Universidad de Oviedo

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN.

MÁSTER UNIVERSITARIO EN INGENIERÍA INFORMÁTICA

ÁREA DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES

TRABAJO FIN DE MÁSTER

CLASIFICACIÓN DE EMISIONES ATMOSFÉRICAS PROCESANDO IMÁGENES CON REDES NEURONALES CONVOLUCIONALES

D.ª FERNÁNDEZ MORENO, Marta
TUTOR: D. GARCÍA MARTÍNEZ, Daniel Fernando

FECHA: 06 de febrero de 2019

Índice

1 Introducción

A lo largo de los últimos años el impacto medioambiental se ha tornado en un aspecto de preocupación constante con objeto de minimizar todos aquellos riesgos que conlleven acciones perjudiciales para el medio ambiente mediante un aumento de medidas en pro a la gestión y control de estos posibles impactos.

El presente proyecto nace del interés de la detección temprana de emisiones en el ambiente, conllevando todas aquellas tareas precisas para la identificación y el análisis de humos emitidos diferenciando entre emisiones de contaminación y vapor de agua.

Para ello se estudiará la aplicación de tecnologías dentro del ámbito del aprendizaje profundo o *deep learning,* llevando a cabo la implementación de una red neuronal convolucional (CNN) encargada de, a partir de un procesamiento de imágenes de humos, obtener una clasificación fiable sobre la existencia de emisiones en el ambiente.

Este problema puede ser resuelto mediante una diversa variedad de paradigmas y tecnologías. No obstante, este proyecto debe su origen al continuo auge del uso de técnicas basadas en aprendizaje automático (*machine learning)* y el alto rendimiento obtenido por estos sistemas de clasificación.

Con objeto de llevar a cabo este proyecto se ha hecho uso de imágenes procedentes de una cámara ubicadas ante una planta siderúrgica. La principal funcionalidad de esta cámara se trata de tomar imágenes de los humos emitidos para su posterior procesamiento, conformando por tanto la base de datos precisa para realizar las fases de entrenamiento, validación y testing de la CNN.

2 Conocimientos iniciales

Con el propósito de cumplir con los objetivos propuestos en la sección anterior, previamente se deberá efectuar un estudio y una debida comprensión de la situación inicial de la que se parte.

2.1 Sistema actual

En la actualidad esta compañía siderúrgica cuenta con un sistema de detección de emisiones compuesto por distintos subsistemas dedicados a la detección de estas de forma independiente mediante diversas técnicas (p. ej. infrarrojos, distancias, movimiento o color).

Por tanto, mediante una concatenación de los resultados obtenidos por cada subsistema se puede determinar la presencia de emisiones de modo que, por ejemplo, en caso de que el sistema de color detecte una emisión, el resto de los subsistemas proporcionarán aquella información necesaria para determinar si la detección es o no correcta en base a distintos criterios.

Debido al buen rendimiento de las redes neuronales en tareas de clasificación, detección y segmentación de objetos, se pretende explorar la posibilidad de implementar un nuevo subsistema que haga uso de esta tecnología, con objeto de aportar una mayor robustez al sistema actual.

2.2 Aprendizaje profundo y redes neuronales

El aprendizaje profundo se trata de una técnica perteneciente al campo del aprendizaje automático que hace uso de una amplia diversidad de ejemplos con objeto de permitir que un computador aprenda de forma similar al patrón de aprendizaje de las personas, obteniendo como resultado la información deseada a partir de ejemplos como van a ser en este proyecto las imágenes de emisiones.

Este ámbito se basa en el uso de redes neuronales consistentes en modelos computacionales conformados por un conjunto de neuronas artificiales interrelacionadas. Cada neurona se trata de un modelo lineal seguido por una no-linealidad en la que cada una de las entradas es multiplicada por un peso.

Estas redes hacen uso del método de propagación hacia atrás o *backpropagation* que tiene como principal objetivo que la red aprenda la asociación existente entre patrones de entrada y sus respectivas salidas. Este algoritmo modifica los pesos mediante la técnica del gradiente descendiente minimizando el error en base al ajuste de los pesos en cada una de las capas.

Las redes neuronales profundas se tratan de redes cuya dimensión de capas puede llegar a ser relativamente alta, pudiendo variar desde dos o tres capas (característica de una red neuronal tradicional) hasta un número de ciento cincuenta capas.

En este proyecto se hará uso de las denominadas redes neuronales convolucionales (CNN), ya que se trata de un problema de procesamiento de imágenes que consta de una dimensión demasiado grande para otras redes neuronales como las totalmente conectadas, que requerirían una gran cantidad de parámetros de entrenamiento, además de no tener en cuenta la relación entre vecinos a la hora de estudiar relaciones entre los píxeles de las imágenes. Por ejemplo, en caso de una imagen RGB de dimensiones 128x128, una única neurona de una red neuronal totalmente conectada haría uso de 49.152 pesos.

En la *¡Error! No se encuentra el origen de la referencia.Figura 1.* Ejemplo de red totalmente conectada se puede apreciar la arquitectura de una red neuronal totalmente conectada compuesta por dos

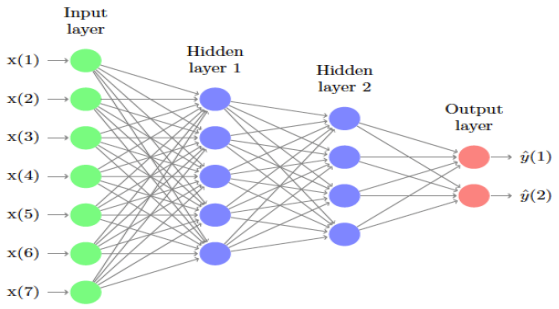

Figura 1. Ejemplo de red totalmente conectada

capas ocultas, cuya entrada consiste en siete datos y cuya salida consiste en dos valores.

Como comparación en la *Figura 2* se muestra una posible arquitectura de una red convolucional cuya entrada se trata de una imagen y consta de diversos tipos de capas a detallar en profundidad posteriormente en este documento.

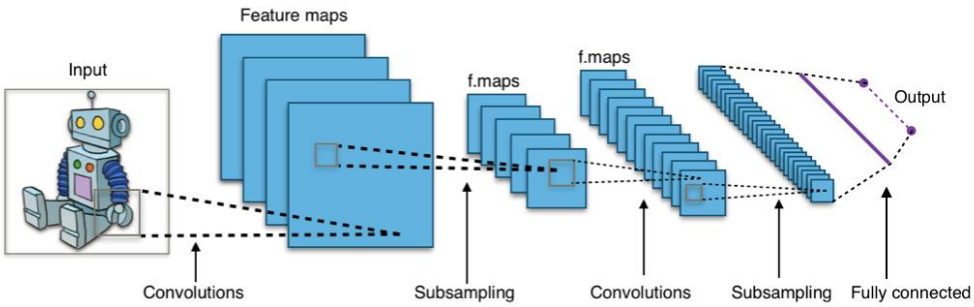

Figura 2. Ejemplo de arquitectura de red convolucional

Las CNN reciben imágenes como entradas interpretadas como múltiples arrays (lo que proporciona una reducción de parámetros de la red y una mayor eficiencia) a los que, tras realizar la operación de producto escalar, aplica una función de activación.

Las imágenes son estructuradas en un agrupamiento de neuronas formando un volumen 3D por cada subsección, consistente en el largo y ancho de la imagen, además de la profundidad según la escala de colores de esta. En este proyecto se hará uso de imágenes en color (RGB) sin incluir el canal alfa, por lo que el volumen de entrada de la red neuronal contará con tres unidades de profundidad. Estos volúmenes 3D consistirán en entradas de cada una de las capas de la red, siendo el volumen de entrada de una capa, la salida resultante de la capa inmediatamente anterior. En la *Figura 3* se muestra un ejemplo de estos volúmenes 3D manipulados por las distintas capas de una CNN.

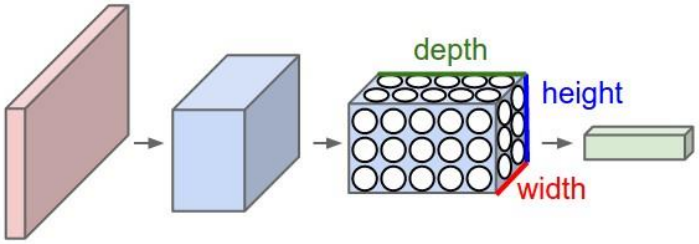

Figura 3. Ejemplo de volúmenes manipulados por una CNN

2.2.1 Principales características de las CNN

Además de contar con un mayor número de capas que las redes neuronales convencionales totalmente conectadas y estar orientadas al tratamiento de imágenes, las CNN cuentan con

otra serie de características que se deberán conocer para tener una correcta comprensión del funcionamiento de estas redes. En base a distintos estudios se pueden destacar una serie de características principales de este tipo de red neuronal [Ciresan, Meier, Masci, Gambardella, and Schmidhuber 2011; LeCun, Bengio, and Hinton 2015; Quintero 2018; Luna-González 2016].

- Pesos compartidos: Los pesos asignados a las neuronas se tratan de matrices n-dimensionales tratados como filtros de imagen, a diferencia de los pesos escalares de las redes neuronales tradicionales.
- Conexiones locales: A diferencia de las redes totalmente conectadas, no todas las neuronas están conectadas entre capas, por lo que estas estarán conectadas únicamente con aquellas que aporten información sobre la región de la imagen a tratar por cada neurona.
- Pooling: Consiste en la reducción de características del volumen manipulado por la red mediante la selección de aquellas que se consideran de mayor peso o importancia.

2.2.2 Capas de una CNN

Este tipo de redes neuronales, tal y como se indicó previamente, cuentan con una gran cantidad de capas. Estas capas se pueden dividir en tres grandes tipos distintos en base a la principal funcionalidad de la capa, así como las operaciones que esta realiza. A continuación, la *Figura 4* muestra un ejemplo ilustrativo de un posible modelo de capas de una CNN.

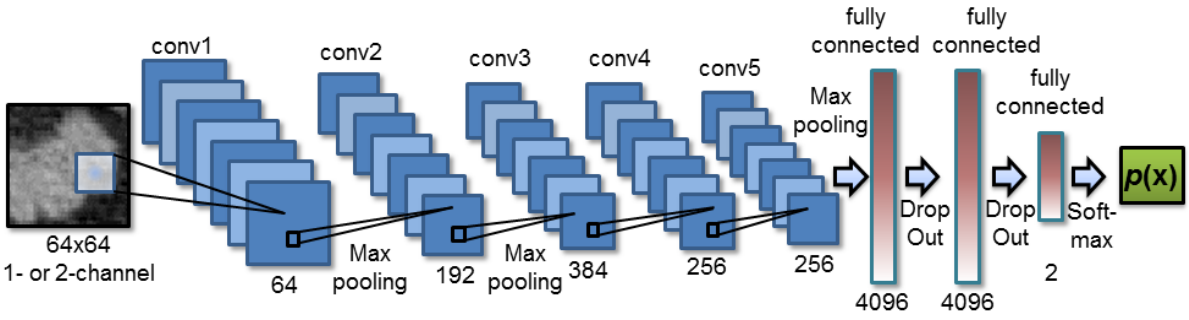

Figura 4. Ejemplo de estructura de capas de una CNN

2.2.2.1 Capas convolucionales

Estas capas toman su nombre de la operación que realizan (denominada convolución) consistente en la multiplicación de matrices. Estas se tratan de la matriz de entrada

resultante de la imagen a tratar y de la matriz respectiva al filtro o kernel de dimensión fija que se aplicará en la convolución.

Estas capas devuelven como salida el resultado obtenido de calcular el producto vectorial entre la matriz del filtro y la matriz de la subregión, sumándole posteriormente el valor a aprender de sesgo (único para cada filtro). Esta operación se realiza desplazándose a lo largo de la matriz de la imagen, obteniendo como resultado un mapa de características de la imagen de entrada en forma de volumen 3D cuya profundidad corresponde al número de filtros aplicados en dicha capa. Un ejemplo ilustrativo de esta operación se encuentra en la *Figura 5*.

En esta capa se tendrán que tener en cuenta los siguientes hiperparámetros principales:

- Tamaño del kernel o filtro a aplicar en la operación de convolución.
- Número de filtros a aplicar en la capa.
- Tamaño de paso, referente a cuantos píxeles se desplaza el filtro en cada movimiento dentro de la matriz de la imagen.

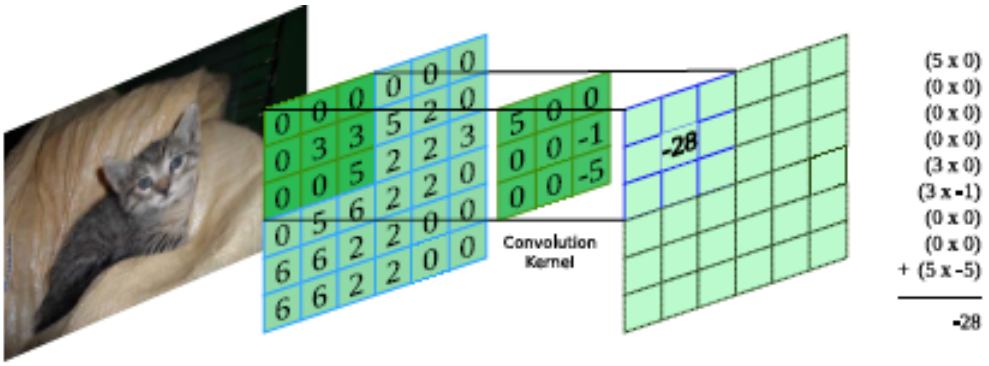

Figura 5. Ejemplo de operación de convolución con tamaño de kernel 3x3

2.2.2.2 Capas de pooling

Su principal función consiste en la extracción de aquella información considerada de mayor relevancia tras llevar a cabo operaciones de extracción de características mediante las capas convolucionales.

Esto permitirá a la red eliminar información irrelevante, permitiendo contar con una complejidad menor que reduzca los costes computacionales.

Para llevar a cabo esta operación se hace uso de un kernel de tamaño fijo que consiste en regiones del volumen de entrada que será reducido a un único valor mediante la elección

del valor más alto (*max pooling*) o del valor promedio (*average pooling*), tal y como se puede observar en la *Figura 6*.

Los dos hiperparámetros de esta capa por tanto consistirán en los siguientes:

- Tamaño del kernel o región a hacer pooling.
- Tamaño de paso.

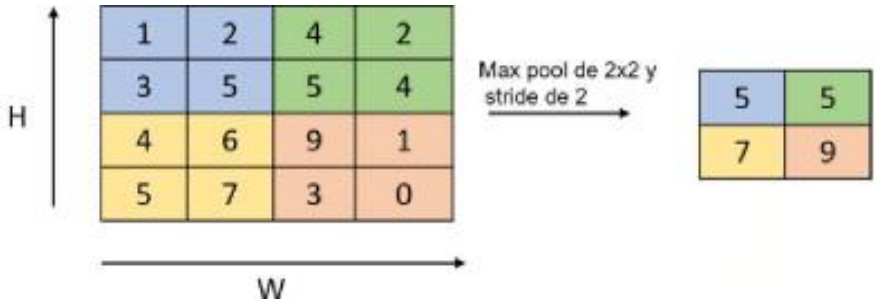

Figura 6. Ejemplo de operación max pooling con kernel de tamaño 2x2 y tamaño de paso 2

2.2.2.3 Capas totalmente conectadas

Tras llevar a cabo todas las operaciones de convolución y pooling pertinentes, se procederá a la obtención de la clasificación final mediante capas totalmente conectadas en base a las características obtenidas previamente. Por tanto, la salida final consistirá en tantos valores como clases existentes en el problema. En la *Figura 7* se muestra una arquitectura de una CNN compuesta por dos capas totalmente conectadas al final.

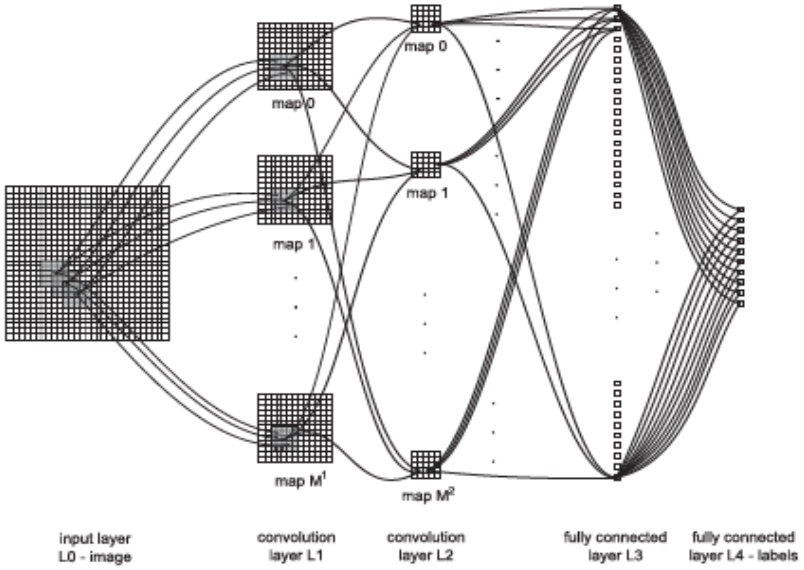

Figura 7. Ejemplo de capas totalmente conectadas en CNNs

2.2.3 Soluciones que aportan las CNN

Mediante la implementación de una gran variedad de modelos de CNN se pueden llevar a cabo soluciones a una gran diversidad de problemas. Dentro del ámbito que concierne a este proyecto las principales tareas a efectuar por una CNN se deben diferenciar de forma clara con objeto de comprender bien qué solución se está buscando. Estas tareas se tratan de las descritas a continuación:

- **Clasificación:** Permite identificar imágenes en base a objetos, en caso de este proyecto consistiría en indicar si en la imagen hay o no una emisión.
- **Segmentación semántica:** Permite determinar la ubicación de todos los objetos de esa clase en la imagen. Para este proyecto se mostraría la ubicación de todas las emisiones sin diferenciar entre ellas, en caso de haber más de una, o provenir de distintas zonas.
- **Detección de objetos:** Muestra cada uno de los objetos de la clase diferenciándolos, pero no presta atención a la superposición espacial de estos. Las emisiones por tanto se ubicarían en la imagen, pero sin indicar superposición espacial entre distintas emisiones.
- **Segmentación de instancias:** Establece qué píxeles pertenecen a cada instancia de esa clase. Por tanto, mostraría de forma segmentada cada emisión, superponiendo aquella instancia que se encuentra más cerca de la cámara en caso de solapamiento.

2.2.4 Análisis de movimiento

Hasta el momento se ha detallado la extracción de características espaciales mediante CNNs cuyas entradas consisten en volúmenes 2D (imágenes), sin embargo, dependiendo del problema a resolver, puede resultar interesante efectuar un análisis de características temporales pudiendo obtener información sobre el movimiento de los objetos de las distintas imágenes que recibe la red como entrada. Para ello la red deberá contener un volumen de entrada 3D que contenga una nueva dimensión para el canal temporal.

Con objeto de dar solución a este paradigma mediante el uso de CNNs se podrán llevar a cabo dos principales enfoques.

El primer enfoque consiste en una arquitectura constituida por dos CNNs idénticas de modo que cada una obtiene una clasificación atendiendo a características temporales o espaciales de forma independiente. La red que analiza características espaciales recibe como entrada las imágenes originales del conjunto, mientras que la encargada de analizar el movimiento

se basará en imágenes preprocesadas que muestren el *optical flow* de cada imagen [Simonyan and Zisserman 2014].

Por otro lado, el segundo enfoque no requiere ninguna etapa previa de preprocesamiento de datos, haciendo uso de un tipo de red denominada 3D CNN. Esta red difiere de la CNN convencional en la que los kernel utilizados pasan a ser volúmenes 3D en vez de matrices, obteniendo información de entrada de la misma región en diversas imágenes consecutivas en el tiempo. En la *Figura 8.* Comparación entre 2D CNN y 3D CNN se puede observar claramente la diferencia entre ambas CNN [Lima 2017].

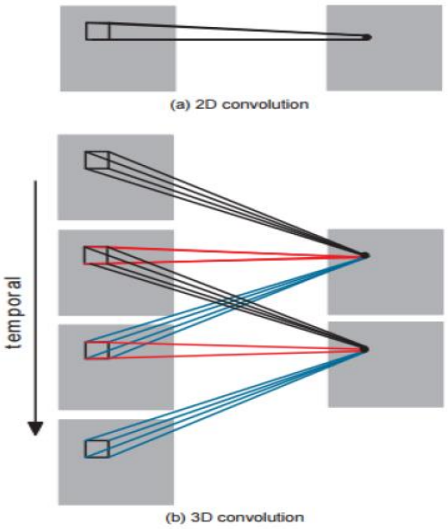

Figura 8. Comparación entre 2D CNN y 3D CNN

2.2.5 Clasificación mediante detección de objetos

Otra estrategia para mejorar el rendimiento a la hora de establecer una clasificación por medio de CNNs, además del análisis de movimiento, consiste en llevar a cabo una fase previa de detección de objetos, eliminando así características no relevantes a la hora de clasificar, además de reducir el tamaño del problema y por consecuencia su complejidad computacional.

Para llevar a cabo esta metodología se pueden implementar diversas estrategias por medio de otros tipos de CNNs y/o algoritmos convencionales.

2.2.5.1 Proposición de regiones

Consiste en la aplicación de algoritmos clásicos para la proposición de regiones en las que se encuentran posibles objetos dentro de una imagen.
A continuación, se muestran tres algoritmos de proposición de regiones basados en distintas estrategias de detección.

Búsqueda selectiva o *selective search*: Consiste en la división de la imagen en regiones que serán comparadas de forma posterior, combinándolas en una única región en caso de similitud entre ellas. Esto da como resultado la detección de objetos por medio de segmentación [Uijlings 2013] tal y como se muestra en la *Figura 9*.

Figura 9. Ejemplo de resultados de selective search

Binarized Normed Gradients (BING): Consiste en la aplicación de la norma a los gradientes de la imagen tras el redimensionado de estas a un tamaño pequeño (p. ej. 8x8). Esta operación consigue detectar características distintivas en la imagen con mayor facilidad tal y como se muestra en la *Figura 10* [Cheng, Zhang, Lin, and Torr 2014].

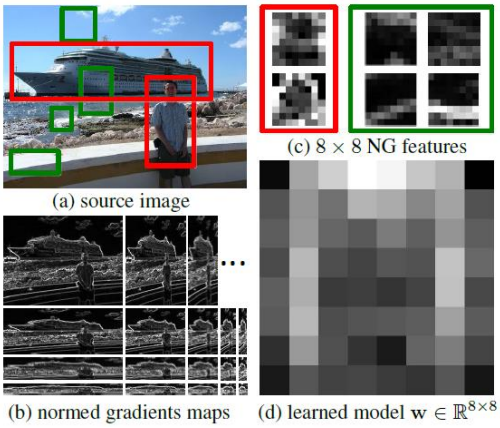

Figura 10. Ejemplo de resultados de BING

Edge Boxes: Se basa en la detección de objetos en base a la cantidad de contornos identificados en las distintas regiones de la imagen [Zitnick 2014]. En la *Figura 11* se muestran cinco ejemplos de detección mediante este agoritmo.

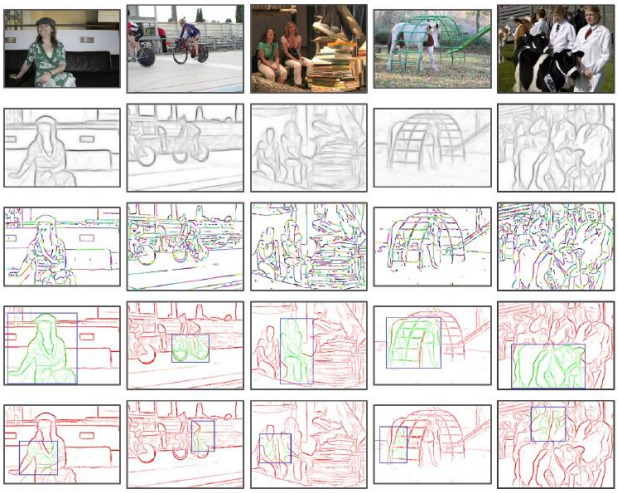

Figura 11. Ejemplo de resultados de Edge Boxes

2.2.5.2 R-CNN, Fast R-CNN y Faster R-CNN

Las regiones identificadas por los algoritmos pueden utilizarse como entradas de una posterior CNN encargada de su clasificación. Este modelo que consta de algoritmos de búsqueda de regiones y una CNN convencional que toma dichas regiones, recibe el nombre de **R-CNN** [Girshick, Donahue, Darrell, and Malik 2013]. En la *Figura 12* se muestra la arquitectura completa de un R-CNN.

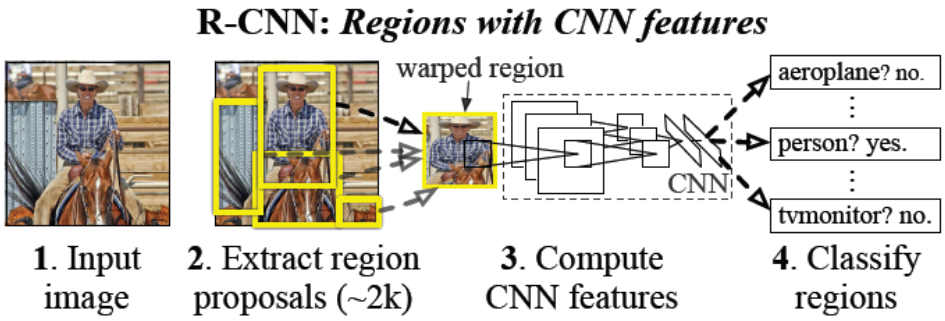

Figura 12. Ejemplo de arquitectura de R-CNN

Estos algoritmos de búsqueda que forman parte de la R-CNN proponen una cantidad de regiones en torno a 2.000, lo que requiere mucho tiempo de ejecución, es por ello por lo que el propio autor de este modelo de red propuso posteriormente una modificación a esta denominándola **Fast R-CNN** [Girshick 2015].

Esta modificación consiste en proporcionar como entrada las imágenes originales en lugar de las regiones generadas por los algoritmos de proposición de regiones. Estas regiones se obtendrán haciendo uso del algoritmo de búsqueda sobre el mapa de características resultante de la convolución y serán la entrada a las capas totalmente conectadas de la CNN. En la *Figura 13* se detalla la conexión entre el algoritmo de búsqueda de regiones y la CNN.

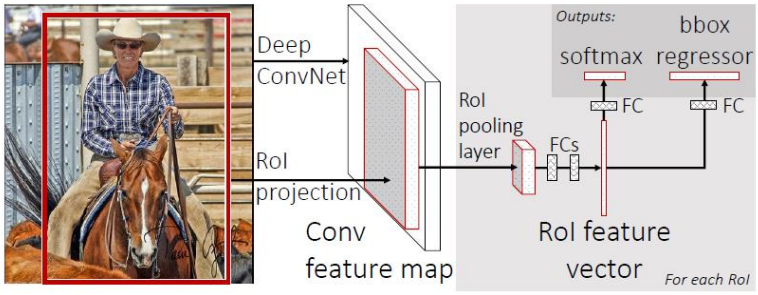

Figura 13. Ejemplo de arquitectura de Fast R-CNN

El rendimiento de la **Fast R-CNN** mejora de forma notable el de la R-CNN debido a la eliminación de las 2000 regiones iniciales propuestas por el algoritmo. Sin embargo, esta R-

CNN sigue haciendo uso de un algoritmo de búsqueda, lo que consume una gran cantidad de tiempo de ejecución.

Debido a esto, Shaoquing Ren propone un nuevo algoritmo [Ren, He, Girshick, and Sun 2015] que permite eliminar la búsqueda selectiva manteniendo el aprendizaje de regiones de la CNN. Para ello se proporcionan las imágenes originales como entrada a la CNN tal y como se hace en la Fast R-CNN y se lleva a cabo la tarea de predicción de regiones por medio de una red neuronal independiente cuyas salidas serán utilizadas para llevar a cabo la clasificación final por parte de la CNN principal, tal y como se puede observar en la *Figura 14*.

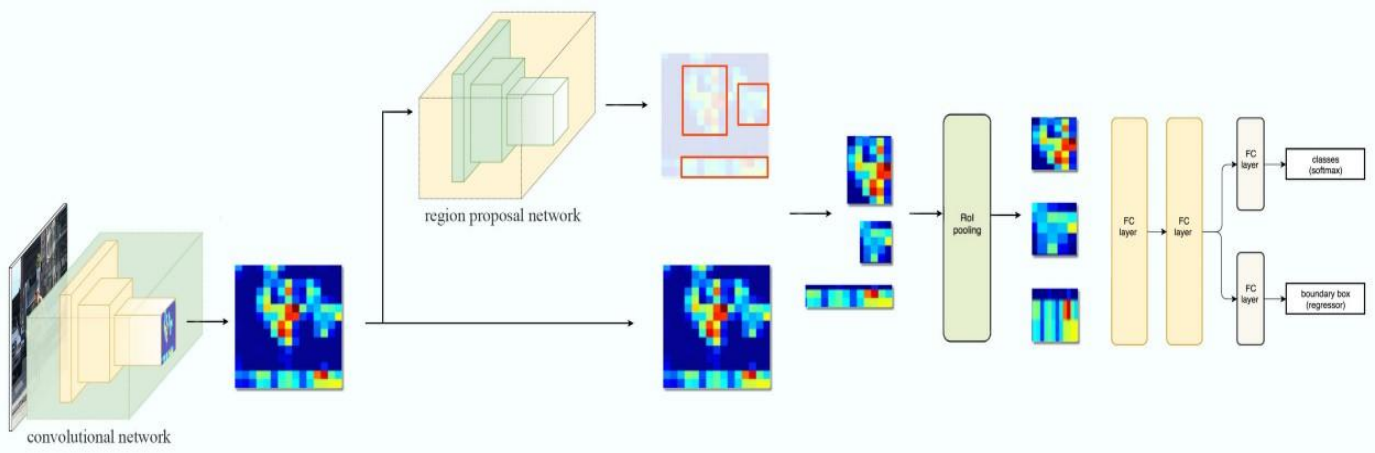

Figura 14. Ejemplo de arquitectura de Faster R-CNN

2.2.6 Segmentación de instancias

Kaiming He propone en [He, Gkioxari, Dollár, and Girshick 2017] una modificación de la Faster R-CNN [Ren, He, Girshick, and Sun 2015] orientada a la segmentación de instancias en lugar de a la detección de objetos mediante la creación de *boundary boxes*.

En este modelo de red se pueden identificar dos fases distintas identificables en la *Figura 15* y descritas a continuación:

1. La generación de áreas con alta probabilidad de contener el objeto (RoI).
2. La clasificación de los objetos identificados y generación de máscaras.

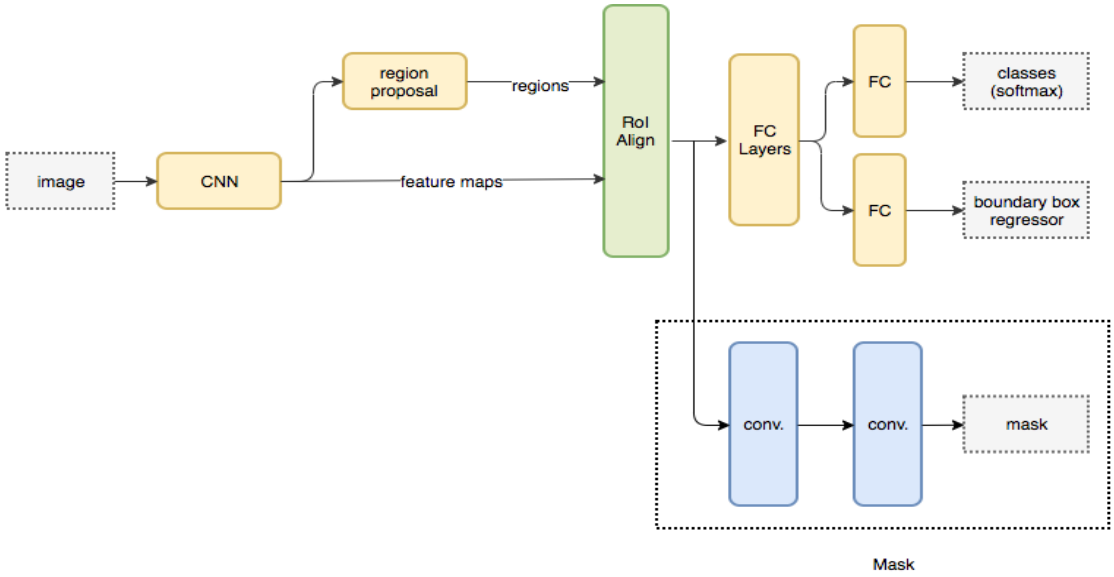

Figura 15. Ejemplo arquitectura de Mask R-CNN

Para ello se desarrolla una arquitectura de red neuronal compuesta por una Faster R-CNN encargada de detectar el objeto y crear las *boundary boxes*, seguida por una posterior FCN (Fully Convolutional Netork) encargada de la asignación de píxeles en los que se encuentra el objeto dentro de la *boundary box*, obteniendo así la segmentación de la instancia. Esta FCN seguiría una arquitectura como la mostrada a en la *Figura 16.* Ejemplo de FCN dentro de una Mask R-CNN.

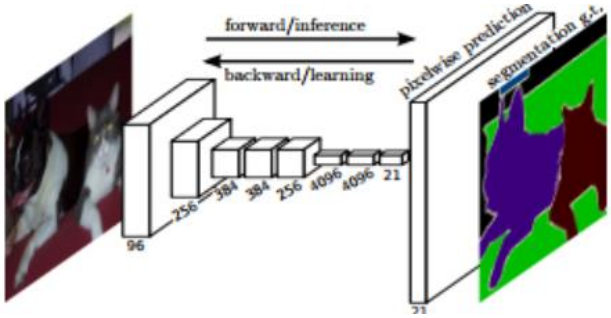

Figura 16. Ejemplo de FCN dentro de una Mask R-CNN

2.2.7 Modelo de Inception

Este modelo de red es introducido por Google como propuesta ganadora en la competición ImageNet de clasificación y detección en imágenes en 2014 bajo el nombre de GoogLeNet.

Se trata de una arquitectura de CNN conformada por uno o más módulos denominados Inception que se traducen en la aplicación de diversos filtros a una misma entrada mediante distintas operaciones de convolución con kernels de distinto tamaño. Estas redes, por tanto,

debido a la gran cantidad de capas y filtros a aplicar conllevan una gran cantidad de parámetros a aprender y una mayor complejidad computacional.

Con objeto de no incrementar demasiado esta complejidad computacional se hace uso de convoluciones con kernels de tamaño 1x1 para reducir la profundidad del volumen que servirá de entrada a otras operaciones de convolución, además de realizar operaciones de pooling para reducir el volumen de datos [Szegedy 2016]. La arquitectura resultante se puede observar en la *Figura 17.* Ejemplo de reducción de dimensión de volúmenes en un módulo Inception .

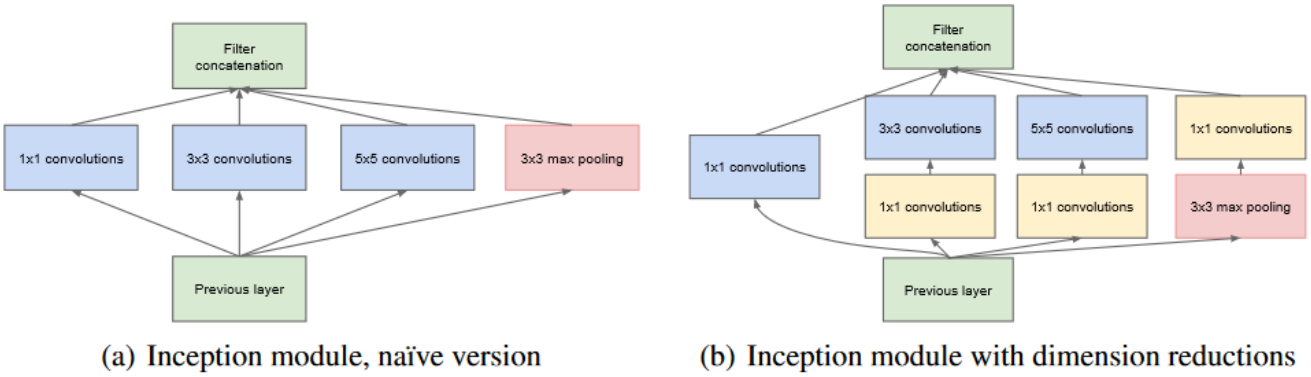

Figura 17. Ejemplo de reducción de dimensión de volúmenes en un módulo Inception

Prosiguiendo con este modelo de red, los mismos creadores de GoogLeNet propusieron en [Szegedy 2016] una propuesta para la mejora del rendimiento de estas redes.

Esta idea se basa en sustituir operaciones convolucionales con tamaños de kernel considerablemente grandes como 5x5 en dos operaciones independientes consecutivas con filtros de tamaño 3x3, tal y como se muestra en la *Figura 18.* Esto se debe a que una operación de convolución con un kernel de tamaño 5x5 requiere 25 parámetros por cada filtro, mientras que en el caso de dos operaciones con dimensiones 3x3 se requieren 18 parámetros por filtro.

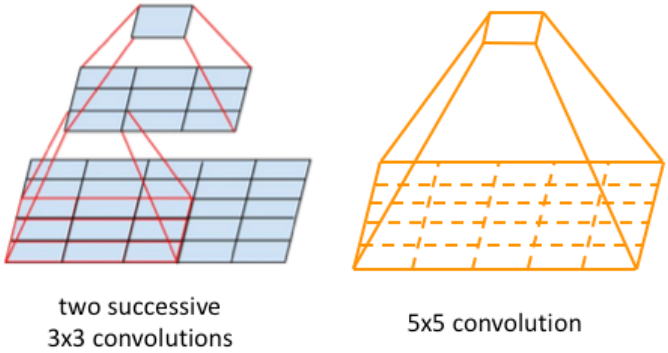

Figura 18. Ejemplo de reducción de complejidad computacional en operaciones convolucionales

2.3 Evaluación de clasificadores

En el presente apartado se exponen de forma meramente teórica las metodologías y métricas a emplear para llevar a cabo la evaluación de un **clasificador binario** de forma adecuada.

Durante este proceso se hará uso de las muestras disponibles en el conjunto de imágenes con objeto de evaluar la capacidad de clasificación del modelo de red implementado, dividiéndolas en subconjuntos destinados al aprendizaje y a la evaluación.

2.3.1 Métodos de evaluación

Tal y como se indicó previamente, se deberá dividir el conjunto de imágenes en subconjuntos de modo que no se emplee el mismo ejemplo para ambas tareas de aprendizaje y evaluación.

Para ello se pueden seguir distintos métodos según se divida dicho conjunto y se empleen los subconjuntos resultantes durante las fases de aprendizaje y evaluación. Los métodos más destacados son los siguientes:

- **Hold-out.** Consiste en dividir el conjunto de forma indistinta en muestras de entrenamiento y muestras de evaluación en base a un porcentaje de volumen de muestras para cada subconjuto. En cuanto al procedimiento de aprendizaje y evaluación, se llevará a cabo *k* veces de modo que el resultado final consistirá en el promedio de los resultados de cada iteración.
- **Validación cruzada o cross-valiation.** Se trata de la división de *m* muestras en *k* lotes de tamaño *m/k*. En cuanto a la fase de entrenamiento y evaluación, se llevarán a cabo *k* iteraciones en las que un único lote será utilizado como conjunto de evaluación y el resto están destinados al entrenamiento, de modo que en cada iteración se utilice un lote de evaluación distinto. El resultado final se obtendrá en base al promedio del resultado de las *k* iteraciones.

En definitiva, el método de validación cruzada asegura que todas las muestras están siendo utilizadas tanto para entrenar como para evaluar, a diferencia de hold-out, en el que no se puede garantizar que esto ocurra. Sin embargo, la validación cruzada conlleva más coste de computación ya que se debe ejecutar *k* veces de forma obligatoria, mientras que hod-out se puede realizar las veces que se crea oportuno.

2.3.2 Métricas de evaluación

Una vez comprendida la metodología a seguir durante el proceso de evaluación, se deberán interpretar los resultados obtenido con objeto de establecer finalmente un juicio crítico sobre los resultados obtenidos para el modelo. Para ello existen una serie de métricas que, en base a los ratios de ejemplos de cada clase etiquetados de forma correcta o incorrecta, se define la capacidad de clasificación del modelo.

Se establecen, por tanto, los conceptos de valor real *v* que hace referencia a la clase correspondiente a la muestra, mientras que el valor predicho *v'* se trata de la clasificación resultante por parte del modelo a evaluar.

2.3.2.1 Matriz de confusión

Una forma de establecer de forma ilustrativa una comparación sobre el valor real y predicho adoptado para cada muestra se trata de la matriz de confusión, que muestra la precisión del clasificador en formato tabular.

Esta herramienta determina qué número de ejemplos se han determinado de forma correcta, conllevando así que el valor real y el predicho coincidan, así como determinar cuántos han sido asignados a otras clases de forma errónea. De este modo, el resultado de la evaluación de cada una de las muestras se encontrará dentro de los siguientes casos:

1. **True Positive o TP.** Se da cuando el valor de *v* y *v'* es positivo y coincide.
2. **True Negative o TN.** Se da cuando el valor de *v* y *v'* es negativo y coinciden.
3. **False Positive o FP.** Se da cuando el valor de *v* es negativo y el de *v'* positivo.
4. **False Negative o FN.** Se da cuando el valor de *v* es positivo y el de *v'* negativo.

La matriz resultante por ejemplo se representaría tal y como se muestra en la *Figura 19.*

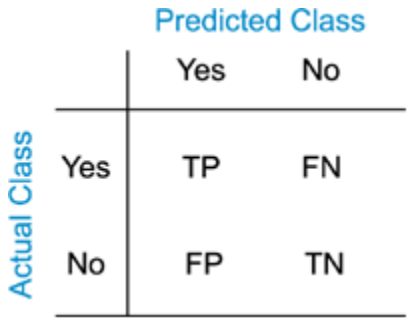

Figura 19. Ejemplo de matriz de confusión binaria

2.3.2.2 Precisión (PPV)

La precisión o Positive Predictive Value (PPV), se basa en los valores TP y FP obtenidos en la evaluación, de modo que se establece la proporción de muestras clasificadas como positivas correctamente frente al número de muestras clasificadas como positivas.

$$PPV = \frac{TP}{TP + FP}$$

2.3.2.3 Negative Predictive Value (NPV)

Se trata de la equivalencia a la métrica anterior para los ejemplos negativos.

$$NPV = \frac{TN}{TN + FN}$$

2.3.2.4 Recall (TPR)

Esta métrica, también llamada True Positive Rate (TPR), hace referencia a la proporción de muestras clasificadas como positivas correctamente frente al total de muestras positivas.

$$TPR = \frac{TP}{TP + FN}$$

2.3.2.5 Specificity (TNR)

Esta métrica, también llamada True Negative Rate, sigue la metodología de la anterior, pero obteniendo las muestras negativas clasificadas de forma correcta respecto al total de muestras negativas.

$$TNR = \frac{TN}{TN + FP}$$

2.3.2.6 Fall-out (FPR)

La métrica fall-out o False Positive Rate (FPR) consiste en la proporción de muestras negativas clasificadas de forma errónea como positivas.

$$FPR = \frac{FP}{FP + TN}$$

2.3.2.7 F1-Score

Se trata de la combinación de las métricas precisión y recall, de modo que establece un índice de rendimiento del clasificador más completo y sencillo a la hora de establecer comparaciones.

$$F1 = 2 \cdot \frac{PPV \cdot TPR}{PPV + TPR}$$

2.3.2.8 Accuracy

Representa la proporción de muestras clasificadas de forma correcta frente al número total de muestras del conjunto.

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP}$$

2.3.2.9 Receiver operating chracteristic (ROC)

Se trata de una gráfica utilizada para ilustrar la evaluación de un clasificador, mediante la representación gráfica de la comparación de la métrica recall o TPR frente a la métrica fall-out o FPR, de modo que cada predicción de los ejemplos se representa como un punto de la curva.

Esta curva representa a su vez el espacio ROC consistente en el área comprendida entre el valor de FPR (en el eje x) y de TPR (en el eje y), de modo que cuanto más arriba a la izquierda se encuentre la curva ROC, mejor será la predicción, ya que indicaría que el ratio de TPR es muy superior al de FPR.

En la *Figura 20* se muestra un ejemplo de la curva ROC resultante de la evaluación de un modelo.

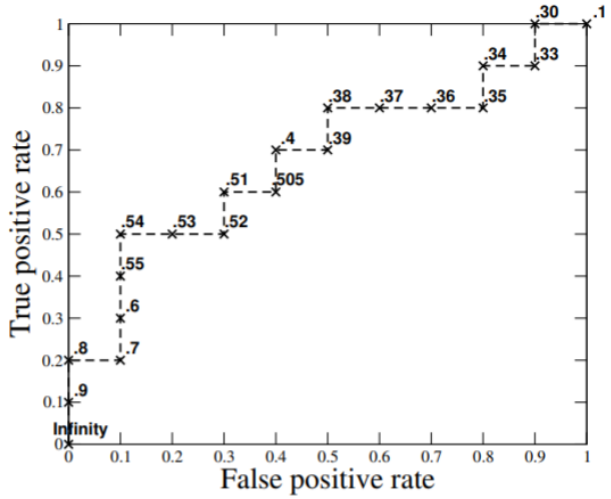

Figura 20. Ejemplo de curva ROC

2.3.2.10 Area under the ROC curve (AUC)

Esta métrica representa el área encerrada, entre el eje x y la curva ROC de la evaluación de un modelo, dentro del espacio ROC. AUC toma valores dentro del rango [0,1], donde un valor de 0.5 correspondería con una clasificación aleatoria por parte del modelo, mientras que un AUC equivalente a 1 corresponder a una clasificación sin FP y FN [Fawcett 2004] .

3 Estudio del estado de la técnica

En primer lugar, de forma previa al desarrollo de la implementación de la solución al problema previamente planteado, se efectuará la tarea de investigación pertinente sobre el estado de la técnica, verificando la existencia de sistemas previos ya existentes o similares al que se plantea, así como el estudio de las distintas técnicas, metodologías o tecnologías empleadas para su resolución, con objeto de obtener un juicio crítico sobre qué técnica o conjunto de ellas se empleará en la implementación final elegida y por qué.

Para ello se recopilará toda aquella información referente al estudio en profundidad realizado sobre el estado del arte sobre la identificación, detección y clasificación de emisiones en el ambiente. Sin embargo, debido a que dentro del ámbito de emisiones no se ha obtenido información alguna relevante en cuanto a los objetivos del proyecto, la tarea de investigación se enfocó en la identificación y detección de humos. Esto se debe a que este problema ofrece una mayor variedad de literatura y soluciones, sobre todo dentro de la identificación de humos con objeto de detectar posibles fuegos de forma temprana. Además, en el sistema actual existente se ha identificado una tasa de fallos relativa a la confusión de humos con nubes, por lo que explorar este campo podrá obtener como resultado una solución a este problema actual.

En cuanto a las soluciones propuestas para la detección de humos, estas se pueden dividir en dos grandes grupos dependiendo de si la información a tratar son imágenes independientes o una secuencia de frames de vídeo. Dentro de cada uno de estos grupos existen diversas implementaciones como solución al problema por medio de técnicas de visión por computador, procesamiento de señales y/o redes neuronales. Estas técnicas pueden ser utilizadas de forma única o conjunta

No obstante, la gran mayoría de estas soluciones pertenecen al segundo grupo, enfocándose en las características de textura, movimiento, energía, frecuencia, color y forma. Esto permite disponer de una gran variedad de características a tener en cuenta a la hora de llevar a cabo la identificación de humos.

Por ejemplo, una implementación existente en el estudio [Krstinić, Stipaničev, and Jakovčević 2009] sobre segmentación de humos mediante imágenes como entrada, hace uso de histogramas clasificando las imágenes píxel a píxel en las clases “humo” y “no humo”.

Mientras tanto, implementaciones cuyo conjunto de datos se trata de vídeos o imágenes consecutivas en el tiempo, como la descrita en [Jakov and Krstini 2010], estudian las características dinámicas del humo, o las características propias de su textura como proponen en los estudios [Maruta 2009; Piccinini, Calderara, and Cucchiara 2008].

En el artículo [Piccinini, Calderara, and Cucchiara 2008] también se propone un modelo de color propio de humos utilizado en conjunto con matrices de co-ocurrecia de niveles de grises (GLCM) y transformadas de wavelet entre otras técnicas y de forma detallada en [Fang 2008].

A su vez [Maruta 2009] hace uso de clasificadores probabilísticos y máquinas de vectores de soporte (SVMs) orientados a la detección de humos en espacios abiertos a una distancia intermedia.

En cuanto a soluciones orientadas a las redes neuronales [Genovese 2011] propone una solución que hace uso de estas en conjunto con un análisis de desplazamiento, color y formas.

En los siguientes apartados se presentan en mayor detalle una serie de soluciones descritas en la literatura consideradas de mayor interés dentro del ámbito del proyecto, que implementan una solución al problema a partir de una combinación de técnicas.

3.1 Técnicas basadas en algoritmos de visión por computador y/o procesamiento de señales

Son varias las implementaciones que han sido desarrolladas por medio de técnicas de visión y procesamiento de señales con objeto de llevar a cabo una solución óptima para problemas de detección de humos. Actualmente, estas técnicas se utilizan ocasionalmente en conjunto con redes neuronales de modo que se obtiene como resultado un sistema más robusto, por lo que se considera necesario adquirir el conocimiento relativo a estas técnicas dentro del ámbito a tratar.

3.1.1 Reglas de decisión cromática y características dinámicas

Tal y como se indicó anteriormente, las soluciones suelen estar más orientadas a la detección en videos que en imágenes debido a la información que proporciona el movimiento del humo. Sin embargo, a su vez, la obtención de estas características de

movimiento puede llevarse a cabo por medio de gran diversidad de técnicas, combinadas a su vez con técnicas de extracción de características espaciales.

Un ejemplo de estas técnicas es el presentado en el estudio [Chen, Yin, Huang, and Ye 2006], que lleva a cabo un análisis de características físicas o estáticas del humo mediante la identificación de grises en la imagen, lo que implica que el valor cada canal R, G y B en un mismo pixel sea el mismo o similar. Junto con la intensidad (**I)** de este valor correspondiente al color gris basándose en un modelo HSI, y los valores L_1, L_2, D_1 y D_2 prefijados como rango de grises se establecen las siguientes reglas de decisión que permitirán identificar humos en una imagen.

1. Regla 1: $R \pm \alpha = G \pm \alpha = B \pm \alpha$
2. Regla 2: $L_1 \leq I \leq L_2$
3. Regla 3: $D_1 \leq I \leq D_2$

Donde el valor α toma valores dentro del rango [15, 20] y el resto corresponden a D_1=80, D_2=150, L_1=150 y L_2=220. De este modo se clasifica el píxel como humo en caso de que se cumpla la condición:

IF (regla 1) AND [(regla 2) OR (regla 3)] = TRUE

Por otro lado, el estudio realiza un análisis de características dinámicas teniendo en cuenta el número de píxeles clasificados como humo, estableciendo así su forma y su ratio de crecimiento y desplazamiento a lo largo de las imágenes. En este caso se establece el parámetro SEP como la suma de áreas clasificadas como humo y segmentadas mediante las reglas de decisión cromática, STP como el número de píxeles clasificados como humo y el valor STD hace referencia al cambio de forma que puede sufrir el humo dentro de una secuencia de imágenes, permitiendo diferenciar el humo de otros objetos con características dinámicas distintas. Mediante estos parámetros se puede determinar si hay o no humo cuando se cumpla la siguiente condición:

$$\text{IF } (\text{SEP}/\text{STD}) \geq STD$$

Con objeto de determinar la tasa de crecimiento del humo en base a su velocidad y desplazamiento, se obtiene la derivada en el tiempo del área clasificada como humo y se comprueba la veracidad de este valor comprobando que esté en el rango [D_1, D_2] y N_d

establece el número mínimo de comprobaciones de si realmente hay o no humo en la imagen. Esta comprobación se define mediante la siguiente regla de decisión:

$$\text{IF } Num(D_1 < \Delta A_{di} < D_2) > N_d$$

Por tanto, teniendo en cuenta las operaciones descritas hasta el momento, se puede definir el flujograma consistente en la clasificación de humos por parte del sistema en la *Figura 21.*

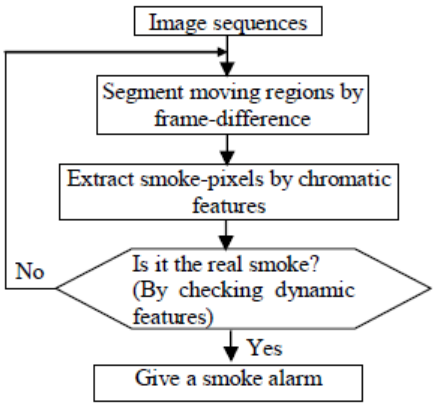

Figura 21. Flujograma del sistema propuesto

Finalmente se evalúa el sistema haciendo uso de un conjunto de vídeos que muestran tanto humos como otros objetos en movimiento. En base a los resultados obtenidos se comprobó que la extracción de características cromáticas da una mayor tasa de falsos positivos clasificando como humo, como se puede observar en la *Figura 22.* Resultados obtenidos por el sistema, un hombre vestido con prendas de tonos grisáceos. Mientras tanto, la extracción de características dinámicas no genera esta clasificación errónea, ya que detecta la diferencia de movimiento de ambos objetos.

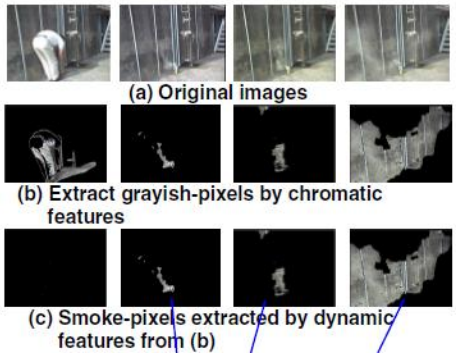

Figura 22. Resultados obtenidos por el sistema

3.1.2 SAKBOT, wavelet y probabilidad bayesiana

Otra metodología existente consiste en la detección de humos mediante la extracción del fondo de la imagen, la comparación con modelos de color y la detección de movimiento mediante transformadas de Wavelet y descrita en [Piccinini, Calderara, and Cucchiara 2008]. Este sistema está compuesto por diversos módulos interconectados cuyo principal objetivo es detectar el objeto en movimiento que posteriormente será analizado para comprobar si se trata o no de humo. En la *Figura 23* se muestra un esquema básico sobre la estructura de este sistema.

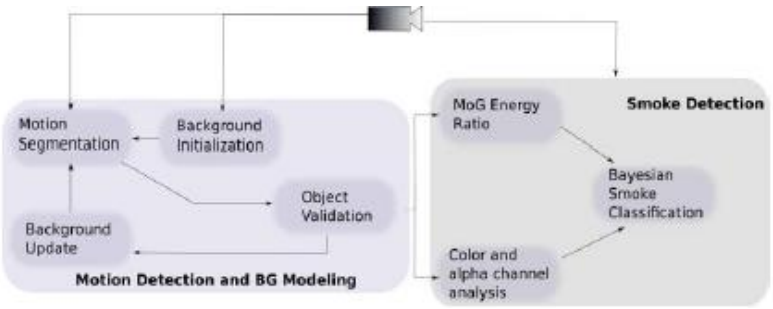

Figura 23. Arquitectura sistema propuesto

Para ello inicialmente se deberá llevar a cabo la detección de objetos en movimiento en la imagen basándose en el método denominado **SAKBOT** (Statistical And Knowledge-Bases Object Traker) descrito en [Cucchiara, Grana, Piccardi, and Prati 2001] consistente en un algoritmo robusto orientado a la **estimación de fondos** de alta complejidad. Este método obtiene el fondo de la imagen mediante el cómputo de aquellos píxeles que se mantienen sin movimiento.

De forma posterior se procede a la detección del humo dentro de la sección de la imagen que no forma parte del fondo mediante el llamado ***análisis de energía*** en el que se evalúan las características de forma y textura por medio de las **transformadas de wavelet** (tal y como se implementó en los estudios [Toreyin, Dedeoglu, and Cetin 2006; Toreyin and Dedeog˘lu 2005]), además de llevar a cabo un análisis de color comparando el color de la imagen con **modelos de color** de humos utilizados como referencia.

Por último, se lleva a cabo la clasificación de los objetos en vista de detectar humo por medio del método de **probabilidad bayesiana**, obteniendo el valor de similitud de un bloque con una región con humos mediante la media de los valores obtenidos por medio del análisis de energía y el análisis de color. Una región es clasificada finalmente cuando el 70% del área que esta conforma está constituida por bloques con probabilidad de ser humo.

Para evaluar este sistema se ha hecho uso del conjunto de imágenes VSSN'06 consistente en cien vídeos tanto del interior de edificios como del exterior mostrando diverdos objetos en movimiento (el humo entre otros).
Los resultados obtenidos se incluyen en la *Figura 24*, comparando los resultados obtenidos para este sistema con lo de la implementación de un modelo de extracción de fondo basado en modelos de mezclas gaussianas(MoGs) tal y como se describe en [Stauffer 1999].

	MOG		SAK	
	Recall	Precision	Recall	Precision
Energy	94%	96%	100%	95%
Color	92%	92%	100%	92%
Total	**94%**	**98%**	**100%**	**99%**

Figura 24. Comparación de resultados de un modelo MoG y el modelo del artículo

Tras evaluar los resultados se llegó a la conclusión de que el sistema MoG genera una alta proporción de falsos positivos debido a la detección de sombras, a diferencia del sistema actual, que elimina las sombras durante la fase de extracción del fondo.

También se puede comprobar que el análisis de características de movimiento obtiene mejores resultados que un análisis de características físicas como puede ser el color.

3.1.3 Extracción del flujo óptico

Una técnica ampliamente utilizada en el procesamiento de imágenes y en identificación de objetos, sobre todo a la hora de analizar características temporales de las imágenes, es el cálculo del flujo óptico.

Dentro de la detección de humos [Altun and Celenk 2013] propone un método que hace uso del flujo óptico obtenido en base a una secuencia de vídeo I(x, y, t).

Tras obtener el gradiente de dicha secuencia, se aplica un filtro 3D Gaussiano como operación de suavizado. Posteriormente se multiplica la matriz resultante de dicha operación por su transpuesta, obteniendo así una matriz simétrica de tensores T como la siguiente.

$$T = \overline{\nabla I} \cdot \overline{\nabla I}' = \begin{bmatrix} t_1 & t_4 & t_5 \\ t_4 & t_2 & t_6 \\ t_5 & t_6 & t_3 \end{bmatrix}$$

Una vez obtenida dicha matriz, se obtienen los valores de velocidad operando sus elementos tal y como se indica en las siguientes ecuaciones.

$$v_x = \frac{t_6 t_4 - t_5 t_2}{t_1 t_2 - t_4^2} \qquad v_y = \frac{t_5 t_4 - t_6 t_1}{t_1 t_2 - t_4^2}$$

Finalmente se aplica un filtro 2D de suavizado al vector de velocidades resultante, obteniendo así las componentes *x* e *y* del flujo óptico de la secuencia de imágenes de entrada. En la *Figura 25* se puede observar el flujo óptico obtenido a partir de dos frames de vídeo.

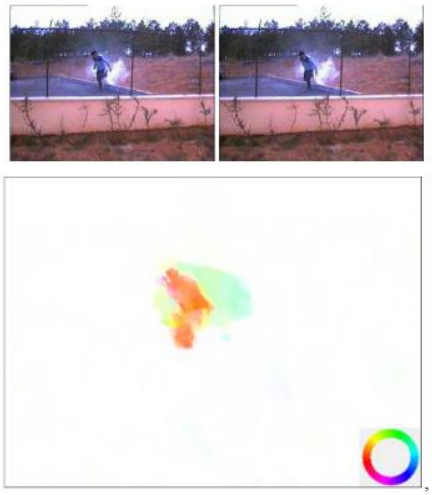

Figura 25. Flujo óptico resultante

A este resultado se le aplica un filtro de color empleando una máscara que permita descartar el flujo óptico de objetos que no son humos, mostrando el flujo óptico de aquellas áreas en las que pueda haber humos. Este filtro permitirá identificar aquella área cuya concentración de humo es mayor, correspondiente al punto en el que se origina el humo.

Una vez se ha obtenido el vector resultante del flujo óptico y el área en el que la probabilidad de que haya humo es alta, se emplea el algoritmo de Green para confirmar la presencia del humo. Este teorema permitirá detectar las áreas en las que se ha disipado el humo a partir del área identificada como punto origen por el filtro de color, mediante el cálculo de la divergencia del humo.

Este valor de divergencia se obtiene del vector ($\vec{V} = \overrightarrow{v_x} + \overrightarrow{v_y} = M\vec{\imath} + N\vec{\jmath}$) correspondiente al flujo óptico siguiendo la siguiente ecuación:

$$\nabla \cdot \vec{V} = \frac{\partial M}{\partial x} + \frac{\partial N}{\partial y}$$

Basándose en el teorema de Green, por tanto, para obtener el valor de divergencia del humo en la imagen se deberá emplear la siguiente ecuación:

$$\oint_C \vec{V} \cdot \vec{n}\, ds = \iint_R \left[\frac{\partial M}{\partial x} + \frac{\partial N}{\partial y}\right] dx\, dy$$

Este sistema es evaluado con vídeos que presentan tanto humos como otros objetos en movimiento mostrando los resultados de cada método, así como el resultado de usarlos en conjunto, demostrando que mediante la combinación de estos métodos se reduce de forma considerable la tasa de falsos negativos, obteniendo (para los vídeos utilizados en la evaluación) un sistema de alta precisión. En la ilustración X se muestra uno de los resultados del sistema, en el que el cuadro verde hacer referencia al área detectada por el filtro de color y la zona trazada por medio de las líneas verdes se trata de la segmentación del humo mediante el teorema de Green.

Figura 26. Ejemplo de resultado del sistema propuesto

Por otro lado [Bogush and Brovko 2011] implementa un algoritmo que sigue el flujograma presentado en la *Figura 27.* Flujograma del sistema.

