

LAB03

Implementación de modelos neuronales profundos



17 de abril de 2023

Diego Viñals Lage

Miguel Ángel Lobo Bartolomé

Pablo Medina de la Iglesia

Javier Garrido Cobo

Contenido

[Práctica 1 1](#_Toc132650866)

[Resultados de la práctica 1 2](#_Toc132650867)

[Preprocesamiento 2](#_Toc132650868)

[Definición del modelo 2](#_Toc132650869)

[Fases del predictor 3](#_Toc132650870)

[1. Explica las funciones y el algoritmo para convertir las imágenes a matrices. ¿Qué tamaños tienen las imágenes? ¿Has normalizado? ¿Cómo y por qué? 4](#_Toc132650871)

[2. Entrena una red para cada clasificación usando como criterio de parada un loss ≤ 0,2. Recoge en la memoria los valores de loss y accuracy tanto del conjunto de entrenamiento como del de validación de los distintos experimentos que hayas llevado a cabo indicando la arquitectura de cada una de las redes empleadas. Dibuja la arquitectura de red que mejor clasifica y pinta, para esa red, la gráfica de variación del error (los) en función de las epochs, para los conjuntos de entrenamiento y validación. 5](#_Toc132650872)

[2.1 Arquitectura 1 5](#_Toc132650873)

[2.2 Arquitectura 2 6](#_Toc132650874)

[2.2 Arquitectura 3 6](#_Toc132650875)

[3. ¿Qué es más fácil: distinguir entre dos tipos de neumonía o distinguir si el paciente está sano o no? ¿Por qué? 9](#_Toc132650876)

[4. ¿En caso de no tener suficientes imágenes, que técnicas usarías? Explícalas con detalle 9](#_Toc132650877)

[5. ¿Se te ocurre alguna manera de modificar las imágenes para que el modelo mejore (en velocidad de entrenamiento o precisión)? 9](#_Toc132650878)

[Práctica 2 10](#_Toc132650879)

[Motivación del problema seleccionado y justificación de la solución que se quiere obtener. 10](#_Toc132650880)

[Resultados obtenidos 11](#_Toc132650881)

[Preprocesamiento 11](#_Toc132650882)

[Selección de arquitectura de red y entrenamiento 12](#_Toc132650883)

[13](#_Toc132650884)

[Clasificación/predicción 14](#_Toc132650885)

[16](#_Toc132650886)

[Bibliografía 17](#_Toc132650887)

# Práctica 1

## Resultados de la práctica 1

### Preprocesamiento

Obtenemos los datos y buscamos las distintas clases para el set de entrenamiento, validación y test. Obtenemos 5216 imágenes de la carpeta de entrenamiento, 16 imágenes en la carpeta validación y 624 en la carpeta de test. Al filtrar por directorio, obtenemos dos clases para cada carpeta (normal y neumonía).

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Aplicamos una función de data augmentation para aplicar transformaciones a las imágenes y posteriormente mostramos un ejemplo de cada clase perteneciente al set de entrenamiento.

Imagen que contiene corbata, ropa, foto, vistiendo

Descripción generada automáticamente

### Definición del modelo

Se realiza un modelo que contiene 4 capas convolucionales y 2 capas densas La entrada contiene las imágenes con un tamaño de 150x150 píxeles y 3 canales de color. Las capas contienen funciones de activación ReLu y la salida contiene dos neuronas con una función sigmoide.

Se implementan las funciones de activación ReLU en todas las capas, menos en la de salida que se debe de usar softmax. Para actualizar los pesos, usaremos el optimizer ‘Adam’ y la función de error ‘categorical\_crossentropy’

### Fases del predictor

Una vez definido el modelo, necesitamos entrenarlo con los datos obtenidos en el preprocesamiento. El modelo con datos de entrenamiento se compara con los datos de validación para comprobar la tasa de acierto ante imágenes que no ha visto antes.

