Auto-Keras y Keras Tuner

C. G. Rodríguez, *Estudiante MCIA, UAQ.*
Deep Learning

***Resumen*— Las librerías de Keras se han vuelto una herramienta popular y prometedora en el campo de la inteligencia artificial, con una rápida expansión de conocimiento y constantes desarrollos.**

En el presente trabajo se probaron las librerías de Autokeras y Keras tuner aplicadas la base de datos propuesta de perros y gatos.

***Temas claves*—Auto-Keras, Keras Tuner, Machine Learning, Clasificación de imágenes, Keras, TensorFlow.**

I. Introducción

AL construir un modelo de aprendizaje profundo, es común el cuestionarse sobre la arquitectura que tendrá el modelo en cuestión, en donde se deben de tomar en consideración varios hiperparámetros.

Normalmente se suele realizar la construcción del modelo en forma de prueba y error, es decir, conforme a los resultados obtenidos en la etapa de entrenamiento se decide que hiperparámetros serán necesarios ajustar para obtener un mejor resultado. Este proceso termina siendo un poco tedioso, ya que es muy probable que se inviertan muchas horas ajustando manualmente los hiperparámetros del modelo con el fin de alcanzar el rendimiento esperado.

Por tal motivo, el uso de la librería de Auto-Keras tiene como objetivo el poder identificar la arquitectura con el mejor rendimiento para la base de datos propuesta. De esa forma es posible reducir el tiempo de búsqueda y selección entre los diferentes modelos.

De igual manera, la biblioteca de Keras Tuner es un marco de optimización de hiperparámetros fácil de usar que resuelve los puntos débiles de realizar una búsqueda de hiperparámetros, es decir, por medio de ella es posible encontrar los mejores valores de hiperparámetros para la base de datos en cuestión.

En el presente trabajo se implementará la librería de Auto-Keras, la cual realizará una búsqueda entre los modelos propuestos por la misma librería, con el fin de encontrar el modelo con los mejores resultados para la clasificación de perros y gatos. Así bien, una vez identificado el mejor modelo se probará con diferentes hiperparámetros por medio de la librería de Keras Tuner.

C. G. Rodríguez estudia la Maestría en Ciencias en Inteligencia Artificial en la Universidad Autónoma de Querétaro (e-mail: @gmail.com)

II. Desarrollo

A. AUTO-KERAS

Auto-Keras es una librería open-source para AutoML (automated Machine Learning). Esta librería provee funciones que permiten la búsqueda automática de arquitecturas e hiperparámetros de modelos Deep Learning.

```
[ ]  !pip install autokeras
```

Fig.1. Instalación de la librería de Auto-Keras.

Para usuarios avanzados, se puede personalizar el espacio de búsqueda utilizando AutoModel en lugar de ImageClassifier, en donde se puede configurar ImageBlock para algunas configuraciones de alto nivel, por ejemplo, block_type para el tipo de red neuronal para buscar, normalizar para realizar la normalización de datos, aumentar para realizar el aumento de datos. Por otra parte, se puede no especificar estos argumentos, lo que dejaría que las diferentes opciones se ajusten automáticamente, que es lo que se probará en el presente trabajo.

```
[ ]  clf = ak.ImageClassifier(overwrite=True, max_trials=3)

     clf.fit(x_train, y_train, epochs=5)
```

Fig. 2. Configuración del número de épocas y el número de modelos a probar.

```
[ ]  puntaje_test = clf.evaluate(x_test,y_test)
     print('La mejor precisión probada en test es: ', 100*max(puntaje_test))
     puntaje_val = clf.evaluate(x_val,y_val)
     print('La mejor precisión probada en validation es: ', 100*max(puntaje_val))
```

Fig.3. Sección del código en donde se visualizará la mejor precisión del conjunto de prueba y validación

B. KERAS TUNER

Keras Tuner facilita la definición de un espacio de búsqueda y aprovecha los algoritmos incluidos para encontrar los mejores valores de hiperparámetros. Keras Tuner viene con algoritmos de optimización bayesiana, hiperbanda y búsqueda aleatoria incorporados, y también está diseñado para que los investigadores puedan extenderlo fácilmente para experimentar con nuevos algoritmos de búsqueda.

```
pip install keras-tuner
```

Fig.4. Instalación de la librería de Keras-Tuner.

