



**Momento de Retroalimentación: Módulo 2 - Módulo 2 Implementación de un
Modelo de Deep Learning**

Raúl Youthan Irigoyen Osorio A01750476

02 de noviembre del 2022

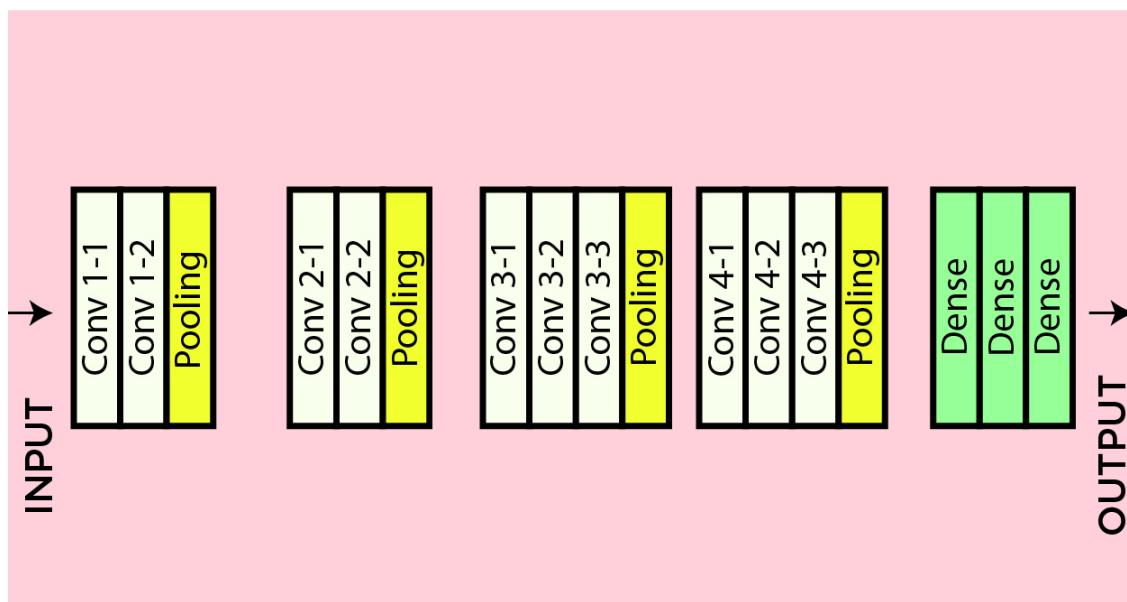
Inteligencia artificial avanzada para la ciencia de datos II

Problema a resolver

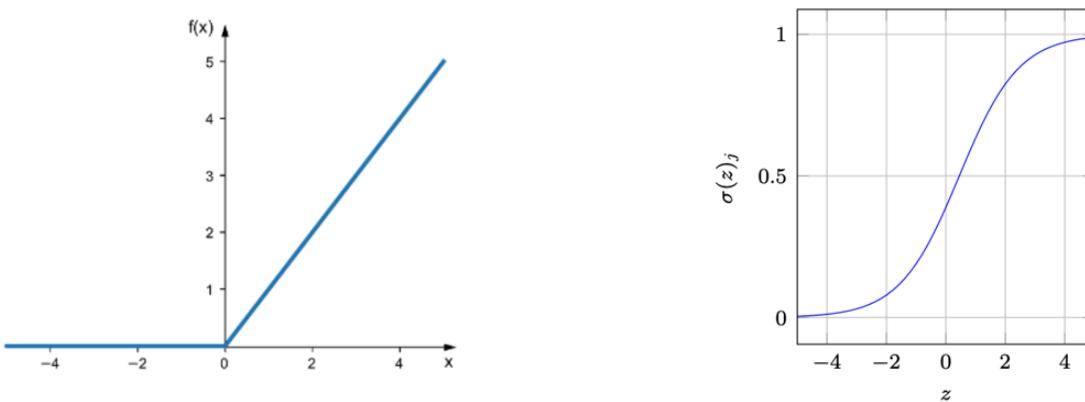
Este documento tiene el propósito de analizar la implementación de una red neuronal convolucional cuyo propósito es la clasificación de imágenes de diferentes personas con disfraces de Halloween, los cuales serán identificados y catalogados como **bruja**, **payaso**, **momia** o **Frankenstein** de acuerdo con sus atributos y características propias. Este modelo fue creado haciendo uso de **Tensorflow**.