Una vez el modelo ha sido entrenado y validado, podemos usarlo para realizar predicciones. Esto implica alimentar al modelo con nuevos datos y dejar que haga una predicción basada en los patrones que ha aprendido durante el entrenamiento. La precisión de las predicciones dependerá de la calidad de los datos de entrada y de la precisión del modelo.

Imagen que contiene radiografía, hombre

Descripción generada automáticamenteImagen en blanco y negro de una persona

Descripción generada automáticamente con confianza bajaMostramos algunas imágenes que el modelo ha clasificado sobre el set de entrenamiento.

Imagen en blanco y negro

Descripción generada automáticamente con confianza bajaImagen en blanco y negro

Descripción generada automáticamente con confianza baja

En los distintos ejemplos podemos ver en “True label” la verdadera pertenencia de la imagen a su clase, posteriormente se muestra en orden descendente el porcentaje de pertenencia a una clase que ha predicho nuestro modelo. El modelo es capaz de predecir correctamente las imágenes.

Realizamos una comprobación para ver la precisión y pérdida del modelo con el set de test. Obtenemos una precisión de 0.89 y un los de 0.33.



## 1. Explica las funciones y el algoritmo para convertir las imágenes a matrices. ¿Qué tamaños tienen las imágenes? ¿Has normalizado? ¿Cómo y por qué?

Hemos redimensionado las imágenes a un tamaño de (150, 150) píxeles y utilizaremos 3 canales de color (R,G,B) para la clasificación. Al asignar un mismo tamaño a todas las imágenes estas podrán ser procesadas por la red convolucional.

Además, las imágenes se normalizan dividiendo cada valor de píxel por 255. Esto se hace para que todos los valores de píxeles estén en el rango de 0 a 1, lo que facilita el proceso de entrenamiento de la red neuronal.

Hemos utilizado la biblioteca Keras y su clase ImageDataGenerator para realizar el preprocesamiento de imágenes. Esta clase carga imágenes y las convierte en lotes de matrices de píxeles que usaremos posteriormente para entrenar nuestro modelo.

Para la clase ImageDataGenerator hemos aplicado varias transformaciones a las imágenes antes de convertirlas en matrices, como escalamiento, desplazamiento, volteo horizontal y zoom.

Adicionalmente hemos usado el método flow\_from\_directory, este método nos permite cargar las imágenes del directorio de datos y convertirlas en lotes de matrices de píxeles. El método asigna automáticamente una etiqueta a cada imagen según el nombre de su directorio, lo que usaremos posteriormente a la hora de realizar las predicciones para saber la etiqueta verdadera de cada imagen.

## 2. Entrena una red para cada clasificación usando como criterio de parada un loss ≤ 0,2. Recoge en la memoria los valores de loss y accuracy tanto del conjunto de entrenamiento como del de validación de los distintos experimentos que hayas llevado a cabo indicando la arquitectura de cada una de las redes empleadas. Dibuja la arquitectura de red que mejor clasifica y pinta, para esa red, la gráfica de variación del error (los) en función de las epochs, para los conjuntos de entrenamiento y validación.

### 2.1 Arquitectura 1

Se realiza un modelo que contiene 3 capas convolucionales y 1 capas densas. La entrada contiene las imágenes con un tamaño de 150x150 píxeles y 3 canales de color. Las capas contienen funciones de activación ReLu y la salida contiene dos neuronas con una función sigmoide. Como resultados obtenemos:

|  |  |  |  |
| --- | --- | --- | --- |
| Entrenamiento | | Validación | |
| Loss: 0.19 | Accuaracy: 0.87 | Loss: 0.87 | Accuaracy: 0.62 |

### 2.2 Arquitectura 2

Se realiza un modelo que contiene 3 capas convolucionales y 2 capas densas La entrada contiene las imágenes con un tamaño de 150x150 píxeles y 3 canales de color. Las capas contienen funciones de activación ReLu y la salida contiene dos neuronas con una función sigmoide. Como resultados obtenemos:

|  |  |  |  |
| --- | --- | --- | --- |
| Entrenamiento | | Validación | |
| Loss: 0.13 | Accuaracy: 0.91 | Loss: 0.74 | Accuaracy: 0.57 |

