Clasificación automática de calidad embrionaria en embriones en estadio de Blastocisto, mediante una Convolutional Neural Network (CNN)

Ignacio Rodríguez García
Master Universitario en Bioinformática y bioestadística UOC-UB
Machine Learning

Consultor: Albert Pla Planas
Profesor responsable de la asignatura: Ferrán Prados Carrasco

5/06/2019

FICHA DEL TRABAJO FINAL

Título del trabajo:	Clasificación automática de calidad embrionaria en embriones en estadio de Blastocisto, mediante una Convolutional Neural Network (CNN)
Nombre del autor:	*Ignacio Rodríguez García*
Nombre del consultor/a:	*Albert Pla Planas*
Nombre del PRA:	*Ferrán Prados Carrasco*
Fecha de entrega (mm/aaaa):	06/2019
Titulación::	Master Universitario en Bioinformática y bioestadística UOC-UB
Área del Trabajo Final:	*Machine Learning*
Idioma del trabajo:	*Castellano*
Palabras clave	*Deep Learning, Convolutional Neural Network, Human Embryos*

Resumen del Trabajo.

Objetivos

El desarrollo de nuevas tecnologías en el campo de la inteligencia artificial debe ser aplicado en campos como la Reproducción Humana. La selección de los embriones a transferir es un proceso de vital importancia realizado por biólogos en base a características morfológicas. El objetivo de este estudio es desarrollar un algoritmo Deep Learning que permita clasificar las imágenes de embriones en función de su calidad.

Métodos

2000 imágenes provenientes de ciclos de reproducción del Insititut Universitari Dexeus fueron analizadas. Se utilizaron 1600 como conjunto de train y 400 de test en un diseño balanceado de embriones de buena y pobre calidad. Se utilizó Keras con Backend Tensorflow sobre Python para ajustar una red neuronal Convolucional. Para aumentar la diversidad de imágenes se utilizó Data Augmentation. Se evaluaron diferentes modelos de redes pre entrenadas para mejorar la utilidad del modelo permitiendo el aprendizaje de las últimas capas (Transfer Learning y Fine Tuning)

Resultados

El mejor modelo obtenido fue Xception Fine Tuning con una Accuracy del 67% en el conjunto de train y de 66% en el de test, precision de 67% en train y de 67% en test, y recall de 68% en train y 72% en test, denotando un sesgo a favor de los casos positivos.

Conclusiones

Los resultados obtenidos por el mejor modelo son pobres no permitiendo su implementación en la práctica diaria.

El pre procesado de imágenes y disponer de un número más elevado son necesarios para evaluar el potencial de estas tecnologías en el ámbito del laboratorio.

Abstract:

Objectives:
The development of new technologies in the field of artificial intelligence must be applied in areas as Human Reproduction. The selection of embryos to be transferred is a very important process carried out by embryologist based on morphological characteristics. The aim of this study is to develop a Deep Learning algorithm that allows the classification of embryo images according to their quality.

Methods:
2000 images from IVF cycles in the Insititut Universitari Dexeus were analyzed. 1600 were used as a set of train and 400 of test in a balanced design of embryos of good and poor quality. We used KERAS with Backend Tensorflow on Python to adjust a convolutional neuronal network. Data Augmentation was used to increase the diversity of images. Different models of pre-trained networks were evaluated to improve the usefulness of the model allowing the learning of the last layers (Transfer Learning and Fine Tuning)

Results:
The best model obtained was Xception Fine Tuning with an Accuracy of 67% in the train set and 66% in the test set, 67% precision in train and 68% in test, and 68% recall in train and 72 % in test, denoting a bias in favor of positive cases.

Conclusions:
The results obtained by the best model are poor, not allowing its implementation in daily practice. The pre-processing of images and having a higher number are necessary to evaluate the potential of these technologies in the field of the laboratory

Índice

Lista de figuras

Lista de Tablas

1. Introducción

1.1 Contexto y justificación del Trabajo

Descripción General.

El retraso en la edad de la maternidad es un problema acuciante en las sociedades modernas(1). Las técnicas de reproducción humana asistida (TRA) tratan, entre otras indicaciones, de mitigar el efecto negativo de ese retraso sobre la probabilidad de llevar una gestación a término. Un proceso de fecundación in vitro es una concatenación de pasos con un elevada complejidad técnica conducidos por un personal altamente cualificado(2–4). Entre ellos, la selección de los embriones a transferir a la paciente resulta de vital importancia(5). Tradionalmente los embriones eran cultivados en incubadores durante varios días, el personal de laboratorio evaluaba la evolución de los embriones, accediendo al incubador para extraer el embrión y seguidamente valorando distintos parámetros visualmente(3). El desarrollo de los incubadores con tecnología TimeLapse(6), ha permitido que ese proceso sea más ágil y menos perturbador para las condiciones de cultivo que necesitan los embriones. Estos incubadores disponen de una cámara que cada cierto tiempo obtiene una imagen que es valorada por el personal de laboratorio para clasificar la calidad del embrión.

El objetivo de este trabajo es desarrollar un algoritmo que permita utilizar la información de las imágenes captadas en el laboratorio por los incubadores Timelapse etiquetando de manera automática al embrión según su calidad.

Justificación del TFG

El desarrollo de las nuevas tecnologías enmarcadas en el concepto Inteligencia Artificial (IA)(7) deben ser evaluadas en diferentes contextos como los laboratorios de reproducción humana. Una rama de la IA son los algoritmos de Machine Learning (ML)(8), estos algoritmos pretenden realizar predicciones sobre los datos en dos direcciones, la resolución de problemas de clasificación o la de predicción numérica. Un subconjunto de los algoritmos ML son los algoritmos Deep Learning (DL)(9), basados en redes neuronales artificiales. Este tipo de procedimientos simulan el funcionamiento de las neuronas del cerebro humano. El termino deep hace referencia al uso de cientos o miles de neuronas. Las redes neuronales convolucionales, (Convolutional Neural Network) CNN, son un tipo de algoritmo Deep Learning especializado en la clasificación de imágenes(10).

Dado que se dispone de un conjunto de embriones etiquetados, se pretende evaluar si está metodología es válida en la clasificación de embriones. La

obtención de un algoritmo preciso permitiría su implementación en el software de los incubadores Timelapse facilitando el trabajo de laboratorio.

1.2 Objetivos del Trabajo

Objetivos generales.

1. Desarrollar un algoritmo DL que permita clasificar embriones en estadio de Blastocito según los criterios de calidad de ASEBIR(11) de manera automática, a partir de las imágenes generadas por los incubadores Timelapse.

Objetivos específicos.

Para resolver los problemas de clasificación descritos en los objetivos generales, se deben alcanzar los siguientes objetivos específicos:

1. Recolección de un datasets de imágenes con la suficiente diversidad y volumen para entrenar una CNN. Se accederá a la base de datos del Departamento de Obstetricia, Ginecología y reproducción del Hospital Universitari Dexeus para obtener las imágenes de los embriones junto con sus etiquetas correspondientes.

2. Diseño y entrenamiento de una CNN que mejore el estado del arte en términos de exactitud y precisión

Determinar si técnicas como Data Augmentation(12) o Transfer Learning (12) pueden mejorar la precisión y la exactitud de la CNN diseñada.

1.3 Enfoque y método seguido

La utilización de CNN es el algoritmo de referencia para problemas de clasificación de imágenes.
Se ha elegido la aplicación de una CNN sobre otros modelos más sencillos como la utilización de un simple multi-layer perceptron model, porque son la tipología de redes que ofrecen mejores resultados para la clasificación de imágenes, además soportan la implementación de Transfer learning y Data Augmentation (12,13). Transfer Learning permite la implementación de neuronas ya entrenadas con cientos de miles de imágenes en nuestro modelo, mientras que Data Augmentation permite la modificación de las imágenes, girarlas, centrarlas, normalizarlas, etc., que hacen que el modelo sea más robusto en la clasificación de imágenes antes no vistas.
Para desarrollar el modelo de clasificación se implementará un entorno de trabajo en Python. Se ajustará el modelo utilizando el paquete Keras(14) con Tensorflow(15,16) como backend.
.

1.4 Planificación del Trabajo

Se desglosan las tareas y el calendario de ejecución de los mismas:

Tareas

Definición contenidos del trabajo.
Plan de trabajo: Elaboración del documento actual.
Elaboración de la memoria: Es una tarea transversal a todo el desarrollo del proyecto.
Búsqueda bibliográfica: A partir de los documentos técnicos facilitados por el consultor, se debe obtener información técnica donde se detalle como programar la red y como configurar el entorno de trabajo.
Trabajo preliminar:
Configuración entorno de trabajo.
Consiste en implementar el software que permita ejecutar el modelo CNN.
Instalar Python, y sus paquetes complementarios.
Ejecutar ejemplos para asegurar que funciona correctamente.
Preparación de DatSets.
Descargar las imágenes y almacenarlas con sus correspondientes etiquetas.
Preprocesado de imágenes si es necesario.
Data Augmentation
Validación del entorno de trabajo.
Ajuste del Modelo.

Diseño de la red.
Ejecución de los primeros experimentos.
Fine Tuning y Transfer Learning

Preparación FASE I. Preparación de la primera entrega del trabajo.
Preparación Fase II. Presentación de la segunda entrega con el modelo finalizado.
Cierre de Memoria. Cierre, revisión y presentación de la memoria.
Elaboración de la presentación.
Defensa Pública. Presentación del trabajo

Se consideran tareas prioritarias del proyecto las relativas a la configuración del entorno de trabajo, preparación de los datasets y ajuste del modelo inicial

Tabla 1. Resumen de Actividades

Tarea	Fecha Inicio	Duración Días	Fecha Fin
Definición Contenidos de trabajo	20-feb.	12	4-mar.
Plan de Trabajo	5-mar.	13	18-mar.
Elaboración de la memoria	19-mar.	78	5-jun.
Búsqueda Bibliográfica	20-feb.	61	22-abr.
Configuración entorno de trabajo	5-mar.	28	2-abr.
Preparación DataSets	5-mar.	28	2-abr.
Ajuste Modelo CNN	3-abr.	15	18-abr.
Preparación FASE I	8-abr.	16	24-abr.
Validación y Mejora Modelo	25-abr.	15	10-may.
Preparación Fase II	11-may.	9	20-may.
Cierre Memoria	21-may.	15	5-jun.
Elaboración Presentación	6-jun.	7	13-jun.
Defensa Pública	14-jun.	12	26-jun.

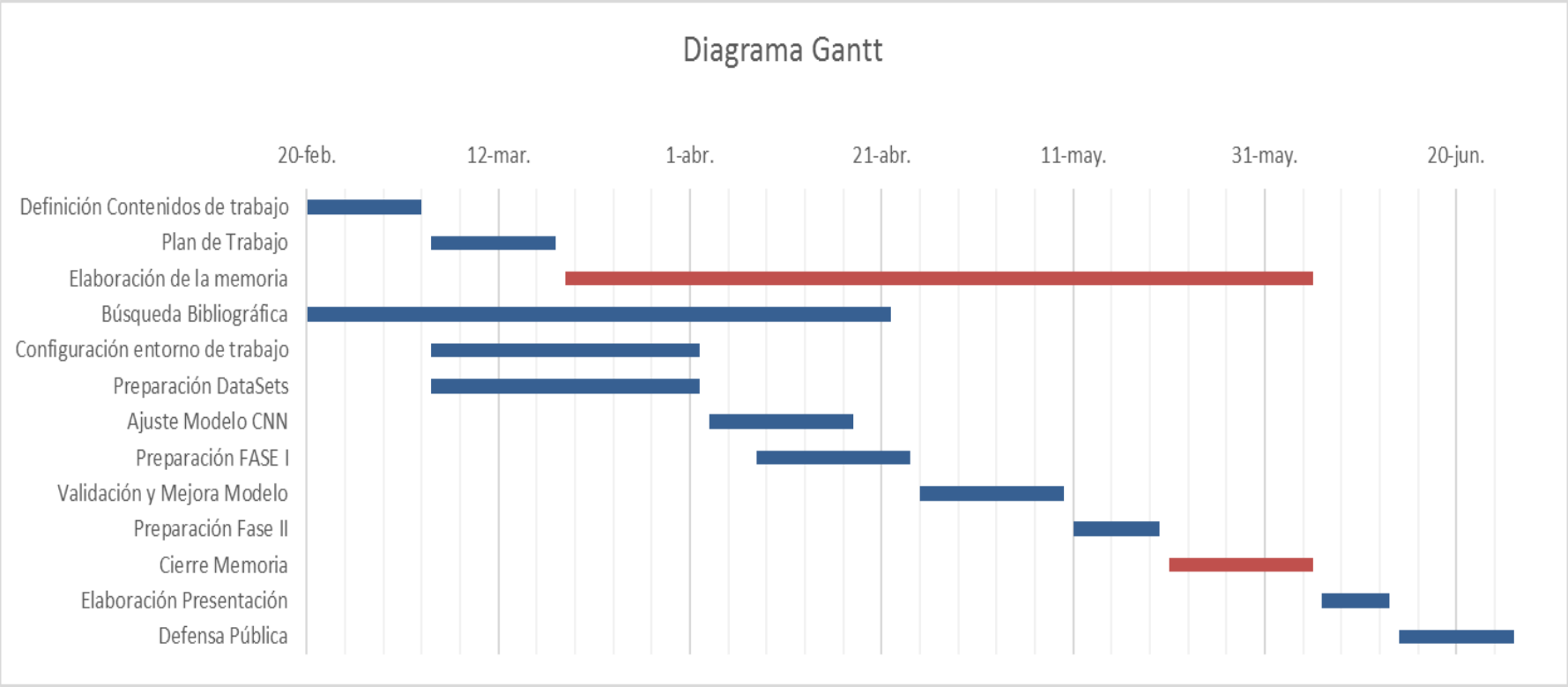


Ilustración 1. Diagrama de Gantt

1.5 Breve sumario de productos obtenidos

El objetivo del estudio ha sido la creación de un algoritmo par la clasificación de las imágenes de manera automática.

Para alcanzarlo se han realizado diversas tareas. En primer lugar, se han localizado las imágenes en la base de datos del Servicio de Reproducción. Mediante una consulta SQL se ha accedido a la base de datos y se han localizado las rutas de la imagen de cada embrión en su servidor correspondiente. Una vez localizadas las imágenes se ha procedido a descárgalas, para ello se ha ejecutado un script en R donde se descargaba cada imagen y se guardaba en una carpeta correspondiente a calidad de la misma. Una vez descargadas se han revisado una por una para eliminar imágenes erróneas o en formatos no apropiados.

Con las imágenes descargadas se ha configurado el entorno de trabajo. Para el cálculo del modelo se ha elegido Keras(17) por su versatilidad y facilidad de aplicación. Tras probar varias opciones como Keras en R en Windows o Keras en Ubuntu, se ha trabajado con la aplicación Colaboratory que permite ejecutar scripts de Python sobre notebooks de Jupyter(18) con acceso a GPU.

Se han ejecutado diversas redes de menor a mayor complejidad. Dado el limitado numero de imágenes, calcular una red propia no era una opción realista. Se ha optado por utilizar varias redes con pesos pre entrenados.

Se presentan los resultados de exactitud para cada tipo de red y mejora del modelo.

Ninguno de los modelos obtenidos alcanza el estado del arte para la clasificación de las redes convolucionales(19), son necesarios estudios más amplios sobre este tipo de imágenes para implementar su uso rutinario como herramienta de ayuda en el laboratorio.

1.6 Breve descripción de los otros capítulos de la memoria

La estructura de la memoria ha seguido los siguientes capítulos:

- Deep Learning. Contexto, definición y fundamentos.
- FIV/ICSI. Contexto y definición
- Entorno de trabajo. Configuración del entorno de trabajo. Software utilizado y descarga de imágenes.
- CNN. Ajuste de la red y mejora de los modelos. Presentación de resultados.

2. Deep Learning

En los últimos años una gran cantidad de tópicos relacionados con la Inteligencia Artificial (IA) han sido tratados por los medios de comunicación. Machine Learning (ML), Deep Learning (DL) y AI se han utilizado en multitud de artículos previendo un futuro con aplicaciones alejadas de la realidad o visionando un contexto en el que las máquinas realizaran tareas que aparten a los humanos de multitud de puestos de trabajo. Es necesario definir claramente el alcance de estas metodologías para alejarse de las visiones exageradas de los medios de comunicación. Su uso debe extenderse a todas las ramas de la actividad humana. De entre ellas las relacionadas con la salud toman una relevancia especial. En este estudio se aplican a la solución de un problema real. La clasificación de imágenes en un laboratorio de embriología.

2.1 Inteligencia Artificial, Machine Learning y Deep Learning

2.1.1 Inteligencia Artificial

La IA nació en la década de los 50, con el objetivo de evaluar si las maquinas podían realizar tareas intelectuales que normalmente eran realizadas por los humanos. AI es un campo que engloba ML y DL pero también otras tareas que no llevan implícita ninguna tarea de aprendizaje como programas de juegos de ajedrez. Al comienzo de IA se consideraba que podían abordarse problemas programando conjuntos suficientemente grandes de reglas, para abordar todas las posibilidades de un problema, esta aproximación se denomina AI Simbólica y ha estado en uso hasta los años 80. Al aparecer nuevos desafíos en problemas como clasificación de imágenes, reconocimientos de textos o traducciones automáticas ha aparecido una nueva aproximación de la IA conocida como Machine Learning(14,20)

2.1.2 Machine Learning

En un contexto clásico de programación, se programan reglas y los datos son procesados de acuerdo a esas reglas. Esas reglas pueden ser aplicadas a nuevos datos para producir respuestas. En un algoritmo de ML el sistema es entrenado sobre un conjunto de datos en lugar de ser programado. Una gran cantidad de ejemplos son pasados al algoritmo que por sí mismo identifica reglas subyacentes en la estructura de los datos que permiten automatizar tareas.(20)

Un ejemplo clásico de aplicación de ML es la clasificación identificación de correos con Spam.

Para utilizar el algoritmo se debe partir de un conjunto de datos, en este caso los correos, que deben estar clasificados (etiquetados) o bien como correos

normales o como Spam. Además, necesitaremos una manera de medir si el algoritmo está clasificando correctamente los correos. Se debe medir la diferencia entre la clasificación del algoritmo y la realidad. Esta medida se utiliza como feedback para evaluar si el algoritmo funciona. Este feedback se hace de manera iterativa y es lo que se denomina aprendizaje o learning.

Machine Learning está dividido en tres grandes categorías, se conoce como aprendizaje supervisado aquel en que los elementos a clasificar están etiquetados previamente. Los algoritmos más utilizados so la regresión lineal y logística, Naive Bayes, Support vector Machine (SVM), árboles de decisión y redes neuronales.

Cuando el problema trata con datos no etiquetados, nos referimos a aprendizaje no supervisado. El algoritmo intentará clasificar la información por sí mismo. Los algoritmos más utilizados son Clustering (K-medias) o el análisis de componentes principales (PCA).

Reinforcement Learning ("aprendizaje de refuerzo"), hace referencia al algoritmo en que un agente deberá implementar acciones mediante prueba y error, tomando las decisiones en función de las recompensas y penalizaciones que recibe. Una aplicación de RL sería en la implementación de un sistema dinámico de señales de tráfico y semáforos.

El funcionamiento tipo de un problema con ML consiste en dos fases.

En una primera fase se realiza un entrenamiento del modelo sobre una parte de los datos obtenidos (por ejemplo 75%) llamada fase de train, posteriormente se aplica el modelo sobre el resto de los datos (por ejemplo 25%) para evaluar la eficacia del mismo, esta fase es conocida como fase de test.

Red neuronal artificial

Una Red neuronal artificial (RNA) es un tipo de algoritmo de ML supervisado.

Una RNA modela la relación entre un conjunto de señales de entrada y unas señales de salida utilizando un modelo que simula como un cerebro humano responde a impulsos sensoriales. Se denomina método de caja negra porque el mecanismo que transforma la información de entrada en información de salida se asemeja a una caja negra que proporciona resultados sin comprender muy bien el mecanismo que hay detrás

Tal y como un cerebro, utiliza una red de células interconectadas llamadas neuronas para crear un procesado masivo en paralelo, una RNA utiliza una red de neuronas artificiales o nodos para resolver problemas de Learning.(20)

El cerebro humano contiene alrededor de 85000 millones de neuronas, resultando en una red capaz de representar un enorme conjunto de conocimiento.

En contraste, una RNA contiene unos pocos cientos de neuronas.

Para entender cómo funciona un RNA debemos aproximarnos a como trabaja un cerebro humano. Las señales entrantes son recibidas por las dendritas de las células a través de un proceso bioquímico. El proceso permite que el impulso sea ponderado de acuerdo a la intensidad, importancia y frecuencia de la señal entrante. La célula va acumulando la señal entrante hasta que un umbral es alcanzado y la célula transmite la señal mediante un proceso electromagnético hacia un axón. En las terminales del axón esta señal eléctrica es de nuevo procesada como una señal química para ser enviada a otra neurona vecina en lo que se conoce como sinapsis.(20)

Una RNA se puede interpretar como el modelo biológico anterior. Se definen las relaciones entre las señales entrantes recibidas por las dendritas (X_i) y la señal de salida (y). Como en la neurona biológica, cada señal de la dendrita está ponderada (w_i) de acuerdo a su importancia. Las señales entrantes son acumuladas por la célula y es transmitida mediante una función de activación (f(x))

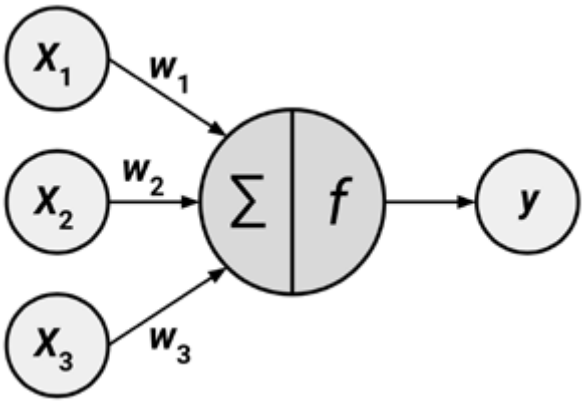


Ilustración 2.Modelo Neurona(20)

Una clásica red neuronal artificial con n dendritas puede ser representada por la formula siguiente:

$$y(x) = f\left(\sum_{i=1}^{n} w_i x_i\right)$$

Ilustración 3. Expresión matemática(20)

W permite a cada entrada a contribuir en mayor o menor medida a la suma de todas las señales. La suma resultante es utilizada por la función de activación f(x) . La señal resultante y(x) es el axón de salida.

Las redes neuronales utilizan conjuntos de neuronas definidas de esta manera para tratar con conjuntos de datos de naturaleza compleja.

Se deben definir:(20)

Función de activación que transforme a las señales entrantes en una única señal de salida para ser distribuida al resto de la red.

Topología o Arquitectura que describe el número de neuronas en el modelo, el número de capas y la manera en que están conectadas.

Un algoritmo de entrenamiento que permita calcular los pesos para activar o desactivar una determinada neurona de acuerdo a la señal de entrada.

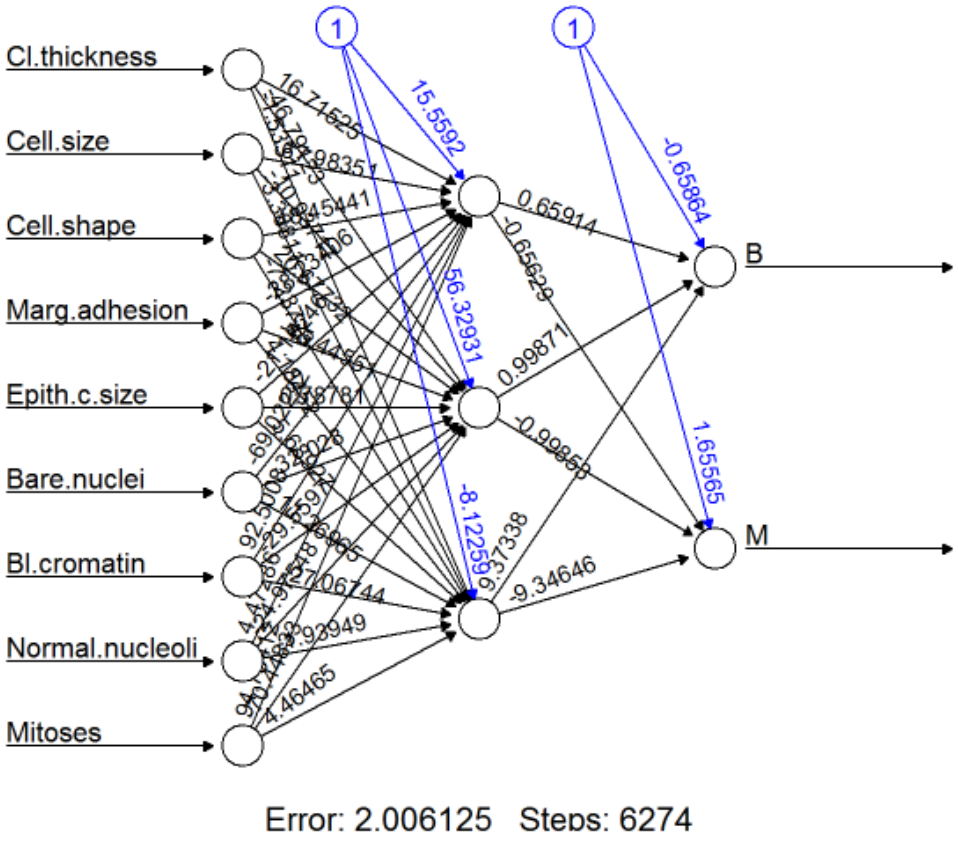


Ilustración 4. Arquitectura RNA.

