

2022

**Red neuronal clasificadora de sillas**

*Recuperativa Inteligencia artificial*

FELIPE URTUBIA MAYORGA

***Contenido***

[**Introducción** 2](#_Toc96809667)

[**Diseño** 3](#_Toc96809668)

[Selección de variables dentro de la red neuronal 4](#_Toc96809669)

[**Manual de testeo del modelo** 5](#_Toc96809670)

[**Conclusión** 6](#_Toc96809671)

# **Introducción**

Dentro del siguiente informe vamos a pasar a revisar el proceso de la creación, implementación, optimización y la conclusión del problema planteado para el proyecto recuperativo de la asignatura de inteligencia artificial.

El problema que nos presenta este proyecto trata de desarrollar una red neuronal que pueda diferenciar entre distintos tipos de sillas, utilizando la herramienta Tensorflow y Keras.

Este informe se va a centrar en el diseño de dicha red, los resultados obtenidos, un manual de instrucciones que capacite al lector a ejecutar la red tanto en su maquina local como en la nube con la ayuda de la plataforma Kaggle.

Desde los primeros borradores hasta la versión final fueron programados de forma local en mi máquina. Ocupando la siguiente lista de programas y lenguajes.

* Anaconda v1.9.0
* Conda v4.10.3
* Jupyter notebook v7.29.0
* Python v3.9.7
* Visual Studio Code v1.64.2
* Numpy v1.20.3
* TensorFlow v2.8.0
* TensorBoard v2.8.0
* Keras v2.8.0

Entre otros paquetes instalados de forma automática por Anaconda.

Tener en cuanta que la versión de entrega, de la que se va a habar en este trabajo, es la que se encuentra en la plataforma Kaggle, bajo el nombre Rn clasificadora de sillas. A la cual se puede visitar mediante el siguiente enlace <https://www.kaggle.com/felipeurtubia/rn-clasificadora-de-sillas>.

Los resultados están basados en los datos encontrados en las carpetas logs y model de la versión 1 del libro de proyecto antes presentado.

# **Diseño**

Antes de la red neuronal, tenemos que definir el alcance de nuestro proyecto. Recordemos que nuestro tema escogido es “red clasificatoria de sillas” y necesitamos enfocar esa clasificación a un numero de categorías, para el caso de este proyecto, vamos a sesgar los datos dentro de 5 clases, sillas de terraza, comedor, taburetes, mecedoras y gamers.

Con las categorías en mano, tenemos que definir el numero de datos que la red neuronal va a analizar, en mi caso, escogí que cada categoría va a contar con 500 imágenes de entrenamiento y 100 imágenes de testeo, dándonos un total de 2500 imágenes de “training” y 500 imágenes de “testing”, este conjunto será referido como dataset desde ahora en adelante.

Con el dataset y las categorías ya definidas, pasemos a la propia red. El código esta dividido en 4 bloques de código, el trabajo del primer bloque es importar las distintas librerías que se va a utilizar dentro de todo el proyecto, destacando tensorflow, keras y tensorboard, definir la configuración inicial de la red, como la dirección de entrada de los datos o las distintas variables que se van a reutilizar varias veces dentro del código. Una ves los datos ya están listos y cargados de forma correcta, se pasa al siguiente bloque, que es el encargado de definir nuestra red, en este caso, la red cuenta con 3 capas convolucionales, con 64 nodos cada capa, crenado en conjunto nuestra capa escondida, y tenemos la ultima capa de salida, una capa densa que como atributo tiene la activación “softmax”, los detalles de cómo se llegó a la conclusión de que la red funcionaria mejor con un formato “3-conv-64-nodes-0-desnse” se va a explicar más adelante.

Una vez el modelo esta compilado a nuestras especificaciones, pasamos al tercer bloque, esté se encuentra encargado de realizar el entrenamiento o “fit” de nuestra red, entre los parámetros ingresados, quiero destacar el callback a tensorboard, herramienta que nos ayuda a entender de manera visual los datos obtenidos en el entrenamiento.