Para ello divide todas las imágenes del conjunto en bloques de tamaño 8x8 o 4x4 píxeles dependiendo del tamaño de la imagen y los utiliza para obtener el flujo óptico mediante el cómputo de similitudes de un bloque $b_{x,y}$ del frame anterior, con los bloques $b_{x-1, y-1}$, $b_{x, y-1}$ y $b_{x+1,y-1}$.

Como resultado se obtiene una máscara binaria de detección de movimiento de cada frame, a la que se aplica de forma posterior una máscara de contraste para facilitar la tarea de clasificación, permitiendo detectar el humo en la imagen.

Finalmente se elimina el ruido mediante la búsqueda de contornos obteniendo resultados como los mostrados en la *Figura 28*.

Figura 28. Eliminación del ruido

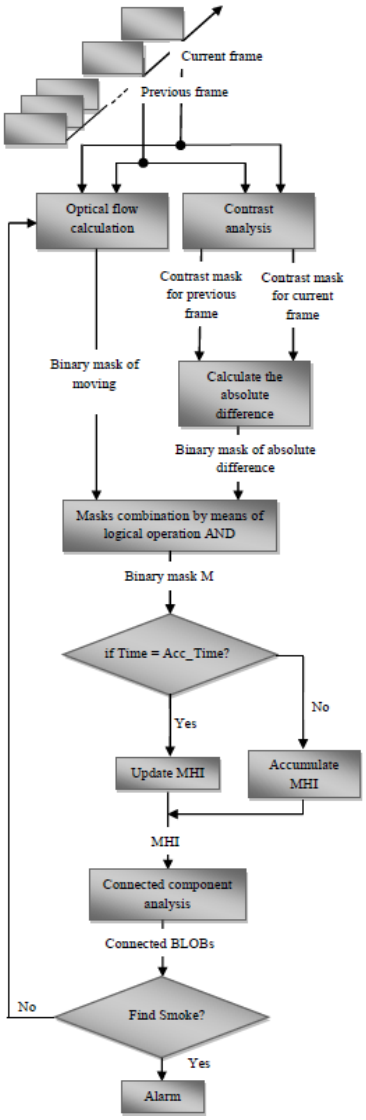

Figura 27. Flujograma del sistema

Este sistema se evaluó mediante el conjunto de imágenes que se encuentra disponible en el enlace: *http://signal.ee.bilkent.edu.tr/VisiFire/Demo/SampleClips.html*.

Obteniendo una baja tasa de falsos positivos, sin embargo, su rendimiento no es alto ante humos muy disipados o de pequeño tamaño.

3.1.4 Extracción de fondo, DoG y k-means

El estudio [Hsu, Dille, Sargent, and Nourbakhsh 2016] describe un sistema de detección de emisiones en el ambiente dentro del ámbito industrial (objetivo a alcanzar en el presente proyecto) mediante el empleo de una gran diversidad de técnicas en conjunto. Para ello hace uso de un conjunto de imágenes formados por 9700 frames de video durante el día, descartando todos aquellos tomados durante la noche.

El primer paso será el preprocesamieto de las imágenes, reduciéndolas a ¼ de su tamaño original, además de **extraer el fondo** de la imagen por medio de la aplicación de la **mediana** sobre las sesenta imágenes anteriores y finalmente convertir las imágenes de enteros de 8 bits a dobles de precisión dentro del rango [0, 1].

El segundo paso consistirá en la detección de movimiento mediante el cómputo de cambios en señales de alta frecuencia (como texturas o bordes) e intensidad de valores (como los colores), aplicando la **diferencia Gaussiana (DoG)** entre otras técnicas para obtener información de la imagen mediante la comparación de esta con el fondo.

El siguiente paso consiste en la segmentación de la imagen basándose en las distintas texturas mediante operaciones de **convolución y el algoritmo k-means++**, con objeto de eliminar pequeñas regiones comúnmente consistentes en el fondo de la imagen, eliminando el ruido tal y como se puede observar en la *Figura 29*:

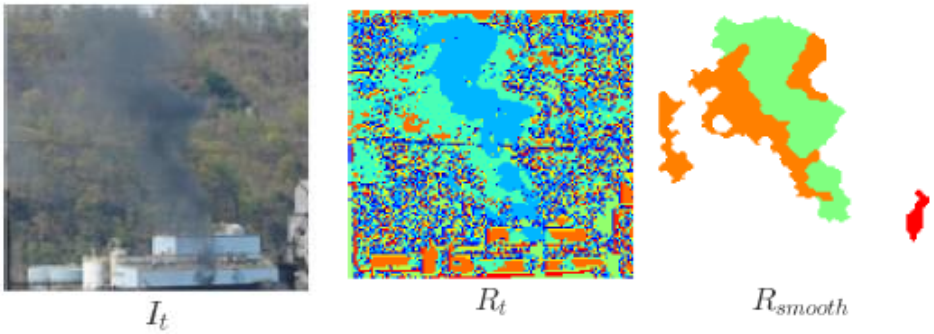

Figura 29. Ejemplo k-means++ y suavizado

A continuación, se lleva a cabo la fase de filtrado de regiones, en la que se comprueba para cada región la similitud con el comportamiento del humo. Para ello se toma como base una imagen resultante de la **saturación de la imagen** original, con el fin de distinguir el objeto a detectar por medio de la aplicación de distintos métodos basados en criterios de iluminación y tonalidades de las regiones.

Finalmente, mediante una fase de detección de eventos, se pretenderá establecer en qué momento comenzó y terminó la emisión detectada.

Este sistema cuenta con una gran cantidad de falsos positivos debido a la confusión de emisiones con sombras cuyo movimiento es más rápido de lo habitual. También obtiene una alta proporción de falsos negativos en caso de emisiones de pequeña dimensión y/o baja opacidad. Es por ello por lo que su precisión no es especialmente alta.

3.2 Técnicas basadas en redes neuronales convolucionales

Una vez se ha llevado a cabo la investigación y comprensión sobre las diversas técnicas más utilizadas dentro del ámbito del proyecto y adquirido todos los conocimientos necesarios relativos a las CNN, se podrá proceder a estudiar distintas soluciones mediante redes neuronales similares a la que se pretende llevar a cabo ya que, hasta la fecha, no se ha encontrado soluciones existentes para la detección y clasificación de emisiones en el ambiente por medio de redes neuronales artificiales.

Se presentarán distintos tipos de soluciones dentro de los siguientes enfoques:

1. Clasificación de imágenes usando características espaciales.
2. Clasificación de imágenes en base a características temporales y espaciales.
3. Detección de objetos mediante segmentación en regiones.

3.2.1 Clasificación de imágenes usando características espaciales

Este enfoque se basa en la idea de clasificar cada imagen de forma independiente a las demás mediante el procesado de todos y cada uno de sus píxeles.

A continuación, se presentan una serie de soluciones descritas en la literatura consideradas de mayor interés dentro de este tipo de clasificación mediante redes neuronales.

3.2.1.1 Detección de humos basada en una CNN

En el estudio [Tao, Zhang, and Wang 2016] se propone una solución a la detección de humos en base al desarrollo de una CNN con la arquitectura presentada en la Tabla 1.

Capa	Entrada	Características
1	3 (224x224)	**Conv1:** 96 (11x11) S=(4x4) Padding + ReLU
	96 (55x55)	**MaxPool1:** (3x3) S=(2x2)
	96 (27x27)	**BatchNorm1:** Momentum=0.9
2	96 (27x27)	**Conv2:** 256 (5x5) S=(2x2) Padding + ReLU
	256 (27x27)	**MaxPool2:** (3x3) S=(2x2) + ReLU
	256 (13x13)	**BatchNorm2:** Momentum=0.9
3	256 (13x13)	**Conv3:** 384 (3x3) S=(2x2) Padding + ReLU
	384 (13x13)	**Conv4:** 192 (3x3) S=(1x1) Padding + ReLU
	192 (13x13)	**Conv5:** 192 (3x3) S=(1x1) Padding + ReLU
	192 (13x13)	**MaxPool3:** (3x3) S=(2x2) + ReLU
4	6912	**FC1:** 4096 + ReLU
	4096	**Drop1:** 0,5
5	4096	**FC2:** 4096 + ReLU
	4096	**Drop2:** 0,5
6	4096	**FC3:** 2 + Softmax

Tabla 1. Arquitectura de red de Tao2016

A continuación, se muestra en la Tabla 3 una recopilación de los principales hiperparámetros de entrenamiento.

Learning rate	0,01
Learning rate decay	0,0005
Tamaño de lote	128
Epochs	1000

Tabla 2. Hiperparámetros de entrenamiento uitlizados en el artículo Tao2016

Tras llevar a cabo un experimento con dichos parámetros esta red obtiene los resultados de la Tabla 3.

Precisión	Fall-out/FPR	Accuracy
94,20 %	0,86 %	96,88 %

Tabla 3. Métricas resultantes del experimento llevado a cabo en el artículo Tao2016

No obstante, el conjunto de imágenes utilizado es muy dispar al conjunto del presente proyecto, tal y como se puede observar en la *Figura 30*, ya que las imágenes de humos consisten en una imagen únicamente con humo, sin ningún otro objeto o fondo, y las imágenes de la otra clase se tratan de imágenes claramente distintas. En este conjunto de

imágenes el balanceo entre clases consiste en 688 imágenes de humos frente a 817 de ejemplos distintos al humo.

Figura 30. Conjunto de imágenes de CNN

3.2.1.2 Detección de humos mediante una DNCNN

A continuación se presenta un sistema de detección de humos por medio de imágenes presentada en [Yin 2017] mediante la implementación de una DNCNN (Deep Normalization and Convolutional Neural Network) basada en la idea de la adición de una fase de normalización por lotes (*batch nomalization*) en cada capa convolucional de la red.

Este estudio comprueba que si a la red presentada en el artículo se le añaden capas de normalización por lotes se consigue eliminar el overfitting existente en dicha red sin estas capas de normalización.

En la *Figura 31* se representa la arquitectura de dicha CNN conformada por catorce capas, en las que los números en rojo hacen referencia al número de mapas de características de la imagen de entrada recogidas en el volumen correspondiente a cada capa y los números verdes referencian al alto y ancho del volumen.

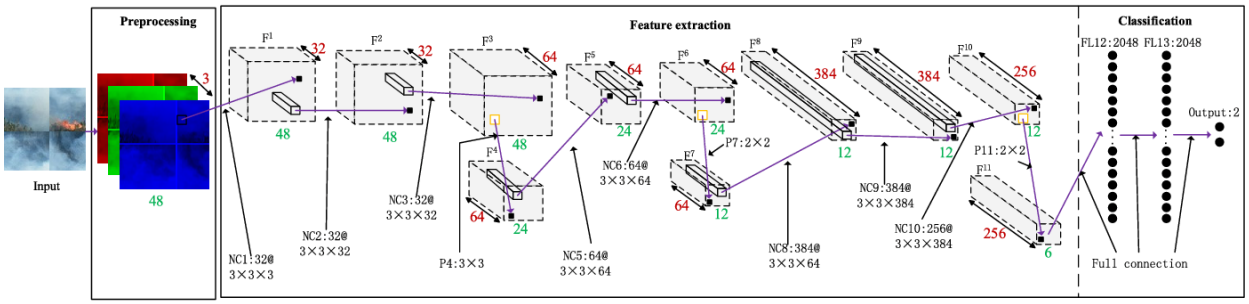

Figura 31. Arquitectura DNCNN

Estas capas pueden observarse en mayor detalle en la Tabla 4, consistente en la especificación de todos los parámetros característicos de cada capa.

Capa	Entrada	Hiperparámetros
1	3 (48x48)	**Conv1:** 32 (3x3) S=(1x1) Padding + ReLU
	32 (48x48)	**BatchNorm1:** Momentum=0.9
	32 (48x48)	**Conv2:** 32 (3x3) S=(1x1) Padding + ReLU
	32 (48x48)	**BatchNorm2:** Momentum=0.9
	32 (48x48)	**Conv3:** 64 (3x3) S=(1x1) Padding + ReLU
	64 (48x48)	**BatchNorm3:** Momentum=0.9
	64 (48x48)	**MaxPool1:** (3x3) S=(2x2)
2	64 (24x24)	**Conv4:** 64 (3x3) S=(1x1) Padding + ReLU
	64 (24x24)	**BatchNorm4:** Momentum=0.9
	64 (24x24)	**Conv5:** 64 (3x3) S=(1x1) Padding + ReLU
	64 (24x24)	**BatchNorm5:** Momentum=0.9
	64 (24x24)	**MaxPool2:** (2x2) S=(2x2)
3	64 (12x12)	**Conv6:** 384 (3x3) S=(1x1) Padding + ReLU
	384 (12x12)	**BatchNorm6:** Momentum=0.9
	384 (12x12)	**Conv7:** 384 (3x3) S=(1x1) Padding + ReLU
	384 (12x12)	**BatchNorm7:** Momentum=0.9
	256 (12x12)	**Conv8:** 256 (3x3) S=(1x1) Padding + ReLU
	256 (12x12)	**BatchNorm8:** Momentum=0.9
	256 (12x12)	**MaxPool3:** (2x2) S=(2x2)
	9216	**FC1:** 2048
	2048	**FC2:** 2048
	2048	**FC3:** 2

Tabla 4. Arquitectura de red de Yin2017

Esta red recibe como entrada un conjunto de imágenes resultante de una fase previa de preprocesamiento de imágenes consistente en la **normalización de la iluminación** de las imágenes mediante el cómputo de cada píxel con la siguiente operación:

$$x_n = \frac{x_r - x_{min}}{x_{max} - x_{min}}$$

Con objeto de obtener un conjunto de imágenes lo suficientemente amplio en cuanto a número de imágenes, el artículo desarrolla la generación de nuevos ejemplos a partir de los existentes relativos a aquellos que contienen humos mediante la **rotación** de cada uno de ellos en el eje **horizontal, vertical y rotación de 90º**.

Este artículo presenta una fase de entrenamiento basada en los hiperparámetros mostrados en la Tabla 5.

Learning rate	0,01
Learning rate decay	0,01
Tamaño de lote	32
Epochs	300

Tabla 5. Hiperparámetros de entrenamiento en Yin2017

A fin de obtener una referencia del rendimiento de estas imágenes, en la fase de experimentación del artículo se presentan tanto el entrenamiento y la evaluación de la red presentada anteriormente (con y sin normalización), como el de otras redes clásicas como AlexNet, ZF-Net y VGG16. En la Tabla 6 se muestra una comparación de los resultados obtenidos.

Deep CNN	Precisión	Fall-out/FPR	Accuracy	Nº parámetros
AlexNet	93,29 %	**0,24 %**	97,18 %	60.000.000
ZF-Net	93,29 %	0,24 %	97,18 %	60.000.000
VGG16	**96,19 %**	0,60 %	97,48 %	120.000.000
DNCNN sin normalización	90,21 %	2,04 %	94,86 %	20.000.000
DNCNN	95,28 %	0,48 %	**97,83 %**	20.000.000

Tabla 6. Resultados de los experimentos presentados en Yin2017

De los resultados obtenidos se puede observar que la red presentada en el artículo se puede observar una notable mejora a la hora de añadir a la red las capas de normalización por lotes. Se puede observar que la red DNCNN obtiene el mayor valor de accuracy, sin embargo, esta métrica no es considerada como buen indicador del rendimiento de una red, ya que evalúa el ratio de ejemplos clasificados de forma correcta, por lo que en conjuntos de imágenes muy desbalanceados si la red clasificase todos los ejemplos como la clase mayoritaria, el valor de esta métrica sería alto.

Sin embargo, dependiendo de qué métrica se considere más importante en el problema a resolver, la red óptima puede variar. Un ejemplo de esto es que la red VGG16 cuenta con una mayor precisión que el resto, sin embargo, es una de las que proporcionan más falsas alarmas debido a que la métrica fall-out es mayor.

Al igual que en la implementación anterior, las imágenes representantes de la clase de imágenes con humos se tratan de imágenes consistentes únicamente en humos, sin contar con ejemplos de paisajes u otro tipo de fondos que contengan emisiones. Se muestra a

continuación un ejemplo que resume el conjunto de imágenes utilizado en esta implementación de red neuronal, en la que *a, c, e* y *g* pertenecen a la clase de imágenes con humos, y el resto a la clase sin humos. Un ejemplo de este conjunto de imágenes se presenta en la *Figura 32*.

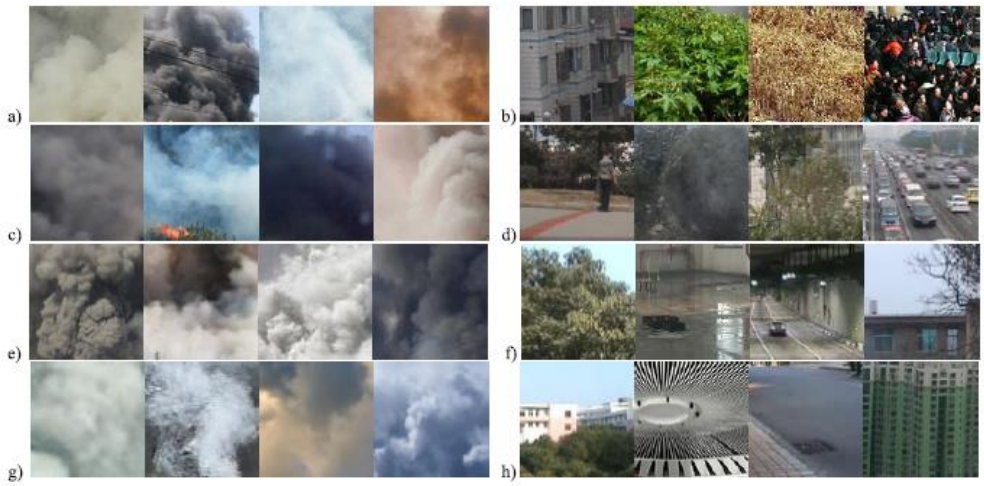

Figura 32. Conjunto de imágenes de DNCNN

3.2.1.3 Conclusión

Esta sección recopila el uso de diversas redes (desde redes diseñadas específicamente para la detección de humos hasta redes clásicas de detección de objetos) para resolver la clasificación de imágenes diferenciando entre aquellas que presentan únicamente humo, de aquellas que presentan una amplia diversidad de objetos. Sin embargo, estas redes pierden la información relativa al tiempo debido a que no tiene en cuenta la cronología de las imágenes.

3.2.2 Clasificación de imágenes en base a características temporales y espaciales

A continuación, se exponen dos implementaciones desarrolladas para la detección de humos mediante el análisis espacio-temporal de las imágenes que conforman el conjunto.

3.2.2.1 CNN de dos canales

Otra propuesta relacionada con el objetivo del proyecto se trata de la expuesta en [Wang 2017] consistente en la detección de fugas de gas natural mediante una red neuronal convolucional de dos canales en la que uno analizará las **características espaciales y otro las temporales.**

Esta red está compuesta por tanto por dos subredes encargadas de cada uno de estos canales. La primera, encargada de las características espaciales, obtendrá una clasificación en base a las imágenes originales como entrada. Sin embargo, la red encargada de las características temporales generará una clasificación a partir de imágenes de entrada preprocesadas.

El preprocesamiento de estas imágenes de entrada para el canal temporal consistirá en el cálculo del flujo óptico de todas y cada una de las imágenes, tal y como se especificó anteriormente en la descripción de la solución planteada en [Altun and Celenk 2013].

A partir de este flujo se obtendrá el ***flujo óptico denso*****,** que permitirá obtener una imagen de entrada más adecuada para la clasificación por parte de una red neuronal que la obtenida a partir del flujo óptico. Este se calculará a partir de la aproximación de funciones obtenida por medio de la expansión polinómica de las series de Taylor.

En la *Figura 33* se muestran dos frames consecutivos que servirán como entrada a la red de características espaciales, además de ser utilizadas para la obtención del flujo óptico (imagen situada en la esquina inferior derecha), necesario para el cálculo del flujo óptico denso correspondiente a la imagen situada en la esquina inferior izquierda en la que los distintos colores representan diversas velocidades y direcciones de los píxeles.

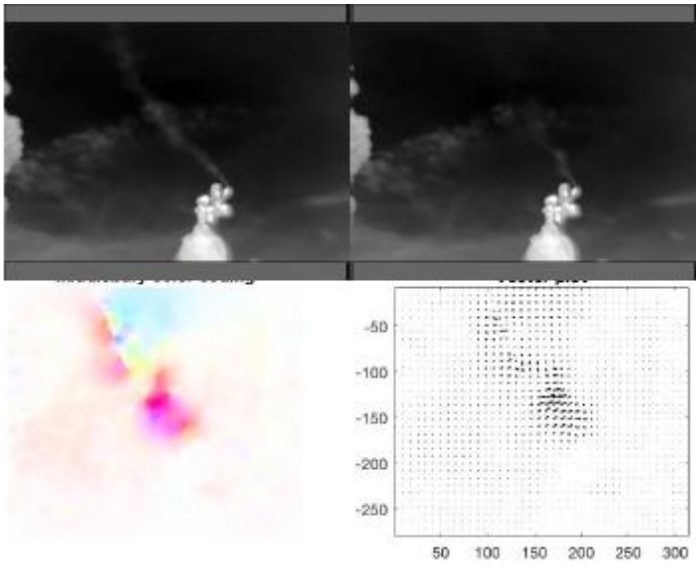

Figura 33. Resultado del preprocesamiento del canal temporal

El conjunto de imágenes utilizado en [Wang 2017] consiste en cinco vídeos de 1.495 frames para cada una de las 6 clases correspondientes a distintos niveles de fuga, tal y como se muestra en la Tabla 7.

Clase 1	Clase 2	Clase 3	Clase 4	Clase 5	Clase 6
$1\,^{L}/_{min}$	$2\,^{L}/_{min}$	$3\,^{L}/_{min}$	$4\,^{L}/_{min}$	$5\,^{L}/_{min}$	$0\,^{L}/_{min}$

Tabla 7. Interpretación de clases en Wang2017

En la Figura 34 se presenta un frame como ejemplo de cada una de las 6 clases existentes, pudiendo comprobar que la clasificación de cada una de ellas de forma independiente no parece, a simple vista, tratarse de una tarea trivial.

Figura 34. Frames pertenecientes a las distintas clases de Wang2017

Cada uno de los vídeos han sido divididos de modo que el 20% del vídeo forma parte del conjunto de test y el 80% restante está destinado a la fase de entrenamiento, en base a la distribución de la Figura 35.

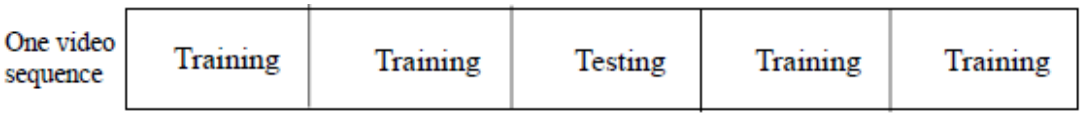

Figura 35. División en entrenamiento y test de una secuencia de vídeo en Wang2017

En cuanto a la arquitectura de red, ambas subredes cuentan con la misma arquitectura de capas, por lo que la CNN resultante tomaría la estructura mostrada en la *Figura 36*.

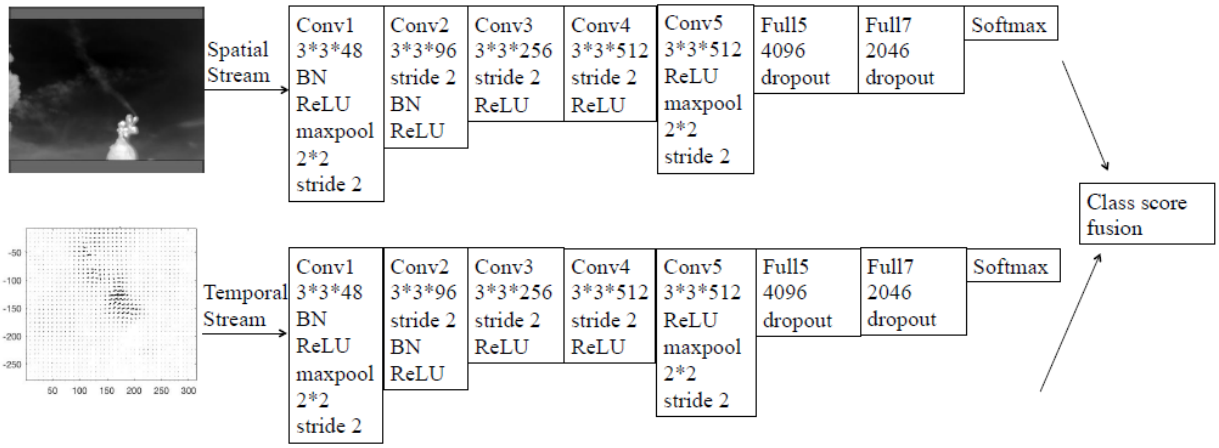

Figura 36. Arquitectura de CNN espacio-temporal compuesta por dos canales

Finalmente se lleva a cabo una comparación de los resultados obtenidos por ambos canales de forma conjunta e independiente, obteniendo los siguientes resultados de la Tabla 8. Resultados de la experimentación de Wnag2017.

	Precisión	
Modelo	**Entrenamiento**	**Testing**
Red espacial	68,30%	65,20%
Red temporal	56,50%	52,30%
Red espacio-temporal	80,70%	77,60%

Tabla 8. Resultados de la experimentación de Wnag2017

Como se puede observar a pesar de que los valores de precisión no son tan altos como los de apartados anteriores esto puede deberse a la dificultad de clasificar el conjunto de imágenes utilizado, el cual presenta una mayor similitud a los imágenes del problema a resolver en este proyecto. No obstante, se puede observar que utilizando una combinación de ambos canales la precisión del modelo aumenta.

Por tanto, este artículo prueba que la implementación de una red que tenga en cuenta tanto características temporales como espaciales obtiene un mejor rendimiento que si contemplase únicamente un tipo de características para el conjunto de imágenes utilizado.

3.2.2.2 Comparativa de 2D y 3D CNN's en detección de humos [Hohberg 2015]

En la tesis doctoral [Hohberg 2015] se lleva a cabo un estudio e implementación de detección de humos en incendios forestales mediante distintos modelos de redes neuronales convolucionales.

El conjunto de imágenes utilizado consiste en una serie de imágenes tomadas en instantes consecutivos, lo que permite que sean interpretados como frames de vídeo. Este conjunto de imágenes es tratado inicialmente durante una fase previa de preprocesamiento con las siguientes operaciones:

1. **Estabilización de imágenes mediante la transformada de Fourier.** Esto tiene como objetivo evitar imágenes borrosas ya que estas son tomadas por cámaras móviles que, además, pueden ser desplazadas por el viento.
2. **Normalización.** Se reescala la intensidad de cada píxel al rango [0, 1].

Tras la fase de preprocesamiento de imágenes se establece una siguiente fase de proposición de regiones con probabilidad de contener humos partiendo de la división de cada una de las imágenes en secciones y etiquetándolas según contengan o no humos. Esto genera un conjunto de imágenes con una gran cantidad de ejemplos negativos, por lo que se debe establecer una estrategia a la hora de seleccionar qué secciones de imágenes se van a utilizar durante el entrenamiento. Para ello [Hohberg 2015] definió dos enfoques distintos realizando un juicio crítico posterior en base a los resultados obtenidos con cada uno de ellos.

El primer enfoque de proposición de regiones consiste en dividir todas y cada una de las imágenes en regiones de las mismas dimensiones y se asignan de forma aleatoria a los distintos conjuntos de entrenamiento, validación o testing. Mientras tanto, el segundo enfoque implementa un **algoritmo de búsqueda selectiva** que se encarga de **proponer regiones** de la imagen que contengan objetos.

Los resultados muestran que el segundo enfoque obtiene un buen rendimiento, mientras que el primero es muy sensible ante bordes de objetos o de la propia imagen confundiéndolos con humo.

Las regiones propuestas por estos algoritmos consistirán la entrada de la red neuronal que constituye la fase final del sistema.

Con objeto de determinar que arquitectura de red es más adecuada para la detección de humos, [Hohberg 2015] lleva a cabo un estudio comparativo de los distintos resultados que obtienen las seis arquitecturas seleccionadas distintas.

La primera se trata del modelo ***CaffeNet***, una red diseñada para la clasificación de más de 1000 clases, cuya arquitectura se puede observar en la *Figura 37* en la que la notación ⊞ hace referencia al tamaño del kernel, mientras que el símbolo □ se refiere al número de filtros y mapas de características de la capa y las flechas al valor de stride.

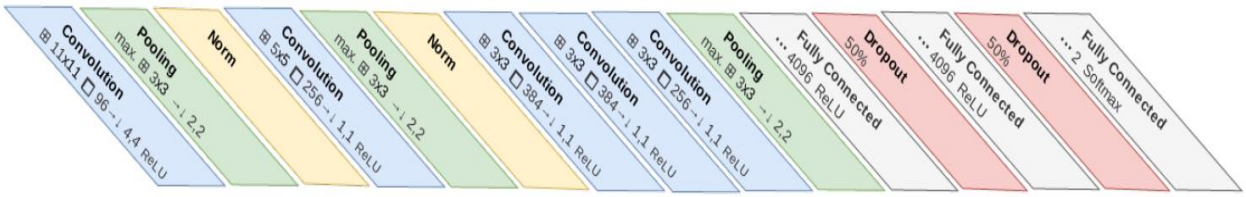

Figura 37. Arquitectura de CaffeNet

Otro modelo utilizado por [Hohberg 2015] es una adaptación del anterior, ya que este está orientado a predecir una gran cantidad de clases cuando va a ser utilizado para llevar a cabo una clasificación binaria. Para ello se define la variante ***CaffeNet$_{red}$*** en la que se reduce a ¼ el número de mapas de características generados en cada capa. Al reducir esta arquitectura se busca mejorar la capacidad de generalización de la red, debido a que cuenta con un menor número de características a aprender.

El siguiente modelo se basa en la implementación *GoogLeNet* que cuenta con módulos de Inception [Szegedy 2016]. Este modelo consiste en tres capas convolucionales y nueve módulos de Inception conformados por seis capas convolucionales, lo que supone una arquitectura de una gran dimensión.

Es por ello por lo que los modelos a utilizar consistirán en las siguientes **dos modificaciones de GoogLeNet** mostradas en la *Figura 38* (GoogLeNet1) y *Figura 39* (GoogleNet2).

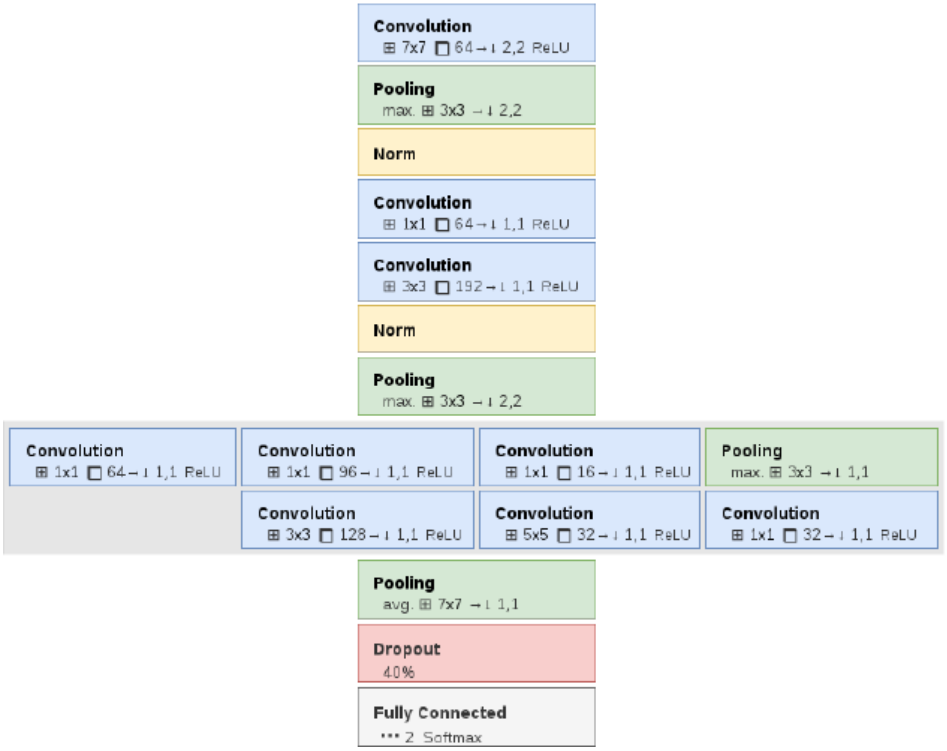

Figura 38. Arquitectura de GoogLeNet1

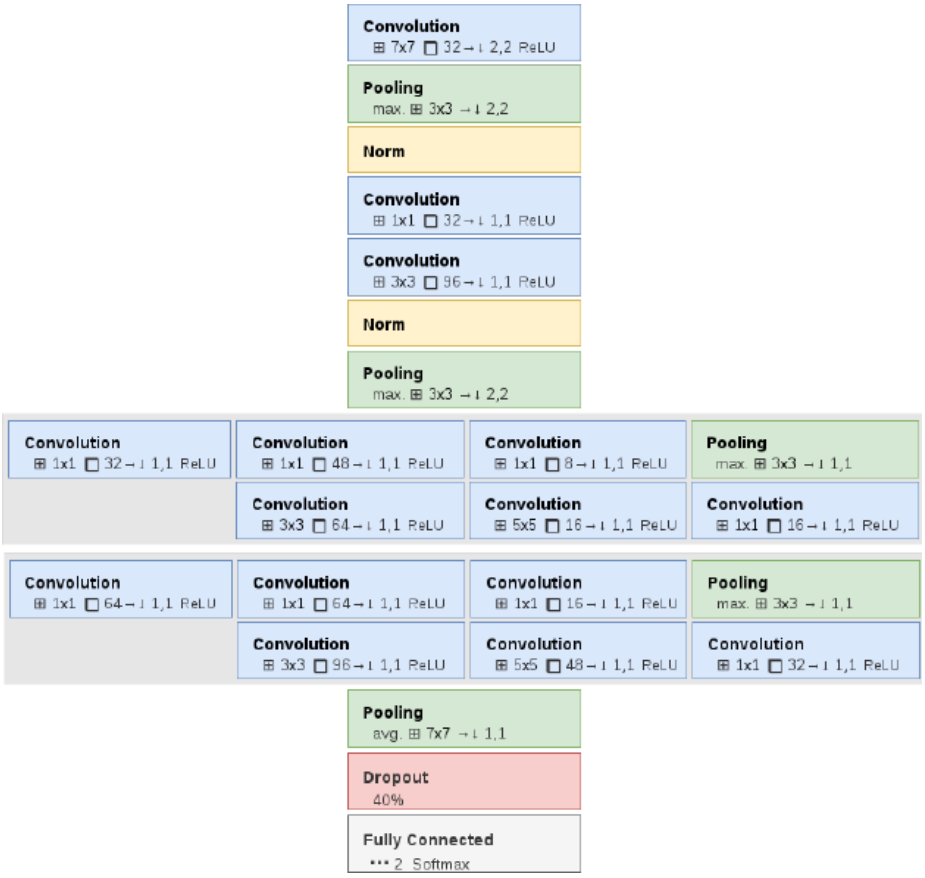

Figura 39. Arquitectura de GoogLeNet2

Hasta ahora los cuatro modelos de CNN utilizados en [Hohberg 2015] toman como entrada un conjunto de imágenes independientes. Sin embargo, [Hohberg 2015] también analiza el rendimiento de modelos de redes neuronales que tomen no solo características espaciales si no también temporales, llamadas redes C3D (*Convolutional 3D)*.

Para ello [Hohberg 2015] ha propuesto **dos modelos de red neuronal C3D** en base a las arquitecturas ***CaffeNet$_{red}$*** **y** ***GoogLeNet$_2$***.

Una vez definidas e implementadas todas las CNNs [Hohberg 2015] muestra una comparación del rendimiento de cada uno de los 6 modelos, además de uno último consistente en un ensamblado de todos los anteriores, estableciendo una clasificación en función del promedio del resultado de las predicciones de los 6 modelos. Estos resultados permitirán establecer un juicio crítico sobre cada uno de estos modelos.

Modelo	**Accuracy**	**Recall/TPR**	**Fall-out/FPR**	**Recall/TPR$_{seq}$**	**Fall-out/FPR$_{seq}$**
Training					
CaffeNet	97,57%	87,77%	2,01%	58,69%	48,72%
CaffeNet$_{red}$	97,06%	88,30%	2,54%	61,97%	54,62%
GoogLeNet$_1$	97,71%	87,30%	1,85%	59,02%	40,33%
GoogLeNet$_2$	98,14%	85,61%	1,40%	52,33%	34,03%
C3D GoogLeNet$_2$	98,53%	87,79%	1,00%	61,84%	25,77%
C3D CaffeNet$_{red}$	98,50%	87,94%	1,03%	60,85%	29,64%
Ensemble	98,79%	87,13%	0,73%	57,05%	23,41%
Validation					
CaffeNet	98,18%	88,62%	1,49%	65,48%	50,97%
CaffeNet$_{red}$	97,99%	89,19%	1,69%	68,71%	51,94%
GoogLeNet$_1$	98,29%	87,96%	1,38%	63,55%	42,90%
GoogLeNet$_2$	98,53%	86,44%	1,12%	55,16%	38,39%
C3D GoogLeNet$_2$	98,81%	88,44%	0,84%	62,90%	27,42%
C3D CaffeNet$_{red}$	98,96%	88,14%	0,97%	60,97%	23,23%
Ensemble	99,00%	87,90%	0,64%	59,03%	19,68%

Tabla 9. Resultados de la experimentación realizad por Hohberg2015

En la Tabla 9 se muestran los resultados de accuracy, recall y fall-out de cada modelo tras aplicar el segundo enfoque de proposición de regiones, además del recall y fall-out de la aplicación del primer enfoque.

Como se puede observar el rendimiento del sistema es mucho mayor aplicando previamente un algoritmo de proposición de regiones óptimo.

También se puede comprobar que, al igual que en el estudio de la sección anterior, aquellas CNN que se basan en características espacio-temporales obtienen mayores valores de accuracy y menores de falsos positivos. Por otro lado, aunque *CaffeNet$_{red}$* cuenta con la mejor medida de recall (seguida por las redes C3D) el índice de falsos positivos es también el más alto, lo que hace que la accuracy sea la más baja de todos los modelos.

Por tanto, el estudio de [Hohberg 2015] también demuestra que los modelos C3D que tienen en cuenta características de movimiento de humos obtienen mejor rendimiento que aquellos modelos convencionales con la misma arquitectura que analizan únicamente características espaciales.

3.2.2.3 Conclusión

Finalmente, de esta sección se puede concluir que los resultados obtenidos son peores que los de la anterior sección, pero esto puede deberse a que las imágenes a clasificar difieren menos entre ellas, ya que no se trata de diferenciar ejemplos que contengan únicamente humos frente a otras imágenes de objetos totalmente distintos. Debido a esto no podemos determinar con certeza qué red es más adecuada para el sistema a implementar.

Por otro lado, se ha comprobado que para ambos casos el uso de características temporales sumado a las características espaciales proporciona un incremento del rendimiento de la red.

3.2.3 Detección de objetos mediante segmentación en regiones

Otra técnica destinada a la detección de humos se trata de la expuesta por [Yao 2017] en la que se hace uso de una red neuronal distinta a las anteriores, llamada ***Faster R-CNN*** (Region-based Convolutional Neural Network).

La arquitectura de esta red consiste básicamente en dos redes neuronales concatenadas. La primera se trata de una **RPN (región proposal network)** encargada de generar regiones como posibles áreas en las que se encuentre el objeto a detectar, mientras que la segunda recibe como entrada estas regiones y se hace cargo de la detección de objetos en dichas regiones. Esta arquitectura se puede observar en la *Figura 40*.

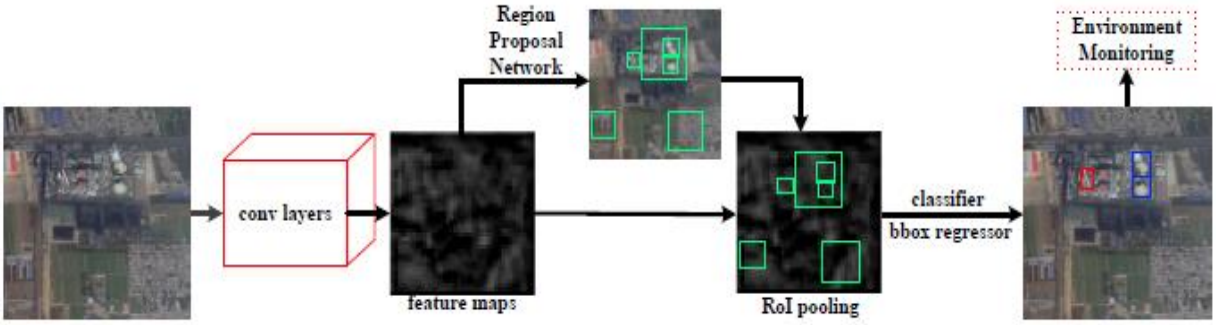

Figura 40. Arquitectura R-CNN

A diferencia de las anteriores implementaciones, esta lleva a cabo una tarea de **detección de objetos** en vez de únicamente establecer una estimación de clases de las imágenes en función de si han detectado o no humos. Esta implementación puede generar como salida diversas regiones dependiendo de si ha detectado o no el objeto deseado.

Sin embargo, esta implementación está más orientada a la detección de chimeneas que de humos, por lo que un posible resultado de la red puede ser una chimenea que no esté emitiendo humos, objetivo que no interesa en el presente proyecto. Además, en el conjunto de imágenes a utilizar, las emisiones no tienen por qué provenir de chimeneas.

A continuación, en la Figura 41 se muestra un ejemplo de las imágenes utilizadas como entrada y en la Figura 42 un ejemplo de los resultados de esta red (conteniendo también resultados consistentes en chimeneas que no emiten humos)

Figura 41. Imágenes entrada de la red Yao2017

Figura 42. Regiones porpuestas por Faster R-CNN

3.2.3.1 Conclusión

Finalmente, este tipo de redes no se tomarán en consideración en este proyecto debido a que se trata de sistemas muy específicos cuyos conjuntos de datos que deben estar segmentados. Debido a esto, en caso de necesidad de ampliar el conjunto de datos, la tarea de etiquetado de los segmentos de las imágenes implicaría emplear mucho más tiempo en la preparación del conjunto.

4 Entorno de trabajo

4.1 Tensorflow

Tensorflow se trata de una librería de código libre basada en aprendizaje automático consistente en la construcción de modelos mediante el uso de redes neuronales de aprendizaje profundo lanzado por Google a finales del año 2015 y desarrollado por el equipo Google Brain.

Es utilizado por Google para llevar a cabo sus productos por medio de redes neuronales artificiales, como es el caso de los sistemas de reconocimiento de voz.

Actualmente está disponible para los lenguajes de programación Python, C++, C, Java, R y Go. Para el desarrollo de este proyecto se hará uso de la última versión de TensorFlow y la versión de Python 3.5.0.

4.2 Keras

Esta librería de redes neuronales implementada en Python es capaz de ejecutar como backend tanto Tensorflow como Theano de forma compatible con CPUs y GPUs.

Permite crear, entrenar y visualizar modelos de aprendizaje profundo de forma más sencilla y modular, pudiendo desarrollar tanto redes recurrentes como convolucionales (interés del presente proyecto).

4.3 Python

Como lenguaje de programación empleado para la implementación de las distintas soluciones propuestas al problema se trata de Python, un lenguaje de alto nivel de tipado dinámico, multiplataforma y multiparadigma, que soporta orientación a objetos, programación imperativa y programación funcional.

Ambas bibliotecas (Keras y Tensorflow) están disponibles tanto en Python como en otros lenguajes de programación. Sin embargo, se ha seleccionado Python debido a la simplicidad a la hora de la codificación y legibilidad del código, además de que es el lenguaje más extendido a la hora de trabajar en este campo.

5 Conjunto de imágenes

Las tareas de comprensión y preparación del conjunto de imágenes a emplear en el presente problema son primordiales debido a que estas imágenes consisten en la información que va a ser tratada por la red neuronal. Por ello, se deberá prestar especial atención a las características del conjunto y la división de este en subconjuntos de entrenamiento, validación y testing, así como todas aquellas tareas de carga de las imágenes en el programa y su tratamiento previamente a introducirlas como entrada en la red neuronal.

5.1 Conjunto inicial

En la fase inicial del proyecto se cuenta con un conjunto constituido por imágenes de una planta siderúrgica. Este conjunto cuenta con imágenes (sin etiquetar) tomadas diariamente cada cinco segundos durante tres meses consecutivos dividiendo cada mes en un directorio distinto y cada día de dicho mes en un subdirectorio.

Estas imágenes han sido tomadas por la misma cámara, por lo que todas ellas cuentan con unas dimensiones de 2048 píxeles de ancho y 1536 de alto con una resolución de 3,779x3,779 píxeles por milímetro. Están representadas en el espacio de color RGBA proporcionando 8 bits para cada uno de los tres canales en los que se representa la imagen. Presentan además un cuarto canal alfa, sin embargo, tras analizar cada una de las capas de las imágenes mediante un software de manipulación de imágenes, se determinó que dicho canal no aportaba información a la imagen, por lo que se trabajará únicamente con los 3 canales principales (RGB).

Teniendo en cuenta que la cámara toma una imagen cada cinco segundos, todo aquel subdirectorio que represente un día debería estar conformado por un total de 17.280 imágenes. Sin embargo, se ha observado que el número de imágenes tomadas cada día varía. Es por ello por lo que en la Tabla 10 se muestra de forma tabular una representación del volumen de imágenes disponible para el actual problema, así como las características meteorológicas principales de cada día, ya que deberán tenerse en cuenta a la hora de conformar un conjunto de imágenes óptimo de entrenamiento, validación y testing.

Día	Mes	Nº de imágenes	Características
Día 1	Mes 1	17.243	Nublado
Día 2	Mes 1	17.213	Nublado
Día 3	Mes 1	17.206	Soleado
Día 4	Mes 1	17.232	Soleado
Día 5	Mes 1	17.231	Soleado
Día 6	Mes 1	17.231	Nublado
Día 7	Mes 1	17.261	Nublado
Día 8	Mes 1	17.198	Nublado
Día 9	Mes 1	17.259	Parcialmente soleado
Día 10	Mes 1	17.058	Parcialmente nublado
Día 11	Mes 2	17.261	Nublado
Día 12	Mes 2	17.224	Parcialmente nublado
Día 13	Mes 2	17.244	Parcialmente nublado
Día 14	Mes 2	17.238	Parcialmente nublado
Día 15	Mes 2	17.260	Parcialmente nublado
Día 16	Mes 2	17.236	Nublado
Día 17	Mes 2	17.168	Parcialmente nublado
Día 18	Mes 2	17.250	Parcialmente soleado
Día 19	Mes 2	16.905	Soleado
Día 20	Mes 2	17.247	Parcialmente soleado Atardecer
Día 21	Mes 2	17.125	Nublado
Día 22	Mes 2	17.268	Parcialmente soleado
Día 23	Mes 2	17.220	Parcialmente soleado
Día 24	Mes 2	17.189	Parcialmente soleado
Día 25	Mes 2	17.262	Nublado
Día 26	Mes 2	17.263	Parcialmente soleado
Día 27	Mes 2	17.259	Nublado
Día 28	Mes 2	16.926	Nublado
Día 29	Mes 2	16.566	Nublado
Día 30	Mes 2	17.216	Nublado Atardecer
Día 31	Mes 2	17.174	Nublado

Día 32	Mes 2	17.184	Soleado
Día 33	Mes 2	17.252	Soleado
Día 34	Mes 2	17.245	Nublado
Día 35	Mes 2	17.187	Nublado
Día 36	Mes 2	17.237	Nublado
Día 37	Mes 2	17.141	Nublado
Día 38	Mes 2	17.254	Nublado
Día 39	Mes 2	17.236	Soleado
Día 40	Mes 2	17.265	Parcialmente nublado
Día 41	Mes 2	17.261	Nublado
Día 42	Mes 3	17.021	Nublado
Día 43	Mes 3	16.990	Soleado
Día 44	Mes 3	17.237	Parcialmente soleado Niebla
Día 45	Mes 3	17.172	Parcialmente nublado
Día 46	Mes 3	17.163	Parcialmente soleado
Día 47	Mes 3	17.238	Nublado
Día 48	Mes 3	16.469	Parcialmente nublado
Día 49	Mes 3	17.030	Parcialmente nublado
Día 50	Mes 3	17.045	Parcialmente nublado
Día 51	Mes 3	17.204	Parcialmente soleado
Día 52	Mes 3	17.257	Soleado
Día 53	Mes 3	17.208	Parcialmente nublado
Día 54	Mes 3	17.264	Parcialmente nublado
Día 55	Mes 3	17.214	Soleado
Día 56	Mes 3	17.202	Soleado
Día 57	Mes 3	17.251	Nublado
Día 58	Mes 3	17.264	Nublado
Día 59	Mes 3	13.965	Soleado
Día 60	Mes 3	17.206	Nublado

Tabla 10. Información relevante recopilada sobre el conjunto de imágenes del problema

5.2 Análisis del conjunto de imágenes

Una vez se ha recopilado toda información respectiva tanto al volumen del conjunto como a las principales características de éste, se procederá a llevar a cabo la búsqueda del conjunto a utilizar durante las fases de entrenamiento, validación y testing.

En base a la información diaria expuesta anteriormente, se ha llevado a cabo un análisis en profundidad de las imágenes con objeto de identificar todos los aspectos para tener en cuenta a la hora de conformar un conjunto de imágenes óptimo. A continuación, se listan aquellos aspectos que se consideran relevantes.

1. **Imágenes nocturnas:** A pesar de que la cámara toma imágenes durante la noche, debido al bajo nivel de luminiscencia no se puede determinar a simple vista la existencia de emisiones, incapacitando la labor de etiquetado de emisiones durante la noche. Debido a esto se procederá a **etiquetar las imágenes nocturnas como imágenes sin emisión**, con objeto de que la red aprenda que a bajos niveles de luminiscencia deberá clasificar las imágenes como "*sin emisión*".
2. **Desbalanceo de clases:** Aún sin contar las imágenes nocturnas etiquetadas como "*sin emisión"*, el número de ejemplos de **esta clase es muy superior al número de ejemplos que presentan emisiones atmosféricas**. Es por ello por lo que se deberá realizar un estudio de cómo afecta el desbalanceo de clases al rendimiento de la red neuronal, mediante la comparación de resultados y tiempos de ejecución ante distintas proporciones bajo las mismas condiciones en términos del modelo de red neuronal a usar, escalado de las imágenes, etc.
3. **Naturaleza de las emisiones**: Se deberán tener en cuenta las características de aquellas imágenes clasificadas como "*con emisión" que* presentan emisiones atmosféricas identificadas como ráfagas de corta duración en la parte superior del edificio de la acería, tomando una forma similar a las nubes, lo que puede conllevar en algunas ocasiones a falsas alarmas.
4. **Características atmosféricas:** El sistema deberá estar preparado para realizar predicciones adecuadas bajo una amplia diversidad de características atmosféricas como un día nublado o de niebla, en el que el sistema pueda confundir emisiones con nubes o niebla, o durante el atardecer, en el que el sistema se pueda confundir debido a las tonalidades (similares a las emisiones) que toman las nubes.
5. **Nivel de precisión deseado:** Se desea que el sistema identifique emisiones de alta y media dimensión espacial, con objeto de detectar todas aquellas emisiones existentes.

5.3 Etiquetado de imágenes

Una vez analizadas las imágenes del conjunto se han seleccionado 5 días considerados más representativos en base a los criterios establecidos en el apartado *5.2*, además de la información recopilada durante el desarrollo del apartado *5.1*. Los días seleccionados se tratan del día 1, 2, 10, 20 y 30, ya que entre ellos conforman un conjunto con ejemplos de sol, nubes de diversas tonalidades y atardeceres y amaneceres (con y sin nubes). Las imágenes de estos días han sido etiquetadas de forma manual en dos clases:

- Clase 0: Sin emisión
- Clase 1: Con emisión

El etiquetado consiste básicamente en la división del directorio que cuenta con todas las imágenes de dicho día en dos subdirectorios (uno para cada una de las clases ya definidas).

Tal y como se indicó previamente, el sistema deberá detectar emisiones de dimensiones y densidades grandes o medianas, descontando aquellas que cuentan con dimensiones pequeñas o de baja densidad.

Este etiquetado consiste en la clasificación de cada imagen basándose en el color de la posible emisión y de la densidad y dimensión de esta. Dado que esta clasificación no se trata de una tarea que se pueda establecer a ciencia cierta es posible que existan incongruencias en el etiquetado del conjunto.

Debido a que posibles errores en el etiquetado supondrán un gran problema para el entrenamiento y evaluación de la red, se han llevado a cabo diversas fases de etiquetado con objeto de obtener un conjunto de datos sólido y fiable. Para ello se han etiquetado los conjuntos en base a 3 criterios.

5.3.1 Primer criterio de etiquetado

Este consiste en clasificar como emisión todo tipo de emisiones independientemente de su densidad y dimensión, y el resto como imágenes sin emisión.

5.3.2 Segundo criterio de etiquetado

Debido a que el anterior criterio incluye emisiones altamente dudosas, lo que hace que la red genere falsos negativos. Para ello se clasifican como emisión aquellas consideradas de densidad y dimensión intermedia o grande, y el resto se clasifican como imágenes sin emisión.

5.4 Etiquetado de secuencias de emisiones

Hasta el momento se ha hablado de emisiones como imágenes en las que se puede observar la presencia de una emisión. Sin embargo, una misma emisión es representada por diversas imágenes tomadas de forma consecutiva, por lo que la salida final del sistema debe ser el número de emisiones detectadas en lugar del número de imágenes que presentan una emisión se ha detectado.

Para ello se deberá desarrollar un algoritmo que detecte emisiones en función de la predicción de imágenes consecutivas en el tiempo. En base al análisis de la duración de emisiones y de la posibilidad de existencia de frames que no presentan emisiones dentro del intervalo de tiempo de existencia de la emisión, se han establecido los siguientes criterios:

I. El inicio de la secuencia de una emisión se establece como la fecha y hora de la primera imagen detectada dentro de una secuencia de imágenes (que presentan dicha emisión) de una duración mínima de 15 segundos. Debido a que este conjunto consiste en imágenes tomadas cada 5 segundos, se buscarán 3 imágenes de la clase *con emisión* seguidas en el tiempo.
II. El final de la secuencia se determina como la fecha y hora de la última imagen perteneciente a la clase *con* emisión dentro de la secuencia si, transcurridos 20 segundos desde la misma no se ha detectado una nueva imagen que presente alguna emisión.

En base a estos criterios, por tanto, se tendrán emisiones de una duración mínima de 3 imágenes y con un máximo de 4 imágenes consecutivas sin emisiones entre imágenes que sí presentan la emisión.

Este criterio será aplicado mediante la aplicación de un algoritmo que desarrolle la lógica explicada al conjunto de imágenes etiquetadas de forma independiente. De este modo se tendrá un etiquetado en base a secuencias, además del etiquetado de imágenes de forma independiente. También se le aplicarán las predicciones que la red realiza de todas y cada una de las imágenes del conjunto de testing.

Una vez se ha establecido la metodología de etiquetado y de predicción de secuencias, se establecerá la metodología respectiva al testing de las secuencia. Se ha de tener en cuenta que una emisión está compuesta por diversas imágenes y el etiquetado de una emisión (tal

y como se indicó anteriormente) se basa en un criterio relativamente subjetivo, en el que la existencia, inicio y final de la secuencia de una emisión se trata muchas veces de algo muy dudoso. Debido a esto se ha seleccionado como criterio de testing el definido a continuación:

$$Diff = \frac{R \cap P}{R}$$

Siendo *R* la secuencia **real** de imágenes que conforma una emisión (incluyendo imágenes de la clase *sin emisión*) y *P* la secuencia **predicha**, *Diff* modela la proporción entre la intersección de R y P en relación con R, ya que esto permitirá obtener un indicio de la similitud entre ambas secuencias, penalizando imágenes no detectadas de la secuencia real sin penalizar imágenes clasificadas como emisión y etiquetadas como *sin emisión*. Esto se debe a que, tras analizar el comportamiento de las redes se ha observado que en algunos casos la red detecta la emisión en frames en los que detectar dicha emisión a simple vista se trata de una tarea compleja.

Por tanto, se establecerá un valor α correspondiente a la proporción mínima de imágenes que componen la secuencia real que debe encontrarse en la secuencia predicha. En caso de que el valor diff sea menor a este valor, se considerará que la secuencia no ha sido predicha, contando por tanto como un falso negativo, a pesar de que se haya detectado alguna de las imágenes que conforman la secuencia real.

5.5 La problemática del desbalanceo de clases

Tal y como se indicó en el anterior apartado, el conjunto de imágenes presenta un gran desbalanceo de clases debido a que la probabilidad de que se dé una emisión es muy inferior a la probabilidad de que no se dé. Esto, por tanto, da lugar a un conjunto de imágenes altamente desbalanceado, lo que implica una reducción del rendimiento de las redes neuronales.

Esto se debe a que las redes neuronales, por defecto, buscan la optimización de la métrica *accuracy* que obtendrá buenos resultados ante una red que prediga todos los ejemplos como "*sin emisión"* debido a que estaría prediciendo de forma correcta la clase mayoritaria obteniendo valores sobre el 97% para las imágenes del conjunto. En base a este problema, se ha investigado sobre distintos enfoques (expuestos a continuación) que puedan ser empleados como posible solución a este problema.

5.5.1 Eliminar ejemplos de la clase mayoritaria (downsampling)

Una forma sencilla de resolver este problema de desbalanceo consistiría en eliminar de forma aleatoria ejemplos de la clase cuya proporción de ejemplos es mayor, en este caso consistiría en eliminar imágenes que no presenten emisiones. Esto permitirá reducir el desbalanceo entres clases, así como reducir el tiempo de entrenamiento.

5.5.2 Añadir ejemplos de la clase minoritaria (upsampling)

Como alternativa contraria a la anterior, se podrán generar nuevos ejemplos de la clase minoritaria (en este caso las imágenes que presentan emisión) mediante la copia de los ejemplos ya existentes de dicha clase hasta contar con una proporción balanceada de clase, o también mediante la creación de nuevos ejemplos mediante el procesamiento de los ejemplos existentes por medio de *data augmentation*.

Sin embargo, se debe prestar especial atención a qué filtros se aplican durante la fase de *data augmentation* ya que algunos filtros pueden afectar de forma negativa al posterior estudio de las características de las imágenes.

5.5.3 Seleccionar una métrica de entrenamiento óptima

Tal y como se indicó anteriormente, la métrica *accuracy* no es la más indicada para ser optimizada ante un conjunto de imágenes desbalanceado como el del actual problema, es

por ello por lo que otro posible enfoque puede ser buscar la optimización de otra métrica distinta durante la fase de entrenamiento.

5.5.4 Conclusión

El método de *downsampling* puede implicar una pérdida de rendimiento, debido a que cuenta con menos ejemplos de los que aprender, sin embargo, en este caso puede no resultar muy problemático debido a la gran cantidad de ejemplos de los que se dispone.

El método upsamplig, por otro lado, puede resultar adecuado dependiendo de qué filtros se apliquen durante la fase de *data augmentation* ya que algunos filtros pueden afectar de forma negativa al posterior estudio de las características de las imágenes. Debido a que se busca la detección de emisiones de tamaño muy pequeño que en ocasiones es difícil de distinguir a simple vista, la posibilidad de perder información aplicando filtros es alta. Este método, por tanto, se considera menos adecuado que el resto.

En cuanto a la selección de otra métrica a optimizar durante la fase de entrenamiento, se trata de una técnica muy utilizada para aplicaciones en las qué es preferible que la red obtenga unos mayores valores de ciertas métricas, como puede ser el caso de sistemas de detección de enfermedades como el cáncer, en los que es preferible que se den falsos positivos a falsos negativos ante una clase minoritaria, como puede ser la presencia de dicha enfermedad. Para este tipo de sistemas se utiliza la métrica recall (TPR) durante la fase de optimización del entrenamiento.

Este último método, sin embargo, se puede emplear en conjunto con el primero, permitiendo decrementar la tasa de descarte de ejemplos de la clase mayoritaria, así como el tiempo de entrenamiento requerido.

5.6 Carga del conjunto de imágenes

Debido a que las redes neuronales se basan en su mayor medida en la información aportada por un gran volumen de imágenes, se deberá prestar especial atención al procedimiento a seguir para cargar y manipular las imágenes en el sistema.

5.6.1 Redimensionado

Tal y como se pudo observar en anteriores apartados, las imágenes son de grandes dimensiones, por lo que si a esto se le suma el alto número de imágenes que conforman la base de datos del problema, el resultado final es un conjunto de imágenes de una dimensión desmesurada.

Por ejemplo, para una imagen de tamaño 2048x1536 píxeles a color, sería preciso almacenar 9.437.184 valores para una única imagen.

En base a estos cálculos y pruebas realizadas procesando un número reducido de imágenes con su tamaño original, además de llevar a cabo un estudio sobre las dimensiones de imágenes utilizadas actualmente en las competiciones de modelos de redes neuronales por parte de desarrolladores en tensorflow, se considera preciso un redimensionado de las imágenes a las dimensiones 256x192, equivalente a una reducción de 1/8 de las dimensiones reales de las imágenes.

Para llevar a cabo dicha reducción de tamaño se han explorado 2 alternativas,

1. Establecer como primera capa de la red neuronal una de pooling para reducir el volumen de imágenes con el que trabajará la red.
2. Realizar un redimensionado de la imagen nada más cargarla del archivo original.

5.6.1.1 Redimensionado mediante la capa pooling

Esta alternativa de redimensionado consiste en modificar la arquitectura de la red neuronal mediante la adición de una primera capa estableciendo el pertinente valor del tamaño de pooling en función al redimensionado que se desea llevar a cabo sobre las imágenes.

El principal problema de esta alternativa se trata de que el redimensionado del conjunto de imágenes no se llevaría a cabo hasta comenzar la primera fase del entrenamiento, al aplicar la primera capa del modelo, por lo que todas las operaciones realizadas anteriormente (como la carga y preprocesamiento del conjunto de imágenes) se verán altamente

afectadas en cuanto a rendimiento y nivel de ocupación de la memoria, llegando a exceder la capacidad del computador utilizado para el entrenamiento de la red.

