutarse a la vez debido al tamaño de la muestra o por la complejidad de la red, ésta se divide en *batches*, y la época se realiza en una serie de iteraciones. El número de épocas y de *batch*, puede afectar significativamente al modelo.
3. **Optimizador y *momentum*:** cuando una red neuronal entrena, usa un algoritmo para determinar los pesos óptimos para el modelo llamado optimizador. La opción básica es el gradiente descendente estocástico (SCD), pero existen otras opciones. Otro algoritmo común es *momentum*, el cual trabaja esperando a que un peso sea actualizado, y este lo actualiza una segunda vez utilizando un valor delta. Esto acelera el entrenamiento gradualmente y reduce el riesgo de oscilación.

Como el trabajo se centrará principalmente en SSD, se entrará en la descripción un poco más detallada del archivo de configuración para conocerlo mejor, aunque muchos de estos parámetros se dejarán en el valor predeterminado.

3.1.1 Parámetros SSD

- ***num_classes*:** es el número de objetos a ser detectados, en nuestro caso 4 clases: sol, heliostatos, receptor y nubes.
- ***image_resizer*:** el dimensionamiento de las imágenes es muy importante para la detección de objetos. Este modelo necesita que las imágenes de entrada tengan siempre el mismo tamaño para que el mapa de características tenga el mismo tamaño, por lo que se usa la siguiente opción:
 - ***Fixed_shape_resize*:** Esta rellenará las imágenes de menos dimensión con píxeles negros o ruido blanco aleatorio, en lugar de deformarlas, así mejorando la estabilidad y los ratios de aspecto. Por ejemplo, si cambiamos el tamaño de una imagen 1000x800 para que sea 416x416, el borde 1000 se convierte en 416, y el 800 en 332.8. El espacio entre 332.8 y 416 se convierte en relleno [85].

- **depth_multiplier:** es usado para reducir el número de canales en cada capa. Sus valores están fijados normalmente entre 0 y 1 para reducir el número de parámetros o el costo computacional del modelo [89].
- **min_depth:** este parámetro fija la profundidad mínima de cada una de las salidas de los mapas de características producidas por el extractor de características. Este y el anterior parámetro controlan la profundidad.
- **conv_hyperparams:** estos son los hiperparámetros de la convolución, es decir, las variables que determinan la estructura de la red y como es entrenada. [90] Dentro de estos, se definen:
 - **Regularizer:** reduce significativamente la varianza del modelo sin un cambio importante en su sesgo o *bias*, es decir, la diferencia entre la predicción esperada de nuestro modelo y los valores verdaderos. El regularizador conocido como norma L2, es el estándar y funciona bien con la mayoría de modelos; pero también hay otros como la norma L1 [91].
 - **Initializer:** puede ser una distribución normal truncada o aleatoria. La diferencia es que la distribución truncada desecha los valores con una desviación mayor a un valor especificado, a diferencia de la distribución normal. La idea es que usando la distribución truncada se supera la saturación de algunas funciones como la sigmoidea, donde si el valor es muy extremo, la neurona para de aprender [92].

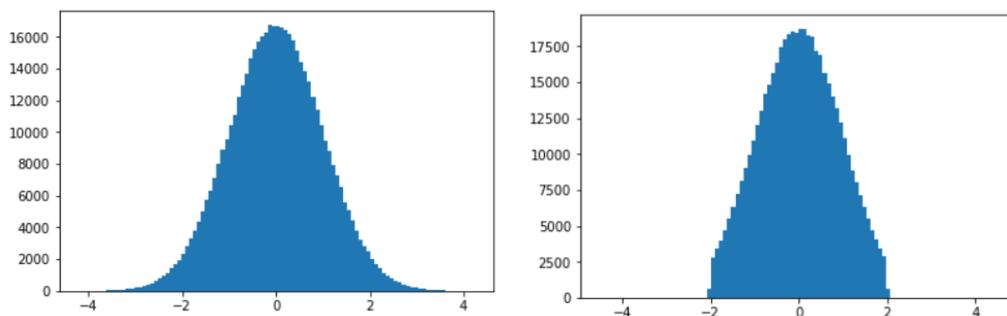


Figura 3.2. Distribución normal a la izquierda, y distribución truncada en la derecha [92].

- **Activation:** la función de activación se usan para introducir no linealidad a los modelos. Generalmente la función RELU es la más popular. Otras funciones, como la función sigmoidea se usa en la capa de salida cuando se hacen predicciones binarias; y la función *softmax* se usa cuando se hacen predicciones de clases múltiples. [93] Para optimizar encontramos parámetros como la escala, el centro, o *decay*, el cual se encuentra cercano al 1, normalmente con valores como 0.999,0.99, probar como mínimo 0.9 si el modelo experimenta un razonable buen entrenamiento pero una validación pobre.
- **Matcher:** la función de este módulo es juntar filas y columnas basadas en la matriz de similitud y otros parámetros; es decir, se establecen límites para dar por válido, o no, cuando una proposición se solapa con un recuadro de objeto a predecir, para establecerlo como un ejemplo positivo o negativo [94].
- **use_dropout:** *dropout* es una técnica de regularización usada para evitar el riesgo de *overfitting*, que es cuando un modelo tiene un resultado muy bueno para el conjunto de entrenamiento, y unos resultados pobres para el conjunto de validación.
- **dropout_keep_probability:** el valor para *dropout* en una capa escondida esta entre 0 y 1.
- **batch_size:** define el número de elementos de trabajo en el lote. Este valor es altamente dependiente del hardware del GPU y las dimensiones de la imagen, y no es estrictamente necesario para resultados de calidad, aunque se ha observado que en la práctica cuando se usa un valor más grande hay una degradación en la calidad del modelo.
- **optimizer:** es un parámetro muy importante y define como los pesos se actualizarán por propagación hacia atrás.

El modo estándar es *momentum_optimizer* el cual es una versión flexible del gradiente descendente estocástico. Este funciona bien para la mayoría de los sistemas, pero para arrays grandes, *Adam_optimizer* funciona mejor.

- ***box_predictor***: este bloque hace referencia a las clases que cogen mapas de características de la imagen como entrada y realizan dos predicciones, una de la localización del cuadro que delimita al objeto, y la otra del tipo de clase de cada cuadro. Tiene una programación parecida a la del extractor de características; junto con algunos parámetros más, éstos son la posibilidad de usar *dropout* y su probabilidad; el tamaño del núcleo final de convolución o si aplicar la función sigmoïdal a la salida de las predicciones de clase [95].
- ***anchor_generator***: su función es generar una serie de cuadros delimitadores para ser usados como *anchors*. Se eligen el número de capas y los ratios de aspecto [96].
- ***post_processing***: bloque para configurar la operación de supresión no máxima en un lote de detecciones. Se establece un límite escalar para eliminar los recuadros con valores bajos; se eliminan también los recuadros que tienen una IoU alta con recuadros previamente seleccionados; se fija un número de detecciones para retener por clase y un número máximo para retener entre todas las clases [97].
- ***normalize_loss_by_num_matches***: se establece si se quiere normalizar la pérdida por el número de cuadros que contienen objetos que coinciden con los *anchors*.
- ***hard_example_miner***: se establece un ratio entre los ejemplos predichos positivos y negativos, ya que el número de negativos probablemente sea mucho mayor. Generalmente se escoge un ratio 3:1 entre negativos y positivos. Hacer esto trae una optimización más rápida y un entrenamiento más estable [98].
- ***num_steps***: el número de pasos de entrenamiento depende de la velocidad de aprendizaje y del tamaño del lote. Normalmente se calcula como tamaño de datos de entrenamiento / tamaño del lote · épocas.
- ***data_augmentation_options***: ajustar algunas opciones de aumento puede incrementar el tamaño del conjunto de datos drásticamente, a la vez que mejorando la robustez de nuestro detector.
- ***train_config***: se establece el tamaño del lote para usar en el entrenamiento; alguna técnica para aumentar los datos de entrada y de esta manera se hace el modelo más robusto; y aplicar algún optimizador.
- ***eval***: el bloque de evaluación donde se elige el número de ejemplos para proceso de evaluación; y el número máximo de veces para ejecutar la evaluación. Si se establece 0, se ejecutará sin parar.

3.1.2 Técnicas de optimización de parámetros

Como se puede apreciar, hay una multitud de diferentes parámetros, y una multitud de posibles valores que pueden tomar estos. Para encontrar unos parámetros que sean los más adecuados para nuestra red neuronal, y por tanto tenga una alta precisión, baja velocidad y las pérdidas de los diferentes sets sean parecidas, se encuentran cuatro distintas técnicas [67]:

- 1 Por sintonización manual de hiperparámetros: tradicionalmente, los hiperparámetros fueron sintonizados manualmente por prueba y error. Este método es comúnmente usado aún, y operadores experimentados pueden ‘adivinar’ valores de parámetros que conseguirán una alta precisión para el modelo de aprendizaje profundo. Sin embargo, hay una constante búsqueda para un método mejor, más rápido y automático para optimizar parámetros. Este será el método que se adoptará en este trabajo. Pros: muy simple y efectivo con operadores experimentados. Contras: nada científico, se desconoce si has alcanzado la máxima optimización para cierto modelo.

- 2 *Grid search*: es ligeramente más sofisticado que la sintonización manual. Involucra sistemáticamente probar múltiples valores de cada hiperparámetro, mediante reentrenando automáticamente el modelo por cada valor del parámetro. Por ejemplo, se puede utilizar un *grid search* para el tamaño del *batch* automáticamente entrenando el modelo para tamaños entre 10 y 100. El modelo ejecutará el entrenamiento las veces que se le indiquen para tamaños distintos y el tamaño del lote seleccionado será el que consiga una mayor precisión. Pros: proporciona más oportunidades de optimización. Cons: puede ser muy lento de realizar para números grandes de hiperparámetros.
- 3 Búsqueda aleatoria: probar valores aleatorios de hiperparámetros es verdaderamente más efectivo que los dos métodos ya mencionados. En otras palabras, en lugar de sistemáticamente probar para cubrir ciertas áreas prometedoras del espacio de trabajo, es preferible testear valores aleatorios que abarquen la totalidad del espacio de trabajo. Pros: según el estudio, provee una mayor precisión con menos ciclos de entrenamiento para problemas con una alta dimensionalidad. Cons: los resultados no son intuitivos, así siendo difícil de entender porque han sido elegidos dichos valores de hiperparámetros.
- 4 Optimización bayesiana: es una técnica que intenta aproximar el modelo entrenado con diferentes posibles valores de hiperparámetros. En otras palabras, entrena el modelo con diferentes valores de hiperparámetros, y observa la función generada por el modelo para cada set de valores. Lo hace una y otra vez, cada vez seleccionando valores que son ligeramente diferentes y ayudan a graficar el siguiente segmento relevante del espacio de trabajo. El algoritmo termina con una lista de posibles valores de hiperparámetros y funciones modelo, de las cuales predice la función más óptima del problema. Pros: se demuestra que la optimización bayesiana obtiene un resultado significativamente mejor comparada con la búsqueda aleatoria. Cons: como la búsqueda aleatoria, los resultados no son intuitivos y por tanto muy difíciles de mejorar, incluso entrenado por operadores experimentados [68].

3.2 Visualización de los resultados y Tensorboard:

Una vez visto los diferentes parámetros con los que se trabajarán, tenemos que buscar una métrica para optimizar. Lo que no se puede medir, no se puede mejorar. Para ello, en aprendizaje automático cuando se trabaja con *Tensorflow*, existe una herramienta, llamada *Tensorboard*, que nos permite visualizar y medir las métricas de nuestro modelo, como son las pérdidas o la precisión; nos permite ver el grafo de nuestra red neuronal o la evolución a lo largo de las épocas de las predicciones que realiza nuestro modelo. [69].

A continuación se presentan las métricas y las gráficas con las que se trabajará en la evaluación de los distintos modelos optimizados, y la explicación de estas.

Loss y loss1

Loss o la función pérdida usada para clasificación es la sigmoideal, y para la localización, L1. Esta es la que el gradiente descendente estocástico está intentando minimizar iterando con los pesos de la red neuronal. Al final de cada época durante el proceso de entrenamiento, la pérdida será calculada comparando las predicciones de salida de la red, y como respectiva entrada, las predicciones reales [70].

En el contexto de un algoritmo de optimización, la función usada para evaluar una posible solución (en este caso, un modelo y unos parámetros definidos) es la función objetivo.

Se persigue maximizar o minimizar la función objetivo, de modo que se busca un candidato que tiene la mayor o menor puntuación respectivamente. Normalmente, en redes neuronales lo que se busca es minimizar el error. De tal modo que la función objetivo es referida como una función de pérdida y el valor calculado por esta función es denominada simplemente como pérdida, o *loss*.

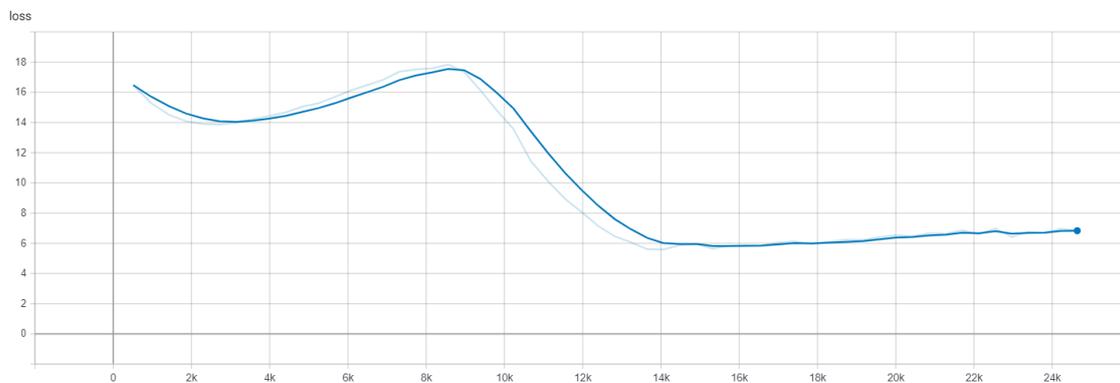


Figura 3.3. Gráfica de pérdida de set de evaluación.



Figura 3.4. Gráfica de pérdida de set de entrenamiento.

Se muestran las dos gráficas de pérdidas, a modo ilustrativo, reproducidas por *Tensorboard*, para el entrenamiento de un modelo.

En la parte superior, de color azul, se muestra la pérdida para el conjunto de evaluación, y en la parte inferior, de color naranja, para el conjunto de entrenamiento. En el eje de ordenadas se encuentran los valores de la pérdida, y en el eje de abscisas se muestran los pasos o *steps* que la red ha sido entrenada.

La idea es que se reduzca al máximo ambos valores, y que estas pérdidas tengan unos valores aproximados. En caso contrario se produciría lo que se denomina *overfitting*, que es cuando la red neuronal tiene unos resultados muy buenos en el set de entrenamiento; pero no puede generalizar más allá de este, es decir, tendrá un resultado pobre en el set de validación. Esto se da cuando tiene un bajo bias (predicciones acertadas para el set de entrenamiento) y una alta varianza (habilidad escasa de hacer predicciones acertadas en el set de validación). Se adjunta imagen de un ejemplo:

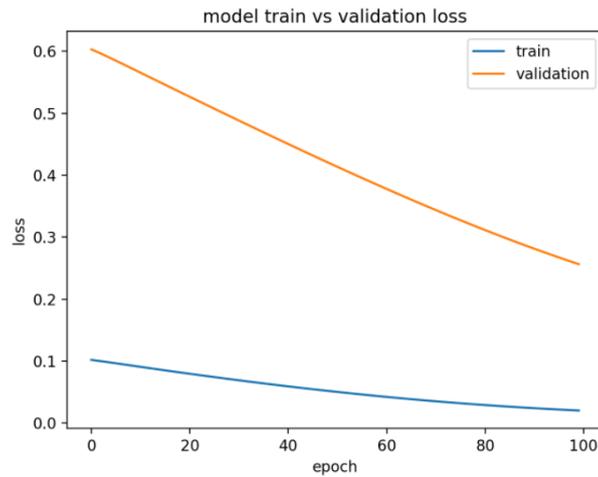


Figura 3.5. Ejemplo de *overfitting*.

Por otro lado, *underfitting* es cuando la red neuronal no es ni siquiera capaz de realizar buenas predicciones en el set de entrenamiento; esto se da por tener un alto bias (predicciones pobres en el set de entrenamiento) y una alta varianza (predicciones pobres en el set de validación). A continuación se adjunta un ejemplo de lo que sería un buen resultado en cuanto a la función pérdida:

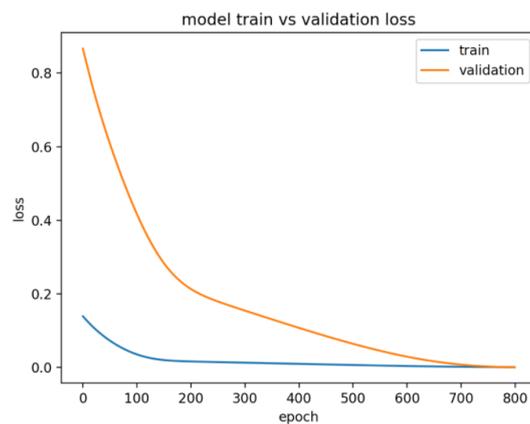


Figura 3.6. Ejemplo de un buen entrenamiento.

mAP:

Otra métrica que se usará para evaluar los modelos es la precisión media, medida en mAP, *Mean Average Precision*, que para explicarlo, se necesita primero entender que es la precisión y la exhaustividad o *recall*.

La precisión, Ecuación 3.1, mide como de exactas son las predicciones, es decir, que porcentaje de las predicciones son correctas; mientras que la exhaustividad, Ecuación 3.2, mide como de bien es capaz de encontrar todos los positivos. Se muestran a continuación sus definiciones matemáticas [79].

$$Precisión = \frac{TP}{TP+FP} \tag{3.1}$$

$$Exhaustividad = \frac{TP}{TP+FN} \tag{3.2}$$

Donde TP es un positivo verdadero, TN es un negativo verdadero, FP es un falso positivo y FN es un falso negativo.

Se podría decir que la precisión nos informa de la calidad del modelo; mientras que la exhaustividad hace referencia a la cantidad que el modelo es capaz de identificar.

Por otro lado, recordar que la intersección sobre la unión (IoU) mide el solapamiento entre 2 cajas delimitadoras que se corresponden a la predicción y al objeto real, como se muestra en Figura 3.7. En algunos conjuntos de datos, se predefine un límite de IoU (pongamos 0.5) para clasificar si la predicción es un positivo verdadero o un falso positivo.

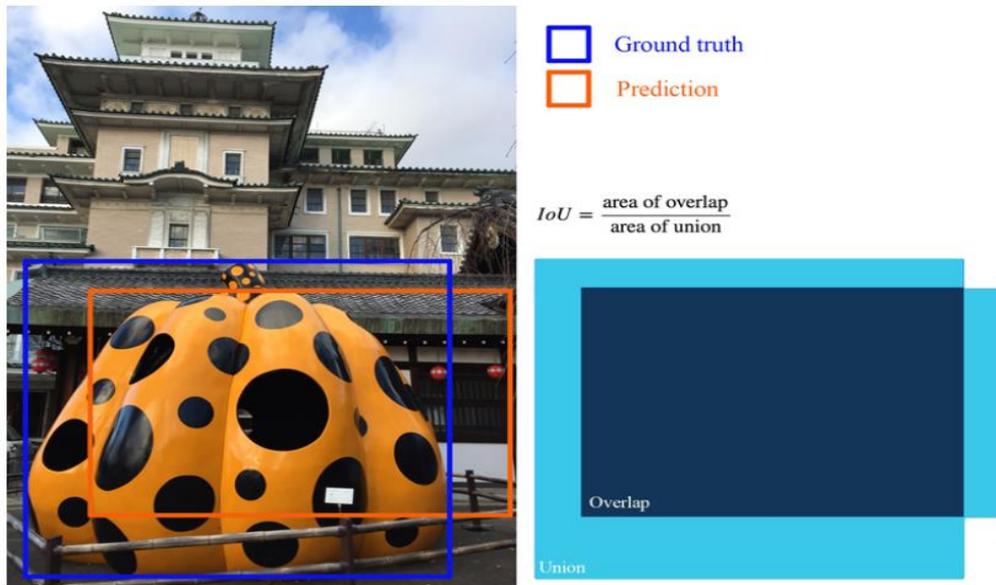


Figura 3.7. Ejemplo de intersección sobre la unión [79].

Finalmente, ya se puede pasar a explicar la precisión media, AP (*Average Precision*). Para ello, se pondrá un ejemplo simplificado para el cálculo de la precisión media en Tabla 3.1. En este ejemplo, el conjunto de datos contiene 5 manzanas únicamente. Se recogen todas las predicciones hechas para manzanas en todas las imágenes y se ordenan en un orden descendente de acuerdo al nivel de confianza predicho. La segunda columna indica si la predicción es correcta o no.

Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0	0.4
3	False	0.67	0.4
4	False	0.5	0.4
5	False	0.4	0.4
6	True	0.5	0.6
7	True	0.57	0.8
8	False	0.5	0.8
9	False	0.44	0.8
10	True	0.5	1.0

Tabla 3.1. Distintos valores de predicciones para cálculo de AP [79].

Se muestra a continuación como se calculan la precisión y la exhaustividad para la tercera columna:

La precisión es la proporción de positivos verdaderos $=2/3=0.67$

La exhaustividad es la proporción de positivos verdaderos de todos los posibles positivos $=2/5=0.4$

Los valores de la exhaustividad incrementan a medida que descendemos en la tabla. Sin embargo, la precisión sigue un comportamiento zigzagueante, descendiendo con falsos positivos y ascendiendo con verdaderos positivos.

Ahora se muestra en Figura 3.8 la precisión junto con la exhaustividad para ver este comportamiento:

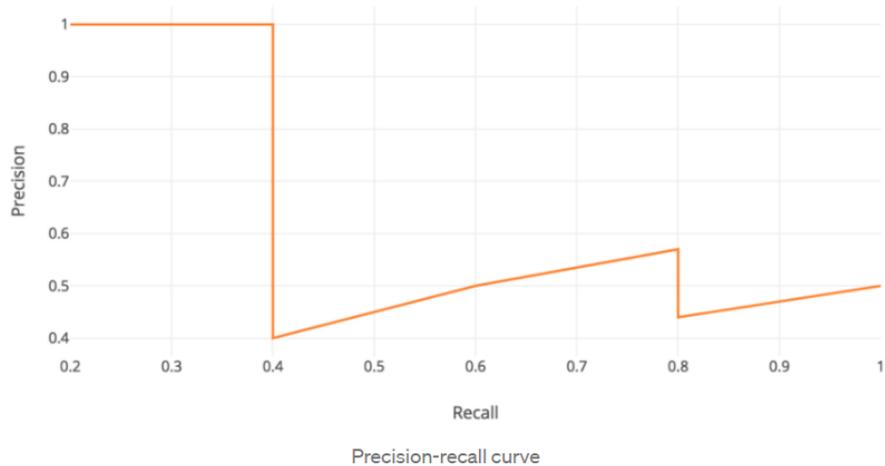


Figura 3.8. Curva precisión – exhaustividad [79].

La definición general de la precisión media AP, Ecuación 3.3, es el área bajo la curva precisión-exhaustividad mostrada arriba:

$$AP = \int_0^1 p(r)dr \quad (3.3)$$

mAP es la media de AP. En algunos contextos, se computan las precisiones medias para cada clase y se éstas se promedian. Pero en otros contextos, significan lo mismo. En el caso del conjunto de datos de COCO, no hay diferencia entre AP y mAP [79].

Para terminar, se muestra en Figura 3.9, a modo de ejemplo, una gráfica de mAP mostrada en *Tensorboard*; donde en el eje de ordenadas se muestra el valor de mAP, y en el eje de abcisas el número de pasos o *steps* que ha sido entrenado el modelo.

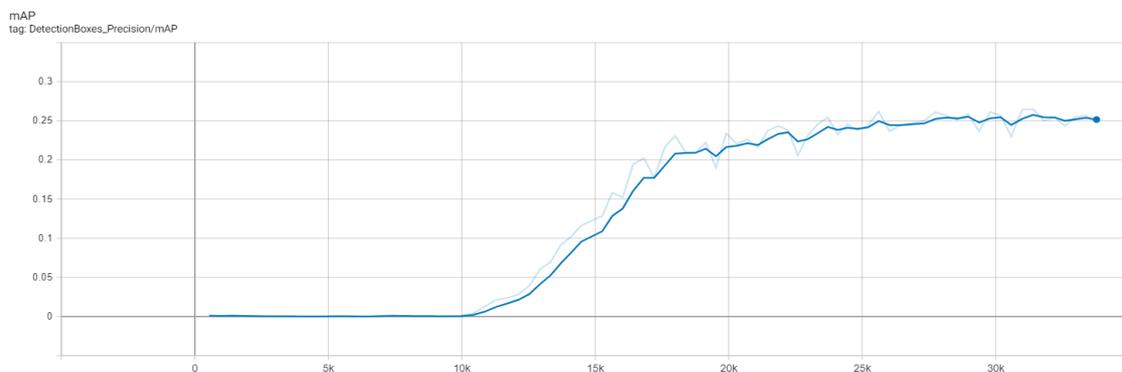


Figura 3.9. Ejemplo de gráfica de precisión media en Tensorboard.

Velocidad

Por otro lado, se encuentra el criterio de velocidad que indica el tiempo que tarda el modelo en realizar una predicción. Este valor se obtendrá de cada entrenamiento; donde se hará el promedio del tiempo de las predicciones que cada modelo precisa.

Finalmente, antes de pasar a la sección de resultados y a la selección de los distintos modelos, mencionar que para la aplicación que se va a llevar a cabo, seguimiento solar, se necesita precisión y velocidad; por lo que estos serán los criterios en los que se basarán los resultados para escoger modelos.

4. RESULTADOS Y DISCUSIÓN

4.1 Elección de modelos

A continuación se presentan los distintos posibles modelos que se utilizarán para entrenar nuestra red neuronal a través del aprendizaje por transferencia. Antes de pasar a verlos y a realizar una selección, se muestran algunas definiciones para el entendimiento de dichos modelos.

Como se indicó anteriormente, la arquitectura para la detección de objetos que se iba a usar es SSD, *Single Shot Detector*, se repasa brevemente:

SSD es un detector de un solo disparo; es decir la red neuronal no tiene regiones de interés, como las tiene la familia de métodos de Faster R-CNN, y predice las cajas delimitadoras y las clases directamente desde los mapas de características en una sola pasada [72].

Además, ya se mencionó que el aprendizaje por transferencia es una técnica de aprendizaje automático en la cual se utiliza el conocimiento adquirido de una red neuronal ya entrenada con un conjunto de datos extenso (que en nuestro caso será el conjunto de datos *COCO*, el cual tiene más de 300.000 imágenes) [78] sobre una nueva red, con un número de datos de entrenamiento y validación significativamente más pequeños.