Por último, el bloque final es el encargado de guardar el modelo generado en el proceso de entrenamiento, bajo un nombre ya definido con la ventaja de que se agrega el momento en el que crea, para no sobre escribir algún modelo anterior por error.

## Selección de variables dentro de la red neuronal

Antes de pasar al manual para ejecutar la red, quiero destacar las decisiones que se tomaron para llegar a la forma final de la red.

Gráfico, Gráfico de dispersión

Descripción generada automáticamenteOcupando la herramienta tensorflow, se creo de manera iterativa una cantidad bastante grande de distintos tipos de modelos, en donde se variaba la cantidad de capas convolucionales, nodos en cada capa y cantidad de capas densas antes de la capa de salida. Esto se llevo a cabo con los parámetros de las capas con las opciones entre 1, 2 o 3 capas convolucionales, 32, 64, 128 o 256 nodos y la opción entre 0, 1 y 2 capas densas, esto llevo al siguiente grafico generado con tensorboard.

Gráfico, Gráfico de líneas

Descripción generada automáticamenteDentro de todas las posibles combinaciones destacamos “3-conv-256-nodes-2-dense”, “3-conv-32-nodes-1-dense”, “3-conv-32-nodes-2-dense” y “3-conv-64-nodes-0-dense”.

Con esta información, se tomó la decisión de ocupar “3-conv-64-nodes-0-dense”, ya que obtuvo el mejor puntaje en 3 de las 4 categorías de medición, y viendo el desempeño que logró en la categoría de “evaluation loss vs iterations”, podemos concluir que la red no está cometiendo “overtraining”, que es el punto donde la red deja de aprender y comienza a memorizar. Con esto en mente, este es el resultado de el entrenamiento que realizo la red que está en kaggle.

Gráfico, Gráfico de líneas

Descripción generada automáticamenteEn este ciclo de entrenamiento la red alcanzó un 95% de precisión de datos.

# **Manual de testeo del modelo**

Para realizar el testeo del modelo creado con la v1 del libro de proyecto de kaggle, es necesario un par de cosas, primero, la carpeta con el modelo, las imágenes de prueba y un archivo con extensión ipynb. para leer ese archivo es necesario el editor jupyter notebook, para obtenerlo, te recomiendo descargar un programa llamado anaconda3. Con este programa tienes que instalar jupyter notebook y anaconda prompt, este último te ayudará a navegar entre tus carpetas e instalar un par de librerías que vamos a necesitar más adelante.

Una vez instalado tanto anaconda prompt como jupyter notebook, inicia jupyter notebook y se te abrirá automáticamente una pestaña en tu navegador predeterminado, si no es así, copia la línea de comando que parta con <http://localhost:8888/?token=>, prueba la primera, si no intenta la segunda. Una vez adentro, ocupa tu pestaña del navegador hasta llegar a la carpeta descargada con los distintos archivos ya mencionados, te darás cuenta de que “prueba\_del\_modelo” tiene un icono como si fuera un cuaderno, haz clic sobre él y se abrirá otra pestaña con lo que parecen 2 bloques con un poco de código dentro de cada uno. Ahora abre anaconda prompt y se te abrirá una terminal negra, usualmente parte dentro de la carpeta “C:\Users\tu\_usuario”, desde ahí tienes que navegar hasta el directorio de la carpeta, recuerda que dentro de la línea de comando te tienes que mover con el comando “cd”, una vez dentro de la carpeta donde está el archivo “prueba\_del\_modelo”, (puedes ver los archivos existentes dentro de la carpeta que estas con el comando “dir”), tienes que correr el comando “pip install opencv-python” y se deberían empezar a llenar unas barras blancas que estén más abajo, eso significa que se están descargando los paquetes, de la misma manera, tienes que instalar numpy y tensorflow, “pip install numpy” y “pip install tensorflow”.