5.6.1.2 Redimensionado durante la carga

El primer paso que se debe realizar es la carga de las imágenes conformando el conjunto de imágenes del problema. Para ello se ha realizado un estudio comparativo de dos grandes librerías disponibles en Python de procesamiento de imágenes denominadas PIL (Python Imaging Library) y OpenCV. Estas librerías permitirán cargar la imagen obteniendo los valores de los tres canales (RGB) y modificar sus dimensiones siguiendo el algoritmo de redimensionado especificado.

Ambas librerías cuentan con los siguientes cuatro algoritmos de redimensionado a analizar:

1. **Bilinear:** Lleva a cabo una interpolación lineal con un kernel de tamaño 2x2.
2. **Bicubic:** Realiza una interpolación spline cúbica mediante un kernel de tamaño 4x4.
3. **Lanczos:** Implementa el algoritmo iterativo de Lanczos[Lanczos 1950].
4. **Nearest:** Hace uso del algoritmo nearest neighbour.

OpenCV, por otro lado, cuenta con otro algoritmo llamado ***área***, diseñado para realizar tareas de *downsampling*. Este algoritmo hace uso de los valores de los píxeles que se encuentran dentro de un área cercana al píxel a tratar, llevando a cabo un redimensionado basado en la relación de los píxeles cercanos.

Con objeto de seleccionar el algoritmo más adecuado para las imágenes del problema, se ha determinado que, en base a la teoría, el algoritmo *área* de la librería OpenCV se trata del algoritmo más adecuado para decrementar las dimensiones de una imagen.

Sin embargo, se comprobará esta hipótesis de forma práctica por medio de la implementación del código (ANEXO I) preciso para generar una comparación de los resultados obtenidos por cada uno de los algoritmos de ambas librerías, aplicando un valor de redimensionado de una imagen de 1/8 de su tamaño original.

Los resultados obtenidos se pueden observar en la *Figura 43, Figura 44, Figura 45, Figura 46*, *Figura 47, Figura 48, Figura 49, Figura 50, Figura 51* y *Figura 52.*

Figura 43. Ejemplo imagen original

Figura 44. Ejemplo PIL redimensión Antialias

Figura 45. Ejemplo PIL redimensión Bicubic

Figura 46. Ejemplo PIL redimensión Bilinear

Figura 47. Ejemplo PIL redimensión Nearest

Figura 48. Ejemplo cv2 redimensión Bicubic

Figura 49. Ejemplo cv2 redimensión Bilinear

Figura 50. Ejemplo cv2 redimensión Antialias

Figura 51. Ejemplo cv2 redimensión Nearest

Figura 52. Ejemplo cv2 redimensión Area

En base a las imágenes obtenidas se puede comprobar que el resultado del algoritmo *área* de la librería OpenCV obtiene el resultado cualitativamente más parecido a la imagen original, por lo que se hará uso de dicho algoritmo para llevar a cabo todas aquellas tareas de ***downsampling*** precisas durante el tratamiento de las imágenes.

5.6.2 TFRecords y API dataset

Inicialmente se implementó un código para de leer de disco cada una de las imágenes del conjunto y cargarlas en memoria, sin embargo, a medida que las dimensiones del conjunto de imágenes aumentan el tiempo de carga aumenta de forma exponencial conllevando alrededor de 3 minutos la carga de 440 imágenes. Teniendo en cuenta la dimensión del conjunto de imágenes será preciso explorar una alternativa más eficiente.

Tensorflow es capaz de generar un archivo binario (*.tfrecords)* en base a un conjunto de imágenes dado, sea cual sea su dimensión. De este modo todo el conjunto está almacenado en un mismo bloque de memoria, a diferencia del caso anterior, en que cada imagen está almacenada de forma separada.

Esto, por tanto, reducirá el tiempo consumido por parte del acceso a cada uno de los archivos, aún más notable en la máquina utilizada para realizar experimentos, ya que las imágenes se encuentran cargadas en un disco hdd en vez de un disco ssd debido a la gran cantidad de espacio de almacenamiento que requiere el conjunto.

En el ANEXO II se puede observar el código desarrollado para generar los tres archivos tfrecord (para entrenamiento, validación y testing). Para ello básicamente almacena las rutas de las imágenes, les asigna una clase en función del directorio en el que se encuentran

dichas imágenes, se cargan y redimensionan en memoria mediante OpenCV (tal y como se indicó anteriormente) y se almacenan mediante funciones específicas de Tensorflow.

Una vez se hayan generado dichos archivos, la carga de las imágenes se simplifica de forma notable, precisando de una única función de parseado y un objeto tipo *dataset* perteneciente a la API *dataset* de Tensorflow. Esta API proporciona una forma sencilla de cargar y manipular toda la información relativa de un conjunto de imágenes en un mismo objeto.

Además, permite aplicar transformaciones a cada uno de los elementos del conjunto de imágenes mediante la función *map*. A esta función se le proporcionará la función que implementa aquellas transformaciones a realizar sobre el conjunto (como el parseado de las imágenes u operaciones de *data augmentation*), además del número de hilos en los que se dividirá dicha tarea, aportando así un mayor rendimiento al de añadir paralelismo.

6 Experimentación con la red de Tao

Debido a que este proyecto se basa en un proceso de investigación sobre un problema no resuelto hasta la fecha, durante el desarrollo de la experimentación se seguirán diversos enfoques para abordar una posible solución a este problema en base a lo establecido en el estudio del estado del arte.

Como enfoque inicial se ha elegido la arquitectura presentada en el artículo [Tao, Zhang, and Wang 2016], ya que consiste en la red más básica del estudio teórico, para llevar a cabo diversos experimentos con objeto de comparar cómo se comporta la red ante distintas variaciones como pueden ser el uso de proporciones de balanceado de clases diferentes o el uso de *dropout* en la red, todo esto haciendo uso de las imágenes de emisiones.

6.1 Experimentos con testing interdías

En el presente apartado se describirá la fase completa de experimentación haciendo uso de distintos días para el entrenamiento y validación que para el testing. A su vez, los experimentos se realizarán empleando las imágenes etiquetadas según la **clasificación de emisiones dudosas como emisiones**.

6.1.1 Experimentos con conjuntos de datos

Inicialmente se deberán establecer los conjuntos de entrenamiento, validación y testing de modo que la red pueda aprender de forma correcta y posteriormente se pueda realizar una evaluación adecuada de su aprendizaje.

6.1.1.1 Experimento inicial

Para realizar el primer experimento se ha hecho uso del **día 2** como conjunto de testing y el **día 1** para el entrenamiento (70%) y la validación (30%) debido a que este día presenta intervalos nublados y soleados proporcionando así una variedad de situaciones**.** Se le ha aplicado al conjunto de entrenamiento y validación la técnica de downsampling, con objeto de obtener la misma proporción de ejemplos para cada clase que la proporción del experimento a reproducir (proporción 1:1) consistente en el experimento realizado en [Tao, Zhang, and Wang 2016].

Se han seleccionado los hiperparámetros de entrenamiento iniciales tomando como base el experimento de [Tao, Zhang, and Wang 2016]. A excepción del valor de decay que ha sido fijado a 0 para una mejor comprensión del comportamiento del entrenamiento de la red de

manera inicial, y el valor de tamaño de lote, que ha sido fijado a 64. Estos hiperparámetros se muestran en la Tabla 11.

Optimizador	Learning rate	Learning rate decay	Momentum	Tamaño de lote	Función de loss de keras	Epochs
SGD	0,01	0,00	0,9	64	Categorical crossentropy	2000

Tabla 11. Hiperparámetros del primer experimento con Tao2016

En la Figura 53 y Figura 54 se muestran los resultados obtenidos de las métricas a optimizar durante el entrenamiento que, en este caso, se tratan de las métricas por defecto accuracy y loss. Las gráficas en dichas figuras consisten en una representación de los valores que toman las métricas loss y accuracy para cada una de las 2000 epochs que conforman la fase de entrenamiento y validación de la red.

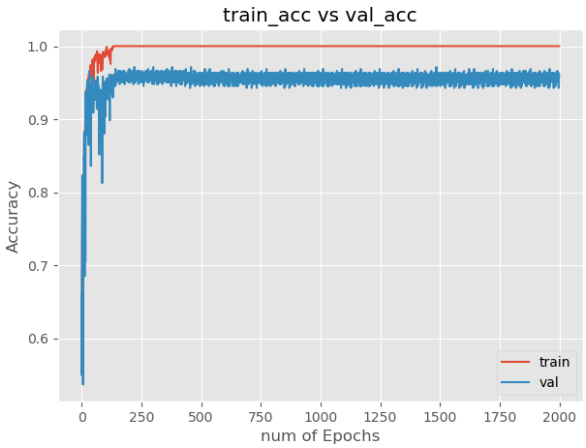

Figura 53. Accuracy de primer experimento Tao

Figura 54. Loss de primer experimento Tao

Como se puede observar en ambas gráficas, todos los valores se estabilizan rápidamente a excepción de los correspondientes a la validación de la métrica loss que, a partir del instante en el que se estabilizan el resto de los valores, este comienza a empeorar. Esto podría ser el resultado de un problema de **overfitting** en el que la red se está sobreentrenando.

Por otro lado, a pesar de que los valores de entrenamiento convergen a su valor óptimo, los resultados de validación se quedan estancados con una diferencia notable a los de entrenamiento. Esto se podría deber a la **insuficiencia de ejemplos** de una o ambas clases, o a **problemas de etiquetado** ya que, tal y como se indicó anteriormente, el etiquetado de una emisión no es una labor que se pueda llevar a cabo de forma exacta sin probabilidad de error.

Una vez se han evaluado y razonado los resultados obtenidos durante el entrenamiento se dispondrá a comprobar las hipótesis establecidas en base a los resultados obtenidos tras llevar a cabo el testing de la red con menor loss resultante del entrenamiento.

Teniendo en cuenta que el conjunto de datos de testing se encuentra altamente desbalanceado y el interés del sistema final consiste en que no cuente con una alta tasa de falsas alarmas, el proceso de testing de la red se llevará a cabo obteniendo las métricas de recall y precisión tomando distintos umbrales para cada 0,05 unidades, evaluando así 19 umbrales en total. Se realizará la elección del umbral óptimo en base a seleccionar aquel que aporte unos valores de precisión y recall más cercanos entre sí, tomando (en caso de que la precisión supere al recall para estos umbrales) aquel umbral en el que la precisión sea inmediatamente superior al recall. Esto permitirá obtener una proporción de falsos positivos y falsos negativos relativamente balanceada, así como el resto de métricas, lo que facilitará la posterior tarea de comparación de experimentos.

En la Figura 55 se muestran los valores de recall y precisión obtenidos para 9 de los umbrales evaluados. En base a estos umbrales, se selecciona el umbral óptimo siguiendo el criterio previamente definido. En este caso se escoge el umbral 0,95, ya que, a pesar de que los valores de precisión no superen a los de recall, la diferencia entre ambas métricas es menor que para el resto de los umbrales. En la Figura 56 se muestran las métricas obtenidas para el umbral 0,95 y la matriz de confusión presentada en la Figura 57, en la que se puede observar que el sistema genera una gran cantidad de falsos positivos, siendo muy superior a la proporción de verdaderos positivos.

A partir de estas métricas se obtiene la curva ROC resultante de la evaluación del modelo, así como el valor de AUC. Esta información se muestra en la Figura 58 haciendo zoom en la esquina izquierda de la gráfica para poder observar fácilmente la distancia de la curva ROC al punto óptimo.

Finalmente, la Figura 59 muestra los resultados de testing obtenidos de la posterior predicción de secuencias basada en la predicción de imágenes de forma independiente. Para esta predicción se lleva a cabo un análisis del umbral óptimo para la predicción de secuencias, pudiendo diferir del umbral utilizado en la predicción de imágenes.

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	79.73	77.41	76.41	76.08	74.75	73.42	72.09	70.76	66.78
PPV	5.94	6.97	7.85	8.75	9.59	10.47	11.71	13.52	16.6

Figura 55. Umbrales para el primer experimento

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.95	19.43	61.13	99.28	95.49	29.49	94.88

Figura 56. Métricas del primer experimento

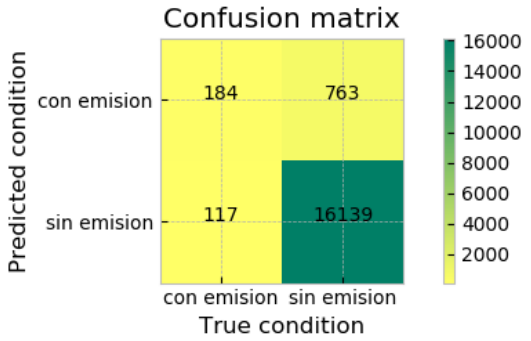

Figura 57. Matriz de confusión del primer experimento

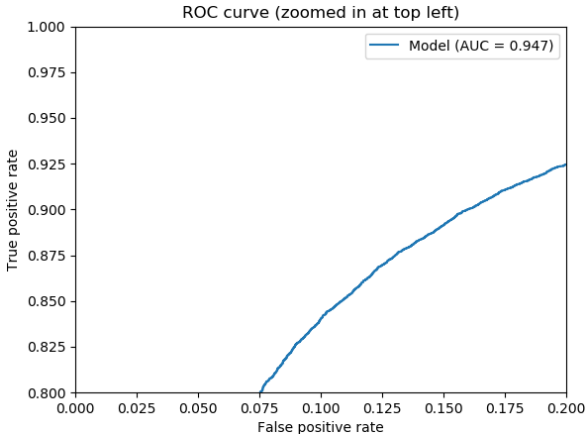

Figura 58. Curva ROC del primer experimento

	TP	FP	FN
0.95	24	82	6

Figura 59. Predicción de secuencias del primer experimento

Tras la realización del experimento se ha comprobado que, a pesar de que cada epoch conlleva aproximadamente 1 segundo, debido a que el experimento consta de 2.000 epochs el tiempo requerido para llevar a cabo la fase de entrenamiento es de aproximadamente una hora para un número de ejemplos relativamente reducido (menos de 1.000 ejemplos por clase). Sin embargo, de los valores de entrenamiento se puede observar que la red converge durante las epochs iniciales, por lo que no es necesario realizar un número tan alto de epochs para entrenar la red, ya que esto únicamente conlleva un agravamiento de la situación de overfitting.

Debido a esta diferencia entre los valores de validación y de entrenamiento, los resultados de testing cuentan con una alta proporción de predicciones incorrectas. Por tanto, la fase de experimentación continuará con la búsqueda de una solución al problema de overfitting existente.

6.1.1.2 Proporción de clases real

Tal y como se indicó anteriormente, una de las posibles razones causantes de esta diferencia entre los resultados de validación y testing con los de entrenamiento, puede deberse a una situación de insuficiencia de ejemplos de entrenamiento.

En base a esto, el siguiente experimento a realizar se basará en entrenar la red en base a la proporción real de ejemplos de modo que se cuente con más ejemplos sin emisiones permitiendo, por tanto, que la red aprenda a identificar mejor aquellas situaciones en las que no hay emisiones.

Para ello se reproducirá el mismo experimento que el anterior, a diferencia de la cantidad de ejemplos sin emisión utilizados para el entrenamiento de la red, que esta vez se tratará de la proporción real y no la similar al experimento realizado en [Tao, Zhang, and Wang 2016].

En la Figura 60 y Figura 61 se muestran los resultados obtenidos del entrenamiento de la red en el que, aunque los resultados de validación sigan siendo inferiores a los de entrenamiento sin llegar a converger, se puede comprobar que los valores difieren menos que para la proporción 1:1 ya que los valores de accuracy son más próximos a 1 y los de loss más cercanos a 0. Sin embargo, al tratarse de un conjunto desbalanceado la métrica accuracy ya no es representativa.

Como resultado de la evaluación de la red se han obtenido la Figura 62, Figura 63 y Figura 64. De estas figuras se puede comprobar que la tasa de falsos positivos ha decrecido de forma notable, sin embargo, esto también ha conllevado un incremento de falsos negativos.

En la Figura 65 se muestra la curva ROC asociada al experimento, así como el valor de AUC, el cual ha ascendido. Esto se debe a que, a pesar del incremento de falsos negativos, el ratio de falsas alarmas ha decrementado de forma notable, comportamiento que se hace evidente en la Figura 66 que presenta un número de falsas alarmas muy inferior al del experimento anterior.

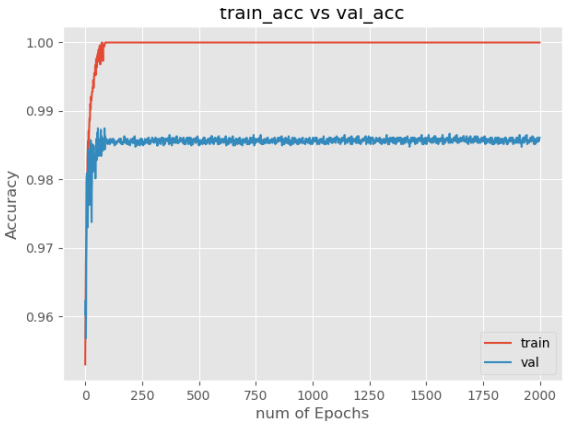

Figura 60. Accuracy de Tao2016 con proporción real

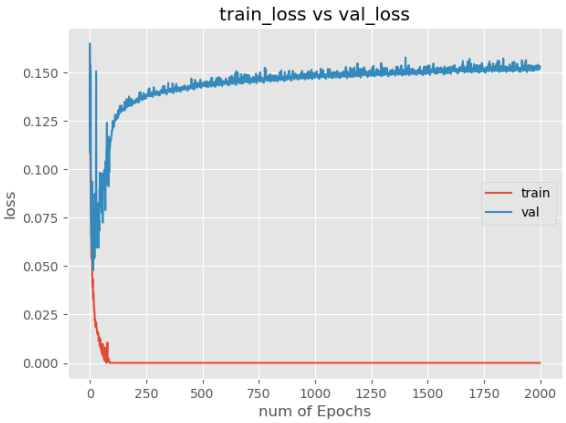

Figura 61. Loss de Tao2016 con proporción real

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	74.42	68.77	66.78	64.45	62.13	59.47	57.48	54.49	49.17
PPV	19.72	22.75	25.9	28.87	30.81	32.49	35.09	39.33	43.92

Figura 62. Umbrales de Tao2016 con proporción real

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.95	47.67	44.19	99.01	99.14	45.86	98.17

Figura 63. Métricas de Tao2016 con proporción real

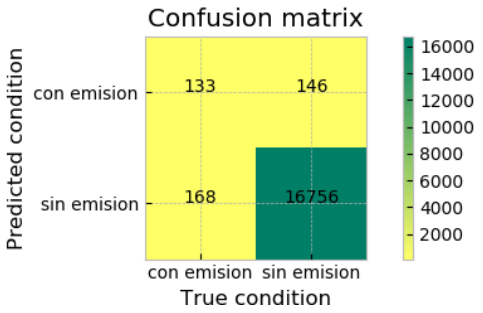

Figura 64. Matriz de confusión de Tao2016 con proporción real

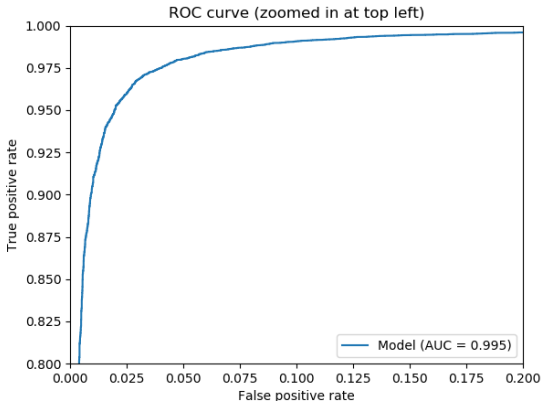

Figura 65. Curva ROC de Tao2016 con proporción real

	TP	FP	FN
0.95	17	12	13

Figura 66. Predicción de secuencias de Tao2016 con proporción real

Como conclusión, se ha podido comprobar que aportando más ejemplos de la clase que no presenta emisiones se ha permitido mejorar de forma notable la identificación de situaciones en las que no se dan emisiones. A pesar de que esto ha conllevado un ascenso de falsos negativos y, por tanto, descenso de verdaderos positivos y de los valores de métricas como recall y F1-score entre otras, la proporción final de predicciones incorrectas es menor.

6.1.1.3 Triplicado de ejemplos positivos

En base a la comprobación de que el uso de más ejemplos de una clase proporciona un mejor rendimiento a la hora de detectar dicho ejemplo, se procederá a llevar a cabo un experimento con el triple de ejemplos con emisión que hasta ahora. Para ello se han incluido al conjunto de entrenamiento y validación los días 10 y 19.

En cuanto a la proporción de ejemplos por cada clase, se hará uso de una proporción 1:1 eliminando de forma aleatoria ejemplos sin emisión, con objeto de que la métrica accuracy (utilizada durante el entrenamiento para la optimización de la red) sea representativa de su respectivo rendimiento.

Además, se reducirá el número de epochs a 150, ya que en base a los anteriores experimentos se puede observar que la red converge aproximadamente sobre ese número de epochs para un conjunto de dimensiones y características similares al del actual experimento, como el conjunto del primer experimento.

En la Figura 67 y Figura 68 se muestra el resultado del entrenamiento del presente experimento, pudiendo comprobar que los valores de loss son similares a los obtenidos en el primer experimento, teniendo en cuenta el decremento de epochs.

En la Figura 69, Figura 70, Figura 71, Figura 72 y Figura 73 se presentan los resultados de testing obtenidos a partir del modelo resultante del entrenamiento. Se puede observar que al aumentar el número de ejemplos con emisión la proporción de falsos negativos ha bajado, lo que supone un ascenso en el número de verdaderos positivos, obteniendo una precisión muy próxima al 50%.

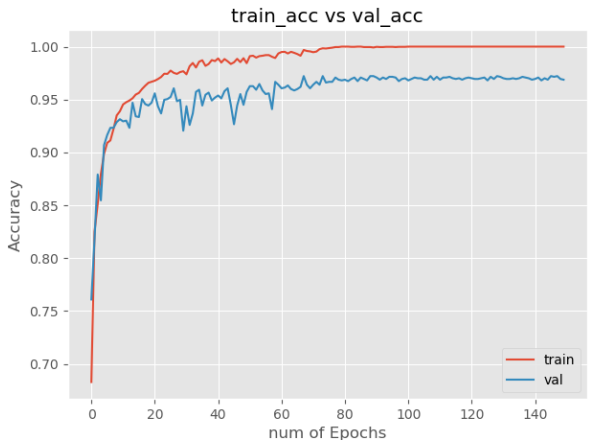

Figura 67. Accuracy de Tao con triplicado de ejemplos

Figura 68. Loss de Tao con triplicado de ejemplos

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	84.39	81.06	79.07	77.41	75.08	71.43	68.44	63.79	57.14
PPV	12.63	15.58	18.27	20.35	22.67	24.68	27.88	31.32	36.99

Figura 69. Umbrales de Tao con triplicado de ejemplos

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.95	41.85	49.5	99.1	98.78	45.36	97.91

Figura 70. Métricas de Tao con triplicado de ejemplos

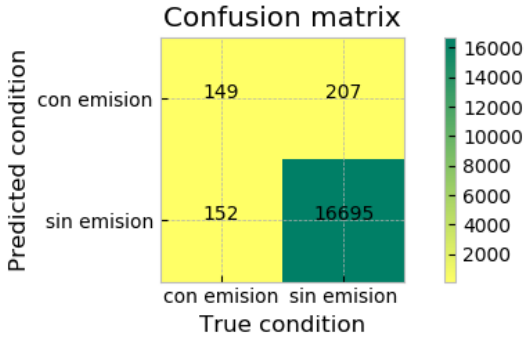

Figura 71. Matriz de confusión de Tao con triplicado de ejemplos

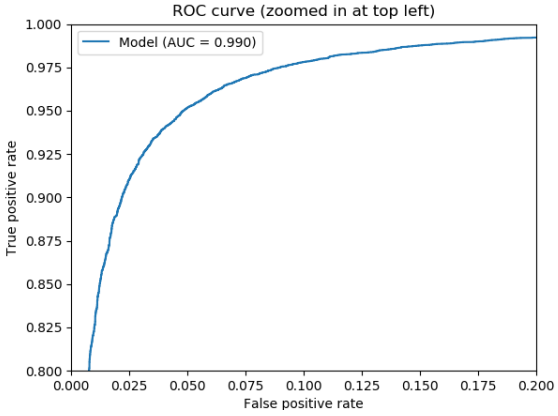

Figura 72. Curva ROC de Tao con triplicado de ejemplos

	TP	FP	FN
0.95	19	29	11

Figura 73. Predicción de secuencias de Tao con triplicado de ejemplos

Mediante este experimento se han conseguido optimizar todas las métricas en comparación al primer experimento, que hace uso de la misma proporción de clases. Sin embargo, previamente se ha comprobado que el uso de un desbalanceo de clases permite optimizar ciertas métricas de la red, por lo que se deberá realizar posteriormente una serie de experimentos en base a la obtención del desbalanceo óptimo.

6.1.2 Experimentos con modificaciones en la arquitectura de la red

Una vez se ha explorado el comportamiento de la red ante las distintas proporciones de ejemplos y la dimensión de los conjuntos, se hará uso de la configuración obtenida del último experimento de la sección anterior para comenzar a explorar el comportamiento de la red ante la modificación de la arquitectura en búsqueda de posibles optimizaciones.

Debido a que esta red es la más sencilla de las expuestas en el estudio teórico, las modificaciones se basarán en la simplificación de la arquitectura, ya que, en caso de querer optar por una red más compleja, se evaluarán el resto de las redes expuestas en el estudio.

6.1.2.1 Eliminación de las capas de dropout

Tomando las conclusiones establecidas en base al experimento inicial se deberá buscar el problema causante del overfitting durante el entrenamiento. En caso de situaciones de overfitting se pueden llevar a cabo diversas tácticas, una de estas es la adición de dropout para eliminar neuronas buscando así mayor generalización por parte de la red a la hora de establecer predicciones.

Debido a que la arquitectura [Tao, Zhang, and Wang 2016] cuenta con capas de dropout se probará a eliminar las dos capas existentes en la red para comprobar si esto supone algún efecto en su comportamiento durante el entrenamiento. Para ello se hará uso de los conjuntos de datos utilizados en el experimento anterior.

En la Figura 74 y Figura 75 se puede observar que la eliminación del dropout no afecta a la existencia de overfitting. Incluso se puede observar que los valores de loss empeoran más lentamente. Sin embargo, durante el experimento de la red con dropout, esta llega a alcanzar valores de los más cercanos a 0, lo que supone que el modelo resultante del entrenamiento contará con una métrica de loss peor que la del experimento anterior.

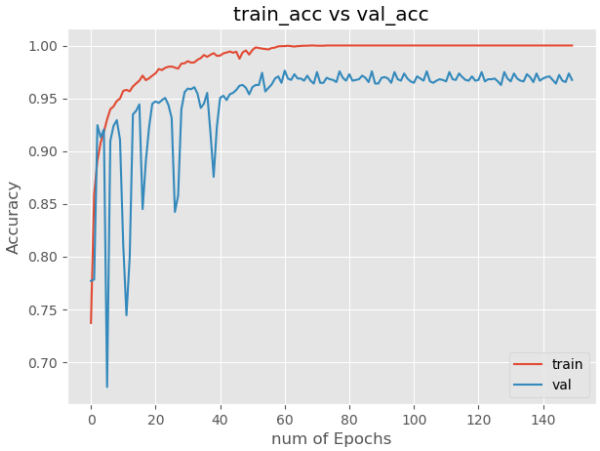

Figura 74. Accuracy de Tao2016 sin dropout

Figura 75. Loss de Tao2016 sin dropout

Tal y como muestran la Figura 76, Figura 77, Figura 78, Figura 79 y Figura 80 este experimento obtiene peores resultados para ambas métricas de precisión y recall, originando en un mayor número de falsos positivos y de falsos negativos.

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	78.07	75.08	73.75	71.43	69.77	68.11	66.11	64.78	61.46
PPV	11.16	13.14	14.74	15.83	16.91	18.14	19.21	21.31	23.57

Figura 76. Umbrales proporción real

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.95	26.2	57.81	99.23	97.1	36.06	96.41

Figura 77. Métricas para la proporción real

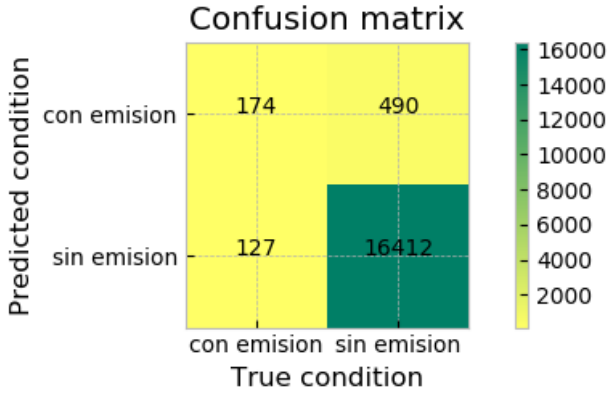

Figura 78. Matriz de confusión para la proporción real

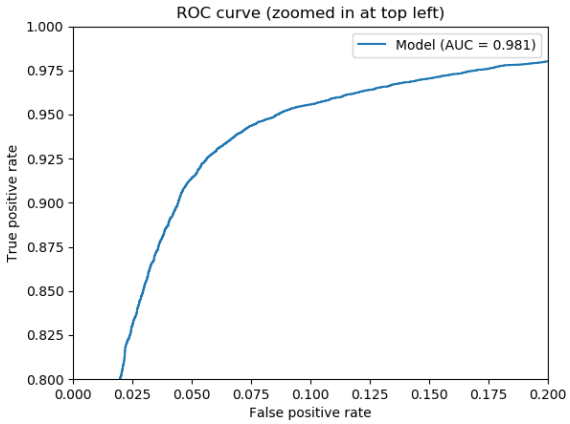

Figura 79. Curva ROC de Tao2016 sin dropout

	TP	FP	FN
0.95	24	62	6

Figura 80. Predicción de secuencias de Tao2016 sin dropout

Tras llevar a cabo este experimento se ha podido comprobar que, a pesar de que los resultados varían levemente, las capas de dropout no están favoreciendo la reducción de overfitting durante el entrenamiento, ya que los resultados de validación siguen empeorando a partir de cierto punto del entrenamiento. Debido a esto, se deberán llevar a cabo otras estrategias durante los siguientes experimentos con objeto de solucionar este problema.

6.1.2.2 Reducción de capas convolucionales

Una vez realizados diversos experimentos con distintas proporciones de ejemplos y balanceado, así como el uso de dropout, se continuará la fase de experimentación mediante la modificación de la arquitectura de la red.

Esta primera variación de la red consistirá en la reducción de la arquitectura mediante la modificación del tercer nivel de capas convolucionales. Esta modificación consistirá en la eliminación de las dos primeras capas convolucionales de dicho nivel, dejando únicamente la última capa, manteniendo así el número de características.

En este experimento se mantendrán los conjuntos de datos utilizados en los anteriores experimentos.

En la Figura 81 y Figura 82 se puede comprobar que ante la reducción de capas convolucionales la métrica de loss toma valores más bajos y, a pesar de que empeore levemente en función a las epochs transcurridas, este agravamiento es más leve que en los anteriores experimentos.

En la Figura 83, Figura 84, Figura 85, Figura 86 y Figura 87 se presentan los resultados de testing, pudiendo comprobar que, a pesar de que el número de falsos positivos aumentase, la proporción de falsos negativos ha decrementado de forma más notable, proporcionando una mayor proporción de verdaderos positivos y mejorando así la métrica de recall.

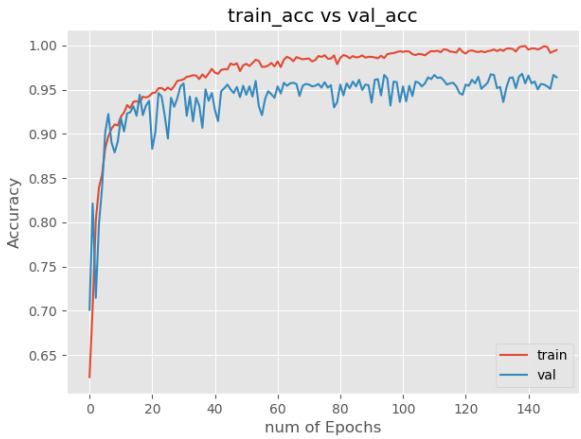

Figura 81. Accuracy de Tao sin capas convolucionales 3 y 4

Figura 82. Loss de Tao sin capas convolucionales 3 y 4

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	91.36	87.38	84.05	81.06	79.4	76.74	73.42	70.76	63.12
PPV	9.88	12.56	14.57	16.62	18.69	21.17	24.05	28.36	34.8

Figura 83. Umbrales de Tao sin capas convolucionales 3 y 4

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.95	41.79	54.15	99.18	98.66	47.18	97.88

Figura 84. Métricas de Tao sin capas convolucionales 3 y 4

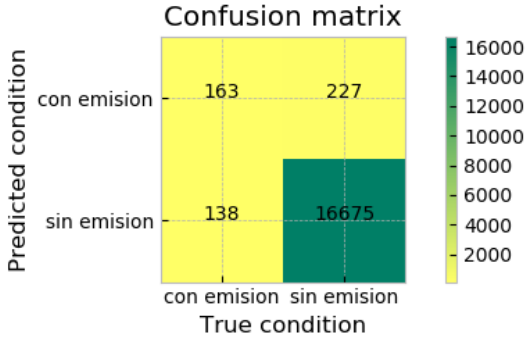

Figura 85. Matriz de confusión de Tao sin capas convolucionales 3 y 4

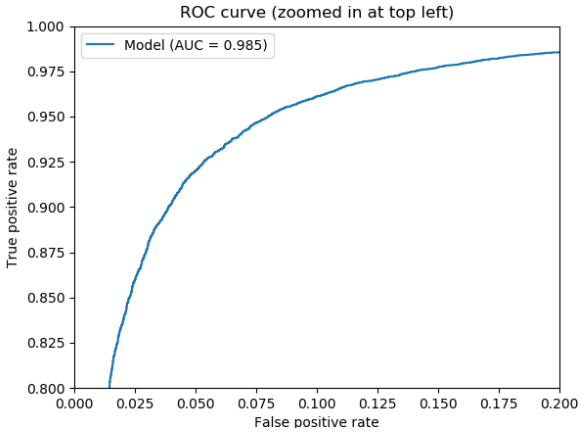

Figura 86. Curva ROC de Tao sin capas convolucionales 3 y 4

	TP	FP	FN
0.95	19	18	11

Figura 87. Predicción de secuencias de Tao sin capas convolucionales 3 y 4

Tras evaluar los resultados obtenidos de este experimento, se puede concluir que todas y cada una de las métricas han sido optimizadas tras la reducción de la arquitectura de la red mediante la eliminación de las dos primeras capas del tercer nivel convolucional, a excepción de pequeñas disminuciones de las métricas precisión, true negative rate y accuracy.

6.1.2.3 Reducción de neuronas en capas totalmente conectadas

Siguiendo con el enfoque tomado en el experimento anterior, se dispondrá a realizar una nueva modificación a la arquitectura de la red. Para ello se partirá de la red original (con todas sus capas convolucionales y de dropout), a la que se reducen las 4096 neuronas de sus dos capas totalmente conectadas a la mitad de las neuronas, quedando en un total de 2048.

En la Figura 88 y Figura 89 se muestran los resultados de entrenamiento, en los que se puede comprobar que los valores de loss han empeorado con relación a los anteriores experimentos.

En la Figura 90, Figura 91, Figura 92, Figura 93 y Figura 94 se puede comprobar que todas las métricas han empeorado en cuanto al experimento sin modificaciones en la red y al experimento con reducción de capas convolucionales, a excepción de las métricas recall y negative predictive value.

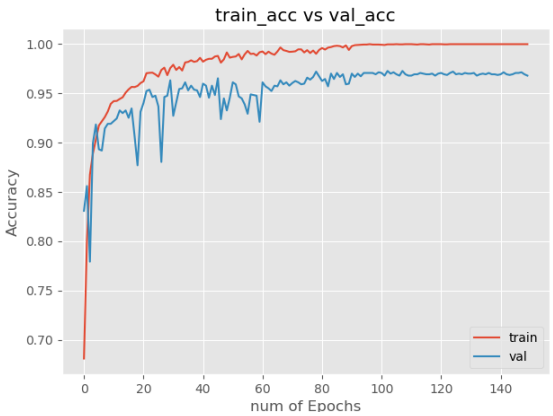

Figura 88. Accuracy de Tao con reducción de neuronas

Figura 89. Loss de Tao con reducción de neuronas

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	82.39	81.4	80.07	79.4	78.07	77.08	74.09	69.77	66.78
PPV	11.46	13.12	14.5	15.63	16.85	18.07	19.03	21.02	25.0

Figura 90. Umbrales de Tao con reducción de neuronas

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.95	28.73	60.13	99.28	97.34	38.88	96.69

Figura 91. Métricas de Tao con reducción de neuronas

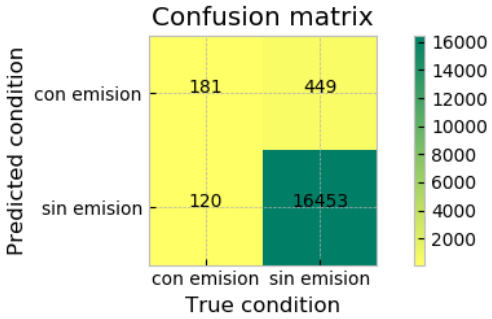

Figura 92. Matriz de confusión de Tao con reducción de neuronas

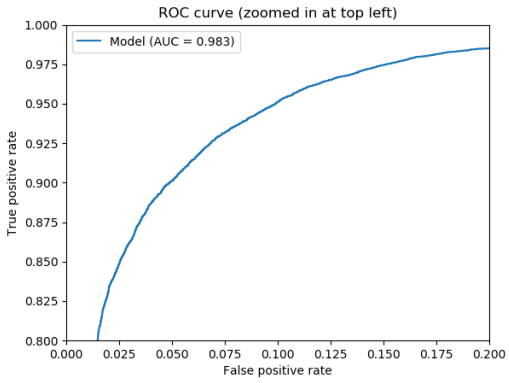

Figura 93. Curva ROC de Tao con reducción de neuronas

	TP	FP	FN
0.95	24	48	6

Figura 94. Predicción de secuencias de Tao con reducción de neuronas

Este experimento ha podido comprobar que la reducción de neuronas en las capas totalmente conectadas conlleva un agravamiento significativo de la gran mayoría de las métricas aumentando de forma notable el número de falsas alarmas, comportamiento contrario al deseado. Por tanto, se puede concluir que la reducción de neuronas no supone una mejora del rendimiento de la red, sino de todo lo contrario, reduciendo su capacidad de clasificación.

6.1.2.4 Reducción de capas convolucionales y de neuronas en capas totalmente conectadas

El siguiente experimento que se llevará a cabo consistirá en la combinación de los anteriores experimentos, realizando el entrenamiento y la evaluación de la red con ambas modificaciones de la arquitectura.

El resto de hiperparámetros de entrenamiento, proporción de balanceado entre clases y conjuntos de testing validación y entrenamiento se mantienen en cuanto a los anteriores experimentos.

En la Figura 95 y Figura 96 se puede comprobar que el comportamiento de los valores de loss que se observó en el anterior experimento al reducir el número de neuronas, se corrige al añadir la reducción de capas neuronales, obteniendo resultados similares a los obtenidos en el experimento en el que se aplica únicamente la modificación de reducción de convoluciones. A partir de la Figura 97, Figura 98, Figura 99, Figura 100 y Figura 101 se puede observar que los resultados se encuentran en un punto intermedio entre los obtenidos en los dos anteriores experimentos. Sin embargo, la reducción de capas convolucionales obtiene en general mejores valores que los del presente experimento.

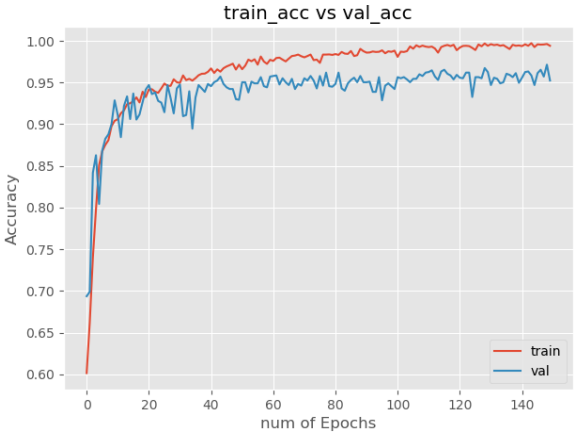

Figura 95. Accuracy de Tao con reducción de convoluciones y de neuronas

Figura 96. Loss de Tao con reducción de convoluciones y de neuronas

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	84.72	82.06	79.07	78.41	76.41	72.43	68.77	62.46	57.48
PPV	17.42	19.86	21.42	23.39	25.33	26.46	27.75	30.13	36.04

Figura 97.Umbrales de Tao con reducción de convoluciones y de neuronas

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.95	40.43	50.5	99.11	98.67	44.9	97.83

Figura 98. Métricas de Tao con reducción de convoluciones y de neuronas

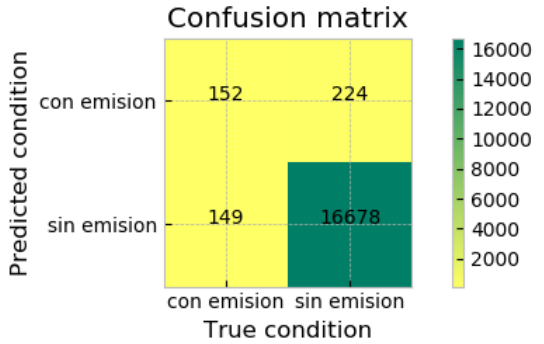

Figura 99. Matriz de confusión de Tao con reducción de convoluciones y de neuronas

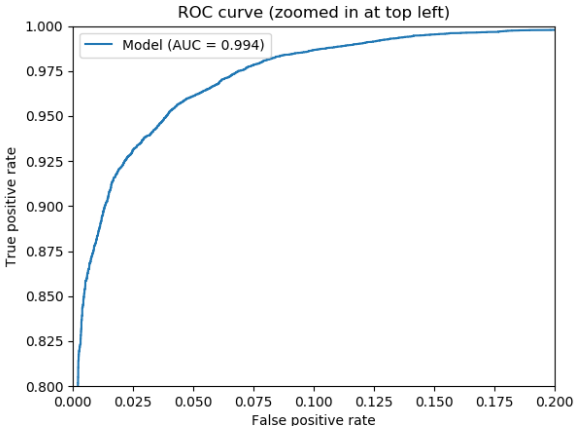

Figura 100. Curva ROC de Tao con reducción de convoluciones y de neuronas

	TP	FP	FN
0.95	21	19	9

Figura 101. Predicción de secuencias de Tao con reducción de convoluciones y de neuronas

Comparando los resultados de los experimentos anteriores, se puede concluir que al realizar ambas modificaciones obtienen mejores resultados que haciendo uso de la arquitectura sin modificaciones. Sin embargo, empeoran en comparación con los obtenidos mediante la única aplicación de la reducción de capas convolucionales. Lo que confirma que la red empeora al reducir el número de neuronas de las capas convolucionales. A pesar de esto, el número de secuencias predichas es mayor a costa de aumentar el número de falsas alarmas. Debido a que el principal objetivo es minimizar los falsos positivos, este enfoque no se tomará como óptimo.

6.1.2.5 Reducción de capas convolucionales y modificación de filtros

En base a los resultados obtenidos hasta ahora aplicando modificaciones a la arquitectura, se puede comprobar que la modificación de convoluciones ha proporcionado una optimización de la red.

En base a esto el siguiente experimento tomará como arquitectura base la que elimina la tercera y cuarta capa convolucional y se procederá a reducir a la mitad el número de filtros de las capas convolucionales que conforman los dos primeros niveles convolucionales de la arquitectura de la red, manteniendo así el número de características de la red.

Resultados del experimento

En la Figura 102 y Figura 103 se puede observar que los resultados obtenidos para ambas métricas son muy similares, siendo los valores de loss de las últimas epochs levemente superiores.

En la Figura 104, Figura 105, Figura 106, Figura 107 y Figura 108 se muestran los resultados de testing de la red. Se puede observar que los resultados son similares a los obtenidos mediante la reducción de capas convolucionales, contando con un valor superior de recall, negative predictive value y AUC, pero inferior de precisión, true negative rate, f1-score y accuracy. Sin embargo, el aumento de falsos positivos se vuelve muy notable tanto en la predicción de imágenes de forma independiente como en la predicción de secuencias.

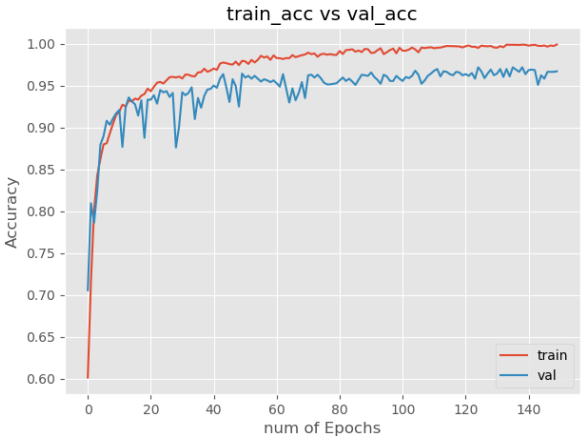

Figura 102. Accuracy de Tao con reducciones de capas convolucionales y reducción de filtros

Figura 103. Loss de Tao con reducciones de capas convolucionales y reducción de filtros

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	89.7	85.38	83.06	80.4	78.41	75.42	73.42	71.43	66.78
PPV	12.91	15.2	17.28	18.98	20.63	22.12	24.29	27.28	32.16

Figura 104. Umbrales de Tao con reducciones de capas convolucionales y reducción de filtros

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.95	36.62	60.47	99.29	98.14	45.61	97.48

Figura 105. Métricas de Tao con reducciones de capas convolucionales y reducción de filtros

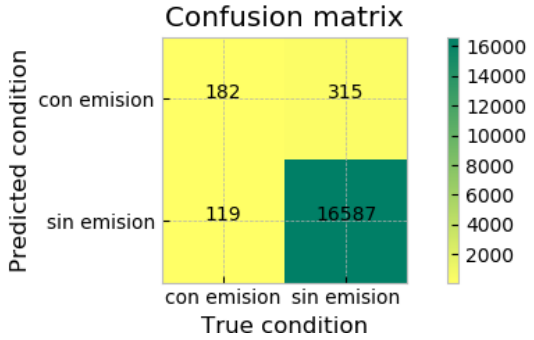

Figura 106. Matriz de confusión de Tao con reducciones de capas convolucionales y reducción de filtros

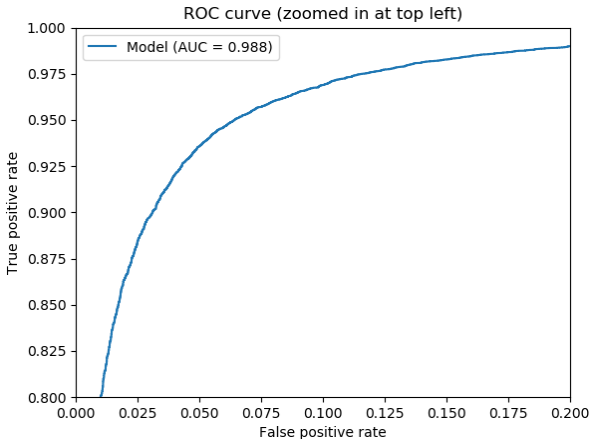

Figura 107. Curva ROC de Tao con reducciones de capas convolucionales y reducción de filtros

	TP	FP	FN
0.95	20	29	10

Figura 108. Predicción de secuencias de Tao con reducciones de capas convolucionales y reducción de filtros

A partir de este experimento se comprueba que la modificación del número de filtros en capas convolucionales supone un aumento altamente notable de falsos positivos, frente a un leve incremento de verdaderos positivos. Debido a que, tal y como se indicó anteriormente numerosas veces, se desea minimizar el número de falsas alarmas, esta modificación no supone una mejora en cuanto a las predicciones realizadas por la red.

6.1.3 Experimentos con desbalanceo de clases e hiperparámetros

Una vez realizados varios experimentos poniendo en práctica la modificación de la red [Tao, Zhang, and Wang 2016] sin llegar a realizar cambios demasiado drásticos que transformen su arquitectura en otra totalmente distinta, se procederá a realizar más experimentos en base a la proporción de clases y número de ejemplos utilizados para el entrenamiento y la validación de la red, ya que en la primera sección dedicada a la exploración de los conjuntos de datos a utilizar en los experimentos se pudo observar que la utilización de un conjunto con una proporción distinta a la 1:1 supone un incremento en todas y cada una de las métricas.

Estos experimentos partirán de la arquitectura resultante del experimento consistente en la eliminación de capas convolucionales.

6.1.3.1 Desbalanceo con proporción 5:1

Se ha de tener en cuenta que el conjunto de datos sobre el que se está trabajando actualmente consta de tres días. Asimismo, aunque en los experimentos iniciales se comprobó que la proporción real obtiene mejores resultados que la 1:1, esto no quiere decir que otras proporciones intermedias de balanceado sean peores.

En base a esto, se realizará un experimento con una proporción de clases 5:1 en los conjuntos de entrenamiento y validación, haciendo uso de los mismos hiperparámetros y conjuntos de imágenes que en los experimentos anteriores.

En la Figura 109 y Figura 110 se observa que tanto los valores de loss como los de accuracy no convergen durante el entrenamiento. El valor de accuracy parece indicar que se están clasificando todos los ejemplos como la clase mayoritaria, en este caso, por tanto, estaría clasificando todos como imágenes sin emisión.

Una vez realizada la evaluación de la Figura 111 y Figura 112 se puede comprobar que la red clasifica todas la imágenes evaluadas como no emisiones.

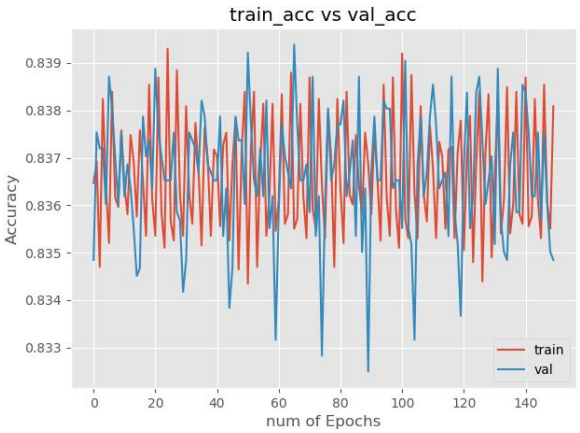

Figura 109. Accuracy de Tao con proporción 5:1

Figura 110. Loss de Tao con proporción 5:1

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
PPV	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figura 111. Umbrales de Tao con proporción 5:1

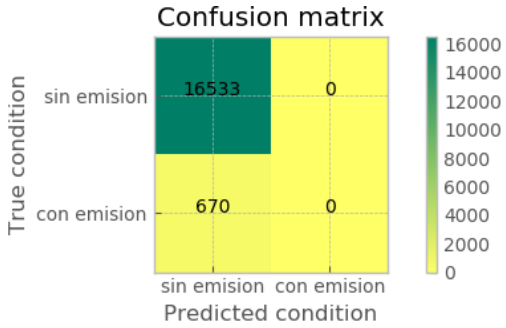

Figura 112. Matriz de confusión de Tao con proporción 5:1

En el presente experimento se puede comprobar que haciendo uso del conjunto actual con más días y ejemplos más dispares la red se comporta de forma distinta al utilizar una proporción distinta a 1:1, dando como resultado que la red no converge durante el entrenamiento si no que diverge. Este comportamiento puede deberse a que los cambios que sufren los pesos en cada iteración son demasiado grandes debido a un valor de learning rate alto. Muchos experimentos se comienzan con un learning rate alto que se va decrementando de forma exponencial, obteniendo un resultado similar al de este experimento en la primeras iteraciones que hacen uso de un valor alto de learning rate y convergiendo durante las iteraciones que hacen uso de un valor de learning rate menor.

6.1.3.2 Desbalanceo con proporción 5:1 y reducción de learning rate

En base a las conclusiones obtenidas del anterior experimento, se procederá a decrementar el valor de learning rate de 0,01 a 0,001 con objeto de comprobar si el comportamiento de la red en dicho experimento se debe a este valor, a pesar de que esto pueda afectar de forma negativa a la optimización del entrenamiento en términos de tiempo de ejecución.

En la Figura 113 y Figura 114 se observa que la red vuelve a converger durante el entrenamiento al reducir el valor de learning rate, obteniendo valores de loss cercanos a 0,05.

En la Figura 115, Figura 116, Figura 117, Figura 118 y Figura 119 se observa que, en general, las métricas han mejorado en comparación con los experimentos anteriores realizados con una proporción de clases 1:1. A pesar de que la métrica recall hasta disminuido levemente, la métrica precisión ha aumentado. Sin embargo, la predicción de secuencias en base a la predicción de imágenes muestra unos resultados opuestos, disminuyendo el número de falsos negativos a cosa de un aumento de falsos positivos, comportamiento no deseado en el sistema.

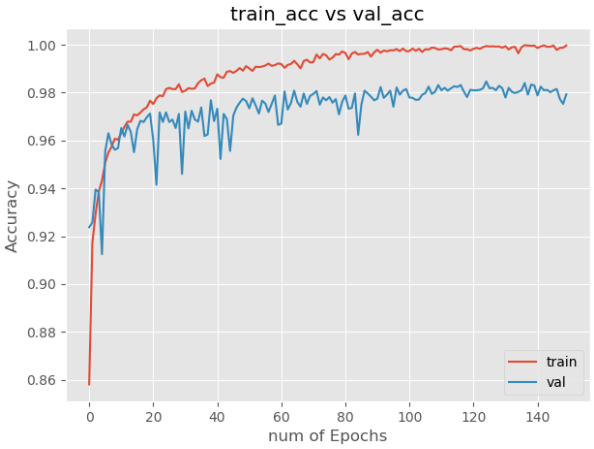

Figura 113. Accuracy de Tao con proporción 5:1 y reducción de learning rate

Figura 114. Loss de Tao con proporción 5:1 y reducción de learning rate

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	67.11	64.78	64.12	62.13	62.13	61.13	59.8	57.81	56.15
PPV	26.03	28.89	30.98	32.47	34.76	35.94	37.5	38.84	42.14

Figura 115. Umbrales de Tao con proporción 5:1 y reducción de learning rate

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.95	45.45	53.16	99.16	98.86	49.0	98.06

Figura 116. Métricas de Tao con proporción 5:1 y reducción de learning rate

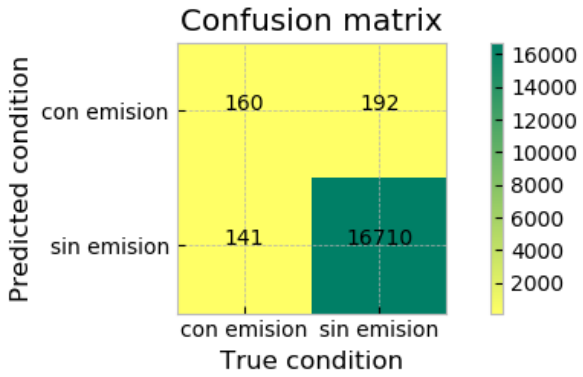

Figura 117. Matriz de confusión de Tao con proporción 5:1 y reducción de learning rate

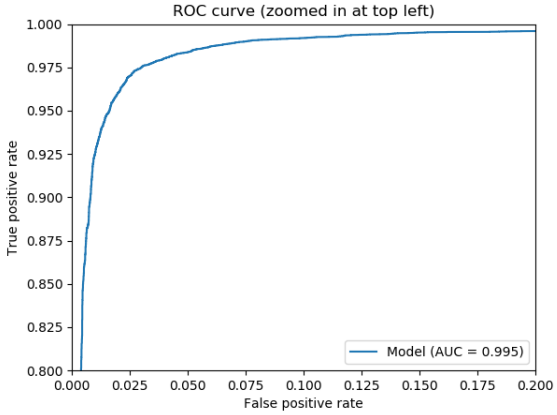

Figura 118. Curva ROC de Tao con proporción 5:1 y reducción de learning rate

	TP	FP	FN
0.95	20	20	10

Figura 119. Predicción de secuencias de Tao con proporción 5:1 y reducción de learning rate

Con este experimento se puede concluir que aplicando un decremento del learning rate y haciendo uso de un conjunto desbalanceado con una proporción 5:1, se ha conseguido aumentar la precisión a nivel de predicción de imágenes de forma independientes, a costa de una leve disminución del recall.

6.1.3.3 Desbalanceo 5:1 y penalización de loss

A pesar de que tras aplicar un desbalanceo al conjunto de datos se han optimizado las métricas, se ha de tener en cuenta lo estudiado previamente en cuanto a la optimización de la métrica accuracy ante conjuntos desbalanceados, ya que no es del todo representativa en estos casos.

Debido a este aspecto, el siguiente experimento a realizar consistirá en modificar la optimización de loss durante el entrenamiento de modo que un error en la clase minoritaria (en este caso se trataría de ejemplos con emisiones) penalizará 5 veces más que un error en la clase mayoritaria (ejemplos sin emisiones).

Se hará uso de los mismos conjuntos e hiperparámetros que los utilizados en el experimento anterior.

En la Figura 120 y Figura 121 se muestran los resultados obtenidos del presente experimentos, en el que se puede observar como los valores de loss son mayores en las primeras iteraciones debido a la penalización de los errores en muestras de la clase minoritaria.

En la Figura 122, Figura 123, Figura 124, Figura 125 y Figura 126 se puede observar que todas las métricas han incrementado levemente, salvo la métrica true negative rate.

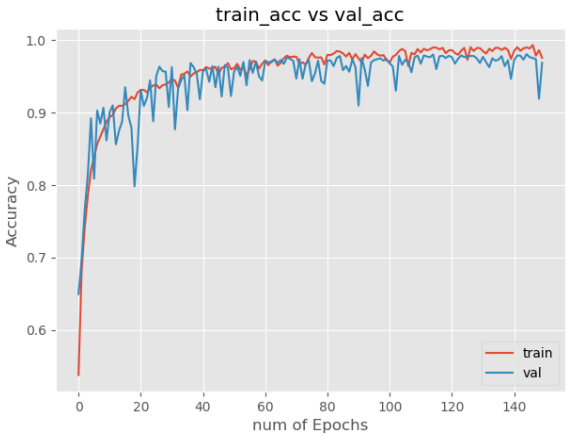

Figura 120. Accuracy de Tao con proporción 5:1 y penalización de loss

Figura 121. Loss de Tao con proporción 5:1 y penalización de loss

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	68.77	66.45	64.45	64.12	64.12	62.79	60.47	59.14	56.81
PPV	28.09	29.54	30.36	31.59	32.71	33.93	34.87	37.87	42.75

Figura 122. Umbrales de Tao con proporción 5:1 y penalización de loss

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.95	45.48	51.83	99.14	98.89	48.45	98.07

Figura 123. Métricas de Tao con proporción 5:1 y penalización de loss

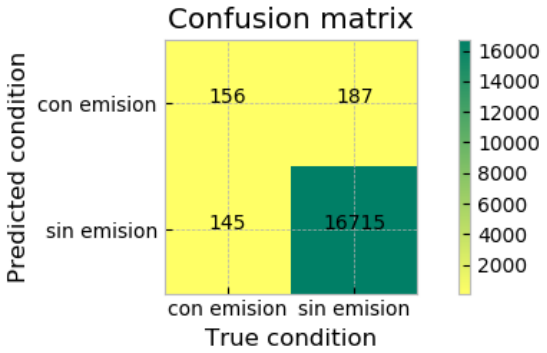

Figura 124. Matriz de confusión de Tao con proporción 5:1 y penalización de loss

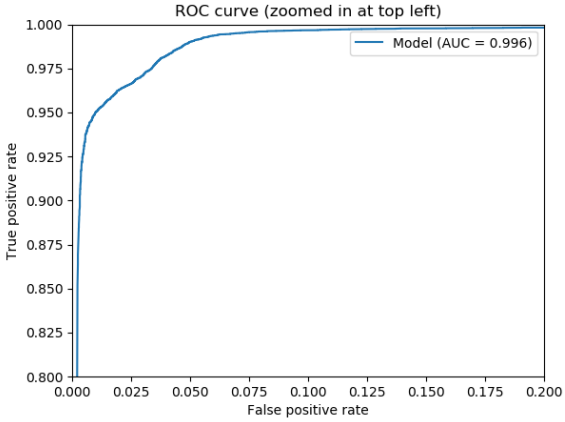

Figura 125. Curva ROC de Tao con proporción 5:1 y penalización de loss

	TP	FP	FN
0.95	19	19	11

Figura 126. Predicción de secuencias de Tao con proporción 5:1 y penalización de loss

Prestando atención a los resultados obtenidos en este experimento y comparándolo con el anterior, se puede concluir que la modificación de la optimización de la función de loss ha supuesto una optimización de la gran mayoría de las métricas en cuanto al experimento anterior. Sin embargo, los resultados siguen sin proporcionar una diferencia sustancial en cuanto a la proporción 1:1 a parte del incremento de precisión.

6.1.3.4 Desbalanceo 5:1 y optimización de la métrica precisión

Al igual que la técnica de penalización de loss en función a la proporción de clases, también se especificó anteriormente el uso de la técnica de optimización de otras métricas.

En base a esto se llevará a cabo un experimento en el que se optimizará la métrica precisión en vez de accuracy, ya que se trata de una métrica de interés para el sistema.

Se mantendrán todos los hiperparámetros y conjuntos de entrenamiento, validación y testing de los anteriores experimentos, sin aplicar la penalización de loss realizada en el anterior experimento.

En la Figura 127 y Figura 128 se muestran los valores del entrenamiento, similares a los obtenidos en el experimento que no hace uso de la penalización de loss.

En la Figura 129, Figura 130, Figura 131, Figura 132 y Figura 133 se puede comprobar que en comparación con el experimento que no hace uso de la penalización de loss, se obtienen mejores valores para la gran parte de las métricas.