Para ajustar un modelo, primero se creará un hipermodelo en el que se definen los distintos hiperparámetros para ajustar y cómo se ve el espacio de ajuste. Aquí está la función utilizada para construir el hipermodelo:

```
hypermodel = MiHiperModelo(num_classes=2)
```

Fig. 5. Función para la construcción del hipermodelo.

Posteriormente, será necesario seleccionar un sintonizador. Keras Tuner tiene cuatro sintonizadores disponibles, pero para el presente trabajo se utilizará el sintonizador RandomSearch, el cual como su nombre sugiere, prueba aleatoriamente una combinación de hiperparámetros de un espacio de búsqueda determinado.

```
tuner = RandomSearch(
    hypermodel,
    objective='val_accuracy',
    max_trials=10, #representa la cantidad de combinaciones de hiperparámetros que probará el sintonizador
    executions_per_trial = 1, #cantidad de modelos que se deben construir y ajustar para cada prueba con fines de robustez.
    directory='my_dir',
    project_name='helloworld6')
```

Fig. 6. Sintonizador RandomSearch.

C. Resultados del entrenamiento y la evaluación

AUTO-KERAS

En la siguiente tabla se puede visualizar los resultados obtenidos en la etapa la evaluación del mejor modelo obtenido por medio de la librería de Auto-keras.

TABLA 1
COMPARATIVA DE PRECISIÓN OBTENIDA DE LOS CONJUNTOS DE PRUEBA Y VALIDACIÓN.

	Mejor Precisión
Conjunto de prueba	70.74999809265137
Conjunto de Validación	69.49999928474426

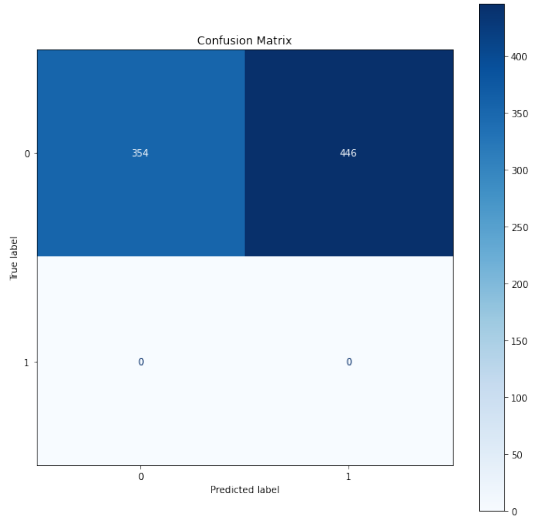

Fig. 7. Matriz de Confusión del mejor modelo .

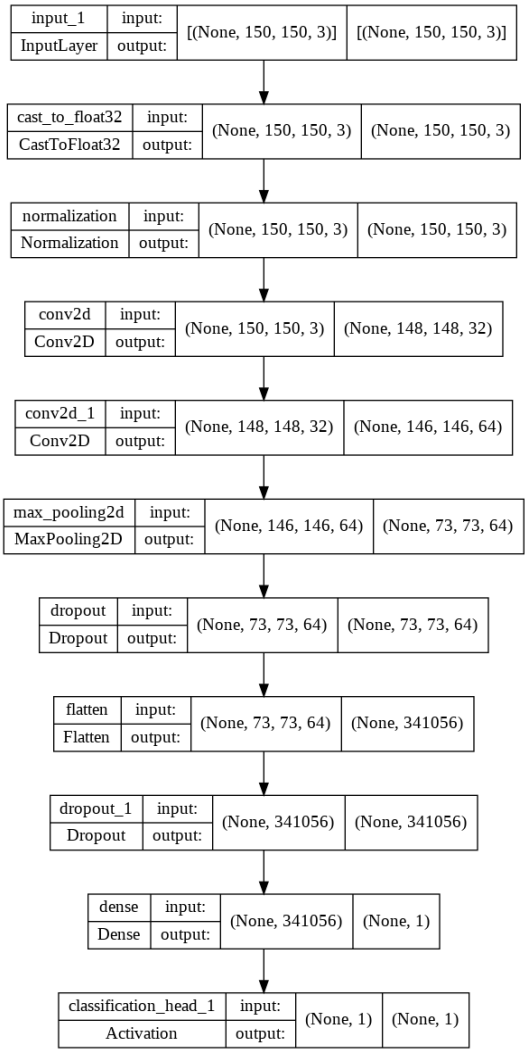

Fig. 8. Arquitectura del mejor modelo.