Tabla 2. Fortalezas y debilidades RNA(20)

Fortalezas	Debilidades
Pueden ser utilizadas en problemas numéricos o de clasificación	Computacionalmente costosas a medida que la arquitectura es compleja
Capaces de modelizar patrones muy complejos	Tendente al overfitting
No necesitan suposiciones sobre las relaciones de los datos	Modelo de caja negra con resultados casi imposible de interpretar

2.1.3 Deep Learning

Los algoritmos Deep Learning son un subconjunto de las redes neuronales. El término Deep hace referencia a la complejidad de las capas y número de neuronas en el modelo. Mientras que los algoritmos las redes neuronales tienden a tratar con dos o tres capas, los algoritmos DL tratan con varias capas cada vez de mayor complejidad.

A partir de una entrada de datos inicial, el algoritmo va extrayendo características en varias etapas hasta presentar la representación final.

Por ejemplo, en una tarea de reconocimiento de imágenes de animales, las entradas son imágenes y la salida es la etiqueta de cada imagen (gato, perro, etc.). A partir de la imagen de entrada en una sucesión de varias capas se van realizando transformaciones de los datos que devienen en una predicción final. Esa predicción del modelo se compara con la etiqueta original de la imagen en la realidad, mediante una función de perdida. El objetivo es ajustar los pesos que minimicen esta función de perdida.

Cada una de las neuronas participantes en la red dispone de su propia función de activación que se encarga de transmitir a los siguientes nodos la información una vez alcanzado una cierta condición.

Fases de entrenamiento de una RNA

Forward Propagation

Es el proceso por el que cada dato de entrenamiento es lanzado a través de la red. Cada una de las neuronas aplica su trasformación pertinente y pasa la información a la capa siguiente. Al final de la red, en la capa final se realiza la predicción o etiquetado correspondiente.

Estimación del error (Loss)

Para cada caso se estima el error o loss, el objetivo es comparar el resultado predicho por la red con la realidad, con el interés de obtener un valor cero.

Backpropagation

Una vez calculado el loss para cada caso, esta información es propagada hacia atrás en la red por todas las capas y neuronas. Cada neurona recibe una proporción de información en función de su aportación relativa, de esta manera el loss total es repartido entre todas las neuronas de la red.

Con la información propagada hacia atrás se realiza el ajuste de los pesos de cada neurona (wi) con el fin de minimizar la función de loss o error. Para ello se utiliza una técnica denominada Gradient Descend que consiste en calcular la derivada de la función de loss para obtener el valor que la minimice (mínimo Local), este proceso se realiza por lotes en sucesivas iteraciones.

Flujo Algoritmo aprendizaje:(21)

- Iniciar los pesos de la red de manera aleatoria.
- Pasar un conjunto de datos por la red y obtener su predicción
- Calcular el error (Función Loss)
- Mediante la Backpropagation, informar a cada neurona de su loss correspondiente
- Actualizar los pesos mediante Gradiente descend (derivativamente) para minimizar el loss
- Repetir hasta que el modelo aprenda.

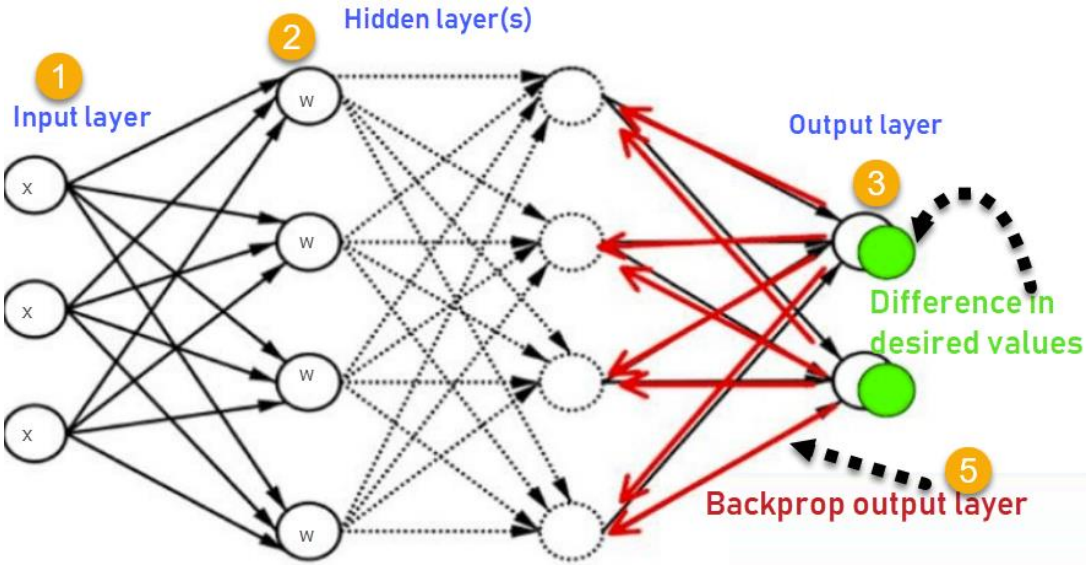
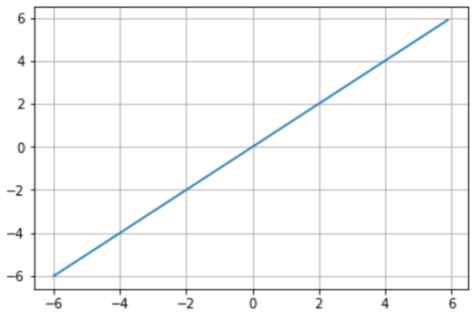
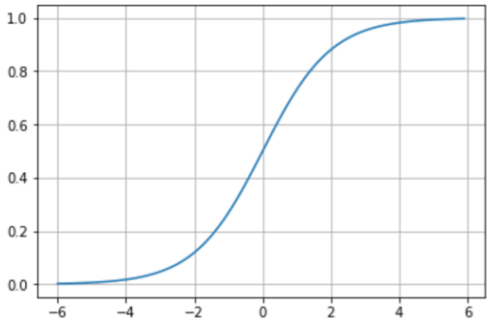
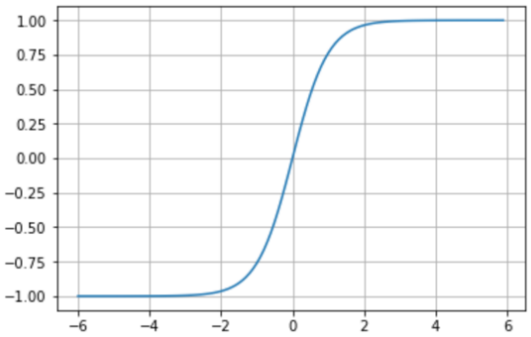
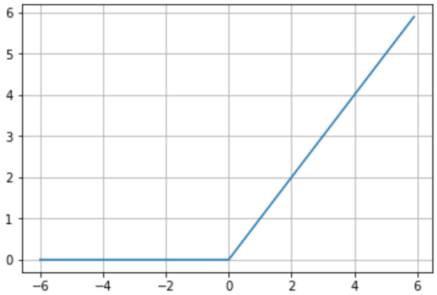


Ilustración 5. Backpropagation(22)

Funciones de Activación

Tabla 3: Funciones activación (19)

Linear: Función identidad	
Sigmoid: Transforma los valores en el rango 0-1	
Tanh: Transforma los valores en el rango --1,1	
ReLU: Transforma los valores positivos en la identidad y los negativos a valor cero	

Función Loss

Existen varias posibilidades en función del tipo de datos:

Tabla 4: Funciones pérdida(20)

Problema	Loss	Capa Activación
Clasificación Binaria	Binary_crossentropy	Sigmoide
Clasificación categórica simple	Categorical_crossentropy	Softmax
Clasificación categórica múltiple	Binary_crossentropy	Sigmoide
Regresión	Mse	Ninguna
Regresión entre 0 y 1	Mse o Binary_crossentropy	Sigmoide

Optimizadores.

Son los algoritmos encargados de encontrar el mínimo local de la función de loss. Existen varias opciones SGD, RMSprop, ADAM, etc.
El funcionamiento es localizar la primera derivada de la función de loss que minimiza su valor para un determinado peso del modelo, para ello modifica el valor del parámetro en sentido contrario al gradiente, añadiendo una cantidad. Este cambio está determinado por el valor del gradiente y un parámetro denominado Learning Rate. Este proceso se repite hasta alcanzar el mínimo local.

Parámetros de una RNA.

El ajuste de una RNA conlleva la selección de diferentes parámetros e híper parámetros.

Epochs.

Número de veces que se pasan por la red todos los datos de train

Batch Size.

Son lotes en los que se parten los datos de train al pasarlos por la red con el fin de actualizar el gradiente.

Learning Rate.

Es el valor que actualiza el gradiente en el proceso de optimización. Se interpreta como la amplitud del paso al recorrer cada punto de la función loss.

Learning Rate Decay.

Es un modulador del Learning rate, a medida que aprende la red el learning rate decay disminuye el learning rate.

Momentum.

Es una constante entre 0 y 1 que se utiliza para ponderar los gradientes anteriores y evitar que el algoritmo seleccione un mínimo local erróneo.

Métricas.

Son las variables que monitorizan el performance del modelo. Ente ellas Accuracy, Precision, Recall y F1.Score

2.2 Red Neuronal Convolucional

Una red convolucional es un tipo especial de red neuronal. Presenta una configuración similar al resto de redes neuronales con una capa de entrada de datos, varias capas ocultas y una capa de salida final clasificatoria, todas ellas interconectadas por neuronas.
El funcionamiento radica en la capacidad de identificar características de una menor a mayor complejidad a medida que profundizamos en las distintas capas de la arquitectura.
En las primeras capas son capaces de identificar características como bordes, marcas, ángulos y a medida que aumentan las capas se aprenden características con mayor nivel de abstracción. Para realizar este proceso estos algoritmos disponen de dos operaciones fundamentales en sus capas, la convolución y el pooling.

Convolución

Cada imagen es transformada en un tensor de tres dimensiones, con altura y anchura el número de pixeles de la imagen, y como profundidad 3 si es en color o 1 si es en blanco y negro.
La operación de convolución consiste en recorrer secuencialmente sobre el tensor una ventana de tamaño 3x3 o 5x5, por lo general. Cada una de esas ventanas está conectada a una neurona. Se empieza por la ventana de la esquina superior izquierda y se desliza hacia la derecha saltando una posición hasta el final de la fila. Así sucesivamente hasta recorrer toda la imagen inicial. La longitud de paso se puede modificar (stride) o se pueden complementar con ceros rellenando las capas más exteriores (padding).

La conexión entre la capa de entrada (5x5) y la capa oculta se hará mediante un sesgo y matriz de pesos (5x5) llamado filtro o Kernel. El valor resultante de este producto se almacena en la siguiente capa oculta. El filtro y sesgo utilizado es el mismo para todas las ventanas de la capa de entrada, lo que implica una reducción computacional. Cada filtro permite identificar una característica por eso en cada capa se aplican varios filtros uno para cada característica, normalmente 32. (cada uno de estos filtros si tiene una matriz de pesos y sesgo diferente)

Pooling

Inmediatamente después de la capa convolucional se aplica una capa de pooling. El objetivo del pooling es hacer una simplificación de la capa original manteniendo la relación espacial, reduciendo la información en un punto de la capa. Este proceso de reducción se puede hacer mediante un promedio average-pooling, o mediante el máximo max-pooling.

La capa de pooling se aplicará a tantas capas como filtros tenga la capa convolucional.

Redes Pre entrenadas (Transfer Learning)

Las CNN requieren de gran cantidad de imágenes para estimar los pesos del modelo. Existen redes pre entrenadas a disposición de los usuarios, estas redes se han entrenado en grandes conjunto de datos (12,23). La idea es utilizar los pesos de las redes y ajustar la arquitectura de la red a nuestra capa de salida de datos. La utilización de TL permite ahorrarse la fase de aprendizaje de las características básicas de las imágenes, computacionalmente lo más costoso, además solo se entrenan las capas finales clasificatorias. Otra ventaja de la utilización de este tipo de redes es la reducción del problema del fading gradient descend y del overfitting. Existe la

posibilidad de dejar entrenar los pesos de alguna capar en particular (Fine Tuning) lo que permite una mejor adaptación a las imágenes del problema en particular.

3. Fecundación in Vitro e ICSI

3.1 Epidemiología de la Reproducción.

Se define la esterilidad como la incapacidad de conseguir una gestación después de haber tenido relaciones sexuales en un periodo de entre uno y dos años sin la utilización de métodos anticonceptivos.
La sociedad actual ha originado nuevas situaciones como la de mujeres sin pareja masculina o parejas homosexuales que estando sanas no pueden concebir por sí mismas, y necesitan del soporte clínico para alcanzar el objetivo.
El objetivo de las TRA (Técnicas de Reproducción humana Asistida) es ayudar tanto a las personas que presentan una patología como a aquellas que sin presentar dicha patología no están en condiciones de gestar por sí mismas.

3.1.1 Edad y Reproducción Humana.

La capacidad reproductora de la mujer se ve afectada de manera natural por la edad. A medida que la mujer envejece la cantidad y calidad de los folículos y ovocitos contenidos en sus ovarios desciende. Estos folículos y ovocitos son denominados reserva ovárica y se estima que pasan de 7000000 de folículos en la semana 20 de gestación a 0 en la menopausia.
En los varones la edad también afecta a la capacidad reproductiva, pero de una manera diferente. Aunque los varones son capaces de generar espermatozoides durante toda su vida, a medida que el varón hace años, sus espermatozoides son portadores de mayores anomalías cromosómicas, que hacen que sea más difícil la concepción o los productos de las mismas tengan mayor probabilidad de padecer cromosomopatías.

3.1.2 Incidencia y Prevalencia de la Esterilidad.

Es difícil estimar estos dos parámetros debido a que no todos los individuos viven en pareja o no mantienen relaciones sexuales con fines reproductivos.
Según datos del Registro SEF (24), en 2016 se realizaron en España 54.024 ciclos de reproducción para obtención de ovocitos propios y 16.133 ciclos re recepción de ovocitos de donante en fresco. A nivel global se estima que la infertilidad afecta a 186 millones de personas(25).

3.1.3 Etiología de la Esterilidad.

Las principales causas que incrementan el riesgo de esterilidad son las asociadas a la edad. Otros factores tienen relación como la enfermedad pélvica inflamatoria, los hábitos poco saludables, el tabaquismo y determinados factores ambientales.

Causas Esterilidad femenina:

- Trastornos de la ovulación
- Patología tubárica
- Endometriosis

Causa Fertilidad Masculina:

- Anomalías seminales.

3.2 Tratamiento de la Esterilidad

Se dividen en dos categorías en función de la complejidad de las mismas.

3.2.2 Tratamientos de baja complejidad en TRA.

Se encuentran la inducción de la ovulación (IO) y la inseminación artificial con semen de pareja (IAC) o con semen de donante (IAD).
La IO es un procedimiento mediante el cual, la mujer recibe un tratamiento farmacológico (Inductor de la ovulación) que hace que mejore su capacidad ovulatoria y se le pauten relaciones sexuales coitales dirigidas.
La IAC o IAD son dos procedimientos en los que se deposita el semen de forma no natural en el útero de la paciente mediante la utilización de un catéter. Previamente el semen ha sido manipulado con el fin de obtener una mayor concentración de espermatozoides móviles.
3.2.2 Tratamientos de alta complejidad en TRA.
Entre los tratamientos de mayor complejidad se encuentra la Fecundación In vitro/ICSI.

3.3 Fecundación in vitro e ICSI.

El hito más importante para el tratamiento de la fertilidad ha sido la fecundación in vitro. En 1978 se publica el nacimiento de la primera niña mediante la utilización de técnicas de FIV(26). Actualmente es el tratamiento generalizado para todas las causas de infertilidad tanto de origen femenino como masculino.
La inyección intracitoplasmatica de espermatozoides (ICSI) permite la selección de un único espermatozoide aparentemente sano y con él inseminar un ovocito.
La FIV se define como la técnica que permite la unión de los gametos femeninos y masculinos fuera del aparato reproductor femenino, en el laboratorio.
Para ello es necesario obtener los ovocitos de la mujer y disponer de una muestra de semen, poner en contacto ambos gametos, cultivar los embriones resultantes, y finalmente transferirlos a la paciente depositándolos en el interior de su útero.

En los últimos años, se ha incorporado como mejora de la FIV el diagnóstico genético preimplantacional (DGP). Mediante el DGP antes de la transferencia embrionaria se realiza un diagnóstico del embrión mediante técnicas genéticas complejas. Aquellos embriones que son portadores de anomalías cromosómicas o de patologías genéticas son descartados y sólo se utilizan para transferir los embriones normales.
Antes de realizar una FIV se debe evaluar el perfil de fertilidad de la pareja. Se debe analizar el perfil hormonal, el estado espermático y la receptividad uterina.

Para ello se realizan tres pruebas básicas.

- Analítica Hormonal
- Seminograma
- Ecografía ginecológica

3. 3.1 Protocolos de Estimulación

En los comienzos de la FIV los embarazos obtenidos fueron en ciclo natural.
En un ciclo natural, una mujer puede ovular normalmente entre uno y dos ovocitos. La eficacia de la fecundación in vitro con este bajo número de ovocitos es muy limitada, por lo que se incorporaron tratamientos para obtener más ovocitos.
El objetivo de los protocolos de estimulación es obtener un número suficiente de ovocitos, superior a los generados de manera natural por la paciente en ciclo natural, y que permitan aumentar la eficacia de la FIV. Obtener un mayor número de ovocitos permite obtener mayor número de embriones. Los mejores embriones son transferidos en primera instancia y los sobrantes son congelados para poder ser utilizados en caso de fallo del primer intento o de deseo de la paciente para aumentar su fecundidad.
Actualmente el tratamiento de elección es la utilización de un Antagonista de la GnRH junto con administración exógena de gonadotrofinas. El antagonista de la GnRH se encarga de la inhibición de la hipófisis para impedir la ovulación y luteinización prematura, mientras que las gonadotrofinas permiten reclutar e iniciar el desarrollo del mayor número de folículos y que la maduración de los mismos sea óptima.
La duración del tratamiento dependerá de cada paciente. Dicha respuesta será controlada cada uno o dos días mediante un seguimiento

3.3.2 Monitorización del ciclo.

El objetivo de la monitorización del ciclo permite dosificar las unidades de gonadotrofinas y así conseguir una buena respuesta folicular, y controlar que no haya un excesivo número de folículos que podrían llevar a un efecto indeseado de hiperestimulación ovárica.

Una vez evaluado que la paciente tiene un número y tamaño folicular óptimo se administra la hormona hCG, está hormona activa el procedimiento de la maduración folicular final y permite el datado de la punción folicular, normalmente entre 34 a 36 horas tras la administración de la hCG

3.3.3 Recuperación de los ovocitos

La recuperación de los ovocitos se realiza mediante punción guiada por ecografía. Es una técnica quirúrgica que consiste en puncionar el ovario y vaciar el contenido folicular, se realiza con sedación anestésica. El cirujano atraviesa la pared vaginal con una aguja hasta llegar al ovario, una vez en él aspira uno a uno los folículos. Este procedimiento se realiza para cada ovario.

3.3.4 Laboratorio de FIV

El líquido obtenido en la punción es enviado al laboratorio de FIV. Los ovocitos son identificados mediante un microscopio, son extraídos del líquido folicular y colocados en un incubador con un medio de cultivo hasta la inseminación.

Existen dos técnicas de inseminación de los ovocitos, la inseminación convencional y la ICSI.
La inseminación convencional consiste en poner en contacto cada ovocito con unos 100000 espermatozoides de buena movilidad. La ICSI consiste en la inyección de un único espermatozoide en el interior del citoplasma del ovocito.
Una vez inseminados los ovocitos son guardados junto con un medio de cultivo en un incubador regulando las condiciones de humedad, temperatura y CO_2 y O_2.
Al día siguiente de la inseminación se comprueba si los ovocitos han sido fecundados. Aquellos que han sido fecundados se dejan en cultivo entre tres y cinco días. Regularmente son evaluados para valorar la evolución, división celular con el fin de identificar los embriones perfectamente evolutivos.

3.2.5 Transferencia Embrionaria.

Es la última fase de la FIV.
Consiste en la selección de los embriones óptimos de entre los disponibles de la paciente y la colocación de los mismos en el interior de útero de la paciente.
La selección del embrión a transferir se hace mediante criterios morfológicos, el embriologo evalúa cada embrión y en función de una serie de características selección a los mejores para transferir.
El objetivo del presente estudio es evaluar si un algoritmo de Deep Learning permite clasificar de manera automática a los embriones a transferir en función de sus imágenes.

3.3 Selección Embrionaria.

El proceso de selección embrionario es aquel por el que se selecciona al embrión o embriones que van a ser depositados en el útero de la mujer.
Tradicionalmente la selección del embrión se lleva a cabo bajo criterios morfológicos basados en estudios retrospectivos que han demostrado asociación entre ciertas características observables en los embriones y la implantación del embrión.(5,11).
En los últimos años han aparecido sistemas de cultivo embrionario con cámaras que permiten la monitorización de los embriones de manera secuencial(27), añadiendo a las clasificaciones morfológicas parámetros morfocinéticos. Otra ventaja es la disponibilidad de las imágenes.

Ilustración 6. Sistema Time Lapse

Para el estudio que nos ocupa se han utilizado los embriones etiquetados bajo el criterio ASEBIR(11). Los embriones son clasificados en función de la masa celular interna y el trofoectodermo. Para el entrenamiento de la CNN se han recodificado los embriones en buena calidad (A+B) y pobre calidad (C+D)

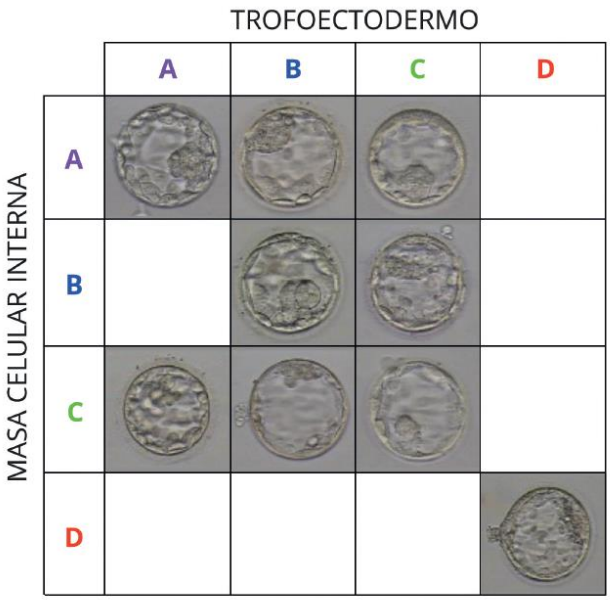


Imagen 50. Clasificación ASEBIR de los blastocistos en función de la calidad de la MCI y el TE.

Ilustración 7. ASEBIR Criterios Morfológicos

4. Configuración Entorno de trabajo

Se ha configurado el entorno de trabajo en diferentes configuraciones:

Windows: Bajo una distribución ANACONDA se instaló Python y los paquetes necesarios para configurar la red (Tensorflow-Keras). Se descartó este escenario al dar muchos problemas y no ser estable. Las pruebas no corrían o daban problemas. Se decidió montar el entorno sobre UBUNTU.

Ubuntu: Se instaló una maquina con UBUNTU 18. Se ha instalado Python y todos los paquetes detallados anteriormente junto con Jupyter Notebook.

Se han ejecutado diferentes pruebas sin problemas (MNIST / Cat & Dogs) descritos en Deep learning with Python. (19)

Dado que el conjunto de imágenes es limitado es necesario implementar Transfer Learning (utilizar los pesos de una red pre entrenada para clasificar los embriones). El entorno de trabajo de Ubuntu sin GPU no es el adecuado para este propósito por lo que se han valorado otros escenarios.

Se ha configurado una máquina virtual en google cloud para poder ejecutar pruebas más complejas.

Adicionalmente se ha utilizado Colaboratory (https://colab.researc.google.com), iniciativa de google de libre acceso que permite ejecutar Jupyter Notebooks sobre Python en un entorno con GPU.

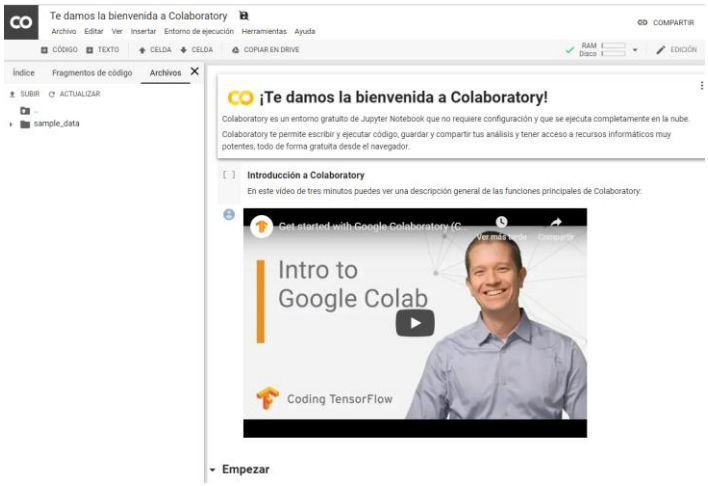


Ilustración 8. Colaboratory

4.1 Preparación de Data-Sets.

Se ha programado una consulta SQL sobre la base de datos del departamento (ver anexo). La consulta localiza los embriones de los ciclos de reproducción en el programa EMBRYO.

.