Por tanto, se necesita encontrar un modelo que sea lo más rápido (tarde pocos ms en realizar una predicción); y lo más exacto posible (tenga un mAP elevado). Se presentan a continuación, en Tabla 4.1, los distintos modelos y sus valores para ambos criterios, pero añadir que estos valores cambiarán cuando reentrenemos nuestra red neuronal:

Nombre del modelo	ms	mAP
ssd_mobilenet_v1_coco	30	21
ssd_mobilenet_v1_0.75_depth_coco	26	18
ssd_mobilenet_v1_quantized_coco	29	18
ssd_mobilenet_v1_0.75_depth_quantized_coco	29	16
ssd_mobilenet_v1_ppn_coco	26	20
ssd_mobilenet_v1_fpn_coco	56	32
ssd_resnet_50_fpn_coco	76	35
ssd_mobilenet_v2_coco	31	22
ssd_mobilenet_v2_quantized_coco	29	22
ssdlite_mobilenet_v2_coco	27	22
ssd_inception_v2_coco	42	24
faster_rcnn_inception_v2_coco	58	28
faster_rcnn_resnet50_coco	89	30
faster_rcnn_resnet50_lowproposals_coco	64	
rfcn_resnet101_coco	92	30
faster_rcnn_resnet101_coco	106	32
faster_rcnn_resnet101_lowproposals_coco	82	
faster_rcnn_inception_resnet_v2_atrous_coco	620	37
faster_rcnn_inception_resnet_v2_atrous_lowproposals_coco	241	
faster_rcnn_nas	1833	43
faster_rcnn_nas_lowproposals_coco	540	

Tabla 4.1. Distintos modelos para utilizar en aprendizaje por transferencia [80].

Como se ve, los métodos que se encuentran son SSD y Faster R-CNN. Por regla general, SSD es más rápido; mientras que Faster R-CNN es un poco más preciso. Por tanto, los modelos Faster R-CNN se descartan, pues se puede conseguir casi la misma precisión con mayor velocidad.

En nuestro caso, hay que tener en cuenta que el número de clases a predecir son 4 y en un entorno no muy cambiante; por lo que la complejidad será menor que, por ejemplo, un modelo de detección de objetos para un vehículo autónomo; donde ya entran cuestiones más complicadas en el que el modelo tendría que tomar decisiones complejas, incluso hasta de índole moral, a una gran velocidad [81].

Se empieza comparando los modelos que se muestran en Tabla 4.2:

Nombre del modelo	ms	mAP
ssd_mobilenet_v1_coco	30	21
ssd_mobilenet_v1_0.75_depth_coco	26	18
ssd_mobilenet_v1_quantized_coco	29	18
ssd_mobilenet_v1_0.75_depth_quantized_coco	29	16
ssd_mobilenet_v2_coco	31	22
ssd_mobilenet_v2_quantized_coco	29	22
ssdlite_mobilenet_v2_coco	27	22

Tabla 4.2. Modelos SSD con arquitectura MobileNet [80].

La primera diferencia que se observa es la versión de *MobileNet*; y a pesar de que se puede ver que la relación velocidad-precisión es mejor en los modelos con la versión 2, esto no siempre es así. *MobileNet* son una serie de modelos diseñados por Google, pero que fueron diseñados para poder ser usados en dispositivos de baja potencia (móviles) [82]. Sin embargo, cuando la versión 2, supuestamente más eficiente, se usa en una GPU, el desempeño disminuye considerablemente. Realmente, baja el desempeño de ambas versiones, debido a que estas arquitecturas tienen un tipo de convolución que no son soportadas por GPU; y como la versión 2 tiene 17 de estas convoluciones, mientras que la versión 1, 13; el desempeño de la versión 2 sufre una degradación mayor. Además, esta degradación también es debida a que el número de canales por los que pasa este tipo de convoluciones es mayor en la versión 2 que en la 1 [83].

Por otro lado, se aprecia que hay un tipo de modelo que se especifican como cuantificados, *quantized*. Esto significa que el modelo usa pesos cuantificados para realizar la predicción; es decir, que el modelo reduce la precisión de los números utilizados para representar los parámetros de un modelo, que por defecto son números de coma flotante de 32 bits. Esto da como resultado un tamaño de modelo más pequeño y un cálculo más rápido [84].

Analizando, la otra diferencia que se observa es que algunos están caracterizados por tener asignados un 0.75 de multiplicador de profundidad, *depth multiplier*. Este parámetro que ya se vio, es un multiplicador flotante para la profundidad de las operaciones de convolución, es decir, para el número de canales.

Por tanto, al ser este último un parámetro modificable en cualquiera de los modelos, se opta por elegir el modelo que no tiene 0.75 de multiplicador de profundidad, el cual viene con un valor de 1 predeterminado; y en la optimización se podrá modificar este valor. También, observando entre los dos modelos restantes el balance de precisión-velocidad, se escoge el modelo no cuantificado, *ssd_mobilenet_v1_coco*.

Seguidamente, en Tabla 4.3 se comparan los 2 modelos restantes con método SSD:

Nombre del modelo	ms	mAP
ssd_resnet_50_fpn_coco	76	35
ssd_inception_v2_coco	42	24

Tabla 4.3. Modelos SSD con diferentes arquitecturas [80].

Se ve que el que se basa en la arquitectura de ResNet, tiene un tiempo bastante elevado en comparación con el resto de modelos SSD, por lo que se descarta, ya que en estos modelos de una sola pasada se busca que brinden rapidez. Por lo que se selecciona la arquitectura *Inception*, desarrollada por Google, *ssd_inception_v2_coco*.

Por tanto los modelos seleccionados se muestran en Tabla 4.4:

Nombre del modelo	ms	mAP
ssd_mobilenet_v1_coco	30	21
ssd_inception_v2_coco	42	24

Tabla 4.4. Modelos seleccionados para entrenamiento [80].

4.2 Entrenamiento de los modelos

Una vez escogidos los modelos a entrenar y definidos los parámetros a modificar, ya se está preparado para realizar los entrenamientos.

A continuación, se van a mostrar los resultados obtenidos para los diferentes modelos, donde se mostrarán los resultados tanto gráficos como numéricos; y se hará una comparación entre los resultados que se vayan obteniendo, para ir en búsqueda de la máxima optimización posible del modelo.

Como se mencionó anteriormente, se sigue la técnica de sintonización manual de hiperparámetros, observándose cómo afectan ciertos parámetros al resultado final. Principalmente, se cambiarán los parámetros que afectan al tamaño de la imagen (*image_size*); los que determinan la profundidad de la red (*min_depth* y *depth_multiplier*); se utilizará también la técnica *dropout*; el ratio de aprendizaje (*learning_rate*); y con la posibilidad del aumento de datos (*data_augmentation*)

Se adjuntarán las gráficas de precisión media y de las pérdidas de validación, o *loss*, (gráfica azul) y de entrenamiento, o *loss1*, (gráfica naranja). De forma general, el eje de ordenadas corresponde al valor de la métrica estudiada; y el eje de abscisas, al número de pasos que se ha entrenado la red.

Se empezarán evaluando los entrenamientos del modelo *ssd_inception_v2_coco*, y a continuación, los de *ssd_mobilenet_v1_coco*.

4.2.1 Entrenamientos *ssd inception v2 coco*

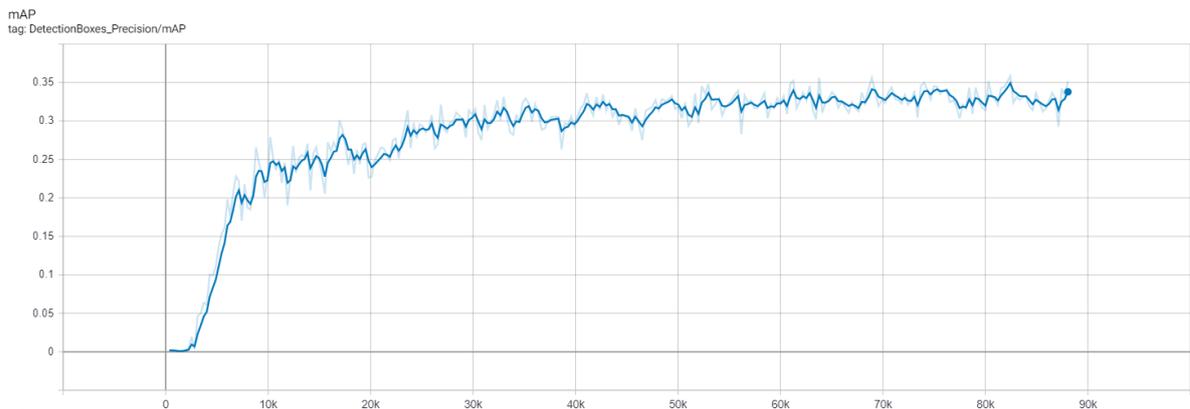
Entrenamiento n° 1:

SSD_inception		Modelo 1
Image_size		300 x 300
dropout		false
min_depth		16
depth_multiplier		1
optimizer		rms
learning_rate		
decay_steps		
data_augmentation	horizontal_flip	X
	random_crop	X
	distort_color	
	adjust_brightness	
	adjust_contrast	
	rgb_to_gray	
adjust_saturation		
shuffle		false
steps (k)		50
batch_size		24

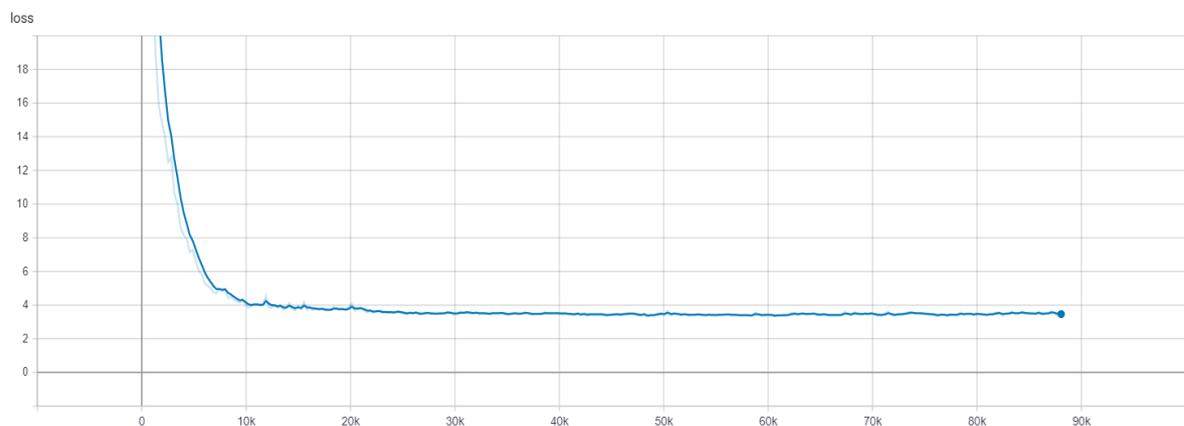
Este primer entrenamiento se lanzó con la configuración que venía predeterminada. En general, los primeros modelos que se mostrarán a continuación tienen el propósito de servir como guía de cara a los futuros modelos. Con esto se quiere decir, que no hay un valor predeterminado a obtener, sino que estos valores los marcarán los primeros entrenamientos, y de ahí se intentará optimizar al máximo los resultados obtenidos.

Tabla 4.5. Configuración modelo 1 *Inception*

- mAP:



- Pérdida de evaluación:



- Pérdida de entrenamiento:



De las gráficas anteriores se puede deducir que en cuanto al mAP, tiene un crecimiento muy rápido hasta el paso 30.000, a partir del cual durante los casi 60.000 pasos siguientes, la precisión tuvo un crecimiento mínimo. En el paso 50.000 tiene una precisión de 0.3212, mientras que en el paso final del entrenamiento, el 88.000, 0.3352. Por lo tanto, se afirma que este algoritmo prácticamente ha alcanzado su precisión máxima.

En cuanto a las pérdidas, se observa que la pérdida de validación sufre un decrecimiento casi exponencial hasta la etapa 30.000, donde el valor de ésta se estabiliza durante el resto del entrenamiento. Este decrecimiento se produce de manera muy drástica para luego estabilizarse también muy rápidamente. Lo que se busca es una caída de la pérdida más gradual, ya que esta pendiente tan inclinada del descenso es indicativa de un ratio de aprendizaje muy elevado. Por parte de la pérdida de entrenamiento, se observa que no para de decrecer prácticamente, hasta alcanzar un valor de 2.95; mientras que la pérdida de validación se estabiliza con un valor de 3.9, por lo que se puede decir que existe *overfitting*.

Para combatir este *overfitting*, una medida puede ser activar la técnica *dropout*, o añadir más datos mediante el parámetro *data_augmentation*. De momento, lo que se pretende es observar el comportamiento del modelo, así que se añadirán opciones de aumento de datos, pero la técnica *dropout* se dejará para más tarde. Con los siguientes modelos, se pretende observar el comportamiento del modelo con las variaciones de la profundidad de la red y las opciones del aumento de datos. El tiempo de entrenamiento ha sido de 1 día, 23 horas, 43 minutos y 44 segundos.

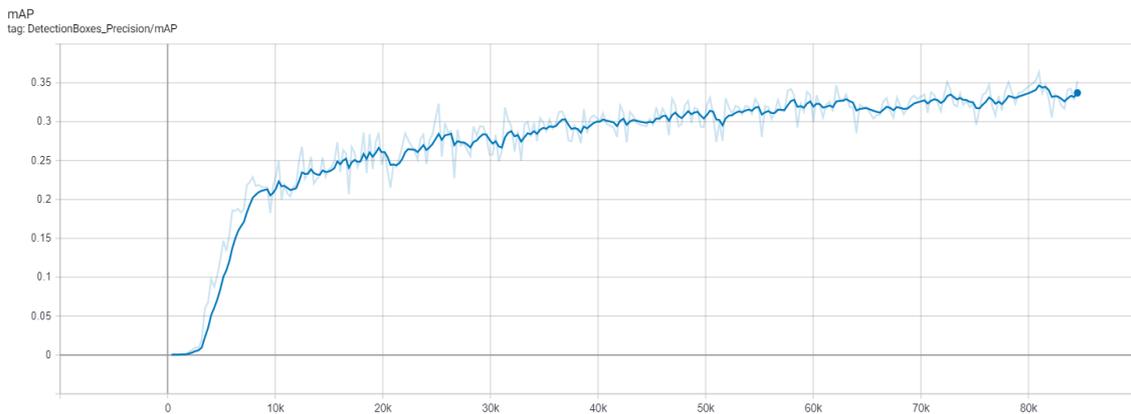
Entrenamiento n°2:

SSD_inception		Modelo 2
image_size		300 x 300
dropout		false
min_depth		8
depth_multiplier		1
optimizer		rms
learning_rate		
decay_steps		
data_augmentation	horizontal_flip	X
	random_crop	X
	distort_color	
	adjust_brightness	
	adjust_contrast	
	rgb_to_gray	X
adjust_saturation		
shuffle		false
steps (k)		85
batch_size		24

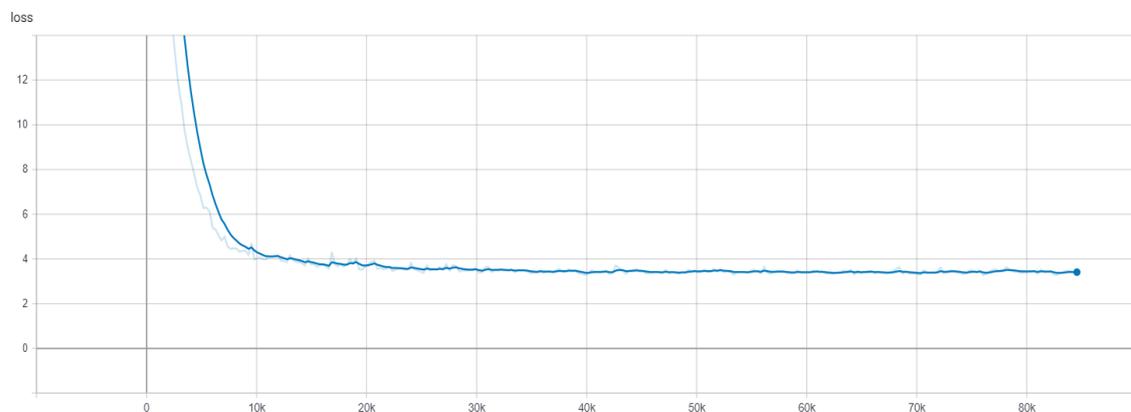
En este modelo, se fija el parámetro *min_depth* a 8, la mitad del modelo anterior, y se recuerda que este parámetro fija la profundidad mínima de cada una de las salidas de los mapas de características producidas por el extractor de características. Al hacer la red menos profunda, ésta debería tener una mayor velocidad, pero seguramente sea a costa de una menor precisión. También se incrementa el volumen de datos con la opción de añadir unas fotos en blanco y negro. Se pasa a analizar las gráficas:

Tabla 4.6. Configuración modelo 2 Inception

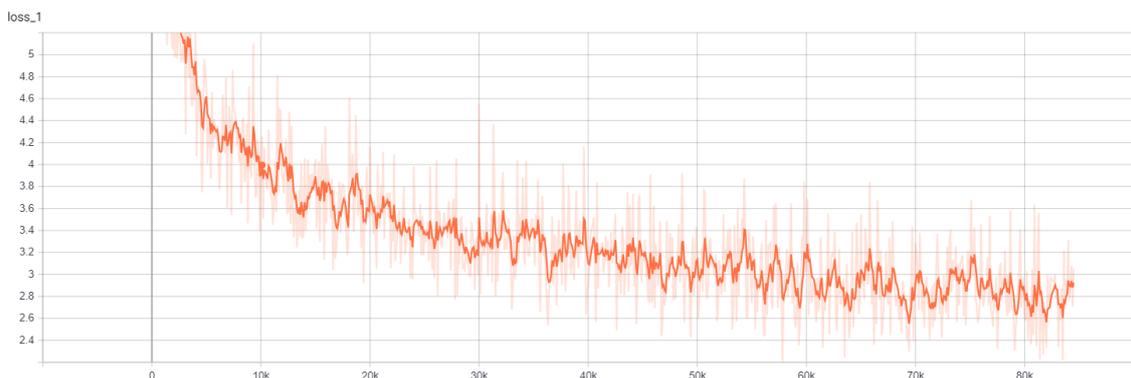
- mAP:



- Pérdida de evaluación:



- Pérdida de entrenamiento:



Este entrenamiento también ha sido de larga duración, 84.000 pasos. La precisión tiene un ascenso más lento que el modelo anterior, ya que en el paso 50.000 tiene una precisión de 0.3084, pero el ascenso es gradual desde la el paso 8.000 hasta el final del entrenamiento. Durante los primeros pasos, se aprecia que la precisión del modelo no mejora nada; esto es debido a que la red tarda un tiempo en empezar a hacer predicciones adecuadas, y conforme van pasando las épocas, estos se van actualizando por prueba y error, hasta que se consiguen unos pesos coherentes y ahí es cuando empieza a realizar buenas predicciones.

En lo que respecta a las pérdidas, se produce un descenso brusco igualmente en la pérdida de evaluación, por lo que el disminuir la profundidad no ha afectado mucho al valor predeterminado del ratio de aprendizaje, que es un ratio constante de 0.004. El añadir una mayor cantidad de imágenes, en este caso blanco y negro, tampoco ha ayudado a disminuir el *overfitting* observado en el modelo anterior, ya que la pérdida de evaluación está en torno a los 3.5; mientras que la pérdida de entrenamiento, 2.8.

El tiempo de entrenamiento ha sido de 1 día, 23 horas, 52 minutos y 42 segundos.

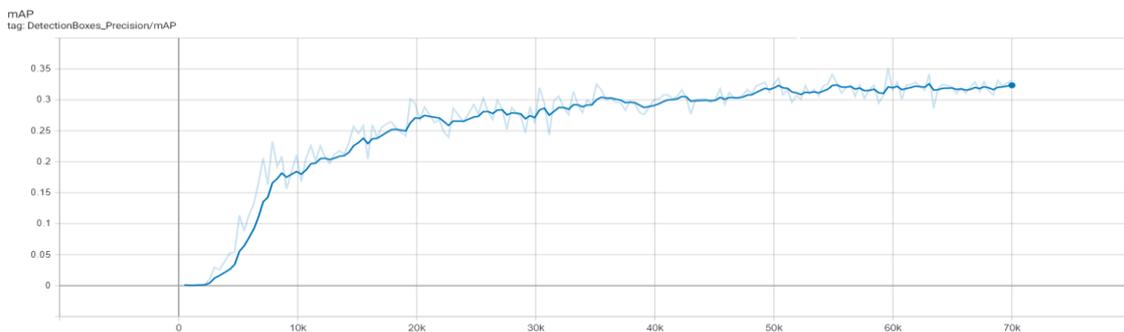
Entrenamiento n° 3:

SSD_inception		Modelo 3
Image_size		300 x 300
dropout		true
min_depth		16
depth_multiplier		0,75
optimizer		rms
learning_rate		
decay_steps		
data_augmentation	horizontal_flip	
	random_crop	X
	distort_color	
	adjust_brightness	
	adjust_contrast	
	rgb_to_gray	X
	adjust_saturation	
shuffle		false
steps (k)		70
batch_size		24

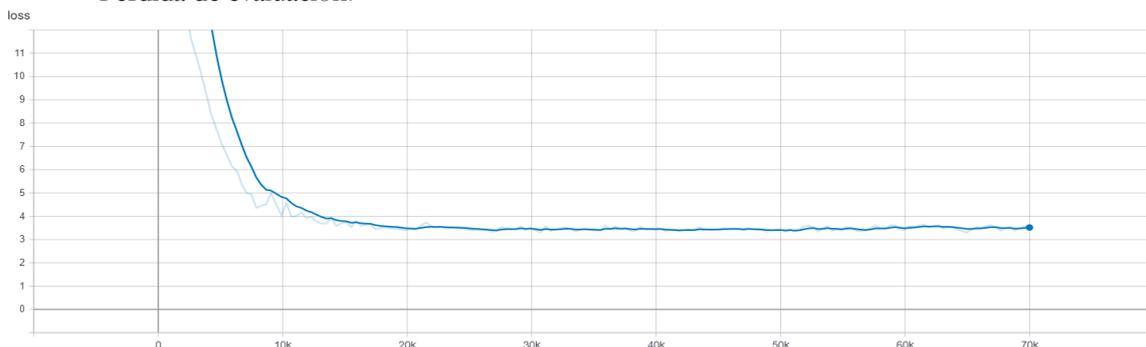
Se sigue con el estudio de la profundidad, en este caso, modificamos el parámetro *depth_multiplier* que se usa para reducir el número de canales en cada capa. De la misma manera, se dejará entrenando durante una gran cantidad de pasos para ver las similitudes con los modelos anteriores. Se pasa a observar las gráficas:

Tabla 4.7. Configuración modelo 3 *Inception*

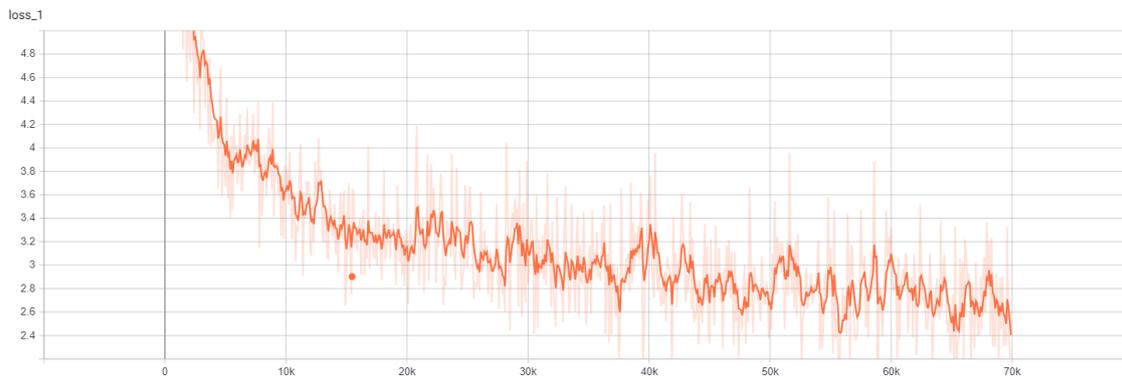
- mAP:



- Pérdida de evaluación:



- Pérdida de entrenamiento:



En este modelo también se da una mejora gradual de la precisión sobre el paso 8.000, después del incremento súbito inicial. Señalar la presencia de mayor ruido, en comparación con modelos anteriores, cuando la curva está a punto de estabilizarse en la pérdida de validación. Además, esta pérdida alcanza el valor en el cual se estanca antes que los anteriores modelos, en el paso 20.000.

En el caso de la pérdida de entrenamiento, alcanza un valor mínimo hasta ahora, siendo de 2.6 aproximadamente; mientras que la otra pérdida se ha estabilizado en el valor 3.6; por lo que se ha producido un incremento de la desigualdad de ambas pérdidas.

El tiempo de entrenamiento ha sido de 2 días, 4 horas, 23 minutos y 22 segundos.

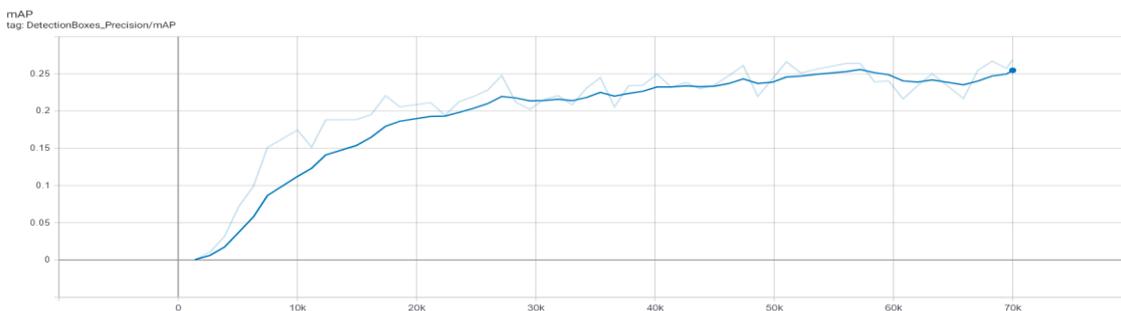
Entrenamiento n° 4:

SSD_inception		Modelo 4
image_size		200 x 200
dropout		false
min_depth		8
depth_multiplier		0,5
optimizer		rms
learning_rate		
decay_steps		
data_augmentation	horizontal_flip	
	random_crop	X
	distort_color	
	adjust_brightness	
	adjust_contrast	
	rgb_to_gray	X
adjust_saturation		
shuffle		false
steps (k)		70
batch_size		24

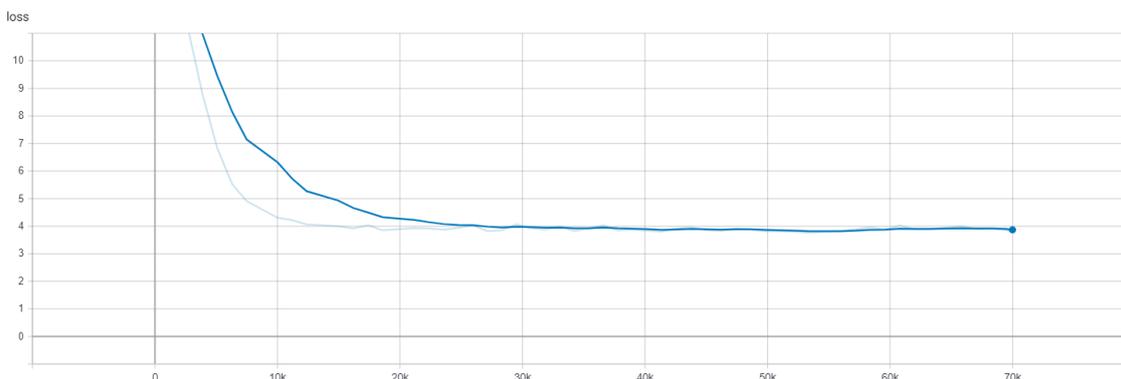
En este entrenamiento, se han seleccionado valores mínimos para la profundidad, al igual que el tamaño de la imagen se ha reducido. Se puede esperar una precisión menor que lo resultado hasta ahora, acompañado de un incremento de velocidad en las predicciones.