Figura 127. Precision de Tao con proporción 5:1 y optimización de precisión

Figura 128. Loss de Tao con proporción 5:1 y optimización de precisión

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	75.08	72.76	71.43	68.77	67.77	66.45	65.78	64.12	60.47
PPV	26.94	29.24	31.2	32.09	33.55	35.03	36.73	38.29	41.36

Figura 129. Umbrales de Tao con proporción 5:1 y optimización de precisión

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.95	44.53	58.14	99.25	98.71	50.43	98.0

Figura 130. Métricas de Tao con proporción 5:1 y optimización de precisión

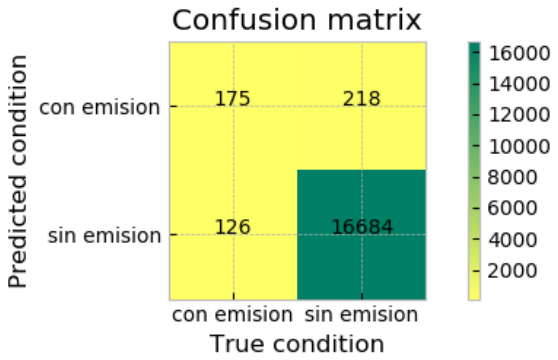

Figura 131. Matriz de confusión de Tao con proporción 5:1 y optimización de precisión

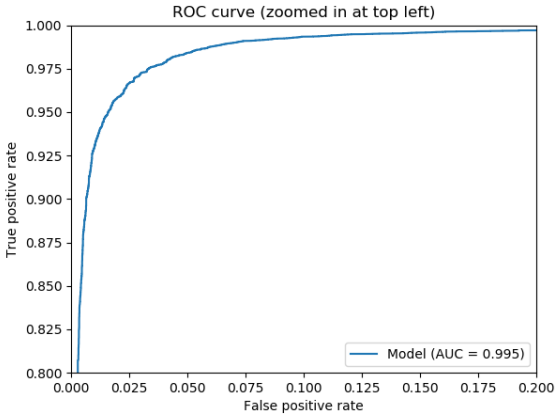

Figura 132. Curva ROC de Tao con proporción 5:1 y optimización de precisión

	TP	FP	FN
0.95	21	18	9

Figura 133. Predicción de secuencias de Tao con proporción 5:1 y optimización de precisión

En base a los resultados obtenidos se puede observar que tras la optimización de la precisión con una proporción 5:1 se ha conseguido mantener el número de falsos positivos a nivel de predicción de secuencias aumentando el número de verdaderos positivos, en cuanto al experimento realizado con proporción 1:1 y optimización de accuracy.

6.1.3.5 Desbalanceo con proporción 5:1 y optimización de recall

En base al anterior experimento, en el que se ha llevado a cabo la optimización de la métrica precisión, en el presente experimento se procederá a realizar una prueba similar eligiendo esta vez la métrica recall para la optimización.

En la Figura 134 y Figura 135 se puede observar que los resultados de entrenamiento son muy similares a los del experimento anterior.

En la Figura 136, Figura 137, Figura 138, Figura 139 y Figura 140 se puede observar que, al igual que en el experimento anterior, se mejoran la mayoría de los resultados del experimento que no hace uso de la penalización de loss.

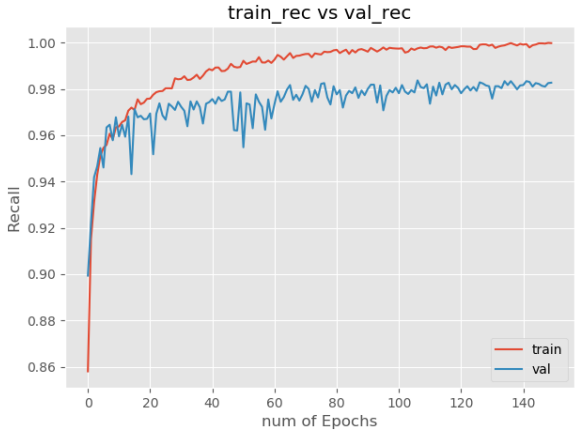

Figura 134. Recall de Tao con proporción 5:1 y optimización de recall

Figura 135. Loss de Tao con proporción 5:1 y optimización de recall

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	72.09	70.43	69.77	68.11	67.44	65.78	64.78	62.79	60.8
PPV	23.33	26.34	28.93	29.93	31.72	32.67	34.33	36.0	40.49

Figura 136. Umbrales de Tao con proporción 5:1 y optimización de precisión

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.95	43.41	59.14	99.27	98.63	50.07	97.94

Figura 137. Métricas de Tao con proporción 5:1 y optimización de recall

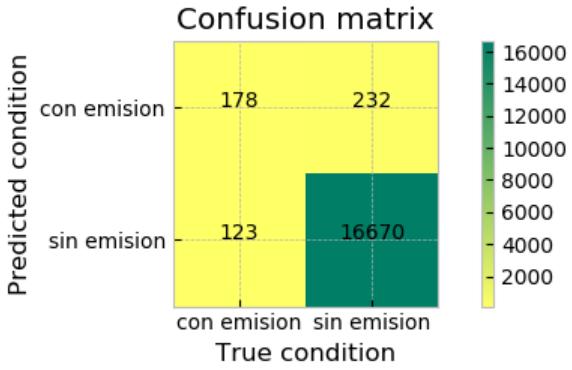

Figura 138. Matriz de confusión de Tao con proporción 5:1 y optimización de recall

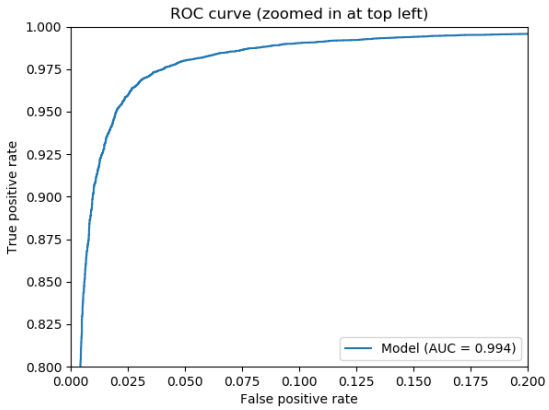

Figura 139. Curva ROC de Tao con proporción 5:1 y optimización de recall

	TP	FP	FN
0.95	22	21	8

Figura 140. Predicción de secuencias de Tao con proporción 5:1 y optimización de recall

En este experimento se puede comprobar que la métrica recall mejora tanto a nivel de predicción individual como de secuencias de emisiones a costa de un agravamiento del ratio de falsos positivos.

Debido a que este comportamiento es el contrario al deseado, el conjunto de datos resultante de la proporción de desbalanceo 5:1 optimizado mediante la métrica recall no obtiene resultados óptimos.

6.1.3.6 Desbalanceo con proporción real

En base a los anteriores experimentos se puede observar que, al aumentar el número de ejemplos, a pesar de que sea de una única clase, los resultados obtenidos mejoran. Es por ello por lo que se realizará un experimento más extremo haciendo uso de una proporción de desbalanceo con el número real de ejemplos.

Como experimento inicial con desbalanceo real de clases, se entrenará mediante la optimización de accuracy sin aplicar ningún tipo de penalización de loss. Los conjuntos utilizados para entrenamiento, validación y testing son los mismos que los utilizados en los anteriores experimentos.

Al realizar inicialmente el experimento se ha podido comprobar un comportamiento similar al obtenido al incrementar el desbalanceo de 1:1 a 5:1 debido a un learning rate demasiado bajo. En base a esto se ha decrementado este valor a **0,0001**, pudiendo observar de nuevo la convergencia de la red a los valores óptimos durante el entrenamiento.

En la Figura 141 y Figura 142 se observa que los valores de validación de accuracy se aproximan al 99% y la gran mayoría de valores obtenidos de loss se mantienen por debajo de 0,05.

A partir del testing del modelo resultado del entrenamiento se han obtenido los resultados reflejados en la Figura 143, Figura 144, Figura 145, Figura 146 y Figura 147. Se puede observar que la diferencia entre los valores de precisión y recall se hace más pequeña al entrenar la red con un conjunto altamente desblanceado, manteniendo ambas métricas con valores superiores a los obtenidos hasta el momento.

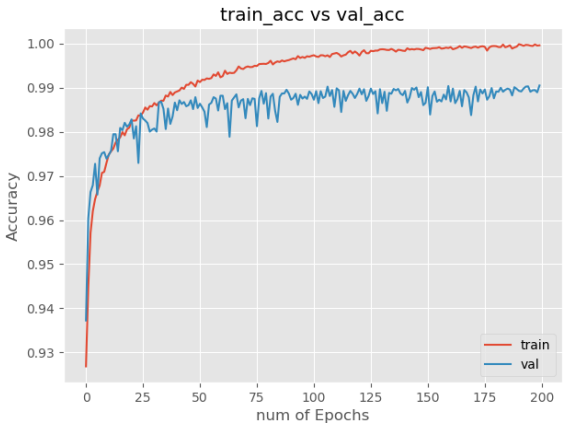

Figura 141. Accuracy de Tao con proporción real

Figura 142. Loss de Tao con proporción real

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	63.79	60.13	59.14	58.8	58.14	56.15	54.49	53.16	52.82
PPV	31.84	34.61	37.63	39.78	41.67	43.56	45.94	48.78	52.82

Figura 143. Umbrales de Tao con proporción real

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.9	52.82	52.82	99.16	99.16	52.82	98.35

Figura 144. Métricas de Tao con proporción real

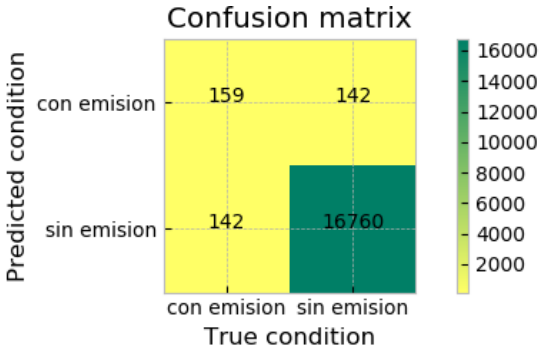

Figura 145. Matriz de confusión de Tao con proporción real

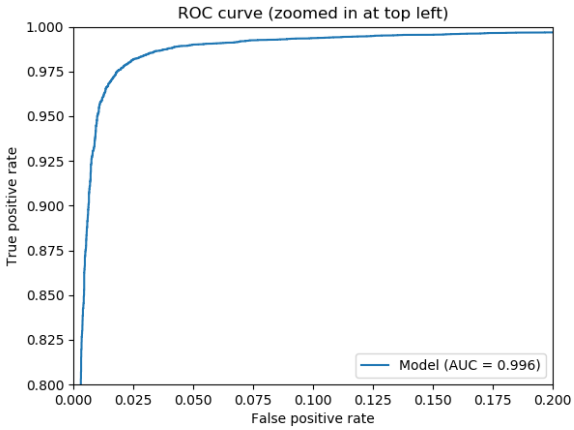

Figura 146. Curva ROC de Tao con proporción real

	TP	FP	FN
0.95	19	11	11

Figura 147. Predicción de secuencias de Tao con proporción real

Al comparar los resultados de este experimento con los obtenidos bajo las mismas condiciones con una proporción 5:1, se observa que la diferencia entre los valores de precisión y recall se incrementa ante el incremento de desbalanceo de clases. Para el testing de imágenes a nivel individual, a partir del leve decremento del umbral considerado óptimo, del decremento de falsos negativos y del incremento de falsos positivos, se puede concluir que la red tiende a asignar a los ejemplos la clase mayoritaria (sin emisión) en mayor medida que las redes resultantes de experimentos anteriores.

En cuanto al testing a nivel de secuencias de emisiones se puede observar que, a pesar de perder un verdadero positivo, se obtienen 9 falsos positivos menos que en el mismo experimento con proporción de desbalanceo 5:1.

6.1.3.7 Desbalanceo con proporción real y penalización de loss

Siguiendo con la experimentación en base al desbalanceo real de clases se repetirá el experimento anterior con la única diferencia consistente en la aplicación de la penalización de loss durante el entrenamiento en función a la proporción de ejemplos.

Al igual que en anteriores experimentos a los que se le ha aplicado la penalización de loss, en la Figura 148 y Figura 149 se puede observar como en las iteraciones iniciales las métricas toman valores poco óptimos mejorando de forma progresiva en función a cada epochs.
A partir de los resultados obtenidos en la Figura 150, Figura 151, Figura 152, Figura 153 y Figura 154 se puede observar que las métricas de recall y precisión se optimizan en comparación con el experimento correspondiente a la proporción 5:1.

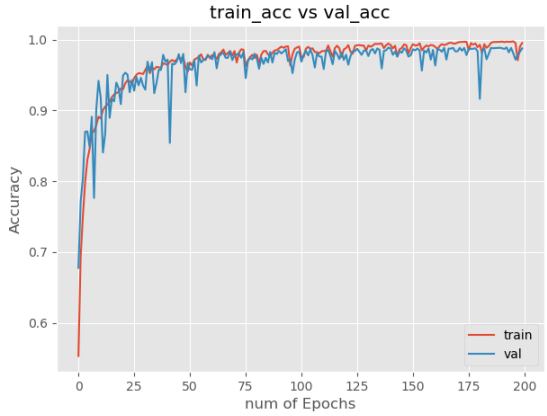

Figura 148. Accuracy de Tao con proporción real y penalización de loss

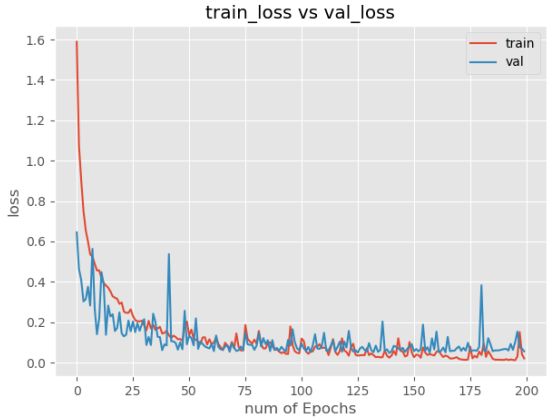

Figura 149. Loss de Tao con proporción real y penalización de loss

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	61.13	59.47	58.14	57.48	56.81	56.15	55.15	54.82	53.49
PPV	35.94	37.53	38.98	40.23	41.11	41.94	43.23	45.33	49.54

Figura 150. Umbrales de Tao con proporción real y penalización de loss

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.95	52.16	52.16	99.15	99.15	52.16	98.33

Figura 151. Métricas de Tao con proporción real y penalización de loss

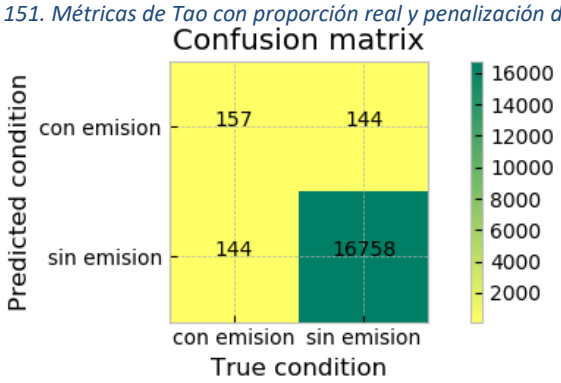

Figura 152. Matriz de confusión de Tao con proporción real y penalización de loss

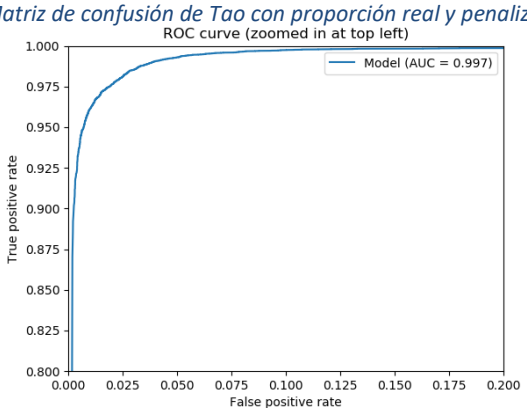

Figura 153. Curva ROC de Tao con proporción real y penalización de loss

	TP	FP	FN
0.95	18	14	12

Figura 154. Predicción de secuencias de Tao con proporción real y penalización de loss

En base a los resultados se puede observar que sigue un comportamiento similar al anterior experimento. Sin embargo, los valores que toman las métricas recall y precisión son ligeramente inferiores a nivel de predicción de imágenes de forma individual, traduciéndose en la predicción de secuencias en la aparición de 3 nuevas falsas alarmas.

6.1.3.8 Desbalanceo con proporción real y optimización de recall

El siguiente experimento consistirá en eliminar la penalización de loss del anterior experimento y emplear la optimización de recall.

En la Figura 155 y Figura 156 se puede observar un comportamiento similar al obtenido en el experimento realizado optimizando la métrica accuracy sin penalización de loss.

Al igual que en los resultados del entrenamiento, los resultados de testing mostrados en la Figura 157, Figura 158, Figura 159 Figura 160 y Figura 161 se puede observar un comportamiento similar al experimento de optimización de accuracy sin penalización de loss.

Figura 155. Recall de Tao con proporción real y optimización de recall

Figura 156. Loss de Tao con proporción real y optimización de recall

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	63.79	61.46	59.8	58.8	56.81	55.15	53.16	52.49	48.84
PPV	32.16	35.65	38.14	39.6	41.61	43.68	45.07	49.53	54.44

Figura 157. Umbrales de Tao con proporción real y optimización de recall

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.85	52.19	51.5	99.14	99.16	51.84	98.33

Figura 158. Métricas de Tao con proporción real y optimización de recall

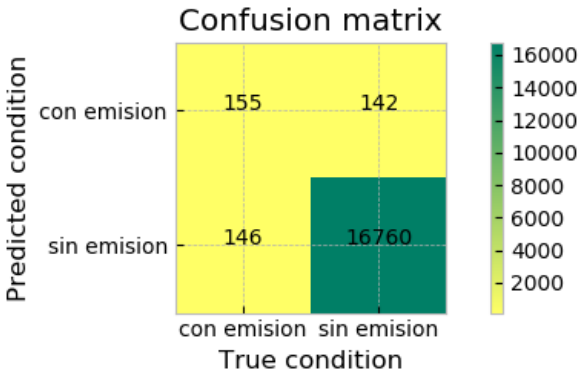

Figura 159. Matriz de confusión de Tao con proporción real y optimización de recall

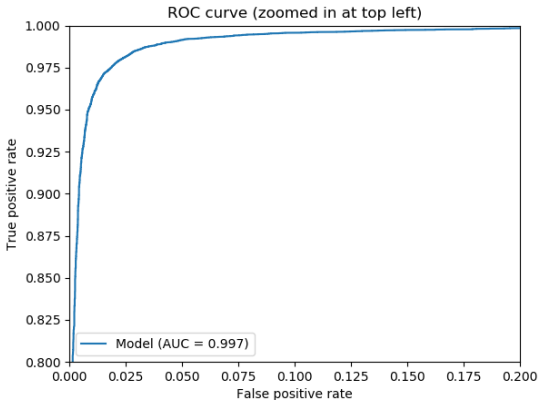

Figura 160. Curva ROC de Tao con proporción real y optimización de recall

	TP	FP	FN
0.95	18	12	12

Figura 161. Predicción de secuencias de Tao con proporción real y optimización de recall

En comparación con el anterior experimento se puede observar que a nivel de predicción individual se ha reducido en 2 unidades el número de falsos positivos y de verdaderos positivos. Esto resulta en un decremento de falsos positivos en la predicción de secuencias. Sin embargo, no llega a superar los resultados obtenidos en el experimento de optimización de accuracy sin penalización e loss con proporción real de ejemplos.

6.1.3.9 Desbalanceo con proporción real y optimización de precisión

Siguiendo el procedimiento de experimentación seguido con la proporción de clases 5:1, se realizará un experimento igual a los anteriores a excepción de la optimización del entrenamiento, que esta vez se hará en función a la métrica precisión.

En la Figura 162 y Figura 163 se muestras los resultados de entrenamiento que, en comparación con los obtenidos con proporción 5:1, se han optimizado levemente ambas métricas.

En la Figura 164, Figura 165, Figura 166, Figura 167 y Figura 168 se puede observar cómo se obtienen valores de precisión superiores a los obtenidos en anteriores experimentos. Sin embargo, esto requiere una notable disminución de recall.

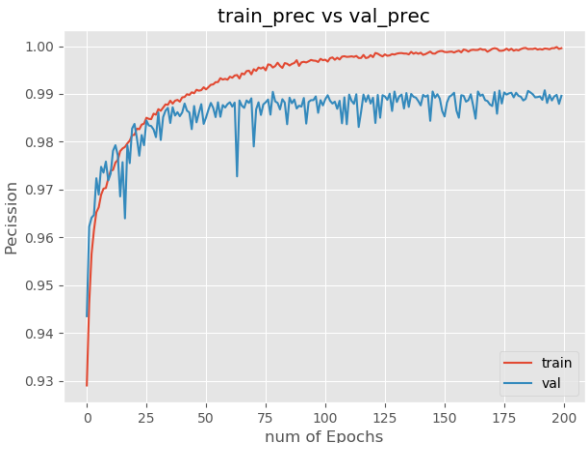

Figura 162. Precision de Tao con proporción real y optimización de precisión

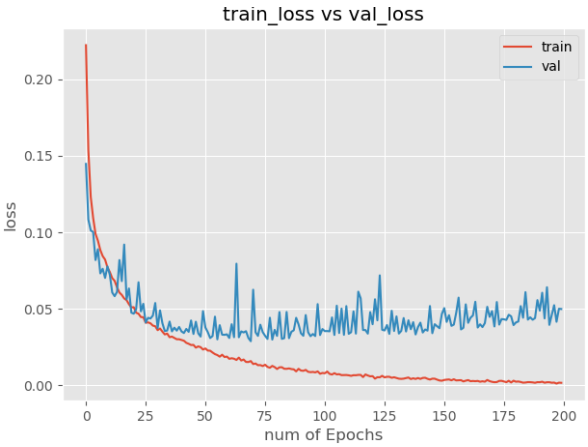

Figura 163. Loss de Tao con proporción real y optimización de precisión

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	61.46	59.14	57.81	56.15	55.48	54.82	52.16	50.83	47.18
PPV	29.84	34.03	37.83	40.33	42.28	44.12	46.31	50.0	54.83

Figura 164. Umbrales de Tao con proporción real y optimización de precisión

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.85	52.11	49.17	99.1	99.2	50.6	98.32

Figura 165. Métricas de Tao con proporción real y optimización de precisión

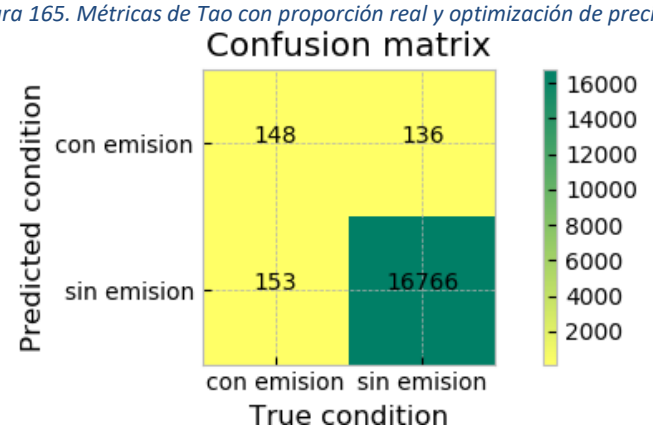

Figura 166. Matriz de confusión de Tao con proporción real y optimización de precisión

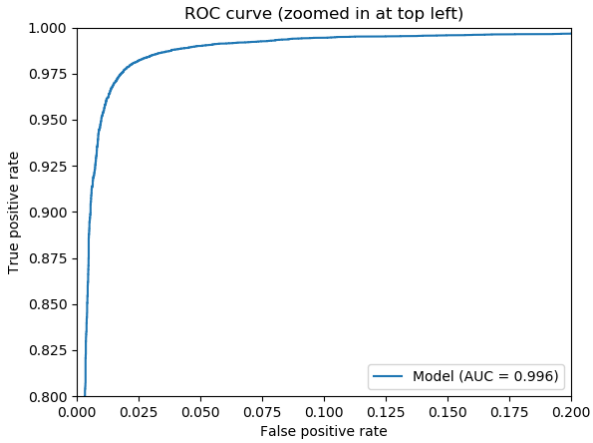

Figura 167. Curva ROC de Tao con proporción real y optimización de precisión

	TP	FP	FN
0.9	18	11	12

Figura 168. Predicción de secuencias de Tao con proporción real y optimización de precisión con umbral 0.9

	TP	FP	FN
0.95	16	10	14

Figura 169. Predicción de secuencias de Tao con proporción real y optimización de precisión con umbral 0.95

A partir de este experimento se puede comprobar cómo, incluso con un umbral inferior, la optimización de precisión ha permitido obtener valores de precisión superiores. Sin embargo, se debe tener en cuenta de que a pesar de que el objetivo sea minimizar los falsos positivos, la proporción de decremento de falsos positivos es muy inferior al decremento de verdaderos positivos, empeorando notablemente la capacidad de predicción de la red,

6.1.3.10 Desbalanceo con proporción real y reducción de learning rate

En base al estudio realizado sobre la relación de convergencia de las métricas durante el entrenamiento de la red y el valor del learning rate, se puede llegar a la conclusión de que, en casos en los que al aumentar el ejemplos de entrenamiento la red la red deja de aprender debido a que el valor de learning rate es demasiado grande, esto puede indicar que posiblemente para proporciones de ejemplos inferiores a pesar de converger durante el entrenamiento, el valor de learning rate puede seguir siendo demasiado grande.

En base a esto se ha reproducido el experimento desarrollado en el apartado 6.1.3.6 (en el que se han obtenido los mejores resultados hasta el momento) reduciendo el valor de learning rate a 0,00001.

En la Figura 170 y Figura 171 se observa que los valores de validación y entrenamiento toman valores similares a los obtenidos en experimentos similares con distinto learning rate, a excepción de que la convergencia a valores óptimos es visiblemente más lenta.

A partir del testing del modelo resultante del entrenamiento se han obtenido los resultados reflejados en la Figura 172, Figura 173, Figura 174, Figura 175 y Figura 176. Se puede observar que, a pesar de decrementar notablemente el valor de recall para todos los umbrales, el valor de precisión es mayor que en anteriores experimentos únicamente para umbrales iguales o superiores a 0,7.

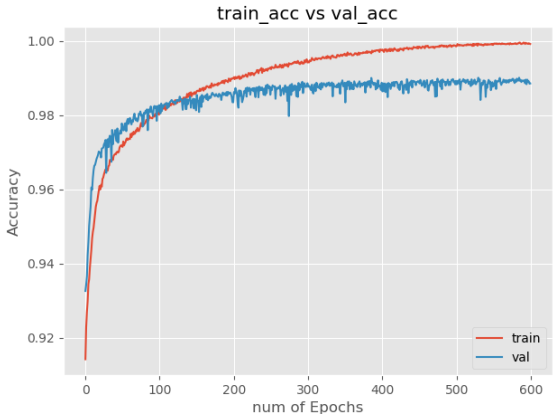

Figura 170. Accuracy de Tao con proporción real y reducción de learning rate

Figura 171. Loss de Tao con proporción real y reducción de learning rate

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	57.48	54.82	54.15	52.49	52.16	49.5	47.51	46.18	42.86
PPV	30.84	34.16	37.39	39.01	41.53	43.44	46.58	50.36	54.2

Figura 172. Umbrales de Tao con proporción real y reducción de learning rate

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.75	48.79	46.84	99.05	99.12	47.8	98.21

Figura 173. Métricas de Tao con proporción real y reducción de learning rate

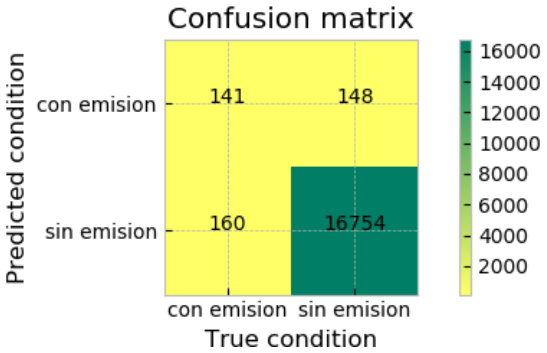

Figura 174. Matriz de confusión de Tao con proporción real y reducción de learning rate

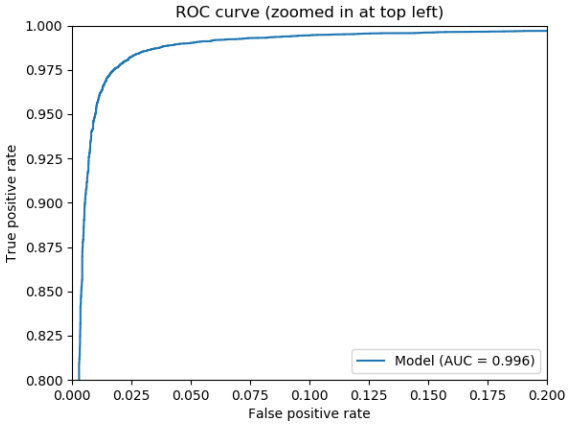

Figura 175. Curva ROC de Tao con proporción real y reducción de learning rate

	TP	FP	FN
0.8	18	12	12

Figura 176. Predicción de secuencias de Tao con proporción real y reducción de learning rate

En base a los resultados obtenidos se puede concluir que, a pesar de decrementar el número de falsos positivos en la predicción individual de imágenes, la proporción de verdaderos positivos decrementa notablemente. Al observar las predicciones a nivel de secuencia de imágenes se comprueba que el número de falsos positivos ha incrementado y el de verdaderos positivos ha decrementado para el umbral óptimo. Si se hiciese uso de un umbral mayor se podría obtener mayor precisión, sin embargo, la métrica recall descendería más drásticamente. Debido a eso, se concluye que esta configuración no obtiene mejores resultados.

Ya que la reducción del learning rate no ha permitido optimizar la red, del presente experimento se puede concluir que el valor de learning rate utilizado durante los experimentos es adecuado.

6.1.4 Selección de la red de Tao óptima

Una vez realizados numerosos experimentos en base a distintos criterios de balanceado de clases, conjuntos de entrenamiento y modificación de la arquitectura, entre otros, se considera que esta red se ha explotado lo suficiente como para poder escoger la red resultante de uno de los experimentos como posible solución al sistema.

Para ello se deberán comparar de forma más exhaustiva cada una de las redes en función a los requisitos que debe cumplir el sistema. Debido a que un aspecto de alta importancia se trata de obtener el mínimo número de falsos positivos (a pesar de que esto pueda generar un aumento de falsos negativos) se mostrará de forma gráfica la curva resultante de la

relación entre las métricas recall y precisión para los 9 umbrales mostrados en las tablas resultantes del testing en cada experimento. Esta gráfica se interpreta de forma que cuanto más arriba se encuentre un punto, mayor es el ratio de detección de emisiones. Asimismo, cuanto más a la derecha se encuentre el punto, menor será el índice de falsas alarmas. Por tanto, cuanto más se aproxime un punto a la esquina superior derecha del gráfico, mejor será la relación entre falsos positivos y falsos negativos.

En la Figura 177 se muestran los resultados obtenidos de cada uno de los experimentos realizados, mostrados en agrupaciones en la Figura 178, Figura 179, Figura 180 y Figura 181. En la leyenda de las gráficas se identifica cada experimento según el apartado del experimento en la memoria.

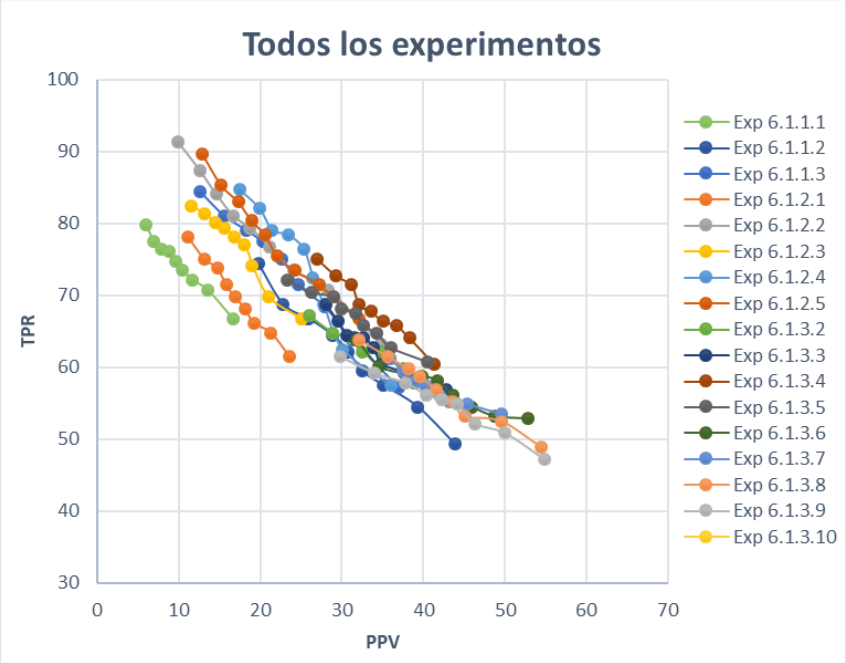

Figura 177. Comparación de todos los experimentos de Tao2016 y testing interdías

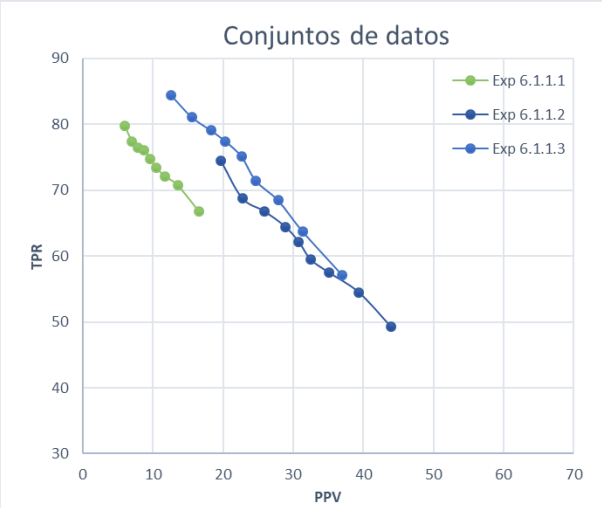

Figura 178. Experimentos Tao variando conjuntos entrenamiento

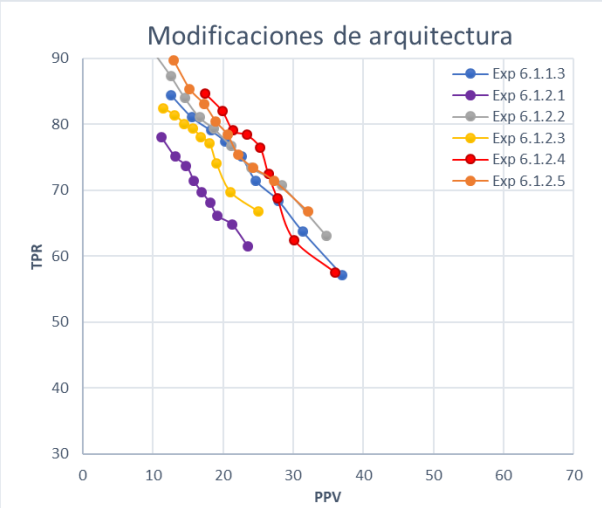

Figura 179. Experimentos Tao variando arquitectura de red

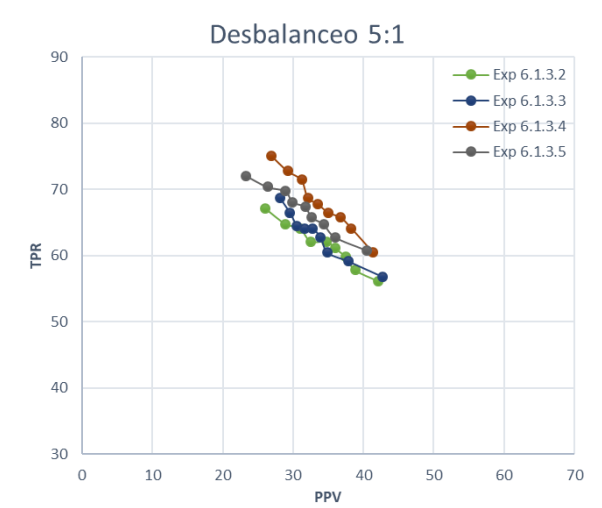

Figura 180. Experimentos Tao con desbalanceo 5:1 variando técnicas de optimización del entrenamiento

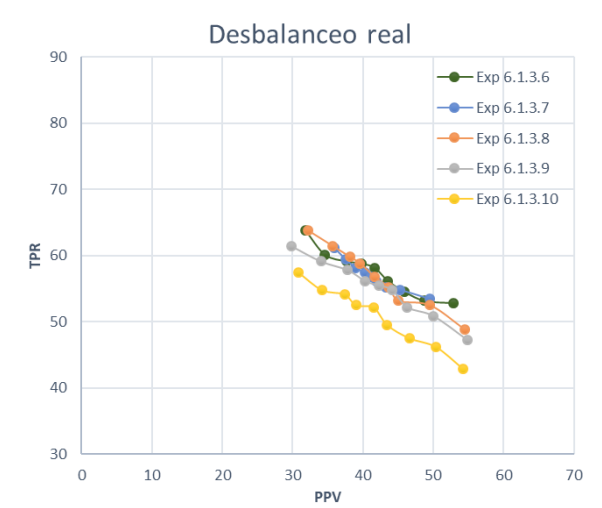

Figura 181. Experimentos Tao con desbalanceo real variando técnicas de optimización del entrenamiento

A partir de la Figura 178 se puede observar que haciendo uso de la proporción de clases real en vez de la de 1:1, se consigue una clara mejor de la métrica precision para cada uno de los umbrales a costa de una reducción de recall. A su vez, se puede observar que, al triplicar el número de ejemplos de cada clase (en cuanto al experimento con proporción 1:1) se consigue optimizar todas las métricas en comparación con ambos experimentos.

A partir del resultado obtenido del experimento que hace uso de tres días para el entrenamiento y la validación de la red con su arquitectura sin modificaciones, se comparará el comportamiento de la red ante distintas modificaciones de su arquitectura en la Figura 179. De esta figura se puede concluir que, a excepción de la eliminación de dropout o la reducción de neuronas en capas totalmente conectadas, las modificaciones de la

arquitectura no suponen grandes variaciones en los resultados finales de la red, ya que las curvas tienden a solaparse. Sin embargo, se tomará como red óptima de este conjunto aquella correspondiente a la reducción de capas convolucionales, ya que se puede observar que se mantiene un poco por encima en recall que el resto de los experimentos, aproximándose levemente al punto óptimo de la gráfica (esquina superior derecha).

En base a dicha arquitectura modificada se han realizado los experimentos de la Figura 180 y Figura 181. Estas gráficas muestran los resultados obtenidos para distintas proporciones de desbalanceo de clases.

Si se comparan los resultados obtenidos entre las distintas gráficas, se puede observar que a base de aumentar el desbalanceo de clases, se gana precisión a costa de bajar recall. En base al análisis de los falsos negativos que suponen esta disminución de recall, se ha observado que las emisiones no detectadas se caracterizan todas por tener una baja dimensión o ser dudosas debido a su gran similitud con nubes, o ser casi imperceptibles debido a condiciones atmosféricas adversas. Ya que las emisiones de baja dimensión no son el principal objetivo del sistema, se tomará como mejores resultados aquellos que maximicen la precisión, sin llegar a agravar de forma sustancial el recall.

En la Figura 181 se puede comprobar que el experimento 6.1.3.6 obtiene valores de precisión similares a los del resto de experimentos del proporción de clases real (conjunto de experimentos con mayor precisión) y valores de recall ligeramente superiores. A su vez el experimento 6.1.3.9 (el cual optimiza la métrica precision durante el entrenamiento) con umbral 0,9 obtiene el valor de precisión más alto, a costa de un mayor decremento de recall. Debido a que la ganancia de precisión no es muy significativa frente a la pérdida de recall, **se seleccionará la configuración y red correspondiente al experimento 6.1.3.6.**

6.1.5 Cross testing con la red seleccionada (Dudosas si son emisión)

Una vez se ha seleccionado la red considerada óptima dentro del conjunto de experimentos realizados en el presente apartado, se procederá a realizar cross testing intercambiando el conjunto de testing con cada uno de los 3 conjuntos utilizados para el entrenamiento y la validación, de modo que se deberá realizar un total de 3 experimentos adicionales, utilizando como conjuntos de testing el día 1, 10 y 19.

Se fija el umbral a 0,5 para todos los experimentos, de modo que se pueda comparar de forma más significativa los resultados de cross testing con cada conjunto.

Con objeto de no sobrecargar el apartado de cross testing con demasiados resultados de todos los experimentos requeridos se han trasladado todos estos resultados al ANEXO III.

Una vez se han realizado todos los experimentos se podrán comparar los resultados obtenidos para cada uno de ellos. En la Figura 182 se muestra una comparación gráfica de los valores que toman las métricas recall y precisión para distintos umbrales en casa uno de los experimentos de cross testing.

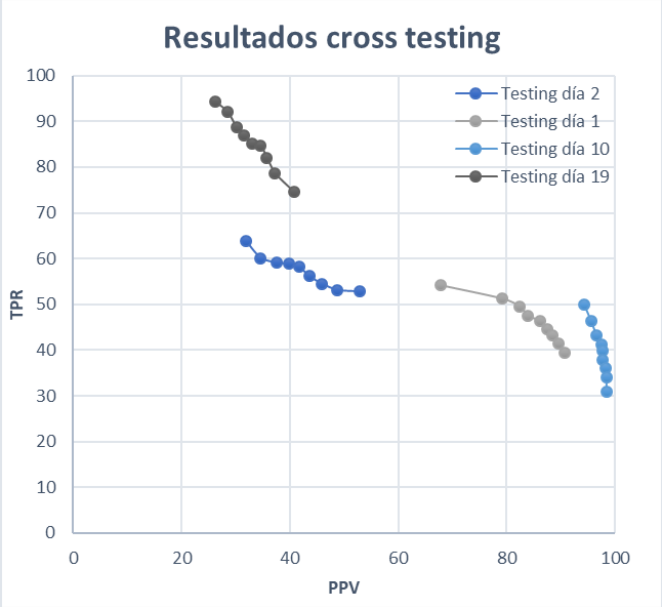

Figura 182. Comparación resultados cross testing con primer criterio de etiquetado

A partir de esta gráfica se puede observar que los resultados de testing para cada uno de los experimentos son completamente diferentes. Esto se puede deber a que a la hora de conformar un conjunto de datos óptimo se han buscado días con características atmosféricas variadas, de modo que el sistema sea más robusto ante estas situaciones.

En caso del experimento que hace uso del día 19 como conjunto de testing, por ejemplo, cuenta con un alto ratio de falsos positivos. Esto se debe a que se trata del único conjunto de datos que cuenta con imágenes de un atardecer en el que el cielo torna algunas nubes con tonos rojizos. Al utilizar este conjunto como testing y no en el entrenamiento, la red no aprende a diferenciar emisiones y nubes rojizas debidas al atardecer, lo que provoca una gran subida de falsos positivos.

El caso contrario a este se encuentran los experimentos con el día 1 y día 10, que cuentan con un ratio de falsos negativos muy bajo debido a que se trata de días con intervalos soleados y nubes de tonalidades blancas fácilmente distinguibles a emisiones. En cuanto al alto número de falsos negativos se debe a que el número de emisiones en estos días es mayor siendo estas emisiones pequeñas o muy difusas, siendo en la mayoría altamente dudosas.

En cuando al experimento con el día 22 (utilizado hasta ahora como conjunto de testing para todos los experimentos) se obtiene una relación entre ambas métricas relativamente similar. Esto se debe a que este día incluye imágenes de cielos grises y emisiones de todos tipo de tamaños y densidades.

En la Tabla 12 se han recopilado los resultados obtenidos a nivel de predicción de secuencias para cada uno de los experimentos que conforman el cross testing. A partir de esta tabla se puede observar que los valores de precision y recall de cada experimento se corresponden con una valor algo superior al obtenido en la predicción a nivel de imágenes de forma independiente.

	TP	FP	FN	TPR	PPV	Umbral
Día 2	22	20	8	73%	52%	0,5
Día 1	29	2	28	51%	94%	0,5
Día 10	35	1	38	48%	97%	0,5
Día 19	21	33	2	91%	39%	0,5

Tabla 12. Comparación de resultados de predicción de secuencias para en cross testing

Tras analizar las secuencias consideradas como falsos negativos de cada uno de lo experimentos se ha comprobado que todas ellas pertenecen a emisiones de tamaños muy pequeños que, en la gran mayoría de los casos, podrían ser consideradas como no emisiones.

6.1.6 Cross testing con la red seleccionada (Dudosas **no** son emisión)

A partir del apartado anterior se puede concluir que el etiquetado del conjunto en base a la inclusión de emisiones pequeñas afecta de forma notable la capacidad de clasificación de la red.

Tal y como se especificó anteriormente estos experimentos se han llevado a cabo haciendo uso de los conjuntos de datos etiquetados según el primer criterio, consistente en la clasificación de las imágenes como ejemplos de emisión independientemente de la dimensión y densidad de dicha emisión.

Sin embargo, al analizar las imágenes en las que se suele equivocar la red, sus respectivos resultados parecen indicar que la inclusión de imágenes de emisiones de baja densidad y dimensión como ejemplos con emisión provoca que la red genere un gran número de falsos positivos, confundiendo ejemplos en los que las nubes toman tonalidades más oscuras. Además, se ha observado que, en general, todos los falsos negativos consisten en emisiones de dimensiones muy pequeñas o dudosas cuya detección no es primordial.

Debido a que el sistema tiene como máxima prioridad la detección de emisiones de densidades y dimensiones intermedias y grandes, se procederá a hacer uso de los mismos días que en los anteriores experimentos etiquetados siguiendo el segundo criterio de etiquetado.

En el ANEXO IV, al igual que en el apartado anterior, se mostrarán los resultados obtenidos para cada uno de los experimentos de cross testing, esta vez utilizando los conjuntos etiquetados según el segundo criterio.

Una vez realizados los experimentos relativos al cross testing utilizando el segundo criterio de etiquetado, se procederá a realizar una comparación de los resultados obtenidos. En la Figura 183 se muestran los valores de precisión y recall obtenidos para distintos umbrales en cada uno de los experimentos.

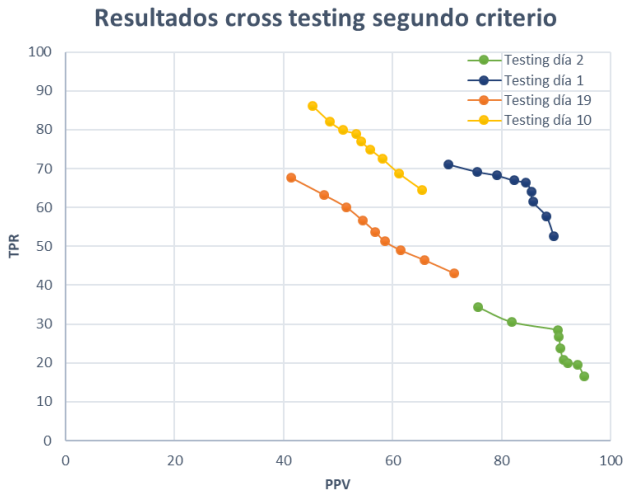

Figura 183. Comparación resultados cross testing con segundo criterio de etiquetado

Se puede observar que los resultados de los experimentos, aunque varían dentro de un área ligeramente inferior que los resultados de los experimentos realizados según el primer criterio, estos siguen siendo dispares. Esto significa, por tanto, que los distintos días que conforman los conjuntos de datos, así como el etiquetado de estos, suponen un factor decisivo en cuanto al rendimiento de la red.

Sin embargo, se puede comprobar que todos y cada uno de los experimentos han mejorado en precisión al emplear el segundo etiquetado, lo que prueba que este etiquetado proporciona una mejor relación entre el número de verdaderos positivos y de falsos positivos.

En la Tabla 13 se muestran los resultados de testing obtenidos a nivel de secuencia.

	TP	FP	FN	TPR	PPV	Umbral
Día 2	9	0	17	35%	100%	0,5
Día 1	18	5	9	67%	78%	0,5
Día 10	26	25	7	79%	51%	0,5
Día 19	15	11	9	63%	58%	0,5

Tabla 13. Comparación de resultados de predicción de secuencias para en cross testing (dudosas no son emisión)

De esta tabla se puede observar que el número de secuencias de emisiones se ha visto reducido por este nuevo criterio de etiquetado. Se puede observar que el primer experimento y el último obtienen una mayor precisión a costa de un recall menor. Sin embargo, los otros dos experimentos reducen los valores de precisión, obteniendo así un recall mayor. En líneas generales, a excepción del primer experimento, se obtienen valores de recall y precisión más cercanos entre sí que en los experimentos con el primer etiquetado, en los que la diferencia entre ambas métricas era muy grande, obteniendo así valores muy extremos.

6.2 Experimentos con testing intradías

De los experimentos de cross testing realizados en el apartado anterior se ha podido comprobar que la variedad existente entre los días que se utilizan tanto en entrenamiento y validación como en testing dan como resultado una inconsistencia en los resultados de testing. Esto parece indicar una gran dispersión de los datos debido a aspectos como la diversidad de formas, dispersión y color de las emisiones, así como la diferencia existente entre las características meteorológicas de cada uno de los días utilizados en los experimentos.

El uso de días distintos para el entrenamiento y el testing, por tanto, da como resultado una gran inconsistencia a la hora de realizar experimentos de cross testing. Debido a esto se probará a utilizar los 4 días para los conjuntos de entrenamiento, validación y testing, en vez de utilizar 3 días para validación y entrenamiento y el restante para testing. Para ello las imágenes se distribuirán en 3 subconjuntos (entrenamiento, validación y testing) de forma ordenada según 3 combinaciones distintas. Si *E* hace referencia a la asignación del ejemplo al subconjunto de entrenamiento, *V* al de validación y *T* al de testing, estas combinaciones son las siguientes:

1. E E E E V T . . .
2. E E V T E E . . .
3. V T E E E E . . .

Estas combinaciones se irán aplicando a todas las imágenes que conforman los 4 días ordenadas de forma cronológica. De este modo, el uso de estas tres combinaciones (sin modificar el resto de hiperparámetros del experimento) conformarán el cross testing de los experimentos que utilizan los mismos días de entrenamiento, validación y testing, ya que se estarán alternando los ejemplos utilizados para cada uno de los subconjuntos.

Antes de realizar los experimentos se ha comprobado que la proporción de ambas clases sea equivalente para los 3 subconjuntos (entrenamiento, validación y testing) entre sí. Estas proporciones concuerdan también entre las tres combinaciones de división de subconjuntos (EEEEVT, EEVTEE y VTEEEE).

En caso de que los resultados obtenidos de los experimentos sean consistentes, de modo que las 3 combinaciones aporten resultados similares, se podrá establecer una consistencia en la experimentación. Debido a esto en los siguientes subapartados se expondrá un reducido conjunto de experimentos realizados con objeto de obtener la red óptima atendiendo a los resultados obtenidos por parte del testing intradías.

6.2.1 Experimentación con la configuración seleccionada para testing intradías

Inicialmente se ha realizado un experimento haciendo uso de los hiperparámetros y la arquitectura de red utilizados en la red seleccionada como óptima en el apartado anterior de experimentación con testing interdías.

En la Figura 184 y Figura 185 se puede observar que la red aprende mucho más rápido con estos nuevos conjuntos de datos, convergiendo rápidamente a un valor aproximado a 0,99 de accuracy y 0,02 de loss. Sin embargo, también se puede observar que los valores de entrenamiento para ambas métricas no llegan a ser nunca el óptimo (1 para accuracy y 0 para loss).

Por otro lado, en la Figura 186, Figura 187, Figura 188 y Figura 189 se muestran unos valores muy superiores a los obtenidos hasta ahora, lo que indica que entrenando y testeando con condiciones similares, la red predice de forma correcta, llegando a obtener valores por encima del 80% en precisión y recall para un umbral de 0,5.

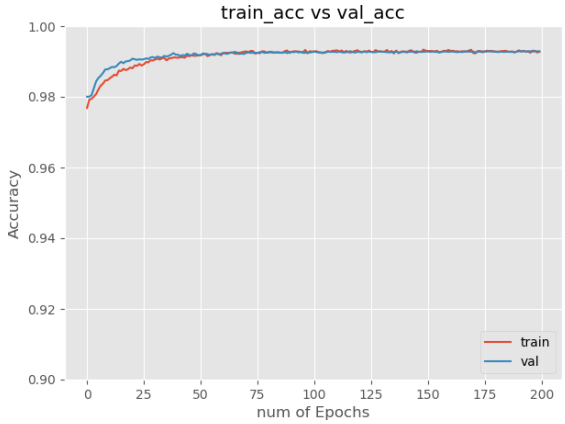

Figura 184. Accuracy de Tao con proporción real para testing intradías

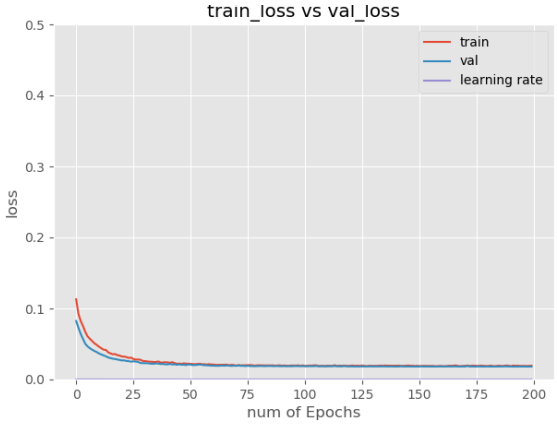

Figura 185. Loss de Tao con proporción real para testing intradías

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	91.88	89.74	87.61	86.32	82.48	77.78	73.5	67.95	57.26
PPV	59.89	69.54	75.65	79.84	85.02	87.92	90.05	93.53	97.1

Figura 186. Umbrales de Tao con proporción real para testing intradías

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	85.02	82.48	99.63	99.7	83.73	99.35

Figura 187. Métricas de Tao con proporción real para testing intradías

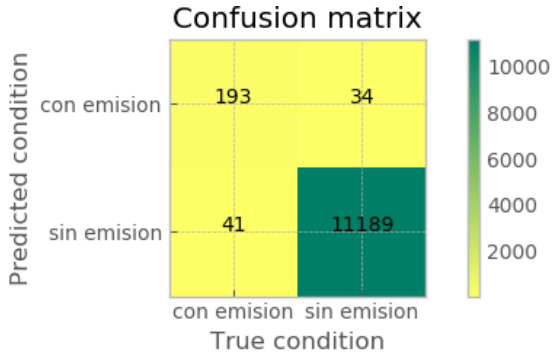

Figura 188. Matriz de confusión de Tao con proporción real para testing intradías

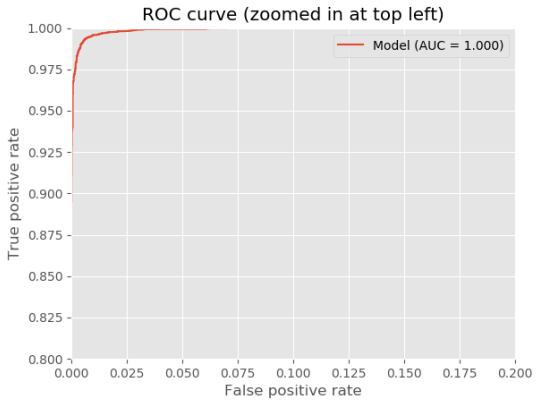

Figura 189. Curva ROC de Tao con proporción real para testing intradías

6.2.2 Sustitución de learning rate fijo por step decay

Una vez realizado el experimento inicial con las mismas condiciones que las de la red elegida como óptima en el testing interdías, se procederá a realizar una breve batería de experimentos con objeto de comprobar cómo afecta la modificación de algunos hiperparámetros haciendo uso del testing intradías en vez del testing interdías. Se comenzará con la sustitución de un learning rate fijo con el uso de un step decay que inicie en un valor de 0,001 y no descienda de 0,0001.

A partir de la Figura 190 y de la Figura 191 se puede observar que esta vez los valores de entrenamiento alcanzan el valor óptimo en ambas métricas. A su vez, los valores de validación son ligeramente más próximos al valor óptimo en comparación con el anterior experimento.

En la Figura 192, Figura 193, Figura 194, Figura 195 se observa que las métricas llegan a valores superiores que los obtenidos en el anterior experimento. En caso de la precisión y el recall obtiene valores superiores al 70% en el peor de los casos para los distintos umbrales que han sido contemplados. Esto se debe a un incremento de verdaderos positivos y un decremento de falsos positivos y falsos negativos en relación con el anterior experimento.

En la Figura 196 se muestra el decremento del valor que ha tomado el hiperparámetro learning rate en función a las epochs del experimento.

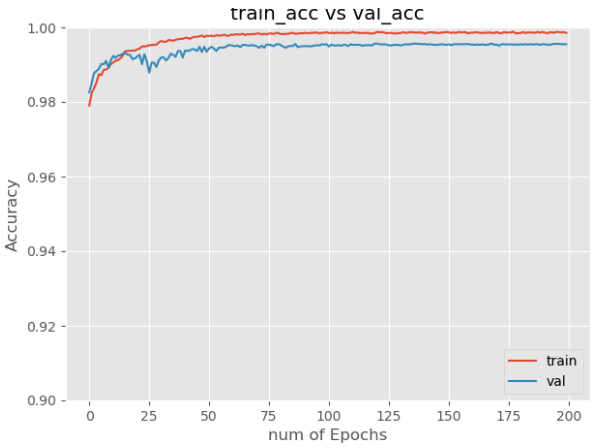

Figura 190. Accuracy de Tao con testing intradías y decay

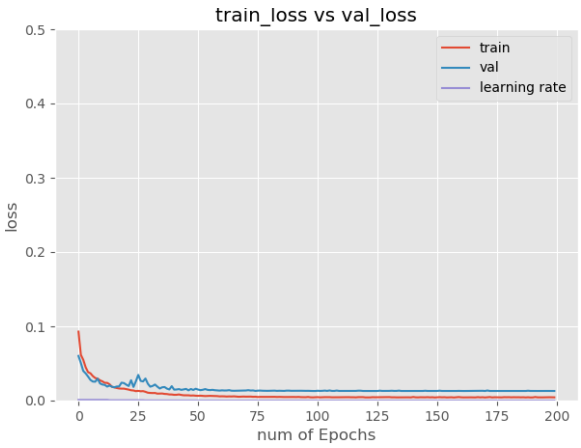

Figura 191. Loss de Tao con testing intradías y decay

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	93.59	91.88	89.32	88.03	85.9	85.04	83.76	81.2	79.06
PPV	75.0	79.63	81.96	84.43	85.9	88.44	91.16	92.68	94.39

Figura 192. Umbrales de Tao con testing intradías y decay

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	85.9	85.9	99.71	99.71	85.9	99.42

Figura 193. Métricas de Tao con testing intradías y decay

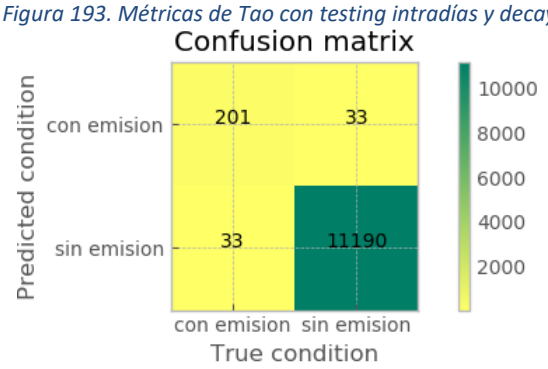

Figura 194. Matriz de confusión de Tao con testing intradías y decay

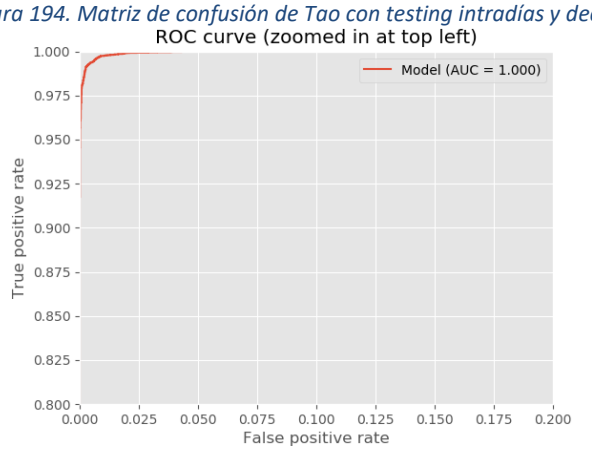

Figura 195. Curva ROC de Tao con testing intradías y decay

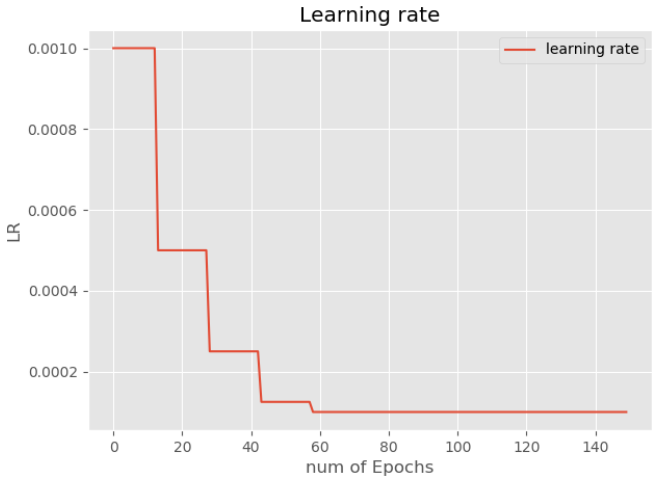

Figura 196. Decremento del learning rate para Tao con testing intradías y decay

6.2.3 Step decay y optimización de la métrica precisión

En base al experimento anterior, se ha podido comprobar que el uso de decay en vez de un learning rate fijo, implica una optimización de todas y cada una de las métricas. Es por ello por lo que los siguientes experimentos a realizar harán uso de la misma función de decay utilizada en el experimento anterior.

A partir de la Figura 197, Figura 198, Figura 199, Figura 200, Figura 201 y Figura 202 se puede observar que al sustituir la optimización de la métrica accuracy (por defecto) por la optimización de precisión, se obtiene una ligera optimización atendiendo a los valores de las métricas obtenidos para el umbral 0,5.