Arquitectura del modelo

La arquitectura del modelo planteada inicialmente es la siguiente:



Cada capa de convolución utiliza la función de activación conocida como **ReLU (Rectified Linear Unit)**, así mismo, cada capa densa utiliza la misma función de activación, excepto por la **capa de salida**, que implementa el algoritmo conocido como **softmax**. A continuación se muestra la interpretación gráfica de ambas funciones:



Finalmente, cada grupo de capas de convolución está acompañada por una capa de **max-pooling**, que determina aquellas características de mayor prominencia en la imagen a través de una serie de operaciones matemáticas.

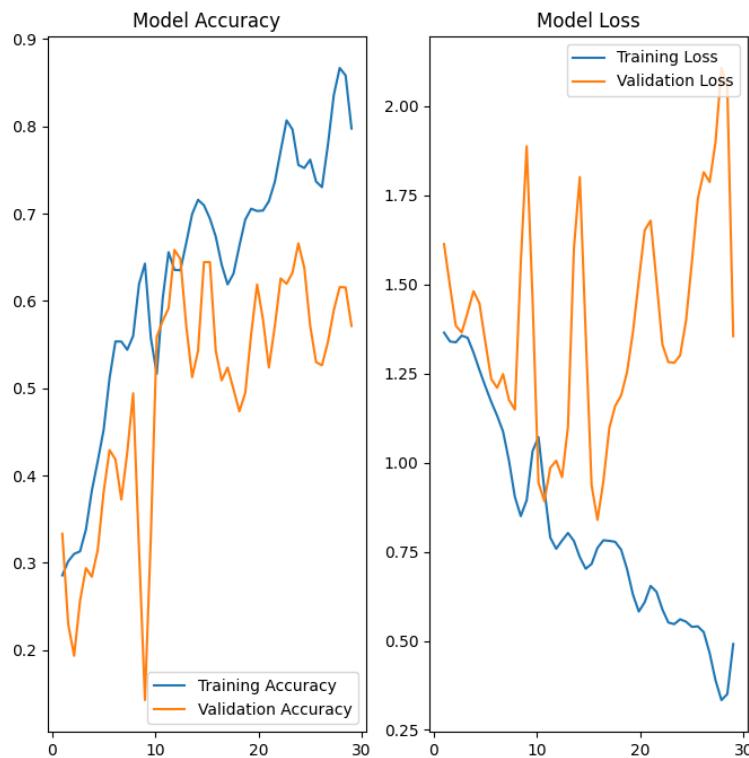
Análisis de resultados

Al realizar la primer prueba, el tamaño de cada imagen era de aproximadamente 96x96 píxeles, por otro lado, los parámetros de cada capa fueron los siguientes:

- **Conv 1:**
 - 32 filtros por capa.
 - Kernel de tamaño 3 x 3.
 - Sin relleno,
 - Función de activación **ReLU**.
- **Tras cada grupo de capas convolucionales, los ajustes fueron los mismos excepto por los filtros, que se duplicaron hasta llegar a 256 en el grupo Conv 4.**
- **Dense 1:**
 - 128 unidades.
 - Función de activación **ReLU**.
- **Dense 2:**
 - 64 unidades.
 - Función de activación **ReLU**.
- **Dense 3:**
 - 4 unidades.
 - Función de activación **softmax**.

Los resultados del entrenamiento se interpretaron utilizando la historia generada automáticamente por el modelo una vez que se implementó el método **fit()** de la librería **keras**. Los resultados están enfocados en los valores generales de precisión y pérdida de entrenamiento,

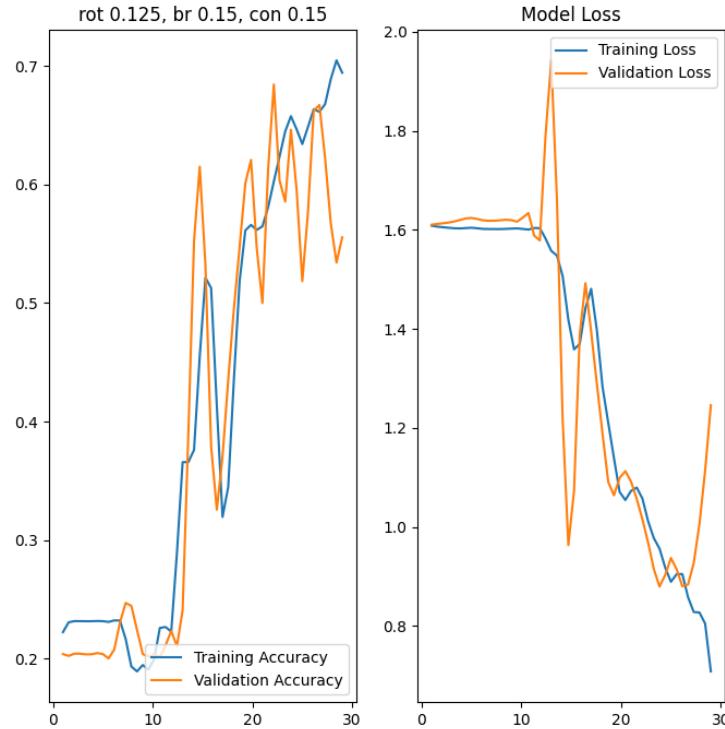
así como los mismos valores en el caso de la validación de resultados, como se puede observar en la siguiente gráfica:



Dado que se esperaba una mejor precisión, se hicieron una serie de cambios que beneficiaron tanto a la prevención de **overfitting** como a los valores evaluados en la gráfica.

En primer lugar, el tamaño del dataset se incrementó al doble, sin embargo la porción de datos correspondientes a la **validación de resultados** se mantuvo en **20%**. Después, se añadió una capa de regularización de tamaño de imágenes, que en **keras** tiene el nombre de **rescaling**, esto se hizo con el fin de mantener los datos en una escala homogénea, permitiendo identificar más rápidamente el mínimo local durante el entrenamiento del modelo. Así mismo, se añadió una capa de **data augmentation**, que alteran la **rotación** de la imagen de forma aleatoria en una proporción no mayor al 20%, lo mismo para el **brillo**, el **contraste** y el **zoom** de las imágenes.

Los resultados de dichos cambios fueron los siguientes:



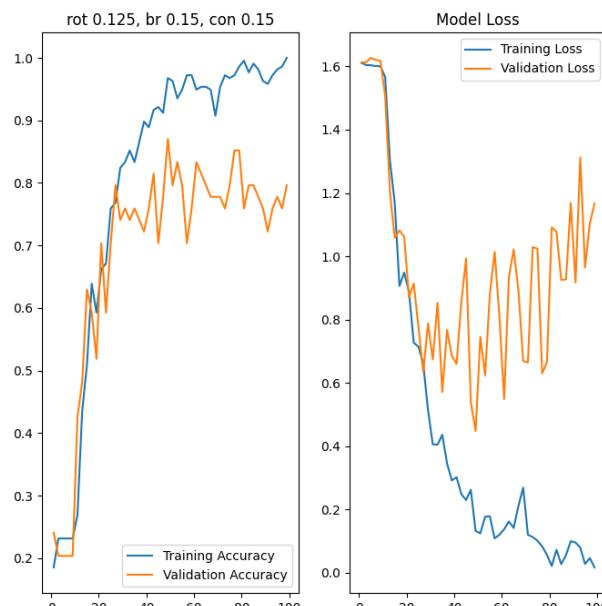
Es visible que en ambos valores tanto el set de **validación** como el de **entrenamiento** siguen un patrón más **similar** que en el entrenamiento anterior, sin embargo, la **precisión** se mantuvo relativamente **igual**, por lo que una última serie de ajustes se realizó para resolver dicho problema, donde la **estructura del modelo** cambió desde la capa de entrada, donde cada imagen por estándar ahora tiene un tamaño de 224 x 244 píxeles y las capas de la red neuronal tienen los siguientes nuevos parámetros:

- **Conv 1:**

- 8 filtros por capa.
- Kernel de tamaño 3 x 3.

- Sin relleno,
- Función de activación **ReLU**.
- Tras cada grupo de capas convolucionales, los ajustes fueron los mismos excepto por los filtros, que se duplicaron hasta llegar a 64 en el grupo Conv 4.
- Dense 1:
 - 128 unidades.
 - Función de activación **ReLU**.
- Dense 2:
 - 64 unidades.
 - Función de activación **ReLU**.
- Dense 3:
 - 32 unidades.
 - Función de activación **ReLU**.
- Dense 4:
 - 4 unidades.
 - Función de activación **softmax**.