### 2.2 Arquitectura 3

Se realiza un modelo que contiene 4 capas convolucionales y 2 capas densas La entrada contiene las imágenes con un tamaño de 150x150 píxeles y 3 canales de color. Las capas contienen funciones de activación ReLu y la salida contiene dos neuronas con una función sigmoide. Como resultados obtenemos:

|  |  |  |  |
| --- | --- | --- | --- |
| Entrenamiento | | Validación | |
| Loss: 0.14 | Accuaracy: 0.95 | Loss: 0.67 | Accuaracy: 0.69 |

La arquitectura elegida ha sido la arquitectura 3 el modelo alcanzó una precisión del 95% en el conjunto de entrenamiento y una precisión del 69% en el conjunto de validación. El valor de loss en el conjunto de entrenamiento fue de 0.1400 y en el conjunto de validación fue de 0.67.

Es probable que el modelo esté sobreajustando los datos de entrenamiento debido a la diferencia significativa que existe entre la pérdida del modelo de entrenamiento y el modelo de validación.

Esto significa que el modelo puede estar aprendiendo detalles específicos de los datos de entrenamiento que no se generalizan bien a los datos de validación. A continuación, se presenta la arquitectura dibujada.

Interfaz de usuario gráfica

Descripción generada automáticamente

Se presentan las gráficas de variación del error en función de las épocas para el entrenamiento y la validación.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Gráfico, Gráfico de líneas

Descripción generada automáticamente

## 3. ¿Qué es más fácil: distinguir entre dos tipos de neumonía o distinguir si el paciente está sano o no? ¿Por qué?

Es más fácil distinguir entre si el paciente está sano o no. A la hora de clasificar si el paciente está sano o no, tendríamos que clasificar entre 2 clases; neumonía y normal/sano. En el caso de tener que distinguir entre tipos de neumonía, tendríamos que clasificar entre 3 clases, lo que aumentaría la complejidad.

## 4. ¿En caso de no tener suficientes imágenes, que técnicas usarías? Explícalas con detalle

Data augmentation: esta técnica se utiliza para crear nuevas imágenes a partir de las imágenes existentes en el conjunto de datos. Se aplican diferentes transformaciones a las imágenes, para la práctica hemos usado las siguientes transformaciones escalamiento, desplazamiento, volteo horizontal y zoom.

Generación de datos sintéticos: se pueden generar imágenes sintéticas utilizando software de simulación. Estas imágenes sintéticas pueden ayudar a mejorar el rendimiento del modelo al proporcionar más variedad en el conjunto de datos.

## 5. ¿Se te ocurre alguna manera de modificar las imágenes para que el modelo mejore (en velocidad de entrenamiento o precisión)?

Para la práctica hemos aplicado las siguiente modificaciones de cara a la velocidad de entrenamiento.

* Normalización: la normalización de las imágenes puede ayudar a mejorar la velocidad de entrenamiento al reducir la variabilidad de los datos.
* Data augmentation: esta técnica implica la generación de nuevas imágenes a partir de las existentes mediante la aplicación de transformaciones. Esto puede ayudar a aumentar el tamaño del conjunto de datos, mejorando la generalización del modelo y la precisión.

# Práctica 2

## Motivación del problema seleccionado y justificación de la solución que se quiere obtener.

Hemos obtenido un data set en kaggle sobre imágenes de distintos estados en plantas. El objetivo de realizar una clasificación para este dataset es para la identificación temprana de enfermedades. Las imágenes contienen 3 categorías, plantas en buen estado, plantas con polvo y plantas con erosiones.

Distinguir entre estas tres categorías de plantas puede ayudar a los agricultores a identificar tempranamente las enfermedades de las plantas en su cultivo. Esto puede permitir un tratamiento más oportuno y efectivo de las enfermedades, lo que a su vez puede aumentar el rendimiento de los cultivos.