Results | Messages

	fila	numero	ovocito	proceso	secuencia	newref	scoreasebir	hora	referencia	fichero	fecha	dia
1	1	2161290	304309	2010005431	3	3	C	112	3_112	datos9\1995008379\2010005431OET20180118142646_3_...	2014-01-04 10:38:59.737	5
2	1	1914723	370386	2011000722	4	4	A	112	4_112	datos2\2010002212\2011000722OET20170625062235_4_...	2017-06-25 12:47:56.397	5
3	1	1914724	370387	2011000722	5	5	B	112	5_112	datos2\2010002212\2011000722OET20170625062245_5_...	2017-06-25 14:02:57.363	5
4	1	2160544	261077	2011002599	7	7	A	112	7_112	datos1\2011016101\2011002599OET20180118142206_7_...	2011-11-29 13:57:26.023	5
5	1	1530653	340019	2011003165	4	4	B	112	4_112	datos7\2011017857\2011003165OET20160119042222_4_...	2016-01-19 14:02:56.143	5
6	1	1530654	340020	2011003165	5	5	C	112	5_112	datos7\2011017857\2011003165OET20160119042229_5_...	2016-01-19 14:02:56.833	5
7	1	1530655	340021	2011003165	6	6	B	112	6_112	datos7\2011017857\2011003165OET20160119042242_6_...	2016-01-19 14:02:57.240	5
8	1	5355999	396903	2011003793	5	5	B	112	5_112	datos6\2011015116\2011003793OET20180826073224_5_...	2018-08-25 09:34:02.030	5
9	1	5356012	396904	2011003793	6	6	B	112	6_112	datos6\2011015116\2011003793OET20180826073225_6_...	2018-08-25 09:34:02.097	5
10	1	5356013	396905	2011003793	7	7	B	112	7_112	datos6\2011015116\2011003793OET20180826073226_7_...	2018-08-25 09:34:02.163	5
11	1	5356015	396908	2011003793	10	10	B	112	10_112	datos6\2011015116\2011003793OET20180826073230_10...	2018-08-25 09:34:02.230	5
12	1	5356016	396910	2011003793	12	12	B	112	12_112	datos6\2011015116\2011003793OET20180826073232_12...	2018-08-25 09:34:02.297	5
13	1	5356017	396912	2011003793	14	14	B	112	14_112	datos6\2011015116\2011003793OET20180826073234_14...	2018-08-25 09:34:02.363	5
14	1	2155762	273353	2012000425	7	7	A	112	7_112	datos4\2006007644\2012000425OET20180118135033_7_...	2012-05-09 12:30:12.000	5
15	1	908191	270255	2012001571	2	2	B	112	2_112	datos2\2009039952\2012001571OET20130101074317_2_...	2013-01-08 17:12:43.467	5
16	1	2154714	276150	2012001583	4	4	B	112	4_112	datos4\2010022394\2012001583OET20180118133945_4_...	2012-06-29 12:07:41.147	5
17	1	847288	277990	2012002121	4	4	B	112	4_112	datos4\2011023234\2012002121OET20120803050003_4_...	2012-08-03 11:09:43.540	5

Ilustración 9. Resultados Consulta SQL

La consulta devuelve la localización de las imágenes de los embriones transferidos o congelados en la ventana de tiempo que nos interesa entre 96 y 120 h, correspondiente al estadio de Blastocisto.

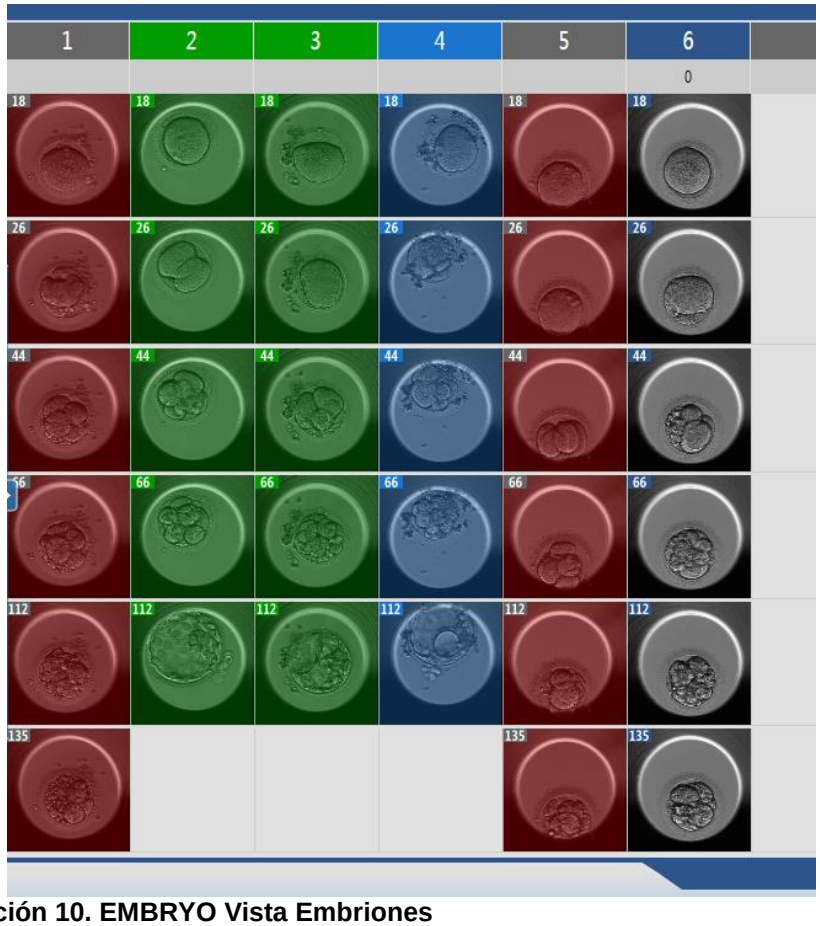


Ilustración 10. EMBRYO Vista Embriones

Una vez programada la consulta se ejecutado un programa en R que lee los datos de la consulta y guarda las imágenes en su carpeta correspondiente. Para ello se ha programado una función que descarga la imagen del servidor y guarda cada imagen según la calidad de la misma en una carpeta good o poor.

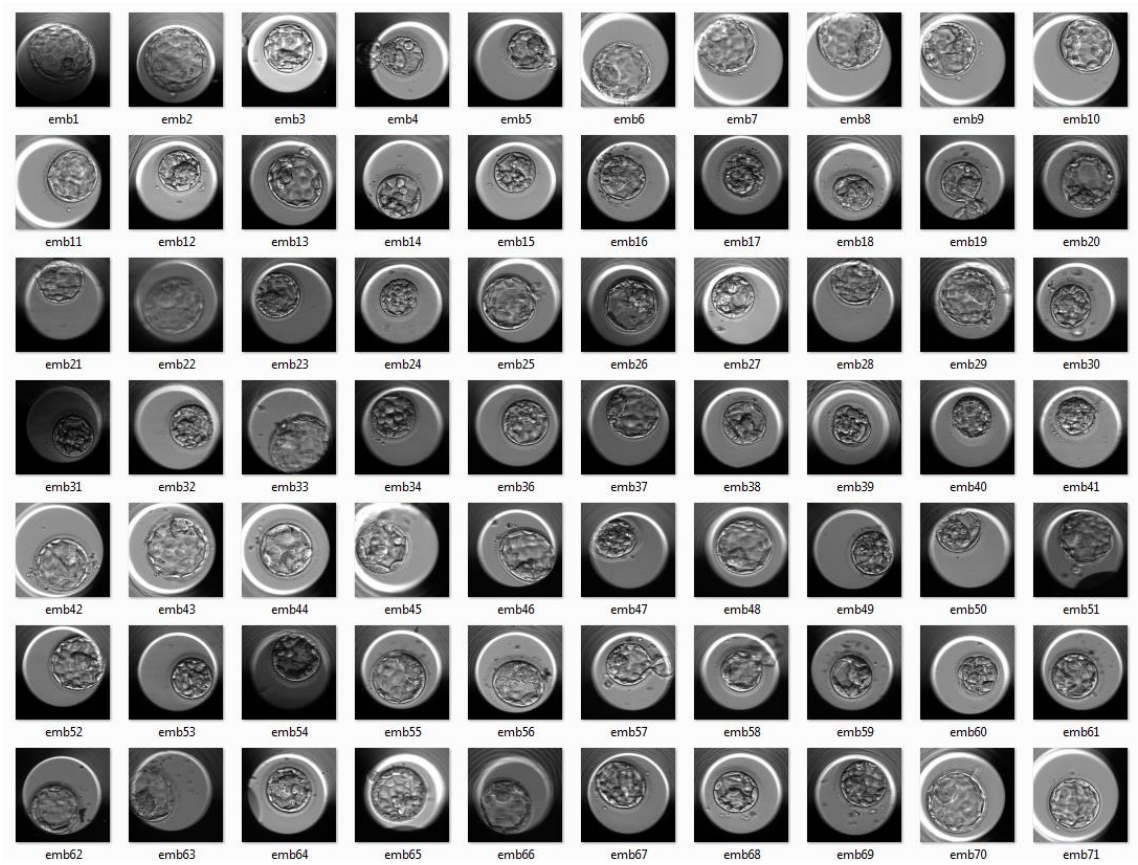


Ilustración 11. Vista Embriones descargados

5. CNN

Se han realizado varias pruebas comprobando con distintos formatos de imágenes. La imagen original es de 500x500, se ha probado reduciendo el tamaño de la misma 250x250 ofreciendo resultados en cuanto a clasificación similares. Finalmente se ha utlizado un tamaño reducido para agilizar el tiempo de ejecución.

Dado que es un problema con dos categorías los grupos en el conjunto de train y de test deben estar balanceados, se presentan las pruebas sobre un subconjunto de 1600 imágenes de train y 400 de test seleccionadas al azar, con igual número de embriones good y poor en cada conjunto.

En esta primera aproximación se han programado tres redes de menor a mayor complejidad. Se ha elegido como métrica para la validación de la clasificación la precisión (Accuracy), con la función de perdida loss='binary_crossentropy'y optimizador RMSprop(lr=1e-4) con un learning rate de 0.0001. La función de activación tras cada capa convolucional ha sido relu, la función de activación de la capa clasificatoria final ha sido sigmoide

La carga de imágenes se ha realizado con la función low_from_directory, en esta primara red no se ha realizado Data Augmentation, tan solo se han normalizado las imágenes.

In [7]:
```
from PIL import ImageFile
ImageFile.LOAD_TRUNCATED_IMAGES = True
train_dir='./balan800/train'
validation_dir='./balan800/test'
from keras.preprocessing.image import ImageDataGenerator

# All images will be rescaled by 1./255
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        # This is the target directory
        train_dir,
        # All images will be resized to 250x250
        target_size=(250, 250),
        batch_size=20,
        # Since we use binary_crossentropy loss, we need binary labels
        class_mode='binary')

validation_generator = test_datagen.flow_from_directory(
        validation_dir,
        target_size=(250, 250),
        batch_size=20,
        class_mode='binary')
```

```
Found 1600 images belonging to 2 classes.
Found 400 images belonging to 2 classes.
```

Ilustración 12. Carga de Imágenes.

5.1 Red Inicial

Se presenta una red con cuatro capas convolucionales con función de activación relu seguidas de un pooling.

```
from keras import layers
from keras import models

model1 = models.Sequential()
model1.add(layers.Conv2D(32, (3, 3), activation='relu',
                         input_shape=(250, 250, 3)))
model1.add(layers.MaxPooling2D((2, 2)))
model1.add(layers.Conv2D(64, (3, 3), activation='relu'))
model1.add(layers.MaxPooling2D((2, 2)))
model1.add(layers.Conv2D(128, (3, 3), activation='relu'))
model1.add(layers.MaxPooling2D((2, 2)))
model1.add(layers.Conv2D(128, (3, 3), activation='relu'))
model1.add(layers.MaxPooling2D((2, 2)))
model1.add(layers.Flatten())
model1.add(layers.Dense(512, activation='relu'))
model1.add(layers.Dense(1, activation='sigmoid'))
```

Ilustración 13. Primera red convolucional

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 248, 248, 32)      896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 124, 124, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 122, 122, 64)      18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 61, 61, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 59, 59, 128)       73856
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 29, 29, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 27, 27, 128)       147584
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 13, 13, 128)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 21632)             0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               11076096
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 513
=================================================================
Total params: 11,317,441
Trainable params: 11,317,441
Non-trainable params: 0
_________________________________________________________________
```

Tras 30 épocas se obtiene una precisión de 99.5% en el conjunto de train y del 57.2% en el de test, denotando un claro overfitting.

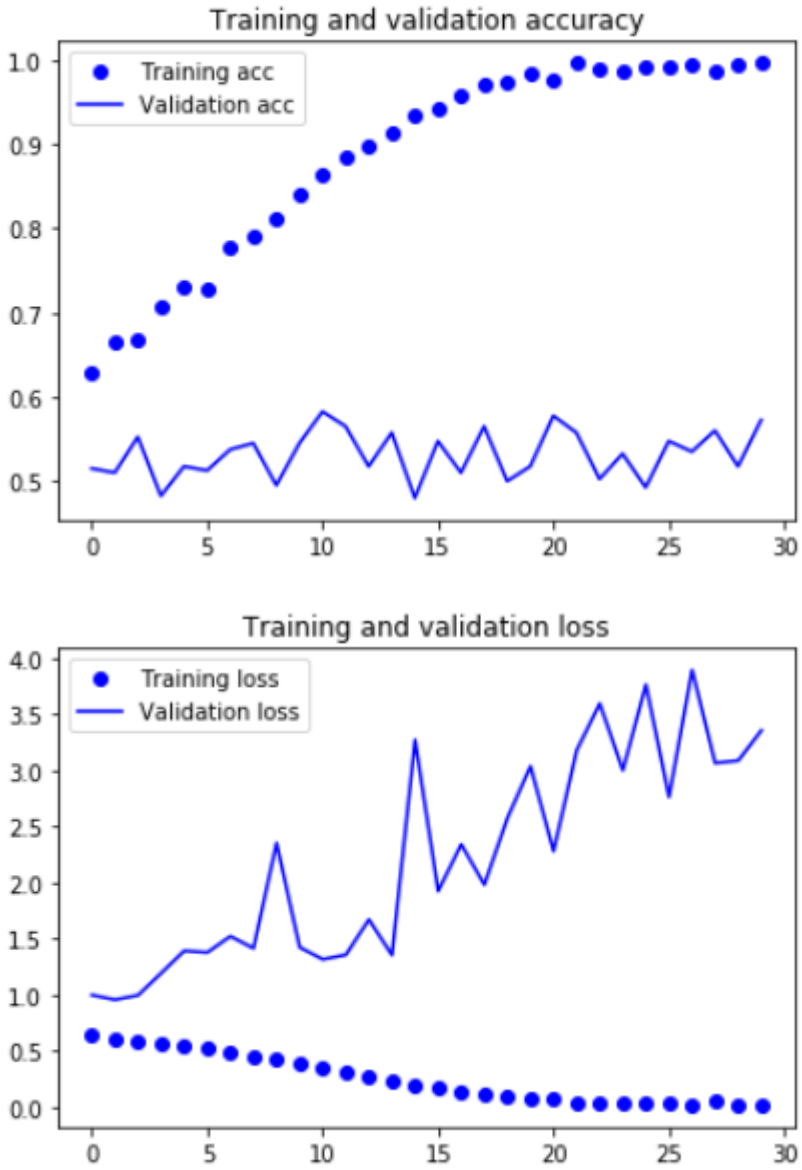


Ilustración 14. Loss por Épocas

5.2 Red con Data Augmentation y Drop-out

El overfitting generado en la primera red es consecuencia de las escasas imágenes disponibles. Esto hace que la red en el conjunto de train en vez de aprender memorice las características y luego no es capaz de generalizarlas en el conjunto de test. Para evitar el overfitting se dispone de dos mecanismos, el DROP-OUT y el Data Augmentation.

En esta segunda red se aplican los dos.

```
In [0]: model = models.Sequential()
        model.add(layers.Conv2D(32, (3, 3), activation='relu',
                                input_shape=(250, 250, 3)))
        model.add(layers.MaxPooling2D((2, 2)))
        model.add(layers.Conv2D(64, (3, 3), activation='relu'))
        model.add(layers.MaxPooling2D((2, 2)))
        model.add(layers.Conv2D(128, (3, 3), activation='relu'))
        model.add(layers.MaxPooling2D((2, 2)))
        model.add(layers.Conv2D(128, (3, 3), activation='relu'))
        model.add(layers.MaxPooling2D((2, 2)))
        model.add(layers.Flatten())
        model.add(layers.Dropout(0.5))
        model.add(layers.Dense(512, activation='relu'))
        model.add(layers.Dense(1, activation='sigmoid'))

        model.compile(loss='binary_crossentropy',
                      optimizer=optimizers.RMSprop(lr=1e-4),
                      metrics=['acc'])
```

Ilustración 15. Red Covolucional con Drop-Out

En Drop-Out desconecta de manera aleatoria nodos de la red, mientras que Data Augmentation genera imágenes con diferentes transformaciones a partir de las imágenes originales, rotándolas, alargándolas, ampliándolas, etc.

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    #shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,)
```

Ilustración 16. Data Augmentation

En la siguiente tabla se presentan los resultados de Data Augmentation sobre la imagen de un mismo embrión.

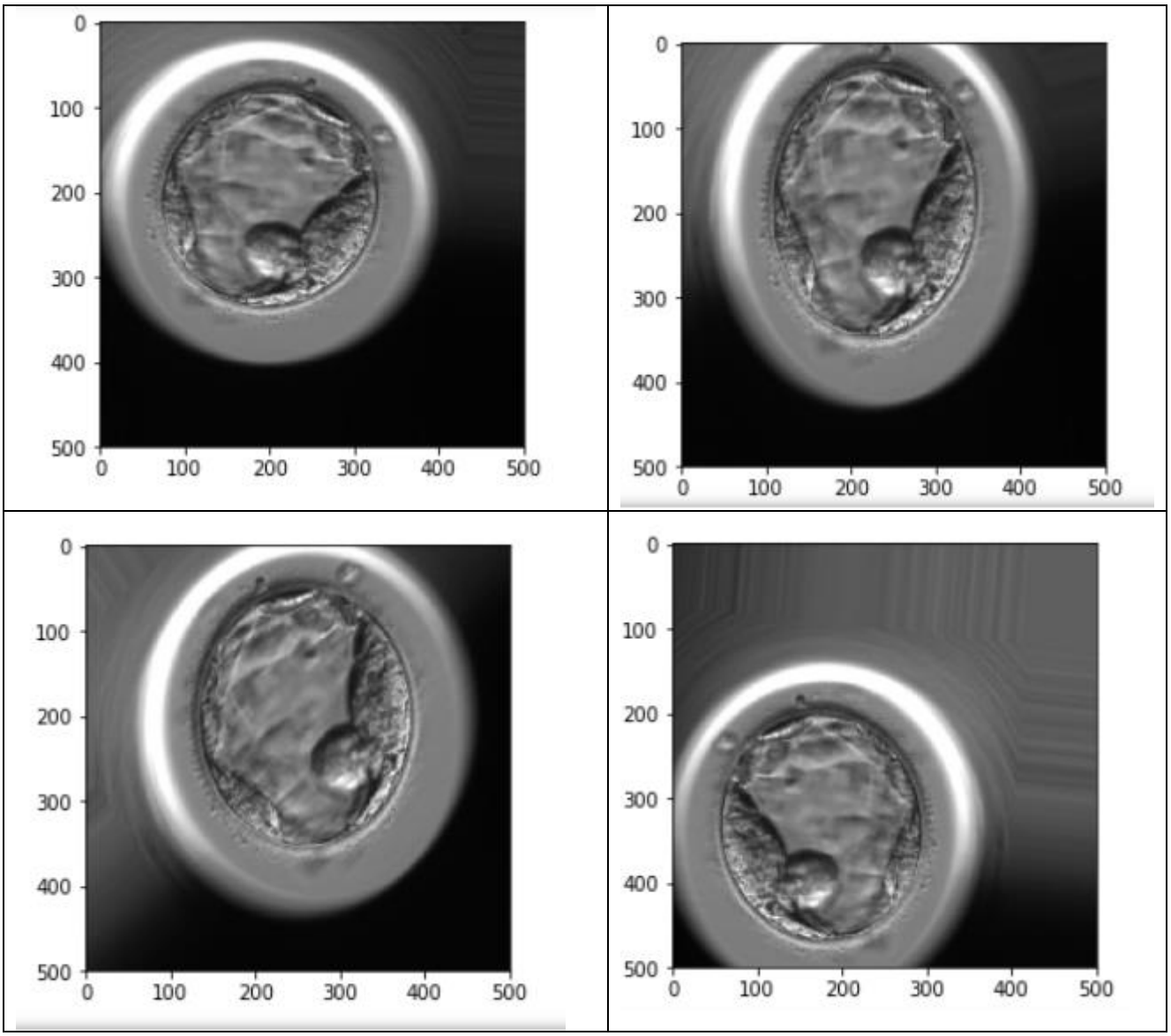


Ilustración 17. Resultados Data Augmentation

Tras 100 épocas se obtiene una precisión de 67.2% en el conjunto de train y del 60.2% en el de test.

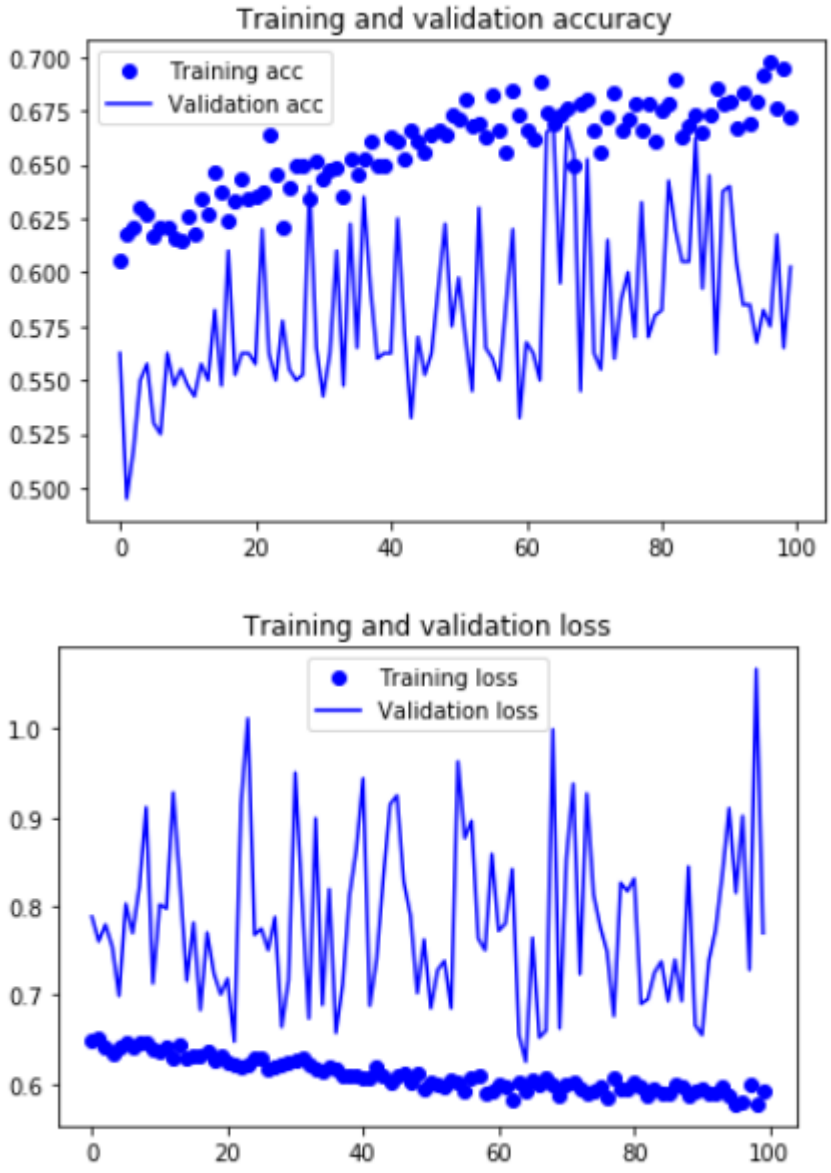


Ilustración 18. Loss por épocas con DA

5.3 Transfer Learning

Transfer Learning permite utilizar una red pre entrenada sobre nuestros datos. En este caso se utiliza VGG16.

```
In [20]: from keras.applications import VGG16

         conv_base = VGG16(weights='imagenet',
                           include_top=False,
                           input_shape=(250, 250, 3))
```

```
Downloading data from https://github.com/fchollet/deep-learning-models/releases/download/v0.1/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 1s 0us/step
```

Ilustración 19. Carga de Pesos. Transfer Learning

```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 250, 250, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 250, 250, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 250, 250, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 125, 125, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 125, 125, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 125, 125, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 62, 62, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 62, 62, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 62, 62, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 62, 62, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 31, 31, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 31, 31, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 31, 31, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 31, 31, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 15, 15, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 15, 15, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 15, 15, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 15, 15, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
=================================================================
Total params: 14,714,688
```

```
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

Aplicamos a VGG16, la estructura de nuestros datos

```
In [0]: from keras import models
        from keras import layers
        model = models.Sequential()
        model.add(conv_base)
        model.add(layers.Flatten())
        model.add(layers.Dense(256, activation='relu'))
        model.add(layers.Dense(1, activation='sigmoid'))
```

Ilustración 20. Red Con Transfer Learning

Utilizamos DA junto con transfer Learning