TABLA 2
REPORTE DE RESULTADOS DEL MEJOR MODELO

	Precisión	Recall	F1-Score	Support
Cats	1.00	0.44	0.61	800
Dogs	0.00	0.00	0.00	0
Accuracy			0.44	800
Macro avg	0.50	0.22	0.31	800
Weighted av	1.00	0.44	0.61	800

KERAS TUNER

Para la realización de esta parte, se decidió en utilizar el modelo propuesto anteriormente por la librería de Autokeras, en el cual se le brindaron distintos hiperparámetros para ajustar, como fue el tamaño de kernel, strides, pool, así como ciertos valores en cada convolución.

En la siguiente tabla se muestra el resultado del espacio de búsqueda en donde se pueden visualizar las mejores 10 combinaciones.

TABLA 3
RESUMEN DE LAS 10 MEJORES COMBINACIONES DEL ESPACIO DE BÚSQUEDA

Combinaciones	Hiperparámetros
Combinación 1	kernel_size1: 3 strides1: 1 pool_size1: 2 filters1: 192 activation: relu filters2: 256 learning_rate: 0.0001 Score: 0.7350000143051147
Combinación 2	kernel_size1: 3 strides1: 1 pool_size1: 4 filters1: 288 activation: tanh filters2: 256 learning_rate: 0.0001 Score: 0.7275000214576721
Combinación 3	kernel_size1: 3 strides1: 1 pool_size1: 4 filters1: 416 activation: tanh filters2: 128 learning_rate: 0.0001 Score: 0.7137500047683716
Combinación 4	kernel_size1: 3 strides1: 2 pool_size1: 4 filters1: 704 activation: tanh filters2: 384 learning_rate: 0.0001 Score: 0.7049999833106995
Combinación 5	kernel_size1: 7 strides1: 1 pool_size1: 4 filters1: 480 activation: tanh filters2: 384 learning_rate: 0.0001 Score: 0.6987500190734863

CONTINUACIÓN TABLA 3
RESUMEN DE LAS 10 MEJORES COMBINACIONES DEL ESPACIO DE BÚSQUEDA

Combinaciones	Hiperparámetros
Combinación 6	kernel_size1: 7 strides1: 1 pool_size1: 4 filters1: 800 activation: tanh filters2: 448 learning_rate: 0.0001 Score: 0.6924999952316284
Combinación 7	kernel_size1: 7 strides1: 2 pool_size1: 2 filters1: 608 activation: tanh filters2: 384 learning_rate: 0.0001 Score: 0.6837499737739563
Combinación 8	kernel_size1: 5 strides1: 2 pool_size1: 2 filters1: 608 activation: tanh filters2: 128 learning_rate: 0.0001 Score: 0.6712499856948853
Combinación 9	kernel_size1: 7 strides1: 1 pool_size1: 4 filters1: 64 activation: tanh filters2: 64 learning_rate: 0.0001 Score: 0.6675000190734863
Combinación 10	kernel_size1: 7 strides1: 1 pool_size1: 2 filters1: 64 activation: tanh filters2: 192 learning_rate: 0.0001 Score: 0.6274999976158142

Para la realización de la búsqueda, se decidió utilizar el conjunto de entrenamiento y el conjunto de validación, así como la cantidad de 10 épocas, en donde al final se seleccionó el modelo con el mejor resultado de validación, el cual se le realizo su correspondiente evaluación.

TABLA 4
RESULTADOS OBTENIDOS DE LA SECCIÓN DE EVALUACIÓN.