Una vez ya este todo descargado e instalado sin ningún mensaje de error, vuelve a la pestaña de jupyter con la prueba del modelo, ahora tienes 2 opciones, puedes correr los bloques por separado con el botón “run” que aparece en la barra de arriba, o puedes presionar el botón con 2 flechas a la derecha, que hará correr los 2 bloques automáticamente. La separación de bloques tiene el propósito de que puedas agregar imágenes a la carpeta “imágenes\_de\_prueba” sin que tenga que cargar el modelo cada vez, que es un poco más de tiempo que solo cargar las imágenes. Este pequeño programa esta cargado con 25 imágenes de prueba catalogadas, si vas a agregar más imágenes, te recomiendo catalogarlas, ya que la forma de cargarlas es en formato alfabético y se confunde puede confundir que resultado pertenece a que imagen, de todas formas, imprime todas las fotos que pudo encontrar con su respectivo nombre.

# **Conclusión**

El tema de la inteligencia artificial tiene una dificultad muy interesante, por que por si solo, no es tan difícil, si no que complejo, tienes que haber entendido muchas cosas de la teoría antes de entrar al código, cada parámetro dentro del código tiene su porque, y muchas personas hacen lo mismo de muchas maneras distintas, cada día se crean librerías nuevas, o se actualizan las antiguas, una forma de hacer las cosas hoy, puede que no sea la misma mañana, ese adapta miento de las circunstancias crea una barrera de entrada un tanto difícil a este tema, pero una vez dentro, sabiendo que pasos tienes que seguir para el resultado que quieres, se vuelve tremendamente mundano, hasta el punto que terminas creando decenas de entornos de prueba cuando al principio no podías entender un ejemplo en código.

Además, existen muchas herramientas que te ayudan a llevar estos conceptos que pereciesen imaginarios a algo más tangible, más visual, de manera que puedas leer los numero que aparecen en la pantalla con un sentido.

Si pensamos en los resultados base, se crearon cientos de redes neuronales muy diferentes unas de las otras, al final se escogió a la mejor de su grupo, y si vamos por los números secos, esa red neuronal consiguió 95,08% de precisión, una perdida entre generación de 14,59%, una evaluación de precisión contra iteraciones de 0.888 puntos y una evaluación de perdida contra iteraciones de 0.4662 puntos, pero esos números sin su contexto no nos dicen nada. Después de muchas pruebas con distintos parámetros tenía 4 candidatos con puntajes tan cercanos que llamar a uno ganador no tenía sentido, pero cada uno tenia su propio recorrido y una forma muy distinta de llegar a su puntaje final, al final, fue el contexto el que le dio la victoria a la red neuronal que se acogió, su baja cantidad de nodos necesario y menos capas densas significaban menos tiempo de procesamiento de datos para el dataset que teníamos, pero lo interesante es que eso puede cambiar con cualquier cosa, un aumento en los datos, un cambio en el código o simplemente hacer otra compilación de varias redes, puede que esta vez, se consigan otros resultados que justifique elegir otro camino, al final, estos sistemas son lo más dinámico e impredecible que se ah creado en el área de informática, podemos guiarlos hacia cierto resultado, pero no sabemos al 100% que tal van a llegar.

Nota: al terminar el informe volví a dar un repaso a todo lo que tenia que entregar, y me di cuenta de que había un error al momento de poner el nombre al modelo en el código que había subido a kaggle, así que aproveche de hacer una v2 que además de arreglar ese error, hice que mostrara una visual del modelo aparte de describirlo, aparte de eso, no se movió nada más del código y se ocupó el modelo creado en la v1.

Lamento que el informe haya sido un poco más largo de lo que se pedía, pero como no va a ver presentación del código, quería explicar un poco más en profundidad mis decisiones, además, me salte los resultados por que el informe ya era largo y planeaba dejarlos en la conclución.