```
from keras import optimizers
from keras.preprocessing.image import ImageDataGenerator
from PIL import ImageFile
ImageFile.LOAD_TRUNCATED_IMAGES = True
train_dir='./balan800/train'
validation_dir='./balan800/test'
train_datagen = ImageDataGenerator(
      rescale=1./255,
      featurewise_center=True,
      featurewise_std_normalization=True,
      rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      #shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True,
      vertical_flip=True,
      fill_mode='nearest')
# Note that the validation data should not be augmented!
test_datagen = ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_directory(
        # This is the target directory
        train_dir,
        # All images will be resized to 150x150
        target_size=(250, 250),
        batch_size=20,
        # Since we use binary_crossentropy loss, we need binary labels
        class_mode='binary')

validation_generator = test_datagen.flow_from_directory(
        validation_dir,
        target_size=(250, 250),
        batch_size=20,
        class_mode='binary')

model.compile(loss='binary_crossentropy',#este es el mejor
              optimizer=optimizers.SGD(lr=1e-4, momentum=0.9),
              metrics=['acc'])
history = model.fit_generator(
      train_generator,
      steps_per_epoch=80,
      epochs=100,
      validation_data=validation_generator,
      validation_steps=20,
      verbose=2)
```

Ilustración 21. Cara y ejecución Transfer learning

Tras 100 épocas se obtiene una precisión de 69.75% en el conjunto de train y del 62.3 % en el de test

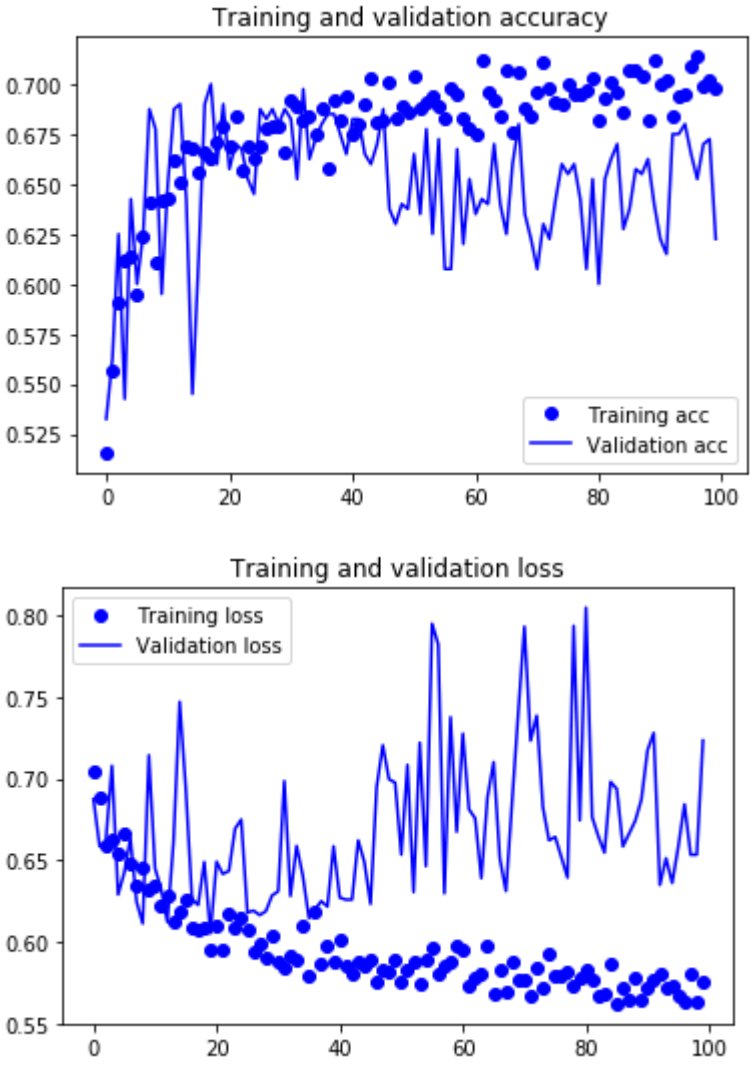


Ilustración 22. Loss por épocas con Transfer Learnig

5.4 Mejora del modelo

Se han ejecutado 6 redes pre entrenadas con los pesos disponibles en Keras. Para cada uno de estos 6 modelos se presentan los resultados de varias métricas en el conjunto de train y en el conjunto de test.
Para cada red se han dejado ajustar los parámetros en las últimas capas
El mejor modelo es Xception-Fine Tuning, con unos resultados de:

- Accuracy .67 en train y .66 en test
- Precision ..67 en train y ..68 en test
- Recall ..68 en train y …72 en test
- F1-Score .68 en train y ..68 en test

La red tiende a clasificar los embriones buenos con mayor facilidad.
Esta red aunque presenta algunas métricas más bajas que otras redes (VGG16-Fine Tuning) presenta una mejor precision en el conjunto de Test. Las probabilidades de que un embrión clasificado como de buena calidad, lo sea es del 68 %. Este resultado hace que esta red sea la que disponga una mayor utilidad práctica, dado que el objetivo es identificar cuantos más embriones de buena calidad que mejoran las expectativas de embarazo.

Se presenta un resumen con todos los resultados de las distintas redes:

VGG 16	Train	Test
Accuray	.7019	.6250
Precision	.6996	.5796
Recall	.7075	.91
F1-Score	.7035	.7082

VGG 16-Fine Tuning	Train	Test
Accuray	.7431	.6625
Precision	.7440	.6132
Recall	.7412	.8800
F1-Score	.7426	.7228

VGG 19	Train	Test
Accuray	.6931	.6625
Precision	.6905	.6102
Recall	.7000	.9000
F1-Score	.6952	.7233

VGG 19 Fine Tuning	Train	Test
Accuray	.7400	.6625
Precision	.7382	.6121
Recall	.7437	.8850
F1-Score	.7410	.7239

Xception	Train	Test
Accuray	.6744	.6250
Precision	.6667	.5977
Recall	.6975	.7650
F1-Score	.6817	.6711

Xception/Fine Tuning	Train	Test
Accuray	.6737	.6650
Precision	.6703	.6773
Recall	.6837	.7250
F1-Score	.6770	.6840

ResNet 50	Train	Test
Accuray	.7056	.5
Precision	.7024	0
Recall	.6962	0
F1-Score	.6993	0

ResNet 50/Fine Tuning	Train	Test
Accuray	.7269	.5775
Precision	.7330	.5447
Recall	.7137	.9450
F1-Score	.7232	.6910

Inception V3	Train	Test
Accuray	.6456	.6150
Precision	.6416	.5858
Recall	.6600	.7850
F1-Score	.6506	.6709

Inception V3/Fine Tuning	Train	Test
Accuray	.6713	.6525
Precision	.6663	.6142
Recall	.6862	.8200
F1-Score	.6771	.7024

InceptionResNetV2	Train	Test
Accuray	.6625	.5125
Precision	.6693	.5063
Recall	.6425	1
F1-Score	.6556	.6723

Inception V3/Fine Tuning	Train	Test
Accuray	.6694	.6650
Precision	.6615	.6587
Recall	.6937	.6850
F1-Score	.6772	.6716

5.5 Cyclying Learning Rate.

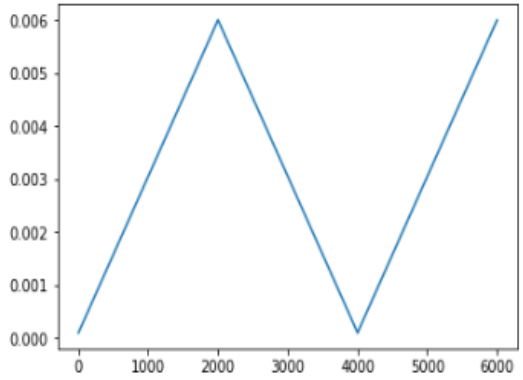
La elección del learning rate es un paso importante en la configuración de la red. En los primeros pasos se han probado diferentes valores estáticos a partir de ejemplos descritos en diferentes ejemplos bibliográficos. A instancias del consultor se ha implementado un método dinámico de selección de learning rate.
Se ha seguido el tutorial descrito en https://github.com/bckenstler/CLR y en este artículo: https://arxiv.org/abs/1506.01186

Se han implementado diferentes políticas (triangular, triangular2, exp_range, custom, custom iteration)

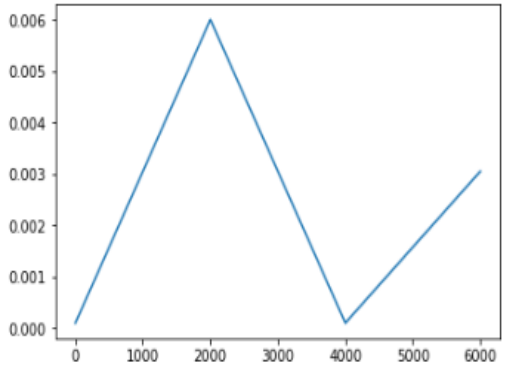
Se ha implementado CLR como técnica de ajuste dinámico del LR sobre la red VGG16.
Se han implementado diversas políticas:

Tabla 5:Resumen CLR

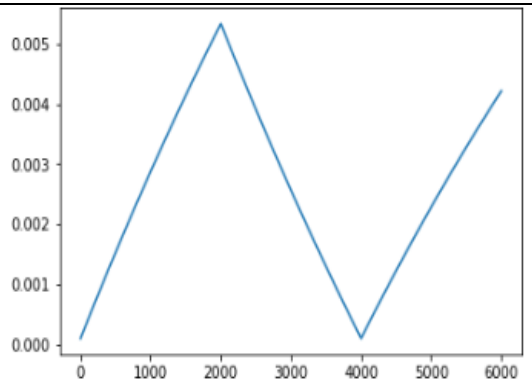
	Triangular		Triangular2		exp_range		Custom (0003-0009)		Custom Iteration		Custom Iteration	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Accuray	0,67	0,66	0,72	0,63	0,71	0,66	0,69	0,6	0,71	0,61	0,72	0,66
Precision	0,68	0,62	0,73	0,59	0,73	0,62	0,7	0,58	0,72	0,57	0,73	0,6
Recall	0,63	0,8	0,71	0,85	0,66	0,82	0,7	0,86	0,67	0,86	0,71	0,89
F1-Score	0,65	0,7	0,72	0,7	0,7	0,71	0,7	0,69	0,69	0,68	0,72	0,72



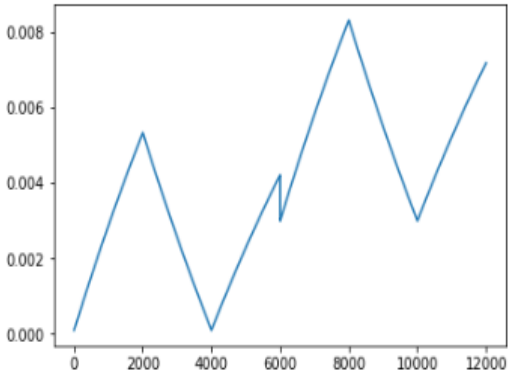
Triangular



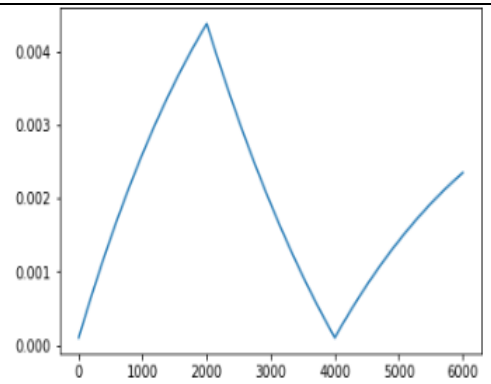
Triangular 2



Exp_range



Custom Iteration



Custom Iteration

Ilustración 23. CLR

5.6 SVM

Debido a que los resultados eran más pobres de lo esperado se ha evaluado si el problema estaba en la extracción de las características o en la última capa de clasificación de la red neuronal. Para ello. se ha aplicado un algoritmo SVM a las características generadas por la red con mejores resultados. Se ha generado un programa en Python que iteraba los archivos en cada directorio, sobre cada archivo se aplicaba el modelo VGG16. Para las imágenes de train como para las de test se ha generado una lista con sus features y su grupo correspondiente. Sobre esta configuración de datos se ha aplicado el SVM utilizando el paquete sklearn. Se ha calculado la matriz de confusión obteniéndose unos resultados similares a la de la red. Esto implica que el problema radica en que los modelos no son capaces identificar features en este conjunto de imágenes.

```
from os import listdir
from os.path import isfile, join
from keras.preprocessing import image
from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import preprocess_input
import numpy as np
from sklearn import metrics
from sklearn.cluster import KMeans
from sklearn.datasets import load_digits
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale
from PIL import ImageFile
ImageFile.LOAD_TRUNCATED_IMAGES = True
model = VGG16(weights='imagenet', include_top=False)
#os.remove(content/balan800/train/good/Thumbs.db)
#os.remove(content/balan800/train/poor/Thumbs.db)
vgg16_feature_list = []
rootpath = "./balan800/train/"
subdir = ['good', 'poor']

for idx, item in enumerate(subdir):
    pathname = rootpath + item
    print("[dir] ", pathname)
    filenames = [f for f in listdir(pathname) if isfile(join(pathname, f))]
    for i, fname in enumerate(filenames):
        a=pathname+"/"+fname
        #print(a)
        img = image.load_img(a,target_size=(224, 224))
        img_data = image.img_to_array(img)
        img_data = np.expand_dims(img_data, axis=0)
        img_data = preprocess_input(img_data)
        vgg16_feature = model.predict(img_data)
        vgg16_feature_np = np.array(vgg16_feature)
        vgg16_feature_list.append(vgg16_feature_np.flatten())


vgg16_feature_list_np = np.array(vgg16_feature_list)
#kmeans = KMeans(n_clusters=2, random_state=0).fit(vgg16_feature_list_np)
```

```
[ ]  import numpy as np
     import matplotlib.pyplot as plt

     from sklearn import svm, datasets
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import confusion_matrix,classification_report
     from sklearn.utils.multiclass import unique_labels
```

```
[ ]  classifier = svm.SVC(kernel='linear', C=0.01)
     y_pred = classifier.fit(vgg16_feature_list_np, Y1).predict(vgg16_feature_list_test_np)
```

```
[ ]    cm = confusion_matrix(Y1_test,y_pred )
       cm
       #class_names
```

```
array([[170,  30],
       [111,  89]])
```

Ilustración 24. Código SVM

Se presenta la Matriz de Confusión y las métricas:

```
Confusion matrix, without normalization
[[170  30]
 [111  89]]
Normalized confusion matrix
[[0.85 0.15]
 [0.56 0.45]]
```

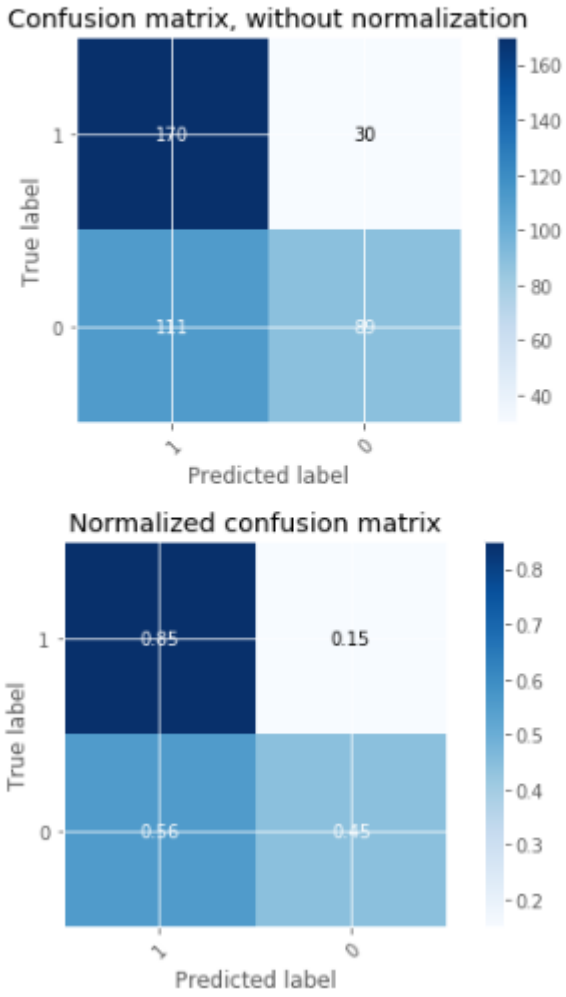


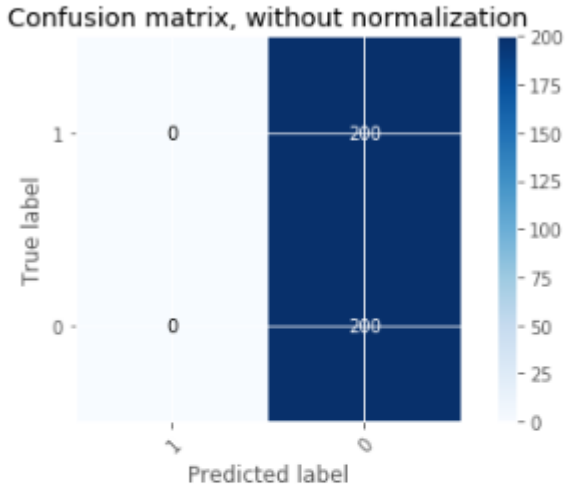
Ilustración 25. Matriz Confusión SVM

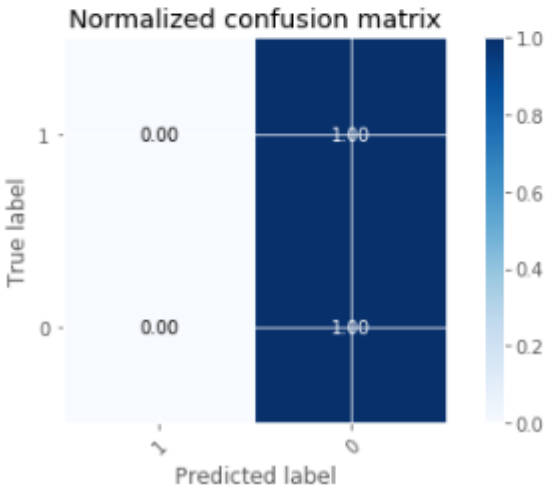
```
target_names =['1','0']
print(classification_report(Y1_test,y_pred, target_names=target_names))
```

```
              precision    recall  f1-score   support

           1       0.60      0.85      0.71       200
           0       0.75      0.45      0.56       200

   micro avg       0.65      0.65      0.65       400
   macro avg       0.68      0.65      0.63       400
weighted avg       0.68      0.65      0.63       400
```

Se ha replicado el análisis con diferentes Kernels, obteniendo resultados similares
RBF

```
Confusion matrix, without normalization
[[  0 200]
 [  0 200]]
Normalized confusion matrix
[[0. 1.]
 [0. 1.]]
```



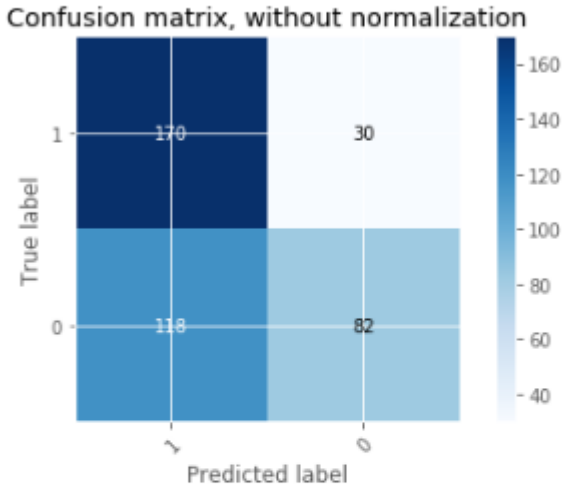


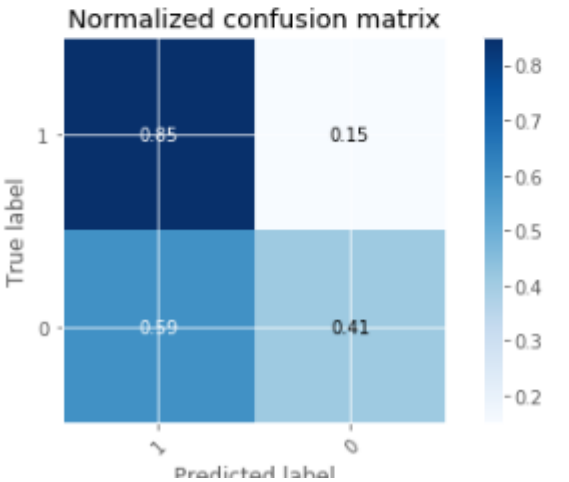
	precision	recall	f1-score	support
1	0.00	0.00	0.00	200
0	0.50	1.00	0.67	200
accuracy			0.50	400
macro avg	0.25	0.50	0.33	400
weighted avg	0.25	0.50	0.33	400

Ilustración 26. SVM Kernel RBF

Polinómico de grado 3:

```
Confusion matrix, without normalization
[[170  30]
 [118  82]]
Normalized confusion matrix
[[0.85 0.15]
 [0.59 0.41]]
```





	precision	recall	f1-score	support
1	0.59	0.85	0.70	200
0	0.73	0.41	0.53	200
accuracy			0.63	400
macro avg	0.66	0.63	0.61	400
weighted avg	0.66	0.63	0.61	400

Ilustración 27. SVM Kernel Polinómico

Los resultados obtenidos son similares. Esto denota que el problema de la baja exactitud de la red no se debe a la última capa clasificatoria, si no ha la incapacidad de generar características, bien por el bajo número de imágenes, baja variabilidad o una falta de diferenciación clara entre los positivos y los negativos.

Ampliación del Número de Imágenes

Dado que los resultados no son satisfactorios, se han localizado nuevas imágenes.
En un principio el conjunto abarcaba a imágenes correspondiente a embriones congelados o transferidos. Se ha aumentado la muestra seleccionado aquellos embriones que por su mala calidad se descartan.
Se han localizado las imágenes, clasificado y revisado de nuevo para conseguir un conjunto de 3302 imágenes en train y 1017 en test

```
Found 3302 images belonging to 2 classes.
Found 1017 images belonging to 2 classes.
Epoch 1/30
```

Los resultados son similares:

```
Epoch 30/30
 - 23s - loss: 0.5219 - precision: 0.6753 - recall: 0.5393 - f1_score: 0.5997 - val_loss: 0.5320 -
```

6. Conclusiones

La generalización del uso de tecnologías de IA debe extenderse a nuevos ámbitos de actuación. La esterilidad es un problema de primer orden en las sociedades modernas(24,25). Este estudio ha abordado un problema real en un contexto nuevo. La selección de los embriones a transferir es un proceso crucial. El personal de embriología selecciona al mejor embrión en función de dos características morfológicas. El objetivo de este estudio ha sido evaluar si un algoritmo de DL es capaz de realizar esta tarea de manera automática a partir de las imágenes captadas de manera automática por el incubador timelapse.
El resultado obtenido en el conjunto de test (66% accuracy) no se acerca a los esperados en la aplicación de las CNN (>90)(19)- Otros grupos han alcanzado recientemente mejores resultados con AUC (>.98)(28), siguiendo una metodología similar. La baja disponibilidad de imágenes en contraste con estos grupos (12.000 imágenes) puede justificar la diferencia en los resultados. Disponer de un mayor número de imágenes y realizar algún preprocesado previo podría mejorar los resultados de la red.

La bibliografía aparecida posteriormente al inicio de este estudio(28) indica que la metodología elegida, el ámbito de aplicación y la manera de abordar el problema originalmente han sido correctas.
Se han analizado las imágenes siguiendo las pautas para conjuntos de datos de bajo número(29), implementado Data Augmentation y Transfer Learning con varias redes pre entrenadas. Respecto al LR se ha utilizado un método dinámico que no ha evidenciado una mejora en el modelo. En última instancia para identificar si el problema era debido a la extracción de características del modelo o la capacidad de clasificación de la última capa, se ha aplicado un algoritmo SVM con varios kernels obteniéndose resultados similares, evidenciando la incapacidad de la red para extraer características con potencial clasificador.

Las redes utilizadas no han sido capaces de identificar en las continuas capas convolucionales las características visuales del trofoectodermo y de la masa celular interna que utilizan los embriólogos para clasificar el embrión.
No obstante, el proceso de afinado de la red requiere de la configuración de múltiples parámetros, la mala elección u omisión de alguno de ellos pudiera afectar al rendimiento del modelo.
Otro posible motivo del bajo rendimiento de la red pudiera ser un erróneo etiquetaje en origen del embrión. Las etiquetas utilizadas se han aplicado en la práctica clínica real, por personal de embriología de diferentes niveles de experiencia. Una posible mejora hubiera sido una revisión previa por personal experto de las imágenes para asegurase que el etiquetado era correcto.
Es necesario realizar múltiples estudios con más imágenes de este tipo para conseguir modelos con pesos pre entrenados que permitan la utilización de estas tecnologías en el laboratorio.

7. Glosario

ASEBIR	Asociación para el estudio de la Biología de la Reproducción
CLR	Cyclying Learning Rate
CNN	Convolutional Neural Network
DA	Data Augmentation
DGP	Diagnóstico Genético preimplantacional
DL	Deep Learning
FIV	Fecundación InVitro
GnRH	Hormona Liberadora de la gonadotropina humana
hCG	Gonadotropìna corionica humana
IA	Inteligencia Artificial
IAC	Inseminación artificial con semen de pareja
IAD	Inseminación artificial con semen de donante
ICSI	inyección intracitoplasmatica de espermatozoides
IO	Inducción de la ovulación
LR	Learning Rate
ML	Machine Learning
PCA	Principal Components analysis
RNA	Red neuronal artificial
SEF	Sociedad Española de Fertilidad
SVM	Support Vector Machine
TL	Transfer Learning
TRA	Técnicas de Reproducción Asistida

8. Bibliografía

1. Devesa M, Tur R, Rodríguez I, Coroleu B, Martínez F, Polyzos NP. Cumulative live birth rates and number of oocytes retrieved in women of advanced age. A single centre analysis including 4500 women ≥38 years old. Hum Reprod Oxf Engl. 01 de 2018;33(11):2010-7.

2. Ledford H. IVF at 40: revisiting the revolution in assisted reproduction. Nature [Internet]. 23 de julio de 2018 [citado 11 de marzo de 2019]; Disponible en: http://www.nature.com/articles/d41586-018-05792-9

3. Niederberger C, Pellicer A, Cohen J, Gardner DK, Palermo GD, O'Neill CL, et al. Forty years of IVF. Fertil Steril. 15 de 2018;110(2):185-324.e5.

4. Meseguer M, Kruhne U, Laursen S. Full in vitro fertilization laboratory mechanization: toward robotic assisted reproduction? Fertil Steril. junio de 2012;97(6):1277-86.

5. Alpha Scientists in Reproductive Medicine and ESHRE Special Interest Group of Embryology. The Istanbul consensus workshop on embryo assessment: proceedings of an expert meeting. Hum Reprod Oxf Engl. junio de 2011;26(6):1270-83.

6. Meseguer M, Herrero J, Tejera A, Hilligsøe KM, Ramsing NB, Remohí J. The use of morphokinetics as a predictor of embryo implantation. Hum Reprod. 1 de octubre de 2011;26(10):2658-71.

7. Artificial Intelligence: A Modern Approach [Internet]. [citado 11 de marzo de 2019]. Disponible en: http://aima.cs.berkeley.edu/

8. Bishop CM. Pattern recognition and machine learning. New York: Springer; 2006. 738 p. (Information science and statistics).

9. Bengio Y. Learning Deep Architectures for AI. Found Trends® Mach Learn. 2009;2(1):1-127.

10. Krizhevsky A, Sutskever I, Hinton GE. ImageNet Classification with Deep Convolutional Neural Networks. En: Pereira F, Burges CJC, Bottou L, Weinberger KQ, editores. Advances in Neural Information Processing Systems 25 [Internet]. Curran Associates, Inc.; 2012 [citado 28 de febrero de 2019]. p. 1097–1105. Disponible en: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

11. ASEBIR. Cuaderno de embriología Clínica. Criterios ASEBIR de valoración Morfológica de Oocitos, Embriones Tempranos y Blastocistos Humanos. 3ª ed. 2015.

12. Ananthram A. Keras Transfer Learning For Beginners [Internet]. Towards Data Science. 2018 [citado 28 de febrero de 2019]. Disponible en:

https://towardsdatascience.com/keras-transfer-learning-for-beginners-6c9b8b7143e

13. Guide to CNNs with Data Augmentation (Keras) | Kaggle [Internet]. [citado 28 de febrero de 2019]. Disponible en: https://www.kaggle.com/moghazy/guide-to-cnns-with-data-augmentation-keras

14. Deep Learning with Keras : Implement various deep-learning algorithms in Keras and see how deep-learning can be used in games [Book] [Internet]. [citado 11 de marzo de 2019]. Disponible en: https://www.oreilly.com/library/view/deep-learning-with/9781787128422/

15. Allaire JJ. TensorFlow for R [Internet]. [citado 28 de febrero de 2019]. Disponible en: /keras/articles/tutorial_basic_classification.html

16. Torres J. «Hello World» en TensorFlow [Internet]. Jordi Torres.AI - Professor and Researcher at UPC & BSC: Supercomputing for Artificial Intelligence and Deep Learning. [citado 28 de febrero de 2019]. Disponible en: https://torres.ai/es/tensorflow/libro-hello-world-en-tensorflow/

17. Home - Keras Documentation [Internet]. [citado 29 de mayo de 2019]. Disponible en: https://keras.io/

18. Project Jupyter [Internet]. [citado 29 de mayo de 2019]. Disponible en: https://www.jupyter.org

19. Deep Learning with Python [Internet]. Manning Publications. [citado 24 de mayo de 2019]. Disponible en: https://www.manning.com/books/deep-learning-with-python

20. Lantz B. Machine Learning with R. Packt Publishing; 2013.

21. Deep Learning – Introducción práctica con Keras [Internet]. Jordi TORRES.AI. [citado 24 de mayo de 2019]. Disponible en: https://torres.ai/deep-learning-inteligencia-artificial-keras/

22. Back Propagation Neural Network: Explained With Simple Example [Internet]. [citado 24 de mayo de 2019]. Disponible en: https://www.guru99.com/backpropogation-neural-network.html

23. Transfer Learning with Keras in R [Internet]. R-bloggers. 2017 [citado 28 de febrero de 2019]. Disponible en: https://www.r-bloggers.com/transfer-learning-with-keras-in-r/

24. García V. Informe estadístico. 2016;50.

25. Inhorn MC, Patrizio P. Infertility around the globe: new thinking on gender, reproductive technologies and global movements in the 21st century. Hum Reprod Update. agosto de 2015;21(4):411-26.

26. Steptoe PC, Edwards RG. Birth after the reimplantation of a human embryo. Lancet Lond Engl. 12 de agosto de 1978;2(8085):366.

27. Armstrong S, Arroll N, Cree LM, Jordan V, Farquhar C. Time-lapse systems for embryo incubation and assessment in assisted reproduction. Cochrane Database Syst Rev. 27 de febrero de 2015;(2):CD011320.

28. Khosravi P, Kazemi E, Zhan Q, Malmsten JE, Toschi M, Zisimopoulos P, et al. Deep learning enables robust assessment and selection of human blastocysts after in vitro fertilization. Npj Digit Med. 4 de abril de 2019;2(1):21.

29. Building powerful image classification models using very little data [Internet]. [citado 22 de marzo de 2019]. Disponible en: https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

9. Anexos

9.1 Consulta SQL:

```
select *, dia=5 from (select ROW_NUMBER() over (partition by mov.proceso,
mov.ovocito
 order                                                                  by
(substring(dig.referencia,CHARINDEX('_',dig.referencia)+1,len(dig.referencia)))*1
desc  ) as fila,
dig.numero,mov.ovocito,mov.proceso
,seg.secuencia,substring(dig.referencia,1,CHARINDEX('_',dig.referencia)-1)     As
newref,cinco.scoreasebir,

substring(dig.referencia,CHARINDEX('_',dig.referencia)+1,len(dig.referencia))  as
hora,
                  dig.referencia,dig.fichero,mov.fecha
                  from embryomovimientos mov
                  inner       join       dbo.embryoseguimiento       seg       on
seg.identificador=mov.ovocito and seg.dia=mov.dia and mov.proceso=seg.numero
                  inner join dbo.digitalarchivos dig on mov.proceso=dig.proceso
and dig.tipo='OET'
                              and
seg.secuencia=substring(dig.referencia,1,CHARINDEX('_',dig.referencia)-1)
                  inner       join       dbo.embryoobscinco       cinco       on
mov.ovocito=cinco.identificador and cinco.scoreasebir is not null
                  where  mov.movimiento  in  ('TRF','CGE')  and  mov.dia=5  and
LEN(dig.referencia)>2
                              and
substring(dig.referencia,CHARINDEX('_',dig.referencia)+1,len(dig.referencia))
between 96 and 120
) as p
                  where fila=1
```

9.2 Código R:

```
myconn <-odbcConnect("dexeus02", uid="nacrod", pwd="marina")
newdf <- sqlQuery(myconn, "select *, dia=5 from (select ROW_NUMBER() over (partition by
mov.proceso, mov.ovocito order by
(substring(dig.referencia,CHARINDEX('_',dig.referencia)+1,len(dig.referencia)))*1 desc ) as
fila,
dig.numero,mov.ovocito,mov.proceso
,seg.secuencia,substring(dig.referencia,1,CHARINDEX('_',dig.referencia)-1) As
newref,cinco.scoreasebir,
                substring(dig.referencia,CHARINDEX('_',dig.referencia)+1,len(dig.referencia))
as hora,
                dig.referencia,dig.fichero,mov.fecha
                from embryomovimientos mov
                inner join dbo.embryoseguimiento seg on seg.identificador=mov.ovocito and
seg.dia=mov.dia and mov.proceso=seg.numero
                inner join dbo.digitalarchivos dig on mov.proceso=dig.proceso and
dig.tipo='OET' and seg.secuencia=substring(dig.referencia,1,CHARINDEX('_',dig.referencia)-1)
                inner join dbo.embryoobscinco cinco on mov.ovocito=cinco.identificador and
cinco.scoreasebir is not null
                where mov.movimiento in ('TRF','CGE') and mov.dia=5 and LEN(dig.referencia)>2
and substring(dig.referencia,CHARINDEX('_',dig.referencia)+1,len(dig.referencia)) between 96 and
120
) as p
                where fila=1")
close(myconn)
table(newdf$scoreasebir)

newdf$quality<-ifelse(trimws(newdf$scoreasebir)=="A"|trimws(newdf$scoreasebir)=="B",
"good","poor" )
table(newdf$quality)

dat<-subset(newdf,newdf$quality=="good"|newdf$quality=="poor" )

dat$dir<-gsub("\\","/",dat$fichero,fixed=TRUE)
dir.create("imagenes")
workingDir <- "F:\\Usuaris\\NACROD\\Master Bioestadistica\\TFM\\imagenes"
setwd(workingDir)
if(!dir.exists("good")) dir.create("good")
good <-file.path(workingDir, "good")
if(!dir.exists("poor")) dir.create("poor")
poor <-file.path(workingDir, "poor")


#creo una fución que lee los paths y guarda las imagenes en cada directorio

repositorioweb<-c("http://bcnimg.dexeus.com:8000/")

fotos<-function (dat,dataDir,x) {
  for (i in 714:x)#salto errorfor (i in 1244:x)#salto error
  {
    download.file(paste0(repositorioweb,dat$dir[i]),
                  paste0(dataDir,"/emb",i,".jpg"),mode = 'wb'
                  )
  }
}

fotos(dat[dat$quality=="good",],good,2670)

fotos(dat[dat$quality=="poor",],poor,1178)
```

PEDRO N. BARRI RAGUÉ, as **Director of the Professorship (IRB) in Obstetrics and Gynaecological Research of Hospital Universitari Dexeus**, centre assigned to the Universidad Autónoma de Barcelona

DECLAIRS THAT

On the 13th March 2019 took place a periodical meeting of the members of this Professorship, to study, among others, the project of Scientific Production titled as follows:

"Clasificación automática de calidad embrionaria en embriones en estado de Blastocisto, mediante una Convolutional Neural network (CNN)"

Ignacio Rodríguez

During the meeting, the members agreed on the approval of the project in the way it has been presented, assigning the following approval number: **520190313.**

In witness whereof and for all pertinent purposes, I sign this certificate in Barcelona, on the 26th March 2019.

DEPARTAMENTO DE OBSTETRICIA, GINECOLOGÍA Y REPRODUCCIÓN – INSTITUT UNIVERSITARI DEXEUS
Gran Vía Carlos III, 71-75 - 08028 BARCELONA -
Tel. 93.2274700 - Fax. 93.4170298

Ilustración 28: Aprobación IRB

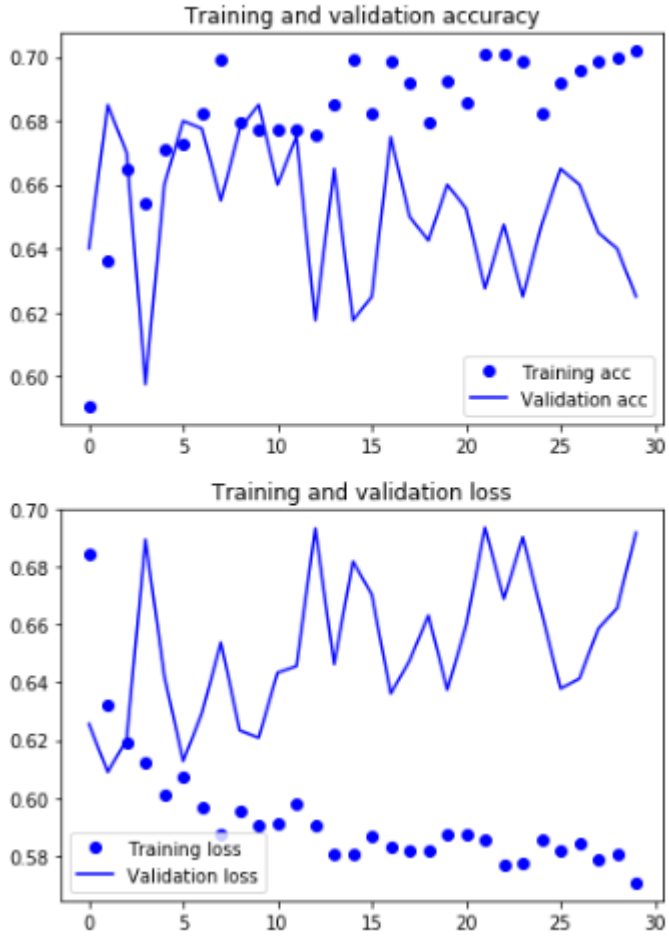
9.3 Mejora Modelos

TRANSFER LEARNING

VGG 16
30 Epochs
Loss: binary_crossentropy
Lr :0.0001

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 4, 4, 512)         14714688
_________________________________________________________________
flatten_1 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               2097408
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 257
=================================================================
Total params: 16,812,353
Trainable params: 16,812,353
Non-trainable params: 0
_________________________________________________________________
```

VGG 16	Train	Test
Accuray	.7019	.6250
Precision	.6996	.5796
Recall	.7075	.91
F1-Score	.7035	.7082


Training and validation accuracy
0.70
0.68
0.66
0.64
0.62
0.60
0
5
10
15
20
25
30
Training acc
Validation acc
Training and validation loss
0.70
0.68
0.66
0.64
0.62
0.60
0.58
0
5
10
15
20
25
30
Training loss
Validation loss

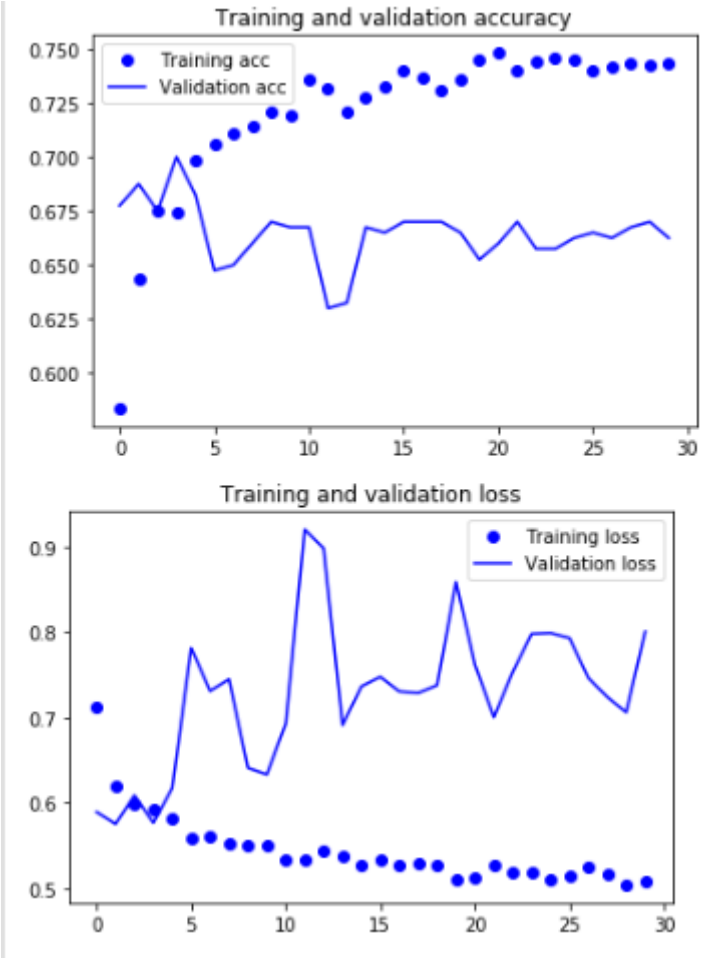
VGG 16 FINE TUNING

30 Epochs
Loss: binary_crossentropy
Lr :0.0001
Se permite entrenar a partir de block5

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 150, 150, 3)	0
block1_conv1 (Conv2D)	(None, 150, 150, 64)	1792
block1_conv2 (Conv2D)	(None, 150, 150, 64)	36928
block1_pool (MaxPooling2D)	(None, 75, 75, 64)	0
block2_conv1 (Conv2D)	(None, 75, 75, 128)	73856
block2_conv2 (Conv2D)	(None, 75, 75, 128)	147584
block2_pool (MaxPooling2D)	(None, 37, 37, 128)	0
block3_conv1 (Conv2D)	(None, 37, 37, 256)	295168
block3_conv2 (Conv2D)	(None, 37, 37, 256)	590080
block3_conv3 (Conv2D)	(None, 37, 37, 256)	590080
block3_pool (MaxPooling2D)	(None, 18, 18, 256)	0
block4_conv1 (Conv2D)	(None, 18, 18, 512)	1180160
block4_conv2 (Conv2D)	(None, 18, 18, 512)	2359808
block4_conv3 (Conv2D)	(None, 18, 18, 512)	2359808
block4_pool (MaxPooling2D)	(None, 9, 9, 512)	0
block5_conv1 (Conv2D)	(None, 9, 9, 512)	2359808
block5_conv2 (Conv2D)	(None, 9, 9, 512)	2359808
block5_conv3 (Conv2D)	(None, 9, 9, 512)	2359808
block5_pool (MaxPooling2D)	(None, 4, 4, 512)	0

Total params: 14,714,688
Trainable params: 0
Non-trainable params: 14,714,688

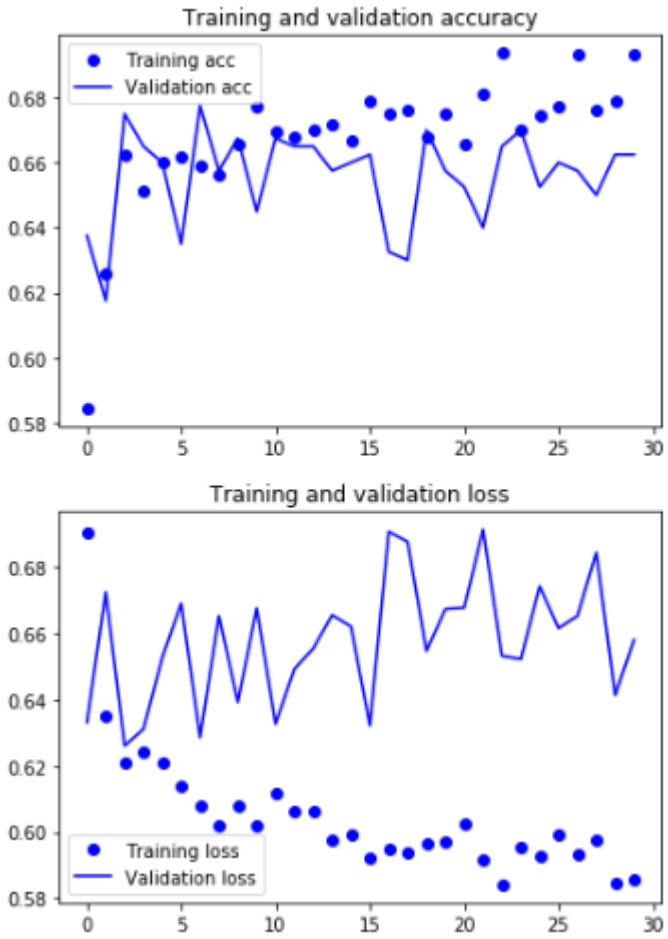
VGG 16-Fine Tuning	Train	Test
Accuray	.7431	.6625
Precision	.7440	.6132
Recall	.7412	.8800
F1-Score	.7426	.7228



VGG 19
30 Epochs
Loss: binary_crossentropy
Lr :0.0001

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg19 (Model)                (None, 4, 4, 512)         20024384
_________________________________________________________________
flatten_2 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_3 (Dense)              (None, 256)               2097408
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 257
=================================================================
Total params: 22,122,049
Trainable params: 22,122,049
Non-trainable params: 0
```

VGG 19	Train	Test
Accuray	.6931	.6625
Precision	.6905	.6102
Recall	.7000	.9000
F1-Score	.6952	.7233



VGG 19 FINE TUNING

30 Epochs
Loss: binary_crossentropy
Lr :0.0001
Se permite entrenar a partir de block5

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 150, 150, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 150, 150, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 150, 150, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 75, 75, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 75, 75, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 75, 75, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 37, 37, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 37, 37, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 37, 37, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 37, 37, 256)       590080
_________________________________________________________________
block3_conv4 (Conv2D)        (None, 37, 37, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 18, 18, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 18, 18, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block4_conv4 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 9, 9, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_conv4 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 4, 4, 512)         0
=================================================================
Total params: 20,024,384
Trainable params: 0
Non-trainable params: 20,024,384
_________________________________________________________________
```

VGG 19 Fine Tuning	Train	Test
Accuray	.7400	.6625
Precision	.7382	.6121
Recall	.7437	.8850
F1-Score	.7410	.7239

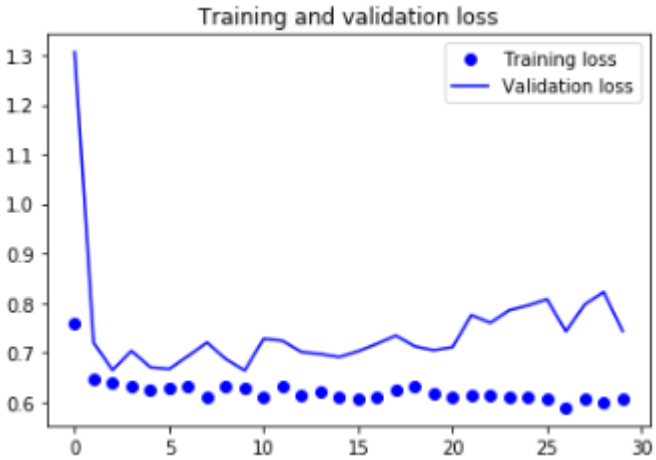

Training and validation accuracy
0.750
0.725
0.700
0.675
0.650
0.625
0.600
0.575
0.550
0
5
10
15
20
25
30
Training acc
Validation acc


Training and validation loss
0.85
0.80
0.75
0.70
0.65
0.60
0.55
Training loss
Validation loss