Tabla 4.8. Configuración modelo 4 *Inception*

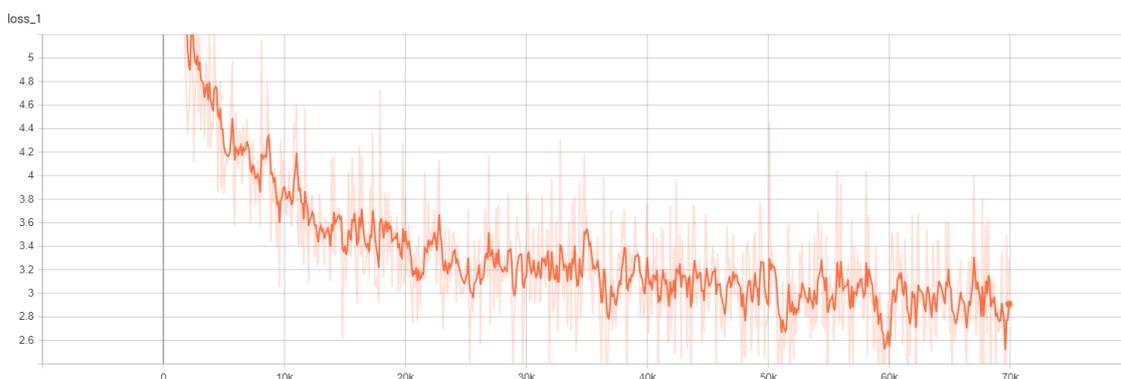
- mAP:



- Pérdida de evaluación:



- Pérdida de entrenamiento:



Este modelo con valores mínimos de profundidad ha sufrido una degradación tanto de precisión como de pérdida de validación, la cual se estabiliza en valores cercanos a 4. La mejora de precisión es muy lenta, y a pesar de que el tiempo de entrenamiento ha sido de 1 día, 5 horas, 21 minutos y 46 segundos; ésta solo ha alcanzado un valor de 0.2547, prácticamente casi 0.1 por debajo de los modelos anteriores.

Antes de seguir lanzando entrenamientos, se puede ir ya entendiendo algunos detalles del comportamiento de este modelo con lo ya visto. Se puede decir que, sobre las etapas 10 – 30.000 es donde el modelo sufre un mayor cambio que es decisivo para la evolución del modelo, con lo que a partir de ahora, se evaluará hasta 30.000 pasos para evitar los entrenamientos de dos días de duración.

De este modo, se evaluará el modelo en dos puntos: a los 20.000 pasos y a los 30.000 pasos. En el caso de que el modelo parezca prometedor, se retomará el entreno hasta los 40-45.000 pasos para ver su evolución. Además, si el modelo presenta unos resultados mediocres, se optará por la técnica de la parada anticipada y se dejará de entrenar el modelo.

A continuación, se presenta en Tabla 4.9 los valores numéricos de los primeros cuatro entrenamientos; y se añadirá esta metodología para el resto de los entrenamientos.

SSD_inception		Modelo 1	Modelo 2	Modelo 3	Modelo 4
mAP	20.000	0,24	0,2591	0,2703	0,1928
	30.000	0,3042	0,2738	0,2837	0,2142
	último paso	0,3212	0,3084	0,3193	0,2547
pérdida_evaluación	20.000	3,906	3,741	3,475	4,224
	30.000	3,504	3,519	3,448	3,967
	último paso	3,546	3,436	3,642	3,869
pérdida_entrenamiento	20.000	3,51	3,645	3,032	3,433
	30.000	3,324	3,311	3,195	3,32
	último paso	3,202	2,827	2,404	2,908

Tabla 4.9. Valores de las métricas de los primeros cuatro modelos *Inception*

Los resultados que aparecen en la tabla, son valores numéricos suavizados, es decir, que no corresponden al valor exacto de ese paso; sino que al haber presencia de ruido, este se reduce y los valores que se dan, son por los que pasa esta curva suavizada.

En primer lugar comentar que para que estén en igualdad de condiciones, se ha establecido 50.000 como último paso. De esta manera, se ve que el modelo que un mAP más alto obtiene en su último paso es también el modelo que más ajustadas se encuentran sus pérdidas, este es el modelo 1. También se observa un buen comportamiento del modelo 3, aunque con una presencia mayor de *overfitting*.

Todas las funciones de las pérdidas de evaluación han presentado curvas con pendientes muy pronunciadas, con forma de codo a 90°; esto es causa de un ratio de aprendizaje muy elevado, ya que se busca una pendiente más suave, para evitar el posible inconveniente de que el gradiente descendiente estocástico se quede en un mínimo local y la pérdida no sea la óptima. Se probarán distintos ratios de aprendizaje más adelante.

Para continuar, se tomará de referencia la configuración del modelo 1 a la hora de seguir lanzando entrenamientos. En los próximos entrenamientos se va a estudiar como las distintas opciones de aumento de datos interfieren en el resultado final del modelo.

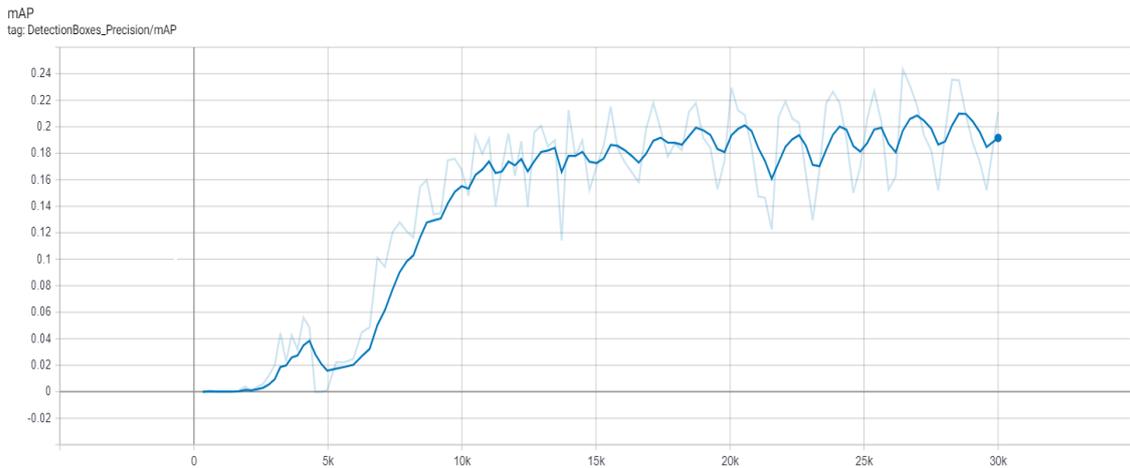
Entrenamiento n° 5:

SSD_inception	Modelo 5	
image_size	300 x 300	
dropout	false	
min_depth	16	
depth_multiplier	1	
optimizer	adam	
learning_rate		
decay_steps		
data_augmentation	horizontal_flip	
	random_crop	
	distort_color	
	adjust_brightness	X
	adjust_contrast	
	rgb_to_gray	
adjust_saturation	X	
shuffle	false	
steps (k)	30	
batch_size	24	

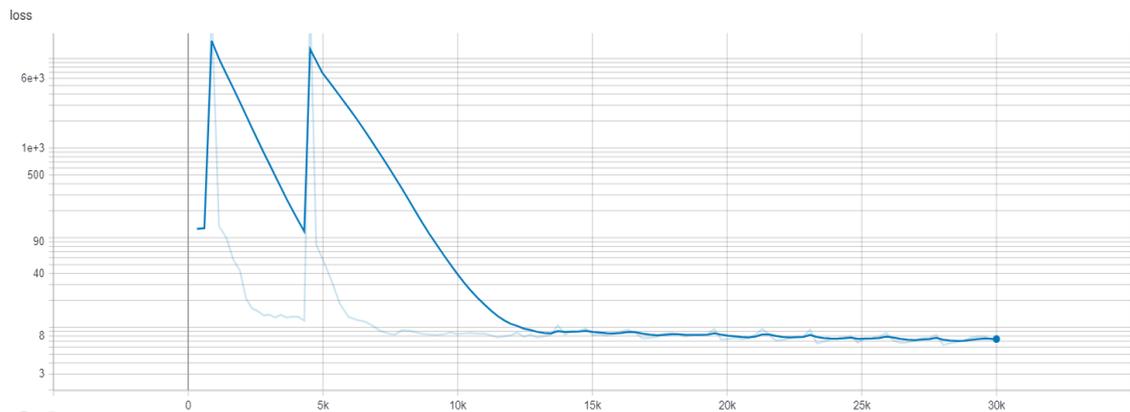
Tabla 4.10. Configuración modelo 5 *Inception*

En este entrenamiento se quiere testear otro tipo de optimizador, llamado Adam. Por otro lado, se ajusta la opción de aumento de datos para añadir imágenes con un brillo y saturación con niveles distintos a los originales. Ya se empieza a evaluar hasta donde el modelo deja de mejorar prácticamente, sobre el paso 30.000.

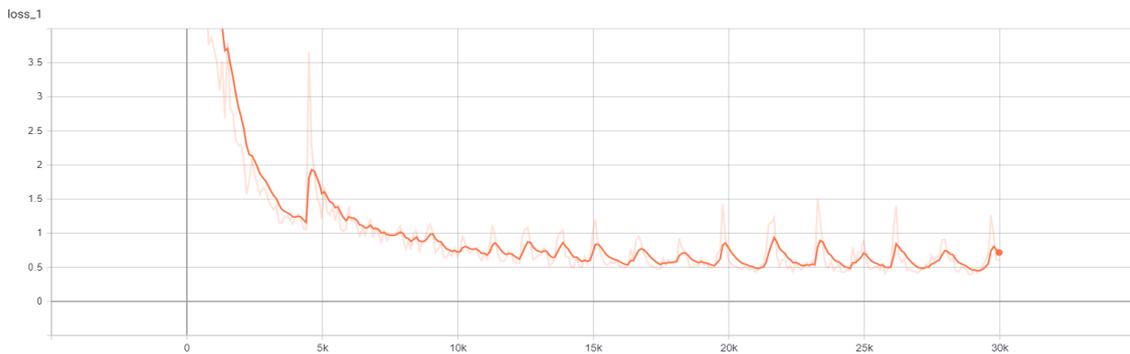
- mAP:



- Pérdidas de evaluación:



- Pérdidas de entrenamiento:



Se observa que en este modelo, conforme avanzan los pasos, hay una mayor presencia de ruido a partir de los 12.000 pasos. Esto puede ser una señal de que el modelo es incapaz de mejorar y que se está contradiciendo a cada rato, consecuencia del gran *overfitting* que se aprecia; ya que el conjunto de entrenamiento consigue una pérdida inferior a 1, mientras que el conjunto de evaluación tiene una pérdida de 7.4.

Además, este modelo tardó en entrenarse 5 días, 8 horas, 28 minutos y 32 segundos.

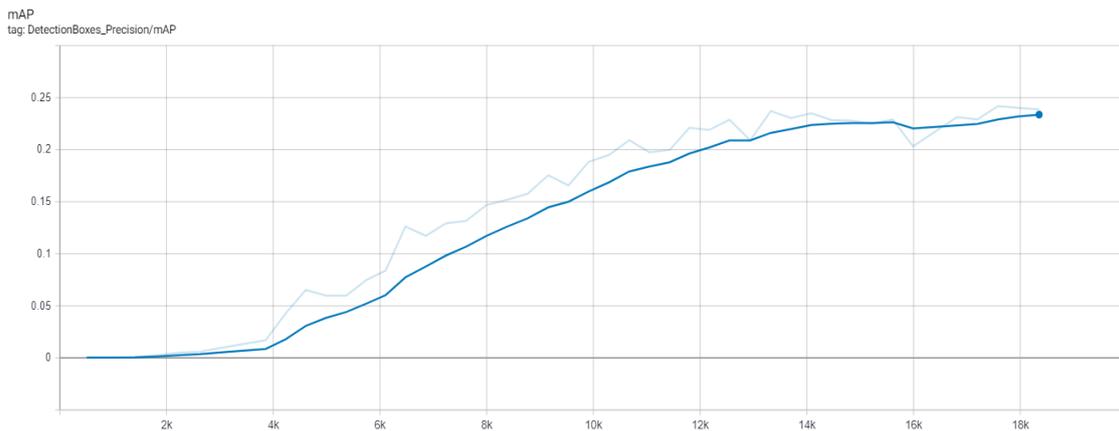
Entrenamiento n°6:

SSD_inception		Modelo 6
image_size		300 x 300
dropout		true
min_depth		16
depth_multiplier		0,75
optimizer		rms
learning_rate		
decay_steps		
data_augmentation	horizontal_flip	
	random_crop	
	distort_color	
	adjust_brightness	X
	adjust_contrast	
	rgb_to_gray	X
adjust_saturation		
shuffle		false
steps (k)		18
batch_size		20

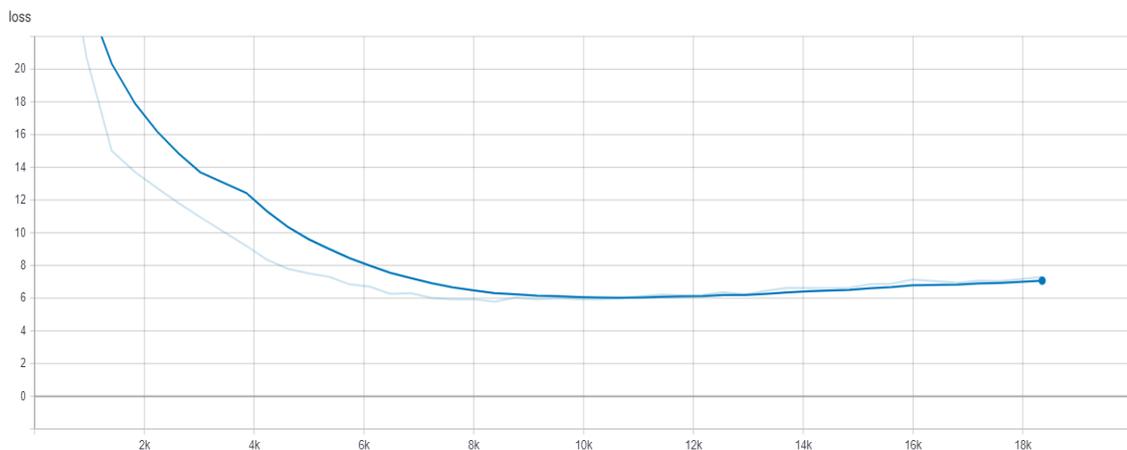
Este modelo se basa en la configuración del entrenamiento n°3, que brindó un buen resultado; con la diferencia de que en la opción del aumento de datos, se seleccionan el cambio de luminosidad y pasar a blanco y negro, para añadir nuevas imágenes. También se reduce el número del *batch*, lo que hace que sea menos costoso computacionalmente. Se activa el *dropout*, para intentar minimizar la diferencia de pérdidas que apareció en el modelo anterior.

Tabla 4.11. Configuración modelo 6 *Inception*

- mAP



- Pérdida de evaluación:



- Pérdida de entrenamiento:



En este caso, la técnica *dropout* no ha dado buenos resultados; y se puede señalar una peculiaridad, que la pérdida de validación llega un punto, sobre el paso 11.000, en el que esta empieza a incrementar, lo que provoca *overfitting*, y es por eso que se paró el entreno. Además, se observa que la precisión sigue aumentando, mientras la pérdida de evaluación también aumenta.

Esto se debe a que el algoritmo, aparte de aprender patrones que solo son beneficiosos para el conjunto de entrenamiento, por lo que esta pérdida sigue decreciendo; a su misma vez algunos patrones válidos para el conjunto de evaluación se refuerzan, aunque otras muchas otras predicciones sean fallidas. Es por eso que la función llega ya a un punto que deja de aprender pero se mantiene estable.

En estos dos últimos modelos, lo que distingue a los previamente entrenados es que han conseguido una pérdida de entrenamiento muy baja; en cambio, la pérdida de evaluación ha aumentado. Se siguió entrenando modelos añadiendo imágenes con una saturación, contraste o brillo diferentes a los originales, y se ha llegado a la conclusión de que el algoritmo añade unos cambios de estos tipos muy drásticos; y luego, a la hora de evaluar el set de imágenes no es capaz de reconocer los objetos dado que ha aprendido a reconocerlos con unas condiciones muy diferentes a las expuestas en las imágenes de evaluación. Se presenta en Tabla 4.12 los resultados de algunos entrenamientos de este tipo.

SSD_inception		Modelo 7	Modelo 8	Modelo 9	Modelo 10
Image_size		150 x 150	300 x 300	300 x 300	300 x 300
dropout		true	true	true	true (0,6)
min_depth		8	16	16	16
depth_multiplier		0,75	1	1	1
optimizer		rms	rms	rms	rms
learning_rate					0,004
decay_steps					3500
data_augmentation	horizontal_flip		X		X
	random_crop				
	distort_color				
	adjust_brightness	X			X
	adjust_contrast			X	
	adjust_saturation	X		X	X
shuffle		false	true	true	true
steps (k)		18	25	20	20
batch_size		16	24	24	24
mAP	20.000		0,2409	0,2323	0,21
	30.000				
	último paso	0,1791	0,2297		
pérdida_evaluación	20.000		7,448	7,729	7,5
	30.000				
	último paso	8,581	7,823		
pérdida_entrenamiento	20.000		0,7036	0,6713	0,9
	30.000				
	último paso	0,5582	0,7378		
velocidad (s)					

Tabla 4.12. Modelos con imágenes

Como se ve en la tabla, todos los entrenos presentan un *overfitting* muy grande. Aparte, ningún modelo ha sido satisfactorio en cuanto a precisión, tomando de referencia el modelo 3 que alcanzó una precisión de 0.27 a los 20.000 pasos. El más aproximado es el modelo 8 que obtiene un buen valor 0.24, pero fijándose bien, este valor se degrada cuando el modelo sigue entrenando.

Se puede observar que en el modelo 10, se han modificado dos parámetros. El primero es *dropout*, el cual se ha aumentado la posibilidad de que se eliminen neuronas al azar estableciendo un 0.6 en lugar del 0.8 predeterminado, para intentar reducir así la problemática del *overfitting*, aunque no ha funcionado. Por otro lado, hasta ahora se había entrenado al modelo con un ratio de aprendizaje constante durante todo el entrenamiento, con un valor de 0.004. Ahora, se ha establecido un ratio de aprendizaje decreciente exponencialmente, de modo que cada 3.500 pasos, hay un pequeño escalón que disminuye este ratio. Con esto se pretende conseguir un aprendizaje más efectivo y afinado, ya que al principio del entrenamiento es preciso un ratio de aprendizaje alto, pero a medida que se avanza, el ratio tiene que ser más pequeño para no caer en un mínimo local, y así la red neuronal consigue converger al valor óptimo.

A continuación, se pasa al estudio del ratio de aprendizaje. Para ello, se entrenarán modelos con la misma configuración que los modelos que mejores resultados han dado hasta ahora, que son los modelos 1 y 3.

Entrenamiento n° 11 y 12:

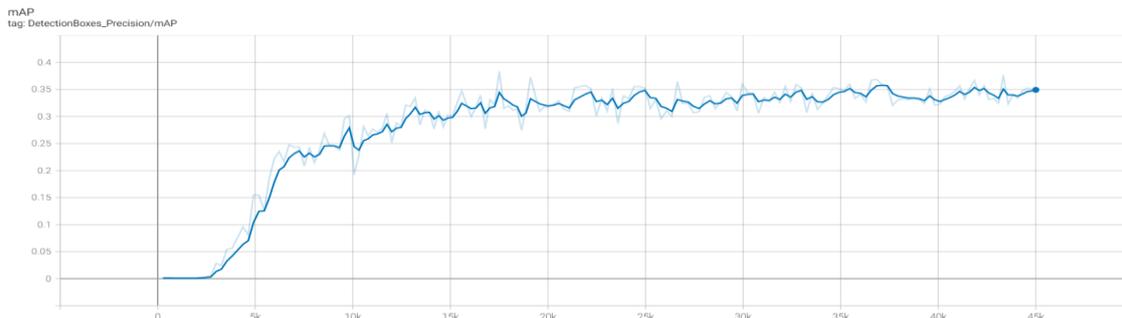
SSD_inception	Modelo 11	Modelo 12
Image_size	300 x 300	400 x 400
dropout	true (0,6)	true (0,6)
min_depth	16	16
depth_multiplier	0,75	0,75
optimizer	rms	rms
learning_rate	0,004	0,004
decay_steps	3500	3500
data_augmentation	horizontal_flip	
	random_crop	X
	distort_color	
	adjust_brightness	
	adjust_contrast	
	rgb_to_gray	X
	adjust_saturation	
shuffle	true	true
steps (k)	45	45
batch_size	24	24

Estos dos modelos parten de la configuración del modelo 3. Las diferencias con este es que se activa la técnica *dropout* con un valor de 0.6, es decir que un 40% de las neuronas de cada capa las eliminará aleatoriamente. Con esto se pretende vencer el *overfitting* que ocurría en el modelo 3. Además, se establece el ratio de aprendizaje decreciente, con un escalón cada 3.500 pasos. La única diferencia entre ambos modelos es el tamaño de la imagen de entrada.

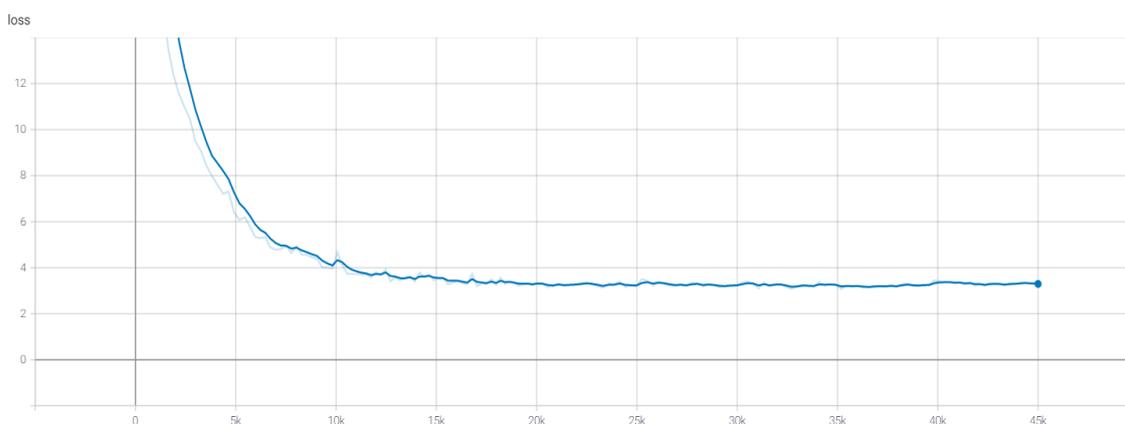
Tabla 4.13. Configuración modelos 11 y 12 *Inception*

Ambos modelos han obtenidos buenos resultados, pero el modelo 12 ha conseguido un mejor desempeño, por lo que solo se mostrarán las gráficas de este.

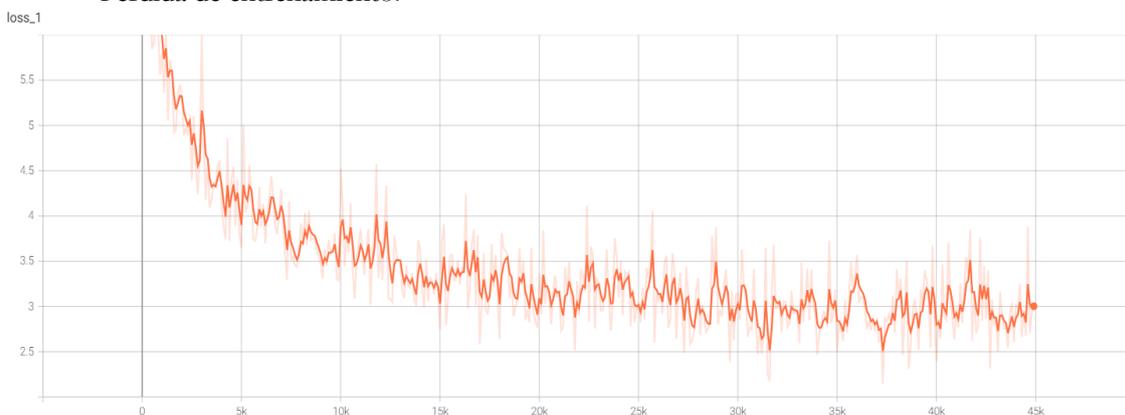
- mAP:



- Pérdida de evaluación:



- Pérdida de entrenamiento:



Ambos modelos fueron entrenados hasta los 45.000 pasos por sus buenos resultados durante las primeras etapas del entreno. Este modelo, cuya imagen de entrada es de 400 x 400, obtiene una precisión prácticamente de 0.35; mientras que ambas pérdidas están muy igualadas en torno a 3; por lo que se puede decir que se ha eliminado el *overfitting* que estaba presente en el modelo de referencia. Además, la curva de la función de la pérdida de evaluación es el ideal de curva buscada, no tiene un descenso muy brusco ni muy lento, lo que significa que para esa configuración, se ha elegido un ratio de aprendizaje adecuado. El entrenamiento fue de 1 día, 7 horas, 57 minutos y 33 segundos.

Para terminar los entrenamientos de este modelo, los siguientes se basarán en la configuración n° 1, que fue una de las que mejores respuestas ha dado, y se modificarán los parámetros en sintonía entre ellos, para ver si pueden optimizar este resultado.