	Resultado obtenido
Precisión	0.8800
Perdida	0.3169

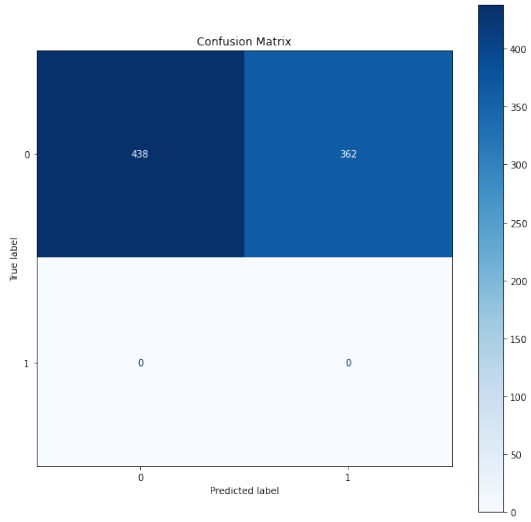

Fig. 9. Matriz del mejor modelo.

TABLA 5
REPORTE DE RESULTADOS DEL MEJOR MODELO

	Precisión	Recall	F1-Score	Support
Cats	1.00	0.55	0.71	800
Dogs	0.00	0.00	0.00	0
Accuracy			0.55	800
Macro avg	0.50	0.27	0.35	800
Weighted av	1.00	0.55	0.71	800

D. Análisis de resultados

La tabla 6, muestra los resultados obtenidos conforme a la evaluación de los modelos, en donde se puede observar que se obtuvo una mejor precisión en el hipermodelo así bien se pudo disminuir un poco el valor de perdida.

TABLA 6
COMPARATIVA DE PRECISIÓN OBTENIDA EN LOS DISTINTOS MÉTODOS.

	Precisión	Perdida
Autokeras	0.7074999809265137	0.5662453174591064
Hipermodelo	0.8800	0.3169

E. Conclusiones

La realización de esta tarea se enfocó en el reconocimiento de animales en específico de perros y gatos basado en imágenes por medio de la implementación de las librerías de Auto-Keras y Keras Tuner.

Los resultados obtenidos muestran que la utilización de estas librerías son un punto importante de partida, debido a que por medio de ellas es posible conseguir un guía sobre la el modelo de arquitectura, así como cuales serían los hiperparámetros adecuados para la base de datos propuesta con lo cual se reduciría de una manera considerable el tiempo invertido ajustando manualmente los hiperparámetros del modelo con el fin de alcanzar el rendimiento esperado.

Así bien, se pudo comprobar de nueva cuenta la importancia del uso de GPU para el desarrollo del trabajo actual, pues reduce los tiempos de ejecución notablemente. De igual forma es importante destacar la importancia de contar un tamaño considerable de memoria RAM con el fin de poder aumentar el número de modelos, así como el número de épocas al realizar la búsqueda del mejor modelo, así como al querer realizar una mayor cantidad de combinaciones para el ajuste óptimo de los hiperparámetros.

III. REFERENCIAS

[1] AutoKeras.(s.f) Clasificación de imágenes. Recuperado el día 25 de Marzo de 2022 de https://autokeras.com/tutorial/image_classification/

[2] Colab (2020) Introducción al afinador Keras. Recuperado el día 25 de Marzo de 2022 de https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/keras_tuner.ipynb#scrollTo=sKwLOzKpFGAj

[3] ICHI.PRO (s.f) Optimización de hiperparámetros con el sintonizador de Keras, parte 1. Recuperado el día 25 de Marzo de 2022 de https://ichi.pro/es/optimizacion-de-hiperparametros-con-el-sintonizador-de-keras-parte-1-68249217029489

[4] ICHI.PRO (s.f) Ajuste de hiperparámetros con Keras Tuner. Recuperado el día 25 de Marzo de 2022 de https://ichi.pro/es/ajuste-de-hiperparametros-con-keras-tuner-2762860380094

[5] Turgut090 (s.f) R interface to Keras Tuner. Recuperado el 25 de Marzo de 2022 de https://github.com/EagerAI/kerastuneR

[6] Prost, J. (s.f) keras-tuner-tutorial. Recuperado el 25 de Marzo de 2022 dehttps://github.com/JulieProst/keras-tuner-tutorial/blob/master/hypermodels.py