Xception
30 Epochs
Loss: binary_crossentropy
Lr :0.0001

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
xception (Model)             (None, 5, 5, 2048)        20861480
_________________________________________________________________
flatten_1 (Flatten)          (None, 51200)             0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               13107456
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 257
=================================================================
Total params: 33,969,193
Trainable params: 33,914,665
Non-trainable params: 54,528
_________________________________________________________________
```

Xception	Train	Test
Accuray	.6744	.6250
Precision	.6667	.5977
Recall	.6975	.7650
F1-Score	.6817	.6711



Xception FINE TUNING

30 Epochs
Loss: binary_crossentropy
Lr :0.0001

Se permite entrenar a partir de block14_sepconv1

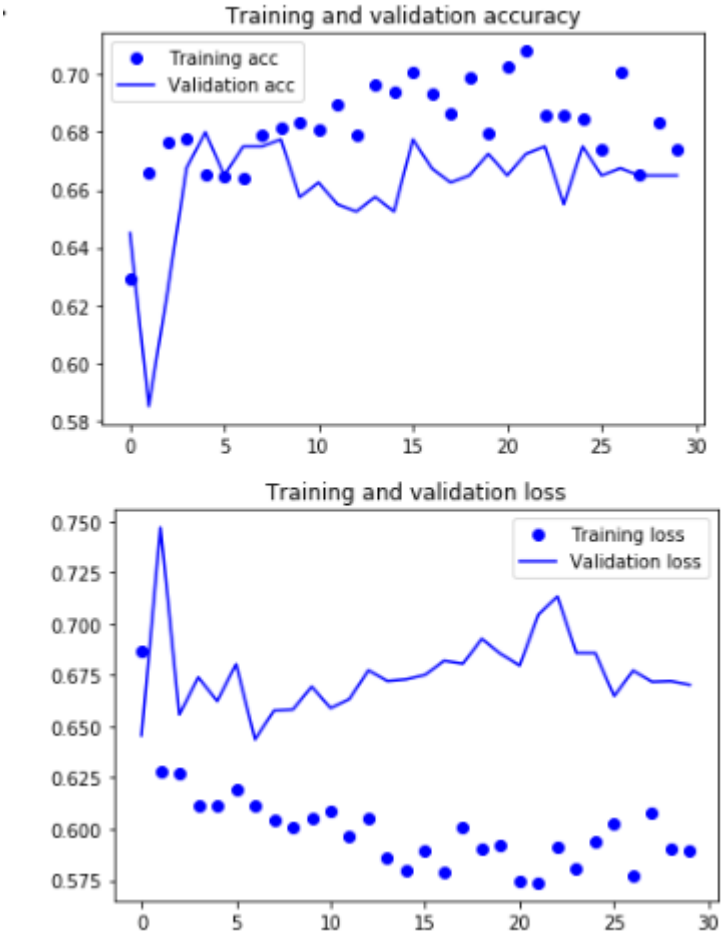
Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 150, 150, 3)	0	
block1_conv1 (Conv2D)	(None, 74, 74, 32)	864	input_1[0][0]
block1_conv1_bn (BatchNormaliza	(None, 74, 74, 32)	128	block1_conv1[0][0]
block1_conv1_act (Activation)	(None, 74, 74, 32)	0	block1_conv1_bn[0][0]
block1_conv2 (Conv2D)	(None, 72, 72, 64)	18432	block1_conv1_act[0][0]
block1_conv2_bn (BatchNormaliza	(None, 72, 72, 64)	256	block1_conv2[0][0]
block1_conv2_act (Activation)	(None, 72, 72, 64)	0	block1_conv2_bn[0][0]
block2_sepconv1 (SeparableConv2	(None, 72, 72, 128)	8768	block1_conv2_act[0][0]
block2_sepconv1_bn (BatchNormal	(None, 72, 72, 128)	512	block2_sepconv1[0][0]
block2_sepconv2_act (Activation	(None, 72, 72, 128)	0	block2_sepconv1_bn[0][0]
block2_sepconv2 (SeparableConv2	(None, 72, 72, 128)	17536	block2_sepconv2_act[0][0]
block2_sepconv2_bn (BatchNormal	(None, 72, 72, 128)	512	block2_sepconv2[0][0]
conv2d_1 (Conv2D)	(None, 36, 36, 128)	8192	block1_conv2_act[0][0]
block2_pool (MaxPooling2D)	(None, 36, 36, 128)	0	block2_sepconv2_bn[0][0]
batch_normalization_1 (BatchNor	(None, 36, 36, 128)	512	conv2d_1[0][0]
add_1 (Add)	(None, 36, 36, 128)	0	block2_pool[0][0] batch_normalization_1[0][0]
block3_sepconv1_act (Activation	(None, 36, 36, 128)	0	add_1[0][0]
block3_sepconv1 (SeparableConv2	(None, 36, 36, 256)	33920	block3_sepconv1_act[0][0]
block3_sepconv1_bn (BatchNormal	(None, 36, 36, 256)	1024	block3_sepconv1[0][0]
block3_sepconv2_act (Activation	(None, 36, 36, 256)	0	block3_sepconv1_bn[0][0]
block3_sepconv2 (SeparableConv2	(None, 36, 36, 256)	67840	block3_sepconv2_act[0][0]
block3_sepconv2_bn (BatchNormal	(None, 36, 36, 256)	1024	block3_sepconv2[0][0]
conv2d_2 (Conv2D)	(None, 18, 18, 256)	32768	add_1[0][0]
block3_pool (MaxPooling2D)	(None, 18, 18, 256)	0	block3_sepconv2_bn[0][0]
batch_normalization_2 (BatchNor	(None, 18, 18, 256)	1024	conv2d_2[0][0]
add_2 (Add)	(None, 18, 18, 256)	0	block3_pool[0][0] batch_normalization_2[0][0]
block4_sepconv1_act (Activation	(None, 18, 18, 256)	0	add_2[0][0]
block4_sepconv1 (SeparableConv2	(None, 18, 18, 728)	188672	block4_sepconv1_act[0][0]
block4_sepconv1_bn (BatchNormal	(None, 18, 18, 728)	2912	block4_sepconv1[0][0]
block4_sepconv2_act (Activation	(None, 18, 18, 728)	0	block4_sepconv1_bn[0][0]
block4_sepconv2 (SeparableConv2	(None, 18, 18, 728)	536536	block4_sepconv2_act[0][0]
block4_sepconv2_bn (BatchNormal	(None, 18, 18, 728)	2912	block4_sepconv2[0][0]
conv2d_3 (Conv2D)	(None, 9, 9, 728)	186368	add_2[0][0]
block4_pool (MaxPooling2D)	(None, 9, 9, 728)	0	block4_sepconv2_bn[0][0]
batch_normalization_3 (BatchNor	(None, 9, 9, 728)	2912	conv2d_3[0][0]
add_3 (Add)	(None, 9, 9, 728)	0	block4_pool[0][0] batch_normalization_3[0][0]
block5_sepconv1_act (Activation	(None, 9, 9, 728)	0	add_3[0][0]
block5_sepconv1 (SeparableConv2	(None, 9, 9, 728)	536536	block5_sepconv1_act[0][0]
block5_sepconv1_bn (BatchNormal	(None, 9, 9, 728)	2912	block5_sepconv1[0][0]
block5_sepconv2_act (Activation	(None, 9, 9, 728)	0	block5_sepconv1_bn[0][0]
block5_sepconv2 (SeparableConv2	(None, 9, 9, 728)	536536	block5_sepconv2_act[0][0]
block5_sepconv2_bn (BatchNormal	(None, 9, 9, 728)	2912	block5_sepconv2[0][0]
block5_sepconv3_act (Activation	(None, 9, 9, 728)	0	block5_sepconv2_bn[0][0]
block5_sepconv3 (SeparableConv2	(None, 9, 9, 728)	536536	block5_sepconv3_act[0][0]
block5_sepconv3_bn (BatchNormal	(None, 9, 9, 728)	2912	block5_sepconv3[0][0]
add_4 (Add)	(None, 9, 9, 728)	0	block5_sepconv3_bn[0][0] add_3[0][0]
block6_sepconv1_act (Activation	(None, 9, 9, 728)	0	add_4[0][0]

block6_sepconv1 (SeparableConv2	(None, 9, 9, 728)	536536	block6_sepconv1_act[0][0]	
block6_sepconv1_bn (BatchNormal	(None, 9, 9, 728)	2912	block6_sepconv1[0][0]	
block6_sepconv2_act (Activation	(None, 9, 9, 728)	0	block6_sepconv1_bn[0][0]	
block6_sepconv2 (SeparableConv2	(None, 9, 9, 728)	536536	block6_sepconv2_act[0][0]	
block6_sepconv2_bn (BatchNormal	(None, 9, 9, 728)	2912	block6_sepconv2[0][0]	
block6_sepconv3_act (Activation	(None, 9, 9, 728)	0	block6_sepconv2_bn[0][0]	
block6_sepconv3 (SeparableConv2	(None, 9, 9, 728)	536536	block6_sepconv3_act[0][0]	
block6_sepconv3_bn (BatchNormal	(None, 9, 9, 728)	2912	block6_sepconv3[0][0]	
add_5 (Add)	(None, 9, 9, 728)	0	block6_sepconv3_bn[0][0] add_4[0][0]	
block7_sepconv1_act (Activation	(None, 9, 9, 728)	0	add_5[0][0]	
block7_sepconv1 (SeparableConv2	(None, 9, 9, 728)	536536	block7_sepconv1_act[0][0]	
block7_sepconv1_bn (BatchNormal	(None, 9, 9, 728)	2912	block7_sepconv1[0][0]	
block7_sepconv2_act (Activation	(None, 9, 9, 728)	0	block7_sepconv1_bn[0][0]	
block7_sepconv2 (SeparableConv2	(None, 9, 9, 728)	536536	block7_sepconv2_act[0][0]	
block7_sepconv2_bn (BatchNormal	(None, 9, 9, 728)	2912	block7_sepconv2[0][0]	
block7_sepconv3_act (Activation	(None, 9, 9, 728)	0	block7_sepconv2_bn[0][0]	
block7_sepconv3 (SeparableConv2	(None, 9, 9, 728)	536536	block7_sepconv3_act[0][0]	
block7_sepconv3_bn (BatchNormal	(None, 9, 9, 728)	2912	block7_sepconv3[0][0]	
add_6 (Add)	(None, 9, 9, 728)	0	block7_sepconv3_bn[0][0] add_5[0][0]	
block8_sepconv1_act (Activation	(None, 9, 9, 728)	0	add_6[0][0]	
block8_sepconv1 (SeparableConv2	(None, 9, 9, 728)	536536	block8_sepconv1_act[0][0]	
block8_sepconv1_bn (BatchNormal	(None, 9, 9, 728)	2912	block8_sepconv1[0][0]	
block8_sepconv2_act (Activation	(None, 9, 9, 728)	0	block8_sepconv1_bn[0][0]	
block8_sepconv2 (SeparableConv2	(None, 9, 9, 728)	536536	block8_sepconv2_act[0][0]	
block8_sepconv2_bn (BatchNormal	(None, 9, 9, 728)	2912	block8_sepconv2[0][0]	
block8_sepconv3_act (Activation	(None, 9, 9, 728)	0	block8_sepconv2_bn[0][0]	
block8_sepconv3 (SeparableConv2	(None, 9, 9, 728)	536536	block8_sepconv3_act[0][0]	
block8_sepconv3_bn (BatchNormal	(None, 9, 9, 728)	2912	block8_sepconv3[0][0]	
add_7 (Add)	(None, 9, 9, 728)	0	block8_sepconv3_bn[0][0] add_6[0][0]	
block9_sepconv1_act (Activation	(None, 9, 9, 728)	0	add_7[0][0]	
block9_sepconv1 (SeparableConv2	(None, 9, 9, 728)	536536	block9_sepconv1_act[0][0]	
block9_sepconv1_bn (BatchNormal	(None, 9, 9, 728)	2912	block9_sepconv1[0][0]	
block9_sepconv2_act (Activation	(None, 9, 9, 728)	0	block9_sepconv1_bn[0][0]	
block9_sepconv2 (SeparableConv2	(None, 9, 9, 728)	536536	block9_sepconv2_act[0][0]	
block9_sepconv2_bn (BatchNormal	(None, 9, 9, 728)	2912	block9_sepconv2[0][0]	
block9_sepconv3_act (Activation	(None, 9, 9, 728)	0	block9_sepconv2_bn[0][0]	
block9_sepconv3 (SeparableConv2	(None, 9, 9, 728)	536536	block9_sepconv3_act[0][0]	
block9_sepconv3_bn (BatchNormal	(None, 9, 9, 728)	2912	block9_sepconv3[0][0]	
add_8 (Add)	(None, 9, 9, 728)	0	block9_sepconv3_bn[0][0] add_7[0][0]	
block10_sepconv1_act (Activatio	(None, 9, 9, 728)	0	add_8[0][0]	
block10_sepconv1 (SeparableConv	(None, 9, 9, 728)	536536	block10_sepconv1_act[0][0]	
block10_sepconv1_bn (BatchNorma	(None, 9, 9, 728)	2912	block10_sepconv1[0][0]	
block10_sepconv2_act (Activatio	(None, 9, 9, 728)	0	block10_sepconv1_bn[0][0]	
block10_sepconv2 (SeparableConv	(None, 9, 9, 728)	536536	block10_sepconv2_act[0][0]	
block10_sepconv2_bn (BatchNorma	(None, 9, 9, 728)	2912	block10_sepconv2[0][0]	
block10_sepconv3_act (Activatio	(None, 9, 9, 728)	0	block10_sepconv2_bn[0][0]	
block10_sepconv3 (SeparableConv	(None, 9, 9, 728)	536536	block10_sepconv3_act[0][0]	
block10_sepconv3_bn (BatchNorma	(None, 9, 9, 728)	2912	block10_sepconv3[0][0]	
add_9 (Add)	(None, 9, 9, 728)	0	block10_sepconv3_bn[0][0]	

				add_8[0][0]
block11_sepconv1_act (Activatio	(None, 9, 9, 728)	0	add_9[0][0]	
block11_sepconv1 (SeparableConv	(None, 9, 9, 728)	536536	block11_sepconv1_act[0][0]	
block11_sepconv1_bn (BatchNorma	(None, 9, 9, 728)	2912	block11_sepconv1[0][0]	
block11_sepconv2_act (Activatio	(None, 9, 9, 728)	0	block11_sepconv1_bn[0][0]	
block11_sepconv2 (SeparableConv	(None, 9, 9, 728)	536536	block11_sepconv2_act[0][0]	
block11_sepconv2_bn (BatchNorma	(None, 9, 9, 728)	2912	block11_sepconv2[0][0]	
block11_sepconv3_act (Activatio	(None, 9, 9, 728)	0	block11_sepconv2_bn[0][0]	
block11_sepconv3 (SeparableConv	(None, 9, 9, 728)	536536	block11_sepconv3_act[0][0]	
block11_sepconv3_bn (BatchNorma	(None, 9, 9, 728)	2912	block11_sepconv3[0][0]	
add_10 (Add)	(None, 9, 9, 728)	0	block11_sepconv3_bn[0][0] add_9[0][0]	
block12_sepconv1_act (Activatio	(None, 9, 9, 728)	0	add_10[0][0]	
block12_sepconv1 (SeparableConv	(None, 9, 9, 728)	536536	block12_sepconv1_act[0][0]	
block12_sepconv1_bn (BatchNorma	(None, 9, 9, 728)	2912	block12_sepconv1[0][0]	
block12_sepconv2_act (Activatio	(None, 9, 9, 728)	0	block12_sepconv1_bn[0][0]	
block12_sepconv2 (SeparableConv	(None, 9, 9, 728)	536536	block12_sepconv2_act[0][0]	
block12_sepconv2_bn (BatchNorma	(None, 9, 9, 728)	2912	block12_sepconv2[0][0]	
block12_sepconv3_act (Activatio	(None, 9, 9, 728)	0	block12_sepconv2_bn[0][0]	
block12_sepconv3 (SeparableConv	(None, 9, 9, 728)	536536	block12_sepconv3_act[0][0]	
block12_sepconv3_bn (BatchNorma	(None, 9, 9, 728)	2912	block12_sepconv3[0][0]	
add_11 (Add)	(None, 9, 9, 728)	0	block12_sepconv3_bn[0][0] add_10[0][0]	
block13_sepconv1_act (Activatio	(None, 9, 9, 728)	0	add_11[0][0]	
block13_sepconv1 (SeparableConv	(None, 9, 9, 728)	536536	block13_sepconv1_act[0][0]	
block13_sepconv1_bn (BatchNorma	(None, 9, 9, 728)	2912	block13_sepconv1[0][0]	
block13_sepconv2_act (Activatio	(None, 9, 9, 728)	0	block13_sepconv1_bn[0][0]	
block13_sepconv2 (SeparableConv	(None, 9, 9, 1024)	752024	block13_sepconv2_act[0][0]	
block13_sepconv2_bn (BatchNorma	(None, 9, 9, 1024)	4096	block13_sepconv2[0][0]	
conv2d_4 (Conv2D)	(None, 5, 5, 1024)	745472	add_11[0][0]	
block13_pool (MaxPooling2D)	(None, 5, 5, 1024)	0	block13_sepconv2_bn[0][0]	
batch_normalization_4 (BatchNor	(None, 5, 5, 1024)	4096	conv2d_4[0][0]	
add_12 (Add)	(None, 5, 5, 1024)	0	block13_pool[0][0] batch_normalization_4[0][0]	
block14_sepconv1 (SeparableConv	(None, 5, 5, 1536)	1582080	add_12[0][0]	
block14_sepconv1_bn (BatchNorma	(None, 5, 5, 1536)	6144	block14_sepconv1[0][0]	
block14_sepconv1_act (Activatio	(None, 5, 5, 1536)	0	block14_sepconv1_bn[0][0]	
block14_sepconv2 (SeparableConv	(None, 5, 5, 2048)	3159552	block14_sepconv1_act[0][0]	
block14_sepconv2_bn (BatchNorma	(None, 5, 5, 2048)	8192	block14_sepconv2[0][0]	
block14_sepconv2_act (Activatio	(None, 5, 5, 2048)	0	block14_sepconv2_bn[0][0]	

Total params: 20,861,480
Trainable params: 0
Non-trainable params: 20,861,480

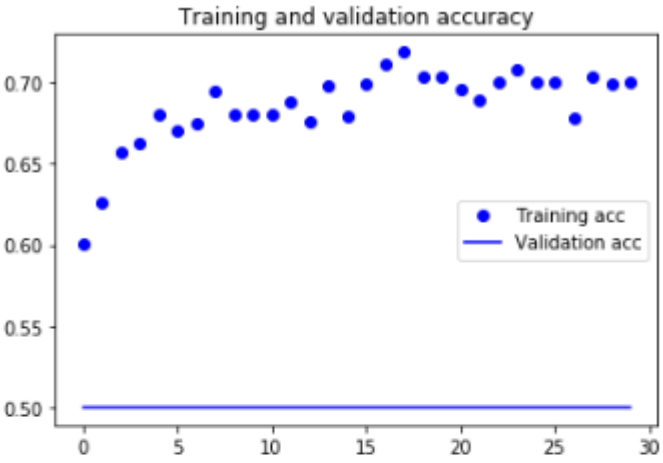
Xception/Fine Tuning	Train	Test
Accuray	.6737	.6650
Precision	.6703	.6773
Recall	.6837	.7250
F1-Score	.6770	.6840

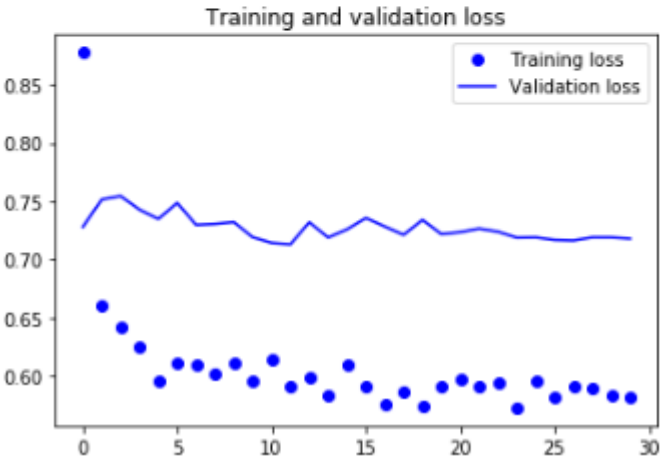


ResNet 50
30 Epochs
Loss: binary_crossentropy
Lr :0.0001

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Model)             (None, 5, 5, 2048)        23587712
_________________________________________________________________
flatten_2 (Flatten)          (None, 51200)             0
_________________________________________________________________
dense_3 (Dense)              (None, 256)               13107456
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 257
=================================================================
Total params: 36,695,425
Trainable params: 13,107,713
Non-trainable params: 23,587,712
_________________________________________________________________
_________________________________________________________________
```

ResNet 50	Train	Test
Accuray	.7056	.5
Precision	.7024	0
Recall	.6962	0
F1-Score	.6993	0





ResNet50 FINE TUNING

30 Epochs
Loss: binary_crossentropy
Lr :0.0001
Se permite entrenar a partir de activation_46

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 150, 150, 3)	0	
conv1_pad (ZeroPadding2D)	(None, 156, 156, 3)	0	input_1[0][0]
conv1 (Conv2D)	(None, 75, 75, 64)	9472	conv1_pad[0][0]
bn_conv1 (BatchNormalization)	(None, 75, 75, 64)	256	conv1[0][0]
activation_1 (Activation)	(None, 75, 75, 64)	0	bn_conv1[0][0]
pool1_pad (ZeroPadding2D)	(None, 77, 77, 64)	0	activation_1[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 38, 38, 64)	0	pool1_pad[0][0]
res2a_branch2a (Conv2D)	(None, 38, 38, 64)	4160	max_pooling2d_1[0][0]
bn2a_branch2a (BatchNormalizati	(None, 38, 38, 64)	256	res2a_branch2a[0][0]
activation_2 (Activation)	(None, 38, 38, 64)	0	bn2a_branch2a[0][0]
res2a_branch2b (Conv2D)	(None, 38, 38, 64)	36928	activation_2[0][0]
bn2a_branch2b (BatchNormalizati	(None, 38, 38, 64)	256	res2a_branch2b[0][0]
activation_3 (Activation)	(None, 38, 38, 64)	0	bn2a_branch2b[0][0]
res2a_branch2c (Conv2D)	(None, 38, 38, 256)	16640	activation_3[0][0]
res2a_branch1 (Conv2D)	(None, 38, 38, 256)	16640	max_pooling2d_1[0][0]
bn2a_branch2c (BatchNormalizati	(None, 38, 38, 256)	1024	res2a_branch2c[0][0]

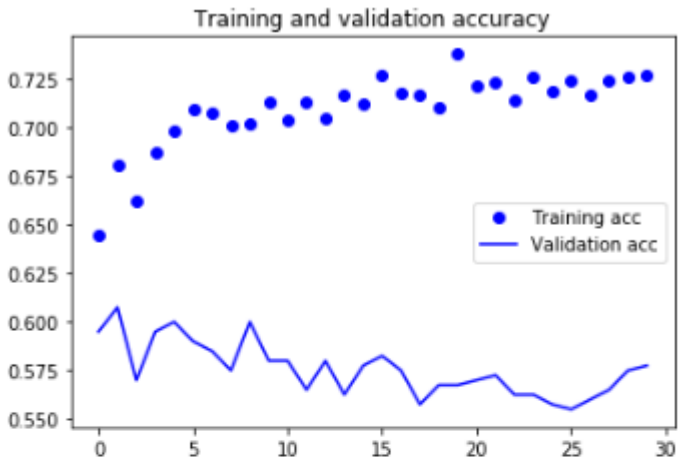
bn2a_branch1 (BatchNormalizatio	(None, 38, 38, 256)	1024	res2a_branch1[0][0]	
add_1 (Add)	(None, 38, 38, 256)	0	bn2a_branch2c[0][0] bn2a_branch1[0][0]	
activation_4 (Activation)	(None, 38, 38, 256)	0	add_1[0][0]	
res2b_branch2a (Conv2D)	(None, 38, 38, 64)	16448	activation_4[0][0]	
bn2b_branch2a (BatchNormalizati	(None, 38, 38, 64)	256	res2b_branch2a[0][0]	
activation_5 (Activation)	(None, 38, 38, 64)	0	bn2b_branch2a[0][0]	
res2b_branch2b (Conv2D)	(None, 38, 38, 64)	36928	activation_5[0][0]	
bn2b_branch2b (BatchNormalizati	(None, 38, 38, 64)	256	res2b_branch2b[0][0]	
activation_6 (Activation)	(None, 38, 38, 64)	0	bn2b_branch2b[0][0]	
res2b_branch2c (Conv2D)	(None, 38, 38, 256)	16640	activation_6[0][0]	
bn2b_branch2c (BatchNormalizati	(None, 38, 38, 256)	1024	res2b_branch2c[0][0]	
add_2 (Add)	(None, 38, 38, 256)	0	bn2b_branch2c[0][0] activation_4[0][0]	
activation_7 (Activation)	(None, 38, 38, 256)	0	add_2[0][0]	
res2c_branch2a (Conv2D)	(None, 38, 38, 64)	16448	activation_7[0][0]	
bn2c_branch2a (BatchNormalizati	(None, 38, 38, 64)	256	res2c_branch2a[0][0]	
activation_8 (Activation)	(None, 38, 38, 64)	0	bn2c_branch2a[0][0]	
res2c_branch2b (Conv2D)	(None, 38, 38, 64)	36928	activation_8[0][0]	
bn2c_branch2b (BatchNormalizati	(None, 38, 38, 64)	256	res2c_branch2b[0][0]	
activation_9 (Activation)	(None, 38, 38, 64)	0	bn2c_branch2b[0][0]	
res2c_branch2c (Conv2D)	(None, 38, 38, 256)	16640	activation_9[0][0]	
bn2c_branch2c (BatchNormalizati	(None, 38, 38, 256)	1024	res2c_branch2c[0][0]	
add_3 (Add)	(None, 38, 38, 256)	0	bn2c_branch2c[0][0] activation_7[0][0]	
activation_10 (Activation)	(None, 38, 38, 256)	0	add_3[0][0]	
res3a_branch2a (Conv2D)	(None, 19, 19, 128)	32896	activation_10[0][0]	
bn3a_branch2a (BatchNormalizati	(None, 19, 19, 128)	512	res3a_branch2a[0][0]	
activation_11 (Activation)	(None, 19, 19, 128)	0	bn3a_branch2a[0][0]	
res3a_branch2b (Conv2D)	(None, 19, 19, 128)	147584	activation_11[0][0]	
bn3a_branch2b (BatchNormalizati	(None, 19, 19, 128)	512	res3a_branch2b[0][0]	
activation_12 (Activation)	(None, 19, 19, 128)	0	bn3a_branch2b[0][0]	
res3a_branch2c (Conv2D)	(None, 19, 19, 512)	66048	activation_12[0][0]	
res3a_branch1 (Conv2D)	(None, 19, 19, 512)	131584	activation_10[0][0]	
bn3a_branch2c (BatchNormalizati	(None, 19, 19, 512)	2048	res3a_branch2c[0][0]	
bn3a_branch1 (BatchNormalizatio	(None, 19, 19, 512)	2048	res3a_branch1[0][0]	
add_4 (Add)	(None, 19, 19, 512)	0	bn3a_branch2c[0][0] bn3a_branch1[0][0]	
activation_13 (Activation)	(None, 19, 19, 512)	0	add_4[0][0]	
res3b_branch2a (Conv2D)	(None, 19, 19, 128)	65664	activation_13[0][0]	
bn3b_branch2a (BatchNormalizati	(None, 19, 19, 128)	512	res3b_branch2a[0][0]	
activation_14 (Activation)	(None, 19, 19, 128)	0	bn3b_branch2a[0][0]	
res3b_branch2b (Conv2D)	(None, 19, 19, 128)	147584	activation_14[0][0]	
bn3b_branch2b (BatchNormalizati	(None, 19, 19, 128)	512	res3b_branch2b[0][0]	
activation_15 (Activation)	(None, 19, 19, 128)	0	bn3b_branch2b[0][0]	
res3b_branch2c (Conv2D)	(None, 19, 19, 512)	66048	activation_15[0][0]	
bn3b_branch2c (BatchNormalizati	(None, 19, 19, 512)	2048	res3b_branch2c[0][0]	
add_5 (Add)	(None, 19, 19, 512)	0	bn3b_branch2c[0][0] activation_13[0][0]	
activation_16 (Activation)	(None, 19, 19, 512)	0	add_5[0][0]	
res3c_branch2a (Conv2D)	(None, 19, 19, 128)	65664	activation_16[0][0]	
bn3c_branch2a (BatchNormalizati	(None, 19, 19, 128)	512	res3c_branch2a[0][0]	
activation_17 (Activation)	(None, 19, 19, 128)	0	bn3c_branch2a[0][0]	
res3c_branch2b (Conv2D)	(None, 19, 19, 128)	147584	activation_17[0][0]	