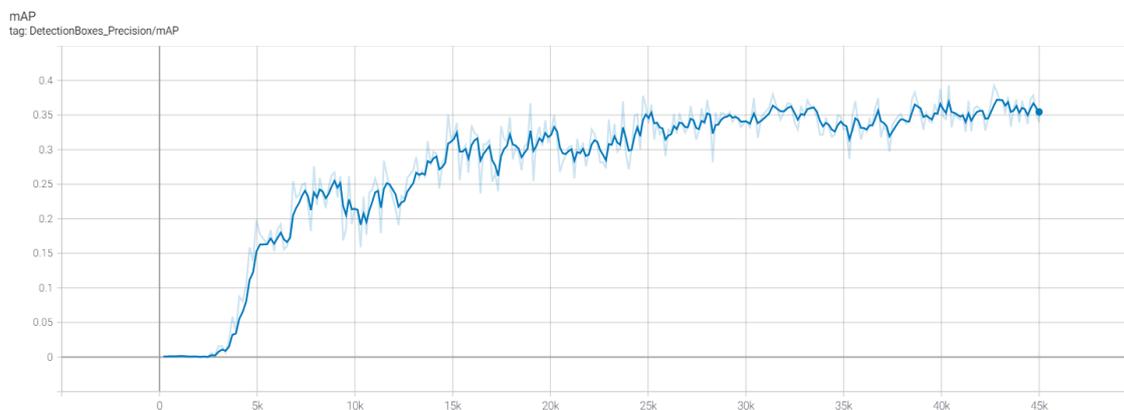
Entrenamiento n° 13:

SSD_inception		Modelo 13
Image_size		400 x 400
dropout		true
min_depth		16
depth_multiplier		1
optimizer		rms
learning_rate		0,01
decay_steps		3500
data_augmentation	horizontal_flip	X
	random_crop	X
	distort_color	
	adjust_brightness	
	adjust_contrast	
	rgb_to_gray	
adjust_saturation		
shuffle		true
steps (k)		45
batch_size		24

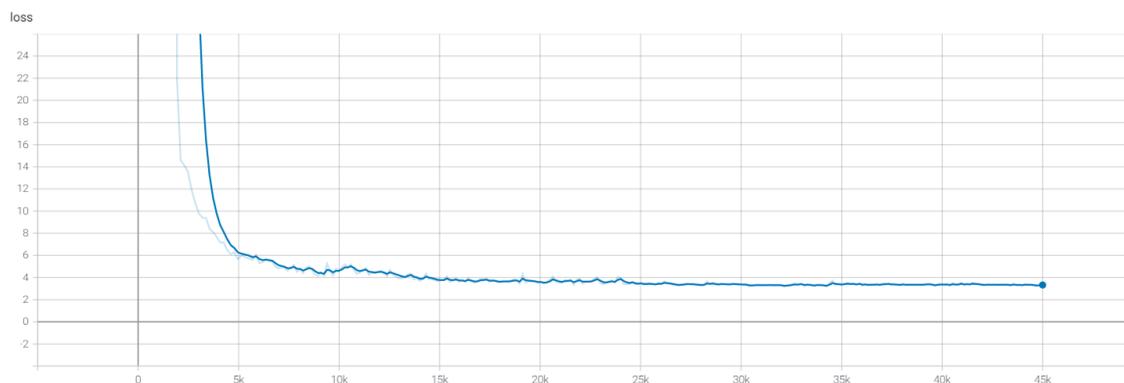
Este modelo se caracteriza por tener una imagen de entrada mayor, al igual que un ratio de aprendizaje mayor que ninguno usado hasta ahora con el fin de estudiar su comportamiento, con un decrecimiento exponencial cada 3.500 pasos.

Tabla 4.14. Configuración modelo 14 *Inception*

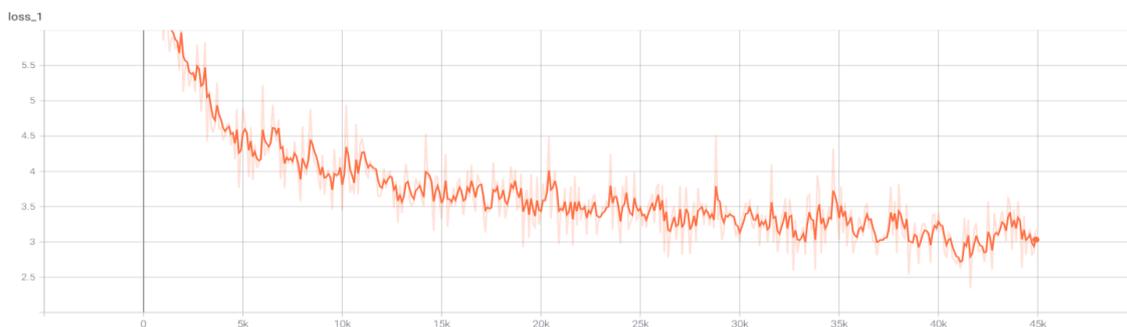
- mAP:



- Pérdida de evaluación:



- Pérdida de entrenamiento:



Se observa que con esta configuración se obtiene unos resultados muy prometedores, es por eso que se decidió seguir entrenando hasta el paso 45.000, y fue todo un acierto, ya que la precisión media obtenida al final del entreno es de 0.3546, la mayor obtenida hasta ahora; mientras que ambas pérdidas toman un valor en torno a 3. A pesar de que el ratio de aprendizaje fijado haya sido muy alto, este modelo no se ha estancado en ningún mínimo local, alcanzando de forma muy rápida la pérdida de evaluación máxima, sobre los 15.000 pasos.

El tiempo empleado para el entrenamiento fue de 1 día, 5 horas, 55 minutos y 22 segundos.

Entrenamiento n° 14:

SSD_inception		Modelo 14
image_size		300 x 300
dropout		true
min_depth		16
depth_multiplier		1
optimizer		rms
learning_rate		0,0005
decay_steps		3500
data_augmentation	horizontal_flip	X
	random_crop	X
	distort_color	
	adjust_brightness	
	adjust_contrast	
	rgb_to_gray	
adjust_saturation		
shuffle		true
steps (k)		20
batch_size		24

Con este modelo se pretende encontrar el ratio de aprendizaje óptimo para este tamaño de imagen de entrada, por lo que se estableció un valor bajo para observar como afectaba a la función de la pérdida de evaluación. Se muestra a continuación esta función:

Tabla 4.15. Configuración modelo 14 *Inception*

- Pérdida de evaluación:



Como se puede observar, es prácticamente una recta, por lo que el entreno se paró, ya que esto indica que el ratio de aprendizaje estipulado es demasiado bajo lo que resulta en que el modelo tarde demasiado tiempo en llegar al valor óptimo.

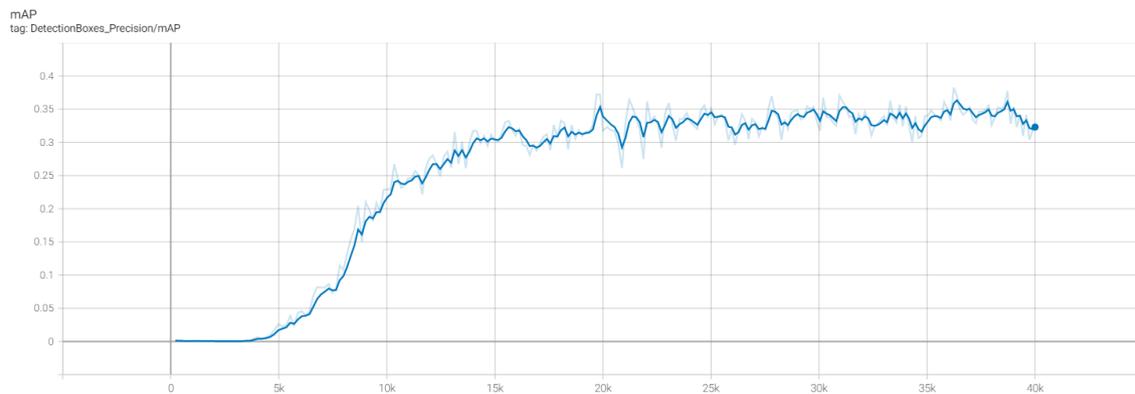
Entrenamientos nº 15 y 16:

SSD_inception		Modelo 15	Modelo 16
Image_size		300 x 300	300 x 300
dropout		true	true
min_depth		16	16
depth_multiplier		1	1
optimizer		rms	rms
learning_rate		0,001	0,001
decay_steps		3500	1000
data_augmentation	horizontal_flip	X	X
	random_crop	X	X
	distort_color		
	adjust_brightness		
	adjust_contrast		
	rgb_to_gray		
shuffle		true	true
steps (k)		40	20
batch_size		24	24

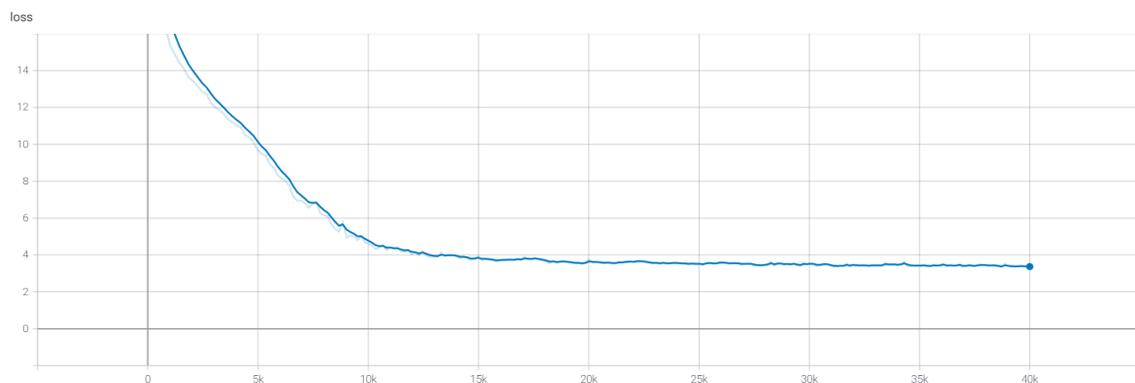
Dado que el valor del ratio de aprendizaje era demasiado bajo en el modelo anterior, en estos dos modelos se ha establecido un valor superior, con valores distintos del paso del escalón en el que se producirá un decrecimiento. Ambos modelos han mostrado un buen desempeño, siendo el modelo 15 el más destacado, por lo que se decidió continuar su entrenamiento hasta los 40.000 pasos para poder evaluar posteriormente los modelos más destacados con un mayor rango de fiabilidad. Se presentan a continuación las gráficas del modelo 15 únicamente:

Tabla 4.16. Configuración modelo 15 y 16 *Inception*

- mAP:



- Pérdida de evaluación:



- Pérdida de entrenamiento:



Respecto a este modelo, se contempla unas gráficas realmente buenas, en las que la precisión media aumenta de forma gradual hasta estabilizarse un poco antes de 0.35; mientras que la pérdida de evaluación tiene una función ideal, ya que el decrecimiento de esta curva no es ni muy precipitado, ni demasiado lento, y se observa que no se llega a estancar, sino que el valor de esta pérdida sigue disminuyendo incluso en los pasos finales del entrenamiento. Además, no hay apenas nada de disparidad entre los valores de ambas pérdidas, por lo que se puede decir que el algoritmo ha aprendido correctamente.

Una vez vistos todos los entrenamientos, se pasa a la comparación de los modelos con los mejores resultados vistos hasta ahora. Estos modelos son el 1, 12, el 13 y el 15. Se presentan a continuación en Tabla 4.17 con sus configuraciones y resultados.

SSD_inception		Modelo 1	Modelo 12	Modelo 13	Modelo 15
Image_size		300 x 300	400 x 400	400 x 400	300 x 300
dropout		false	true (0,6)	true	true
min_depth		16	16	16	16
depth_multiplier		1	0,75	1	1
optimizer		rms	rms	rms	rms
learning_rate			0,004	0,01	0,001
decay_steps			3500	3500	3500
data_augmentation	horizontal_flip	X		X	X
	random_crop	X	X	X	X
	rgb_to_gray		X		
shuffle		false	true	true	true
steps (k)		50	45	45	40
batch_size		24	24	24	24
mAP	20.000	0,24	0,3195	0,3214	0,3381
	30.000	0,3042	0,3388	0,341	0,3468
	último paso	0,3212	0,3492	0,3546	0,3415
pérdida_evaluación	20.000	3,906	3,304	3,596	3,642
	30.000	3,504	3,228	3,378	3,517
	último paso	3,546	3,161	3,326	3,371
pérdida_entrenamiento	20.000	3,51	3,093	3,59	3,519
	30.000	3,324	3,031	3,135	3,255
	último paso	3,202	3,003	3,033	3,432
velocidad (s)		0,5177604	0,41505367	0,242584	0,25613083

Tabla 4.17. Comparación modelos prometedores *Inception*

Como se puede observar, el modelo 1 es claramente el que peor resultados ha obtenido. Esto se debe a que el ratio de aprendizaje de este modelo, uno de los parámetros más importantes, se dejó el predeterminado; mientras que en los demás se ha seguido un estudio de prueba y error, con lo que claramente, el desempeño de estos últimos ha sido mejor; por lo que este modelo se descarta.

Respecto a la precisión media obtenida, los valores de los modelos restantes son todos muy parecidos, sin embargo, los modelos 12 y 13 tienen una progresión considerable a lo largo de las medidas; mientras que el modelo 15, como se apreció en su gráfica, obtiene un valor muy alto rápidamente pero esta mejora se estanca. El modelo 13 es el que obtiene un mejor valor en esta métrica.

En cuanto a las pérdidas, es el modelo 12 el que obtiene un resultado menor, estando estos valores muy cerca de 3. Aun así, en general todas las pérdidas son muy parecidas y se afirma que no hay presencia de *overfitting* en estos modelos.

Respecto a las velocidades, el modelo 12 es el más lento en realizar las predicciones; mientras que el modelo 13 y el 15 tienen velocidades prácticamente iguales, siendo el 13 más rápido por apenas 0.01 segundos. Esta velocidad representa el tiempo que tarda en realizar las predicciones del lote, *batch*, cuyo valor es 24, por lo que este modelo apenas tarda 10 milisegundos de media en realizar las predicciones de una imagen.

Por tanto, en el cómputo global de todas las métricas, es el modelo 13 el que mejor resultados presenta, por lo que se define esta configuración como la óptima para la arquitectura *Inception*.

Se muestra en Tabla 4.18 los entrenamientos realizados y sus resultados.

SSD_inception	Modelo 1	Modelo 2	Modelo 3	Modelo 4	Modelo 5	Modelo 6	Modelo 7	Modelo 8	Modelo 9	Modelo 10	Modelo 11	Modelo 12	Modelo 13	Modelo 14	Modelo 15	Modelo 16
image_size	300 x 300	300 x 300	300 x 300	200 x 200	300 x 300	300 x 300	150 x 150	300 x 300	300 x 300	300 x 300	300 x 300	400 x 400	400 x 400	300 x 300	300 x 300	300 x 300
dropout	false	false	true	false	false	true	true	true	true	true (0,6)	true (0,6)	true (0,6)	true	true	true	true
min_depth	16	8	16	8	16	16	8	16	16	16	16	16	16	16	16	16
depth_multiplier	1	1	0,75	0,5	1	0,75	0,75	1	1	1	0,75	0,75	1	1	1	1
optimizer	rms	rms	rms	rms	adam	rms	rms	rms	rms	rms	rms	rms	rms	rms	rms	rms
learning_rate										0,004	0,004	0,004	0,01	0,0005	0,001	0,001
decay_steps										3500	3500	3500	3500	3500	3500	1000
horizontal_flip	X	X						X		X			X		X	X
random_crop	X	X	X	X							X	X	X	X	X	X
distort_color																
adjust_brightness					X	X	X			X						
adjust_contrast									X							
rgb_to_gray		X	X	X		X	X		X	X	X	X				
adjust_saturation					X			X								
shuffle	false	false	false	false	false	false	false	true	true	true	true	true	true	true	true	true
steps (k)	50	85	70	70	30	18	18	25	20	20	45	45	45	20	40	20
batch_size	24	24	24	24	24	20	16	24	24	24	24	24	24	24	24	24
mAP	20.000	0,24	0,2591	0,2703	0,1928	0,1935		0,2409	0,2323	0,21	0,26	0,3195	0,3214		0,3381	0,312
	30.000	0,3042	0,2738	0,2837	0,2142	0,1916					0,273	0,3388	0,341		0,3468	
	último paso	0,3212	0,3084	0,3193	0,2547	0,1916	0,1791	0,2297			0,288	0,3492	0,3546		0,3415	
pérdida_evaluación	20.000	3,906	3,741	3,475	4,224	8,045		7,448	7,729	7,5	3,55	3,304	3,596		3,642	4,3
	30.000	3,504	3,519	3,448	3,967	7,397					3,66	3,228	3,378		3,517	
	último paso	3,546	3,436	3,642	3,869	7,397	8,581	7,823			3,37	3,161	3,326		3,371	
pérdida_entrenamiento	20.000	3,51	3,645	3,032	3,433	0,7373		0,7036	0,6713	0,9	3,43	3,093	3,59		3,519	3,794
	30.000	3,324	3,311	3,195	3,32	0,717					3,1	3,031	3,135		3,255	
	último paso	3,202	2,827	2,404	2,908	0,717	0,5582	0,7378			2,96	3,003	3,033		3,432	

Tabla 4.18. Entrenamientos Inception evaluados

4.2.2 Entrenamientos ssd_mobilenet_v1_coco

Como se hizo con el modelo anterior, los primeros modelos se entrenarán para ver el comportamiento de este tipo de arquitectura, que al estar diseñada para implementación en dispositivos móviles, tendrán un menor costo computacional y se entrenarán de manera más veloz. Se seguirá una metodología parecida a la anterior, donde se cambian ciertos parámetros y se estudia el efecto de estos; se adopta al modelo que mejor haya resultado, y se le vuelven a cambiar otros parámetros distintos, para así llegar al modelo óptimo.

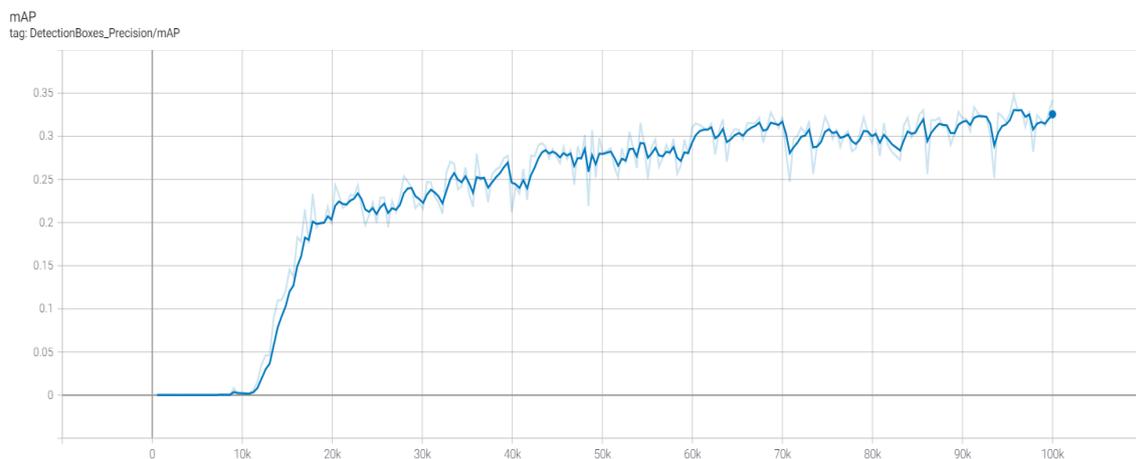
Entrenamiento n° 1:

SSD_mobilenet		Modelo 1
image_size		300 x 300
dropout		false
min_depth		16
depth_multiplier		1
optimizer		rms
learning_rate		
decay_step		
data_augmentation	horizontal_flip	X
	random_crop	X
	distort_color	
	adjust_brightness	
	rgb_to_gray	
	adjust_contrast	
adjust_saturation		
shuffle		false
steps (k)		100
batch_size		24

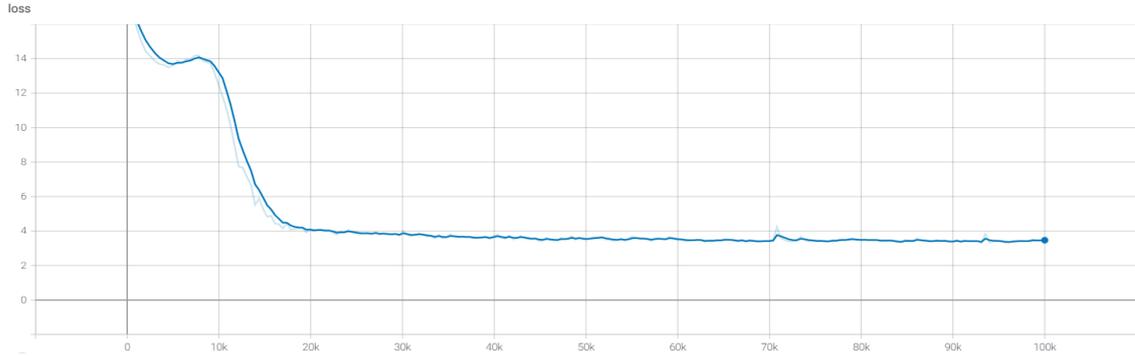
Se empiezan con los hiperparámetros predeterminados, configuración que en la arquitectura anterior dio muy buenos resultados. Se le asigna un valor grande de número de pasos para observar el comportamiento.

Tabla 4.19. Configuración modelo 1 *MobileNet*

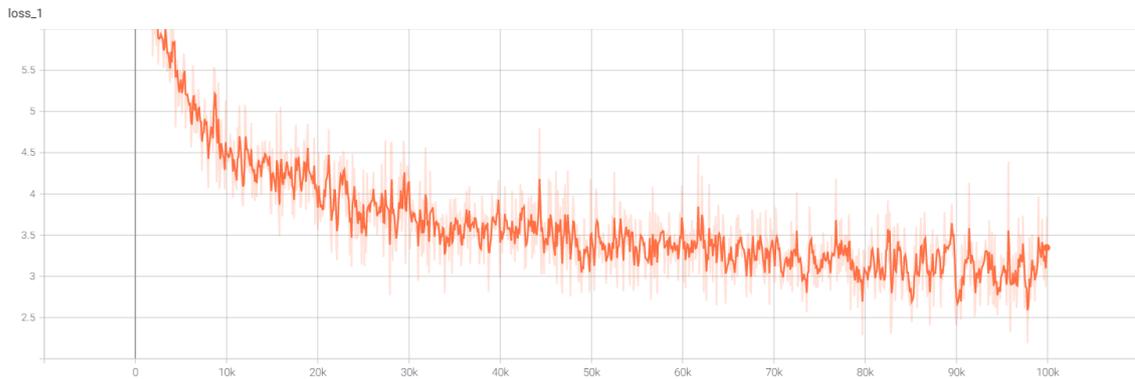
- mAP:



● Pérdida de evaluación:



● Pérdida de entrenamiento:



Se puede observar con este modelo, que esta arquitectura presentará un comportamiento similar a la realizada anteriormente. Con una precisión media por debajo de 0.35, con un ascenso lento pero gradual; y unas pérdidas en torno a 3.5. Algunas diferencias se aprecian como puede ser el número de pasos en el que el modelo empieza a incrementar su precisión media, esta empieza más tarde de los 10.000 pasos, mientras que en el modelo anterior era antes de los 5.000. Por otro lado, en la función de evaluación se observa un pequeño incremento en la curva antes de volver a decrecer. Se estudiará más adelante esta anomalía. El tiempo de entrenamiento ha sido 1 día, 14 horas y 37 minutos.

Entrenamiento n° 2:

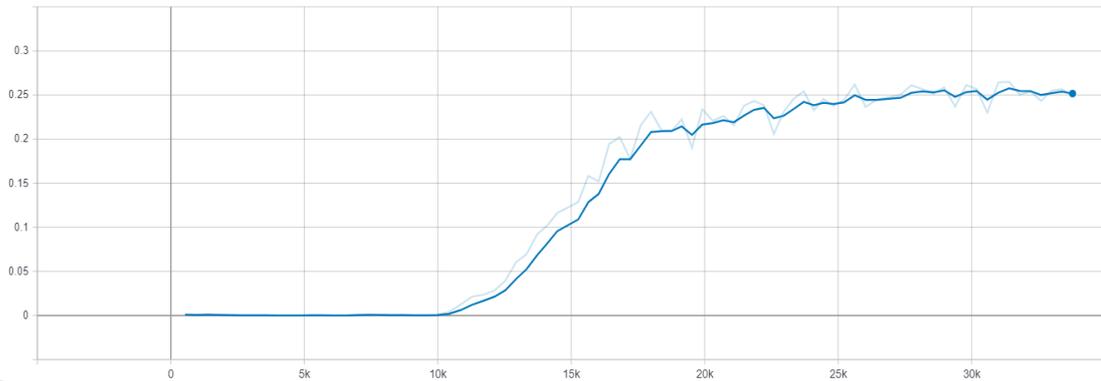
SSD_mobilenet		Modelo 2
Image_size		300X300
dropout		false
min_depth		8
depth_multiplier		1
optimizer		rms
learning_rate		
decay_step		
data_augmentation	horizontal_flip	X
	random_crop	X
	distort_color	
	adjust_brightness	
	rgb_to_gray	X
	adjust_contrast	
adjust_saturation		
shuffle		false
steps (k)		33
batch_size		24

En esta configuración, se fija el parámetro *min_depth* a 8, a la mitad del modelo anterior, y se recuerda que este parámetro fija la profundidad mínima de cada una de las salidas de los mapas de características producidas por el extractor de características. Al hacer la red menos profunda, ésta debería tener una mayor velocidad, pero seguramente sea a costa de una menor precisión. También se incrementa el volumen de datos con la opción de añadir unas fotos en blanco y negro. Se pasa a analizar las gráficas:

Tabla 4.20. Configuración modelo 2 *MobileNet*

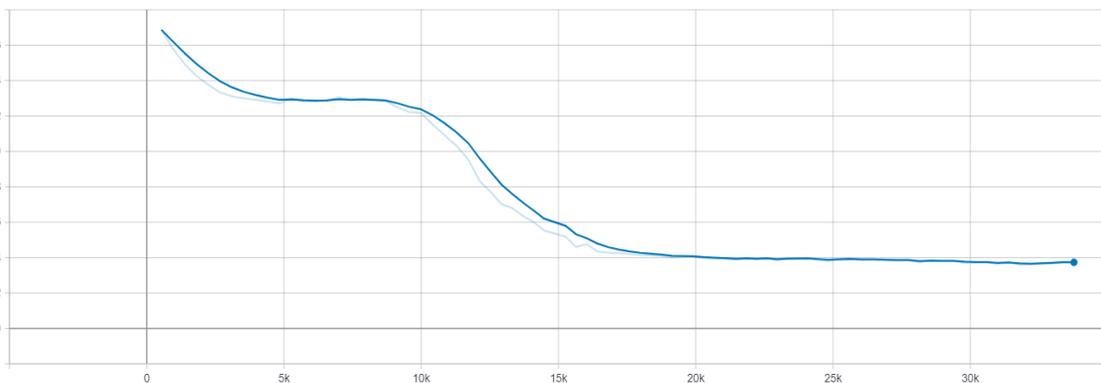
- mAP:

mAP
tag: DetectionBoxes_Precision/mAP



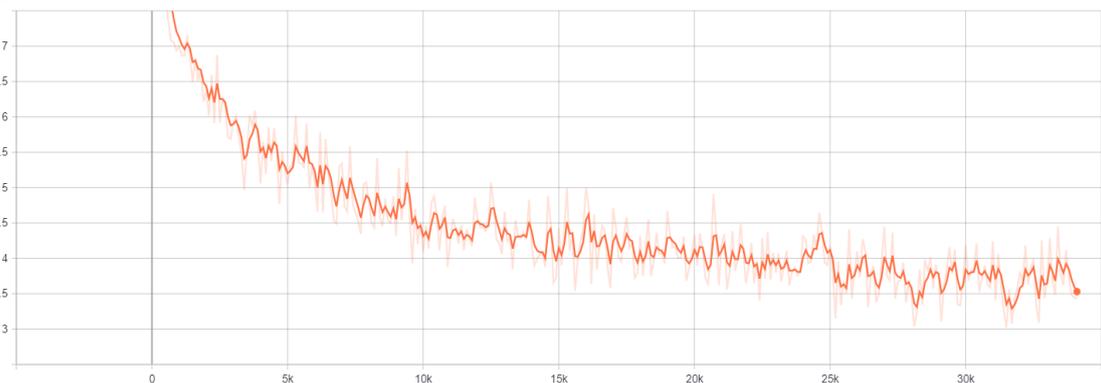
- Pérdida de evaluación:

loss



- Pérdida de entrenamiento:

loss_1



En este caso, el modelo presenta un buen ascenso en la función de la precisión media, aunque esta a partir de los 25.000 pasos se estanca y apenas aumenta. En cuanto a las pérdidas se sigue obteniendo valores inferiores a 4, y en la pérdida de evaluación se ve un estancamiento previo a llegar al valor mínimo. Esto puede ser debido al valor del ratio de aprendizaje, el cual se estudiará posteriormente y se intentará eliminar esta anomalía.