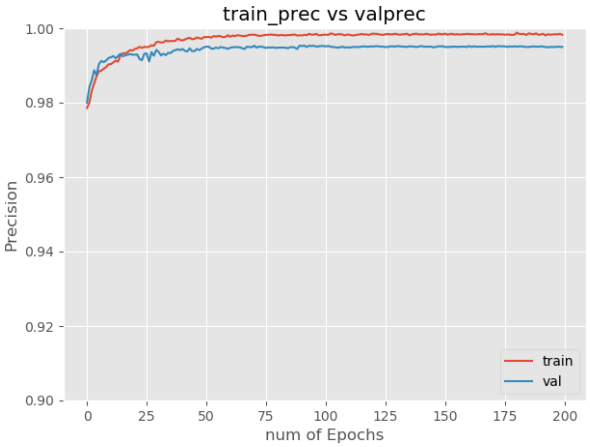

Figura 197. Precision de Tao con testing intradías, decay y optimización de precisión

Figura 198. Loss de Tao con testing intradías, decay y optimización de precisión

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	94.87	92.31	90.6	88.89	87.61	86.75	83.76	80.77	74.79
PPV	75.77	81.51	84.46	87.03	89.13	90.62	91.59	92.65	94.59

Figura 199. Umbrales de Tao con testing intradías, decay y optimización de precisión

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	89.13	87.61	99.74	99.78	88.36	99.53

Figura 200. Métricas de Tao con testing intradías, decay y optimización de precisión

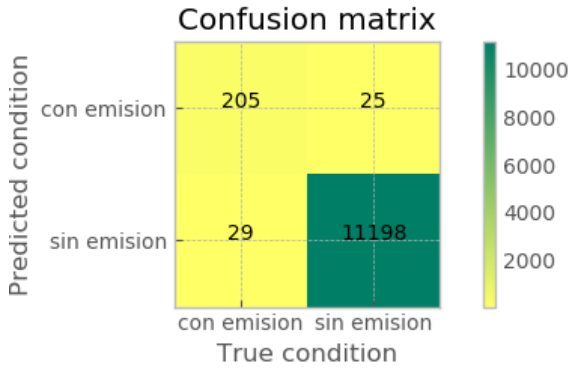

Figura 201. Matriz de confusión de Tao con testing intradías, decay y optimización de precisión

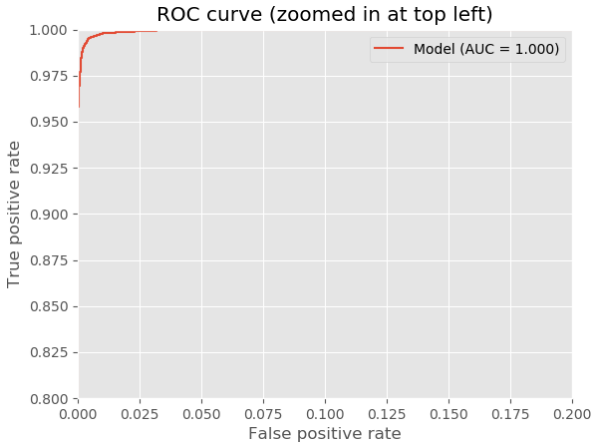

Figura 202. Curva ROC de Tao con testing intradías, decay y optimización de precisión

6.2.4 Step decay y optimización de la métrica recall

El siguiente experimento (siguiendo la metodología del experimento anterior) hará uso del step decay y optimizar la métrica recall.

En la Figura 203, Figura 204, Figura 205, Figura 206, Figura 207 y Figura 208, se puede observar que los resultados obtenidos son similares a los correspondientes a la optimización de accuracy, sin mejorar los resultados obtenidos mediante la optimización de la métrica precisión.

Figura 203. Recall de Tao con testing intradías, decay y optimización de recall

Figura 204. Loss de Tao con testing intradías, decay y optimización de recall

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	91.45	90.17	88.46	86.32	85.47	84.19	80.77	79.91	74.36
PPV	74.05	80.84	83.81	85.23	88.89	89.95	91.3	92.57	95.08

Figura 205. Umbrales de Tao con testing intradías, decay y optimización de recall

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.45	86.96	85.47	99.7	99.73	86.21	99.44

Figura 206. Métricas de Tao con testing intradías, decay y optimización de recall

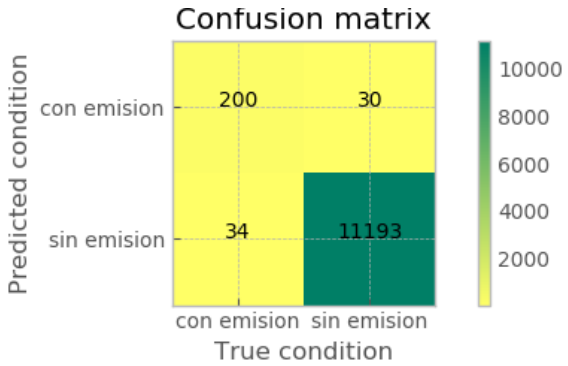

Figura 207. Matriz de confusión de Tao con testing intradías, decay y optimización de recall

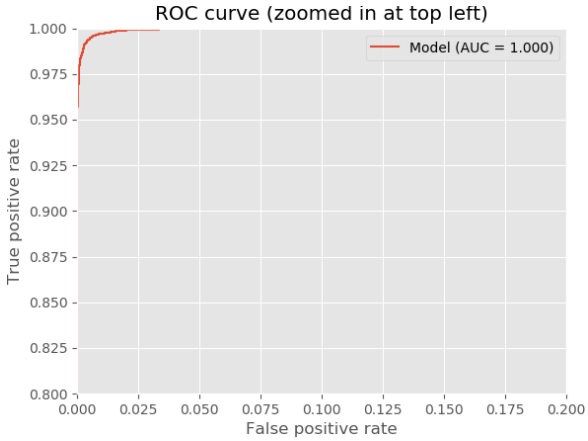

Figura 208. Curva ROC de Tao con testing intradías, decay y optimización de recall

6.2.5 Step decay y penalización de loss

Una vez experimentado con la optimización de las métricas consideradas de mayor interés (precisión y recall) se procederá a optimizar accuracy junto con la penalización de la métrica loss de forma proporcional al desbalanceo existente entre clases en el conjunto de entrenamiento.

A partir de la Figura 209, Figura 210, Figura 211, Figura 212, Figura 213 y Figura 214 se puede observar que, en comparación del experimento que optimiza accuracy sin penalizar la métrica loss, el número de verdaderos positivos ha incrementado levemente a cambio de un gran incremento de falsos positivos. Esto se traduce en que la red clasifica como emisión un mayor número de ejemplos, lo que repercute muy negativamente en la métrica precisión. Debido a que prevalece la disminución de falsos positivos ante el aumento de verdaderos positivos este enfoque no se considera el óptimo.

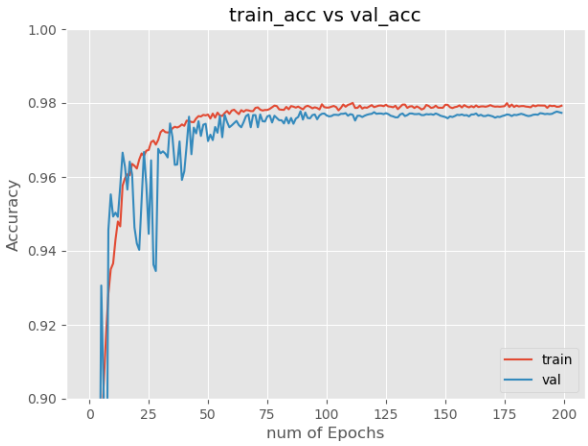

Figura 209. Accuracy de Tao con testing intradías, decay y penalización de loss

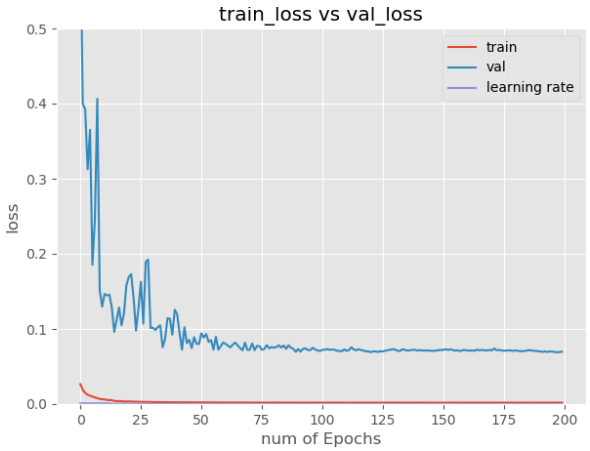

Figura 210. Loss de Tao con testing intradías, decay y penalización de loss

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	100.0	99.15	98.72	98.29	97.86	97.01	97.01	95.3	91.45
PPV	30.12	36.54	41.03	44.66	47.31	50.44	54.7	59.47	66.67

Figura 211. Umbrales de Tao con testing intradías, decay y penalización de loss

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.95	71.33	87.18	99.73	99.27	78.46	99.02

Figura 212. Métricas de Tao con testing intradías, decay y penalización de loss

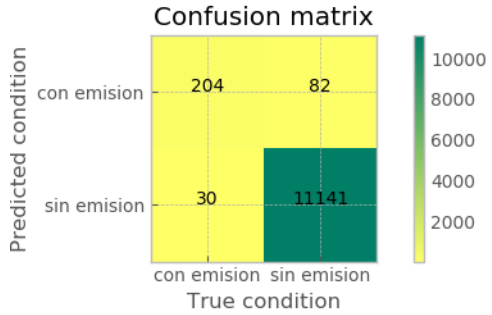

Figura 213. Matriz de confusión de Tao con testing intradías, decay y penalización de loss

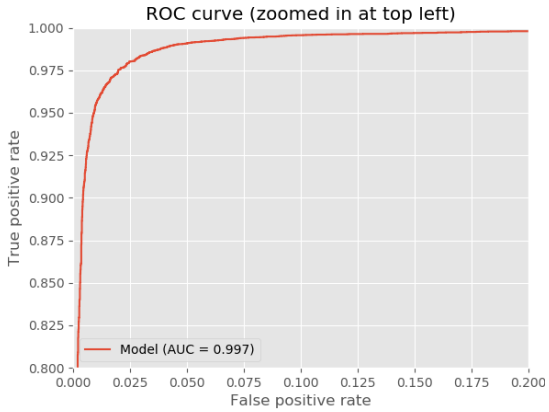

Figura 214. Curva ROC de Tao con testing intradías, decay y penalización de loss

6.2.6 Selección de la red de Tao óptima

Una vez realizados un breve conjunto de experimentos haciendo uso del testing intradías se procederá a seleccionar la red óptima resultante de los experimentos anteriores. Para ello se seguirá la metodología seguida en la sección 6.1.4 mostrando una gráfica en la que se comparen las curvas resultantes de la relación entre las métricas recall y precisión para distintos umbrales en cada uno de los experimentos.

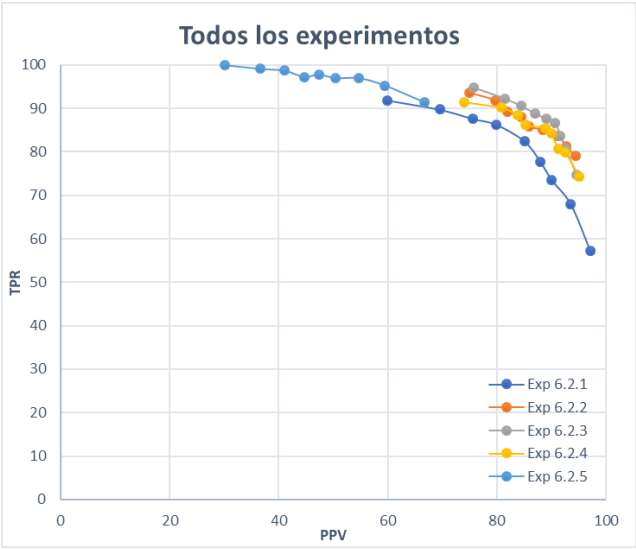

Figura 215. Comparación de todos los experimentos de Tao2016 y testing intradías

A partir de esta gráfica se puede observar que el experimento 6.2.3 se encuentra levemente por encima del resto de experimentos para la mayor parte de umbrales, aproximándose más a la esquina superior derecha (considerada la óptima), lo que significa que los valores de ambas métricas son mayores para dicho experimento.

En base a estos juicios y a los realizados anteriormente en los respectivos apartados de cada uno de los experimentos, se seleccionará como red óptima la resultante del apartado 6.2.3 relativo al uso de decay en conjunto con la experimentación

6.2.7 Cross testing con la red seleccionada (Dudosas **no** son emisión)

Al igual que en la sección 6.1.6 se procederá a realizar una fase de cross testing con la red seleccionada en el apartado anterior con objeto de comprobar la consistencia de los conjuntos utilizados para entrenamiento, validación y testing.

Este cross testing se realizará comparando los resultados obtenidos en el apartado 6.2.3 con el conjunto resultante de la combinación EEEEVT con los resultados obtenidos en dos nuevos experimentos que únicamente modificará los conjuntos de datos a utilizar (uno según el criterio EEVTEE y otro según el criterio restante VTEEEE). Los resultados de estos experimentos se encuentran en el ANEXO V.

En la Figura 216 se muestran los valores obtenidos a partir de cada uno de los tres experimentos de las métricas precisión y recall para cada uno de los umbrales evaluados.

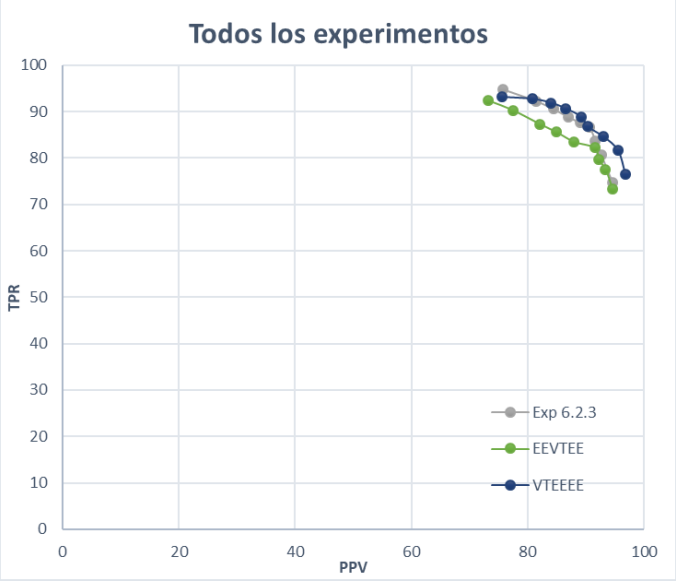

Figura 216. Comparación resultados cross testing con testing intradías

A partir de este gráfico se puede observar que las curvas obtenidas siguen un patrón muy similar, a diferencia de los experimentos de cross testing realizados hasta el momento (con el método de testing interdías). Por tanto, esto comprueba que la red aprende correctamente dentro del mismo día debido a características atmosféricas similares independientemente de las imágenes utilizadas para el entrenamiento, validación y testing.

Como conclusión de los resultados obtenidos, se ha confirmado que la inconsistencia entre los distintos experimentos de cross testing se debe a que los días son muy distintos entre sí, lo que parece indicar que para llevar a cabo un sistema estable obteniendo resultados similares para cada día se debería hacer uso de un gran volumen de imágenes que representen muchos más días que los utilizados en los experimentos actuales.

6.2.8 Testing con un día independiente al conjunto de entrenamiento

A partir del apartado 6.2.7 se ha podido comprobar que el método de testing intradías es consistente, obteniendo resultados similares para distintos datos de entrenamiento, validación o testing, siempre y cuando pertenezcan al mismo día.

El testear con un día diferente a los utilizados para el entrenamiento y la validación, por tanto, parece indicar que los resultados obtenidos para cada métrica peores. Sin embargo, no se puede determinar si este decrecimiento del valor de las métricas será el mismo para todos los experimentos realizados. En base a esto, se procederá a realizar otra fase de testing (esta vez con un día completamente nuevo) a las redes resultantes de los experimentos realizados para el testing intradías. Esto permitirá comprobar que la red seleccionada es la óptima independientemente del conjunto de testing empleado.

En el ANEXO VI se recogen los resultados de testing obtenidos para los experimentos realizados con testing intradías haciendo uso del **día 20**. A su vez, en la Figura 217 se muestra una comparación de los resultados obtenidos para cada experimento teniendo en cuenta los valores de recall y precisión para distintos umbrales.

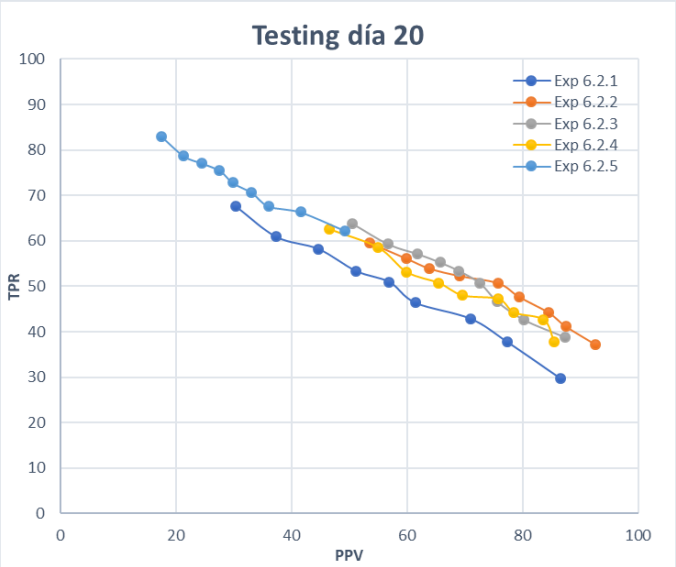

Figura 217. Comparación de todos los experimentos de Tao2016 y testing intradías para el día 20

Tal y como se puede observar en la gráfica los experimentos obtienen peores resultados al analizar un día completamente diferente a los utilizados durante el entrenamiento. Sin embargo, a la hora de comparar los experimentos entre sí con objeto de seleccionar la red óptima se puede llegar a razonamientos similares a los establecidos en el apartado 6.2.6. Como conclusión de esto, se ha podido comprobar que el proceso de selección de la red óptima en base a los resultados de testing intradías es consistente con la selección de la red óptima.

7 Experimentación con la red de Yin

Una vez se ha realizado un amplio conjunto de experimentos con la primera red, explorando el comportamiento de esta ante variaciones en conjuntos de datos, la arquitectura de la red y los hiperparámetros de entrenamiento a utilizar, se procederá a realizar experimentos haciendo uso de la red expuesta en [Yin 2017].

7.1 Modificación inicial de la arquitectura

Al comenzar a trabajar con esta red se ha observado que, haciendo uso de las imágenes de dimensiones 256x192, tras realizar la operación flattening se obtiene un vector de 196.608 características. Este vector al ser multiplicado por las 2048 neuronas de las capas full-connected llegan a generar más de 402 millones de parámetros. En el experimento realizado en el artículo se lleva a cabo el entrenamiento con imágenes muy pequeñas de dimensiones 48x48, generando así un vector de 9.216 características.

La dimensión del vector generado con las imágenes 256x192 es demasiado grande para la GPU, por lo que se deberán realizar cambios en la red de modo que se pueda mantener la arquitectura de red dentro de lo posible aplicando las modificaciones mínimas.

Para ello inicialmente se ha procedido a reducir a la mitad el número de filtros de todas las capas convolucionales de la red, disminuyendo así el número de mapas de características generados. Asimismo, se modificarán las dos primeras capas de pooling de modo que, tanto el stride como el kernel size, tengan dimensiones 4x4. Esto permitirá contar con una última capa de 128 mapas de 8x6 que generará un vector de 6.144 características, dimensión adecuada para la GPU. Tras aplicar estas modificaciones la red tendría la arquitectura de la Tabla 14.

Capa	Entrada	Hiperparámetros
1	3 (256x192)	**Conv1:** 16 (3x3) S=(1x1) Padding + ReLU
	16 (256x192)	**BatchNorm1:** Momentum=0.9
	16 (256x192)	**Conv2:** 16 (3x3) S=(1x1) Padding + ReLU
	16 (256x192)	**BatchNorm2:** Momentum=0.9
	16 (256x192)	**Conv3:** 32 (3x3) S=(1x1) Padding + ReLU
	32 (256x192)	**BatchNorm3:** Momentum=0.9
	32 (256x192)	**MaxPool1:** (4x4) S=(4x4)
2	32 (64x48)	**Conv4:** 32 (3x3) S=(1x1) Padding + ReLU
	32 (64x48)	**BatchNorm4:** Momentum=0.9
	32 (64x48)	**Conv5:** 32 (3x3) S=(1x1) Padding + ReLU
	32 (64x48)	**BatchNorm5:** Momentum=0.9
	32 (64x48)	**MaxPool2:** (4x4) S=(4x4)
3	32 (16x12)	**Conv5:** 192 (3x3) S=(1x1) Padding + ReLU
	192 (16x12)	**BatchNorm5:** Momentum=0.9
	192 (16x12)	**Conv6:** 192 (3x3) S=(1x1) Padding + ReLU
	192 (16x12)	**BatchNorm6:** Momentum=0.9
	128 (16x12)	**Conv7:** 128 (3x3) S=(1x1) Padding + ReLU
	128 (16x12)	**BatchNorm7:** Momentum=0.9
	128 (16x12)	**MaxPool3:** (2x2) S=(2x2)
	6.144	**FC1:** 2.048
	2.048	**FC2:** 2.048
	2.048	**FC3:** 2

Tabla 14. Modificación inicial de la red Yin2017

7.2 Experimentos

En el siguiente apartado se recogerán todos y cada uno de los experimentos realizados con esta nueva red partiendo de la arquitectura modificada en el apartado anterior con objeto de poder ser manejada por la GPU.

7.2.1 Experimentos de modificaciones en capas convolucionales

Siguiendo la metodología llevada con la red [Tao, Zhang, and Wang 2016], se realizará inicialmente una fase de experimentos consistentes en la simplificación de la red mediante la modificación de las capas convolucionales. Esto permitirá comprobar el comportamiento

de la misma ante la eliminación de características que puedan no ser significativas para establecer la clasificación.

7.2.1.1 Experimento inicial

El primer experimento que se realizará con esta red hará uso de los conjuntos de datos intradías resultantes del método EEEEVT (explicado en el aparado 6.2), así como la optimización de la métrica por defecto accuracy y el uso de step decay comenzando con un valor de learning rate de 0,001 y terminando en 0,0001. Esta limitación de learning rate se debe a que valores inferiores a 0,0001 se consideran demasiado pequeños, además en el apartado 6.1.3.10 se comprobó que la red [Tao, Zhang, and Wang 2016] era capaz de aprender mejor con un learning rate de 0,0001 que con uno de 0,00001.

Al igual que en los experimentos realizados con la red [Tao, Zhang, and Wang 2016] se mostrarán los resultados obtenidos del entrenamiento. En la Figura 218 y Figura 219 se comprueba que el comportamiento de la red durante el entrenamiento es similar al obtenido con la red [Tao, Zhang, and Wang 2016], en el que los valores de entrenamiento convergen a su valor óptimo y los resultados de validación se estancan a poca distancia de estos.

A partir de los resultados obtenidos en la Figura 220, Figura 221, Figura 222, Figura 223 y Figura 224 se puede observar que en comparación con los experimentos de la red [Tao, Zhang, and Wang 2016] la diferencia no es drástica. Sin embargo, se puede apreciar una subida en el número de falsos positivos.

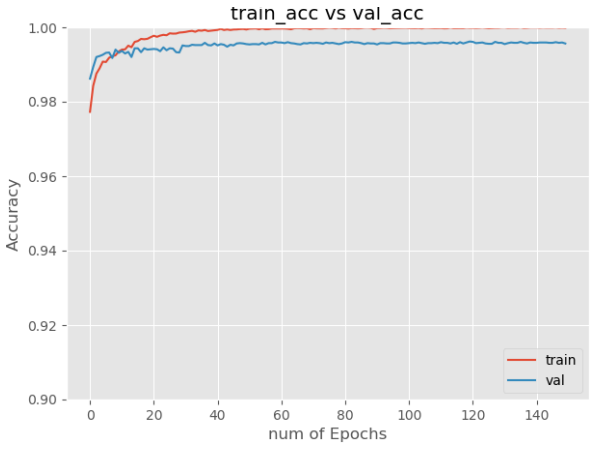

Figura 218. Accuracy del primer experimento con Yin2017

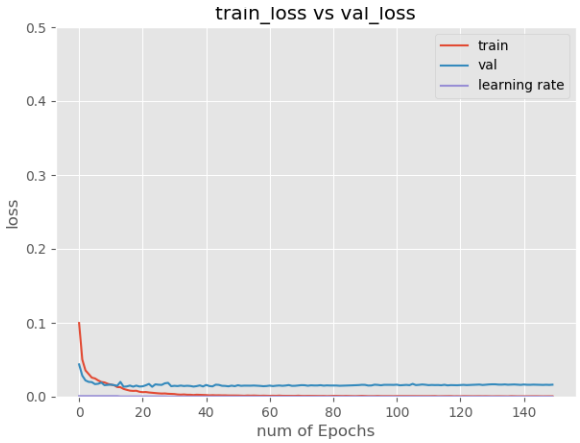

Figura 219. Loss del primer experimento con Yin2017

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	94.02	92.31	91.45	90.6	89.74	88.03	87.61	85.47	79.91
PPV	70.97	76.6	78.97	80.61	84.34	86.55	88.74	90.5	93.97

Figura 220. Umbrales del primer experimento con Yin2017

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	84.34	89.74	99.79	99.65	86.96	99.45

Figura 221. Métricas del primer experimento con Yin2017

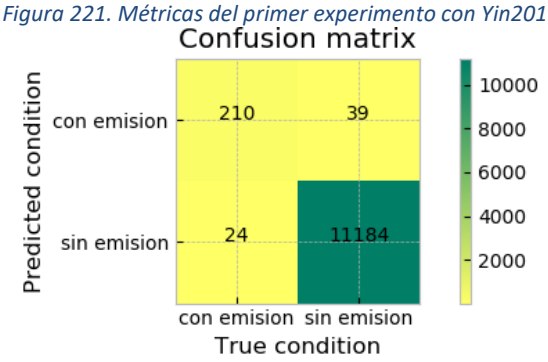

Figura 222. Matriz de confusión del primer experimento con Yin2017

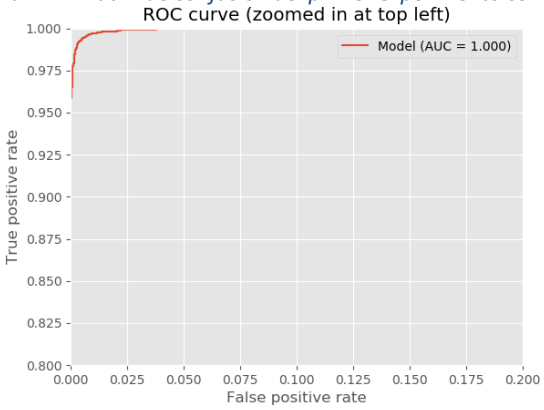

Figura 223. Curva ROC del primer experimento con Yin2017

	TP	FP	FN
0.5	17	9	4

Figura 224. Secuencias de emisiones para el día 20 del primer experimento con Yin2017

7.2.1.2 Reducción del tercer nivel de capas convolucionales

En la sección 7.1 se puede observar que el número de características del vector de flattening aumenta de forma significativa debido al elevado número de filtros realizados en el tercer nivel de capas convolucionales. El siguiente experimento consistirá en realizar una simplificación de la red reduciendo a la mitad el número de filtros de este nivel convolucional respecto a la configuración del experimento anterior, lo que es igual a una reducción de un cuarto del número de filtros de la arquitectura original presentada en [Yin 2017]. La arquitectura de la red a utilizar, por tanto, consistiría en la presentada en la Tabla 15.

Capa	Entrada	Hiperparámetros
1	3 (256x192)	**Conv1:** 16 (3x3) S=(1x1) Padding + ReLU
	16 (256x192)	**BatchNorm1:** Momentum=0.9
	16 (256x192)	**Conv2:** 16 (3x3) S=(1x1) Padding + ReLU
	16 (256x192)	**BatchNorm2:** Momentum=0.9
	16 (256x192)	**Conv3:** 32 (3x3) S=(1x1) Padding + ReLU
	32 (256x192)	**BatchNorm3:** Momentum=0.9
	32 (256x192)	**MaxPool1:** (4x4) S=(4x4)
2	32 (64x48)	**Conv4:** 32 (3x3) S=(1x1) Padding + ReLU
	32 (64x48)	**BatchNorm4:** Momentum=0.9
	32 (64x48)	**Conv5:** 32 (3x3) S=(1x1) Padding + ReLU
	32 (64x48)	**BatchNorm5:** Momentum=0.9
	32 (64x48)	**MaxPool2:** (4x4) S=(4x4)
3	32 (16x12)	**Conv5:** 96 (3x3) S=(1x1) Padding + ReLU
	96 (16x12)	**BatchNorm5:** Momentum=0.9
	96 (16x12)	**Conv6:** 96 (3x3) S=(1x1) Padding + ReLU
	96 (16x12)	**BatchNorm6:** Momentum=0.9
	96 (16x12)	**Conv7:** 64 (3x3) S=(1x1) Padding + ReLU
	64 (16x12)	**BatchNorm7:** Momentum=0.9
	64 (16x12)	**MaxPool3:** (2x2) S=(2x2)
	3072	**FC1:** 2.048
	2.048	**FC2:** 2.048
	2.048	**FC3:** 2

Tabla 15. Reducción 1/4 en el tercer nivel convolucional de la red Yin2017

En la Figura 225, Figura 226, Figura 227Figura 228, Figura 229, Figura 230 y Figura 231 se puede observar que tanto a nivel de secuencias para el día 20 como a nivel de imagen con el conjunto intradías de testing el número de falsos positivos es menor, sin embargo esto repercute en un decremento de verdaderos positivos, lo que se traduce en que la red clasifica más imágenes como no emisión que la resultante del experimento anterior.

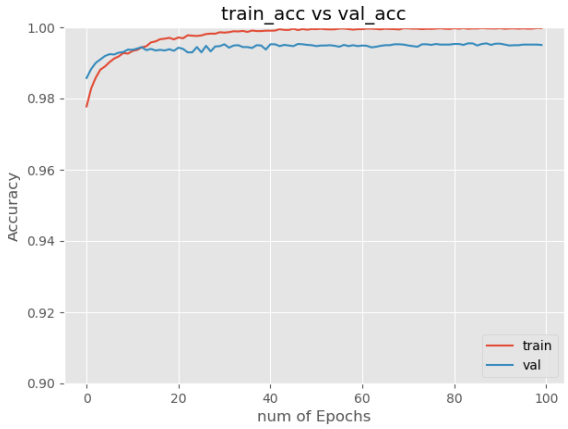

Figura 225. Accuracy de reducción del tercer nivel de capas convolucionales en Yin2017

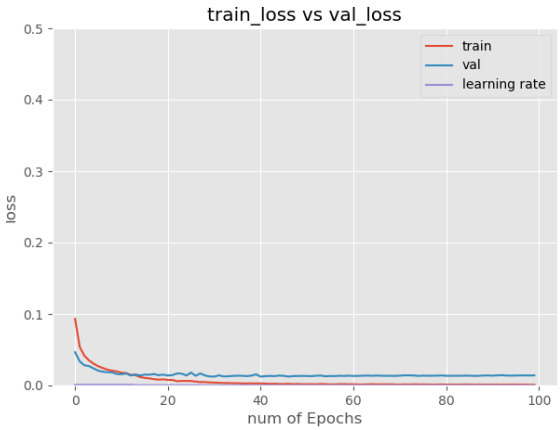

Figura 226. Loss de reducción del tercer nivel de capas convolucionales en Yin2017

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	92.74	90.6	88.89	87.61	86.75	86.75	85.9	84.19	79.06
PPV	77.78	81.54	84.21	85.42	87.12	88.65	89.33	91.63	92.5

Figura 227. Umbrales de reducción del tercer nivel de capas convolucionales en Yin2017

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	87.12	86.75	99.72	99.73	86.94	99.47

Figura 228. Métricas de reducción del tercer nivel de capas convolucionales en Yin2017

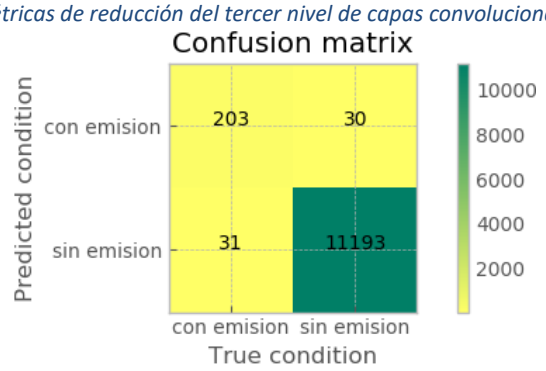

Figura 229. Matriz de confusión de reducción del tercer nivel de capas convolucionales en Yin2017

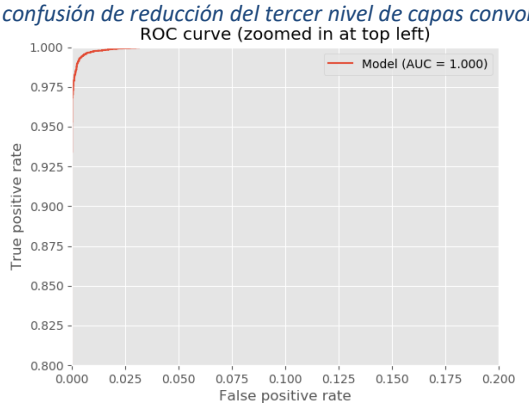

Figura 230. Curva ROC de reducción del tercer nivel de capas convolucionales en Yin2017

	TP	FP	FN
0.5	15	5	6

Figura 231. Secuencias de emisiones para el día 20 del tercer nivel de capas convolucionales en Yin2017

7.2.1.3 Eliminación de capas en el primer y tercer nivel de capas convolucionales

Debido a que en el anterior experimento se ha observado como la simplificación de la arquitectura de la red ha dado lugar a una minimización de falsos positivos tanto a nivel de secuencias como de imágenes de forma independiente, se proseguirá realizando reducciones a nivel de capas convoluciones con objeto de seguir optimizando la red.

Capa	Entrada	Hiperparámetros
1	3 (256x192)	**Conv1:** 16 (3x3) S=(1x1) Padding + ReLU
	16 (256x192)	**BatchNorm1:** Momentum=0.9
	16 (256x192)	**Conv2:** 32 (3x3) S=(1x1) Padding + ReLU
	32 (256x192)	**BatchNorm2:** Momentum=0.9
	32 (256x192)	**MaxPool1:** (4x4) S=(4x4)
2	32 (64x48)	**Conv3:** 32 (3x3) S=(1x1) Padding + ReLU
	32 (64x48)	**BatchNorm3:** Momentum=0.9
	32 (64x48)	**Conv4:** 32 (3x3) S=(1x1) Padding + ReLU
	32 (64x48)	**BatchNorm4:** Momentum=0.9
	32 (64x48)	**MaxPool2:** (4x4) S=(4x4)
3	32 (16x12)	**Conv5:** 96 (3x3) S=(1x1) Padding + ReLU
	96 (16x12)	**BatchNorm5:** Momentum=0.9
	96 (16x12)	**Conv6:** 64 (3x3) S=(1x1) Padding + ReLU
	64 (16x12)	**BatchNorm6:** Momentum=0.9
	64 (16x12)	**MaxPool3:** (2x2) S=(2x2)
	3072	**FC1:** 2.048
	2.048	**FC2:** 2.048
	2.048	**FC3:** 2

En la Figura 232, Figura 233, Figura 234, Figura 235, Figura 236, Figura 237 y Figura 238 se muestran los resultados de entrenamiento y testing, a partir de los cuales se puede observar que la nueva modificación de la arquitectura ha repercutido de tal modo que el número de falsos positivos aumenta a nivel de imagen y de secuencia y el número de verdaderos positivos aumenta levemente a nivel de imagen y se mantiene para las secuencias.

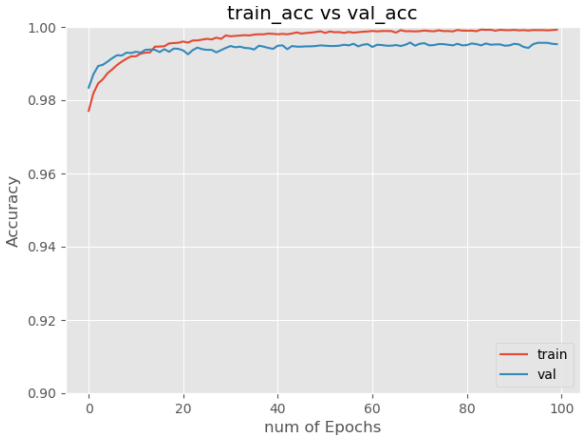

Figura 232. Accuracy de eliminación de capas en el primer y tercer nivel de capas convolucionales en Yin2017

Figura 233. Loss de eliminación de capas en el primer y tercer nivel de capas convolucionales en Yin2017

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	93.59	91.45	91.03	89.32	87.61	85.47	85.47	84.19	79.91
PPV	77.66	80.45	82.56	84.62	85.77	86.96	88.5	92.49	93.5

Figura 234. Umbrales de eliminación de capas en el primer y tercer nivel de capas convolucionales en Yin2017

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	85.77	87.61	99.74	99.7	86.68	99.45

Figura 235. Métricas de eliminación de capas en el primer y tercer nivel de capas convolucionales en Yin2017

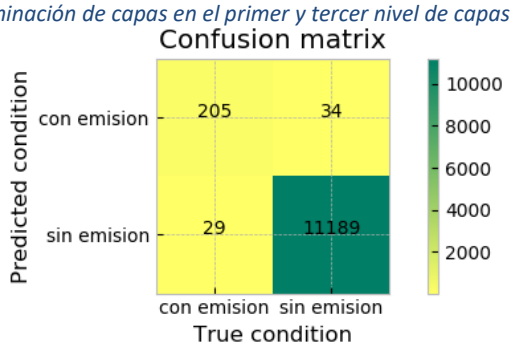

Figura 236. Matriz de confusión de eliminación de capas en primer y tercer nivel de capas convolucionales en Yin2017

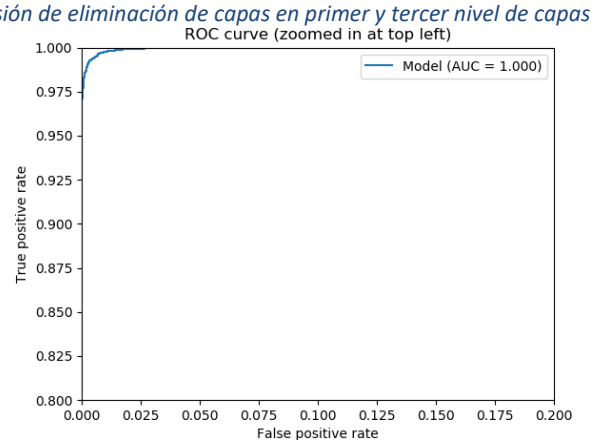

Figura 237. Curva ROC de eliminación de capas en el primer y tercer nivel de capas convolucionales en Yin2017

	TP	FP	FN
0.5	15	6	6

Figura 238. Secuencias de emisiones para el día 20 de eliminación de capas en el primer y tercer nivel de capas convolucionales en Yin2017

7.2.2 Experimentos de modificaciones en capas full connected

En la sección anterior se realizaron modificaciones en las capas convolucionales hasta que se dio con un experimento que obtuvo un rendimiento menor ante la simplificación de la red. En dicha sección se ha podido comprobar que el experimento del apartado 7.2.1.2 obtiene un mejor rendimiento, es por ello por lo que en la sección actual se procederá a realizar modificaciones en la capas full connected a partir de la arquitectura resultante de dicho experimento.

Durante la fase de experimentación con la red [Tao, Zhang, and Wang 2016] se comprobó que dicha red obtenía un mejor rendimiento con dos capas con 4096 neuronas que con 2048 neuronas. En base a esta información se comenzará esta fase de experimentación incrementado el número de neuronas de capa en capa y se finalizará decrementándolo con objeto de comprobar la repercusión del número de neuronas en las distintas métricas de evaluación.

La Tabla 16 muestra de forma gráfica el orden que se seguirá con los experimentos de variación de número de neuronas en el que el experimento 1 se trata del ya realizado en el apartado 7.2.1.2.

		2ª capa		
		1024	2048	4096
1ª capa	4096		**2**	**3**
	2048	**4**	**1**	
	1024	**5**		

Tabla 16. Experimentos consistentes en variación del nº de neuronas

7.2.2.1 Incremento de número de neuronas en primera capa full connected

Siguiendo la Tabla 16 el experimento consistente en la primera variación del número de neuronas se trata del experimento 2, consistente en el incremento del número de neuronas en la primera neurona de 2048 a 4096.

En la Figura 239, Figura 240, Figura 241, Figura 242, Figura 243, Figura 244 y Figura 245 se puede observar como el comportamiento de la red es bastante similar, sin embargo, el número de falsas alarmas a nivel de secuencias decrementa de 5 a 2 manteniendo a 15 el número de verdaderos positivos. A nivel de imagen de forma individual el número de verdaderos positivos incrementa mientras se mantienen los falsos positivos en cuanto al experimento 7.2.1.2.

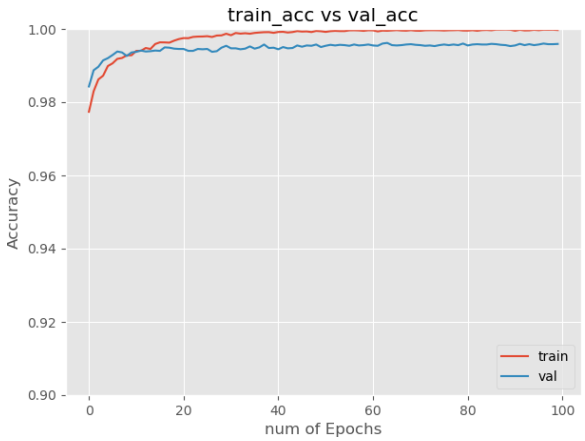

Figura 239. Accuracy de incremento de número de neuronas en primera capa full connected en Yin2017

Figura 240. Loss de incremento de número de neuronas en primera capa full connected en Yin2017

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	92.74	91.88	91.03	89.74	88.03	86.75	85.04	83.33	80.77
PPV	78.62	83.33	85.2	86.07	87.29	87.88	89.24	90.7	91.75

Figura 241. Umbrales de incremento de número de neuronas en primera capa full connected en Yin2017

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	87.29	88.03	99.75	99.73	87.66	99.49

Figura 242. Métricas de incremento de número de neuronas en primera capa full connected en Yin2017

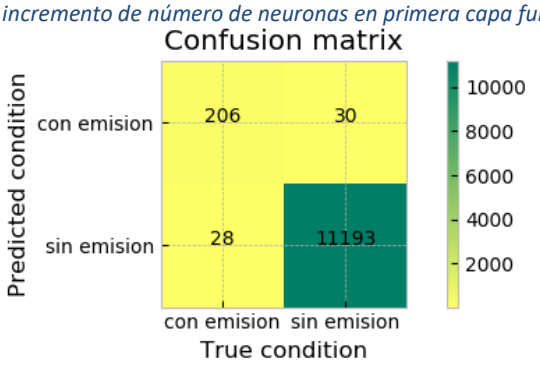

Figura 243. Matriz de confusión de incremento de número de neuronas en primera capa full connected en Yin2017

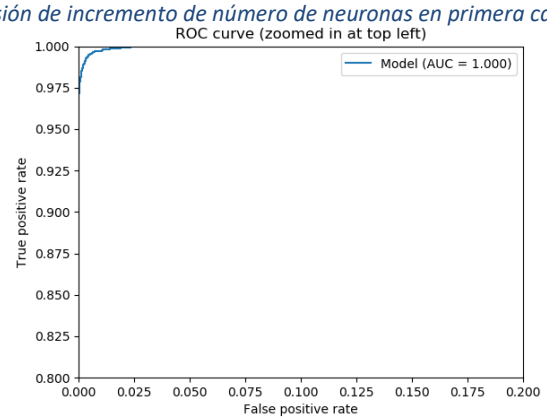

Figura 244. Curva ROC de incremento de número de neuronas en primera capa full connected en Yin2017

	TP	FP	FN
0.5	15	2	6

Figura 245. Secuencias de emisiones para el día 20 de incremento de número de neuronas en primera capa full connected en Yin2017

7.2.2.2 Incremento de número de neuronas en ambas capas full connected

Una vez comprobado que la red obtiene un mejor rendimiento aumentando el número de neuronas en la primera capa full connected, se procederá a comprobar el comportamiento de la red ante una arquitectura con dos capas full connected de 4096 neuronas.

La Figura 246, Figura 247, Figura 248, Figura 249, Figura 250, Figura 251 y Figura 252 se puede observar que a pesar de decrementar los falsos negativos en el testing de imágenes individuales, los falsos negativos en secuencias se mantienen. Por otro lado, los falsos positivos incrementan en ambos casos.

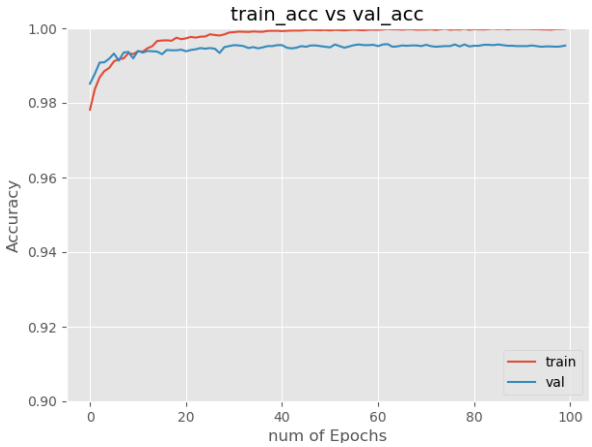

Figura 246. Accuracy de incremento de número de neuronas en ambas capas full connected en Yin2017

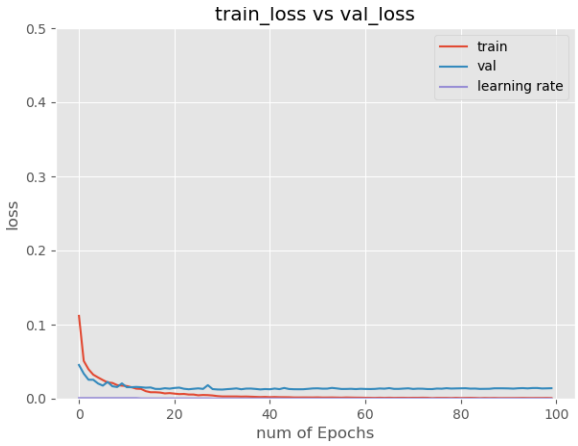

Figura 247. Loss de incremento de número de neuronas en ambas capas full conecte en Yin2017

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	93.16	92.31	91.03	90.17	89.74	88.03	85.47	84.19	80.77
PPV	76.22	80.3	81.92	84.4	86.07	87.29	88.89	91.2	94.03

Figura 248. Umbrales de incremento de número de neuronas en ambas capas connected en Yin2017

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	86.07	89.74	99.79	99.7	87.87	99.49

Figura 249. Métricas de incremento de número de neuronas en ambas capas full connected en Yin2017

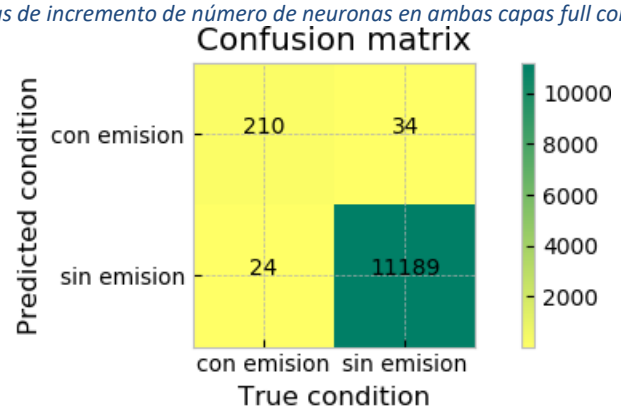

Figura 250. Matriz de confusión de incremento de número de neuronas en ambas capas full connected en Yin2017

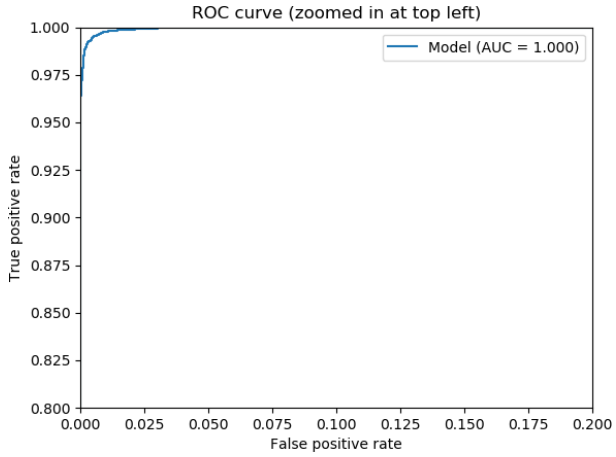

Figura 251. Curva ROC de incremento de número de neuronas en ambas capas full connected en Yin2017

	TP	FP	FN
0.5	15	6	6

Figura 252. Secuencias de emisiones para el día 20 de incremento de número de neuronas en ambas capas full connected en Yin2017

7.2.2.3 Reducción de número de neuronas en segunda capa full connected

A continuación, se procederá a experimentar con el caso contrario a los dos anteriores mediante el decremento de número de neuronas en la segunda capa full connected. Esta vez se aplicará el cambio a la segunda capa y no a la primera ya que se seguirá la lógica de comenzar con un mayor número de neuronas en la primera capa y reducir este número en la segunda.

A partir de la Figura 253, Figura 254, Figura 255, Figura 256, Figura 257, Figura 258 y Figura 259 comparando con el experimento 7.2.1.2 se puede comprobar tanto los verdaderos positivos como los falsos positivos decrementan, incrementando así la cantidad de falsos negativos.

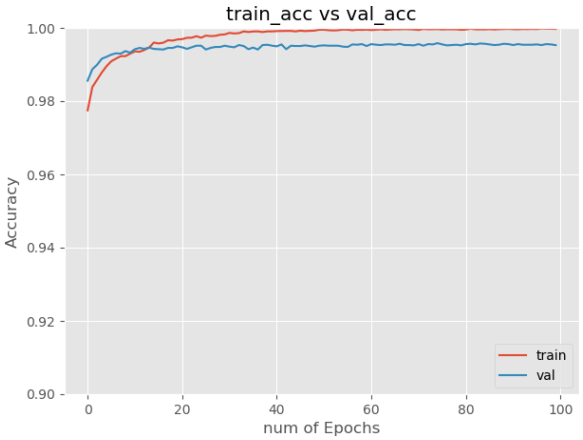

Figura 253. Accuracy de reducción de número de neuronas en segunda capa full connected en Yin2017

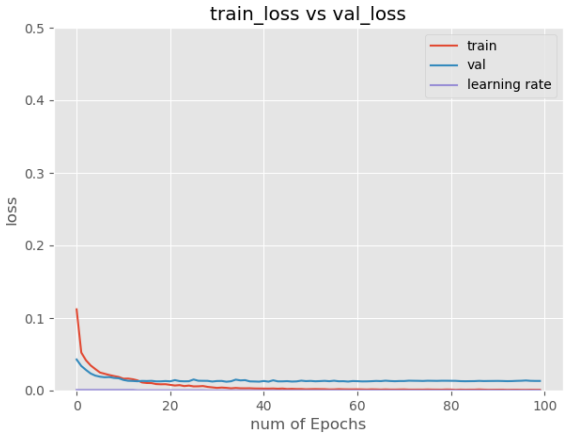

Figura 254. Loss de reducción de número de neuronas en segunda capa full connected en Yin2017

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	93.16	91.88	91.03	89.32	88.46	86.75	86.32	84.62	80.77
PPV	77.86	82.69	85.2	86.01	86.97	88.65	90.58	91.67	94.5

Figura 255. Umbrales de reducción de número de neuronas en segunda capa full connected en Yin2017

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	86.97	88.46	99.76	99.72	87.71	99.49

Figura 256. Métricas de reducción de número de neuronas en segunda capa full connected en Yin2017

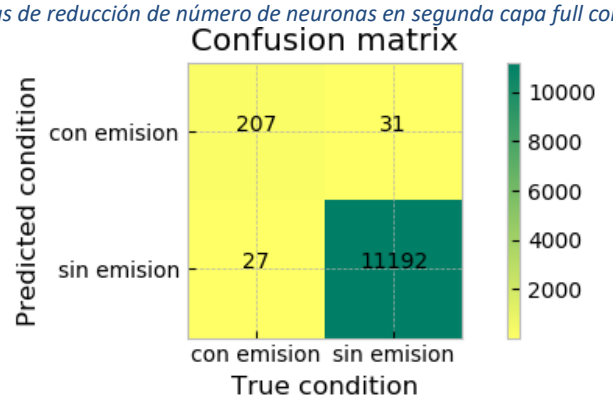

Figura 257. Matriz de confusión de reducción de número de neuronas en segunda capa full connected en Yin2017

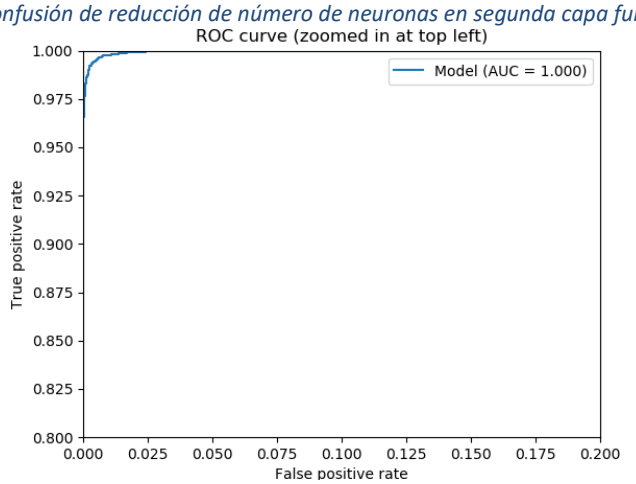

Figura 258. Curva ROC de reducción de número de neuronas en segunda capa full connected en Yin2017

	TP	FP	FN
0.5	14	4	7

Figura 259. Secuencias de emisiones para el día 20 de reducción de número de neuronas en segunda capa full connected en Yin2017

7.2.2.4 Reducción de número de neuronas en ambas capas full connected

Finalmente se probará a comprobar el comportamiento de la red ante la reducción del número de neuronas a 1024 en las dos capas full connected.

Como resultado de este experimento se obtienen la Figura 260, Figura 261, Figura 262, Figura 263, Figura 264, Figura 265 y Figura 266 en las que se puede comprobar como decrementando el número de neuronas en la primera capa full connected el número de

falsos positivos y de verdaderos positivos decrementa aún más que en el experimento 7.2.2.3, siguiendo su mismo comportamiento.

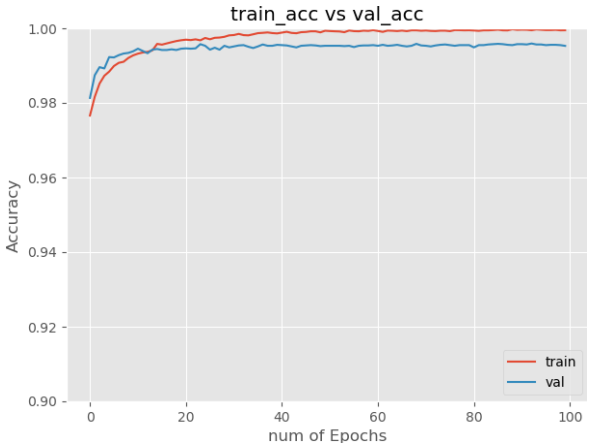

Figura 260. Accuracy de reducción de número de neuronas en ambas capas full connected en Yin2017

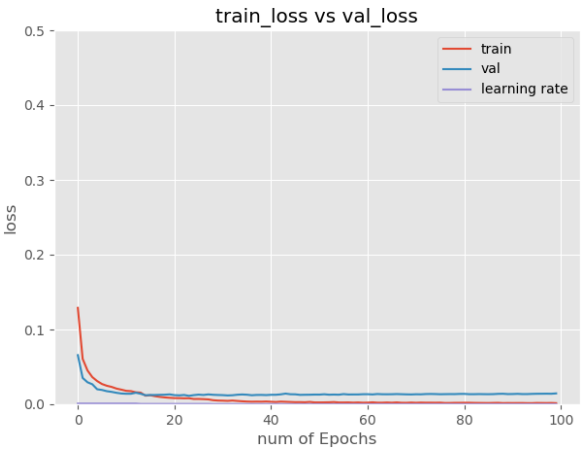

Figura 261. Loss de reducción de número de neuronas en ambas capas full connected en Yin2017

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	93.59	91.88	90.17	88.03	87.18	83.76	80.77	76.92	74.36
PPV	73.0	78.18	81.15	85.48	87.93	88.69	91.75	95.74	96.13

Figura 262. Umbrales de reducción de número de neuronas en ambas capas connected en Yin2017

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	87.93	87.18	99.73	99.75	87.55	99.49

Figura 263. Métricas de reducción de número de neuronas en ambas capas full connected en Yin2017

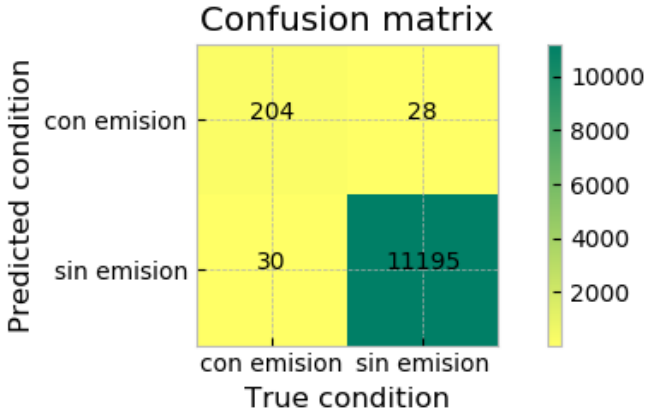

Figura 264. Matriz de confusión de reducción de número de neuronas en ambas capas full connected en Yin2017

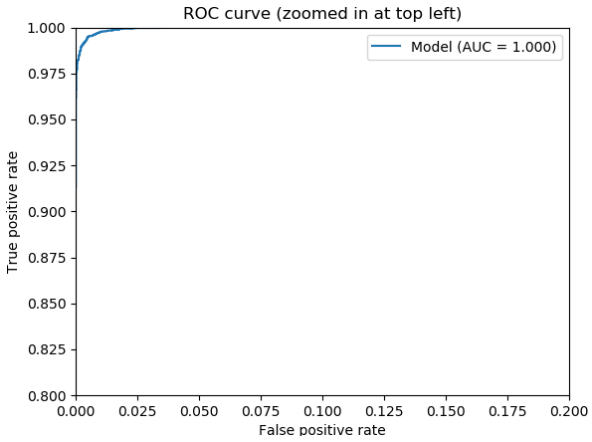

Figura 265. Curva ROC de reducción de número de neuronas en ambas capas full connected en Yin2017

	TP	FP	FN
0.5	13	3	8

Figura 266. Secuencias de emisiones para el día 20 de reducción de número de neuronas en ambas capas full connected en Yin2017

7.2.3 Experimentos de modificaciones de hiperparámetros de entrenamiento

Una vez realizados los experimentos pertinentes a la modificación de la arquitectura de la red tanto en capas convolucionales como en capas full connected se procederá a modificar parámetros de optimización del entrenamiento como pueden ser la métricas a optimizar o el uso de penalización en función al desbalanceo de clases, siguiendo el proceso de experimentación seguido con la red [Tao 2016]. Para ello se hará uso de la red resultante del experimento 7.2.2.1, ya que es el que ha obtenido los resultados de testing óptimos hasta ahora.

7.2.3.1 Experimento con optimización de la métrica precisión

El primer experimento que se realizará en la fase de modificación de hiperparámetros será el aplicar la optimización de la métrica precisión en vez de accuracy durante el entrenamiento.

A partir de la Figura 267 y Figura 268 se observa que la fase de entrenamiento mediante optimización de precisión sigue un comportamiento similar al obtenido en experimentos que optimizan la métrica accuracy. Por otro lado, la Figura 269, Figura 270, Figura 271, Figura 272 y Figura 273 permiten comprobar que este experimento ha obtenido un aumento de falsos positivos y falsos negativos a nivel de clasificación de imagen individualmente. A nivel de detección de secuencias de emisiones los resultados muestran que la red resultante detecta dos emisiones más que la red resultante de la optimización de accuracy, sin embargo, esto repercute en un incremento de falsas alarmas de 2 a 6.

