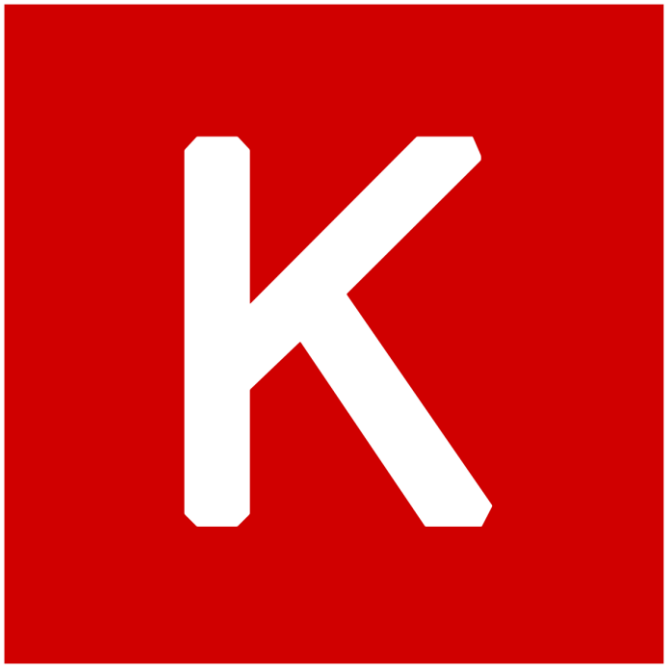
9 DE OCTUBRE DE 2022



TRABAJO SEMANA 9-10

REDES NEURONALES ARTIFICIALES Y DEEP LEARNING

UNIVERSIDAD AUTONOMA DE OCCIDENTE
Santiago de Cali, Valle del Cauca, Colombia

Brahyan Camilo Marulanda Muñoz	Diego Iván Perea Montealegre	Daniel Alejandro Tobar Álvarez	Henry Carmona Collazos
2186092	2185751	2185884	2185965
brahyan.marulanda@uao.edu.co	Diego.perea@uao.edu.co	Daniel_ale.tobar@uao.edu.co	Henry.carmona@uao.edu.co

Inciso 1. Seleccione un conjunto de datos ubicados en el siguiente enlace para un problema de clasificación (diferente al wine y al iris data set): https://archive.ics.uci.edu/ml/index.html Entrene una red neuronal MLP superficial y profunda que realice la clasificación definida para el problema seleccionado usando Tensorflow-Keras. Tenga en cuenta entrenar y validar la red con datos no usados en el proceso de entrenamiento.

Se selecciona el data set Raisin, el cual está compuesto de en un conjunto de 7 características morfológicas obtenidas de imágenes de variedades de pasas Kecimen y Besni con el fin de ser clasificadas. En este se hizo uso de 900 granos de pasas, conformados por las mismas cantidades para ambas variaciones de pasas.

A continuación, se presentan los atributos que conforman el data set:

1.) Area: Da el número de píxeles dentro de los límites de la pasa.

2.) Perimeter: Mide el entorno calculando la distancia entre los límites de la pasa y los píxeles que la rodean.

3.) MajorAxisLength: Da la longitud del eje principal, que es la línea más larga que se puede dibujar en la pasa.

4.) MinorAxisLength: Da la longitud del eje menor, que es la línea más corta que se puede dibujar en la pasa.

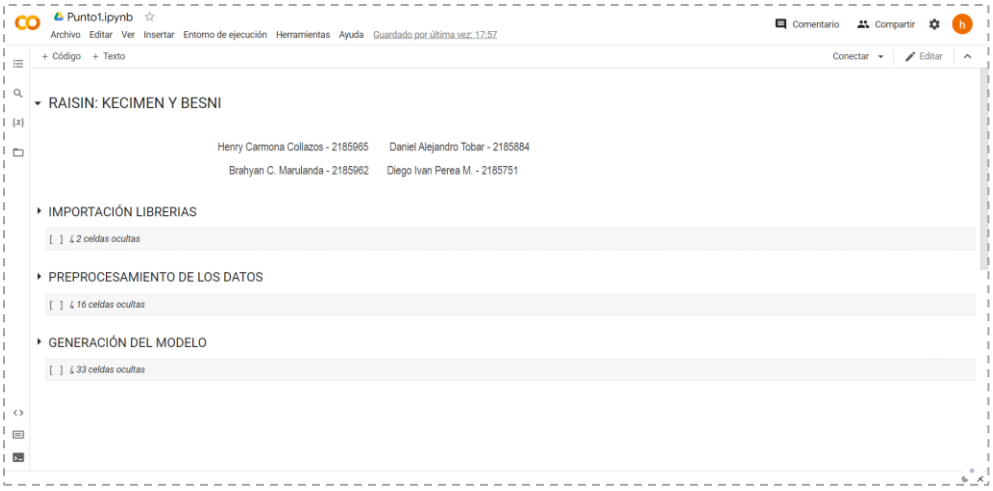
5.) Eccentricity: Da una medida de la excentricidad de la elipse, que tiene los mismos momentos que las pasas.