El tiempo del entrenamiento ha sido de 13 horas, 58 minutos y 57 segundos.

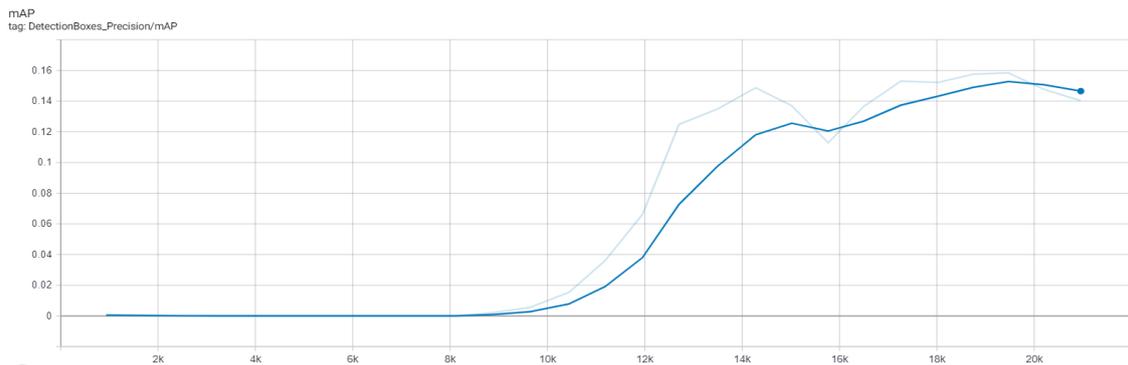
Entrenamiento n° 3:

SSD_mobilenet		Modelo 3
image_size		300 x 300
dropout		false
min_depth		8
depth_multiplier		0,5
optimizer		rms
learning_rate		
decay_step		
data_augmentation	horizontal_flip	X
	random_crop	
	distort_color	
	adjust_brightness	
	rgb_to_gray	X
	adjust_contrast	
adjust_saturation		
shuffle		
steps (k)		20
batch_size		24

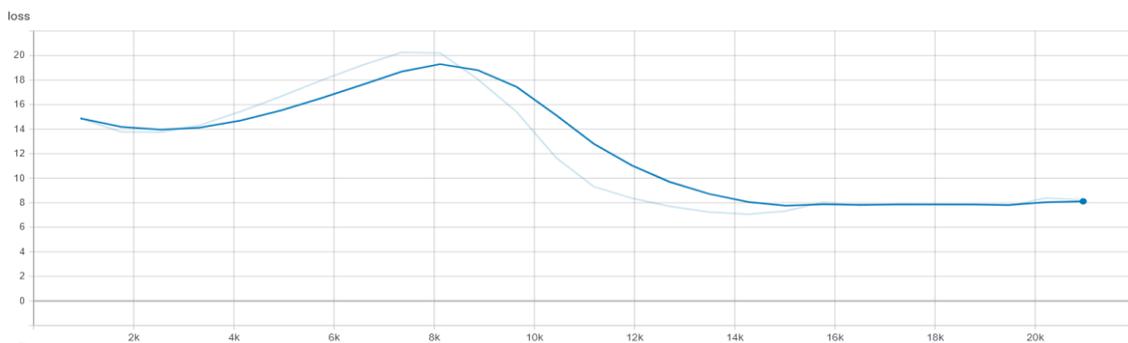
Se establece para esta configuración una profundidad mínima, para estudiar cómo afecta la profundidad a las métricas a estudiar, ya que estas configuraciones tienen un costo computacional relativamente bajo en comparación con modelos más profundos.

Tabla 4.21. Configuración modelo 3 *MobileNet*

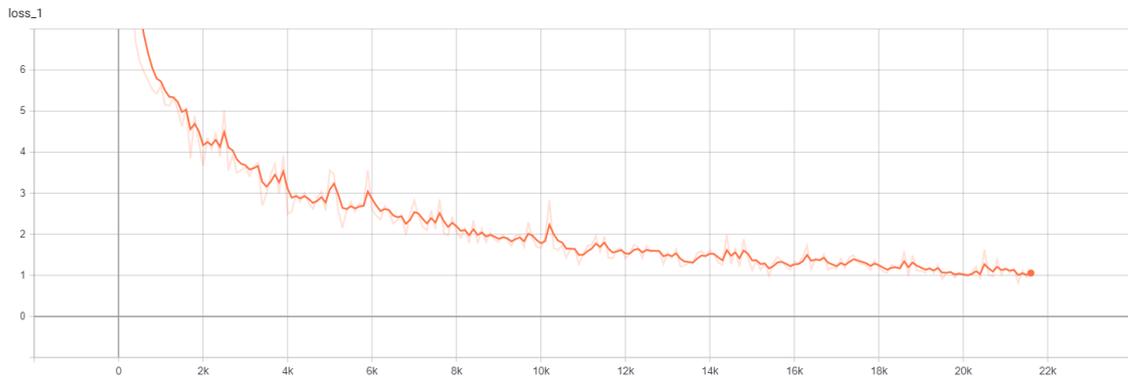
- mAP:



- Pérdida de evaluación:



- Pérdida de entrenamiento:



Se observa como el reducir la profundidad, degrada considerablemente el resultado del modelo; obteniendo una precisión media muy baja, la cual no llega ni a un valor de 0.16 a los 20.000 pasos; y una pérdida de evaluación alta y que presenta un comportamiento no deseado, ya que esta llega un punto en que empieza a aumentar. La pérdida de entrenamiento si consigue un valor bastante bajo, en torno a 1, lo que significa que el modelo ha aprendido muy bien los patrones en el conjunto de datos de entrenamiento; sin embargo, es incapaz de generalizar para el conjunto de evaluación. El tiempo de entrenamiento del modelo, como se comentó anteriormente, ha disminuido notablemente siendo de 4 horas, 20 minutos y 3 segundos.

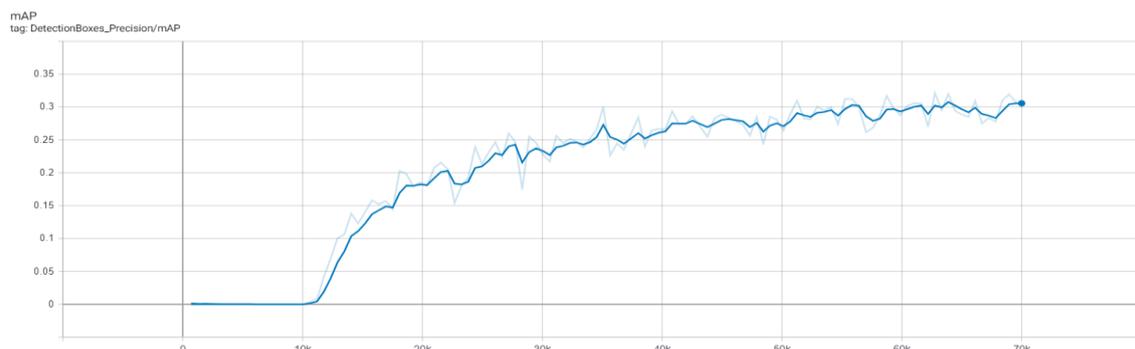
Entrenamiento n° 4:

SSD_mobilenet		Modelo 4
image_size		300 x 300
dropout		true
min_depth		16
depth_multiplier		0,75
optimizer		rms
learning_rate		
decay_step		
data_augmentation	horizontal_flip	
	random_crop	X
	distort_color	
	adjust_brightness	
	rgb_to_gray	X
	adjust_contrast	
adjust_saturation		
shuffle		
steps (k)		70
batch_size		24

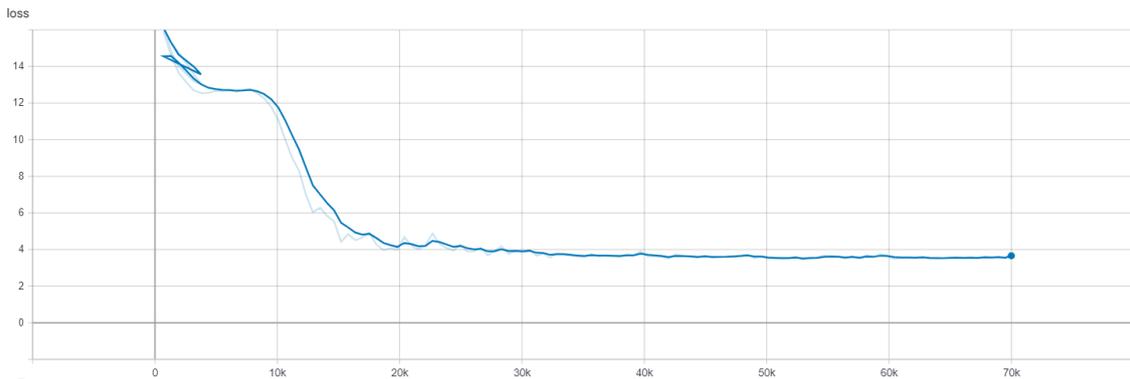
En este modelo se siguen probando diferentes variantes de la profundidad; en este caso, se reduce el número de canales de entrada de cada capa. Como se vio, esta configuración fue también una de las más óptimas para la otra arquitectura, es por ello que se decide entrenar un mayor tiempo.

Tabla 4.22. Configuración modelo 4 *MobileNet*

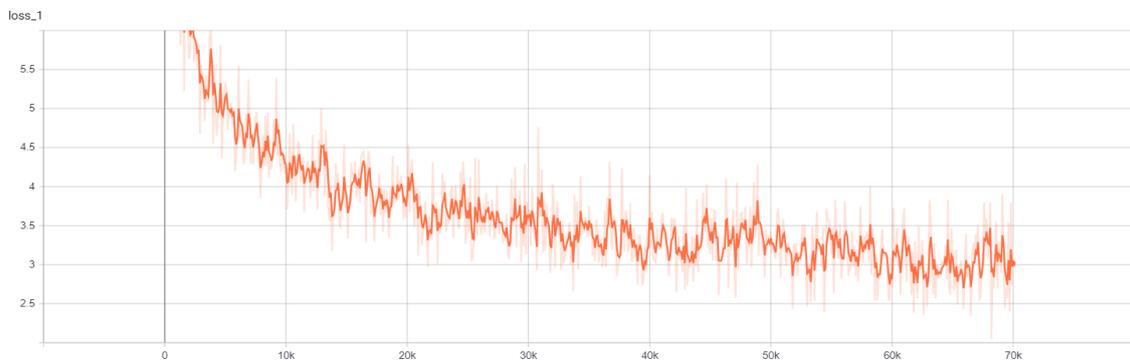
- mAP:



- Pérdida de evaluación:



- Pérdida de entrenamiento:



Este modelo ha conseguido unos resultados mediocres en cuanto a precisión media si se pone el foco durante los pasos 20.000 - 30.000. Por otro lado, la diferencia de ambas pérdidas se ha incrementado, de manera que el *overfitting* es mayor en este caso.

A continuación se comparan los cuatro modelos que se han entrenado hasta ahora, con los resultados resumidos en Tabla 4.23 para que se pueda visualizar mejor:

SSD_mobilenet		Modelo 1	Modelo 2	Modelo 3	Modelo 4
steps (k)		100	33	20	70
mAP	20.000	0,2195	0,2182	0,14	0,2084
	30.000	0,2227	0,2547		0,2678
	último paso	0,3256	0,2516		0,3191
pérdida_evaluación	20.000	3,986	4,032	8,2	4,136
	30.000	3,869	3,749		3,741
	último paso	3,468	3,734		3,485
pérdida_entrenamiento	20.000	3,838	4,124	1,12	3,943
	30.000	3,727	3,842		3,813
	último paso	3,349	3,531		2,876

Tabla 4.23. Comparación resultados de los primeros cuatro modelos *MobileNet*

Se observa que todos los modelos, excepto el 3 tienen unos resultados muy parecidos. Para empezar, el modelo 3 se descarta por la gran presencia de *overfitting* que hay. En cuanto al resto de los modelos, el 1 tiene una buena actuación, donde incluso en el paso 100k no se aprecia el *overfitting*.

Los valores de la precisión al principio parece que se estancan, pero van incrementándose gradualmente hasta obtener un buen resultado. El modelo 4, tiene una muy buena precisión media, y aunque las pérdidas tienen una ligera diferencia, se aceptará y se estudiará posteriormente, pues ésta no es significativa. En cambio, el modelo 2, empieza obteniendo muy buenos resultados, pero estos se estancan cuando pasa del paso 30k; excepto la pérdida de entrenamiento que sigue mejorando, lo que indica que si avanzáramos el entrenamiento, el *overfitting* posiblemente se incrementaría. Por tanto, de estos primeros cuatro modelos, se escogen el modelo 1 y el modelo 4 como modelos de referencia para configuraciones posteriores.

Se sigue con el entrenamiento de modelos, y se pasa al estudio de como ciertas opciones de aumento de datos afectan a la respuesta, para descubrir si esta arquitectura tiene una respuesta similar a la entrenada anteriormente.

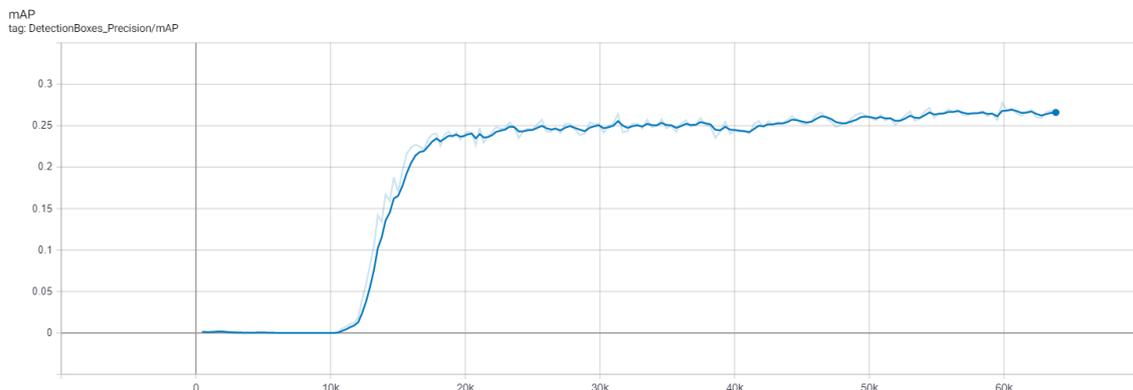
Entrenamiento n° 5:

SSD_mobilenet		Modelo 5
Image_size		300 X 300
dropout		true
min_depth		16
depth_multiplier		1
optimizer		rms
learning_rate		
decay_step		
data_augmentation	horizontal_flip	
	random_crop	
	distort_color	
	adjust_brightness	X
	rgb_to_gray	
	adjust_contrast	
	adjust_saturation	X
shuffle		
steps (k)		50
batch_size		24

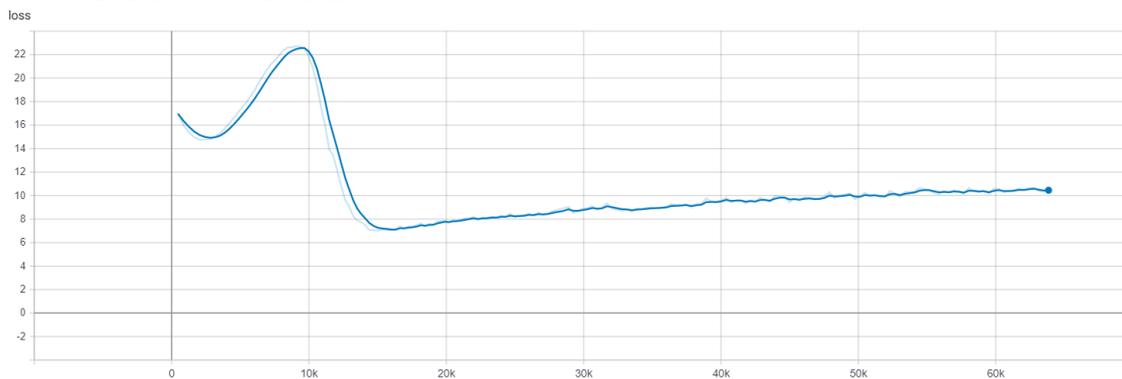
Esta configuración surge a partir del modelo 1, cambiando las opciones del aumento de datos, para un aumento de imágenes con variabilidad en los niveles de luminosidad y de saturación diferentes.

Tabla 4.24. Configuración modelo 5 *MobileNet*

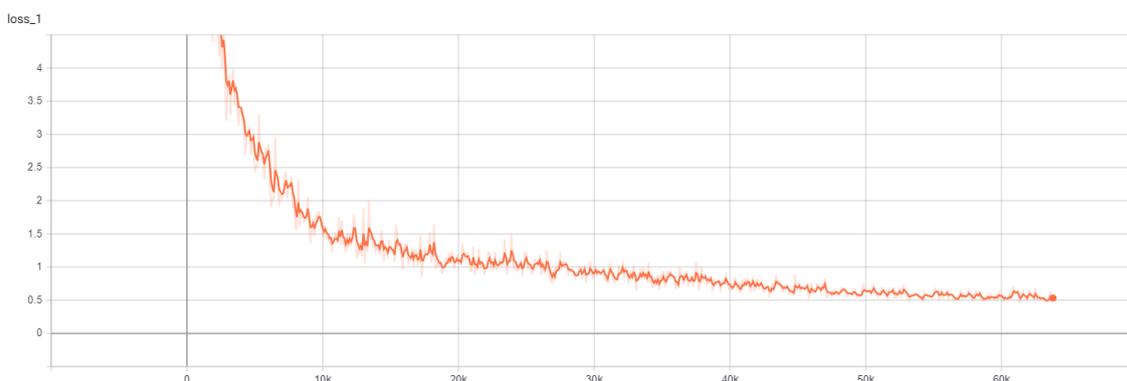
- mAP:



- Pérdida de evaluación:



- Pérdida de entrenamiento:



Se optó por quitar el entrenamiento, ya que la disparidad de las funciones de pérdida iba en aumento. Se observa un resultado en el conjunto de entrenamiento excepcional; mientras que en el conjunto de evaluación, la red diverge y no es capaz de realizar buenas detecciones. Esto como, ya se ha comentado, es que el modelo se ha aprendido los patrones específicos para el conjunto de entrenamiento, y no es capaz de generalizar para el conjunto de evaluación, con lo que la pérdida de éste incrementa.

Resultados similares han ocurrido con los entrenamientos en los que se ha activado la opción de aumentar datos mediante luminosidad, saturación, contraste, entre otros; como ocurría en la arquitectura anterior. Con estos tipos de datos, se pretendía que la red fuera capaz de detectar bien los objetos en días con mucha luminosidad o con presencia de niebla; pero los cambios que aplica a las fotos la red son muy drásticos, y no se corresponden a la realidad, con lo que la red es incapaz de generalizar para el conjunto de datos de evaluación. Se resumirán los resultados de forma esquemática en Tabla 4.25 a continuación.

Todas estas configuraciones anteriores tienen una característica en común, aprenden muy bien para el conjunto de entrenamiento; pero no son capaces de generalizar para imágenes nuevas. Claro casos de *overfitting*. Además, todas las pérdidas de evaluación en un momento empiezan a incrementar su valor.

En el modelo 6 se intentó solucionar este inconveniente estableciendo un ratio de aprendizaje mucho menor del predeterminado, que es 0.004, pero fue en vano; aunque se observa que las pérdidas están más cercanas entre ellas, la función de evaluación vuelve a incrementar su valor tras ciertos pasos.

SSD_mobilenet		Modelo 5	Modelo 6	Modelo 7	Modelo 8
Image_size		300 X 300	300 x 300	300 X 300	150 x 150
dropout		true	true	true	true
min_depth		16	16	16	8
depth_multiplier		1	0,75	0,75	0,5
optimizer		rms	rms	rms	rms
learning_rate			0,0009		
decay_step					
data_augmentation	horizontal_flip		X	X	
	random_crop				
	distort_color		X	X	
	adjust_brightness	X			X
	rgb_to_gray				X
	adjust_contrast				
adjust_saturation		X			
shuffle					
steps (k)		50	60	43	29
batch_size		24	16	20	16
mAP	20.000	0,2395	0,1497	0,1874	0,1
	30.000	0,247	0,1775	0,208	
	último paso	0,2659	0,2089	0,1976	0,1459
pérdida_evaluación	20.000	7,749	4,864	6,364	10,3
	30.000	8,833	5,061	7,096	
	último paso	10,46	6,192	7,926	9,983
pérdida_entrenamiento	20.000	1,071	3,172	1,745	0,7713
	30.000	0,953	2,33	1,274	
	último paso	0,5334	1,565	0,8296	0,684

Tabla 4.25. Configuraciones y resultados de modelos

Se continúan entrenando modelos, los cuales tendrán como configuración la del modelo 1, y se cambiará el tamaño de imagen de entrada y el ratio de aprendizaje, para llegar a un modelo óptimo.

Entrenamiento n° 9, 10 y 11:

En estos tres modelos se ha modificado el ratio de aprendizaje respecto al usado en los modelos anteriores, se ha incrementado de 0.004 a 0.01. También se ha definido un descenso exponencial de este ratio cada 3.500 pasos en forma de escalón. La diferencia de estos tres modelos radica en el tamaño de la imagen de entrada, como se puede ver en Tabla 4.26 a continuación.

Se ve en los resultados obtenidos de los modelos que todos los modelos presentan un desempeño realmente bueno; es por eso que se decidió entrenar hasta los 40.000 pasos. Se empiezan valorando las diferentes pérdidas, y se concluye que los tres obtienen valores muy parecidos, que abarca entre los 3.3 – 3.5. Todas las pérdidas de evaluación y de entrenamiento son muy parecidas, por lo que no existe el *overfitting*.

Lo que diferencia a estos modelos es la precisión media, y es que el que un peor resultado ha obtenido ha sido el modelo con una imagen de entrada menor. Éste empezó durante los primeros pasos satisfactoriamente, pero como se aprecia en los resultados, no obtuvo una mejora notable durante los siguientes 20.000 pasos. En cuanto a los otros dos modelos, cabría esperar la posibilidad de que, como el que tiene una imagen de entrada más pequeña es el que peor resultado ha obtenido, cuanto más grande sea la imagen, mejor resultados obtendrá. Pero esto no es así, como se puede comprobar el que mejor precisión media final ha tenido durante los últimos 10.000 pasos ha sido ese cuya imagen ha sido de 400 x 400, superando así al modelo con tamaño de imagen de 500 x 500.

SSD_mobilenet		Modelo 9	Modelo 10	Modelo 11
Image_size		300 x 300	400 x 400	500 x 500
dropout		true	true	true
min_depth		16	16	16
depth_multiplier		1	1	1
optimizer		rms	rms	rms
learning_rate		0,01	0,01	0,01
decay_step		3500	3500	3500
data_augmentation	horizontal_flip	X	X	X
	random_crop	X	X	X
	distort_color			
	adjust_brightness			
	rgb_to_gray			
	adjust_contrast			
shuffle		true	true	true
steps (k)		40	40	40
batch_size		24	24	24
mAP	20.000	0,2542	0,2715	0,2859
	30.000	0,2855	0,3556	0,3384
	último paso	0,2842	0,3456	0,3415
pérdida_evaluación	20.000	3,749	3,784	3,935
	30.000	3,428	3,325	3,466
	último paso	3,361	3,364	3,374
pérdida_entrenamiento	20.000	3,756	3,808	3,424
	30.000	3,684	3,428	3,221
	último paso	3,313	3,454	3,408

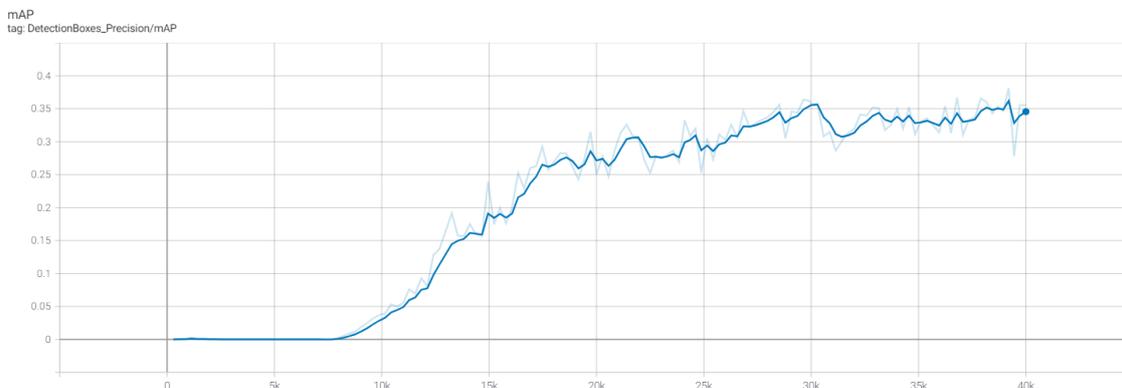
Tabla 4.26. Configuración modelos 9, 10 y 11 *MobileNet*

Así mismo, marcar la diferencia del coste computacional de estos modelos, a causa de la diferencia del tamaño de sus imágenes de entrada, señalando que el primer modelo necesito 15 horas; el segundo, 24 horas; y el tercero, 1 día y 17 horas.

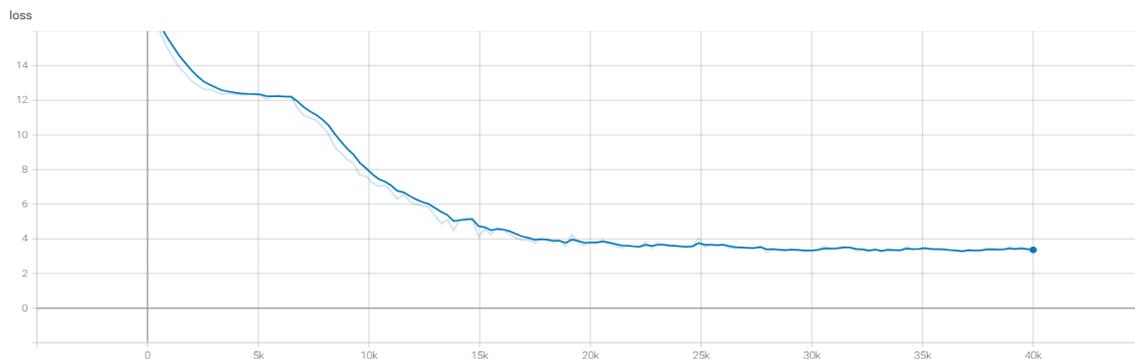
Aun así, como los dos modelos últimos denotan un desempeño excepcional, se estudiarán más adelante, para ver cual tiene una mayor velocidad.