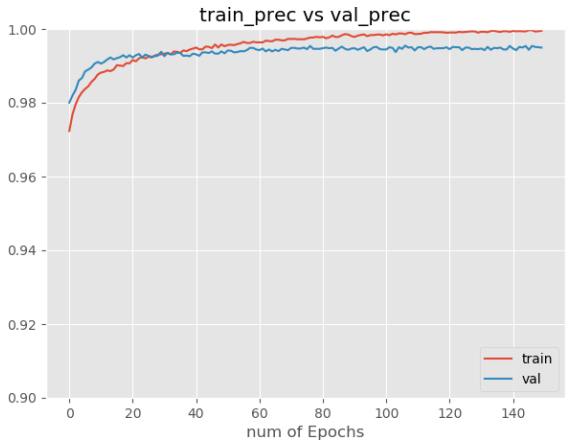

Figura 267. Precision de optimización de precision en Yin2017

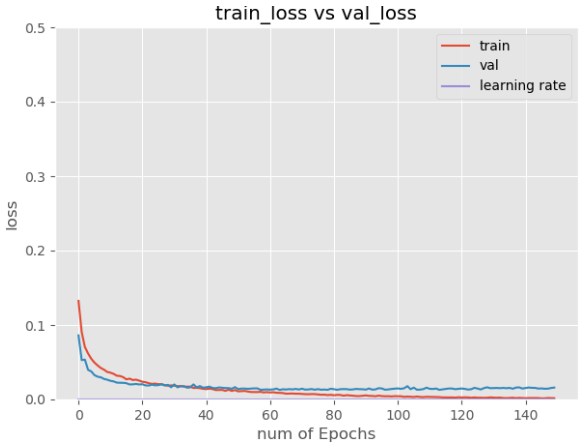

Figura 268. Loss de optimización de precision en Yin2017

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	95.73	92.31	90.17	88.89	87.18	84.62	81.62	79.06	73.93
PPV	70.89	76.33	82.42	85.6	86.81	88.79	90.09	92.96	95.58

Figura 269. Umbrales de optimización de precision en Yin2017

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	86.81	87.18	99.73	99.72	86.99	99.47

Figura 270. Métricas de optimización de precision en Yin2017

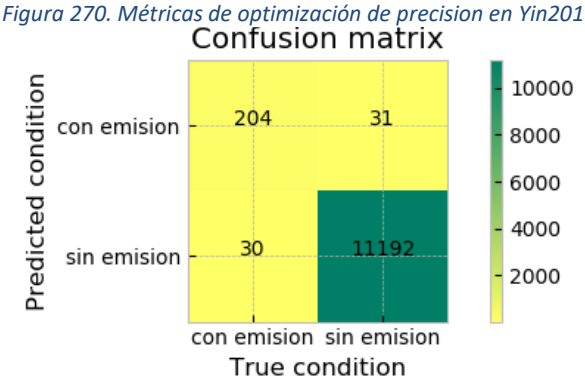

Figura 271. Matriz de confusión de optimización de precision en Yin2017

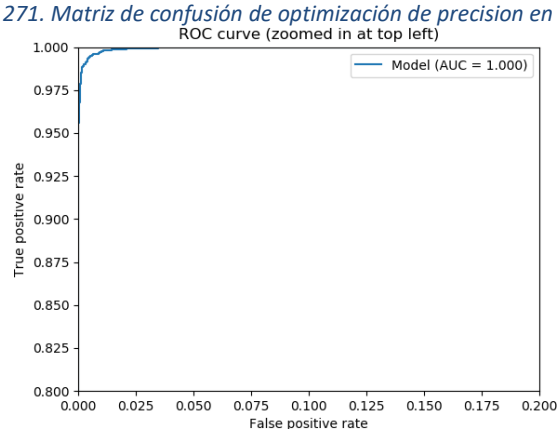

Figura 272. Curva ROC de optimización de precision en Yin2017

	TP	FP	FN
0.5	17	6	4

Figura 273. Secuencias de emisiones para el día 20 de optimización de precision en Yin2017

7.2.3.2 Experimento con optimización de la métrica recall

Una vez realizado el experimento consistente en la optimización de precisión, se probará a optimizar la red durante el entrenamiento mediante la métrica recall.

La Figura 274 y Figura 275, al igual que en el experimento anterior, permite observar que la red sigue el mismo patrón de entrenamiento para la métrica recall que para la métrica accuracy o precisión. Por parte de los resultados de testing representados por la Figura 276, Figura 277, Figura 278, Figura 279 y Figura 280 se observa que esta vez los falsos positivos decrementan notablemente a nivel de imagen individual pero se mantiene en 2 a nivel de secuencia de emisiones. Esto ha repercutido en un incremento de falsos negativos en ambas clasificaciones (individual y de secuencias).

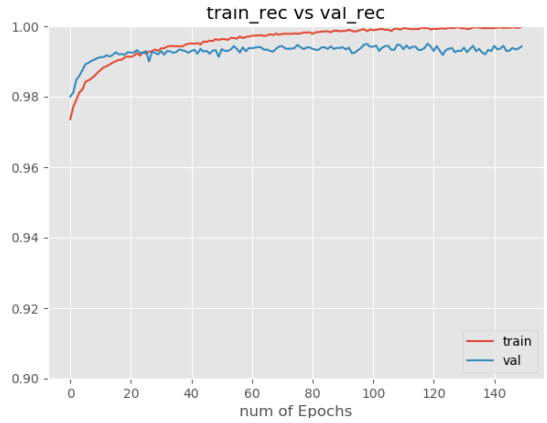

Figura 274. Recall de optimización de recall en Yin2017

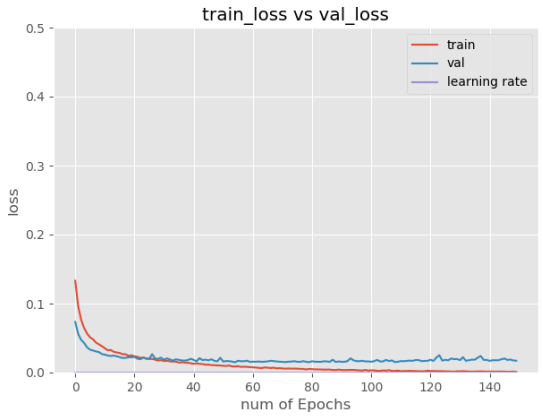

Figura 275. Loss de optimización de recall en Yin2017

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	88.89	86.32	85.9	84.19	82.05	79.49	77.35	74.79	68.8
PPV	74.82	81.12	85.9	88.74	90.14	92.08	92.82	94.59	95.83

Figura 276. Umbrales de optimización de recall en Yin2017

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	90.14	82.05	99.63	99.81	85.91	99.45

Figura 277. Métricas de optimización de recall en Yin2017

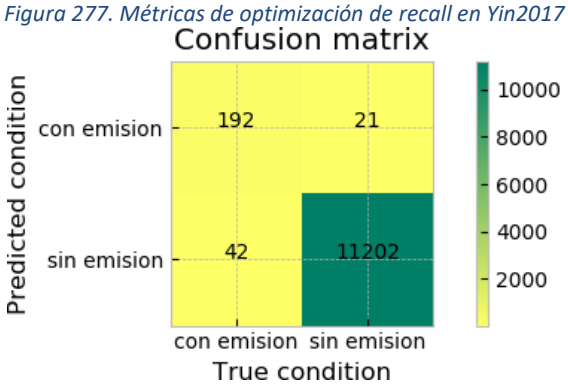

Figura 278. Matriz de confusión de optimización de recall en Yin2017

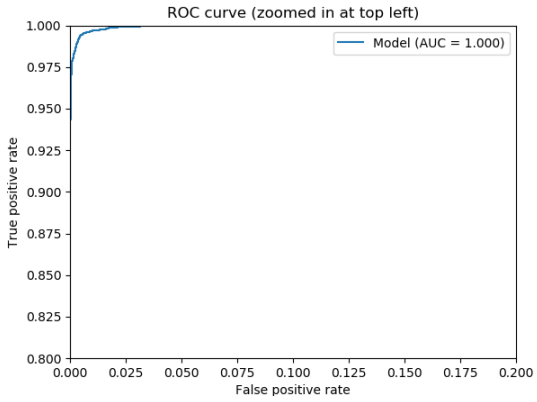

Figura 279. Curva ROC de optimización de recall en Yin2017

	TP	FP	FN
0.5	14	2	7

Figura 280. Secuencias de emisiones para el día 20 de optimización de recall en Yin2017

7.2.3.3 Experimento con penalización de loss

Como experimento final se probará a entrenar la red optimizando la métrica accuracy y aplicando una penalización correspondiente al desbalanceo existente entre clases.

La Figura 281 y Figura 282 permiten comprobar que el uso de penalización de las métricas durante el entrenamiento proporciona unas gráficas algo más discontinuas con picos en ciertas epochs.

En cuanto a la fase de testing la Figura 283, Figura 284, Figura 285, Figura 286 y Figura 287 muestran un comportamiento totalmente inverso al del anterior experimento debido a que esta vez decrementan notablemente el número de falsos negativos, sobre todo a nivel de imágenes individuales. Sin embargo, esto repercute negativamente aumentando también el número de falsos positivos.

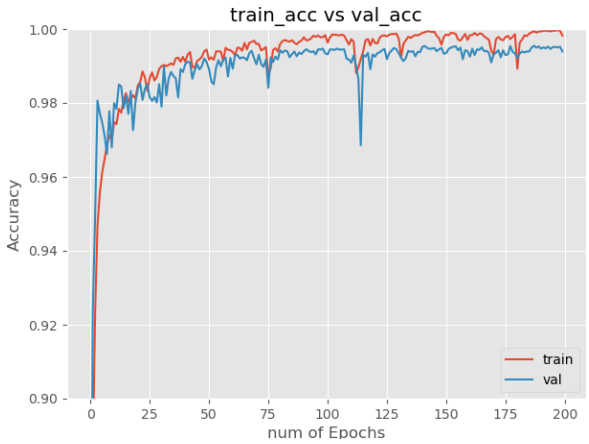

Figura 281. Accuracy de penalización de loss en Yin2017

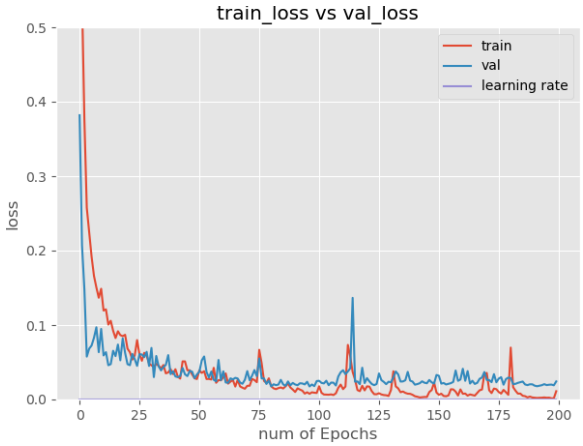

Figura 282. Loss de penalización de loss en Yin2017

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	94.87	94.02	94.02	93.59	93.16	92.74	92.31	91.45	87.61
PPV	77.89	79.42	80.59	80.81	81.65	83.46	84.71	85.6	87.23

Figura 283. Umbrales de penalización de loss en Yin2017

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	81.65	93.16	99.86	99.56	87.03	99.43

Figura 284. Métricas de penalización de loss en Yin2017

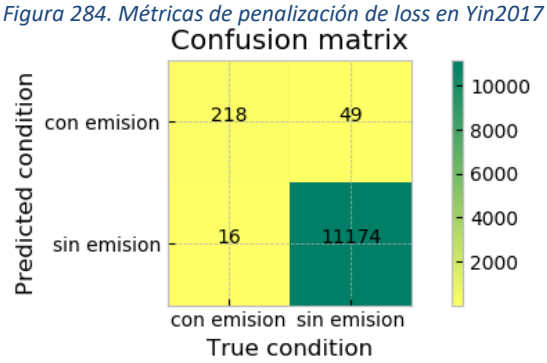

Figura 285. Matriz de confusión de penalización de loss en Yin2017

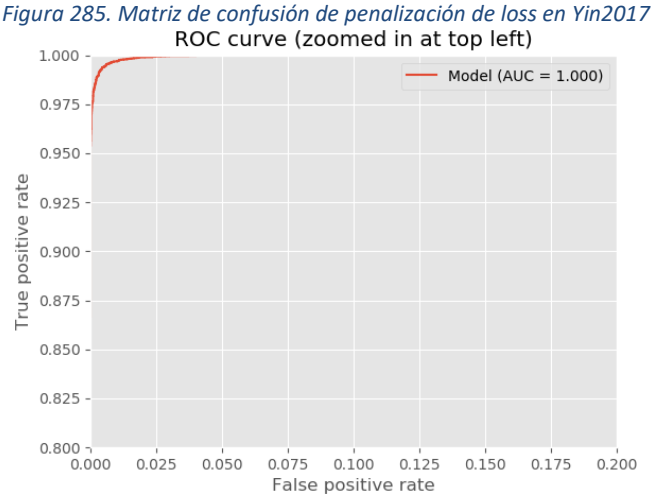

Figura 286. Curva ROC de penalización de loss en Yin2017

	TP	FP	FN
0.5	17	8	4

Figura 287. Secuencias de emisiones para el día 20 de penalización de loss en Yin2017

7.3 Selección de la red de Yin óptima

Una vez realizado el conjunto de experimentos referente a la red [Yin 2017] se deberá seleccionar la red óptima dentro del conjunto de redes resultantes de los experimentos realizados. Para ello se hará uso del mismo método utilizado en el apartado 6.2.6, comparando los distintos experimentos mediante una gráfica que represente los valores de recall y precisión para distintos umbrales.

A continuación se muestran la Figura 288, Figura 289, Figura 290, Figura 291, Figura 292, Figura 293, Figura 294 y Figura 295, consistentes en gráficas comparativas de los distintos experimentos realizados en la sección 7.2. Tal y como se realizó con la red [Tao 2016] se comparan los valores de las métricas precisión y recall para diversos umbrales en cada uno de los experimentos, por lo que cuanto más se aproxime a la esquina superior derecha mejor será el rendimiento de la red. Por otro lado, cabe destacar que en el sistema debe prevalecer la optimización de la precisión que la de recall, ya que se debe evitar en su mayor medida las falsas alarmas.

Los resultados mostrados en las gráficas se corresponden a los obtenidos mediante el conjunto de testing resultante del método de división EEEEVT y del conjunto resultante de utilizar el día 20 completo.

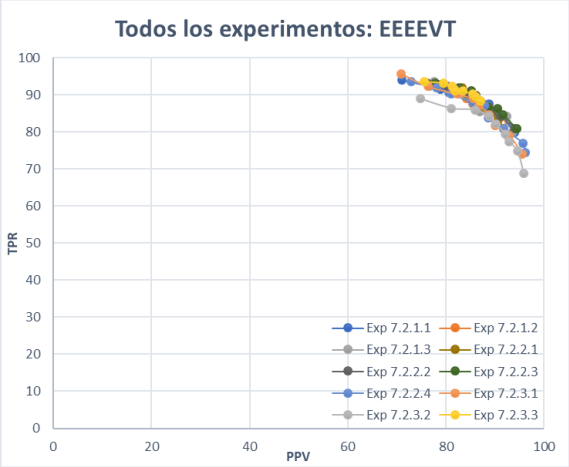

Figura 288. Todos los experimentos con Yin2017 con el conjunto de testing EEEEVT

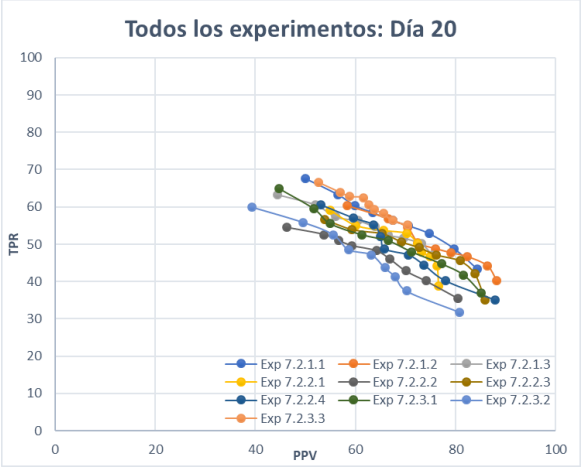

Figura 289. Todos los experimentos con Yin2017 con el día 20

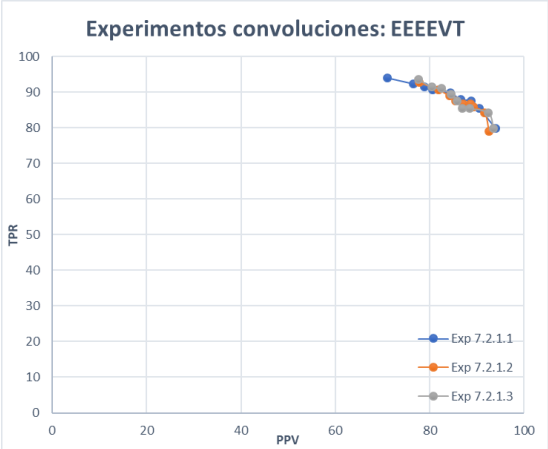

Figura 290. Experimentos de capas convolucionales en Yin2017 con el conjunto de testing EEEEVT

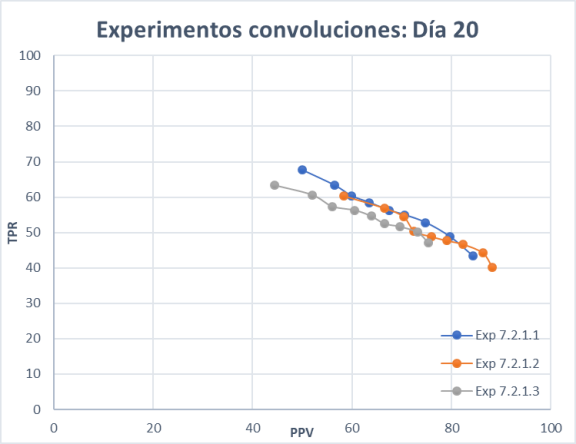

Figura 291. Experimentos de capas convolucionales en Yin2017 con el día 20

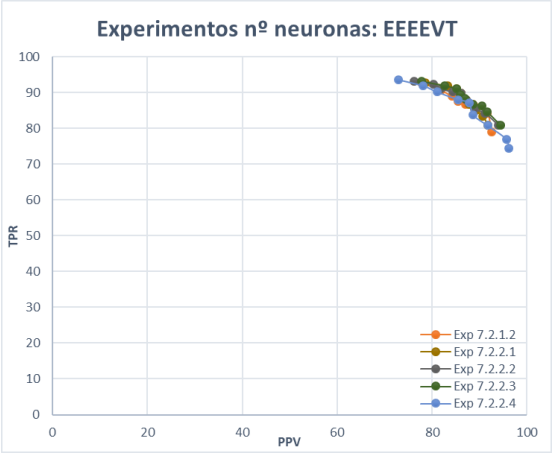

Figura 292. Experimentos de nº de neuronas en Yin2017 con el conjunto de testing EEEEVT

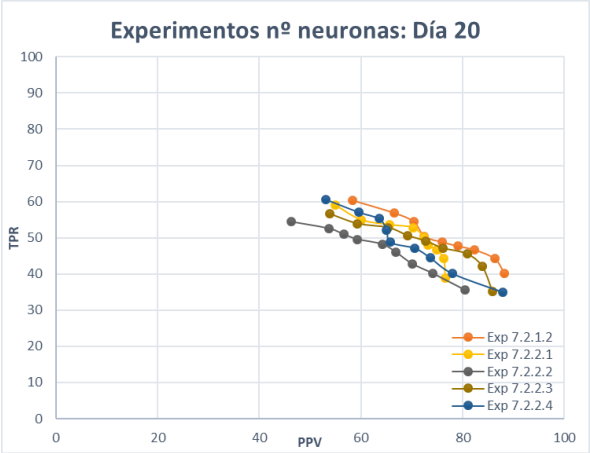

Figura 293. Experimentos de nº de neuronas en Yin2017 con el día 20

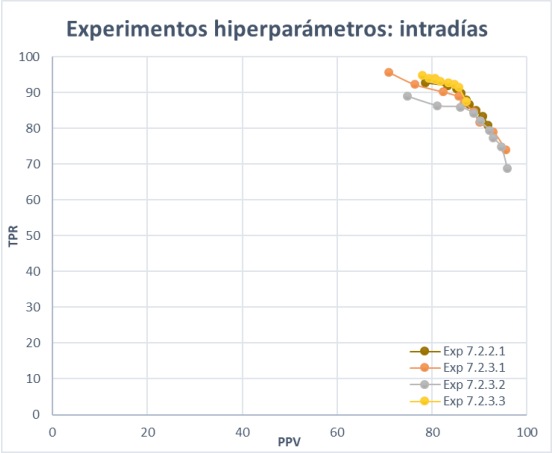

Figura 294. Experimentos hiperparámetros en Yin2017 con el conjunto de testing EEEEVT

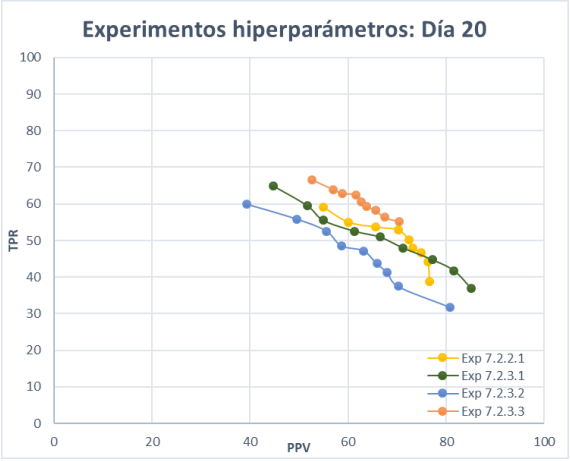

Figura 295. Experimentos de hiperparámetros en Yin2017 con el día 20

En la Tabla 17 se muestran los resultados obtenidos para el testing de secuencias de emisiones.

	TP	FP	FN	TPR	PPV	Threshold
Exp 7.2.1.1	17	9	4	80,95 %	65,38 %	0,5
Exp 7.2.1.2	15	5	6	71,43 %	75,00 %	0,5
Exp 7.2.1.3	15	6	6	71,43 %	71,43 %	0,5
Exp 7.2.2.1	15	2	6	71,43 %	88,24 %	0,5
Exp 7.2.2.2	15	6	6	71,43 %	71,43 %	0,5
Exp 7.2.2.3	14	4	7	66,67 %	77,78 %	0,5
Exp 7.2.2.4	13	3	8	61,90 %	81,25 %	0,5
Exp 7.2.3.1	17	6	4	80,95 %	73,91 %	0,5
Exp 7.2.3.2	14	2	7	66,67 %	87,50 %	0,5
Exp 7.2.3.3	17	8	4	80,95 %	68,00 %	0,5

Tabla 17. Comparación de resultados de predicción de secuencias de red Yin2017

Durante la realización de los experimentos en el apartado 7.2 se ha ido analizando los resultados obtenidos de la fase de testing tanto a nivel individual como de secuencias. En base a este análisis, junto con los resultados de las gráficas anteriores y de la tabla que recopila la predicción de secuencias de cada experimento, se ha seleccionado como red óptima la resultante del experimento 7.2.2.1.

En base a que los resultados del conjunto EEEEVT son muy similares, esta selección se ha realizado eligiendo el experimento con menor número de falsas alarmas para el conjunto de testing del día 20, lo que deja tan solo los experimentos del apartado 7.2.2.1 y 7.2.3.3. A partir de estos dos experimentos se ha seleccionado el que cuente con un porcentaje de detección (TPR) mayor.

7.4 Cross testing con la red seleccionada

Para finalizar la experimentación con la red [Yin 2017] se realizarán los experimentos de cross testing pertinentes con los conjuntos resultantes de los métodos de división de conjuntos EEVTEE y VTEEEE.

Debido a que el objeto de esta sección es comparar los resultados de los 3 experimentos en conjunto y no de forma individual los resultados de los dos experimentos realizados de forma adicional se encuentran en el Anexo VII. De este modo se incluye la Figura 296 como representación de la comparación de cada uno de los experimentos de cross testing para la red utilizada en el apartado 7.2.2.1.

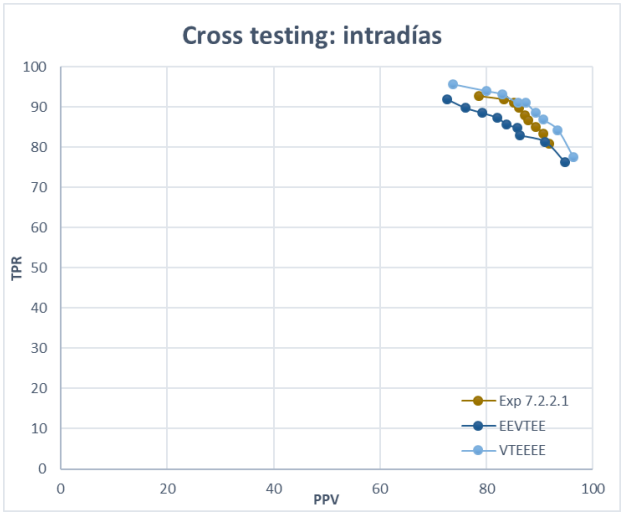

Figura 296. Cross testing de Yin2017 con testing intradías

A partir de esta figura se observa que las curvas representativas de los valores de precisión y accuracy en cada experimento no son idénticas, pero sí son similares, comprobando que la red aprende análogamente ante conjuntos de entrenamiento, validación y testing diferentes y que los **resultados de testing obtenidos son consistentes**.

8 Experimentación con la red caffenet de Hohberg

Como última red de la fase de experimentación del presente proyecto se hará uso de la red caffenet utilizada en la tesis doctoral [Hohberg 2015]. Esta red sigue un enfoque similar a la red [Tao, Zhang, and Wang 2016] aplicando una reducción de características drástica en la primera capa convolucional a través de un filtro convolucional con un stride de dimensiones 4x4.

8.1 Experimentos

A continuación, se llevarán a cabo todos y cada uno de los experimentos que se consideren necesarios para dar con una posible optimización de la red original caffenet.

8.1.1 Experimento con caffenet original

En esta tesis [Hohberg 2015] se hace uso de una reducción de mapas de características de ¼ de la red caffenet debido a que esta red está destinada a manejar 1000 clases distintas y el sistema de detección de emisiones que se pretende implementar se trata de un clasificador binario. Sin embargo, se realizará como prueba inicial un experimento con la arquitectura de red original caffenet.

En la Figura 297 y Figura 298 se muestran los resultados del entrenamiento en el que se observa que esta nueva red no se aproxima al valor óptimo de loss ni durante el entrenamiento ni durante la validación. Los resultados de testing, por otro lado, representados por la Figura 299, Figura 300, Figura 301, Figura 302 y Figura 303 muestran unos resultados un tanto similares a los obtenidos hasta el momento con otras redes. A pesar de esto se puede observar que, tanto clasificando imágenes independientes como secuencias, la proporción de falsos negativos es más alta de lo obtenido de forma usual hasta el momento.

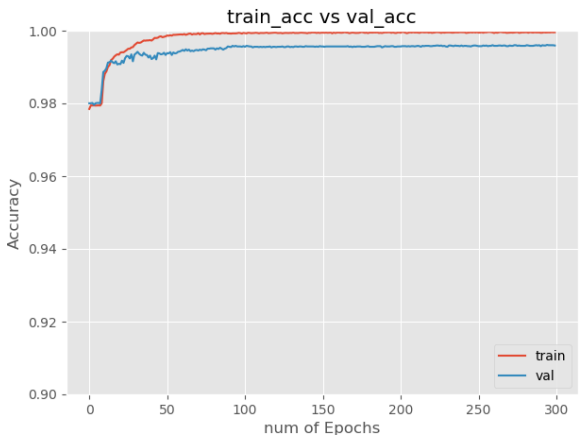

Figura 297. Accuracy de experimento con caffenet original

Figura 298. Loss de experimento con caffenet original

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	90.6	87.61	85.9	85.04	83.33	81.62	79.49	77.35	75.21
PPV	76.81	82.0	84.45	87.28	89.45	90.95	92.54	93.78	95.14

Figura 299. Umbrales de experimento con caffenet original

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	89.45	83.33	99.65	99.8	86.28	99.46

Figura 300. Métricas de experimento con caffenet original

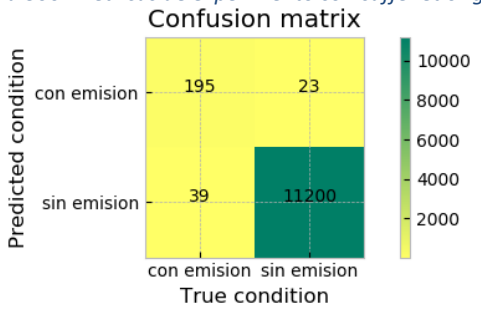

Figura 301. Matriz de confusión de experimento con caffenet original

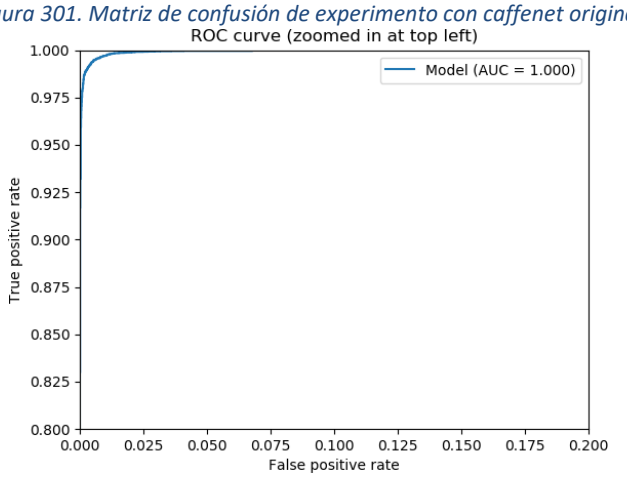

Figura 302. Curva ROC de experimento con caffenet original

	TP	FP	FN
0.5	12	8	9

Figura 303. Secuencias de emisiones para el día 20 de experimento con caffenet original

8.1.2 Experimentos de modificaciones en capas convolucionales

Tras experimentar con la red original caffenet se procederá (tal y como se realizó con las dos redes anteriores) a realizar pequeñas modificaciones en la capas convolucionales de la red.

8.1.2.1 Experimento con reducción de caffenet según Hohberg

Una vez realizado el experimento inicial con la arquitectura de la red caffenet original se procederá a experimentar con la red propuesta en [Hohberg 2015] reduciendo a ¼ el número de mapas de características generados por cada capa convolucional.

Los resultados del entrenamiento y testing se ven representados en la Figura 339, Figura 340, Figura 341, Figura 342, Figura 343, Figura 344 y Figura 345 en las que se puede observar un comportamiento similar al obtenido en el anterior experimento, a excepción de una

mejora notable en la clasificación de secuencias en las que disminuye en dos el número tanto de falsos positivos como de falsos negativos.

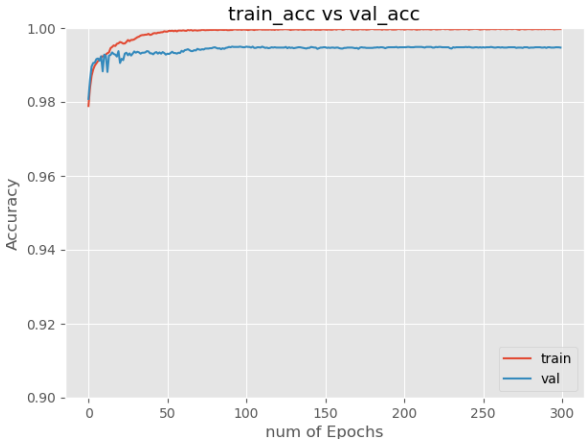

Figura 304. Accuracy de experimento con caffenet reducida

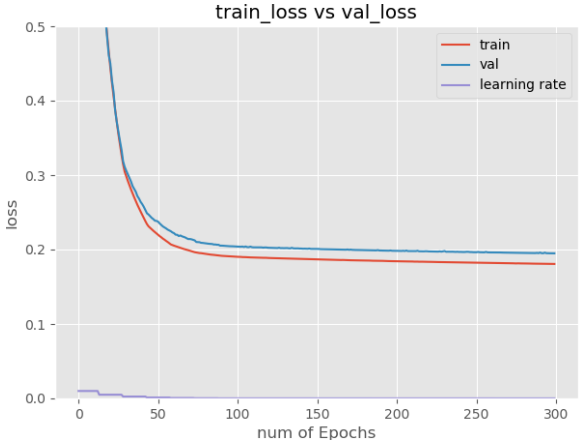

Figura 305. Loss de experimento con caffenet reducida

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	91.03	88.46	86.75	84.19	83.33	82.48	81.2	78.63	74.79
PPV	77.45	81.18	85.65	86.4	88.64	90.19	92.68	94.36	94.59

Figura 306. Umbrales de experimento con caffenet reducida

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	88.64	83.33	99.65	99.78	85.9	99.44

Figura 307. Métricas de experimento con caffenet reducida

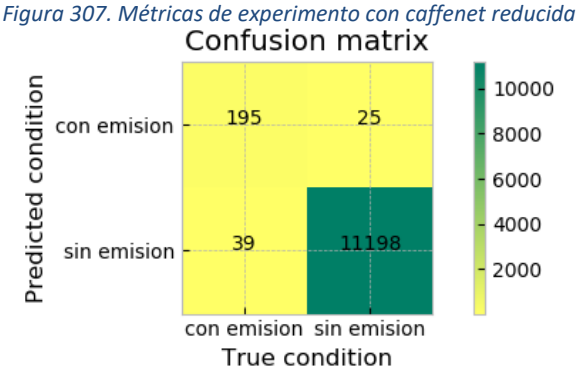

Figura 308. Matriz de confusión de experimento con caffenet reducida

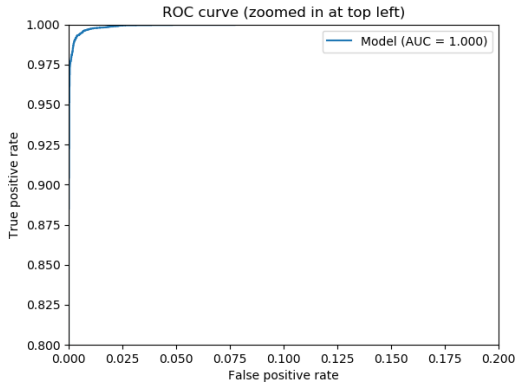

Figura 309. Curva ROC de experimento con caffenet reducida

	TP	FP	FN
0.5	14	6	7

Figura 310. Secuencias de emisiones para el día 20 de experimento con caffenet reducida

8.1.2.2 Experimento de eliminación de dos capas convolucionales

Debido a que el experimento consistente en la reducción de mapas de características a ¼ ha obtenido una notable optimización en la fase de predicción de secuencias, se partirá de esta arquitectura y se seguirán introduciendo simplificaciones a nivel de capas convolucionales, al igual que se hizo para las dos redes anteriores. Es por ello por lo que el presente experimento consistirá en la modificación de la red de modo que se eliminen la primera y la última capa convolucional del tercer nivel convolucional conformado por 3 capas, dejando así una única capa de 96 filtros y aumentando el número de características del vector de flattening.

A partir de la Figura 311, Figura 312, Figura 313, Figura 314, Figura 315, Figura 316 y Figura 317 se observa que el número de secuencias de emisiones falsas se decrementa en una unidad, sin embargo, esto supone un mayor decremento del número de verdaderos positivos descendiendo a 12 secuencias detectadas de 21.

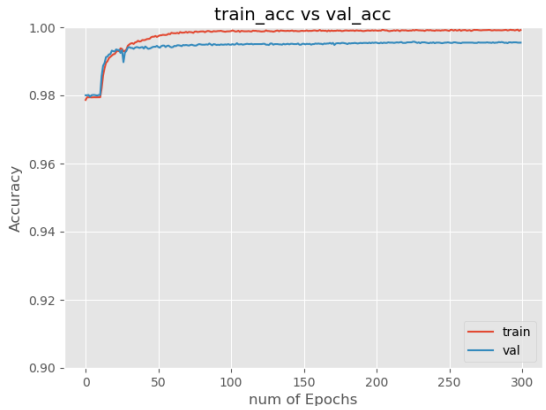

Figura 311. Accuracy de experimento de eliminación de 2 capas convolucionales en caffenet

Figura 312. Loss de experimento e eliminación de 2 capas convolucionales en caffenet

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	92.31	89.74	88.46	85.9	83.76	79.49	77.78	74.79	69.66
PPV	76.87	82.68	85.19	86.27	89.5	90.29	91.46	92.59	95.32

Figura 313. Umbrales de experimento e eliminación de 2 capas convolucionales en caffenet

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	89.5	83.76	99.66	99.8	86.53	99.47

Figura 314. Métricas de experimento e eliminación de 2 capas convolucionales en caffenet

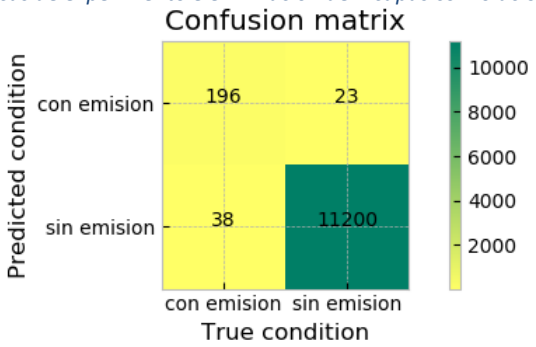

Figura 315. Matriz de confusión de experimento e eliminación de 2 capas convolucionales en caffenet

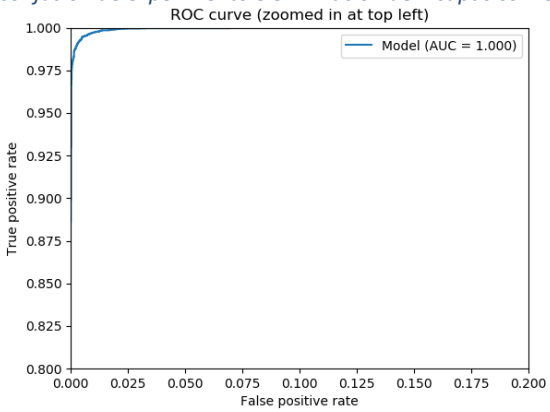

Figura 316. Curva ROC de experimento e eliminación de 2 capas convolucionales en caffenet

	TP	FP	FN
0.5	12	5	9

Figura 317. Secuencias de emisiones para el día 20 de experimento de eliminación de 2 capas convolucionales en caffenet

8.1.2.3 Experimento de eliminación de una capa convolucional

A partir de los resultados obtenidos al eliminar dos capas convolucionales del tercer nivel convolucional, se procederá a eliminar únicamente la primera capa convolucional de dicho nivel convolucional, manteniendo así el número de características del vector de flattening en cuanto al experimento 8.1.2.1.

La Figura 318, Figura 319, Figura 320, Figura 321, Figura 322, Figura 323 y Figura 324 muestra que la reducción de una capa convolucional manteniendo el número de características supone un decremento de detección de secuencias y un incremento de falsas alarmas.

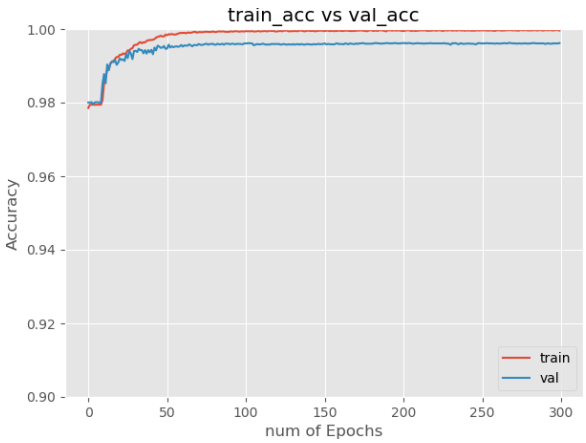

Figura 318. Accuracy de experimento de eliminación de una capa convolucional en caffenet

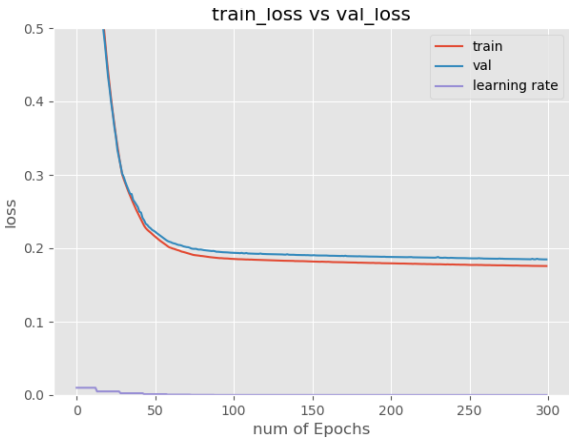

Figura 319. Loss de experimento de una capa convolucional en caffenet

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	93.16	90.17	88.46	85.9	83.76	79.91	78.63	75.64	72.22
PPV	78.14	83.73	85.89	87.77	89.09	89.47	91.54	93.65	95.48

Figura 320. Umbrales de experimento de una capa convolucional en caffenet

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	89.09	83.76	99.66	99.79	86.34	99.46

Figura 321. Métricas de experimento de una capa convolucional en caffenet

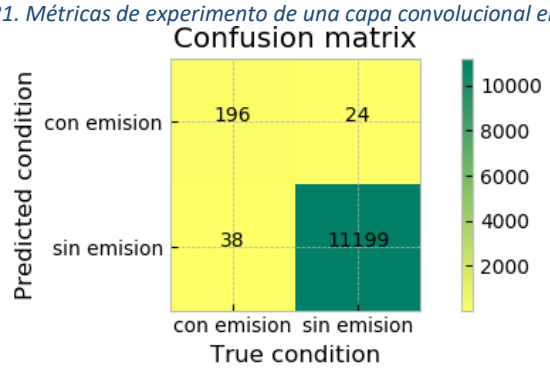

Figura 322. Matriz de confusión de experimento de una capa convolucional en caffenet

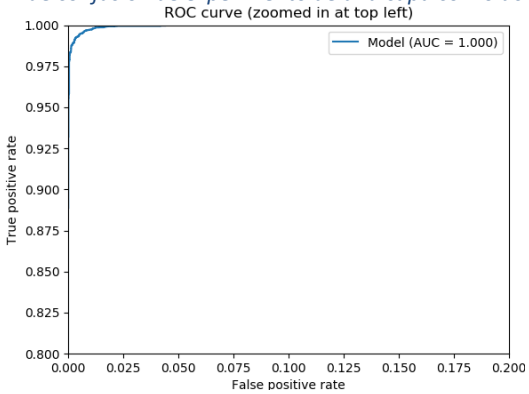

Figura 323. Curva ROC de experimento de una capa convolucional en caffenet

	TP	FP	FN
0.5	12	9	9

Figura 324. Secuencias de emisiones para el día 20 de experimento de una capa convolucional en caffenet

8.1.3 Experimentos de modificaciones en capas full connected

Debido a que las dos modificaciones de la arquitectura en base a cambios en capas convolucionales (a parte de la reducción propuesta en [Hohberg 2015]) no han resultado en una optimización de la red, se procederá a establecer modificaciones en las capas full connected. Siguiendo el procedimiento establecido en el apartado 7.2.2 consistente en la experimentación con diversas combinaciones de número de neuronas.

8.1.3.1 Decremento a 2048 neuronas en segunda capa full connected

Al igual que en los experimentos con la red [Yin 2017] se iniciará con el cambio menos significativo, consistente en la reducción de neuronas de la segunda capa full connected de 4096 a 2048.

A partir de la Figura 325, Figura 326, Figura 327, Figura 328, Figura 329, Figura 330 y Figura 331 se comprueba que a pesar de que a nivel de detección de imágenes individuales se decrementa el número de falsos positivos y falsos negativos, la predicción final de secuencia cuenta con un falso positivo más.

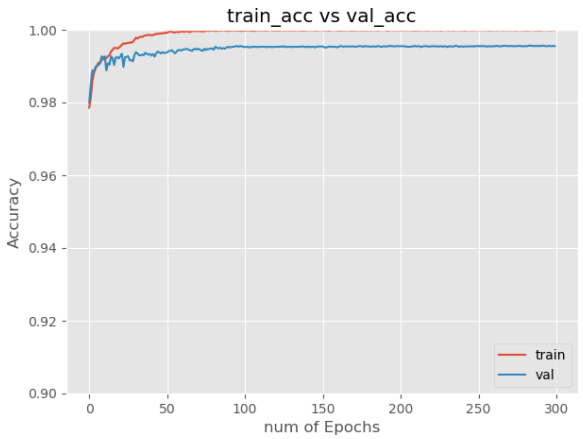

Figura 325. Accuracy de experimento de decremento a 2048 neuronas en segunda capa de caffenet

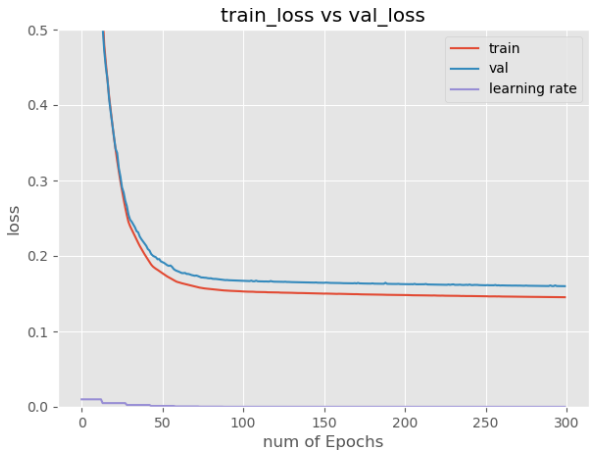

Figura 326. Loss de experimento de decremento a 2048 neuronas en segunda capa de caffenet

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	88.46	87.61	85.9	85.9	85.04	83.33	81.2	78.63	76.5
PPV	80.54	85.06	87.39	88.55	89.64	90.28	91.79	92.93	93.72

Figura 327. Umbrales de experimento de decremento a 2048 neuronas en segunda capa de caffenet

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	89.64	85.04	99.69	99.8	87.28	99.49

Figura 328. Métricas de experimento de decremento a 2048 neuronas en segunda capa de caffenet

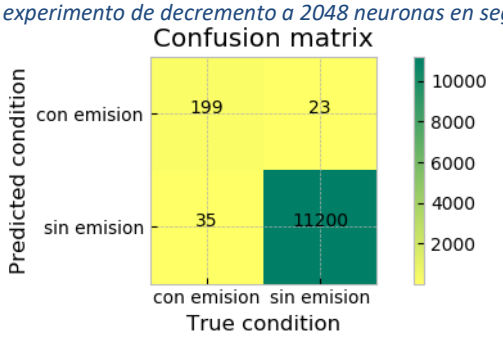

Figura 329. Matriz de confusión de experimento de decremento a 2048 neuronas en segunda capa de caffenet

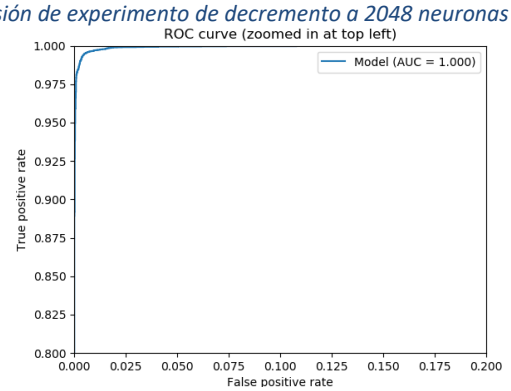

Figura 330. Curva ROC de experimento de decremento a 2048 neuronas en segunda capa de caffenet

	TP	FP	FN
0.5	13	6	8

Figura 331. Secuencias de emisiones para el día 20 de experimento de decremento a 2048 neuronas en segunda capa de caffenet

8.1.3.2 Decremento a 2048 neuronas en ambas capas full connected

A continuación, se procederá a experimentar con dos capas full connected de 2048 neuronas, a diferencia de las dos capas de 4096 neuronas de la red original.

A partir de la Figura 332, Figura 333, Figura 334, Figura 335, Figura 336, Figura 337 y Figura 338 se puede observar que los resultados de predicción de secuencias son exactamente iguales a los obtenidos para la misma red con dos capas full connected de 4096 neuronas, lo que significa que simplificando la parte full connected a la mitad se obtiene el mismo porcentaje de detección de secuencias de emisiones. Por otro lado, la clasificación de imágenes de forma independiente obtiene más verdaderos positivos y verdaderos negativos. Estas dos comparaciones, por tanto, permiten establecer que la simplificación del número de neuronas en ambas capas full connected a ½ permite optimizar la red.

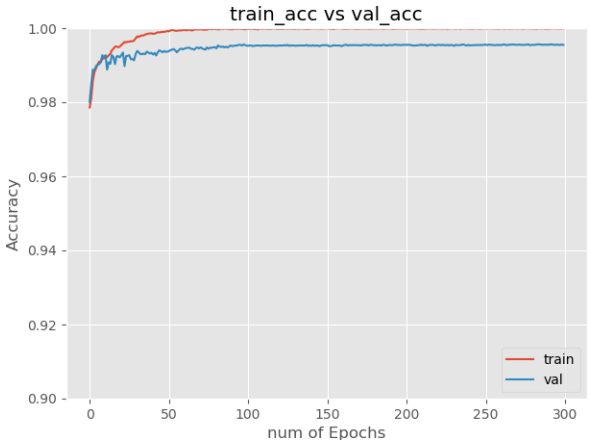

Figura 332. Accuracy de experimento de decremento a 2048 neuronas ambas capas de caffenet

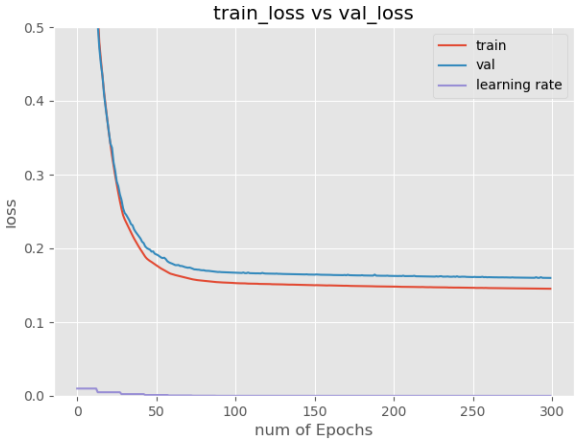

Figura 333. Loss de experimento de decremento a 2048 neuronas ambas capas de caffenet

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	89.74	88.89	87.61	86.32	86.32	84.19	81.62	78.63	74.36
PPV	81.08	84.55	85.06	86.7	89.78	91.2	92.27	93.4	94.57

Figura 334. Umbrales de experimento de decremento a 2048 neuronas ambas capas de caffenet

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	89.78	86.32	99.72	99.8	88.02	99.52

Figura 335. Métricas de experimento de decremento a 2048 neuronas ambas capas de caffenet

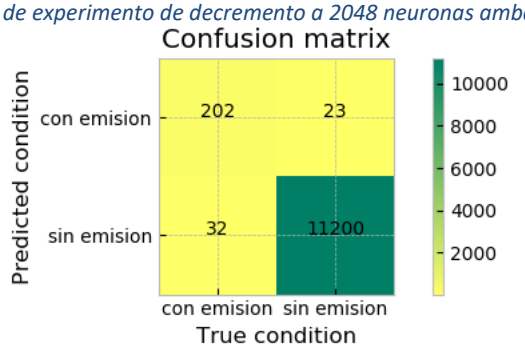

Figura 336. Matriz de confusión de experimento de decremento a 2048 neuronas ambas capas de caffenet

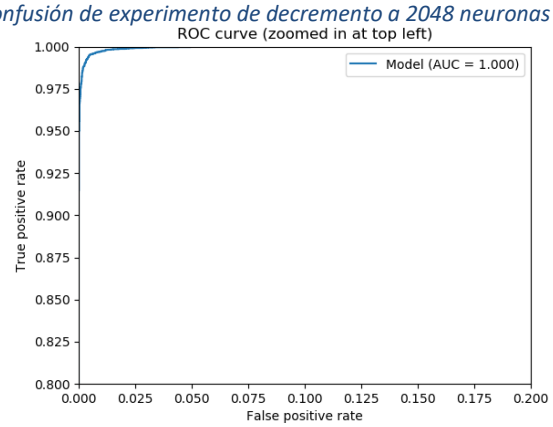

Figura 337. Curva ROC de experimento de decremento a 2048 neuronas ambas capas de caffenet

	TP	FP	FN
0.5	14	6	7

Figura 338. Secuencias de emisiones para el día 20 de experimento de decremento a 2048 neuronas ambas capas de caffenet

8.1.3.3 Decremento a 1024 neuronas en la segunda capa full connected

Debido a que el decremento de ambas capas a 2048 neuronas ha supuesto una optimización de la red, se comprobará la repercusión del decremento del número de neuronas a 1024 capa por capa. Por lo tanto, este experimento consistirá en entrenar la red con una segunda capa full connected de 1024 neuronas, manteniendo la primera capa en 2048.

En la Figura 339, Figura 340, Figura 341, Figura 342, Figura 343, Figura 344 y Figura 345 se muestran los resultados de entrenamiento y testing obtenidos, a partir de los cuales se puede observar que, por primera vez con esta red, se ha conseguido disminuir el número de falsas alarmas en la detección de secuencias de emisiones sin disminuir el número de verdaderos positivos.

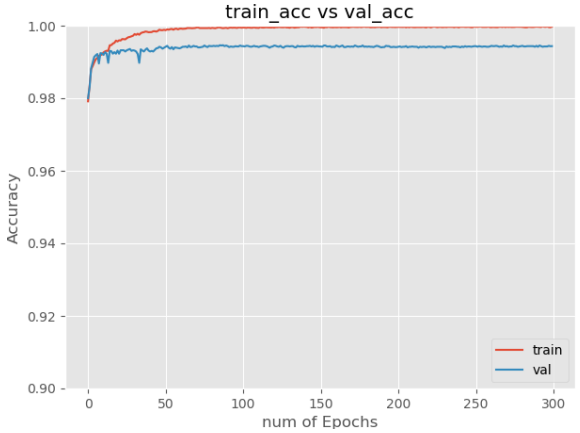

Figura 339. Accuracy de experimento de decremento a 1024 neuronas en segunda capa de caffenet

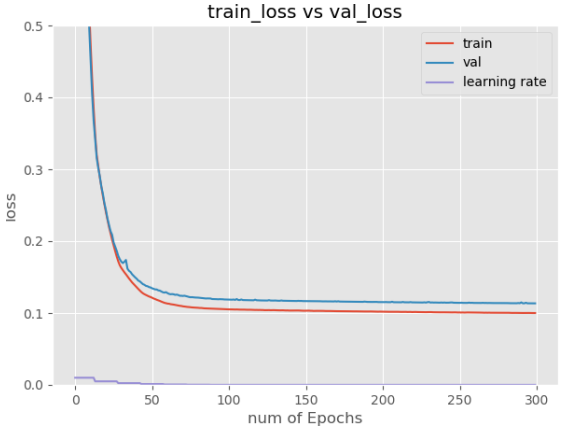

Figura 340. Loss de experimento de decremento a 1024 neuronas en segunda capa de caffenet

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	90.17	85.9	85.47	84.62	84.19	82.91	82.05	79.06	75.21
PPV	78.73	82.38	85.47	87.61	90.37	90.23	92.31	94.39	95.65

Figura 341. Umbrales de experimento decremento a 1024 neuronas en segunda capa de caffenet

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	90.37	84.19	99.67	99.81	87.17	99.49

Figura 342. Métricas de experimento decremento a 1024 neuronas en segunda capa de caffenet

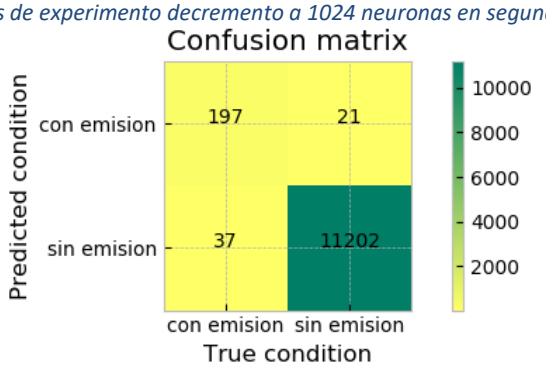

Figura 343. Matriz de confusión de experimento decremento a 1024 neuronas en segunda capa de caffenet

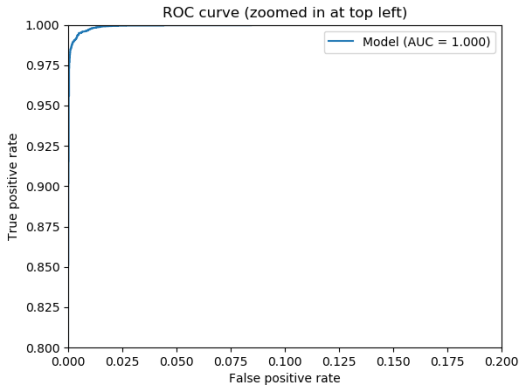

Figura 344. Curva ROC de experimento decremento a 1024 neuronas en segunda capa de caffenet

	TP	FP	FN
0.5	14	4	7

Figura 345. Secuencias de emisiones para el día 20 de experimento decremento a 1024 neuronas en segunda capa de caffenet

8.1.3.4 Decremento a 1024 neuronas en ambas capas full connected

Finalmente se realizará un experimento con la reducción de ambas capas full connected de 4096 a 1024.

Esta vez la Figura 346, Figura 347, Figura 348, Figura 349, Figura 350, Figura 351 y Figura 352 muestran que el aplicar una reducción de neuronas a 1024 en ambas capas obtiene una optimización dentro de la clasificación de secuencias, detectando 2 emisiones más. Sin embargo, el número de falsas alarmas asciende.

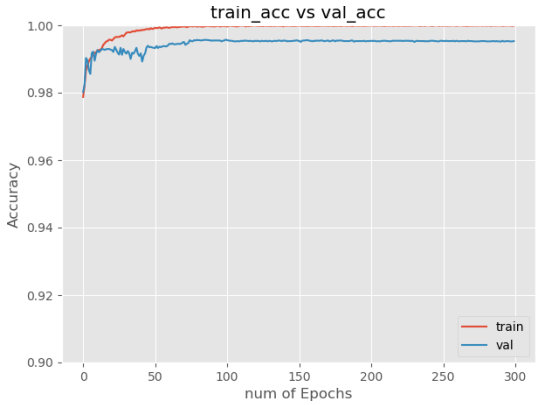

Figura 346. Accuracy de experimento de decremento a 1024 neuronas ambas capas de caffenet

Figura 347. Loss de experimento de decremento a 2048 neuronas ambas capas de caffenet

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	89.74	87.61	87.61	87.18	84.19	82.05	80.34	79.06	74.79
PPV	80.77	84.02	85.77	87.55	89.55	90.57	92.61	92.96	94.09

Figura 348. Umbrales de experimento de decremento a 2048 neuronas ambas capas de caffenet

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	89.55	84.19	99.67	99.8	86.78	99.48

Figura 349. Métricas de experimento de decremento a 2048 neuronas ambas capas de caffenet

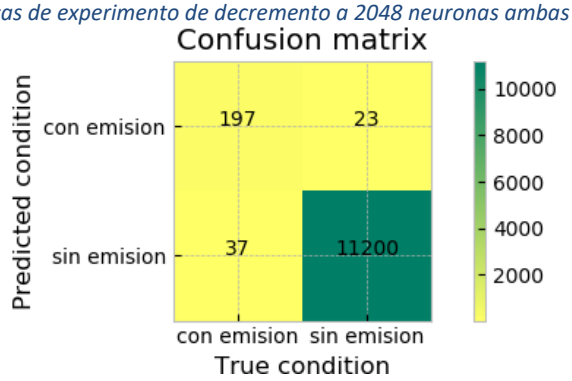

Figura 350. Matriz de confusión de experimento de decremento a 2048 neuronas ambas capas de caffenet

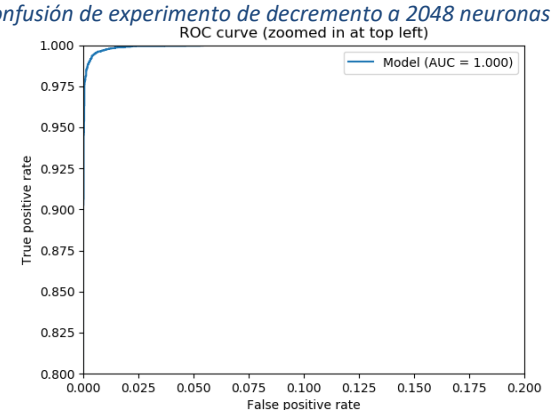

Figura 351. Curva ROC de experimento de decremento a 2048 neuronas ambas capas de caffenet

	TP	FP	FN
0.5	16	7	5

Figura 352. Secuencias de emisiones para el día 20 de experimento de decremento a 2048 neuronas ambas capas de caffenet

8.1.4 Experimentos de modificaciones de hiperparámetros de entrenamiento

En base a los resultados obtenidos en los experimentos realizados hasta el momento, se partirá de la arquitectura de red testeada en la sección 8.1.3.3 consistente en la reducción de la primera capa full connected a 2048 neuronas y la segunda a 1024. Haciendo uso de esta arquitectura se probará a optimizar la red en base a distintas métricas durante el entrenamiento.

8.1.4.1 Optimización de métrica precisión

El primer experimento de esta sección consistirá en la optimización de la métrica precisión. En la Figura 353 y Figura 354 se observa que las curvas de entrenamiento y validación de precisión y loss siguen un patrón muy similar al que sigue una red entrenada mediante la optimización de accuracy y loss. Por otro lado, la Figura 355, Figura 356, Figura 357, Figura

358 y Figura 359 muestran como la optimización de la métrica precisión ha supuesto un aumento de falsos positivos tanto a nivel de imagen como de secuencia. En cuanto a la clasificación de imágenes de forma individual también se puede observar un aumento de falsos negativos, observando por tanto una reducción notable tanto en precisión como en recall.