Por ejemplo, si se identifican plantas con enfermedad en polvo, se pueden aplicar fungicidas específicos para controlar la propagación de la enfermedad.

La solución que se desea obtener es un clasificador que permita para una imagen de una planta, un porcentaje de pertenencia a cada una de las clases mencionadas anteriormente (sano, polvo, erosión).

Al dar un porcentaje de pertenencia, plantas sanas que posean porcentajes altos de otras clases podrían determinar que las plantas están empezando a ponerse enfermas. Esto ayudaría a los agricultores a actuar a tiempo antes de que la enfermedad se expanda.

Se implementará una CNN para resolver el problema.

## Resultados obtenidos

### Preprocesamiento

Obtenemos los datos y buscamos las distintas clases para el set de entrenamiento, validación y test. Obtenemos 1322 imágenes de la carpeta de entrenamiento, 150 imágenes en la carpeta validación y 60 en la carpeta de test. Al filtrar por directorio, obtenemos dos clases para cada carpeta (Healthy/sano, Powdery/polvo y Rust/erosión).

Texto

Descripción generada automáticamente

Una flor blanca

Descripción generada automáticamente con confianza mediaAplicamos una función de data augmentation para aplicar transformaciones a las imágenes y posteriormente mostramos un ejemplo de cada clase perteneciente al set de entrenamiento.

### Selección de arquitectura de red y entrenamiento

Se realiza un modelo que contiene 4 capas convolucionales y 1 capas densas La entrada contiene las imágenes con un tamaño de 360x360 píxeles y 3 canales de color. Las capas contienen funciones de activación ReLu y la salida contiene tres neuronas con una función sigmoide.

Interfaz de usuario gráfica, Diagrama

Descripción generada automáticamenteSe presenta la arquitectura dibujada:

Obtenemos los siguientes resultados para los valores de entrenamiento y validación:

|  |  |  |  |
| --- | --- | --- | --- |
| Entrenamiento | | Validación | |
| Loss: 0.07 | Accuaracy: 0.98 | Loss: 0.22 | Accuaracy: 0.95 |

Gráfico, Gráfico de líneas

Descripción generada automáticamenteObtenemos las siguientes gráficas:

# Gráfico, Gráfico de líneas Descripción generada automáticamente

### Clasificación/predicción

Una vez definido el modelo, necesitamos entrenarlo con los datos obtenidos en el preprocesamiento. El modelo con datos de entrenamiento se compara con los datos de validación para comprobar la tasa de acierto ante imágenes que no ha visto antes.

Una vez el modelo ha sido entrenado y validado, podemos usarlo para realizar predicciones. Esto implica alimentar al modelo con nuevos datos y dejar que haga una predicción basada en los patrones que ha aprendido durante el entrenamiento. La precisión de las predicciones dependerá de la calidad de los datos de entrada y de la precisión del modelo.

Un conjunto de árboles

Descripción generada automáticamente con confianza bajaMostramos algunas imágenes que el modelo ha clasificado sobre el set de entrenamiento.

Un conjunto de árboles

Descripción generada automáticamente con confianza baja

Una flor amarilla

Descripción generada automáticamente con confianza media

En los distintos ejemplos podemos ver en “True label” la verdadera pertenencia de la imagen a su clase, posteriormente se muestra en orden descendente el porcentaje de pertenencia a una clase que ha predicho nuestro modelo. El modelo es capaz de predecir correctamente las imágenes.

Realizamos una comprobación para ver la precisión y pérdida del modelo con el set de test. Obtenemos una precisión de 0.94 y un los de 0.28.

# 

# Bibliografía

Tejedor, Á. J. (s.f.). *IAII\_Canvas*. Obtenido de https://ufv-es.instructure.com/courses/26539/pages/sobre-la-asignatura

https://www.kaggle.com/datasets/rashikrahmanpritom/plant-disease-recognition-dataset

https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia?datasetId=17810&searchQuery=cnn

https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/

https://guru99.es/convnet-tensorflow-image-classification/