bn3c_branch2b (BatchNormalizati	(None, 19, 19, 128)	512	res3c_branch2b[0][0]
activation_18 (Activation)	(None, 19, 19, 128)	0	bn3c_branch2b[0][0]
res3c_branch2c (Conv2D)	(None, 19, 19, 512)	66048	activation_18[0][0]
bn3c_branch2c (BatchNormalizati	(None, 19, 19, 512)	2048	res3c_branch2c[0][0]
add_6 (Add)	(None, 19, 19, 512)	0	bn3c_branch2c[0][0] activation_16[0][0]
activation_19 (Activation)	(None, 19, 19, 512)	0	add_6[0][0]
res3d_branch2a (Conv2D)	(None, 19, 19, 128)	65664	activation_19[0][0]
bn3d_branch2a (BatchNormalizati	(None, 19, 19, 128)	512	res3d_branch2a[0][0]
activation_20 (Activation)	(None, 19, 19, 128)	0	bn3d_branch2a[0][0]
res3d_branch2b (Conv2D)	(None, 19, 19, 128)	147584	activation_20[0][0]
bn3d_branch2b (BatchNormalizati	(None, 19, 19, 128)	512	res3d_branch2b[0][0]
activation_21 (Activation)	(None, 19, 19, 128)	0	bn3d_branch2b[0][0]
res3d_branch2c (Conv2D)	(None, 19, 19, 512)	66048	activation_21[0][0]
bn3d_branch2c (BatchNormalizati	(None, 19, 19, 512)	2048	res3d_branch2c[0][0]
add_7 (Add)	(None, 19, 19, 512)	0	bn3d_branch2c[0][0] activation_19[0][0]
activation_22 (Activation)	(None, 19, 19, 512)	0	add_7[0][0]
res4a_branch2a (Conv2D)	(None, 10, 10, 256)	131328	activation_22[0][0]
bn4a_branch2a (BatchNormalizati	(None, 10, 10, 256)	1024	res4a_branch2a[0][0]
activation_23 (Activation)	(None, 10, 10, 256)	0	bn4a_branch2a[0][0]
res4a_branch2b (Conv2D)	(None, 10, 10, 256)	590080	activation_23[0][0]
bn4a_branch2b (BatchNormalizati	(None, 10, 10, 256)	1024	res4a_branch2b[0][0]
activation_24 (Activation)	(None, 10, 10, 256)	0	bn4a_branch2b[0][0]
res4a_branch2c (Conv2D)	(None, 10, 10, 1024)	263168	activation_24[0][0]
res4a_branch1 (Conv2D)	(None, 10, 10, 1024)	525312	activation_22[0][0]
bn4a_branch2c (BatchNormalizati	(None, 10, 10, 1024)	4096	res4a_branch2c[0][0]
bn4a_branch1 (BatchNormalizatio	(None, 10, 10, 1024)	4096	res4a_branch1[0][0]
add_8 (Add)	(None, 10, 10, 1024)	0	bn4a_branch2c[0][0] bn4a_branch1[0][0]
activation_25 (Activation)	(None, 10, 10, 1024)	0	add_8[0][0]
res4b_branch2a (Conv2D)	(None, 10, 10, 256)	262400	activation_25[0][0]
bn4b_branch2a (BatchNormalizati	(None, 10, 10, 256)	1024	res4b_branch2a[0][0]
activation_26 (Activation)	(None, 10, 10, 256)	0	bn4b_branch2a[0][0]
res4b_branch2b (Conv2D)	(None, 10, 10, 256)	590080	activation_26[0][0]
bn4b_branch2b (BatchNormalizati	(None, 10, 10, 256)	1024	res4b_branch2b[0][0]
activation_27 (Activation)	(None, 10, 10, 256)	0	bn4b_branch2b[0][0]
res4b_branch2c (Conv2D)	(None, 10, 10, 1024)	263168	activation_27[0][0]
bn4b_branch2c (BatchNormalizati	(None, 10, 10, 1024)	4096	res4b_branch2c[0][0]
add_9 (Add)	(None, 10, 10, 1024)	0	bn4b_branch2c[0][0] activation_25[0][0]
activation_28 (Activation)	(None, 10, 10, 1024)	0	add_9[0][0]
res4c_branch2a (Conv2D)	(None, 10, 10, 256)	262400	activation_28[0][0]
bn4c_branch2a (BatchNormalizati	(None, 10, 10, 256)	1024	res4c_branch2a[0][0]
activation_29 (Activation)	(None, 10, 10, 256)	0	bn4c_branch2a[0][0]
res4c_branch2b (Conv2D)	(None, 10, 10, 256)	590080	activation_29[0][0]
bn4c_branch2b (BatchNormalizati	(None, 10, 10, 256)	1024	res4c_branch2b[0][0]
activation_30 (Activation)	(None, 10, 10, 256)	0	bn4c_branch2b[0][0]
res4c_branch2c (Conv2D)	(None, 10, 10, 1024)	263168	activation_30[0][0]
bn4c_branch2c (BatchNormalizati	(None, 10, 10, 1024)	4096	res4c_branch2c[0][0]
add_10 (Add)	(None, 10, 10, 1024)	0	bn4c_branch2c[0][0] activation_28[0][0]
activation_31 (Activation)	(None, 10, 10, 1024)	0	add_10[0][0]

res4d_branch2a (Conv2D)	(None, 10, 10, 256)	262400	activation_31[0][0]
bn4d_branch2a (BatchNormalizati	(None, 10, 10, 256)	1024	res4d_branch2a[0][0]
activation_32 (Activation)	(None, 10, 10, 256)	0	bn4d_branch2a[0][0]
res4d_branch2b (Conv2D)	(None, 10, 10, 256)	590080	activation_32[0][0]
bn4d_branch2b (BatchNormalizati	(None, 10, 10, 256)	1024	res4d_branch2b[0][0]
activation_33 (Activation)	(None, 10, 10, 256)	0	bn4d_branch2b[0][0]
res4d_branch2c (Conv2D)	(None, 10, 10, 1024)	263168	activation_33[0][0]
bn4d_branch2c (BatchNormalizati	(None, 10, 10, 1024)	4096	res4d_branch2c[0][0]
add_11 (Add)	(None, 10, 10, 1024)	0	bn4d_branch2c[0][0] activation_31[0][0]
activation_34 (Activation)	(None, 10, 10, 1024)	0	add_11[0][0]
res4e_branch2a (Conv2D)	(None, 10, 10, 256)	262400	activation_34[0][0]
bn4e_branch2a (BatchNormalizati	(None, 10, 10, 256)	1024	res4e_branch2a[0][0]
activation_35 (Activation)	(None, 10, 10, 256)	0	bn4e_branch2a[0][0]
res4e_branch2b (Conv2D)	(None, 10, 10, 256)	590080	activation_35[0][0]
bn4e_branch2b (BatchNormalizati	(None, 10, 10, 256)	1024	res4e_branch2b[0][0]
activation_36 (Activation)	(None, 10, 10, 256)	0	bn4e_branch2b[0][0]
res4e_branch2c (Conv2D)	(None, 10, 10, 1024)	263168	activation_36[0][0]
bn4e_branch2c (BatchNormalizati	(None, 10, 10, 1024)	4096	res4e_branch2c[0][0]
add_12 (Add)	(None, 10, 10, 1024)	0	bn4e_branch2c[0][0] activation_34[0][0]
activation_37 (Activation)	(None, 10, 10, 1024)	0	add_12[0][0]
res4f_branch2a (Conv2D)	(None, 10, 10, 256)	262400	activation_37[0][0]
bn4f_branch2a (BatchNormalizati	(None, 10, 10, 256)	1024	res4f_branch2a[0][0]
activation_38 (Activation)	(None, 10, 10, 256)	0	bn4f_branch2a[0][0]
res4f_branch2b (Conv2D)	(None, 10, 10, 256)	590080	activation_38[0][0]
bn4f_branch2b (BatchNormalizati	(None, 10, 10, 256)	1024	res4f_branch2b[0][0]
activation_39 (Activation)	(None, 10, 10, 256)	0	bn4f_branch2b[0][0]
res4f_branch2c (Conv2D)	(None, 10, 10, 1024)	263168	activation_39[0][0]
bn4f_branch2c (BatchNormalizati	(None, 10, 10, 1024)	4096	res4f_branch2c[0][0]
add_13 (Add)	(None, 10, 10, 1024)	0	bn4f_branch2c[0][0] activation_37[0][0]
activation_40 (Activation)	(None, 10, 10, 1024)	0	add_13[0][0]
res5a_branch2a (Conv2D)	(None, 5, 5, 512)	524800	activation_40[0][0]
bn5a_branch2a (BatchNormalizati	(None, 5, 5, 512)	2048	res5a_branch2a[0][0]
activation_41 (Activation)	(None, 5, 5, 512)	0	bn5a_branch2a[0][0]
res5a_branch2b (Conv2D)	(None, 5, 5, 512)	2359808	activation_41[0][0]
bn5a_branch2b (BatchNormalizati	(None, 5, 5, 512)	2048	res5a_branch2b[0][0]
activation_42 (Activation)	(None, 5, 5, 512)	0	bn5a_branch2b[0][0]
res5a_branch2c (Conv2D)	(None, 5, 5, 2048)	1050624	activation_42[0][0]
res5a_branch1 (Conv2D)	(None, 5, 5, 2048)	2099200	activation_40[0][0]
bn5a_branch2c (BatchNormalizati	(None, 5, 5, 2048)	8192	res5a_branch2c[0][0]
bn5a_branch1 (BatchNormalizatio	(None, 5, 5, 2048)	8192	res5a_branch1[0][0]
add_14 (Add)	(None, 5, 5, 2048)	0	bn5a_branch2c[0][0] bn5a_branch1[0][0]
activation_43 (Activation)	(None, 5, 5, 2048)	0	add_14[0][0]
res5b_branch2a (Conv2D)	(None, 5, 5, 512)	1049088	activation_43[0][0]
bn5b_branch2a (BatchNormalizati	(None, 5, 5, 512)	2048	res5b_branch2a[0][0]
activation_44 (Activation)	(None, 5, 5, 512)	0	bn5b_branch2a[0][0]
res5b_branch2b (Conv2D)	(None, 5, 5, 512)	2359808	activation_44[0][0]
bn5b_branch2b (BatchNormalizati	(None, 5, 5, 512)	2048	res5b_branch2b[0][0]
activation_45 (Activation)	(None, 5, 5, 512)	0	bn5b_branch2b[0][0]
res5b_branch2c (Conv2D)	(None, 5, 5, 2048)	1050624	activation_45[0][0]

```
bn5b_branch2c (BatchNormalizati (None, 5, 5, 2048)   8192        res5b_branch2c[0][0]
__________________________________________________________________________________________________
add_15 (Add)                    (None, 5, 5, 2048)   0           bn5b_branch2c[0][0]
                                                                 activation_43[0][0]
__________________________________________________________________________________________________
activation_46 (Activation)      (None, 5, 5, 2048)   0           add_15[0][0]
__________________________________________________________________________________________________
res5c_branch2a (Conv2D)         (None, 5, 5, 512)    1049088     activation_46[0][0]
__________________________________________________________________________________________________
bn5c_branch2a (BatchNormalizati (None, 5, 5, 512)    2048        res5c_branch2a[0][0]
__________________________________________________________________________________________________
activation_47 (Activation)      (None, 5, 5, 512)    0           bn5c_branch2a[0][0]
__________________________________________________________________________________________________
res5c_branch2b (Conv2D)         (None, 5, 5, 512)    2359808     activation_47[0][0]
__________________________________________________________________________________________________
bn5c_branch2b (BatchNormalizati (None, 5, 5, 512)    2048        res5c_branch2b[0][0]
__________________________________________________________________________________________________
activation_48 (Activation)      (None, 5, 5, 512)    0           bn5c_branch2b[0][0]
__________________________________________________________________________________________________
res5c_branch2c (Conv2D)         (None, 5, 5, 2048)   1050624     activation_48[0][0]
__________________________________________________________________________________________________
bn5c_branch2c (BatchNormalizati (None, 5, 5, 2048)   8192        res5c_branch2c[0][0]
__________________________________________________________________________________________________
add_16 (Add)                    (None, 5, 5, 2048)   0           bn5c_branch2c[0][0]
                                                                 activation_46[0][0]
__________________________________________________________________________________________________
activation_49 (Activation)      (None, 5, 5, 2048)   0           add_16[0][0]
==================================================================================================
Total params: 23,587,712
Trainable params: 0
Non-trainable params: 23,587,712
```

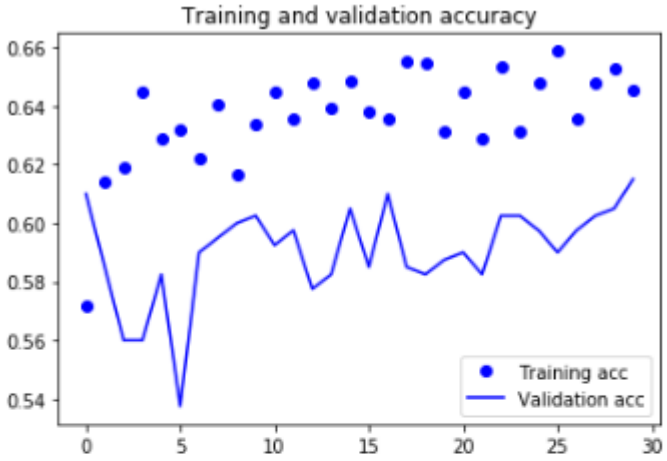
ResNet 50/Fine Tuning	Train	Test
Accuray	.7269	.5775
Precision	.7330	.5447
Recall	.7137	.9450
F1-Score	.7232	.6910