Se muestran a continuación las gráficas del modelo con mejores resultados, el modelo 10:

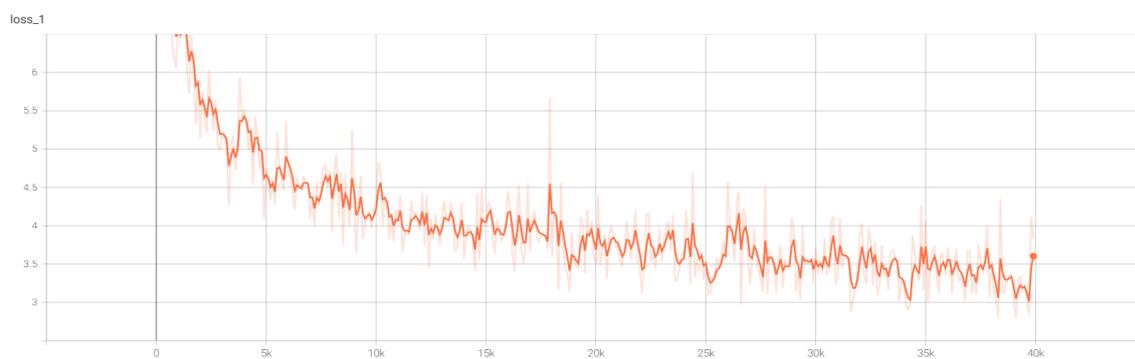
- mAP:



- Pérdida de evaluación:



- Pérdida de entrenamiento:



Se observa unos buenos resultados como se indicó, tanto de precisión media como de pérdidas, que no difieren mucho entre ellas. La precisión media tiene un aumento constante y gradual, indicando que si se dejara más tiempo entrenando, ésta seguiría creciendo.

En cuanto a la función de evaluación, se observa el estancamiento o meseta que se lleva dando con este modelo desde el inicio, con la diferencia de que se mantiene constante prácticamente, y de que tiene un recorrido mucho menor que los vistos anteriormente, empezando en el paso 4.500 y terminando un poco antes del 7.000.

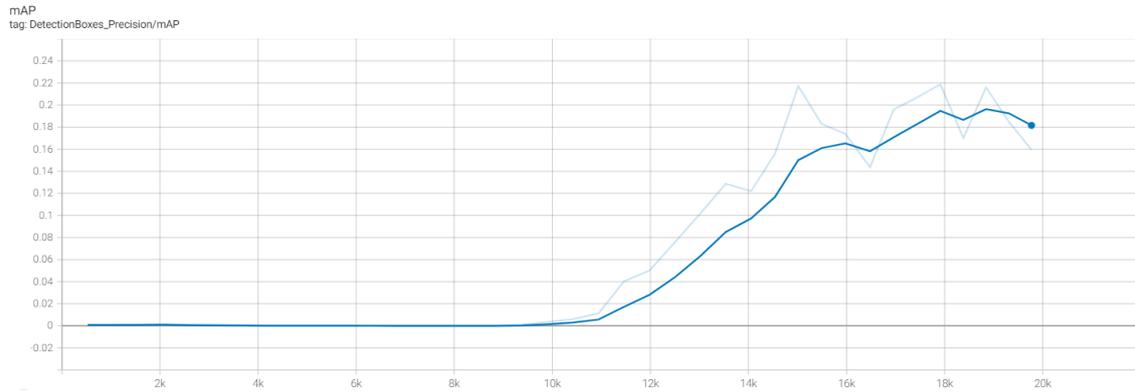
Entrenamiento n° 12:

SSD_mobilenet		Modelo 12
image_size	300x300	
dropout	true	
min_depth	16	
depth_multiplier	1	
optimizer	rms	
learning_rate	0,007	
decay_step	3500	
data_augmentation	horizontal_flip	X
	random_crop	X
	distort_color	
	adjust_brightness	
	adjust_contrast	
	adjust_saturation	
shuffle	true	
steps (k)	30k	
batch_size	24	

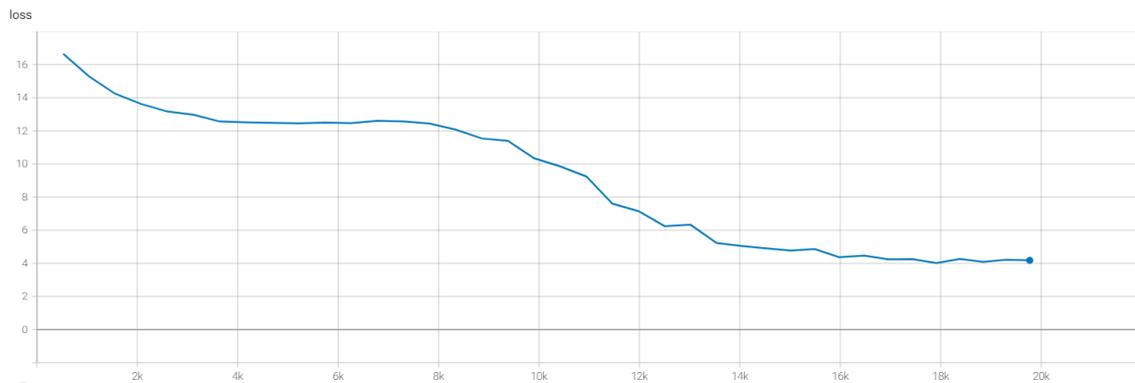
Con este modelo se quiere estudiar el efecto de disminuir unas milésimas el valor del ratio de aprendizaje utilizado en el modelo 9, para intentar mejorar los resultados mediocres que obtuvo.

Tabla 4.27. Configuración modelo 12 *MobileNet*

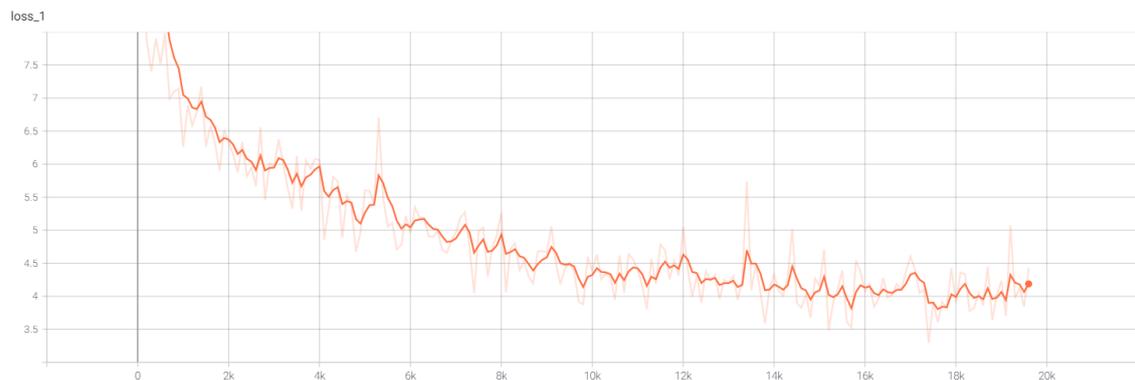
- mAP:



- Pérdida de evaluación:



- Pérdida de entrenamiento:



Con la disminución del ratio de aprendizaje se ha conseguido un modelo más lento, pero además, el ruido en la gráfica de la precisión media ha aumentado, el valor de esta ha disminuido, y no parece tener una mejora considerable. En cuanto a las gráficas de las pérdidas, tampoco han conseguido una mejora, ya que al tener un ratio menor y solo entrenarlo durante 20.000 pasos, es normal este resultado; sin embargo, se decidió no seguir entrenando este modelo, ya que la meseta observada en la gráfica de evaluación, ha aumentado considerablemente su extensión, sobrepasando los 8.000 pasos, y empezando antes del paso 4.000; y lo que se busca es eliminar en la medida de lo posible, este estancamiento previo al valor final.

Este entrenamiento ha tenido una duración de 6 horas, 20 minutos y 25 segundos.

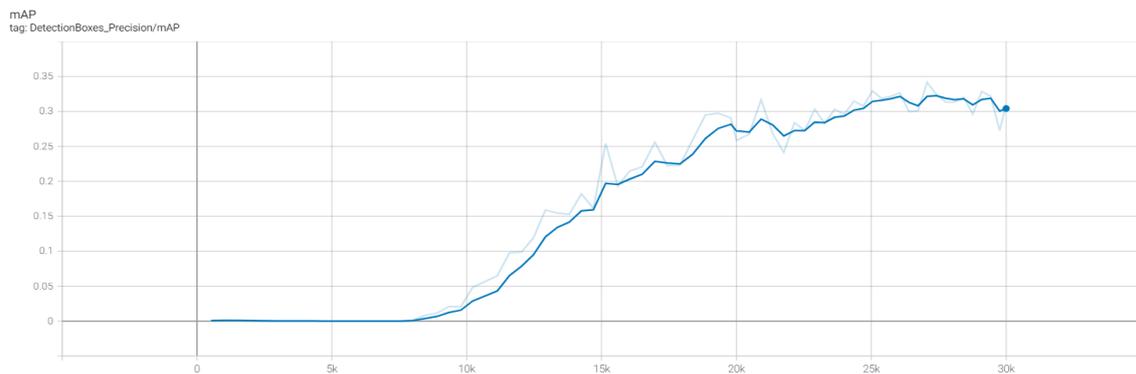
Entrenamiento n° 13:

SSD_mobilenet		Modelo 13
Image_size	300x300	
dropout	true	
min_depth	16	
depth_multiplier	1	
optimizer	rms	
learning_rate	0,012	
decay_step	1000	
data_augmentation	horizontal_flip	X
	random_crop	X
	distort_color	
	adjust_brightness	
	rgb_to_gray	
	adjust_contrast	
adjust_saturation		
shuffle	true	
steps (k)	30	
batch_size	24	

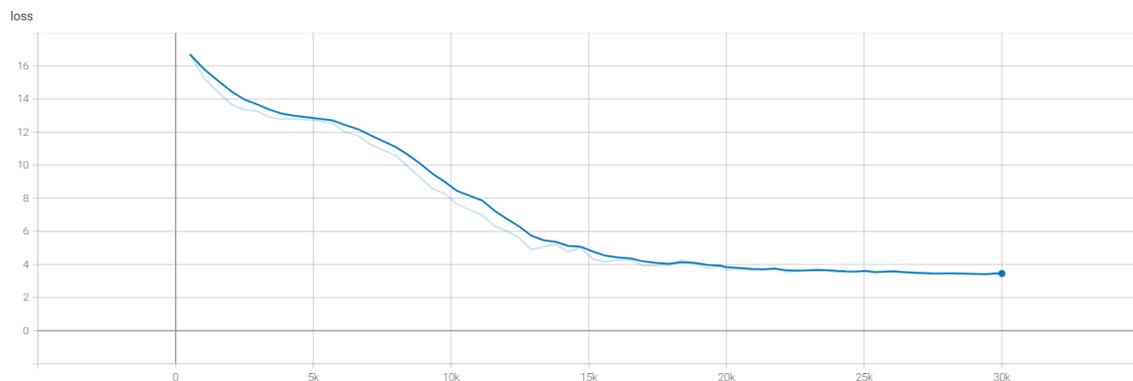
Dado que en el modelo anterior, la disminución del ratio de aprendizaje supuso una degradación en el desempeño del modelo; en esta configuración se ha optado por aumentar este valor, y utilizar un decrecimiento exponencial en forma de escalón cada 1.000 pasos. De este modo, se busca una optimización del modelo con tamaño de imagen de entrada de 300 x 300.

Tabla 4.28. Configuración modelo 13 *MobileNet*

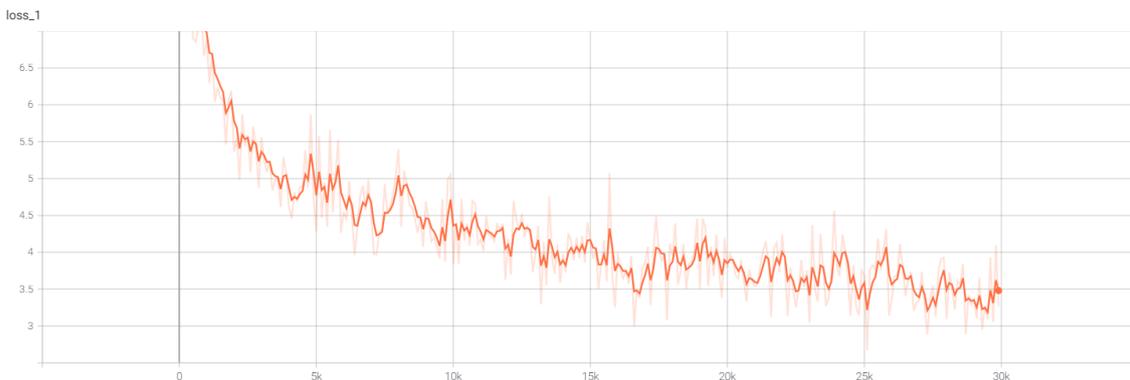
- mAP:



- Pérdida de evaluación:



- Pérdida de entrenamiento:



En este modelo se observa que su ejecución es notablemente mejor al modelo anterior, superando la barrera de la precisión media mayor de 0.3; con pérdidas menores que 3.5, y con una la presencia de una meseta en la función de evaluación mucho menor. Por tanto, se confirma que el aumento del ratio de aprendizaje para esta configuración fue un acierto.

Entrenamiento n° 14 y 15:

SSD_mobilenet		Modelo 14	Modelo 15
image_size		400 x 400	400 x 400
dropout		true	true
min_depth		16	16
depth_multiplier		1	1
optimizer		rms	rms
learning_rate		0,008	0,012
decay_step		3500	1000
data_augmentation	horizontal_flip	X	X
	random_crop	X	X
	distort_color		
	adjust_brightness		
	rgb_to_gray		
	adjust_saturation		
shuffle		true	true
steps (k)		30	20
batch_size		24	24

Tabla 4.29. Configuración modelos 14 y 15 *MobileNet*

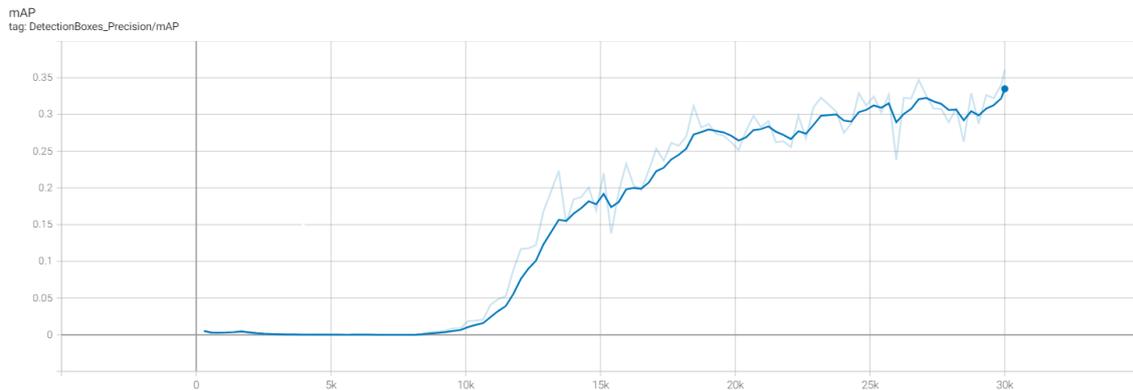
En estos dos últimos entrenamientos se sigue la misma metodología que para los dos anteriores; encontrar el ratio de aprendizaje óptimo para esta configuración con un tamaño de imagen de entrada de 400 x 400.

A pesar de que este ha sido el modelo con mejor resultado hasta ahora visto, se quiere investigar si hay un ratio de aprendizaje más adecuado para optimizar dicha configuración.

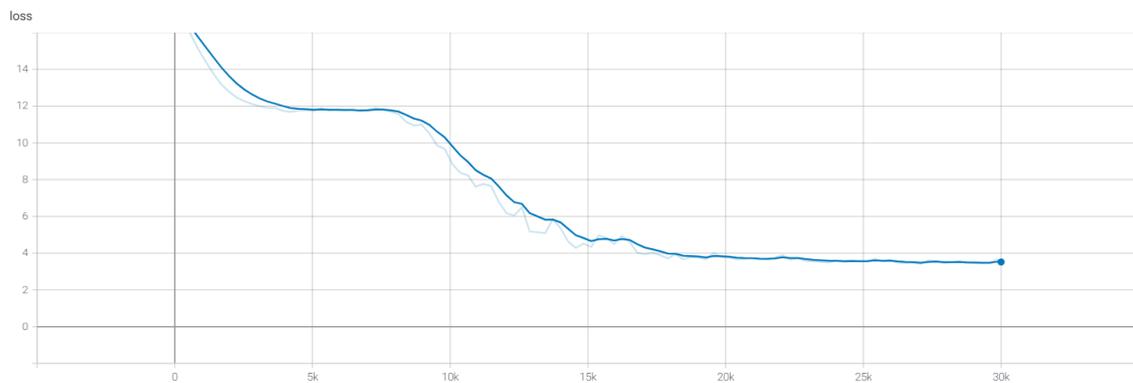
También apuntar que se ha optado por no realizar esta metodología con modelos cuya imagen de entrada es de 500 x 500, ya que en los modelos previamente estudiados, este no mostró un mejor comportamiento con respecto al que se procede a optimizar a continuación.

Entre estos dos modelos, el modelo 14 obtuvo unos valores más destacables, por lo que se presentan a continuación sus gráficas; y más adelante se estudiarán los resultados de estos cuatro últimos modelos.

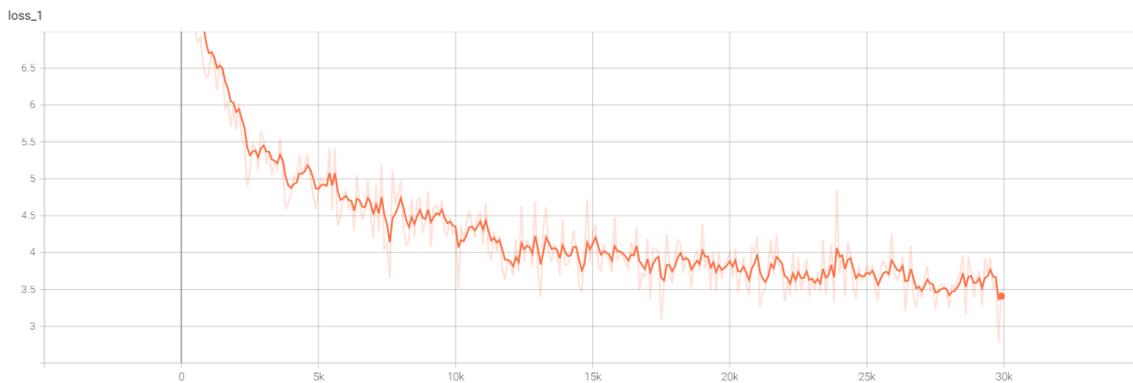
- mAP:



- Pérdida de evaluación:



- Pérdida de entrenamiento:



En las gráficas se ve el resultado notable de este entrenamiento, alcanzando un valor superior a 0.3 en la precisión media; y unas pérdidas alrededor de 3.5, como viene siendo de costumbre esta aplicación. Además, se observa que la meseta ha aumentado respecto de la meseta del modelo número 10; mientras que el modelo con un ratio de aprendizaje mayor, el modelo 15, también presentó una extensión de meseta mayor que dicho modelo.

A continuación se presentan los resultados numéricos de los últimos entrenamientos:

SSD_mobilenet		Modelo 12	Modelo 13	Modelo 14	Modelo 15
steps (k)		20	30	30	20
mAP	20.000	0,1816	0,2702	0,2646	0,2354
	30.000		0,3041	0,3249	
	último paso				
pérdida_evaluación	20.000	4,213	3,874	3,81	3,818
	30.000		3,457	3,519	
	último paso				
pérdida_entrenamiento	20.000	4,187	3,89	3,884	3,805
	30.000		3,482	3,409	
	último paso				

Tabla 4.30. Resultados modelos 12, 13, 14 y 15 *MobileNet*

Se observa claramente que el modelo 13 y 14 han sido los que mejor resultados han obtenido, por lo que se puede extraer una conclusión clara a partir de los modelos 9, 10 y 11, los cuales se han tomado de referencia para fijar los ratios de aprendizaje: un menor tamaño de imagen de entrada, 300 x 300, ha necesitado un mayor ratio de aprendizaje, 0.012; mientras que el tamaño de imagen 400 x 400, ha mostrado un mejor resultado con un ratio de aprendizaje menor, 0.008.

Una vez se han estudiado todos los modelos, se presenta un esquema de todos ellos en la página siguiente, Tabla 4.32. Se ha deducido que los modelos 10, 11, 13 y 14 son los seleccionados como óptimos para esta arquitectura. A continuación, se estudia su velocidad para añadir otra variable con la que se pueda finalmente elegir el modelo más óptimo.

SSD_mobilenet		Modelo 10	Modelo 11	Modelo 13	Modelo 14
Image_size		400 x 400	500 x 500	300x300	400 x 400
dropout		true	true	true	true
min_depth		16	16	16	16
depth_multiplier		1	1	1	1
optimizer		rms	rms	rms	rms
learning_rate		0,01	0,01	0,012	0,008
decay_step		3500	3500	3500	3500
data_augmentation	horizontal_flip	X	X	X	X
	random_crop	X	X	X	X
shuffle		true	true	true	true
steps (k)		40	40	30	30
batch_size		24	24	24	24
mAP	20.000	0,2715	0,2859	0,2702	0,2646
	30.000	0,3556	0,3384	0,3041	0,3249
	último paso	0,3456	0,3415		
pérdida_evaluación	20.000	3,784	3,935	3,874	3,81
	30.000	3,325	3,466	3,457	3,519
	último paso	3,364	3,374		
pérdida_entrenamiento	20.000	3,808	3,424	3,89	3,884
	30.000	3,428	3,221	3,482	3,409
	último paso	3,454	3,408		
velocidad (s)		0,4277545	0,25473675	0,6257605	0,45733475

Tabla 4.32. Modelos prometedores *MobileNet*

Estos cuatro modelos son los que un mejor desempeño han mostrado para la arquitectura *MobileNet*. Tienen una característica en común, y es que los cuatro tienen prácticamente la misma configuración que el primer modelo cuyos hiperparámetros eran los predeterminados. La única diferencia radica en el ratio de aprendizaje y el tamaño de imágenes con las que se alimenta el modelo.

Se empieza comparando las distintas precisiones medias, y se observa que los modelos 10 y 11 obtuvieron mejores valores en la etapa decisiva del entrenamiento, entre los 20.000 y 30.000 pasos; y estos dos obtuvieron finalmente en el paso 45.000 valores muy similares.

En lo que respecta a las distintas pérdidas, todas están en torno a los 3.5, tanto la evaluación como el entrenamiento, por lo que se afirma que no hay casos de *overfitting* entre los modelos presentes. Además, en el cómputo global de ambas pérdidas, los modelos 10 y 11 vuelven a ser los que mejores valores presentan; estando muy igualados, otra vez, estos dos modelos.

Para terminar, se ha calculado la velocidad de los distintos modelos en realizar las predicciones del *batch*, y se aprecia, que a mayor tamaño de la imagen de entrada, una mayor velocidad en realizar predicciones; llegando a ser más del doble la diferencia entre un tamaño de 500 x 500 y de 300 x 300. Por tanto, el modelo 11 es el que ha conseguido una mayor velocidad de predicción y con mucha diferencia respecto a los demás.

Así que se puede afirmar que es la velocidad la variable que marca la diferencia para elegir el modelo óptimo entre el modelo 10 y 11, siendo este último el que mejor desempeño ha mostrado para la arquitectura *MobileNet*.

Se muestra en Tabla 4.33 todos los modelos entrenados y sus resultados.

SSD_mobilenet	Modelo 1	Modelo 2	Modelo 3	Modelo 4	Modelo 5	Modelo 6	Modelo 7	Modelo 8	Modelo 9	Modelo 10	Modelo 11	Modelo 12	Modelo 13	Modelo 14	Modelo 15
image_size	300x300	150x150	300x300	400x400	500x500	300x300	300x300	400x400	400x400						
dropout	false	false	false	true	true	true	true	true	true						
min_depth	16	8	8	16	16	16	16	8	16	16	16	16	16	16	16
depth_multiplier	1	1	0.5	0.75	1	0.75	0.75	0.5	1	1	1	1	1	1	1
optimizer	rms	rms	rms	rms	rms	rms									
learning_rate					0,0009				0,01	0,01	0,01	0,007	0,012	0,008	0,012
decay_step									3500	3500	3500	3500	3500	3500	1000
horizontal_flip	X	X	X			X	X		X	X	X	X	X	X	X
random_crop	X	X		X					X	X	X	X	X	X	X
distort_color						X	X								
adjust_brightness					X			X							
rgb_to_gray		X	X	X				X							
adjust_contrast															
adjust_saturation					X										
shuffle	false	false	false						true	true	true	true	true	true	true
steps (k)	100	33	20	70	50	60	43	29	40	40	40	20	30	30	20
batch_size	24	24	24	24	24	16	20	16	24	24	24	24	24	24	24
mAP	0,2195	0,2182	0,14	0,2084	0,2395	0,1497	0,1874	0,1	0,2542	0,2715	0,2859	0,1816	0,2702	0,2646	0,2354
	0,2227	0,2547		0,2678	0,247	0,1775	0,208		0,2855	0,3556	0,3384		0,3041	0,3249	
último paso	0,3256	0,2516		0,3191	0,2659	0,2089	0,1976	0,1459	0,2842	0,3456	0,3415				
20.000	3,986	4,032	8,2	4,136	7,749	4,864	6,364	10,3	3,749	3,784	3,935	4,213	3,874	3,81	3,818
30.000	3,869	3,749		3,741	8,833	5,061	7,096		3,428	3,325	3,466		3,457	3,519	
último paso	3,468	3,734		3,485	10,46	6,192	7,926	9,983	3,361	3,364	3,374				
20.000	3,838	4,124	1,12	3,943	1,071	3,172	1,745	0,7713	3,756	3,808	3,424	4,187	3,89	3,884	3,805
30.000	3,727	3,842		3,813	0,953	2,33	1,274		3,684	3,428	3,221		3,482	3,409	
último paso	3,349	3,531		2,876	0,5334	1,565	0,8296	0,684	3,313	3,454	3,408				

Tabla 4.33. Entrenamientos MobileNet evaluados

4.3 Validación de los modelos

A continuación, se lleva a cabo la validación de los modelos anteriormente elegidos. Para ello, se mostrarán imágenes etiquetadas, y estas mismas imágenes tras analizarlas este modelo. Así que si se obtienen unos porcentajes altos de precisión, y unas buenas predicciones contrastadas con la imagen etiquetada real, se podrá declarar que sirven como un modelo óptimo para la aplicación de un sistema de seguimiento solar.

La métrica para determinar la fiabilidad de las detecciones es mAP, la cual se apoya en los valores de precisión, P , y exhaustividad, R , como se vio anteriormente, se establecen 6 fronteras para las intersecciones sobre la unión. De manera, que siguiendo la Figura 4.2, lo que esté por debajo de la línea roja tendrá más de un 90%; mientras que la naranja corresponde a un 10%. Así que si se obtienen unos porcentajes altos de precisión, y unas buenas predicciones contrastadas con la imagen etiquetada real, se podrá declarar el modelo como un modelo óptimo para la aplicación de un sistema de seguimiento solar.