Así mismo, se **incrementó** el número de **épocas de entrenamiento**, dado que un mínimo local no se alcanzó en los entrenamientos anteriores, se **redujeron** también los rangos de valores aleatorios que toman las alteraciones en brillo, contraste y rotación de las imágenes a un **máximo de 10%**, lo que arrojó los siguientes resultados, **aumentando** alrededor de **15%** la precisión:



Predicciones realizadas

1. Imagen utilizada:



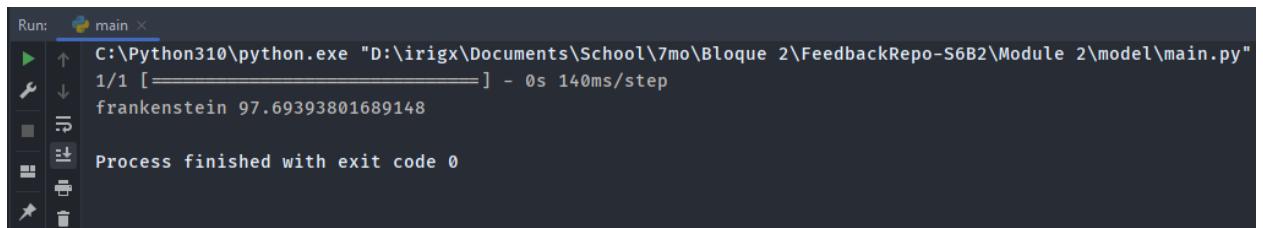
Resultado:

```
Run: main ×
C:\Python310\python.exe "D:\irigx\Documents\School\7mo\Block 2\FeedbackRepo-S6B2\Module 2\model\main.py"
1/1 [=====] - 0s 152ms/step
clown 99.86827969551086
Process finished with exit code 0
```

2. Imagen utilizada:



Resultado:

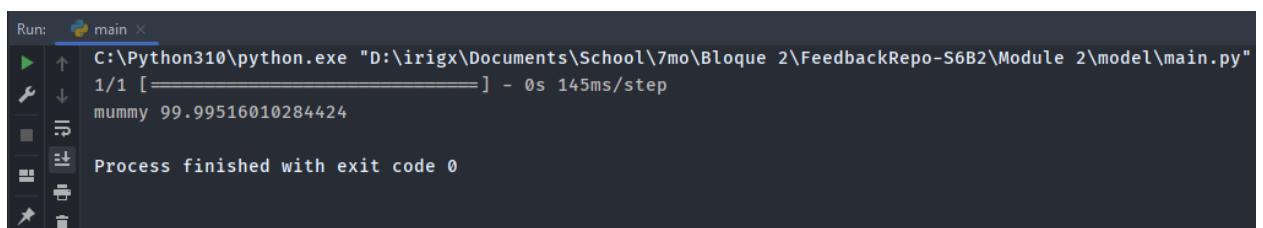


```
Run: main ×
C:\Python310\python.exe "D:\irigx\Documents\School\7mo\Block 2\FeedbackRepo-S6B2\Module 2\model\main.py"
1/1 [=====] - 0s 140ms/step
frankenstein 97.69393801689148
Process finished with exit code 0
```

3. Imagen utilizada:



Resultado:

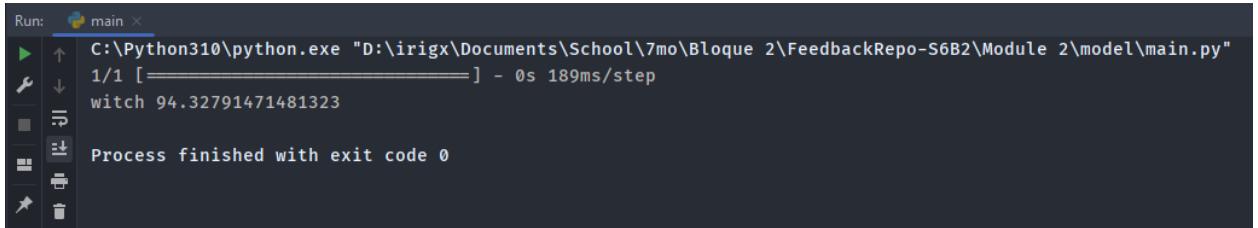


```
Run: main ×
C:\Python310\python.exe "D:\irigx\Documents\School\7mo\Block 2\FeedbackRepo-S6B2\Module 2\model\main.py"
1/1 [=====] - 0s 145ms/step
mummy 99.99516010284424
Process finished with exit code 0
```

4. Imagen utilizada:



Resultado:



The screenshot shows a terminal window titled "Run: main". The command executed was "C:\Python310\python.exe "D:\irigx\Documents\School\7mo\Block 2\FeedbackRepo-S6B2\Module 2\model\main.py"". The output indicates a single test case passed ("1/1 [=====] - 0s 189ms/step") with a result of "witch 94.32791471481323". The message "Process finished with exit code 0" is displayed at the bottom.