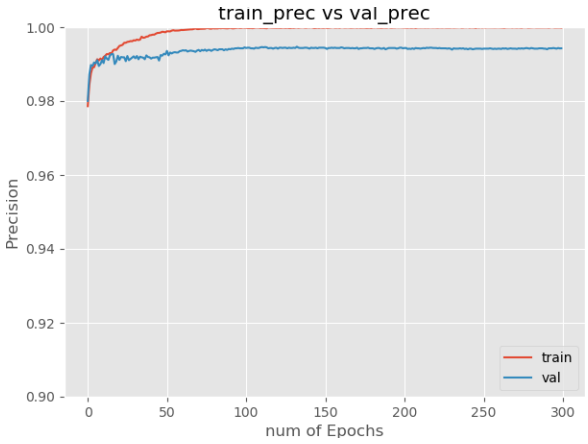

Figura 353. Precision de optimización de precision en caffenet

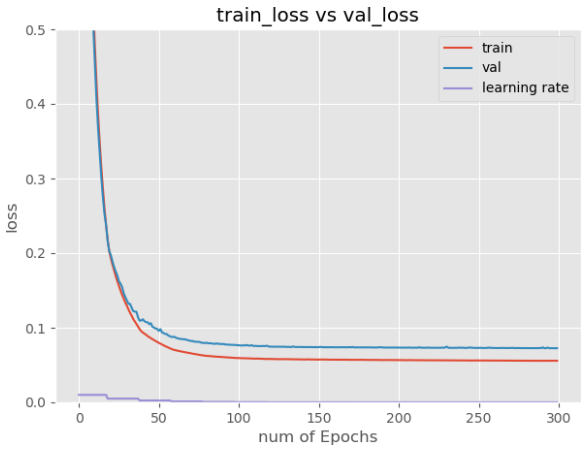

Figura 354. Loss de optimización de precision en caffenet

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	90.17	88.89	85.47	83.76	82.48	82.05	80.34	78.21	77.35
PPV	76.73	81.89	84.03	85.59	87.33	89.3	89.52	91.5	95.26

Figura 355. Umbrales de optimización de precision en caffenet

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	87.33	82.48	99.64	99.75	84.84	99.4

Figura 356. Métricas de optimización de precision en caffenet

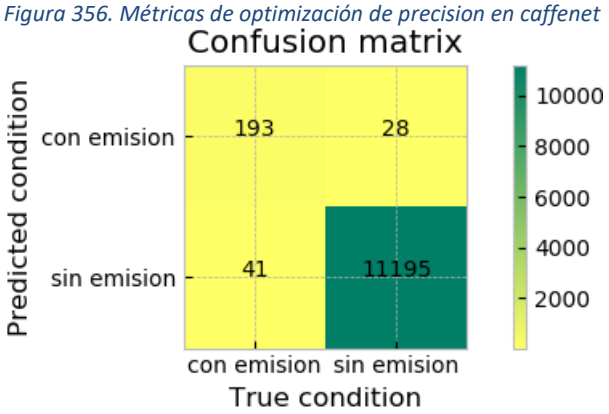

Figura 357. Matriz de confusión de optimización de precision en caffenet

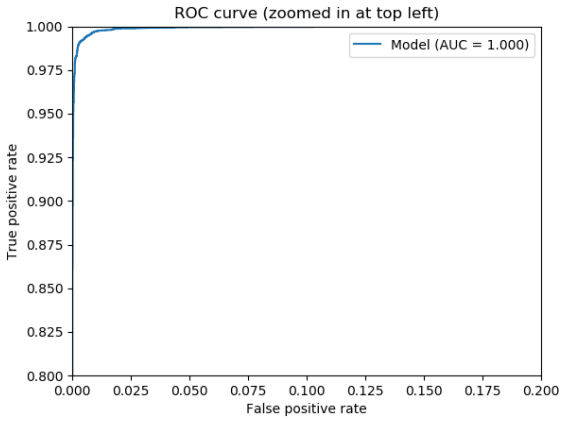

Figura 358. Curva ROC de optimización de precision en caffenet

	TP	FP	FN
0.5	14	9	7

Figura 359. Secuencias de emisiones para el día 20 de optimización de precision en caffenet

8.1.4.2 Optimización de métrica recall

Siguiendo la metodología del experimento anterior, se continuará experimentando con la optimización de la métrica recall durante la fase de entrenamiento de la red.

A partir de la Figura 360, Figura 361, Figura 362, Figura 363, Figura 364, Figura 365 y Figura 366 se comprueba que, a pesar de que durante el entrenamiento las métricas obtienen unos resultados similares a los de la optimización de accuracy, al igual que al optimizar la métrica precisión en la clasificación de imágenes de forma individual se obtiene un mayor número de falsos positivos y falsos negativos y en la clasificación de secuencias, a pesar de que el número de verdaderos positivos se mantiene las falsas alarmas aumentan, aunque en menor medida que para la optimización de precisión.

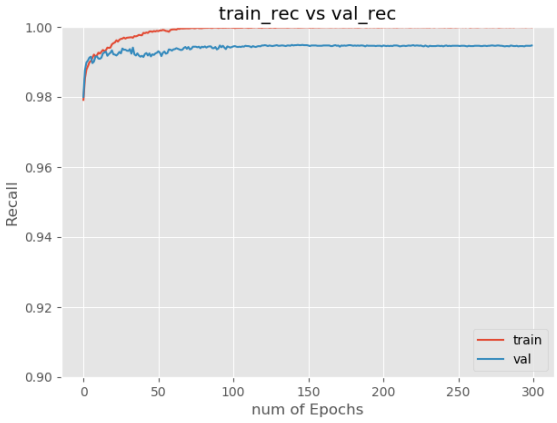

Figura 360. Recall de optimización de recall en caffenet

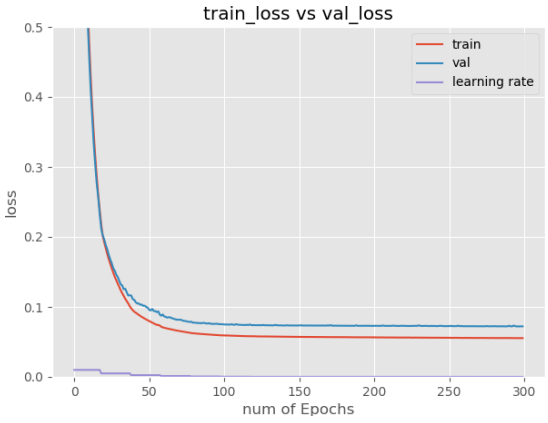

Figura 361. Loss de optimización de recall en caffenet

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	91.45	89.74	85.9	83.76	83.33	82.05	81.62	80.77	77.78
PPV	75.89	80.15	81.71	84.12	85.53	87.27	88.43	90.43	93.81

Figura 362. Umbrales de optimización de recall en caffenet

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	85.53	83.33	99.65	99.71	84.42	99.37

Figura 363. Métricas de optimización de recall en caffenet

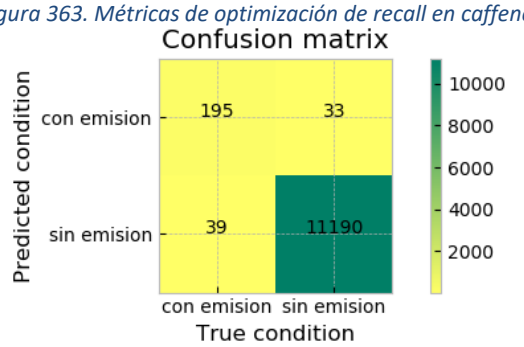

Figura 364. Matriz de confusión de optimización de recall en caffenet

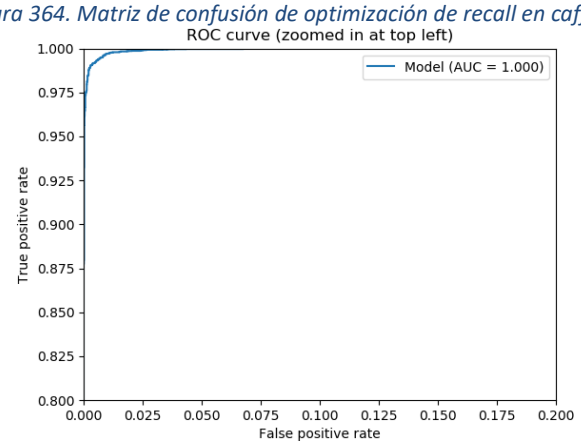

Figura 365. Curva ROC de optimización de recall en caffenet

	TP	FP	FN
0.5	14	7	7

Figura 366. Secuencias de emisiones para el día 20 de optimización de recall en caffenet

8.1.4.3 Penalización de loss

Finalmente se ha experimentado con la aplicación de una penalización a la métrica loss equivalente al desbalanceo del conjunto de datos. En este caso a partir de la Figura 367 se puede observar como las curvas de entrenamiento y validación sufren algunas variaciones mostrando algunos picos hasta que se estabilizan a partir de la epoch 400 aproximadamente. Asimismo, en la Figura 368 no se pueden observar valores ya que la penalización de loss hace que estos valores ronden aproximadamente el valor 0,6, quedando fuera de los límites establecidos como

En cuanto a los resultados de testing representados en la Figura 369, Figura 370, Figura 371, Figura 372 y Figura 373 esta vez la clasificación de secuencias empeora tanto en número de detecciones como en número de falsas alarmas. Sin embargo, para la clasificación de imágenes individuales, a pesar de que el ratio de falsos positivos aumenta (aunque menos

que en los dos experimentos anteriores), el número de falsos negativos ha decrecido en comparación con los resultados obtenidos en la sección 8.1.3.3.

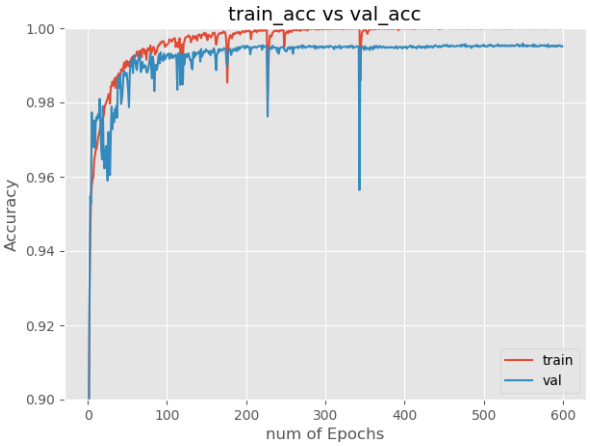

Figura 367. Accuracy con penalización de loss en caffenet

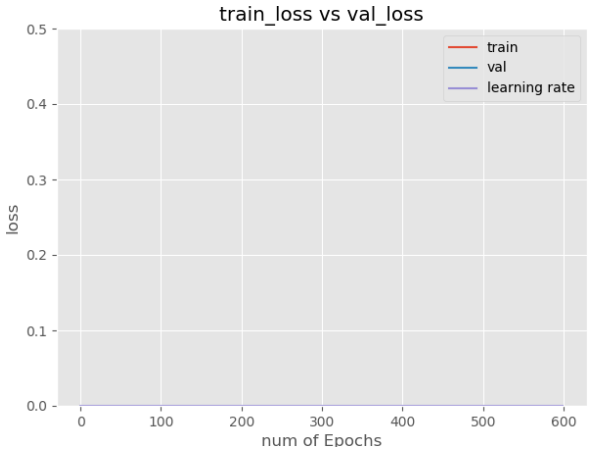

Figura 368. Loss con penalización en caffenet

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	92.31	90.17	88.03	87.61	86.75	85.04	84.19	83.33	81.2
PPV	80.3	83.07	84.08	85.77	87.5	87.67	88.74	89.04	90.91

Figura 369. Umbrales de penalización de loss en caffenet

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	87.5	86.75	99.72	99.74	87.12	99.48

Figura 370. Métricas de penalización de loss en caffenet

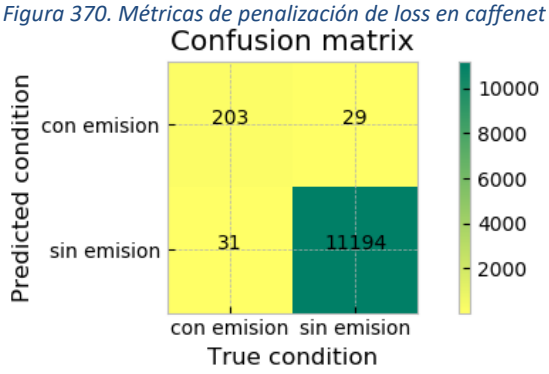

Figura 371. Matriz de confusión de penalización de loss en caffenet

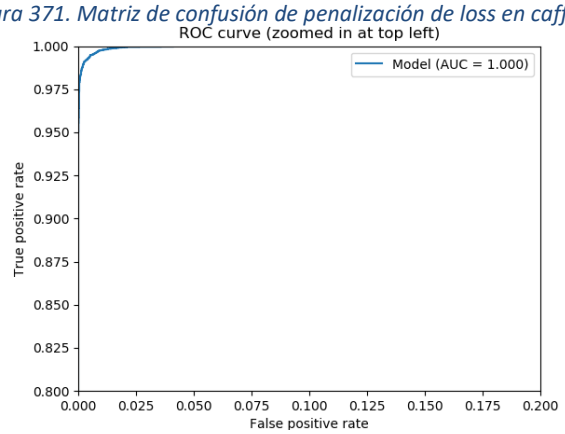

Figura 372. Curva ROC de penalización de loss en caffenet

	TP	FP	FN
0.5	13	7	8

Figura 373. Secuencias de emisiones para el día 20 de penalización de loss en caffenet

8.2 Selección de la red caffenet óptima

Tras llevar a cabo los experimentos relativos a las diversas modificaciones de la red caffenet se elegirá la red óptima resultante de esta fase de experimentación al igual que se ha realizado con las dos redes anteriores.

Esta decisión se tomará en base a las comparaciones realizadas en cada experimento en la sección 8.1, las detecciones de secuencias de emisiones de la Tabla 18 y las comparaciones de métricas recall y precisión para la clasificación de imágenes de forma individual de la Figura 374, Figura 375, Figura 376, Figura 377, Figura 378, Figura 379, Figura 380 y Figura 381.

Los resultados mostrados en las gráficas se corresponden a los obtenidos mediante el conjunto de testing resultante del método de división EEEEVT y del conjunto resultante de utilizar el día 20 completo.

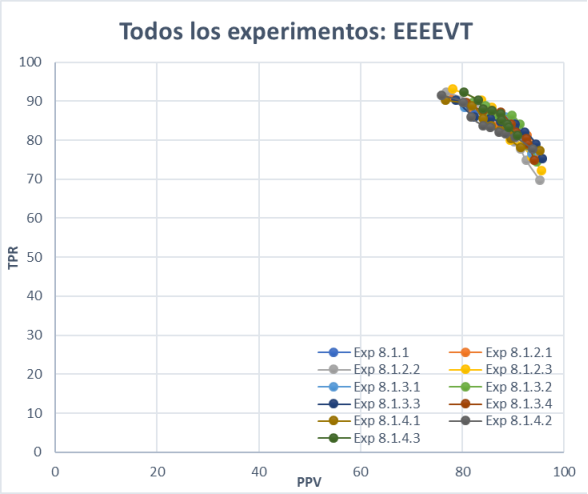

Figura 374. Todos los experimentos con caffenet con el conjunto de testing EEEEVT

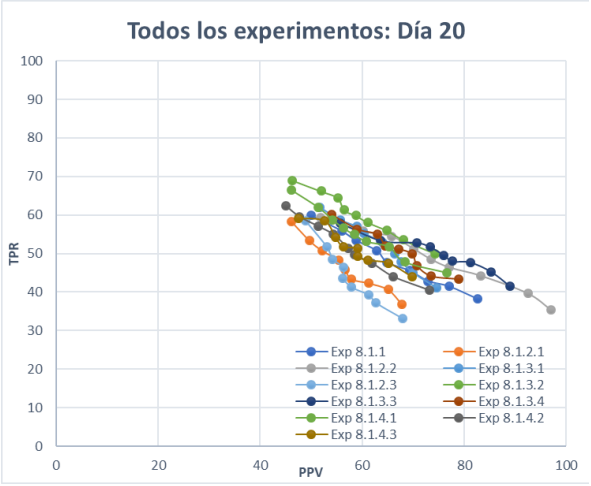

Figura 375. Todos los experimentos con caffenet con el día 20

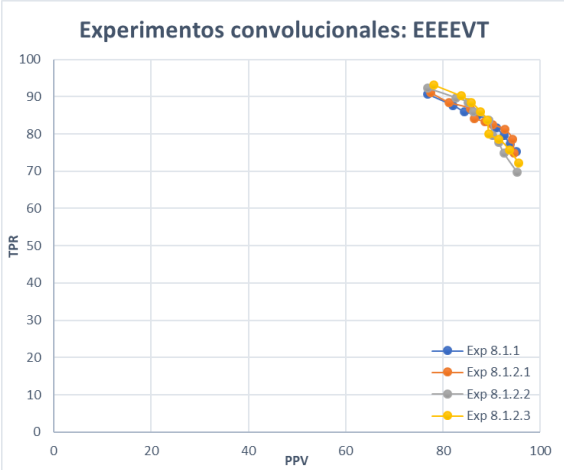

Figura 376. Experimentos de capas convolucionales en caffenet con el conjunto de testing EEEEVT

Figura 377. Experimentos de capas convolucionales en caffenet con el día 20

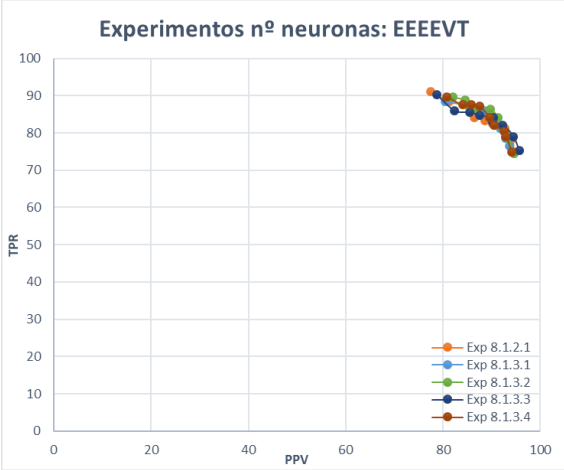

Figura 378. Experimentos de nº de neuronas en caffenet con el conjunto de testing EEEEVT

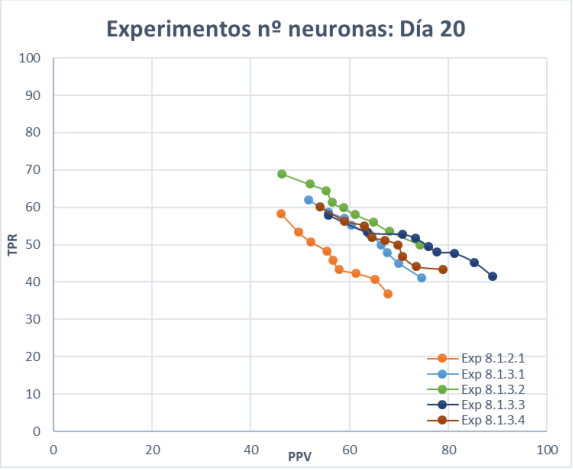

Figura 379. Experimentos de nº de neuronas caffenet con el día 20

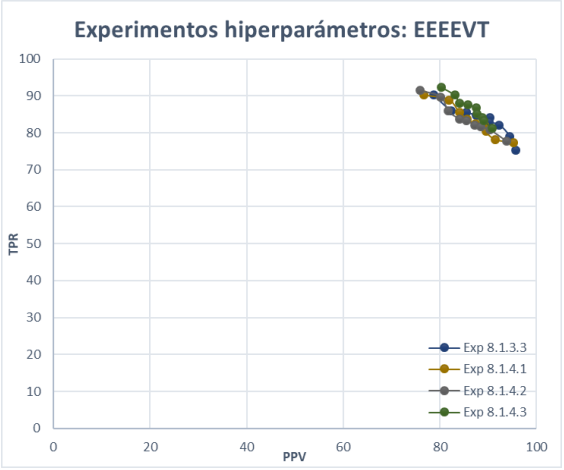

Figura 380. Experimentos hiperparámetros en caffenet con el conjunto de testing EEEEVT

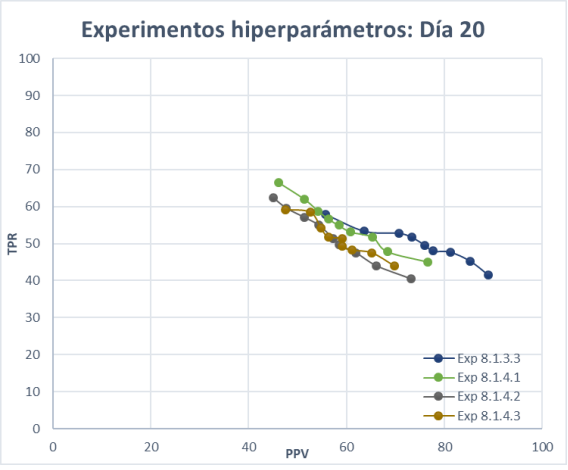

Figura 381. Experimentos de hiperparámetros en caffenet con el día 20

	TP	FP	FN	TPR	PPV	Threshold
Exp 8.1.1	12	8	8	60,00%	60,00%	0,5
Exp 8.1.2.1	14	6	7	66,67%	70,00%	0,5
Exp 8.1.2.2	12	5	9	57,14%	70,59%	0,5
Exp 8.1.2.3	12	9	9	57,14%	57,14%	0,5
Exp 8.1.3.1	13	6	8	61,90%	68,42%	0,5
Exp 8.1.3.2	14	6	7	66,67%	70,00%	0,5
Exp 8.1.3.3	14	4	7	66,67%	77,78%	0,5
Exp 8.1.3.4	16	7	5	76,19%	69,57%	0,5
Exp 8.1.4.1	14	9	7	66,67%	60,87%	0,5
Exp 8.1.4.2	14	7	7	66,67%	66,67%	0,5
Exp 8.1.4.3	13	7	8	61,90%	65,00%	0,5

Tabla 18. Comparación de resultados de predicción de secuencias de la red caffenet

Debido a que durante el desarrollo de la sección 8.1 se ha debido realizar un continuo contraste de cada uno de los experimentos realizados con objeto de partir de la configuración óptima en experimentos futuros, se puede determinar que **la red caffenet óptima se trata de la resultante del experimento 8.1.3.3**.

Esto se debe a que en la Tabla 18 se puede observar que este experimento cuenta con el valor más alto de precisión (PPV). Asimismo, cuenta también con el segundo valor de recall (TPR) más alto de la tabla.

En cuanto a la clasificación de imágenes de forma individual, al igual que en la sección 7.3 con la red [Yin 2017], los resultados del conjunto EEEEVT son muy similares y los del día 20 difieren, por lo que basándose en las comparaciones de resultados para la predicción de imágenes en el día 20 se observa que el experimento 8.1.3.3 obtiene los mejores resultados de precisión sin empeorar notablemente la métrica recall.

8.3 Cross testing

En concordancia con el procedimiento seguido hasta ahora se procederá a realizar los experimentos de cross testing con la red óptima seleccionada en la sección anterior, en este caso ha sido la resultante del apartado 8.1.3.3. En el Anexo VIII se encuentran los detalles de los resultados obtenidos para los experimentos con el conjunto EEVTEE y el conjunto VTEEEE.

A continuación, la Figura 382 sirve como comparativa de los resultados obtenidos del testing de una red entrenada bajo la misma configuración de la red caffenet y distintos conjuntos de datos de entrenamiento, validación y testing.

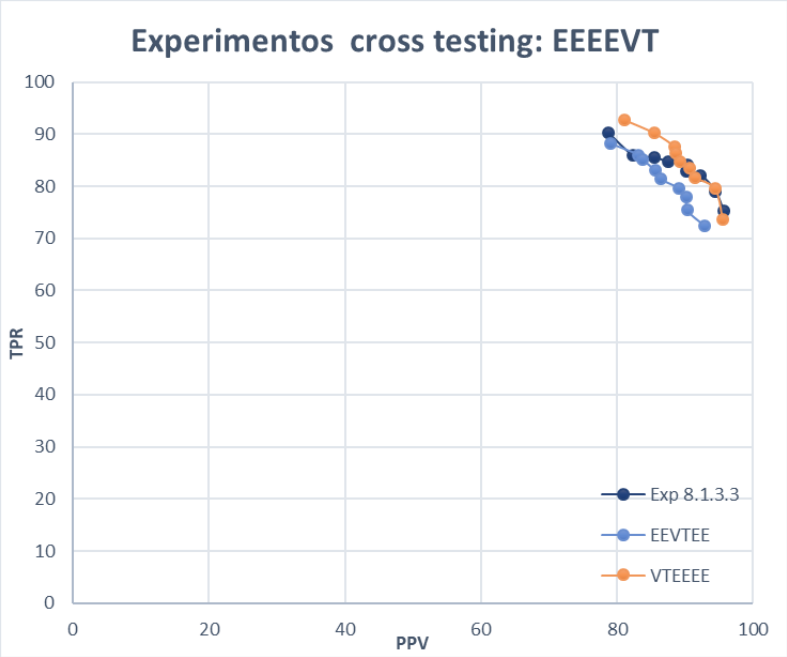

Figura 382. Cross testing de caffenet con testing intradías

Este gráfico muestra como el uso de distintos conjuntos de entrenamiento, validación o testing afecta ligeramente pero no de forma drástica ya que las curvas siguen un patrón muy similar llegando incluso a solaparse entre sí. Se comprueba, por tanto, que los resultados de obtenidos de la red **son consistentes entre sí**.

9 Selección de la red más apropiada

Una vez realizados los experimentos con las tres redes ([Tao, Zhang, and Wang 2016], [Yin 2017] y caffenet) se da por concluida la fase de experimentación con la comparativa de los principales resultados obtenidos por las configuraciones de red seleccionadas como óptimas para cada una de estas tres arquitecturas de red distintas. La Figura 383 y Figura 384 muestra la comparativa de los resultados de las métricas precisión (PPV) y recall (TPR) para cada una de las tres redes haciendo uso tanto del conjunto de testing resultante del conjunto intradías EEEEVT como del conjunto de testing consistente en el día 20. Estas gráficas permiten observar el rendimiento de cada red en la clasificación de imágenes de forma individual.

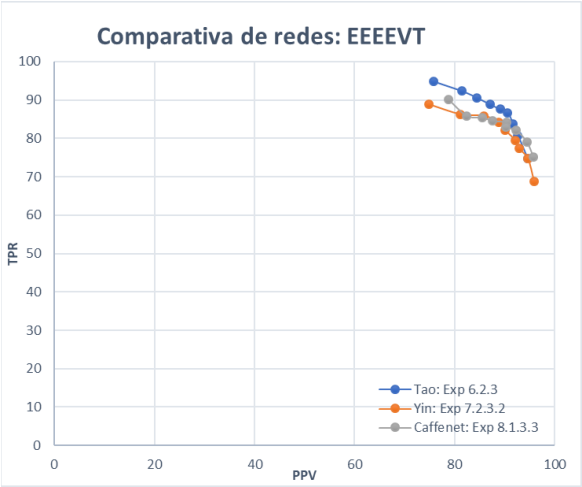

Figura 383. Comparativa de redes con el conjunto de testing EEEEVT

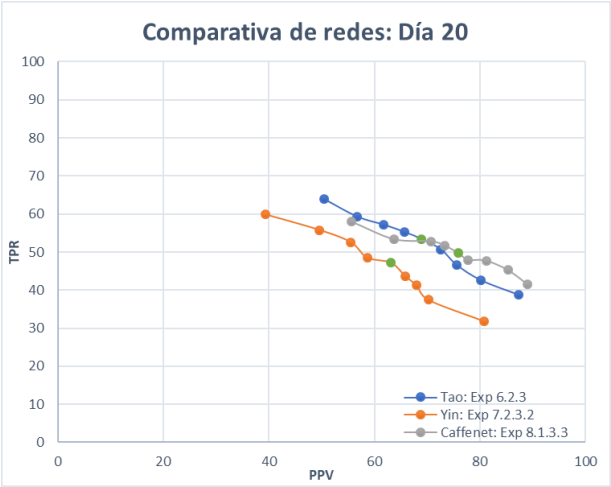

Figura 384. Comparativa de redes con el día 20

La Tabla 19 muestra los resultados del algoritmo de detección de secuencias de emisiones que se aplica tras la clasificación de las imágenes por parte de la red.

	TP	FP	FN	TPR	PPV	Threshold
Tao: Exp 6.2.3	17	4	4	80,95%	80,95%	0,5
Yin: Exp 7.2.3.2	14	2	7	66,67%	87,50%	0,5
Caffenet: Exp 8.1.3.3	14	4	7	66,67%	77,78%	0,5

Tabla 19. Predicción de secuencias de emisiones en cada red

A partir de esta comparativa de resultados se puede observar cómo, tras aplicar el algoritmo de predicción de secuencias de emisiones en base a las predicciones de imágenes individuales de cada red, la red de Yin obtiene un menor número de falsos positivos y mayor precisión, mientras que sin aplicarle el algoritmo de predicción de secuencias en la Figura 384 se puede observar como la curva de esta red se encuentra claramente por debajo y

hacia la izquierda, lo que indica valores inferiores de precisión y recall. En la Figura 383 se pueden observar también como los valores de recall son levemente inferiores.

Las otras dos redes (Tao y caffenet) muestran comportamientos muy similares en la Figura 383 y Figura 384 llegando a solaparse ambas curvas. Sin embargo, en ambas figuras se puede observar como la red Tao se encuentra desplazada ligeramente hacia la izquierda y arriba, lo que indica que a nivel de clasificación de imágenes individuales esta red posiblemente aportará un valor de recall mayor y menor de precisión. No obstante, esta diferencia no es muy grande.

Finalmente, la Tabla 19 muestra como la red de Tao y caffenet obtienen la misma cantidad de falsos positivos y la red de Tao detecta 17 de 21 emisiones, 3 más que las otras dos redes. Esto hace que la red de Tao obtenga mayor precisión que la red caffenet y el mayor valor de recall de las tres redes comparadas.

En base a estos razonamientos se considera que la **red de Tao se trata de la red óptima** resultante del proceso de experimentación llevado a cabo en el presente proyecto.

10 Planificación

El presente proyecto ha seguido una cronología similar al orden seguido a la hora de documentar la memoria. Esto se debe en cierto modo a que la memoria se ha ido documentando y revisando de forma iterativa durante la realización del proyecto.

En la Figura 385 y Figura 386 se muestra un diagrama Gantt con las principales tareas que conforman la realización del proyecto.

Estas tareas comienzan con una serie de estudios teóricos que se han debido de realizar de forma previa tanto a nivel de tecnologías como de redes neuronales convolucionales.

Una vez los estudios se han realizado y documentado se procederá a aplicar el conocimiento adquirido en CNNs y en las tecnologías. Para ello se realiza un sistema base capaz de cargar un conjunto de imágenes propias y entrenar y testear una red en base a dicho conjunto.

Tras implementar dicho sistema base se ha observado que es necesario un conjunto de datos más amplio, lo que da comienzo a una amplia fase de etiquetado y de documentación del conjunto recopilado. A su vez, se da comienzo a un estudio de las diversas técnicas de visión existentes y aplicables a sistemas de detección y clasificación de humos. Todo el conocimiento adquirido de ambas tareas será debidamente documentado en la memoria de forma paralela.

Tras contar con un conjunto de datos listo para la experimentación y un estudio de técnicas aplicables a la detección de humos se procede a convertir el sistema base en un sistema de detección de humos.

Finalmente se da paso a las tres grandes fases de experimentación (una para cada red) cuyas tareas constan en la realización y documentación de cada uno de los apartados de esta memoria. La primera fase de experimentación (la referente a la red [Tao, Zhang, and Wang 2016]) incluye los experimentos realizados con los conjuntos interdías e intradías y con la segunda fase de etiquetado del conjunto de imágenes, tomando las emisiones dudosas como ejemplos sin emisión.

Cabe destacar que cada tarea de documentación está sujeta a una revisión y corrección posterior de forma iterativa hasta que se solventen cada uno de los errores.

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin
1		Estudio redes neuronales	5 días	lun 16/07/1	vie 20/07/1
2		Estudio tecnologías	5 días	lun 16/07/1	vie 20/07/1
3		**Implementar sistema basico tensorflow**	**10 días**	**lun 23/07/1**	**vie 03/08/1**
4		Cargar un conjunto de datos propio	5 días	lun 23/07/1	vie 27/07/1
5		Entrenar red con conjunto	3 días	lun 30/07/1	mié 01/08/
6		Testing red entrenada	2 días	jue 02/08/1	vie 03/08/1
7		Estudio sistemas existentes detección de er	10 días	lun 06/08/1	vie 17/08/1
8		Documentar estudios realizados	15 días	lun 20/08/1	vie 07/09/1
9		Etiquetado de imágenes: Dudosas son emis	25 días	lun 06/08/1	vie 07/09/1
10		Documentar conjunto de datos	5 días	lun 10/09/1	vie 14/09/1
11		**Implementar sistema de detección de emis**	**10 días**	**lun 17/09/1**	**vie 28/09/1**
12		Implementar redes del estudio teórico	2 días	lun 17/09/18	mar 18/09/18
13		Implementar salida de los resultados de testing	3 días	mié 19/09/18	vie 21/09/18
14		Implementar clasificación de secuencias de imágenes	5 días	lun 24/09/18	vie 28/09/18
15		Documentar implementación	10 días	lun 17/09/18	vie 28/09/18
16		**Experimentos red Tao**	**55 días**	**lun 01/10/1**	**vie 14/12/1**
17		**Experimentos interdías**	**19 días**	**lun 01/10/1**	**jue 25/10/1**
18		Experimentación de conjuntos de datos	4 días	lun 01/10/18	jue 04/10/18
19		Experimentación de modificaciones en la arquitectura	5 días	vie 05/10/18	jue 11/10/18
20		Experimentación con hiperparámetros	5 días	vie 12/10/18	jue 18/10/18
21		Selección de red óptima	2 días	vie 19/10/1	lun 22/10/1
22		Cross testing	3 días	mar 23/10/	jue 25/10/1
23		Documentar la fase de experimentación interdías	19 días	lun 01/10/18	jue 25/10/18
24		Etiquetado de imágenes: Dudosas No emisión	25 días	vie 26/10/18	jue 29/11/18
25		Documentar conjunto de datos	2 días	vie 30/11/18	lun 03/12/18
26		**Experimentos intradías**	**9 días**	**mar 04/12/**	**vie 14/12/1**
27		Experimentación con hiperparámetros	4 días	mar 04/12/18	vie 07/12/18
28		Selección de red óptima	2 días	lun 10/12/1	mar 11/12/
29		Cross testing	3 días	mié 12/12/1	vie 14/12/1
30		Testing con día 20	3 días	mié 12/12/1	vie 14/12/1
31		Documentar la fase de experimentación intradías	9 días	mar 04/12/18	vie 14/12/18

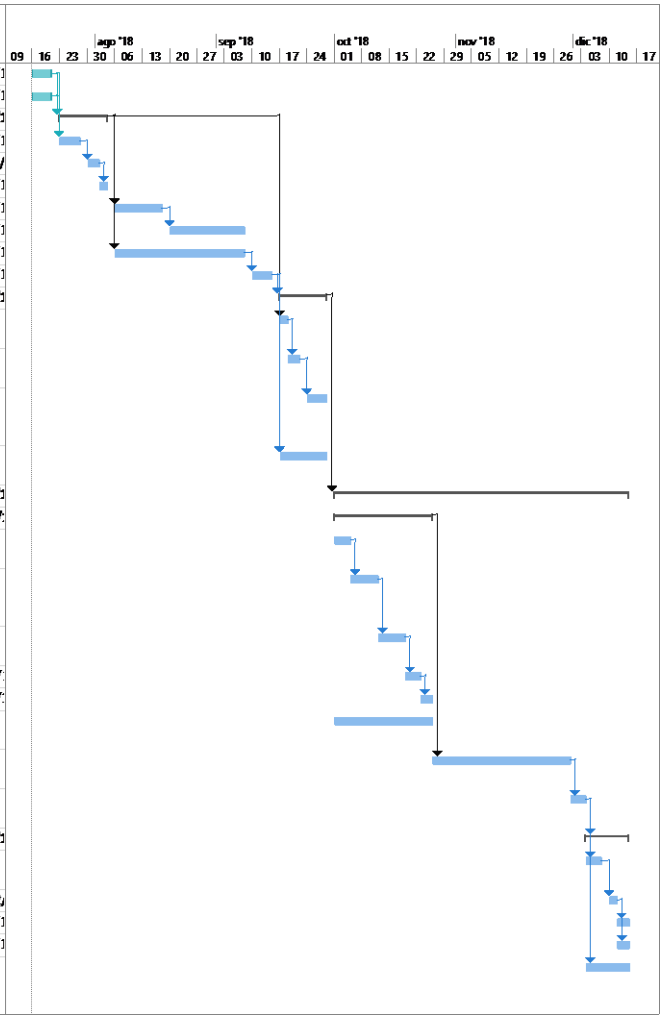

Figura 385. 1ª parte de la planificación del proyecto

Id	Modo de tare	Nombre de tarea	Duració	Comienzo	Fin
32		**Experimentos red Yin**	**16 días**	**lun 17/12/**	**lun 07/01/**
33		Estudio modificación arquitectura inicial	1 día	lun 17/12/18	lun 17/12/18
34		Experimentación modificaciones de capas convolucionales	3 días	mar 18/12/18	jue 20/12/18
35		Experimentación modificaciones de capas full connected	4 días	vie 21/12/18	mié 26/12/18
36		Experimentación de hiperparámetros de entrenamiento	3 días	jue 27/12/18	lun 31/12/18
37		Selección de la red óptima	2 días	mar 01/01,	mié 02/01/
38		Cross testing	3 días	jue 03/01/	lun 07/01/
39		Documentar experimentación con red Yin	16 días	lun 17/12/18	lun 07/01/19
40		**Experimentos red Caffenet**	**17 días**	**mar 08/01,**	**mié 30/01/**
41		Experimento inicial	1 día	mar 08/01,	mar 08/01/
42		Experimentación modificaciones de capas convolucionales	4 días	mié 09/01/19	lun 14/01/19
43		Experimentación modificaciones de capas full connected	4 días	mar 15/01/19	vie 18/01/19
44		Experimentación de hiperparámetros de entrenamiento	3 días	lun 21/01/19	mié 23/01/19
45		Selección de la red óptima	2 días	jue 24/01/	vie 25/01/1
46		Cross testing	3 días	lun 28/01/	mié 30/01/
47		Documentar experimentación con red Yi	16 días	mar 08/01,	mar 29/01/
48		**Finalizar la memoria del proyecto**	**4 días**	**jue 31/01/**	**mar 05/02/**
49		Documentar planificación y presupuesto	3 días	jue 31/01/19	lun 04/02/19
50		Documentar comparación de experimentos	3 días	jue 31/01/19	lun 04/02/19
51		Documentar conlusiones	1 día	mar 05/02,	mar 05/02/

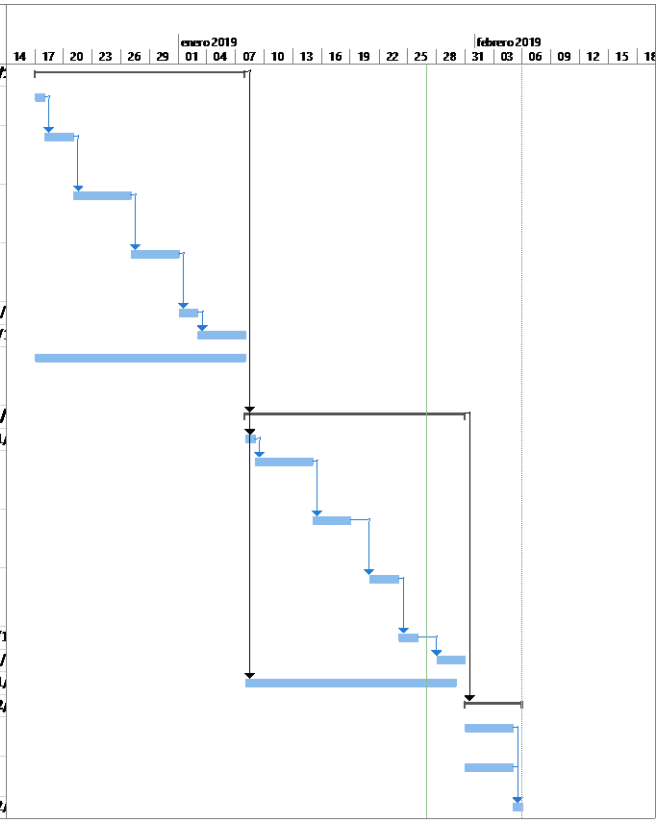

Figura 386. 2ª parte de la planificación del proyecto

11 Presupuesto

En el presente apartado se recogen las principales partidas presupuestarias para la realización del proyecto organizadas en distintos capítulos de acuerdo con la categoría de cada una de estas partidas.

11.1 Capítulo I. Recursos hardware

La Tabla 20 recoge todos aquellos recursos hardware necesarios para la implementación del sistema y la realización de experimentos con las redes neuronales.

Descripción	Cantidad	Precio Ud	Precio total
Equipo de gama media	1 Ud	629,00 €	629,00 €
Equipo de procesamiento	1 Ud	2.969,00 €	2.969,00 €
Disco duro para imágenes	1 Ud	50, 00 €	150 00 €
TOTAL			3.748,00 €

Tabla 20. Presupuesto de los recursos hardware del proyecto

11.2 Capítulo II. Recursos software

En la Tabla 21 se muestran todas aquellas licencias de software utilizadas para la realización del proyecto.

Descripción	Cantidad	Precio Ud	Precio total
Windows 10 Home	1 licencia	135,00 €	135,00 €
Windows 7 Professional	1 licencia	199,00 €	199,00 €
Microsoft Office 365 ProPlus	6 meses	12,90 €	77,40 €
Jetbrains Pycharm Professional	1 año	89,00 €	89,00 €
TOTAL			500,40 €

Tabla 21. Presupuesto de los recursos software del proyecto

11.3 Capítulo III. Recursos humanos

La Tabla 22 muestra los recursos humanos necesarios para implementar el sistema, así como el coste hora sujeto a dicho recurso con los constes de seguridad social ya imputados.

Descripción	Cantidad	Coste/hora	Precio total
Ingeniero de datos	1.029 horas	18,62 €	19.159,98 €
TOTAL			19.159,98 €

Tabla 22.Presupuesto de los recursos humanos del proyecto

11.4 Resumen del presupuesto

Finalmente, en la Tabla 23 se recopilan la suma de cada uno de los capítulos del presupuesto, así como los gastos generales imputados al proyecto asociados a costes de electricidad, internet y viajes, entre otros.

DESCRIPCIÓN	PRECIO
Presupuesto de ejecución material	
CAPÍTULO 1. RECURSOS HARDWARE	3.748,00 €
CAPÍTULO 2. RECURSOS SOFTWARE	500,40 €
CAPÍTULO 3. RECURSOS HUMANOS	19.159,98 €
TOTAL CAPÍTULOS	23.408,38 €
Gastos generales	
12% PEM	2.809,012 €
TOTAL PRESUPUESTO	26.217,39 €

Tabla 23. Resumen del presupuesto del proyecto

12 Conclusión

En definitiva, este proyecto ha consistido en el estudio de la aplicación de redes neuronales convolucionales a sistemas de detección de emisiones de nubes contaminantes a la atmósfera, así como la experimentación con CNNs que hacen uso de información física de la imagen, sin tener en cuenta el flujo temporal. Este análisis temporal se realiza, de forma posterior a la clasificación de imágenes de forma independiente de la red, por medio de un algoritmo que tiene en cuenta la clase de cada imagen de acuerdo con el orden cronológico de las imágenes.

Tras llevar a cabo el estudio del estado de la técnica se ha podido comprobar la escasez de trabajos basados en CNNs con objeto de abordar este problema. Cabe destacar que los trabajos encontrados no abordan la problemática de detección de emisiones, sino la de detección de humos con imágenes y condiciones ambientales muy diferentes a las consideradas en este trabajo.

Debido a esto el trabajo se ha orientado a determinar las modificaciones y optimizaciones necesarias para aplicar las redes de los problemas seleccionados al problema de la detección de emisiones atmosféricas. Tal y como se ha podido comprobar tras optimizar estas redes se ha logrado mejorar los resultados llegando a obtener un 80,95% de precisión (PPV) y de recall (TPR) a la hora de predecir secuencias de emisiones en un día con condiciones atmosféricas adversas como atardeceres y amaneceres.

Esto es un gran avance en comparación con el sistema actual basado en técnicas de visión ya que este no es capaz diferenciar entre emisiones y nubes durante las fases de atardecer y amanecer.

13 REFERENCIAS

Altun M, Celenk M. 2013. Smoke Detection in Video Surveillance Using Optical Flow and Green's Theorem.

Bogush R, Brovko N. 2011. An Efficient Smoke Detection Algorithm for Video Surveillance Systems Based on Optical Flow. *The Eleventh International Conference on Pattern Recognition and Information Processing, At Minsk, Belarus*.

Chen T, Yin Y, Huang S, Ye Y. 2006. The Smoke Detection for Early Fire-Alarming System Base on Video Processing. In *2006 International Conference on Intelligent Information Hiding and Multimedia,*. IEEE: Pasadena, CA, USA http://ieeexplore.ieee.org/document/4041753/.

Cheng M-M, Zhang Z, Lin W-Y, Torr P. 2014. BING: Binarized Normed Gradients for Objectness Estimation at 300fps. In *2014 IEEE Conference on Computer Vision and Pattern Recognition,*. IEEE: Columbus, OH, USA http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6909816.

Ciresan DC, Meier U, Masci J, Gambardella LM, Schmidhuber J. 2011. Flexible, High Performance Convolutional Neural Networks for Image Classification. *Twenty-Second International Joint Conference on Artificial Intelligence.*

Cucchiara R, Grana C, Piccardi M, Prati A. 2001. Detecting objects, shadows and ghosts in video streams by exploiting color and motion information. In *Proceedings 11th International Conference on Image Analysis and Processing,*. IEEE Comput. Soc: Palermo, Italy http://ieeexplore.ieee.org/document/957036/.

Fang L. 2008. A Study of the Key Technology of Forest Fire Prevention Based on a Cooperation of Video Monitor and GIS. IEEE Fourth International Conference on Natural Computation: China http://ieeexplore.ieee.org/document/4667463/.

Fawcett T. 2004. ROC Graphs: Notes and Practical Considerations for Researchers. *ResearchGate: Kluwer Academic Publishers*.

Genovese A. 2011. Wildfire smoke detection using computational intelligence techniques. IEEE: Milano, Italy http://ieeexplore.ieee.org/document/6059930/.

Girshick R. 2015. Fast R-CNN. *Cornell University Library*, April 30 http://arxiv.org/abs/1504.08083.

Girshick R, Donahue J, Darrell T, Malik J. 2013. Rich feature hierarchies for accurate object detection and semantic segmentation. *Cornell University Library*, November 11 http://arxiv.org/abs/1311.2524.

He K, Gkioxari G, Dollár P, Girshick R. 2017. Mask R-CNN. *Cornell University Library*, March 20 http://arxiv.org/abs/1703.06870.

Hohberg SP. 2015. Wildfire Smoke Detection using Convolutional Neural Networks. *Institut für Planetenforschung Master Thesis*.

Hsu Y-C, Dille P, Sargent R, Nourbakhsh I. 2016. Industrial Smoke Detection and Visualization. *Cornell University Library*.

Jakov T, Krstini D. 2010. Wildfire smoke-detection algorithms evaluation. *VI International Conference on Forest Fire Research*.

Krstinić D, Stipaničev D, Jakovčević T. 2009. HISTOGRAM-BASED SMOKE SEGMENTATION IN FOREST FIRE DETECTION SYSTEM. *Information Technology And Control*.

Lanczos C. 1950. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, **45**: 255.

LeCun Y, Bengio Y, Hinton G. 2015. Deep learning. *Nature*, **521**. http://www.nature.com/articles/nature14539.

Lima T. 2017. Human action recognition with 3D convolutional neural network. IEEE http://ieeexplore.ieee.org/document/8285700/.

Luna-González JA. 2016. Comparación de Arquitecturas de Redes Neuronales Convolucionales para la Clasificación de Imágenes de Ojos. ResearchGate.

Maruta. 2009. Smoke detection in open areas using its texture features and time series properties. IEEE http://ieeexplore.ieee.org/document/5214564/.

Piccinini P, Calderara S, Cucchiara R. 2008. RELIABLE SMOKE DETECTION SYSTEM IN THE DOMAINS OF IMAGE ENERGY AND COLOR.

Quintero C. 2018. Uso de Redes Neuronales Convolucionales para el Reconocimiento Automático de Imágenes de Macroinvertebrados para el Biomonitoreo Participativo. *KnE Engineering*, **3**. https://knepublishing.com/index.php/KnE-Engineering/article/view/1462.

Ren S, He K, Girshick R, Sun J. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *Cornell University Library*, June 4 http://arxiv.org/abs/1506.01497.

Simonyan K, Zisserman A. 2014. Two-Stream Convolutional Networks for Action Recognition in Videos. *Cornell University Library*.

Stauffer C. 1999. Adaptive background mixture models for real-time tracking. IEEE Comput. Soc http://ieeexplore.ieee.org/document/784637/.

Szegedy C. 2016. Rethinking the Inception Architecture for Computer Vision. IEEE http://ieeexplore.ieee.org/document/7780677/.

Tao C, Zhang J, Wang P. 2016. Smoke Detection Based on Deep Convolutional Neural Networks. In *2016 International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration (ICIICII)*,. IEEE: Wuhan, China http://ieeexplore.ieee.org/document/7823512/.

Toreyin BU, Dedeogˇlu Y. 2005. Wavelet based real-time smoke detection in video.

Toreyin BU, Dedeoglu Y, Cetin AE. 2006. Contour based smoke detection in video using wavelets.

Uijlings JRR. 2013. Selective Search for Object Recognition. *International Journal of Computer Vision*, **104**. http://link.springer.com/10.1007/s11263-013-0620-5.

Wang J. 2017. Two-stream Convolutional Neural Network for Natural Gas Leak Quantification.

Yao Y. 2017. Chimney and condensing tower detection based on faster R-CNN in high resolution remote sensing images. IEEE http://ieeexplore.ieee.org/document/8127710/.

Yin Z. 2017. A Deep Normalization and Convolutional Neural Network for Image Smoke Detection. *IEEE Access*, **5**. http://ieeexplore.ieee.org/document/8022860/.

Zitnick CL. 2014. Edge Boxes: Locating Object Proposals from Edges. In *Computer Vision – ECCV 2014*,. Springer International Publishing: Cham; 391–405 http://link.springer.com/10.1007/978-3-319-10602-1_26.

14ANEXOS

14.1 ANEXO I

Código de comparación de algoritmos de redimensionado.

```
import os
from PIL import Image
import cv2
import numpy as np
from keras.preprocessing import image


# Load the image
PATH = os.getcwd()
originalImg = image.load_img(PATH + "/.."
+"/VighumImages/0_SinEmision/20170309_152355.jpg")
originalImg = originalImg.convert('RGB')

# Get the new dimensions of the image
original_width, original_height = originalImg.size
width = int(original_width/16)
height = int(original_height/16)

# Resizing algorithms
nearest = originalImg.resize((width, height), Image.NEAREST)       # based
on nearest neighbour algorithm
bilinear = originalImg.resize((width, height), Image.BILINEAR)      #
linear interpolation with a 2x2 kernel
bicubic = originalImg.resize((width, height), Image.BICUBIC)       # cubic
spline interpolation with a 4x4 kernel
lanczos = originalImg.resize((width, height), Image.LANCZOS)     # based
on lanczos algorithm

originalImgCV2 = cv2.imread(PATH + "/.."
+"/VighumImages/0_SinEmision/20170309_152355.jpg")
originalImgCV2 = cv2.cvtColor(originalImgCV2, cv2.COLOR_BGR2RGB)
cv2_area = cv2.resize(originalImgCV2, (width, height),
interpolation=cv2.INTER_AREA)
cv2_nearest = cv2.resize(originalImgCV2, (width, height),
interpolation=cv2.INTER_NEAREST)
cv2_linear = cv2.resize(originalImgCV2, (width, height),
interpolation=cv2.INTER_LINEAR)
cv2_cubic = cv2.resize(originalImgCV2, (width, height),
interpolation=cv2.INTER_CUBIC)
cv2_lanczos = cv2.resize(originalImgCV2, (width, height),
interpolation=cv2.INTER_LANCZOS4)
cv2_area = cv2.cvtColor(cv2_area, cv2.COLOR_RGB2BGR)
cv2_nearest = cv2.cvtColor(cv2_nearest, cv2.COLOR_RGB2BGR)
cv2_linear = cv2.cvtColor(cv2_linear, cv2.COLOR_RGB2BGR)
cv2_cubic = cv2.cvtColor(cv2_cubic, cv2.COLOR_RGB2BGR)
cv2_lanczos = cv2.cvtColor(cv2_lanczos, cv2.COLOR_RGB2BGR)

nearest.save("PIL_NEAREST.jpg")
bilinear.save("PIL_BILINEAR.jpg")
```

```
bicubic.save("PIL_BICUBIC.jpg")
lanczos.save("PIL_LANCZOS.jpg")

cv2.imwrite("CV2_AREA.jpg", cv2_area)
cv2.imwrite("CV2_NEAREST.jpg", cv2_nearest)
cv2.imwrite("CV2_LINEAR.jpg", cv2_linear)
cv2.imwrite("CV2_CUBIC.jpg", cv2_cubic)
cv2.imwrite("CV2_LANCZOS.jpg", cv2_lanczos)
```

14.2 ANEXO II

```
from random import shuffle
import glob
import sys
import json
import os
import tensorflow as tf
import Dataset.ImgTools as img_tools


def _int64_feature(value):
    # value must be a numpy array.
    return tf.train.Feature(int64_list=tf.train.Int64List(value=[value]))


def _bytes_feature(value):
    return tf.train.Feature(bytes_list=tf.train.BytesList(value=[value]))


def create_tfrecord(out_filename, addrs, labels, img_height, img_width):
    print("Converting: " + out_filename)
    # Number of images. Used when printing the progress.
    num_images = len(addrs)
    # Open a TFRecordWriter for the output-file.
    with tf.python_io.TFRecordWriter(out_filename) as writer:
        # Iterate over all the image-paths and class-labels.
        for i, (path, label) in enumerate(zip(addrs, labels)):
            # Print the percentage-progress.
            if not i % 50:
                print('Data: {}/{}'.format(i, len(addrs)))
                sys.stdout.flush()
            # Load the image-file using matplotlib's imread function.
            img, shape = img_tools.load_img_cv2(path, img_height,
img_width)
            # Create a dict with the data we want to save in the
            # TFRecords file. You can add more relevant data here.
            feature = {'data': _bytes_feature(img.tostring()),
                       'label': _int64_feature(label),
                       'index': _bytes_feature(path.encode("utf8"))}

            # Create an example protocol buffer
            example =
tf.train.Example(features=tf.train.Features(feature=feature))

            # Serialize to string and write on the file
            writer.write(example.SerializeToString())
```

```
################# CREATING TF RECORDS #################
os.chdir("..")
PATH = os.getcwd()
CONFIG = json.loads(open(os.path.join(PATH, "Config.json")).read())
IS_LABELED = CONFIG["dataset_labeled"]
GEN_TEST = CONFIG["generate_test_dataset"]

# Read from label directories if the dataset is labeled
if GEN_TEST == 0:
    DATASET_PATH = CONFIG["dataset_path"]
else:
    DATASET_PATH = CONFIG["test_dataset_path"]

if IS_LABELED == 1:
    IMGS_DATASET_PATH = os.path.join(DATASET_PATH, '*/*.jpg')
else:
    if GEN_TEST == 1:
        IMGS_DATASET_PATH = os.path.join(DATASET_PATH, '*.jpg')
    else:
        print("Cannot create a TFRecord for training and validation from
not labeled directory")
        exit()

RAW_IMG_HEIGHT = CONFIG["raw_img_height"]
RAW_IMG_WIDTH = CONFIG["raw_img_width"]
RESIZE_RATE = CONFIG["resize_rate"]
IMG_HEIGHT = int(RAW_IMG_HEIGHT * RESIZE_RATE)
IMG_WIDTH = int(RAW_IMG_WIDTH * RESIZE_RATE)

# read addresses and labels from the dataset folder
img_paths = glob.glob(IMGS_DATASET_PATH)
labels = [1 if '1_ConEmision' in addr else 0 for addr in img_paths]  # 0
= 0_SinEmision, 1 = 1_ConEmision

# shuffle data
c = list(zip(img_paths, labels))
shuffle(c)
img_paths, labels = zip(*c)

if GEN_TEST == 0:
    TRAIN_SET_SIZE = CONFIG["train_size"]
    VAL_SET_SIZE = CONFIG["val_size"]

    train_addrs = img_paths[0:int(TRAIN_SET_SIZE * len(img_paths))]
    train_labels = labels[0:int(TRAIN_SET_SIZE * len(labels))]
    val_addrs = img_paths[int(TRAIN_SET_SIZE *
len(img_paths)):int((VAL_SET_SIZE + TRAIN_SET_SIZE) * len(img_paths))]
    val_labels = labels[int(TRAIN_SET_SIZE *
len(img_paths)):int((VAL_SET_SIZE + TRAIN_SET_SIZE) * len(img_paths))]

    create_tfrecord(os.path.join(DATASET_PATH, 'train.tfrecords'),
                    train_addrs, train_labels, IMG_HEIGHT, IMG_WIDTH)
    create_tfrecord(os.path.join(DATASET_PATH, 'val.tfrecords'),
                    val_addrs, val_labels, IMG_HEIGHT, IMG_WIDTH)

else:
    TEST_SET_SIZE = CONFIG["test_size"]
```

```
if IS_LABELED == 1:
    create_tfrecord(os.path.join(DATASET_PATH, 'test.tfrecords'),
                    img_paths, labels, IMG_HEIGHT, IMG_WIDTH)
else:
    create_tfrecord(os.path.join(DATASET_PATH, 'test_day.tfrecords'),
                    img_paths, labels, IMG_HEIGHT, IMG_WIDTH)
```

14.3 ANEXO III

14.3.1 Cross testing con el día 1

A partir de la Figura 387, Figura 388, Figura 389, Figura 390, Figura 391, Figura 392 y Figura 393 se obtienen los resultados obtenidos del entrenamiento y testing con el día 1.

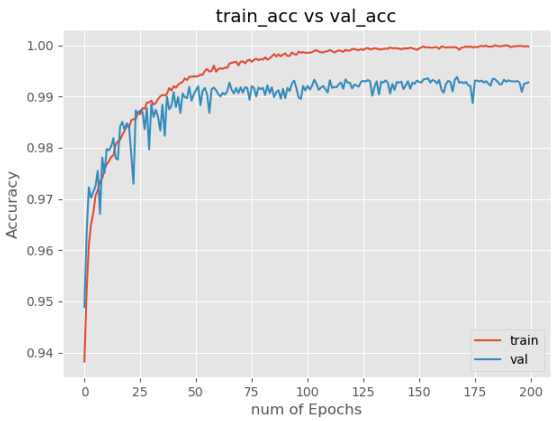

Figura 387. Accuracy de Tao y cross testing con el día 1

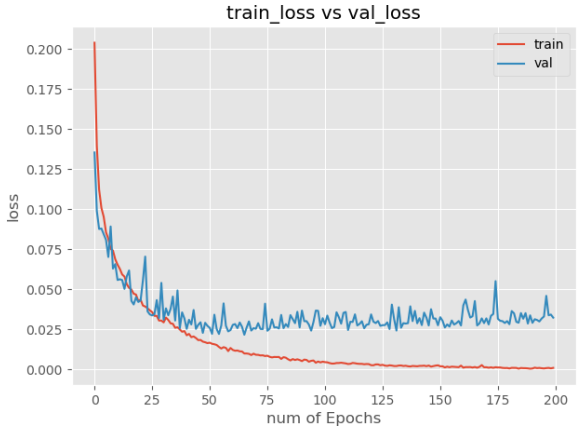

Figura 388. Loss de Tao y cross testing con el día 1

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	54.19	51.27	49.49	47.59	46.45	44.67	43.27	41.5	39.47
PPV	67.89	79.22	82.45	83.89	86.12	87.56	88.34	89.59	90.67

Figura 389. Umbrales de Tao y cross testing con el día 1

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	86.12	46.45	97.49	99.64	60.35	97.21

Figura 390. Métricas de Tao y cross testing con el día 1

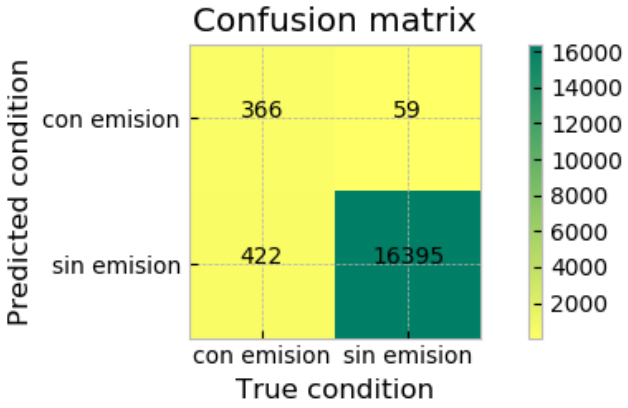

Figura 391. Matriz de confusión de Tao y cross testing con el día 1

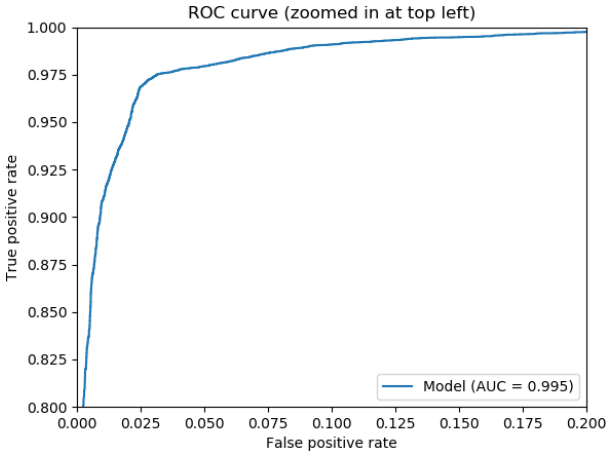

Figura 392. Curva ROC de Tao y cross testing con el día 1

	TP	FP	FN
0.5	29	2	28

Figura 393. Predicción de secuencias de Tao y cross testing con el día 1

14.3.2 Cross testing con el día 10

A partir de Figura 394, Figura 395, Figura 396, Figura 397, Figura 398 y Figura 400 se obtienen los resultados obtenidos del entrenamiento y testing con el día 10.