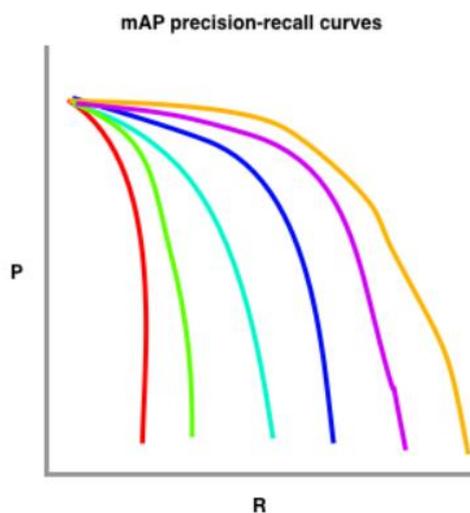


Figura 4.2. Distribución de las fronteras para métrica mAP [9].

Se muestran a continuación diferentes imágenes para los dos modelos a estudiar:

Inception.

En Figura 4.3 se observa la calidad de las predicciones realizadas por el modelo, ya que se trata de una imagen un poco alejada de la planta solar, por lo que el receptor está a bastante distancia del punto donde se tomó la fotografía. Aun así, el modelo es capaz de identificarlo con un 99% de precisión, al igual que el sol. En cuanto a los heliostatos, también es capaz de reconocerlos con un alto porcentaje de precisión, siendo el más bajo un 72 %; y detectando 5 de los 6 heliostatos que se etiquetaron en la imagen original; por lo que el resultado de las detecciones de este modelo sobre esta imagen es bastante aceptable.



Figura 4.3. Detecciones del modelo 13 *Inception* sobre imagen n° 1



Figura 4.4. Etiquetas de la imagen n° 1

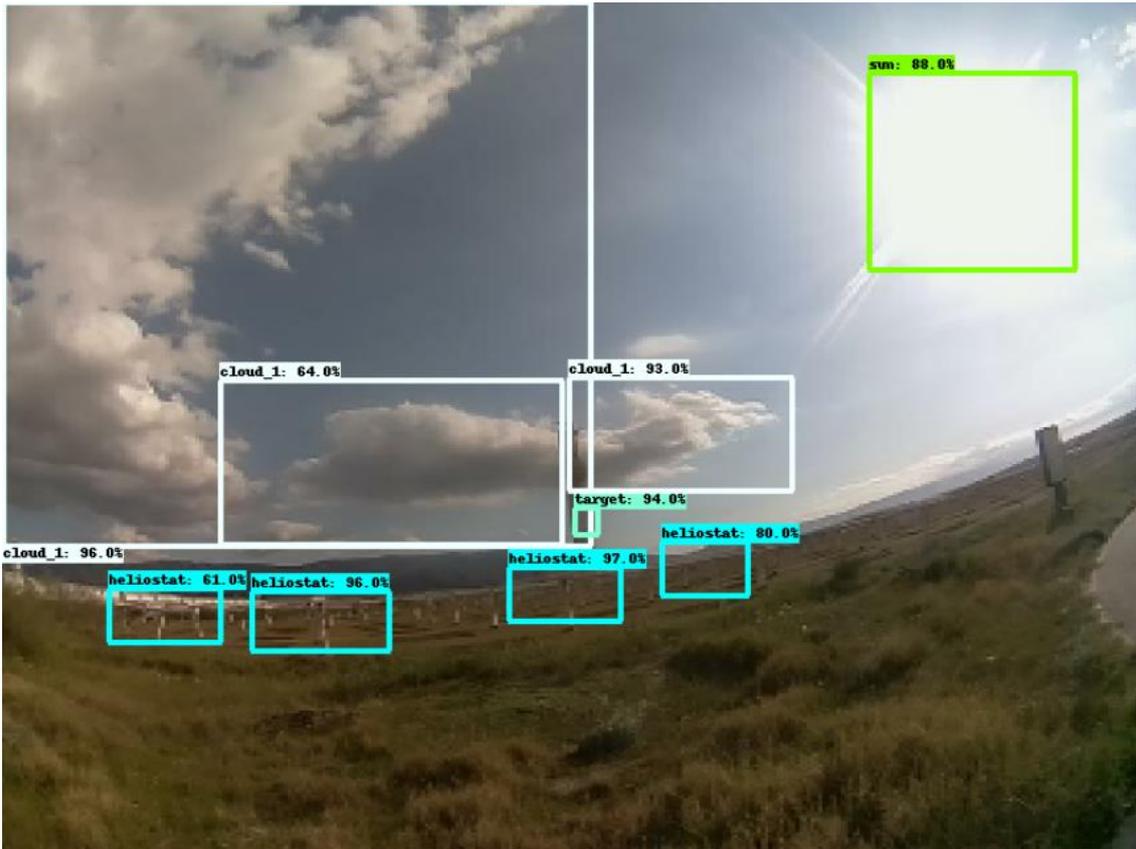


Figura 4.7. Detecciones del modelo 13 *Inception* sobre imagen nº 3

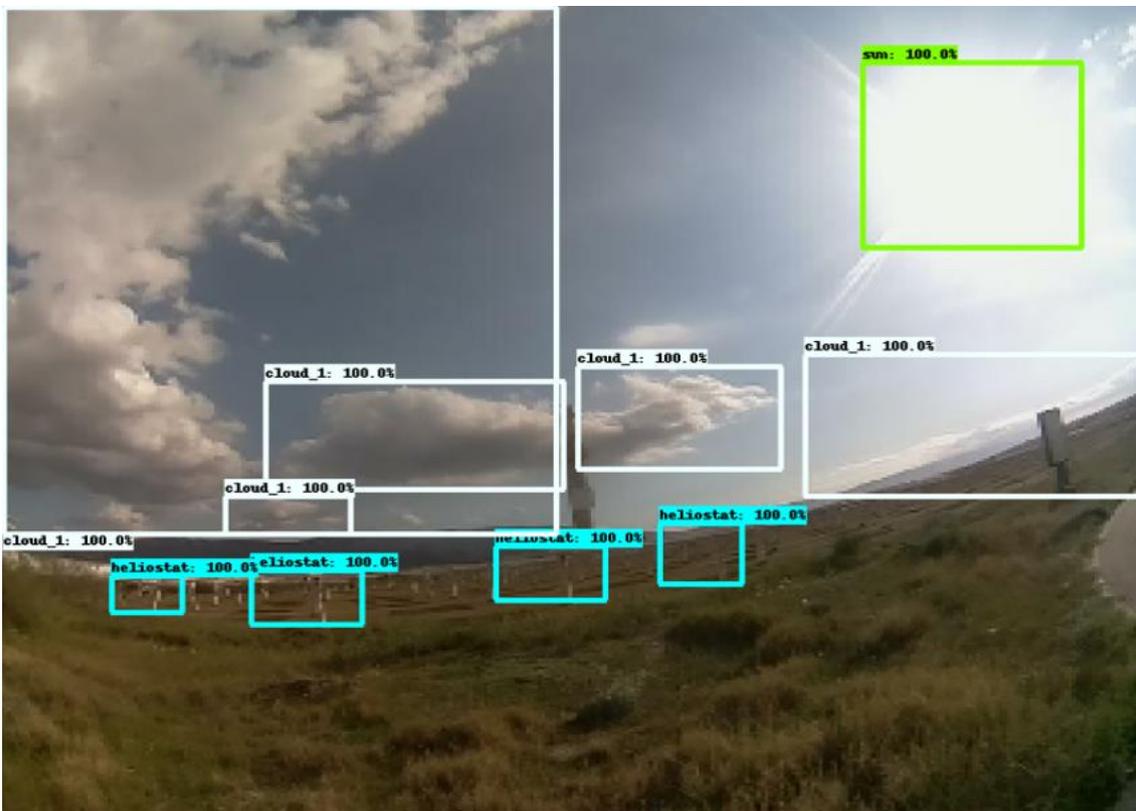


Figura 4.8. Etiquetas de la imagen nº 3

Finalmente, se elige una foto para ver el rendimiento del modelo de cara a reconocer las posibles nubes que se presenten a lo largo del día. A lo que el modelo responde satisfactoriamente, reconociendo las nubes de mayor tamaño; mientras que la nube lejana de menor tamaño que se encuentra a la derecha, no ha sido capaz de reconocerla. Tal vez, con un mayor tiempo de entrenamiento, la red hubiera sido capaz de reconocer esta nube. Además, reconoce los heliostatos más cercanos, coincidiendo con los cuatro que han sido etiquetados en la imagen original. Finalmente, se puede verificar la bondad de este modelo viendo que ha sido capaz de reconocer el receptor, que se encuentra a una gran distancia, haciendo un buen encuadre; cuando ni en la foto original, el receptor se etiquetó como objeto a detectar.

Por tanto, se concluye diciendo que el modelo 13 de la arquitectura *Inception* es adecuado para implementarlo en sistemas de seguimiento solar mediante visión artificial.

MobileNet

En esta primera imagen de vital importancia, como se comentó antes, ya que podría ser una de imagen de una cámara de cualquier heliostato situado en la planta, se ve que tiene unas predicciones de todos los heliostatos vecinos con una gran precisión y un alto porcentaje de fiabilidad; aunque se ven heliostatos que están un poco más alejados que no es capaz de reconocer. Aunque estos no son problemas ya que no están suficientemente cerca del foco de la cámara donde se situaría el heliostato, como para intervenir en la eficiencia de la planta. Además, el receptor es encuadrado de buena manera y con un 99%.

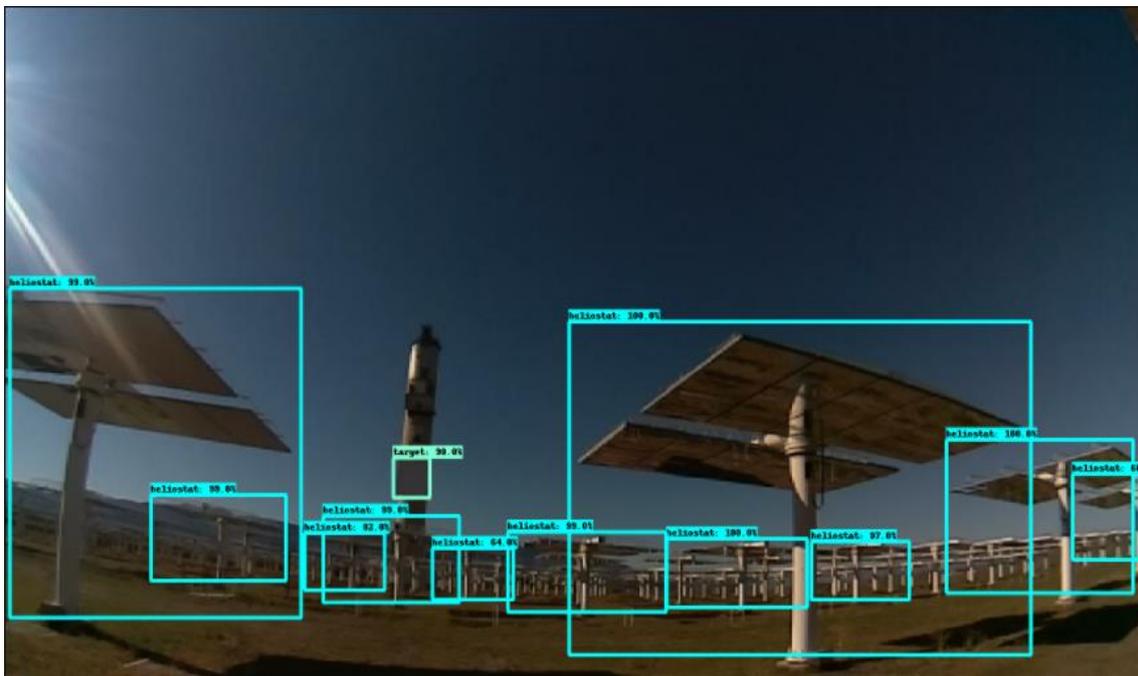


Figura 4.9. Detecciones del modelo 11 *MobileNet* sobre imagen n° 1

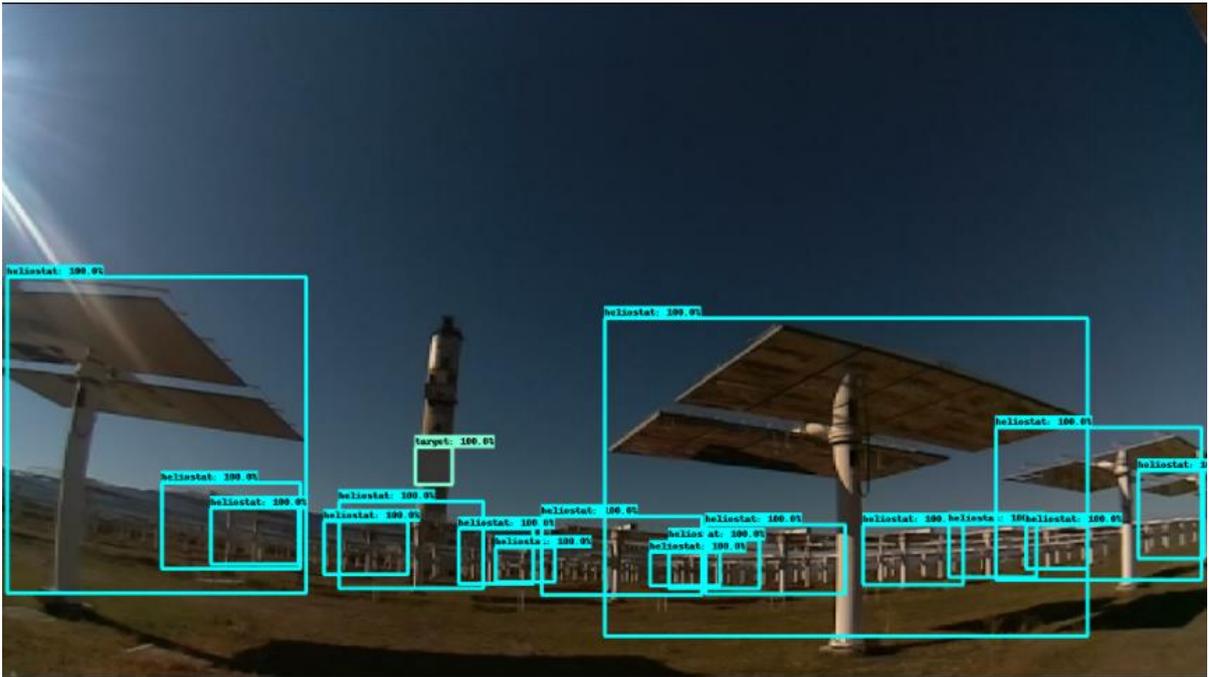


Figura 4.10. Etiquetas de la imagen nº 1



Figura 4.11. Detecciones del modelo 11 *MobileNet* sobre imagen nº 2

En esta segunda imagen, Figura 4.11, sin embargo, se aprecian algunas deficiencias del modelo. Como puede ser el mal encuadre del receptor, lo que podría llevar al desbordamiento del reflejo de radiación, una de las principales pérdidas de eficiencia de la planta de heliostatos. Una de las nubes es capaz de detectarla, sin embargo, la nube que se encuentra detrás de la torre no la detecta a pesar de que es una nube compacta y no está muy alejada. Los heliostatos los detecta con una precisión de alrededor un 80%. Estas deficiencias podrían ser solventadas, seguramente, si se dejará un mayor tiempo al modelo entrenando.

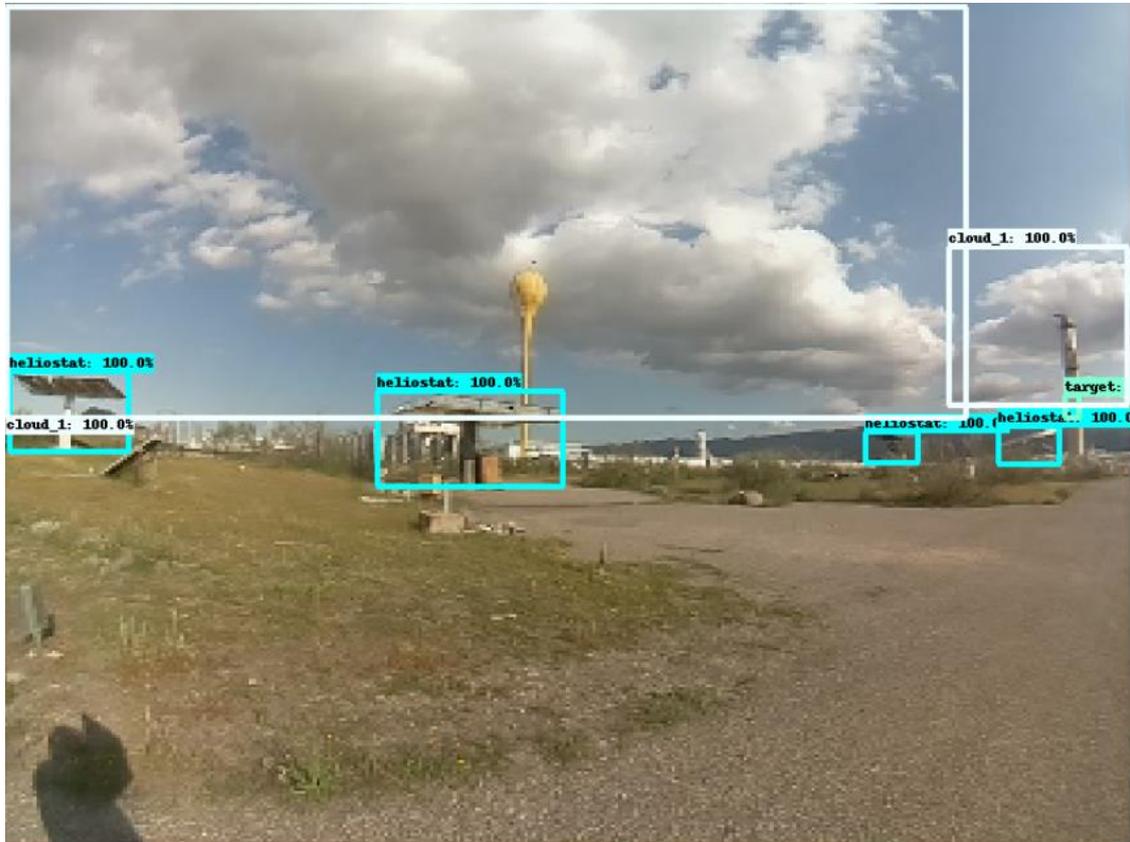


Figura 4.12. Etiquetas de la imagen n° 2

Para terminar, se ve que en otra imagen tomada desde el medio de la planta solar, el modelo es capaz de reconocer los heliostatos vecinos, el sol y el receptor con una precisión bastante alta. En cambio, como se vio en las fotos anteriores, los heliostatos que están más alejados, tienen una predicción pobre o no es capaz de reconocerlos.

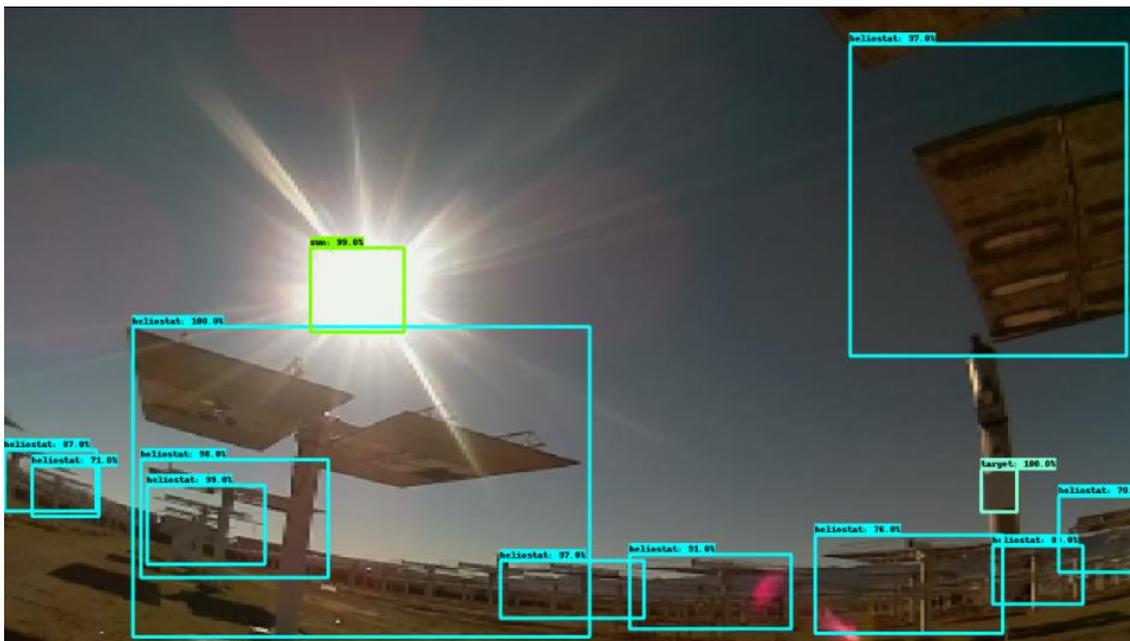


Figura 4.13. Detecciones del modelo 11 *MobileNet* sobre imagen n° 3

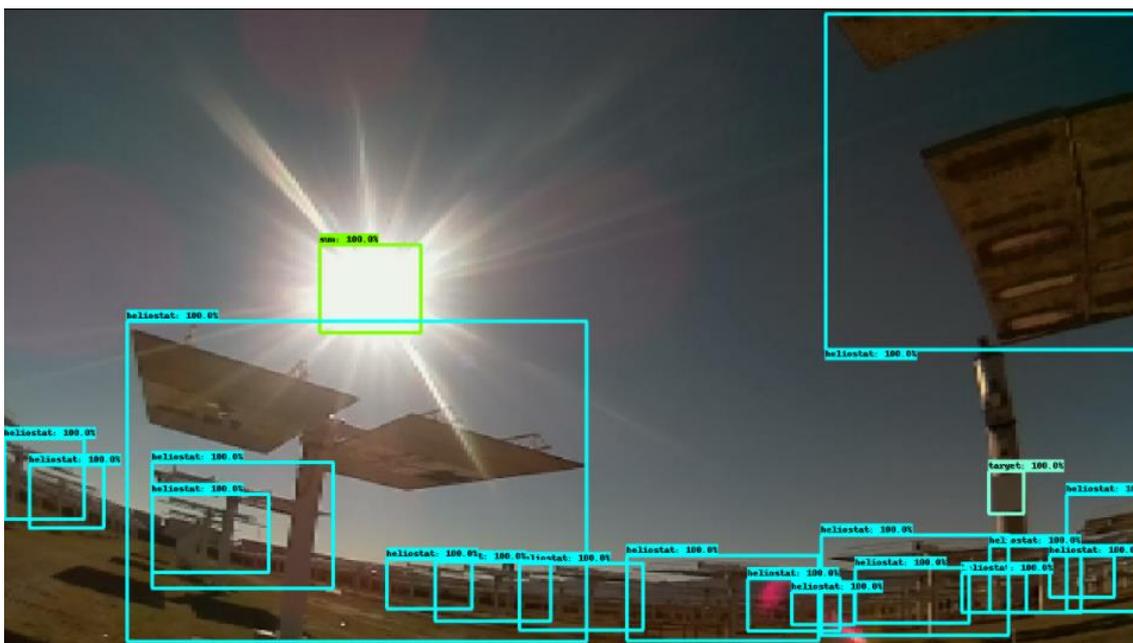


Figura 4.14. Etiquetas de la imagen n° 3

Por lo general, el desempeño de este modelo también ha sido bastante satisfactorio, encuadrando adecuadamente todos los objetos detectados, y obteniendo altos porcentajes de precisión, por lo que se afirma que puede ser implementado en sistemas de seguimiento solar mediante visión artificial.

4.4 Estudio del modelo óptimo

Una vez comprobado que los modelos seleccionados son válidos para la aplicación, se pasa a valorar ambos modelos juntos para comprobar si alguno de ellos tiene un mejor resultado que el otro. Para ello, se adjunta Tabla 4.34 con los parámetros y resultados de los modelos.

SSD		Inception	MobileNet
Image_size	fixed_shape_resizer	400 x 400	500 x 500
	keep_aspect_ratio		
dropout		true	true
min_depth		16	16
depth_multiplier		1	1
optimizer		rms	rms
learning_rate		0,01	0,01
data_augmentation	horizontal_flip	X	X
	random_crop	X	X
	distort_color		
	adjust_brightness		
	adjust_contrast		
	adjust_saturation		
shuffle		true	true
steps (k)		45	40
batch_size		24	24
mAP	20k	0,3214	0,2859
	30k	0,341	0,3384
	último k	0,3546	0,3415
loss_evaluacion	20k	3,596	3,935
	30k	3,378	3,466
	último k	3,326	3,374
loss_entrenamiento	20k	3,59	3,424
	30k	3,135	3,221
	último k	3,033	3,408
velocidad (s)		0,242584	0,25473675

Tabla 4.34. Modelos óptimos de cada arquitectura.

Como se estudió, el ratio de aprendizaje es uno de los parámetros más importantes a modificar, ya que el mínimo cambio que hiciéramos en este, acompañado de si era un ratio lineal o decreciente, provocaba en los resultados unos cambios significativos. Es por eso que se siguió una estrategia de prueba – error para el ratio de aprendizaje, valorando valores tanto grandes como pequeños para poder dar con el valor más adecuado. A esto se le suma la dificultad de que a distintos tamaños de entrada, el mismo ratio de aprendizaje es favorable o no; lo que en este caso, se tienen dos modelos con diferentes tamaños de entrada, y el ratio de aprendizaje es el mismo. Esto se debe a que no son de la misma arquitectura, por lo que no es posible compararlos.

De entre las opciones para el aumento de datos, solamente hemos empleado las que modifican la sin cambiar sus características, como la saturación, brillo o contraste; ya que estas provocaban una disparidad entre las pérdidas, haciendo que el modelo fuera capaz de identificar realmente bien los patrones existentes en el conjunto de validación, pero no fuera capaz de generalizar para imágenes que no conociera. Recordar que la mayor diferenciación de los modelos ocurría durante la etapa 20.000-30.000.

También añadir que, en líneas generales, los entrenamientos del modelo *MobileNet* han llevado un menor tiempo, ya que este tiene una arquitectura más ligera que *Inception*, ya que está preparado para incorporarse en dispositivos móviles.

Se empieza la comparación haciendo alusión al parecido de todos los valores de las métricas en un primer vistazo. En lo que respecta a la precisión media, se observa que el modelo de *Inception* ha tenido una progresión rápida al principio del entreno, pero que ha ido progresando lentamente al final de éste. Sin embargo, el modelo de *MobileNet*, ha tenido una progresión más gradual a lo largo del entrenamiento, y el valor final es un poco menor debido a que se entrenó 5.000 pasos menos que el otro modelo. Sin embargo, haciendo referencia a las imágenes que se han presentado con anterioridad sobre las predicciones del modelo, el modelo de *Inception* era capaz de reconocer objetos que estuvieran más lejanos, mientras que *MobileNet* tenía una deficiencia en esto.

En cuanto a las pérdidas, ambos modelos presentan unas pérdidas similares en torno a 3.3; observándose que el modelo de *Inception* tiene una ligera diferencia de pérdidas; pero es tan ligera que apenas se ha podido notar a la hora de evaluar las predicciones de las imágenes, por lo que se afirma que los dos modelos han sido entrenados correctamente sin *overfitting*.