Inception V3
30 Epochs
Loss: binary_crossentropy
Lr :0.0001

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_v3 (Model)         (None, 3, 3, 2048)        21802784
_________________________________________________________________
flatten_1 (Flatten)          (None, 18432)             0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               4718848
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 257
=================================================================
Total params: 26,521,889
Trainable params: 4,719,105
Non-trainable params: 21,802,784
```

Inception V3	Train	Test
Accuray	.6456	.6150
Precision	.6416	.5858
Recall	.6600	.7850
F1-Score	.6506	.6709



Inception V3 FINE TUNING

30 Epochs
Loss: binary_crossentropy
Lr :0.0001
Se permite entrenar a partir de conv2d_86

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 150, 150, 3)	0	
conv2d_1 (Conv2D)	(None, 74, 74, 32)	864	input_1[0][0]
batch_normalization_1 (BatchNor	(None, 74, 74, 32)	96	conv2d_1[0][0]
activation_1 (Activation)	(None, 74, 74, 32)	0	batch_normalization_1[0][0]
conv2d_2 (Conv2D)	(None, 72, 72, 32)	9216	activation_1[0][0]
batch_normalization_2 (BatchNor	(None, 72, 72, 32)	96	conv2d_2[0][0]
activation_2 (Activation)	(None, 72, 72, 32)	0	batch_normalization_2[0][0]
conv2d_3 (Conv2D)	(None, 72, 72, 64)	18432	activation_2[0][0]
batch_normalization_3 (BatchNor	(None, 72, 72, 64)	192	conv2d_3[0][0]
activation_3 (Activation)	(None, 72, 72, 64)	0	batch_normalization_3[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 35, 35, 64)	0	activation_3[0][0]
conv2d_4 (Conv2D)	(None, 35, 35, 80)	5120	max_pooling2d_1[0][0]
batch_normalization_4 (BatchNor	(None, 35, 35, 80)	240	conv2d_4[0][0]
activation_4 (Activation)	(None, 35, 35, 80)	0	batch_normalization_4[0][0]
conv2d_5 (Conv2D)	(None, 33, 33, 192)	138240	activation_4[0][0]
batch_normalization_5 (BatchNor	(None, 33, 33, 192)	576	conv2d_5[0][0]
activation_5 (Activation)	(None, 33, 33, 192)	0	batch_normalization_5[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 16, 16, 192)	0	activation_5[0][0]
conv2d_9 (Conv2D)	(None, 16, 16, 64)	12288	max_pooling2d_2[0][0]
batch_normalization_9 (BatchNor	(None, 16, 16, 64)	192	conv2d_9[0][0]
activation_9 (Activation)	(None, 16, 16, 64)	0	batch_normalization_9[0][0]
conv2d_7 (Conv2D)	(None, 16, 16, 48)	9216	max_pooling2d_2[0][0]
conv2d_10 (Conv2D)	(None, 16, 16, 96)	55296	activation_9[0][0]
batch_normalization_7 (BatchNor	(None, 16, 16, 48)	144	conv2d_7[0][0]
batch_normalization_10 (BatchNo	(None, 16, 16, 96)	288	conv2d_10[0][0]
activation_7 (Activation)	(None, 16, 16, 48)	0	batch_normalization_7[0][0]
activation_10 (Activation)	(None, 16, 16, 96)	0	batch_normalization_10[0][0]
average_pooling2d_1 (AveragePoo	(None, 16, 16, 192)	0	max_pooling2d_2[0][0]
conv2d_6 (Conv2D)	(None, 16, 16, 64)	12288	max_pooling2d_2[0][0]
conv2d_8 (Conv2D)	(None, 16, 16, 64)	76800	activation_7[0][0]
conv2d_11 (Conv2D)	(None, 16, 16, 96)	82944	activation_10[0][0]
conv2d_12 (Conv2D)	(None, 16, 16, 32)	6144	average_pooling2d_1[0][0]
batch_normalization_6 (BatchNor	(None, 16, 16, 64)	192	conv2d_6[0][0]
batch_normalization_8 (BatchNor	(None, 16, 16, 64)	192	conv2d_8[0][0]
batch_normalization_11 (BatchNo	(None, 16, 16, 96)	288	conv2d_11[0][0]
batch_normalization_12 (BatchNo	(None, 16, 16, 32)	96	conv2d_12[0][0]
activation_6 (Activation)	(None, 16, 16, 64)	0	batch_normalization_6[0][0]
activation_8 (Activation)	(None, 16, 16, 64)	0	batch_normalization_8[0][0]
activation_11 (Activation)	(None, 16, 16, 96)	0	batch_normalization_11[0][0]
activation_12 (Activation)	(None, 16, 16, 32)	0	batch_normalization_12[0][0]
mixed0 (Concatenate)	(None, 16, 16, 256)	0	activation_6[0][0] activation_8[0][0] activation_11[0][0] activation_12[0][0]
conv2d_16 (Conv2D)	(None, 16, 16, 64)	16384	mixed0[0][0]

batch_normalization_16 (BatchNo	(None, 16, 16, 64)	192	conv2d_16[0][0]
activation_16 (Activation)	(None, 16, 16, 64)	0	batch_normalization_16[0][0]
conv2d_14 (Conv2D)	(None, 16, 16, 48)	12288	mixed0[0][0]
conv2d_17 (Conv2D)	(None, 16, 16, 96)	55296	activation_16[0][0]
batch_normalization_14 (BatchNo	(None, 16, 16, 48)	144	conv2d_14[0][0]
batch_normalization_17 (BatchNo	(None, 16, 16, 96)	288	conv2d_17[0][0]
activation_14 (Activation)	(None, 16, 16, 48)	0	batch_normalization_14[0][0]
activation_17 (Activation)	(None, 16, 16, 96)	0	batch_normalization_17[0][0]
average_pooling2d_2 (AveragePoo	(None, 16, 16, 256)	0	mixed0[0][0]
conv2d_13 (Conv2D)	(None, 16, 16, 64)	16384	mixed0[0][0]
conv2d_15 (Conv2D)	(None, 16, 16, 64)	76800	activation_14[0][0]
conv2d_18 (Conv2D)	(None, 16, 16, 96)	82944	activation_17[0][0]
conv2d_19 (Conv2D)	(None, 16, 16, 64)	16384	average_pooling2d_2[0][0]
batch_normalization_13 (BatchNo	(None, 16, 16, 64)	192	conv2d_13[0][0]
batch_normalization_15 (BatchNo	(None, 16, 16, 64)	192	conv2d_15[0][0]
batch_normalization_18 (BatchNo	(None, 16, 16, 96)	288	conv2d_18[0][0]
batch_normalization_19 (BatchNo	(None, 16, 16, 64)	192	conv2d_19[0][0]
activation_13 (Activation)	(None, 16, 16, 64)	0	batch_normalization_13[0][0]
activation_15 (Activation)	(None, 16, 16, 64)	0	batch_normalization_15[0][0]
activation_18 (Activation)	(None, 16, 16, 96)	0	batch_normalization_18[0][0]
activation_19 (Activation)	(None, 16, 16, 64)	0	batch_normalization_19[0][0]
mixed1 (Concatenate)	(None, 16, 16, 288)	0	activation_13[0][0] activation_15[0][0] activation_18[0][0] activation_19[0][0]
conv2d_23 (Conv2D)	(None, 16, 16, 64)	18432	mixed1[0][0]
batch_normalization_23 (BatchNo	(None, 16, 16, 64)	192	conv2d_23[0][0]
activation_23 (Activation)	(None, 16, 16, 64)	0	batch_normalization_23[0][0]
conv2d_21 (Conv2D)	(None, 16, 16, 48)	13824	mixed1[0][0]
conv2d_24 (Conv2D)	(None, 16, 16, 96)	55296	activation_23[0][0]
batch_normalization_21 (BatchNo	(None, 16, 16, 48)	144	conv2d_21[0][0]
batch_normalization_24 (BatchNo	(None, 16, 16, 96)	288	conv2d_24[0][0]
activation_21 (Activation)	(None, 16, 16, 48)	0	batch_normalization_21[0][0]
activation_24 (Activation)	(None, 16, 16, 96)	0	batch_normalization_24[0][0]
average_pooling2d_3 (AveragePoo	(None, 16, 16, 288)	0	mixed1[0][0]
conv2d_20 (Conv2D)	(None, 16, 16, 64)	18432	mixed1[0][0]
conv2d_22 (Conv2D)	(None, 16, 16, 64)	76800	activation_21[0][0]
conv2d_25 (Conv2D)	(None, 16, 16, 96)	82944	activation_24[0][0]
conv2d_26 (Conv2D)	(None, 16, 16, 64)	18432	average_pooling2d_3[0][0]
batch_normalization_20 (BatchNo	(None, 16, 16, 64)	192	conv2d_20[0][0]
batch_normalization_22 (BatchNo	(None, 16, 16, 64)	192	conv2d_22[0][0]
batch_normalization_25 (BatchNo	(None, 16, 16, 96)	288	conv2d_25[0][0]
batch_normalization_26 (BatchNo	(None, 16, 16, 64)	192	conv2d_26[0][0]
activation_20 (Activation)	(None, 16, 16, 64)	0	batch_normalization_20[0][0]
activation_22 (Activation)	(None, 16, 16, 64)	0	batch_normalization_22[0][0]
activation_25 (Activation)	(None, 16, 16, 96)	0	batch_normalization_25[0][0]
activation_26 (Activation)	(None, 16, 16, 64)	0	batch_normalization_26[0][0]
mixed2 (Concatenate)	(None, 16, 16, 288)	0	activation_20[0][0] activation_22[0][0] activation_25[0][0] activation_26[0][0]
conv2d_28 (Conv2D)	(None, 16, 16, 64)	18432	mixed2[0][0]
batch_normalization_28 (BatchNo	(None, 16, 16, 64)	192	conv2d_28[0][0]
activation_28 (Activation)	(None, 16, 16, 64)	0	batch_normalization_28[0][0]

conv2d_29 (Conv2D)	(None, 16, 16, 96)	55296	activation_28[0][0]
batch_normalization_29 (BatchNo	(None, 16, 16, 96)	288	conv2d_29[0][0]
activation_29 (Activation)	(None, 16, 16, 96)	0	batch_normalization_29[0][0]
conv2d_27 (Conv2D)	(None, 7, 7, 384)	995328	mixed2[0][0]
conv2d_30 (Conv2D)	(None, 7, 7, 96)	82944	activation_29[0][0]
batch_normalization_27 (BatchNo	(None, 7, 7, 384)	1152	conv2d_27[0][0]
batch_normalization_30 (BatchNo	(None, 7, 7, 96)	288	conv2d_30[0][0]
activation_27 (Activation)	(None, 7, 7, 384)	0	batch_normalization_27[0][0]
activation_30 (Activation)	(None, 7, 7, 96)	0	batch_normalization_30[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 7, 7, 288)	0	mixed2[0][0]
mixed3 (Concatenate)	(None, 7, 7, 768)	0	activation_27[0][0] activation_30[0][0] max_pooling2d_3[0][0]
conv2d_35 (Conv2D)	(None, 7, 7, 128)	98304	mixed3[0][0]
batch_normalization_35 (BatchNo	(None, 7, 7, 128)	384	conv2d_35[0][0]
activation_35 (Activation)	(None, 7, 7, 128)	0	batch_normalization_35[0][0]
conv2d_36 (Conv2D)	(None, 7, 7, 128)	114688	activation_35[0][0]
batch_normalization_36 (BatchNo	(None, 7, 7, 128)	384	conv2d_36[0][0]
activation_36 (Activation)	(None, 7, 7, 128)	0	batch_normalization_36[0][0]
conv2d_32 (Conv2D)	(None, 7, 7, 128)	98304	mixed3[0][0]
conv2d_37 (Conv2D)	(None, 7, 7, 128)	114688	activation_36[0][0]
batch_normalization_32 (BatchNo	(None, 7, 7, 128)	384	conv2d_32[0][0]
batch_normalization_37 (BatchNo	(None, 7, 7, 128)	384	conv2d_37[0][0]
activation_32 (Activation)	(None, 7, 7, 128)	0	batch_normalization_32[0][0]
activation_37 (Activation)	(None, 7, 7, 128)	0	batch_normalization_37[0][0]
conv2d_33 (Conv2D)	(None, 7, 7, 128)	114688	activation_32[0][0]
conv2d_38 (Conv2D)	(None, 7, 7, 128)	114688	activation_37[0][0]
batch_normalization_33 (BatchNo	(None, 7, 7, 128)	384	conv2d_33[0][0]
batch_normalization_38 (BatchNo	(None, 7, 7, 128)	384	conv2d_38[0][0]
activation_33 (Activation)	(None, 7, 7, 128)	0	batch_normalization_33[0][0]
activation_38 (Activation)	(None, 7, 7, 128)	0	batch_normalization_38[0][0]
average_pooling2d_4 (AveragePoo	(None, 7, 7, 768)	0	mixed3[0][0]
conv2d_31 (Conv2D)	(None, 7, 7, 192)	147456	mixed3[0][0]
conv2d_34 (Conv2D)	(None, 7, 7, 192)	172032	activation_33[0][0]
conv2d_39 (Conv2D)	(None, 7, 7, 192)	172032	activation_38[0][0]
conv2d_40 (Conv2D)	(None, 7, 7, 192)	147456	average_pooling2d_4[0][0]
batch_normalization_31 (BatchNo	(None, 7, 7, 192)	576	conv2d_31[0][0]
batch_normalization_34 (BatchNo	(None, 7, 7, 192)	576	conv2d_34[0][0]
batch_normalization_39 (BatchNo	(None, 7, 7, 192)	576	conv2d_39[0][0]
batch_normalization_40 (BatchNo	(None, 7, 7, 192)	576	conv2d_40[0][0]
activation_31 (Activation)	(None, 7, 7, 192)	0	batch_normalization_31[0][0]
activation_34 (Activation)	(None, 7, 7, 192)	0	batch_normalization_34[0][0]
activation_39 (Activation)	(None, 7, 7, 192)	0	batch_normalization_39[0][0]
activation_40 (Activation)	(None, 7, 7, 192)	0	batch_normalization_40[0][0]
mixed4 (Concatenate)	(None, 7, 7, 768)	0	activation_31[0][0] activation_34[0][0] activation_39[0][0] activation_40[0][0]
conv2d_45 (Conv2D)	(None, 7, 7, 160)	122880	mixed4[0][0]
batch_normalization_45 (BatchNo	(None, 7, 7, 160)	480	conv2d_45[0][0]
activation_45 (Activation)	(None, 7, 7, 160)	0	batch_normalization_45[0][0]
conv2d_46 (Conv2D)	(None, 7, 7, 160)	179200	activation_45[0][0]
batch_normalization_46 (BatchNo	(None, 7, 7, 160)	480	conv2d_46[0][0]

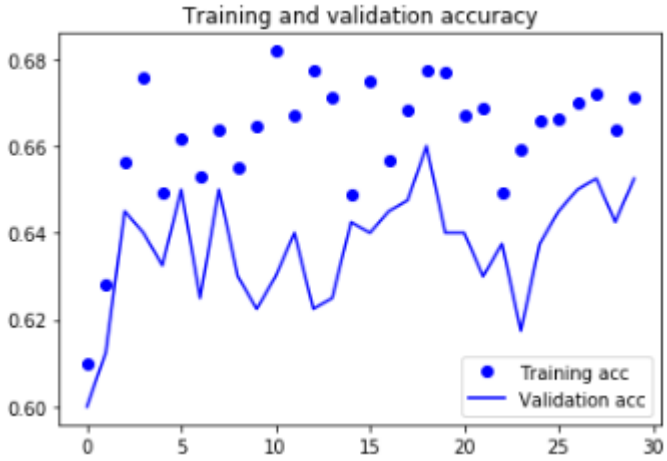
activation_46 (Activation)	(None, 7, 7, 160)	0	batch_normalization_46[0][0]
conv2d_42 (Conv2D)	(None, 7, 7, 160)	122880	mixed4[0][0]
conv2d_47 (Conv2D)	(None, 7, 7, 160)	179200	activation_46[0][0]
batch_normalization_42 (BatchNo	(None, 7, 7, 160)	480	conv2d_42[0][0]
batch_normalization_47 (BatchNo	(None, 7, 7, 160)	480	conv2d_47[0][0]
activation_42 (Activation)	(None, 7, 7, 160)	0	batch_normalization_42[0][0]
activation_47 (Activation)	(None, 7, 7, 160)	0	batch_normalization_47[0][0]
conv2d_43 (Conv2D)	(None, 7, 7, 160)	179200	activation_42[0][0]
conv2d_48 (Conv2D)	(None, 7, 7, 160)	179200	activation_47[0][0]
batch_normalization_43 (BatchNo	(None, 7, 7, 160)	480	conv2d_43[0][0]
batch_normalization_48 (BatchNo	(None, 7, 7, 160)	480	conv2d_48[0][0]
activation_43 (Activation)	(None, 7, 7, 160)	0	batch_normalization_43[0][0]
activation_48 (Activation)	(None, 7, 7, 160)	0	batch_normalization_48[0][0]
average_pooling2d_5 (AveragePoo	(None, 7, 7, 768)	0	mixed4[0][0]
conv2d_41 (Conv2D)	(None, 7, 7, 192)	147456	mixed4[0][0]
conv2d_44 (Conv2D)	(None, 7, 7, 192)	215040	activation_43[0][0]
conv2d_49 (Conv2D)	(None, 7, 7, 192)	215040	activation_48[0][0]
conv2d_50 (Conv2D)	(None, 7, 7, 192)	147456	average_pooling2d_5[0][0]
batch_normalization_41 (BatchNo	(None, 7, 7, 192)	576	conv2d_41[0][0]
batch_normalization_44 (BatchNo	(None, 7, 7, 192)	576	conv2d_44[0][0]
batch_normalization_49 (BatchNo	(None, 7, 7, 192)	576	conv2d_49[0][0]
batch_normalization_50 (BatchNo	(None, 7, 7, 192)	576	conv2d_50[0][0]
activation_41 (Activation)	(None, 7, 7, 192)	0	batch_normalization_41[0][0]
activation_44 (Activation)	(None, 7, 7, 192)	0	batch_normalization_44[0][0]
activation_49 (Activation)	(None, 7, 7, 192)	0	batch_normalization_49[0][0]
activation_50 (Activation)	(None, 7, 7, 192)	0	batch_normalization_50[0][0]
mixed5 (Concatenate)	(None, 7, 7, 768)	0	activation_41[0][0] activation_44[0][0] activation_49[0][0] activation_50[0][0]
conv2d_55 (Conv2D)	(None, 7, 7, 160)	122880	mixed5[0][0]
batch_normalization_55 (BatchNo	(None, 7, 7, 160)	480	conv2d_55[0][0]
activation_55 (Activation)	(None, 7, 7, 160)	0	batch_normalization_55[0][0]
conv2d_56 (Conv2D)	(None, 7, 7, 160)	179200	activation_55[0][0]
batch_normalization_56 (BatchNo	(None, 7, 7, 160)	480	conv2d_56[0][0]
activation_56 (Activation)	(None, 7, 7, 160)	0	batch_normalization_56[0][0]
conv2d_52 (Conv2D)	(None, 7, 7, 160)	122880	mixed5[0][0]
conv2d_57 (Conv2D)	(None, 7, 7, 160)	179200	activation_56[0][0]
batch_normalization_52 (BatchNo	(None, 7, 7, 160)	480	conv2d_52[0][0]
batch_normalization_57 (BatchNo	(None, 7, 7, 160)	480	conv2d_57[0][0]
activation_52 (Activation)	(None, 7, 7, 160)	0	batch_normalization_52[0][0]
activation_57 (Activation)	(None, 7, 7, 160)	0	batch_normalization_57[0][0]
conv2d_53 (Conv2D)	(None, 7, 7, 160)	179200	activation_52[0][0]
conv2d_58 (Conv2D)	(None, 7, 7, 160)	179200	activation_57[0][0]
batch_normalization_53 (BatchNo	(None, 7, 7, 160)	480	conv2d_53[0][0]
batch_normalization_58 (BatchNo	(None, 7, 7, 160)	480	conv2d_58[0][0]
activation_53 (Activation)	(None, 7, 7, 160)	0	batch_normalization_53[0][0]
activation_58 (Activation)	(None, 7, 7, 160)	0	batch_normalization_58[0][0]
average_pooling2d_6 (AveragePoo	(None, 7, 7, 768)	0	mixed5[0][0]
conv2d_51 (Conv2D)	(None, 7, 7, 192)	147456	mixed5[0][0]
conv2d_54 (Conv2D)	(None, 7, 7, 192)	215040	activation_53[0][0]
conv2d_59 (Conv2D)	(None, 7, 7, 192)	215040	activation_58[0][0]
conv2d_60 (Conv2D)	(None, 7, 7, 192)	147456	average_pooling2d_6[0][0]

batch_normalization_51 (BatchNo	(None, 7, 7, 192)	576	conv2d_51[0][0]
batch_normalization_54 (BatchNo	(None, 7, 7, 192)	576	conv2d_54[0][0]
batch_normalization_59 (BatchNo	(None, 7, 7, 192)	576	conv2d_59[0][0]
batch_normalization_60 (BatchNo	(None, 7, 7, 192)	576	conv2d_60[0][0]
activation_51 (Activation)	(None, 7, 7, 192)	0	batch_normalization_51[0][0]
activation_54 (Activation)	(None, 7, 7, 192)	0	batch_normalization_54[0][0]
activation_59 (Activation)	(None, 7, 7, 192)	0	batch_normalization_59[0][0]
activation_60 (Activation)	(None, 7, 7, 192)	0	batch_normalization_60[0][0]
mixed6 (Concatenate)	(None, 7, 7, 768)	0	activation_51[0][0] activation_54[0][0] activation_59[0][0] activation_60[0][0]
conv2d_65 (Conv2D)	(None, 7, 7, 192)	147456	mixed6[0][0]
batch_normalization_65 (BatchNo	(None, 7, 7, 192)	576	conv2d_65[0][0]
activation_65 (Activation)	(None, 7, 7, 192)	0	batch_normalization_65[0][0]
conv2d_66 (Conv2D)	(None, 7, 7, 192)	258048	activation_65[0][0]
batch_normalization_66 (BatchNo	(None, 7, 7, 192)	576	conv2d_66[0][0]
activation_66 (Activation)	(None, 7, 7, 192)	0	batch_normalization_66[0][0]
conv2d_62 (Conv2D)	(None, 7, 7, 192)	147456	mixed6[0][0]
conv2d_67 (Conv2D)	(None, 7, 7, 192)	258048	activation_66[0][0]
batch_normalization_62 (BatchNo	(None, 7, 7, 192)	576	conv2d_62[0][0]
batch_normalization_67 (BatchNo	(None, 7, 7, 192)	576	conv2d_67[0][0]
activation_62 (Activation)	(None, 7, 7, 192)	0	batch_normalization_62[0][0]
activation_67 (Activation)	(None, 7, 7, 192)	0	batch_normalization_67[0][0]
conv2d_63 (Conv2D)	(None, 7, 7, 192)	258048	activation_62[0][0]
conv2d_68 (Conv2D)	(None, 7, 7, 192)	258048	activation_67[0][0]
batch_normalization_63 (BatchNo	(None, 7, 7, 192)	576	conv2d_63[0][0]
batch_normalization_68 (BatchNo	(None, 7, 7, 192)	576	conv2d_68[0][0]
activation_63 (Activation)	(None, 7, 7, 192)	0	batch_normalization_63[0][0]
activation_68 (Activation)	(None, 7, 7, 192)	0	batch_normalization_68[0][0]
average_pooling2d_7 (AveragePoo	(None, 7, 7, 768)	0	mixed6[0][0]
conv2d_61 (Conv2D)	(None, 7, 7, 192)	147456	mixed6[0][0]
conv2d_64 (Conv2D)	(None, 7, 7, 192)	258048	activation_63[0][0]
conv2d_69 (Conv2D)	(None, 7, 7, 192)	258048	activation_68[0][0]
conv2d_70 (Conv2D)	(None, 7, 7, 192)	147456	average_pooling2d_7[0][0]
batch_normalization_61 (BatchNo	(None, 7, 7, 192)	576	conv2d_61[0][0]
batch_normalization_64 (BatchNo	(None, 7, 7, 192)	576	conv2d_64[0][0]
batch_normalization_69 (BatchNo	(None, 7, 7, 192)	576	conv2d_69[0][0]
batch_normalization_70 (BatchNo	(None, 7, 7, 192)	576	conv2d_70[0][0]
activation_61 (Activation)	(None, 7, 7, 192)	0	batch_normalization_61[0][0]
activation_64 (Activation)	(None, 7, 7, 192)	0	batch_normalization_64[0][0]
activation_69 (Activation)	(None, 7, 7, 192)	0	batch_normalization_69[0][0]
activation_70 (Activation)	(None, 7, 7, 192)	0	batch_normalization_70[0][0]
mixed7 (Concatenate)	(None, 7, 7, 768)	0	activation_61[0][0] activation_64[0][0] activation_69[0][0] activation_70[0][0]
conv2d_73 (Conv2D)	(None, 7, 7, 192)	147456	mixed7[0][0]
batch_normalization_73 (BatchNo	(None, 7, 7, 192)	576	conv2d_73[0][0]
activation_73 (Activation)	(None, 7, 7, 192)	0	batch_normalization_73[0][0]
conv2d_74 (Conv2D)	(None, 7, 7, 192)	258048	activation_73[0][0]
batch_normalization_74 (BatchNo	(None, 7, 7, 192)	576	conv2d_74[0][0]
activation_74 (Activation)	(None, 7, 7, 192)	0	batch_normalization_74[0][0]
conv2d_71 (Conv2D)	(None, 7, 7, 192)	147456	mixed7[0][0]

conv2d_75 (Conv2D)	(None, 7, 7, 192)	258048	activation_74[0][0]
batch_normalization_71 (BatchNo	(None, 7, 7, 192)	576	conv2d_71[0][0]
batch_normalization_75 (BatchNo	(None, 7, 7, 192)	576	conv2d_75[0][0]
activation_71 (Activation)	(None, 7, 7, 192)	0	batch_normalization_71[0][0]
activation_75 (Activation)	(None, 7, 7, 192)	0	batch_normalization_75[0][0]
conv2d_72 (Conv2D)	(None, 3, 3, 320)	552960	activation_71[0][0]
conv2d_76 (Conv2D)	(None, 3, 3, 192)	331776	activation_75[0][0]
batch_normalization_72 (BatchNo	(None, 3, 3, 320)	960	conv2d_72[0][0]
batch_normalization_76 (BatchNo	(None, 3, 3, 192)	576	conv2d_76[0][0]
activation_72 (Activation)	(None, 3, 3, 320)	0	batch_normalization_72[0][0]
activation_76 (Activation)	(None, 3, 3, 192)	0	batch_normalization_76[0][0]
max_pooling2d_4 (MaxPooling2D)	(None, 3, 3, 768)	0	mixed7[0][0]
mixed8 (Concatenate)	(None, 3, 3, 1280)	0	activation_72[0][0] activation_76[0][0] max_pooling2d_4[0][0]
conv2d_81 (Conv2D)	(None, 3, 3, 448)	573440	mixed8[0][0]
batch_normalization_81 (BatchNo	(None, 3, 3, 448)	1344	conv2d_81[0][0]
activation_81 (Activation)	(None, 3, 3, 448)	0	batch_normalization_81[0][0]
conv2d_78 (Conv2D)	(None, 3, 3, 384)	491520	mixed8[0][0]
conv2d_82 (Conv2D)	(None, 3, 3, 384)	1548288	activation_81[0][0]
batch_normalization_78 (BatchNo	(None, 3, 3, 384)	1152	conv2d_78[0][0]
batch_normalization_82 (BatchNo	(None, 3, 3, 384)	1152	conv2d_82[0][0]
activation_78 (Activation)	(None, 3, 3, 384)	0	batch_normalization_78[0][0]
activation_82 (Activation)	(None, 3, 3, 384)	0	batch_normalization_82[0][0]
conv2d_79 (Conv2D)	(None, 3, 3, 384)	442368	activation_78[0][0]
conv2d_80 (Conv2D)	(None, 3, 3, 384)	442368	activation_78[0][0]
conv2d_83 (Conv2D)	(None, 3, 3, 384)	442368	activation_82[0][0]
conv2d_84 (Conv2D)	(None, 3, 3, 384)	442368	activation_82[0][0]
average_pooling2d_8 (AveragePoo	(None, 3, 3, 1280)	0	mixed8[0][0]
conv2d_77 (Conv2D)	(None, 3, 3, 320)	409600	mixed8[0][0]
batch_normalization_79 (BatchNo	(None, 3, 3, 384)	1152	conv2d_79[0][0]
batch_normalization_80 (BatchNo	(None, 3, 3, 384)	1152	conv2d_80[0][0]
batch_normalization_83 (BatchNo	(None, 3, 3, 384)	1152	conv2d_83[0][0]
batch_normalization_84 (BatchNo	(None, 3, 3, 384)	1152	conv2d_84[0][0]
conv2d_85 (Conv2D)	(None, 3, 3, 192)	245760	average_pooling2d_8[0][0]
batch_normalization_77 (BatchNo	(None, 3, 3, 320)	960	conv2d_77[0][0]
activation_79 (Activation)	(None, 3, 3, 384)	0	batch_normalization_79[0][0]
activation_80 (Activation)	(None, 3, 3, 384)	0	batch_normalization_80[0][0]
activation_83 (Activation)	(None, 3, 3, 384)	0	batch_normalization_83[0][0]
activation_84 (Activation)	(None, 3, 3, 384)	0	batch_normalization_84[0][0]
batch_normalization_85 (BatchNo	(None, 3, 3, 192)	576	conv2d_85[0][0]
activation_77 (Activation)	(None, 3, 3, 320)	0	batch_normalization_77[0][0]
mixed9_0 (Concatenate)	(None, 3, 3, 768)	0	activation_79[0][0] activation_80[0][0]
concatenate_1 (Concatenate)	(None, 3, 3, 768)	0	activation_83[0][0] activation_84[0][0]
activation_85 (Activation)	(None, 3, 3, 192)	0	batch_normalization_85[0][0]
mixed9 (Concatenate)	(None, 3, 3, 2048)	0	activation_77[0][0] mixed9_0[0][0] concatenate_1[0][0] activation_85[0][0]
conv2d_90 (Conv2D)	(None, 3, 3, 448)	917504	mixed9[0][0]
batch_normalization_90 (BatchNo	(None, 3, 3, 448)	1344	conv2d_90[0][0]
activation_90 (Activation)	(None, 3, 3, 448)	0	batch_normalization_90[0][0]

```
conv2d_87 (Conv2D)              (None, 3, 3, 384)    786432      mixed9[0][0]
__________________________________________________________________________________________________
conv2d_91 (Conv2D)              (None, 3, 3, 384)    1548288     activation_90[0][0]
__________________________________________________________________________________________________
batch_normalization_87 (BatchNo (None, 3, 3, 384)    1152        conv2d_87[0][0]
__________________________________________________________________________________________________
batch_normalization_91 (BatchNo (None, 3, 3, 384)    1152        conv2d_91[0][0]
__________________________________________________________________________________________________
activation_87 (Activation)      (None, 3, 3, 384)    0           batch_normalization_87[0][0]
__________________________________________________________________________________________________
activation_91 (Activation)      (None, 3, 3, 384)    0           batch_normalization_91[0][0]
__________________________________________________________________________________________________
conv2d_88 (Conv2D)              (None, 3, 3, 384)    442368      activation_87[0][0]
__________________________________________________________________________________________________
conv2d_89 (Conv2D)              (None, 3, 3, 384)    442368      activation_87[0][0]
__________________________________________________________________________________________________
conv2d_92 (Conv2D)              (None, 3, 3, 384)    442368      activation_91[0][0]
__________________________________________________________________________________________________
conv2d_93 (Conv2D)              (None, 3, 3, 384)    442368      activation_91[0][0]
__________________________________________________________________________________________________
average_pooling2d_9 (AveragePoo (None, 3, 3, 2048)   0           mixed9[0][0]
__________________________________________________________________________________________________
conv2d_86 (Conv2D)              (None, 3, 3, 320)    655360      mixed9[0][0]
__________________________________________________________________________________________________
batch_normalization_88 (BatchNo (None, 3, 3, 384)    1152        conv2d_88[0][0]
__________________________________________________________________________________________________
batch_normalization_89 (BatchNo (None, 3, 3, 384)    1152        conv2d_89[0][0]
__________________________________________________________________________________________________
batch_normalization_92 (BatchNo (None, 3, 3, 384)    1152        conv2d_92[0][0]
__________________________________________________________________________________________________
batch_normalization_93 (BatchNo (None, 3, 3, 384)    1152        conv2d_93[0][0]
__________________________________________________________________________________________________
conv2d_94 (Conv2D)              (None, 3, 3, 192)    393216      average_pooling2d_9[0][0]
__________________________________________________________________________________________________
batch_normalization_86 (BatchNo (None, 3, 3, 320)    960         conv2d_86[0][0]
__________________________________________________________________________________________________
activation_88 (Activation)      (None, 3, 3, 384)    0           batch_normalization_88[0][0]
__________________________________________________________________________________________________
activation_89 (Activation)      (None, 3, 3, 384)    0           batch_normalization_89[0][0]
__________________________________________________________________________________________________
activation_92 (Activation)      (None, 3, 3, 384)    0           batch_normalization_92[0][0]
__________________________________________________________________________________________________
activation_93 (Activation)      (None, 3, 3, 384)    0           batch_normalization_93[0][0]
__________________________________________________________________________________________________
batch_normalization_94 (BatchNo (None, 3, 3, 192)    576         conv2d_94[0][0]
__________________________________________________________________________________________________
activation_86 (Activation)      (None, 3, 3, 320)    0           batch_normalization_86[0][0]
__________________________________________________________________________________________________
mixed9_1 (Concatenate)          (None, 3, 3, 768)    0           activation_88[0][0]
                                                                 activation_89[0][0]
__________________________________________________________________________________________________
concatenate_2 (Concatenate)     (None, 3, 3, 768)    0           activation_92[0][0]
                                                                 activation_93[0][0]
__________________________________________________________________________________________________
activation_94 (Activation)      (None, 3, 3, 192)    0           batch_normalization_94[0][0]
__________________________________________________________________________________________________
mixed10 (Concatenate)           (None, 3, 3, 2048)   0           activation_86[0][0]
                                                                 mixed9_1[0][0]
                                                                 concatenate_2[0][0]
                                                                 activation_94[0][0]
==================================================================================================
Total params: 21,802,784
Trainable params: 0
Non-trainable params: 21,802,784
```

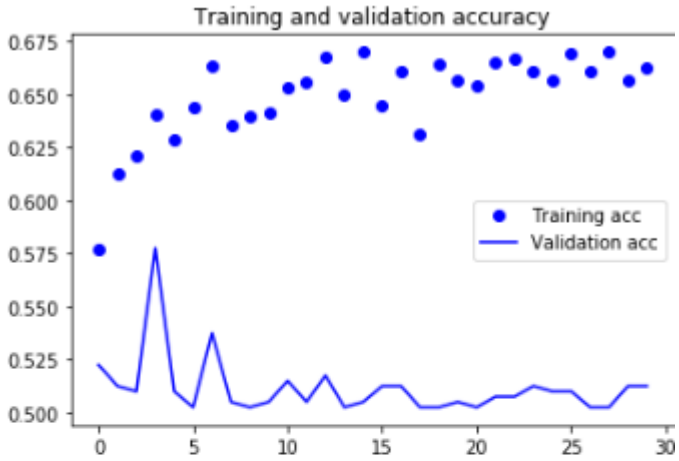
Inception V3/Fine Tuning	Train	Test
Accuray	.6713	.6525
Precision	.6663	.6142
Recall	.6862	.8200
F1-Score	.6771	.7024



InceptionResNetV2
30 Epochs
Loss: binary_crossentropy
Lr :0.0001

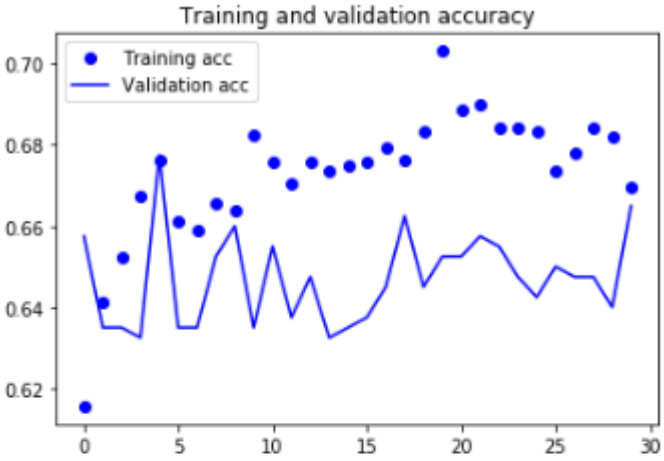
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_resnet_v2 (Model)  (None, 3, 3, 1536)        54336736
_________________________________________________________________
flatten_1 (Flatten)          (None, 13824)             0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               3539200
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 257
=================================================================
Total params: 57,876,193
Trainable params: 57,815,649
Non-trainable params: 60,544
_________________________________________________________________
```

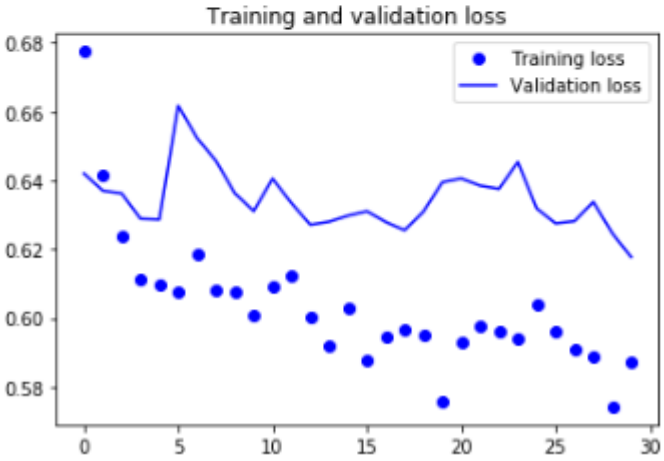
InceptionResNetV2	Train	Test
Accuray	.6625	.5125
Precision	.6693	.5063
Recall	.6425	1
F1-Score	.6556	.6723



InceptionResNetV2/Fine Tunning
30 Epochs
Loss: binary_crossentropy
Lr :0.0001
Entrenable a partir de block8_10_conv

InceptionResNetV2/Fine Tuning	Train	Test
Accuray	.6694	.6650
Precision	.6615	.6587
Recall	.6937	.6850
F1-Score	.6772	.6716


Training and validation accuracy
Training acc
Validation acc
0.70
0.68
0.66
0.64
0.62
0
5
10
15
20
25
30


Training and validation loss
Training loss
Validation loss
0.68
0.66
0.64
0.62
0.60
0.58
0
5
10
15
20
25
30