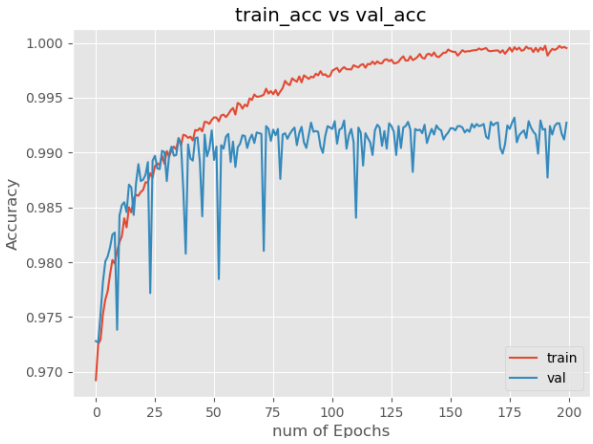

Figura 394. Accuracy de Tao y cross testing con el día 10

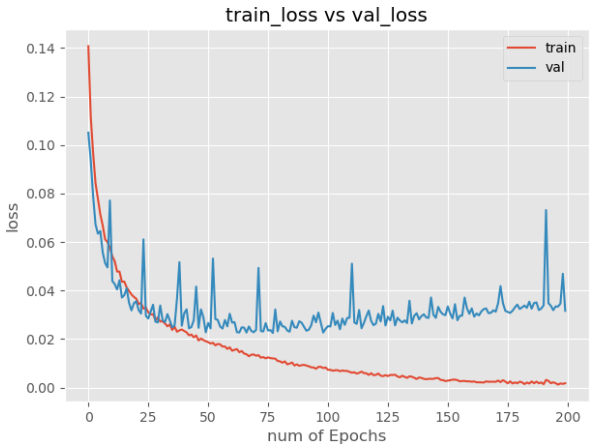

Figura 395. Loss de Tao y cross testing con el día 10

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	49.98	46.3	43.22	41.23	39.84	37.94	36.04	34.01	30.88
PPV	94.34	95.64	96.6	97.41	97.62	97.71	98.16	98.41	98.38

Figura 396. Umbrales de Tao y cross testing con el día 10

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	97.62	39.84	91.15	99.84	56.59	91.52

Figura 397. Métricas de Tao y cross testing con el día 10

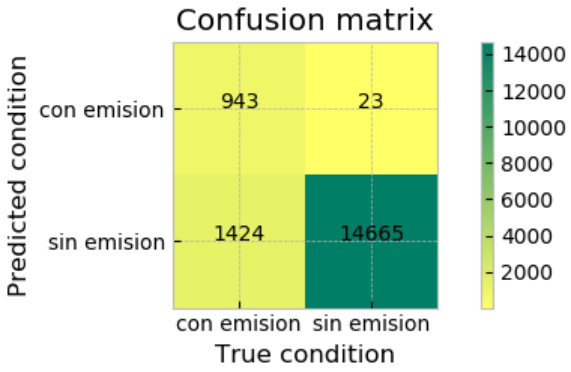

Figura 398. Matriz de confusión de Tao y cross testing con el día 10

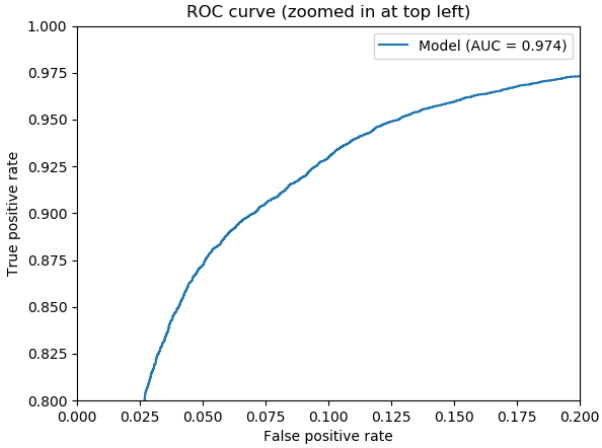

Figura 399. Curva ROC de Tao y cross testing con el día 10

	TP	FP	FN
0.5	35	1	38

Figura 400. Predicción de secuencias de Tao y cross testing con el día 10

14.3.3 Cross testing con el día 19

A partir de Figura 401, Figura 402, Figura 403, Figura 404, Figura 405, Figura 406 y Figura 407 se obtienen los resultados obtenidos del entrenamiento y testing con el día 19.

Figura 401. Accuracy de Tao y cross testing con el día 19

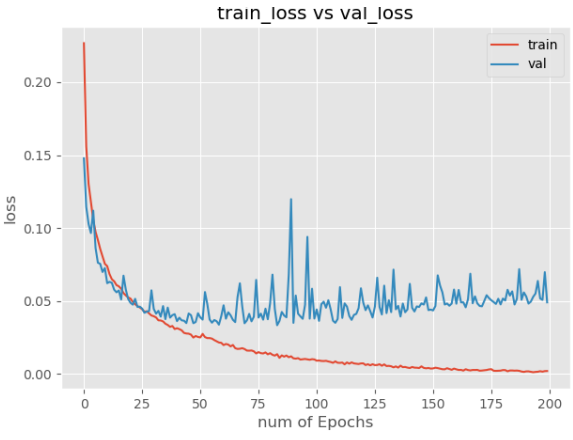

Figura 402. Loss de Tao y cross testing con el día 19

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	94.34	92.14	88.68	86.79	85.22	84.59	82.08	78.62	74.53
PPV	26.22	28.5	30.16	31.58	33.13	34.58	35.75	37.26	40.72

Figura 403. Umbrales de Tao y cross testing con el día 19

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	33.13	85.22	99.71	96.77	47.71	96.56

Figura 404. Métricas de Tao y cross testing con el día 19

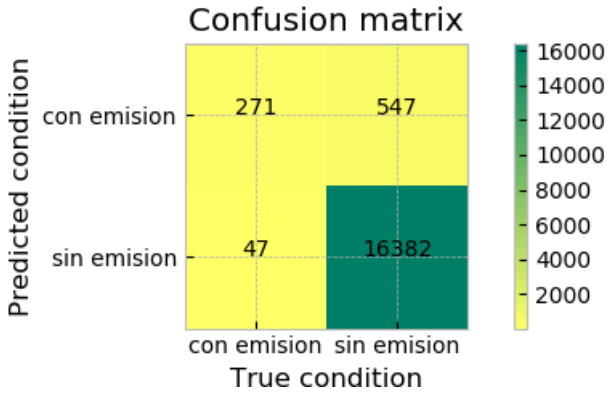

Figura 405. Matriz de confusión de Tao y cross testing con el día 19

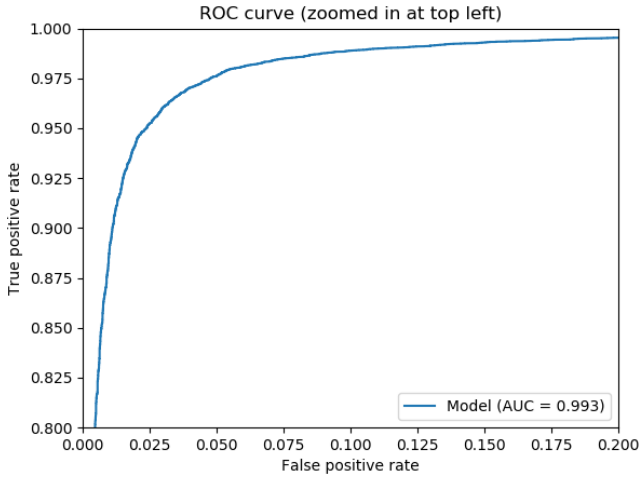

Figura 406. Curva ROC de Tao y cross testing con el día 19

	TP	FP	FN
0.5	21	33	2

Figura 407. Predicción de secuencias de Tao y cross testing con el día 19

14.4 ANEXO IV

14.4.1 Testing con el día 1

En la Figura 408, Figura 409, Figura 410, Figura 411, Figura 412, Figura 413 y Figura 414 se muestran los resultados de entrenamiento y testing, en los que se puede comprobar que, tal y como se esperaba, el clasificar imágenes de baja dimensión o densidad como imágenes sin emisión supondrá un decremento del número de falsas alarmas, pero también supone un gran decremento de verdaderos positivos.

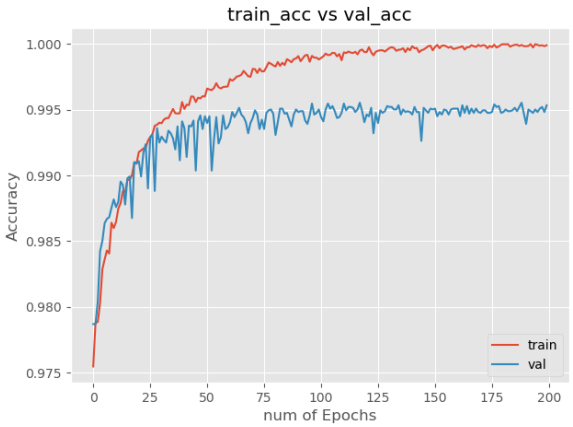

Figura 408. Accuracy de Tao y cross testing con el día 1 etiquetado según el segundo criterio

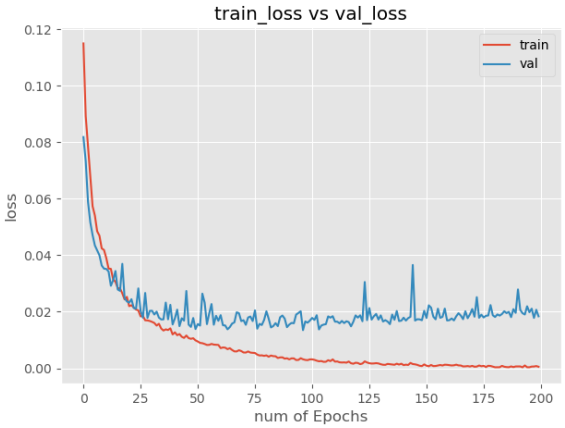

Figura 409. Loss de Tao y cross testing con el día 1 etiquetado según el segundo criterio

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	71.15	69.23	68.27	66.99	66.35	64.1	61.54	57.69	52.56
PPV	70.25	75.52	79.18	82.28	84.49	85.47	85.71	88.24	89.62

Figura 410. Umbrales de Tao y cross testing con el día 1 etiquetado según el segundo criterio

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	84.49	66.35	99.38	99.78	74.33	99.17

Figura 411. Métricas de Tao y cross testing con el día 1 etiquetado según el segundo criterio

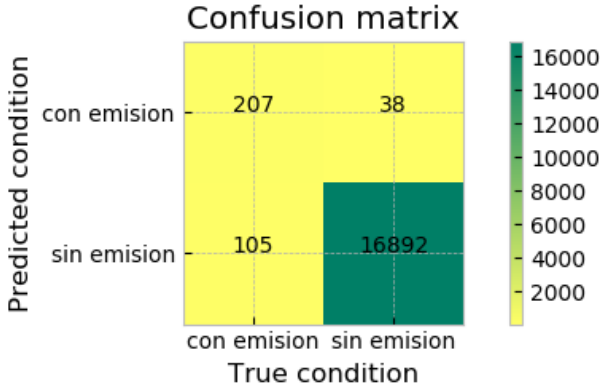

Figura 412. Matriz de confusión de Tao y cross testing con el día 1 etiquetado según el segundo criterio

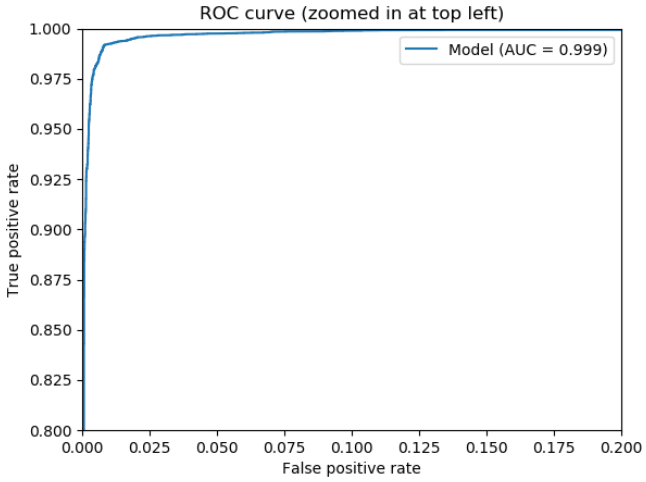

Figura 413. Curva ROC de Tao y cross testing con el día 1 etiquetado según el segundo criterio

	TP	FP	FN
0.5	18	5	9

Figura 414. Predicción de secuencias de Tao y cross testing con el día 1 etiquetado según el segundo criterio

14.4.2 Testing con el día 2

A partir de Figura 415, Figura 416, Figura 417, Figura 418, Figura 419, Figura 420 y Figura 421 se obtienen los resultados obtenidos del entrenamiento y testing con el día 2.

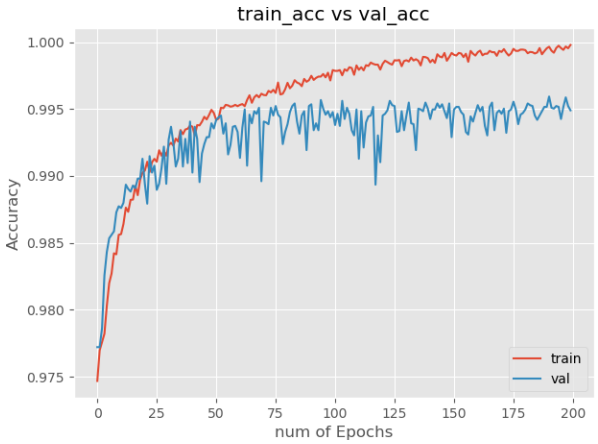

Figura 415. Accuracy de Tao y cross testing con el día 2 etiquetado según el segundo criterio

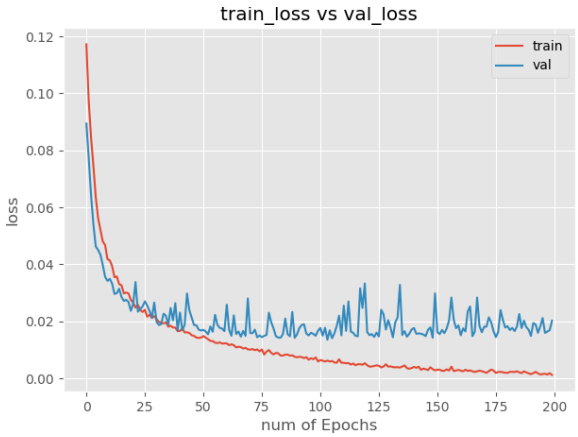

Figura 416. Loss de Tao y cross testing con el día 2 etiquetado según el segundo criterio

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	34.47	30.64	28.51	26.81	23.83	20.85	20.0	19.57	16.6
PPV	75.7	81.82	90.54	91.3	90.32	90.74	92.16	93.88	95.12

Figura 417. Umbrales de Tao y cross testing con el día 2 etiquetado según el segundo criterio

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	90.32	23.83	98.96	99.96	37.71	98.92

Figura 418. Métricas de Tao y cross testing con el día 2 etiquetado según el segundo criterio

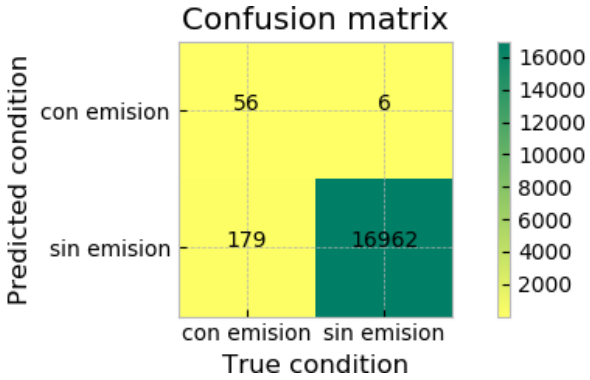

Figura 419. Matriz de confusión de Tao y cross testing con el día 2 etiquetado según el segundo criterio

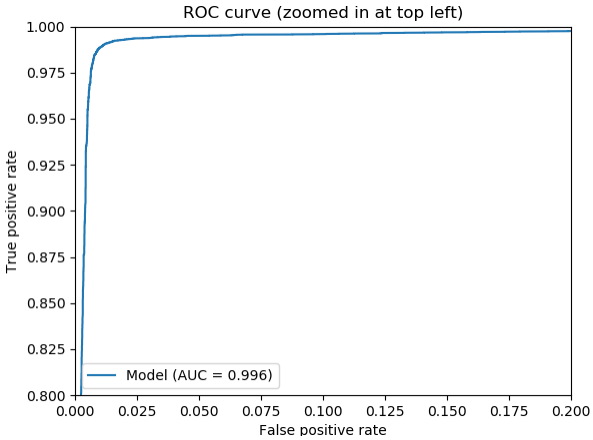

Figura 420. Curva ROC de Tao y cross testing con el día 2 etiquetado según el segundo criterio

	TP	FP	FN
0.5	9	0	17

Figura 421. Predicción de secuencias de Tao y cross testing con el día 2 etiquetado según el segundo criterio

14.4.3 Testing con el día 10

A partir de Figura 422, Figura 423, Figura 424, Figura 425, Figura 426, Figura 427 y Figura 428 se obtienen los resultados obtenidos del entrenamiento y testing con el día 10.

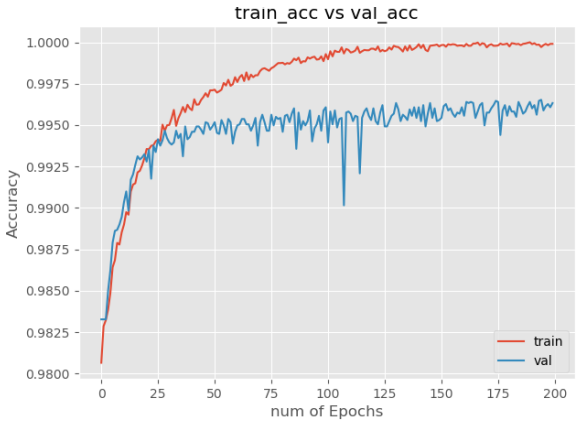

Figura 422. Accuracy de Tao y cross testing con el día 10 etiquetado según el segundo criterio

Figura 423. Loss de Tao y cross testing con el día 10 etiquetado según el segundo criterio

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	86.14	82.07	79.85	78.93	77.08	74.86	72.46	68.76	64.51
PPV	45.37	48.52	50.82	53.31	54.16	55.94	58.16	61.18	65.36

Figura 424. Umbrales de Tao y cross testing con el día 10 etiquetado según el segundo criterio

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.9	65.36	64.51	98.84	98.88	64.93	97.79

Figura 425. Métricas de Tao y cross testing con el día 10 etiquetado según el segundo criterio

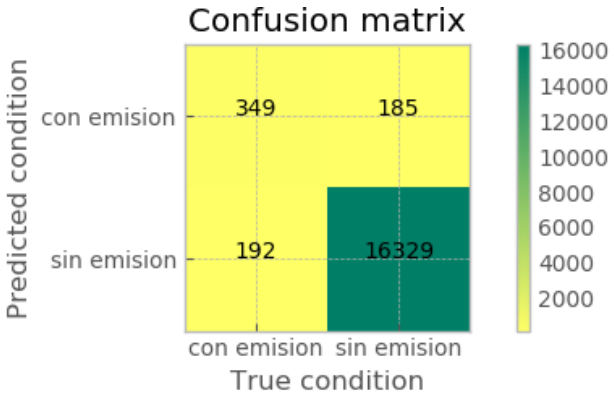

Figura 426. Matriz de confusión de Tao y cross testing con el día 10 etiquetado según el segundo criterio

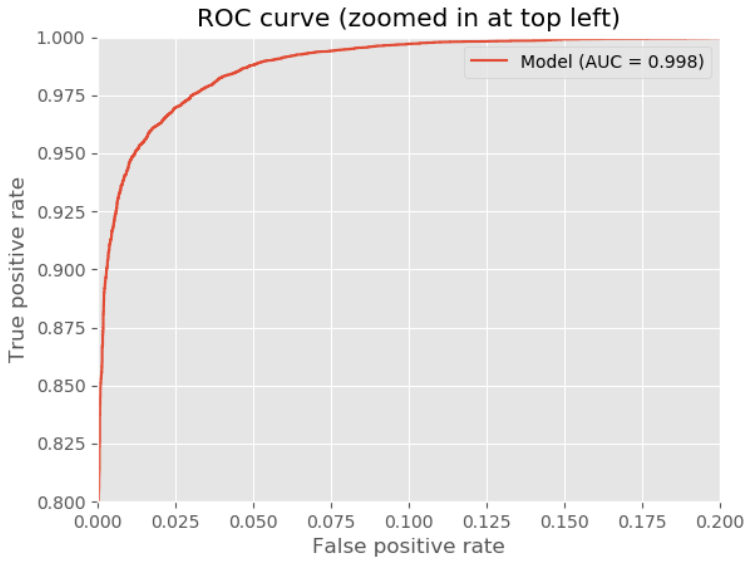

Figura 427. Curva ROC de Tao y cross testing con el día 10 etiquetado según el segundo criterio

	TP	FP	FN
0.5	26	25	7

Figura 428. Predicción de secuencias de Tao y cross testing con el día 10 etiquetado según el segundo criterio

14.4.4 Testing con el día 19

A partir de Figura 429, Figura 430, Figura 431, Figura 432, Figura 433, Figura 434 y Figura 435 se obtienen los resultados obtenidos del entrenamiento y testing con el día 19.

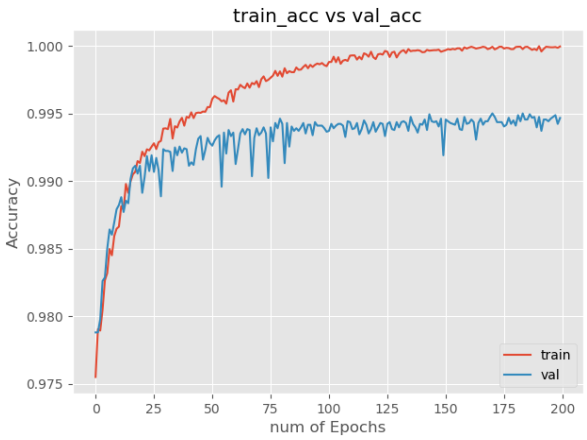

Figura 429. Accuracy de Tao y cross testing con el día 19 etiquetado según el segundo criterio

Figura 430. Loss de Tao y cross testing con el día 19 etiquetado según el segundo criterio

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	67.61	63.21	60.06	56.6	53.77	51.26	49.06	46.54	43.08
PPV	41.43	47.41	51.48	54.55	56.81	58.63	61.42	65.78	71.35

Figura 431. Umbrales de Tao y cross testing con el día 19 etiquetado según el segundo criterio

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	56.81	53.77	99.13	99.23	55.25	98.39

Figura 432. Métricas de Tao y cross testing con el día 19 etiquetado según el segundo criterio

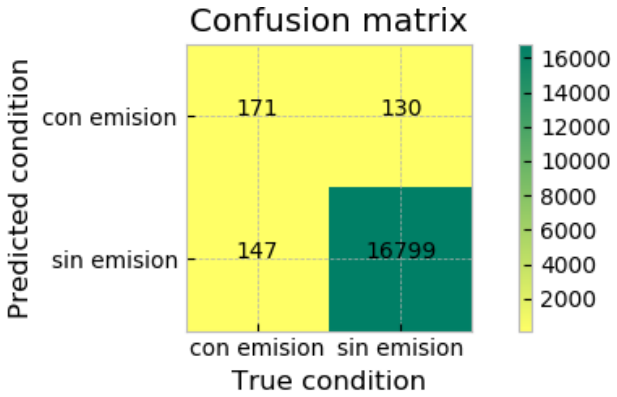

Figura 433. Matriz de confusión de Tao y cross testing con el día 19 etiquetado según el segundo criterio

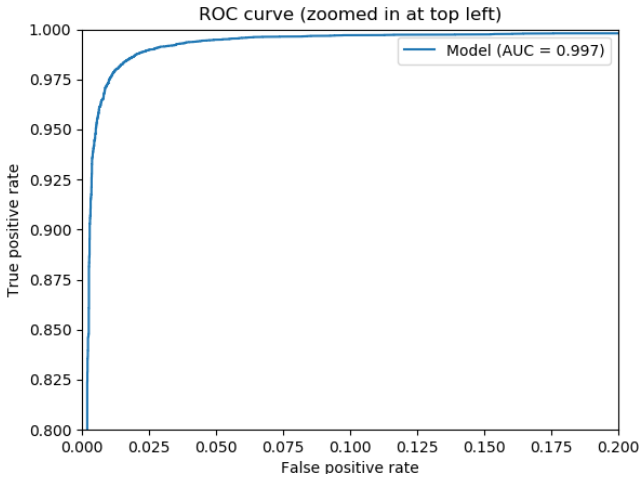

Figura 434. Curva ROC de Tao y cross testing con el día 19 etiquetado según el segundo criterio

	TP	FP	FN
0.5	15	11	9

Figura 435. Predicción de secuencias de Tao y cross testing con el día 19 etiquetado según el segundo criterio

14.5 ANEXO V

14.5.1 Testing con la combinación EEVTEE

A partir de la Figura 387, Figura 388, Figura 389, Figura 390, Figura 391, Figura 392 y Figura 393 se obtienen los resultados obtenidos del entrenamiento y testing intradías siguiendo el criterio EEVTEE de división de conjuntos.

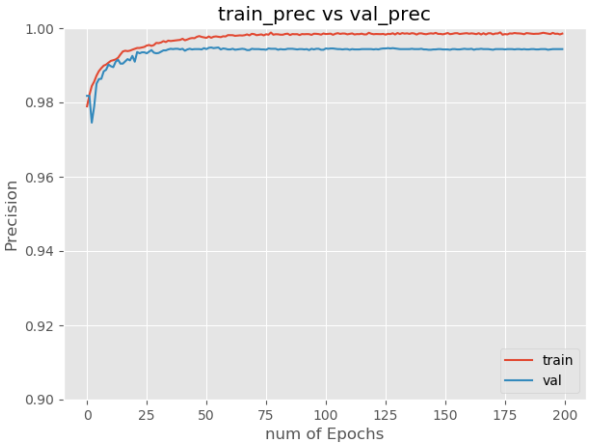

Figura 436. Precision de Tao y cross con la combinación EEVTEE

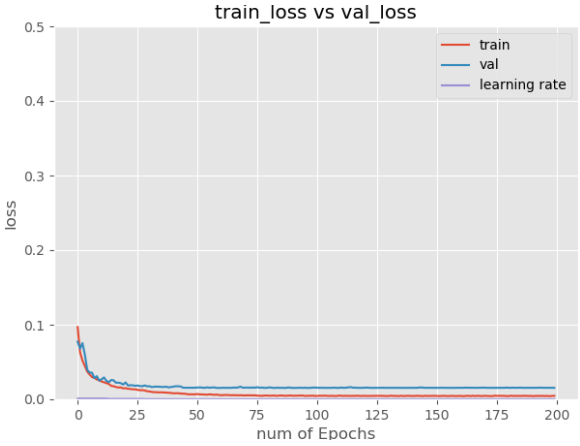

Figura 437. Loss de Tao y cross testing con la combinación EEVTEE

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	92.37	90.25	87.29	85.59	83.47	82.2	79.66	77.54	73.31
PPV	73.15	77.45	82.07	84.87	87.95	91.51	92.16	93.37	94.54

Figura 438. Umbrales de Tao y cross testing con la combinación EEVTEE

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.45	86.32	85.59	99.7	99.71	85.96	99.42

Figura 439. Métricas de Tao y cross testing con la combinación EEVTEE

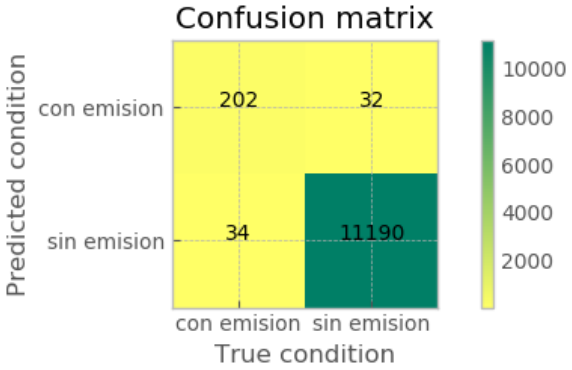

Figura 440. Matriz de confusión de Tao y cross testing con la combinación EEVTEE

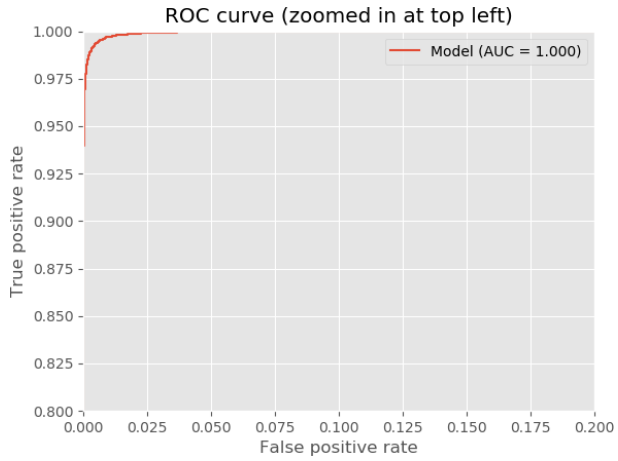

Figura 441. Curva ROC de Tao y cross testing con la combinación EEVTEE

14.5.2 Testing con la combinación VTEEEE

A partir de la Figura 387, Figura 388, Figura 389, Figura 390, Figura 391, Figura 392 y Figura 393 se obtienen los resultados obtenidos del entrenamiento y testing intradías siguiendo el criterio VTEEEE de división de conjuntos.

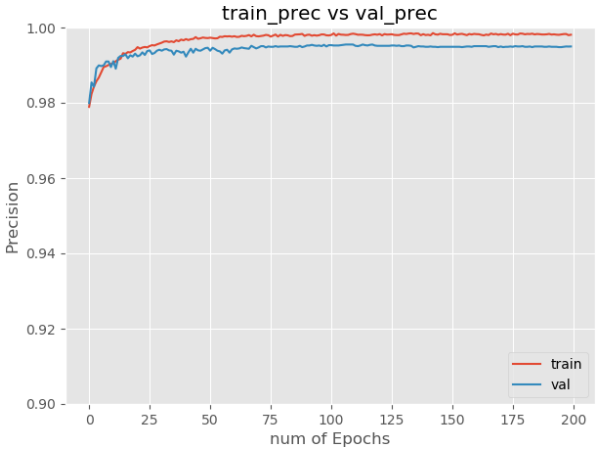

Figura 442. Precision de Tao y cross con la combinación VTEEEE

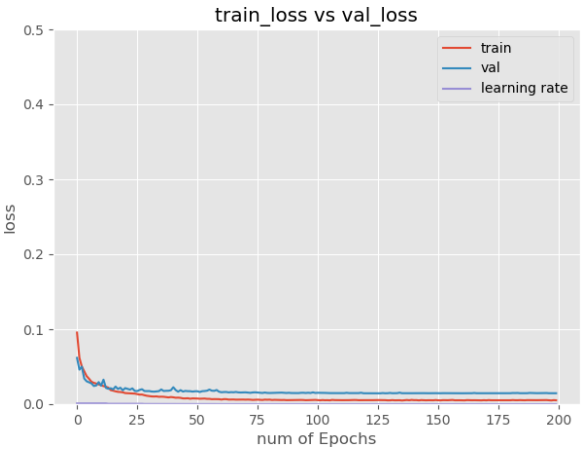

Figura 443. Loss de Tao y cross testing con la combinación VTEEEE

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	93.19	92.77	91.91	90.64	88.09	86.81	84.68	81.7	76.6
PPV	75.52	80.74	84.05	86.59	89.22	90.27	92.99	95.05	96.77

Figura 444. Umbrales de Tao y cross testing con la combinación VTEEEE

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	89.22	88.09	99.75	99.78	88.65	99.54

Figura 445. Métricas de Tao y cross testing con la combinación VTEEEE

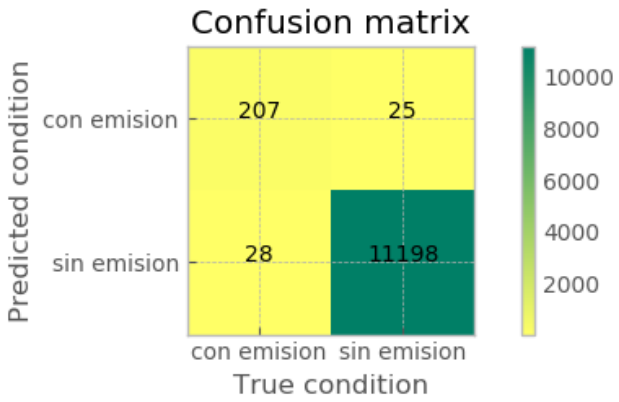

Figura 446. Matriz de confusión de Tao y cross testing con la combinación VTEEEE

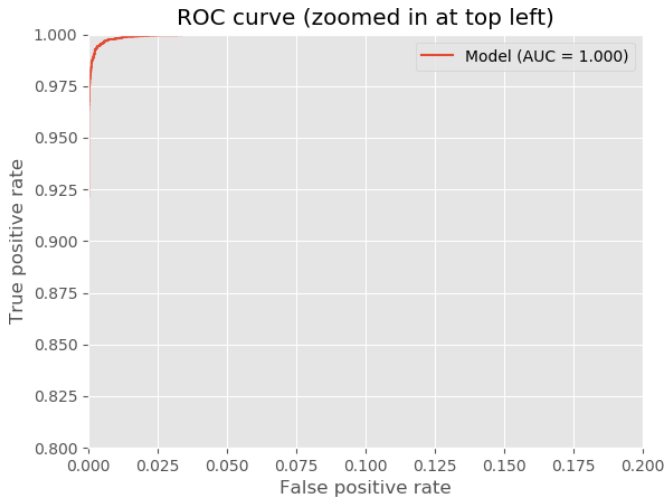

Figura 447. Curva ROC de Tao y cross testing con la combinación VTEEEE

14.6 ANEXO VI

14.6.1 Testing del día 20 al experimento con la configuración seleccionada para testing interdías

A partir de la Figura 387, Figura 388, Figura 389, Figura 390, Figura 391, Figura 392 y Figura 393 se obtienen los resultados obtenidos del testing con el día 20 del experimento testing del día 20 al experimento la configuración seleccionada para testing interdías.

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	67.65	60.92	58.22	53.37	50.94	46.36	42.86	37.74	29.65
PPV	30.28	37.29	44.63	51.16	56.93	61.43	70.98	77.35	86.61

Figura 448. Umbrales de Tao con proporción real para testing interdías con el día 20

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	56.93	50.94	98.92	99.15	53.77	98.11

Figura 449. Métricas de Tao con proporción real para testing interdías con el día 20

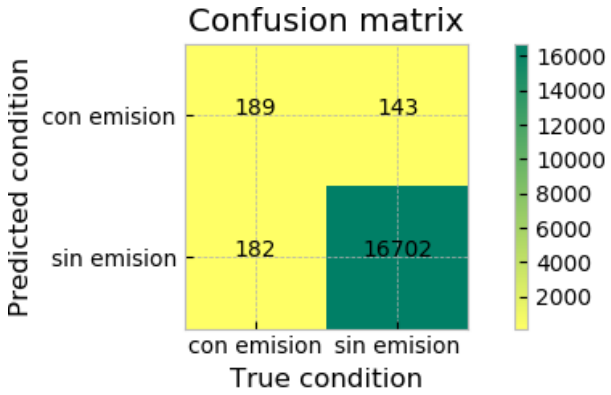

Figura 450. Matriz de confusión de Tao con proporción real para testing interdías con el día 20

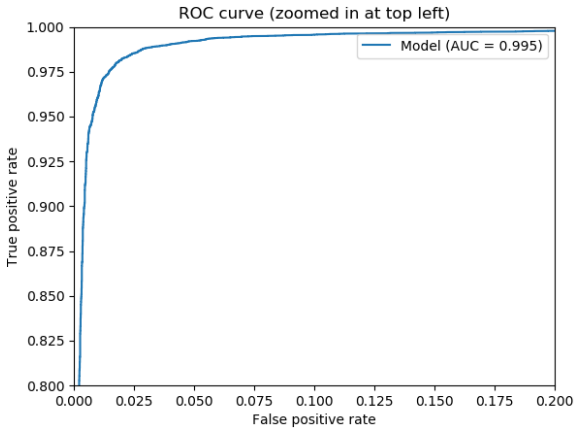

Figura 451. Curva ROC de Tao con proporción real para testing interdías con el día 20

14.6.2 Testing del día 20 al experimento de sustitución de learning rate fijo por step decay

A partir de la Figura 387, Figura 388, Figura 389, Figura 390, Figura 391, Figura 392 y Figura 393 se obtienen los resultados obtenidos del testing con el día 20 del experimento de sustitución de learning rate fijo por step decay.

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	59.57	56.06	53.91	52.29	50.67	47.71	44.2	41.24	37.2
PPV	53.51	59.94	63.9	69.04	75.81	79.37	84.54	87.43	92.62

Figura 452. Umbrales de Tao con testing intradías y decay con el día 20

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	75.81	50.67	98.92	99.64	60.74	98.59

Figura 453. Métricas de Tao con testing intradías y decay con el día 20

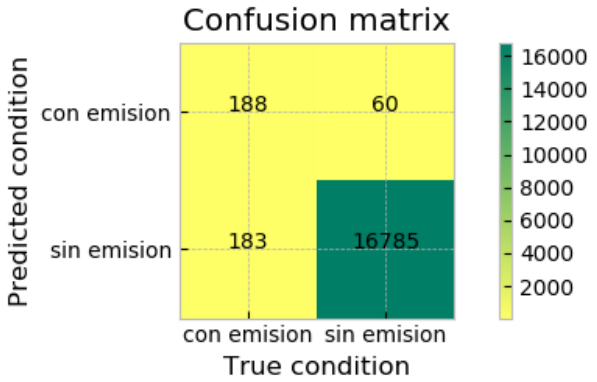

Figura 454. Matriz de confusión de Tao con testing intradías y decay con el día 20

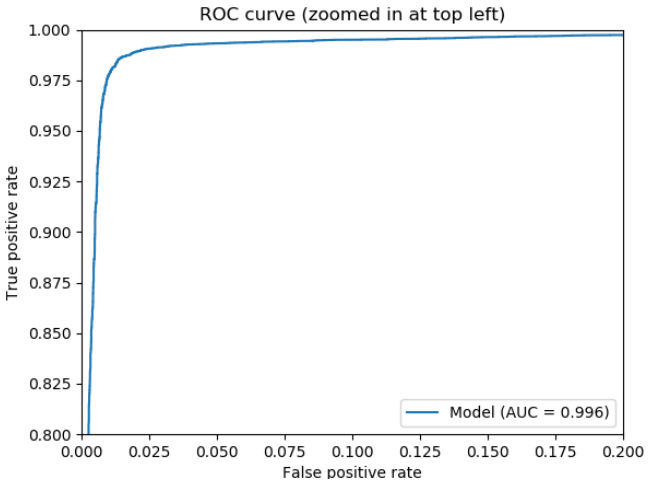

Figura 455. Curva ROC de Tao con testing intradías y decay con el día 20

14.6.3 Testing del día 20 al experimento de step decay y optimización de la métrica precisión

A partir de la Figura 387, Figura 388, Figura 389, Figura 390, Figura 391, Figura 392 y Figura 393 se obtienen los resultados obtenidos del testing con el día 20 del experimento de step decay y optimización de la métrica precisión.

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	63.88	59.3	57.14	55.26	53.37	50.67	46.63	42.59	38.81
PPV	50.43	56.7	61.81	65.71	68.99	72.59	75.55	80.2	87.27

Figura 456. Umbrales de Tao con testing intradías, decay y optimización de precisión con el día 20

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	68.99	53.37	98.98	99.47	60.18	98.48

Figura 457. Métricas de Tao con testing intradías, decay y optimización de precisión con el día 20

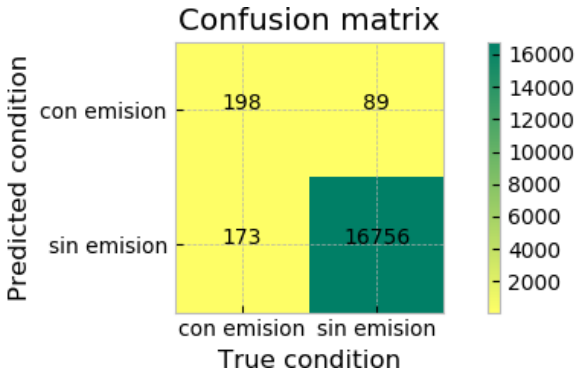

Figura 458. Matriz de confusión de Tao con testing intradías, decay y optimización de precisión con el día 20

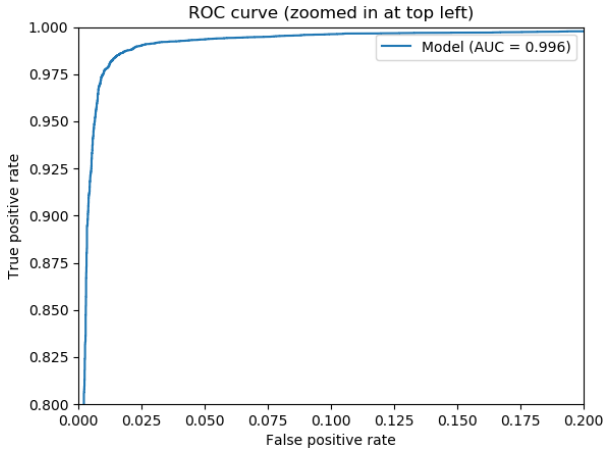

Figura 459. Curva ROC de Tao con testing intradías, decay y optimización de precisión con el día 20

14.6.4 Testing del día 20 al experimento de step decay y optimización de la métrica recall

A partir de la Figura 387, Figura 388, Figura 389, Figura 390, Figura 391, Figura 392 y Figura 393 se obtienen los resultados obtenidos del testing con el día 20 del experimento de step decay y optimización de la métrica recall.

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	62.53	58.49	53.1	50.67	47.98	47.17	44.2	42.59	37.74
PPV	46.49	54.94	59.88	65.51	69.53	75.76	78.47	83.6	85.37

Figura 460. Umbrales de Tao con testing intradías, decay y optimización de recall con el día 20

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	69.53	47.98	98.86	99.54	56.78	98.43

Figura 461. Métricas de Tao con testing intradías, decay y optimización de recall con el día 20

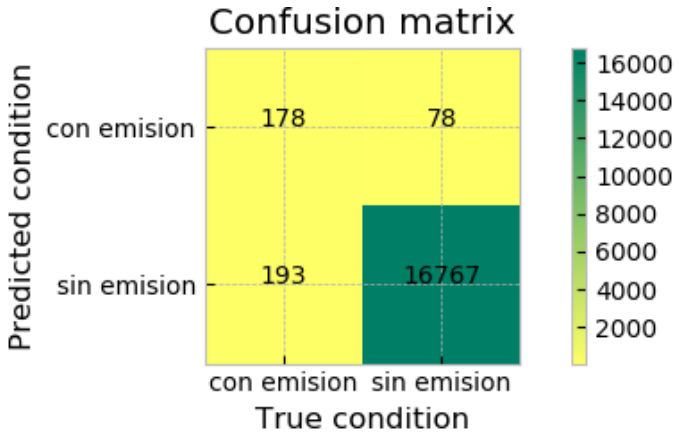

Figura 462. Matriz de confusión de Tao con testing intradías, decay y optimización de recall con el día 20

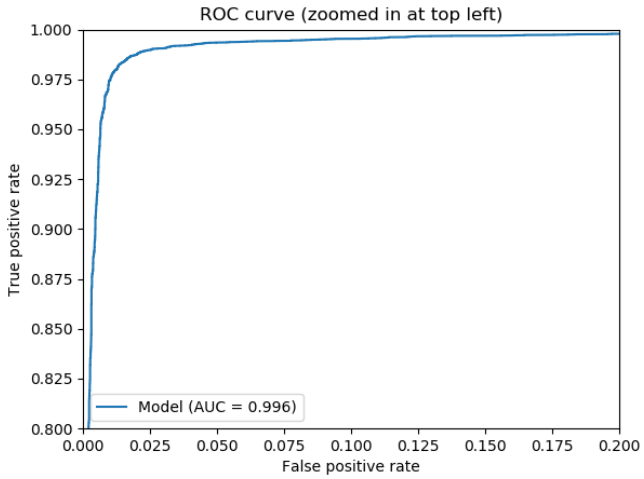

Figura 463. Curva ROC de Tao con testing intradías, decay y optimización de recall con el día 20

14.6.5 Testing del día 20 al experimento de step decay y penalización de loss

A partir de la Figura 387, Figura 388, Figura 389, Figura 390, Figura 391, Figura 392 y Figura 393 se obtienen los resultados obtenidos del testing con el día 20 del experimento de step decay y penalización de loss.

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	83.02	78.71	77.09	75.47	72.78	70.62	67.65	66.31	62.26
PPV	17.44	21.28	24.44	27.56	29.83	33.12	36.01	41.62	49.25

Figura 464. Umbrales de Tao con testing intradías, decay y penalización de loss con el día 20

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	29.83	72.78	99.38	96.23	42.32	95.72

Figura 465. Métricas de Tao con testing intradías, decay y penalización de loss con el día 20

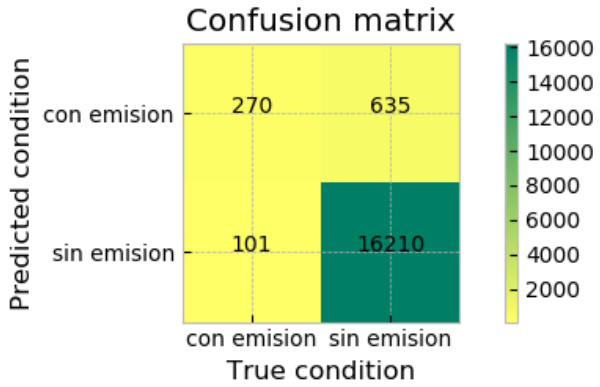

Figura 466. Matriz de confusión de Tao con testing intradías, decay y penalización de loss con el día 20

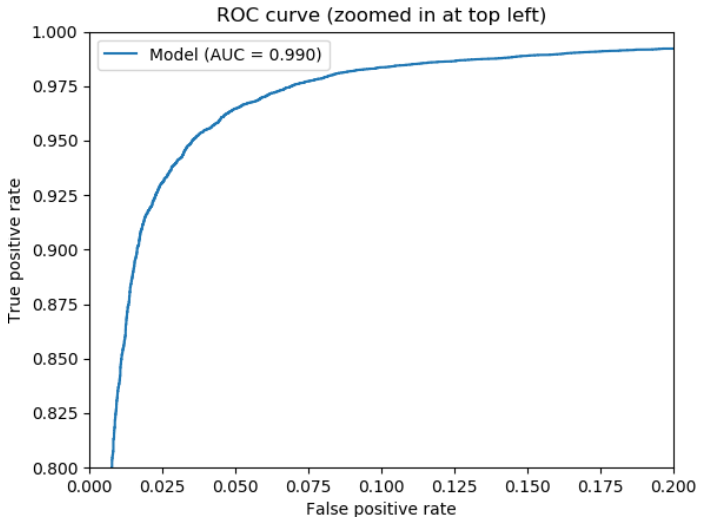

Figura 467. Curva ROC de Tao con testing intradías, decay y penalización de loss con el día 20

14.7 Anexo VII

14.7.1 Testing con la combinación EEVTEE

A partir de la Figura 387, Figura 388, Figura 389, Figura 390, Figura 391, Figura 392 y Figura 393 se obtienen los resultados obtenidos del entrenamiento y testing intradías siguiendo el criterio EEVTEE de división de conjuntos con la red [Yin 2017].

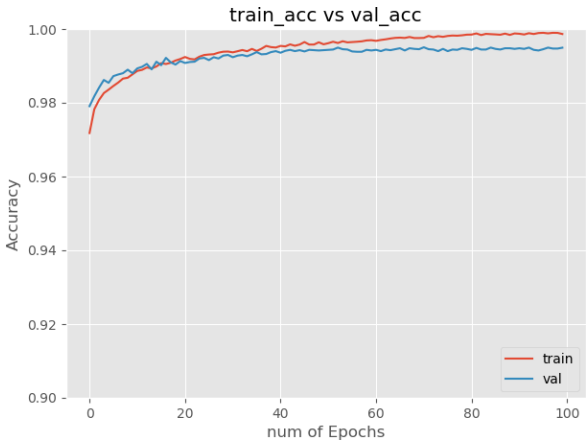

Figura 468. Accuracy de Yin y cross testing con la combinación EEVTEE

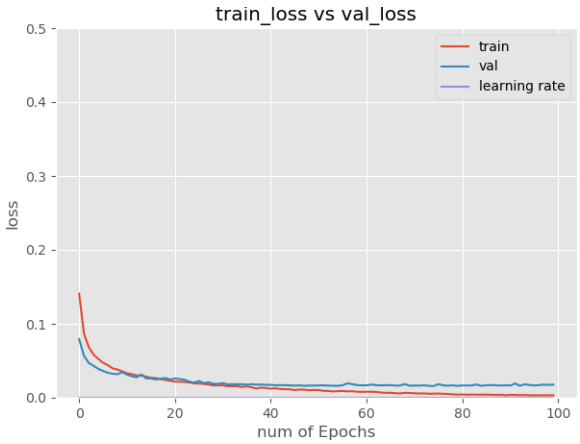

Figura 469. Loss de Yin y cross testing con la combinación EEVTEE

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	91.95	89.83	88.56	87.29	85.59	84.75	83.05	81.36	76.27
PPV	72.58	75.99	79.17	82.07	83.82	85.84	86.34	91.0	94.74

Figura 470. Umbrales de Yin y cross testing con la combinación EEVTEE

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	83.82	85.59	99.7	99.65	84.7	99.36

Figura 471. Métricas de Yin y cross testing con la combinación EEVTEE

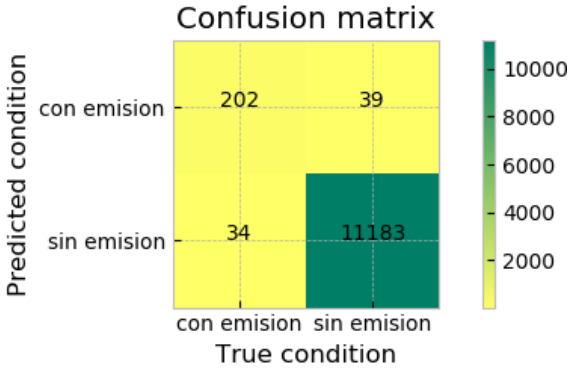

Figura 472. Matriz de confusión de Yin y cross testing con la combinación EEVTEE

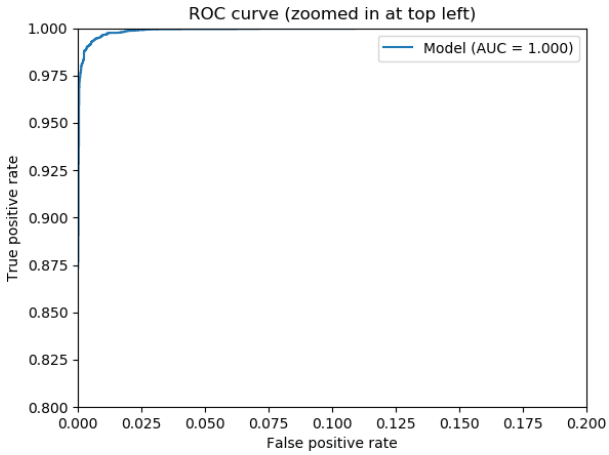

Figura 473. Curva ROC de Yin y cross testing con la combinación EEVTEE

14.7.2 Testing con la combinación VTEEEE

A partir de la Figura 387, Figura 388, Figura 389, Figura 390, Figura 391, Figura 392 y Figura 393 se obtienen los resultados obtenidos del entrenamiento y testing intradías siguiendo el criterio VTEEEE de división de conjuntos con la red [Yin 2017].

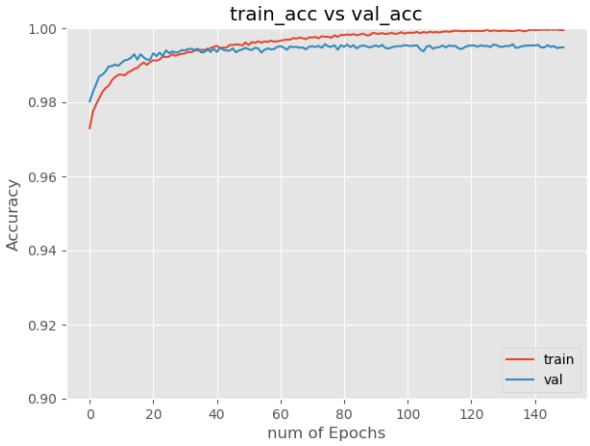

Figura 474. Accuracy de Yin y cross con la combinación VTEEEE

Figura 475. Loss de Yin y cross testing con la combinación VTEEEE

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	95.74	94.04	93.19	91.06	91.06	88.51	86.81	84.26	77.45
PPV	73.77	80.07	82.95	85.94	87.35	89.27	90.67	93.4	96.3

Figura 476. Umbrales de Yin y cross testing con la combinación VTEEEE

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	87.35	91.06	99.81	99.72	89.17	99.55

Figura 477. Métricas de Yin y cross testing con la combinación VTEEEE

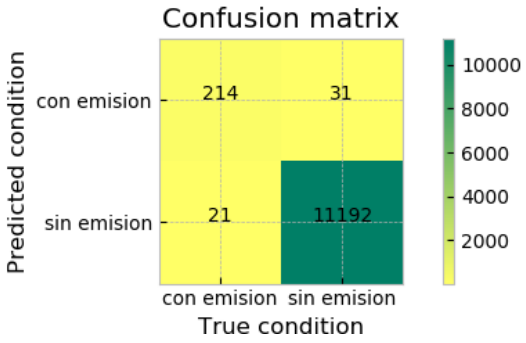

Figura 478. Matriz de confusión de Yin y cross testing con la combinación VTEEEE

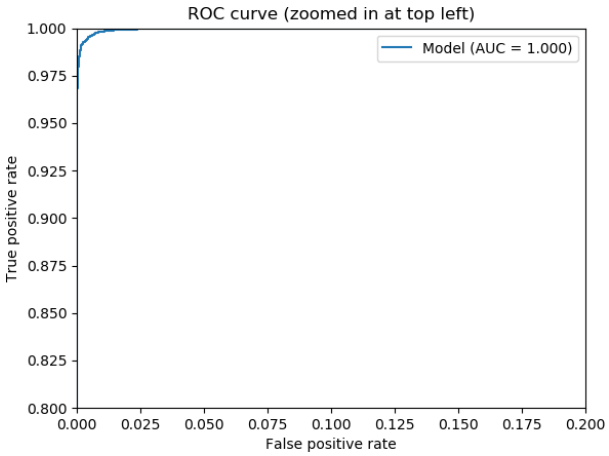

Figura 479. Curva ROC de Yin y cross testing con la combinación VTEEEE

14.8 Anexo VIII

14.8.1 Testing con la combinación EEVTEE

A partir de la Figura 387, Figura 388, Figura 389, Figura 390, Figura 391, Figura 392 y Figura 393 se obtienen los resultados obtenidos del entrenamiento y testing intradías siguiendo el criterio EEVTEE de división de conjuntos con la red caffenet.

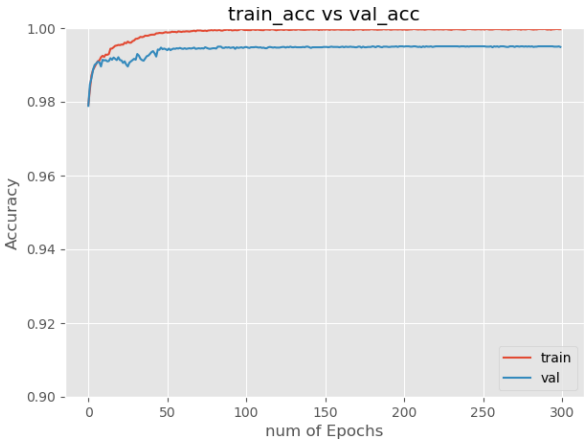

Figura 480. Accuracy de caffenet y cross con la combinación EEVTEE

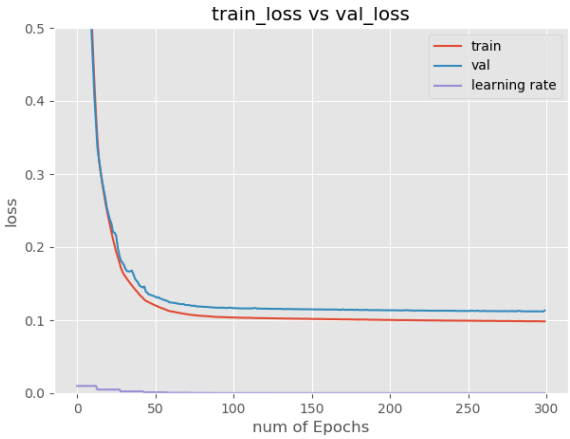

Figura 481. Loss de caffenet y cross testing con la combinación EEVTEE

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	88.14	86.02	85.17	83.05	81.36	79.66	77.97	75.42	72.46
PPV	79.09	83.2	83.75	85.59	86.49	89.1	90.2	90.36	92.93

Figura 482. Umbrales de caffenet y cross testing con la combinación EEVTEE

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	86.49	81.36	99.61	99.73	83.84	99.35

Figura 483. Métricas de caffenet y cross testing con la combinación EEVTEE

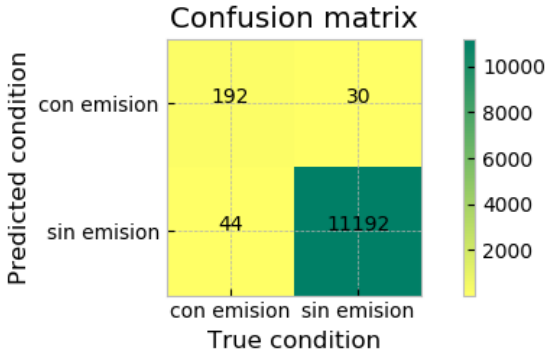

Figura 484. Matriz de confusión de caffenet y cross testing con la combinación EEVTEE

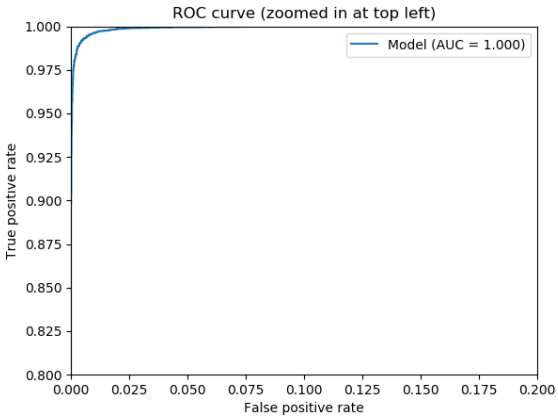

Figura 485. Curva ROC de caffenet y cross testing con la combinación EEVTEE

14.8.2 Testing con la combinación VTEEEE

A partir de la Figura 387, Figura 388, Figura 389, Figura 390, Figura 391, Figura 392 y Figura 393 se obtienen los resultados obtenidos del entrenamiento y testing intradías siguiendo el criterio VTEEEE de división de conjuntos con la red caffenet.

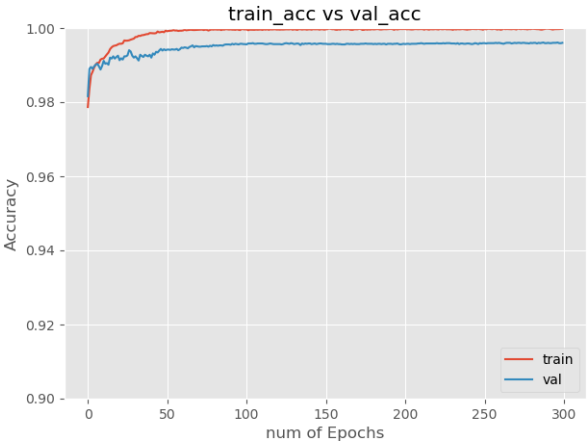

Figura 486. Accuracy de caffenet y cross con la combinación VTEEEE

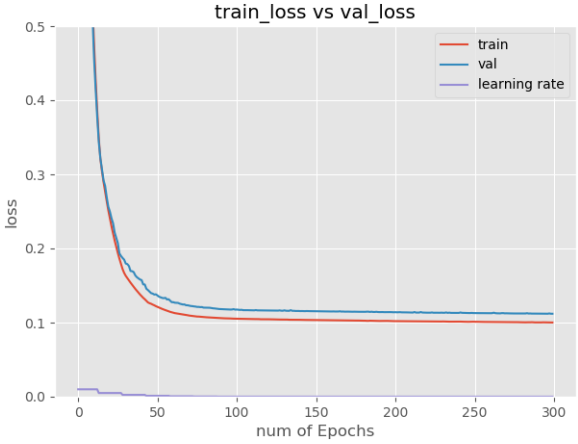

Figura 487. Loss de caffenet y cross testing con la combinación VTEEEE

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TPR	92.77	90.21	87.66	86.38	84.68	83.4	81.7	79.57	73.62
PPV	81.04	85.48	88.41	88.65	89.24	90.74	91.43	94.44	95.58

Figura 488. Umbrales de caffenet y cross testing con la combinación VTEEEE

	PPV	TPR	NPV	TNR	F1-Score	ACC
0.5	89.24	84.68	99.68	99.79	86.9	99.48

Figura 489. Métricas de caffenet y cross testing con la combinación VTEEEE

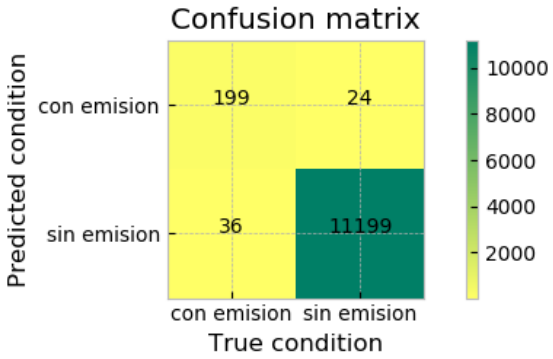

Figura 490. Matriz de confusión de caffenet y cross testing con la combinación VTEEEE

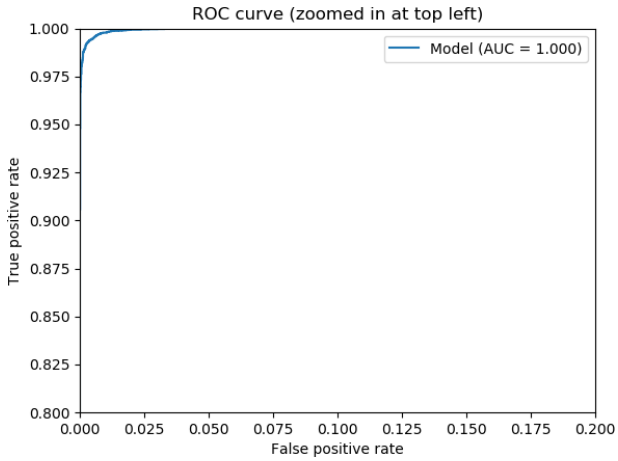

Figura 491. Curva ROC de caffenet y cross testing con la combinación VTEEEE