Para finalizar, se puede observar la pequeña diferencia entre las distintas velocidades de predicción de los modelos, que se recuerda que corresponde a la predicción del tamaño del lote, que en nuestro caso son 24; por lo que ese tiempo corresponde a la predicción de 24 imágenes, consiguiendo ambos una predicción por imagen de apenas 10 milisegundos. Por tanto, en la tecnología que se está intentando implantar, al ser detección en tiempo real, la velocidad y la precisión son los parámetros que más se valoran, así que por tener unos valores mayores en estos dos parámetros; y por el desempeño que ha mostrado anteriormente en la evaluación de las imágenes, el modelo elegido como óptimo para esta aplicación es el modelo 13 de *Inception*.

Si comparamos este modelo con el que realizamos el primer entrenamiento cuyos valores eran pre-determinados, de cara a verificar la optimización realizada, se observa que se ha conseguido aumentar disminuir el *overfitting* que este presentaba, disminuyendo las pérdidas hacia valores cercanos a 3; y obteniendo más de 0.35 mAP en 45.000 pasos; mientras que el otro obtuvo 0.32 mAP en 88.000 pasos.

5. CONCLUSIONES Y FUTUROS TRABAJOS

En este trabajo fin de grado se han optimizado una serie de modelos basados en redes neuronales orientadas a la aplicación para un sistema de seguimiento solar. Se ha estudiado principalmente los modelos SSD los cuales detectan objetos de una sola pasada.

Uno de los objetivos iniciales era la toma de imágenes para aumentar el conjunto de datos ya existente, formado por más de 1.300 imágenes; ya que lo ideal para la tecnología de detección de objetos mediante redes neuronales sería el disponer de aproximadamente 1.000 imágenes por cada clase de objeto que se quiera reconocer. Esta aplicación se ha configurado para 4 clases diferentes, por lo que se precisaría un conjunto de datos de 4.000 imágenes. Este conjunto de datos resultaría en un desempeño de los modelos mucho más acertado, al igual que un mayor costo computacional; sin embargo, la toma de imágenes no se pudo llevar a cabo debido a la situación provocada por el COVID-19. Aun así, gracias a la técnica del aprendizaje por transferencia y con la opción de aumento de datos mediante modificaciones de imágenes ya existentes del conjunto de datos, se han podido obtener unos resultados muy prometedores que pueden ser probados en la instalación real.

Se han estudiado las respuestas de dos arquitecturas basadas en el método SSD, *Inception* y *MobileNet*, mediante aprendizaje por transferencia, las cuales estaban previamente entrenadas en el conjunto de datos COCO. Se han llevado a cabo una serie de entrenamientos para cada una de las arquitecturas, los cuales se han ido basando unos en otros para intentar optimizar lo máximo posible los modelos.

Durante el estudio de los entrenamientos de los distintos modelos, se llegó a la conclusión de que es necesario cambiar los valores predeterminados de configuración, siendo el ratio de aprendizaje el hiperparámetro que mayor importancia tiene para el desempeño del modelo. Además, el tamaño de imagen de entrada tiene también una importancia relevante y jugó un papel muy importante en la optimización del modelo, ya que un tamaño pequeño o un tamaño muy grande degradan los resultados considerablemente. El parámetro de opción de aumento de datos ha sido de vital importancia, y sobre todo para un conjunto de datos limitado como el que se tenía, aunque si se establecen unas opciones que afecten a los atributos de la imagen, como son el brillo o la saturación estos afectan negativamente al rendimiento del modelo, ya que establecen unos cambios demasiado drásticos comparados con lo que el modelo analiza en el conjunto de evaluación. Es por ello que estas opciones se descartaron. También una característica común es que los modelos presentan durante las etapas 20.000 y 30.000 unos cambios que resultan decisivos, por lo que si no muestran durante estas etapas un buen resultado, tendrán unos resultados pobres por mucho tiempo que se entrene.

En general, la mayoría de los modelos entrenados obtuvieron una buena ejecución, con valores de pérdidas muy parecidos, y diferenciándose entre ellos por la precisión media y la velocidad. Los modelos con una imagen de entrada mayor han obtenido una velocidad bastante alta, aunque también afectan a esta variable otros parámetros como es la técnica del *dropout* o la profundidad del modelo. Como la aplicación para la que se busca estos algoritmos es un sistema de seguimiento solar en tiempo real, se necesita un modelo que sea preciso y veloz.

El modelo que mejor cumple con los requisitos ha sido finalmente el correspondiente a la arquitectura *Inception*; con un tamaño de imagen de entrada de 400 x 400; con la técnica *dropout* activada; y con un ratio de aprendizaje de 0.01 con decrecimientos exponenciales cada 3.000 pasos.

Con esto se ha conseguido una precisión media de 0.35, y una velocidad de 0.24 segundos de predicción por lote, y como este lo forman 24 imágenes, se ha conseguido una velocidad de predicción de 10 milisegundos por imagen en el clúster.

Estos resultados son muy buenos en comparación con los que venían dados para el conjunto de datos de COCO, que corresponden a un mAP de 24, y 42 milisegundos en realizar una predicción. Esto se debe a que este conjunto tiene 80 clases en comparación con las 4 que se han estudiado en esta aplicación, por lo que son unos resultados razonables. Además, en comparación con el primer modelo de esta arquitectura que fue entrenado con los valores predeterminados; el modelo optimizado obtiene una precisión media de más de 0.35 mAP a los 45.000 pasos; mientras que el predeterminado, 0.32 mAP a los 88.000. Las pérdidas del modelo optimizado también son menores que las del predeterminado, estando en torno a 3, sin presencia de *overfitting*.

Como se estudió en la sección de validación, el modelo de *Inception* presenta un desempeño realmente bueno, siendo capaz de detectar objetos a gran distancia y encuadrándolos correctamente, como el receptor. Esto es importante, ya que una de las principales pérdidas en un sistema de torre central es el desbordamiento del reflejo de la luz solar debido a la incorrecta calibración; y que el modelo sea capaz de reconocer exactamente el receptor, y con ello la bisectriz entre este y el sol, es fundamental. Sin embargo el modelo de la arquitectura *MobileNet*, presenta fallos en cuanto a objetos que están a larga distancia, ya que es un modelo que tiene dificultad en reconocer objetos de pequeño tamaño.

Personalmente, pienso que la aplicación de la visión artificial a los sistemas de seguidores solares es muy prometedora, dado que esta ha mostrado un comportamiento excelente a pesar de haber dispuesto de un conjunto de datos relativamente pequeño; para lo que la técnica del aprendizaje por transferencia y la opción de aumento de datos ha sido realmente útil.

En cuanto a futuros trabajos, se podría realizar una detección semántica mediante Mask R-CNN, ya que al ser una segmentación semántica, permite al modelo encuadrar perfectamente el receptor y el sol, consiguiendo así un offset mínimo y disminuyendo las pérdidas actuales asociadas a eso; aunque habría que tener en cuenta la velocidad del algoritmo, ya que esta familia de modelos presenta un mayor coste computacional. Por otro lado, de cara a la elección de los parámetros, se podría implementar la optimización bayesiana para encontrar el modelo más óptimo de una forma más automatizada.

REFERENCIAS

- [1] H. Elsheikh, S. W. Sharshir, M. Abd Elaziz, A. E. Kabeel, W. Guilan y Z. Haiou, "Modeling of solar energy systems using artificial neural network: A comprehensive review" *Solar Energy*, vol. 180, pp. 622-639, 2019.
- [2] A. Khosravi, M. Malekan, J. J. G. Pabon, X. Zhao y M. E. H. Assad, "Design parameter modelling of solar power tower system using adaptive neuro-fuzzy inference system optimized with a combination of genetic algorithm and teaching learning-based optimization algorithm" *Journal of Cleaner Production*, vol. 244, p. 118904, 2020.
- [3] V. Masson-Delmotte et al. "Calentamiento global de 1.5°C." IPCC, 2018.
- [4] A. Peinado, A. P. Marugán, y F. P. Márquez. "A Review of the Application Performances of Concentrated Solar Power Systems." *Applied Energy*. Elsevier, 17 Sept. 2019. Web.
- [5] J. Gosens et al. "China's Role in the next Phase of the Energy Transition: Contributions to Global Niche Formation in the Concentrated Solar Power Sector." *Environmental Innovation and Societal Transitions*, Elsevier, 11 Jan. 2020. Disponible en: www.sciencedirect.com/science/article/pii/S2210422420300058.
- [6] E. Thomas, "Machine Learning." *Machine Learning - an Overview | ScienceDirect Topics*. Disponible en: www.sciencedirect.com/topics/computer-science/machine-learning.
- [7] E. Batuecas, C. Mayo, R. Díaz y F. J. Pérez, "Life Cycle Assessment of heat transfer fluids in parabolic trough concentrating solar power technology" *Solar Energy Materials and Solar Cells*, vol. 171, pp. 91-97, 2017.
- [8] M. T. Islam, N. Huda, A. B. Abdullah y R. Saidur, "A comprehensive review of state-of-the-art concentrating solar power (CSP) technologies: Current status and research trends" *Renewable and Sustainable Energy Reviews*, vol. 91, pp. 987-1018, 2018.
- [9] J. Solawetz. "What Is Mean Average Precision (MAP) in Object Detection?" *Roboflow Blog*, 5 Oct. 2020. Disponible en: blog.roboflow.com/mean-average-precision/.
- [10] Centro Extremeño De Tecnologías Avanzadas - CETA-Ciemat. *Ministerio De Ciencia, Innovación y Universidades*, www.ceta-ciemat.es/
- [11] J. G. Gomila. "Curso Online 'Machine learning de la A a la Z: R y Python para Data Science'." *Udemy*. Disponible en: <https://www.udemy.com/course/machinelearning-es/>
- [12] Ciemat. "Noticias." *Noticias CIEMAT*, disponible en: www.ciemat.es/cargarAplicacionNoticias.do;jsessionid=6DA85D56D357BA513EED2A91159619FD?idArea=0.
- [13] International Energy Agency (IEA) "Market report series: Renewables 2018. Analysis and forecast to 2023", 2018.

- [14] M. Berenguel, F. R. Rubio, A. Valverde, P. J. Lara, M. R. Arahal, E. F. Camacho y M. López, "An artificial vision-based control system for automatic heliostat positioning offset correction in a central receiver solar power plant." *Solar Energy*, vol. 76, n° 5, pp. 563-575, 2004.
- [15] CENER - Centro Nacional De Energías Renovables. *CENER*, 30 Nov. 2020. Disponible en: www.cener.com/
- [16] M. J. Martínez Pelayo. "Diseño y optimización del campo solar de un sistema de receptor central con sobrecalentamiento de vapor". Escuela Técnica Superior de Ingenieros de la Universidad de Sevilla. Proyecto Fin de Carrera, 2005.
- [17] A. Z. Hafez, A. M. Yousef y N. M. Harag, "Solar tracking systems: Technologies and trackers drive types – A review" *Renewable and Sustainable Energy Reviews*, vol. 91, pp. 754-782, 2018.
- [18] E. Clark. "A Beginners Guide To Concentrated Solar Power (CSP)." *Medium*, 4 Oct. 2017. Disponible en: medium.com/cleantech-rising/a-beginners-guide-to-concentrated-solar-power-csp-d5e6f9d12cdc.
- [19] J. C. Sattler, M. Röger, P. Schwarzbözl, R. Buck, A. Macke, C. Raeder, and J. Götsche. "Review of Heliostat Calibration and Tracking Control Methods." *Solar Energy*. Pergamon, 30 June 2020. Web. May 2020.
- [20] D. G. Fuentes. "Estudio De Las Diferentes Tipologías De Receptores Volumétricos En Centrales Termosolares." Escuela Técnica Superior de Ingenieros de la Universidad de Sevilla. Proyecto fin de carrera.
- [21] *TensorFlow*. www.tensorflow.org/.
- [22] E. Calvo. *Centrales Termosolares, Operación y Mantenimiento*. Disponible en: www.opex-energy.com/termosolares/funcionamiento_termosolar.html.
- [23] *Planta De Cogeneración*. Disponible en: www.energiza.org/index.php?option=com_k2.
- [24] "Nevada Software S.L." *Solarweb.net*. May. 2020. Disponible en: www.solarweb.net/directorio/empresa/3448/Nevada-Software-S.L.>.
- [25] A. Pfahl, J. Coventry, M. Röger, F. Wolfertstetter, J. F. Vásquez-Arango, F. Gross, M. Arjomandi, P. Schwarzbözl, M. Geiger y P. Liedke, "Progress in heliostat development" *Solar Energy*, vol. 152, pp. 3-37, 2017.
- [26] "CSP Projects Around the World." *SolarPACES*, 16 Dec. 2020. Disponible en: www.solarpaces.org/csp-technologies/csp-projects-around-the-world/.
- [27] IEA (2018), *Renewables 2018*, IEA, Paris. Disponible en: www.iea.org/reports/renewables-2018
- [28] A. Sánchez-González y J. Yellowhair, "Reflections between heliostats: Model to detect alignment errors" *Solar Energy*, vol. 201, pp. 373-386, 2020.
- [29] C. Villasante, J. Mabe, I. Les, A. Peña, M. Sanchez y S. Lopez, "SmartCSP: The industry 4.0 approach for an effective CSP cost reduction"
- [30] O. Behar, A. Khellaf y K. Mohammedi, "A review of studies on central receiver solar thermal power plants" *Renewable and Sustainable Energy Reviews*, vol. 23, pp. 12-39, 2013.

- [31] O. Achkari y A. El Fadar, "Latest developments on TES and CSP technologies – Energy and environmental issues, applications and research trends." *Applied Thermal Engineering*, vol. 167, p. 114806, 2020.
- [32] W. Nsengiyumva, S. G. Chen, L. Hu y X. Chen, "Recent advancements and challenges in Solar Tracking Systems (STS): A review" *Renewable and Sustainable Energy Reviews*, vol. 81, pp. 250-279, 2018.
- [33] J. A. Carballo, J. Bonilla, M. Berenguel, J. Fernández-Reche y G. García, "New approach for solar tracking systems based on computer vision, low cost hardware and deep learning." *Renewable Energy*, vol. 133, pp. 1158-1166, 2019.
- [34] M. Berenguel, F. R. Rubio, A. Valverde, P. J. Lara, M. R. Arahal, E. F. Camacho y M. López, "An artificial vision-based control system for automatic heliostat positioning offset correction in a central receiver solar power plant." *Solar Energy*, vol. 76, n° 5, pp. 563-575, 2004.
- [35] M. Röger, S. Ulmer, et al. "Fast Determination of Heliostat Shape and Orientation by Edge Detection and Photogrammetry." Proc. 14th CSP SolarPACES Symposium 2008.
- [36] S. Meisera, E. Lüpfer, B. Schirckeb, R. Pitz-Paal, et al. "Analysis of Parabolic Trough Concentrator Mirror Shape Accuracy in Different Measurement Setups." *Energy Procedia*, Elsevier, 1 June 2014.
- [37] S. Ulmer et al. "Automated High Resolution Measurement of Heliostat Slope Errors." *Solar Energy*, Pergamon, 1 Feb. 2010.
- [38] K. He, G. Gkioxari, P. Dollár y R. B. Girshick, "Mask R-CNN" *CoRR*, vol. abs/1703.06870, 2017.
- [39] A. K. Yadav y S. S. Chandel, "Solar radiation prediction using Artificial Neural Network techniques: A review." *Renewable and Sustainable Energy Reviews*, vol. 33, pp. 772-781, 2014.
- [41] Y. LeCun y G. Hinton. "Deep Learning." *Nature*, 2015.
- [42] C. Amorim. "Red Neuronal En Python Con Numpy – Parte 1." *Art From Code*, 18 Apr. 2017. Disponible en: artfromcode.wordpress.com/2017/04/18/red-neuronal-en-python-con-numpy-parte-1/.
- [43] J. B. Ahire. "The Artificial Neural Networks Handbook: Part 4." *Medium*, 11 Nov. 2018. Disponible en: medium.com/@jayeshbahire/the-artificial-neural-networks-handbook-part-4-d2087d1f583e.
- [44] J. Segarra et al. "Deep Learning-Based Forecasting of Aggregated CSP Production." *Mathematics and Computers in Simulation*, North-Holland, 25 Feb. 2020.
- [45] D. Faria et al. "A Study on CNN Transfer Learning for Image Classification." 2015.
- [46] J. Wu. "Introduction to Convolutional Neural Networks." *LAMDA Group*, 2017.
- [47] R. Fergus. "Neural Networks". Disponible en: mlss.tuebingen.mpg.de/2015/slides/fergus/Fergus_1.pdf.

- [49] D. Sarkar. "A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning." *Towards Data Science*, 17 Nov. 2018. Disponible en: towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a.
- [50] S. J. Pan y Q. Yang, "A Survey on Transfer Learning," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345-1359, Oct. 2010.
- [51] J. A. Carballo, J. Bonilla, M. Berenguel, J. Fernández-Reche y G. García, "New approach for solar tracking systems based on computer vision, low cost hardware and deep learning." *Renewable Energy*, vol. 133, pp. 1158-1166, 2019.
- [52] J. Brownlee. "A Gentle Introduction to Object Recognition With Deep Learning." *Machine Learning Mastery*, 5 July 2019. Disponible en: machinelearningmastery.com/object-recognition-with-deep-learning/.
- [53] D. Mwit. "A 2019 Guide to Semantic Segmentation." *Heartbeat*, 23 June 2020. Disponible en: heartbeat.fritz.ai/a-2019-guide-to-semantic-segmentation-ca8242f5a7fc.
- [54] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama y K. Murphy, "Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors." de *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [55] "ArcGIS API for Python." *How Single-Shot Detector (SSD) Works? | ArcGIS for Developers*. Disponible en: developers.arcgis.com/python/guide/how-ssd-works/.
- [56] M. Zaforas. "¿Es Python El Lenguaje Del Futuro?" *Paradigma*. Disponible en: www.paradigmadigital.com/dev/es-python-el-lenguaje-del-futuro/.
- [59] J. Zambrano. "¿Aprendizaje Supervisado o No Supervisado?" *Medium*, 31 Mar. 2018. Disponible en: medium.com/@juanzambrano/aprendizaje-supervisado-o-no-supervisado-39ccf1fd6e7b.
- [60] W. Liu, A. Berg, et al. "SSD: Single Shot MultiBox Detector." 2016.
- [61] K. Simonyan. "Very Deep Convolutional Networks for Large-Scale Visual Recognition." *Visual Geometry Group - University of Oxford*. Disponible en: www.robots.ox.ac.uk/~vgg/research/very_deep/.
- [62] E. Forson. "Understanding SSD MultiBox - Real-Time Object Detection In Deep Learning." *Towards Data Science*, 9 June 2019. Disponible en: towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab.
- [63] "Project 6: Deep Learning Introduction to Computer Vision." *Project 6: Deep Learning*. Disponible en: cs.brown.edu/courses/csci1430/2017_Spring/proj6a/.
- [65] "How Bounding Box Annotation Helps Object Detection in Machine Learning: Use Cases." *Medium*, 11 May 2020. Disponible en: medium.com/analytics/how-bounding-box-annotation-helps-object-detection-in-machine-learning-use-cases-431d93e7b25b.
- [66] A. U. Arasanipalai. "A Practical Guide To Hyperparameter Optimization." *AI & Machine Learning Blog*, 5 Aug. 2019. Disponible en: nanonets.com/blog/hyperparameter-optimization/.

- [67] “Hyperparameters: Optimization Methods and Real World Model Management.” *MissingLink.ai*. Disponible en: missinglink.ai/guides/neural-network-concepts/hyperparameters-optimization-methods-and-real-world-model-management/.
- [68] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams and N. de Freitas, "Taking the Human Out of the Loop: A Review of Bayesian Optimization," in *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148-175, Jan. 2016.
- [69] “Get Started with TensorBoard : TensorFlow.” *TensorFlow*. Disponible en: www.tensorflow.org/tensorboard/get_started.
- [70] “What Are L1 and L2 Loss Functions?” *AfterAcademy*. Disponible en: afteracademy.com/blog/what-are-l1-and-l2-loss-functions.
- [71] A. Khazri. “Faster RCNN Object Detection.” *Towards Data Science*, 9 Apr. Disponible en: towardsdatascience.com/faster-rcnn-object-detection-f865e5ed7fc4.
- [72] J. Hui. “SSD Object Detection: Single Shot MultiBox Detector for Real-Time Processing.” *Medium*, 15 Dec. 2020. Disponible en: jonathan-hui.medium.com/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06.
- [73] M. Fromm, J. Mcdermid, et al. “Cromated Detection of Conifer Seedlings in Drone Imagery Using Convolutional Neural Networks. Remote Sensing.” 2017.
- [74] “Qué Es Overfitting y Underfitting y Cómo Solucionarlo.” *Aprende Machine Learning*, 29 Feb. 2020. Disponible en: www.aprendemachinelarning.com/que-es-overfitting-y-underfitting-y-como-solucionarlo
- [75] M. Alberti. “Deep Neural Networks an Introduction.” *DeepLearningItalia*, 5 June 2018. Disponible en: www.deeplearningitalia.com/guia-para-arquitecturas-de-redes-profundas/.
- [76] A. Nasirahmadi, et al. “Deep Learning and Machine Vision Approaches for Posture Detection of Individual Pigs.” *Sensors (Basel, Switzerland)*, MDPI, 29 Aug. 2019.
- [77] “Comparing MobileNet Models in TensorFlow.” *KDnuggets*. Disponible en: www.kdnuggets.com/2019/03/comparing-mobilenet-models-tensorflow.html.
- [78] “Common Objects in Context.” *COCO*, cocodataset.org/.
- [79] R. J. Tan “Breaking down Mean Average Precision (MAP).” *Towards Data Science*, 6 July 2020. Disponible en: towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52.
- [80] Tensorflow. “Tensorflow/Models.” *GitHub*, 28 July 2020. Disponible en: github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf1_detection_zoo.md.
- [81] “The ethics of Self-driving cars.” *Medium*. Disponible en: towardsdatascience.com/the-ethics-of-self-driving-cars-efaaaaf9e320.
- [82] J. D.Polo. “Google Presenta MobileNets, Modelos Para Que Los Móviles Usen Sistemas De Reconocimiento De Imágenes.” *WWWhat's New*, 15 June 2017. Disponible en: www.whatsnew.com/2017/06/15/google-presenta-mobilenets-modelos-para-que-los-moviles-usen-sistemas-de-reconocimiento-de-imagenes/.

- [83] M. Oršić, et al. “In Defense of Pre-Trained ImageNet Architectures for Real-Time Semantic Segmentation of Road-Driving Images.” *ArXiv.org*, 20 Mar. 2019.
- [84] “Model Optimization : TensorFlow Lite.” *TensorFlow*. Disponible en: www.tensorflow.org/lite/performance/model_optimization.
- [85] J. Nelson. “You Might Be Resizing Your Images Incorrectly.” *Roboflow Blog*, 20 July 2020. Disponible en: blog.roboflow.com/you-might-be-resizing-your-images-incorrectly/.
- [86] Tensorflow. “Tensorflow/Models.” *GitHub*. Disponible en: github.com/tensorflow/models/blob/master/research/object_detection/core/anchor_generator.py
- [87] Tensorflow. “Tensorflow/Models.” *GitHub*. Disponible en: github.com/tensorflow/models/blob/master/research/object_detection/core/regularizer.py
- [88] Tensorflow. “Tensorflow/Models.” *GitHub*. Disponible en: github.com/tensorflow/models/blob/0f332b026632267847c863308b5c55c49ba7ccdf/research/object_detection/protos/post_processing.proto.
- [89] J.Nelson. “You Might Be Resizing Your Images Incorrectly.” *Roboflow Blog*, 20 July 2020. Disponible en: blog.roboflow.com/you-might-be-resizing-your-images-incorrectly/.
- [90] P. Radhakrishnan. “What Are Hyperparameters ? and How to Tune the Hyperparameters in a Deep Neural Network?” *Towards Data Science*, 18 Oct. 2017. Disponible en: towardsdatascience.com/what-are-hyperparameters-and-how-to-tune-the-hyperparameters-in-a-deep-neural-network-d0604917584a.
- [91] P. Gupta. “Regularization in Machine Learning” *Towards Data Science*, 16 Nov. 2017. Disponible en: towardsdatascience.com/regularization-in-machine-learning-76441ddcf99a.
- [92] *Anchor Boxes for Object Detection - MATLAB & Simulink*. Disponible en: www.mathworks.com/help/vision/ug/anchor-boxes-for-object-detection.html.
- [93] P. Radhakrishnan. “What Are Hyperparameters ? and How to Tune the Hyperparameters in a Deep Neural Network?” *Towards Data Science*, 18 Oct. 2017. Disponible en: towardsdatascience.com/what-are-hyperparameters-and-how-to-tune-the-hyperparameters-in-a-deep-neural-network-d0604917584a.
- [94] Tensorflow. “Tensorflow/Models.” *GitHub*. Disponible en: github.com/tensorflow/models/blob/1af55e018eebce03fb61bba9959a04672536107d/research/object_detection/protos/matcher.proto.
- [95] L. Wei, A. Berg, et al. “SSD: Single Shot MultiBox Detector.” *Computer Vision – ECCV*. Lecture Notes in Computer Science, vol 9905. Springer, Cham. 2016.

El objetivo fundamental de este Trabajo Fin de Grado (TFG) consiste en la aplicación de técnicas de aprendizaje automático y visión artificial a la calibración de heliostatos. Con este fin, se ha desglosado cada área del trabajo y se ha realizado un estudio teórico previo para conocer cómo pueden llegar a funcionar todas estas áreas juntas. A continuación, se ha pasado a la configuración y optimización de distintos modelos para comprobar cuál de ellos tiene un mejor rendimiento para la aplicación a estudiar, los sistemas de seguimiento solar.

El trabajo nace como respuesta al problema de calibración de los heliostatos. Estos requieren una calibración distinta unos de otros, al estar situados en diferentes posiciones con respecto al receptor; y se busca el desarrollo de un algoritmo de calibración que sea eficaz y preciso, de este modo, el error que se cometa por la planta de heliostatos será menor, y la eficiencia de esta, mayor. Para ello, se entrenarán modelos basados en redes neuronales los cuales aprenderán a reconocer los objetos que están en los alrededores de los heliostatos de la planta CESA, en la Plataforma Solar de Almería (PSA); y de este modo, el algoritmo será capaz de predecir la posición del heliostato óptima para que refleje una mayor de radiación solar al receptor.

The main aim of this thesis consists in the application of machine learning and computer vision techniques to heliostat calibration. For this purpose, every area of interest has been theoretically studied in order to learn how well these areas can work together in practice. Then, it has been done fine-tuning to some models to optimize them as good as possible, and check which one of them presents a better overall efficiency in the solar system tracking.

This thesis arise from the heliostat calibration problem. Each one of them requires a different calibration, since they are in different positions regarding the receiver. It is looked for the development of an effective and accurate calibration algorithm, in that way, the whole heliostat plant offset will decrease, and efficiency will improve. To this effect, neural networks models will be trained, capable of detecting objects around heliostats in CESA plant, in Plataforma Solar de Almería (PSA); and these algorithms will be able to predict the optimum heliostat position for reflecting a larger solar radiation into the receiver.