6.) ConvexArea: Da el número de píxeles de la cáscara convexa más pequeña de la región formada por la pasa.

7.) Extent: Da la relación entre la región formada por la pasa y el total de píxeles de la caja delimitadora.

8.) Class: Kecimen y Besni raisin.

A partir de esta información se realiza la arquitectura presente en el Google Colab.



Inciso 2. En el siguiente enlace se resuelve el problema de clasificación de imágenes definido por el data ser Cifar10 usando una red neuronal convolucional https://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html

a) Explique en que consiste el problema de clasificación trabajado (CIFAR10 data set)

El problema de clasificación trabajado con el dataset CIFAR-10 consta de 60000 imágenes en color de 32x32 en 10 clases, con 6000 imágenes por clase. Hay 50000 imágenes de entrenamiento y 10000 de prueba. El conjunto de datos se divide en cinco lotes de entrenamiento y un lote de prueba, cada uno con 10000 imágenes. El lote de prueba contiene exactamente 1000 imágenes seleccionadas al azar de cada clase. Los lotes de entrenamiento contienen el resto de las imágenes en orden aleatorio, pero algunos lotes de entrenamiento pueden contener más imágenes de una clase que de otra. Entre ellos, los lotes de entrenamiento contienen exactamente 5000 imágenes de cada clase [1].

Según paperswithcode en [2], los criterios para decidir si una imagen pertenece a una clase fueron los siguientes:

- El nombre de la clase debe ocupar un lugar destacado en la lista de respuestas probables a la pregunta "¿Qué hay en esta imagen?"
- La imagen debe ser fotorrealista. Se instruyó a los etiquetadores para que rechazaran los dibujos de líneas.
- La imagen debe contener solo una instancia destacada del objeto al que se refiere la clase. El objeto puede estar parcialmente ocluido o visto desde un punto de vista inusual, siempre y cuando su identidad siga siendo clara para el etiquetador.

Según Tensor Flow en [3], el dataset está dividido o tiene un Split que permite que se maneje tanto el entrenamiento de los modelos como su validación o testeo. De los 60000 datos que se tienen, el 16.67% son de testeo y el 83.33 son para entrenamiento:

'test'	10,000
'train'	50,000

Además, su estructura se puede apreciar de la siguiente manera:

```
FeaturesDict({
    'id': Text(shape=(), dtype=tf.string),
    'image': Image(shape=(32, 32, 3), dtype=tf.uint8),
    'label': ClassLabel(shape=(), dtype=tf.int64, num_classes=10),
})
```

Debido a lo anterior, es posible tener en cuenta una deducción de cuales podrían ser las características estipuladas en el dataset. Por eso, se tiene:

Característica	Clase	Forma	Dtype	Descripción: _____
	CaracterísticasDict			
identificación	Mensaje de texto		tf.string	
imagen	Imagen	(32, 32, 3)	tf.uint8	
etiqueta	ClassLabel		tf.int64	

La implementación del conjunto de datos CIFAR-10 o CIFAR-10 dataset ha sido efectivo en diferentes aplicaciones. Un ejemplo práctico del uso de una RN con CIFAR es que se publica en la página web de ,unipython donde hacen uso de las CNN (Redes Neuronales Convolucionales) donde se tiene una serie diferente de imágenes y se hace la salvedad que los valores de los píxeles están en el rango de 0 a 255 para cada uno de los canales rojo, verde y azul. Por eso, es considerado como una buena práctica trabajar con datos normalizados. Debido a que los valores de entrada son bien conocidos, podemos fácilmente normalizar al rango de 0 a 1 dividiendo cada valor por la observación máxima que es 255 [4].

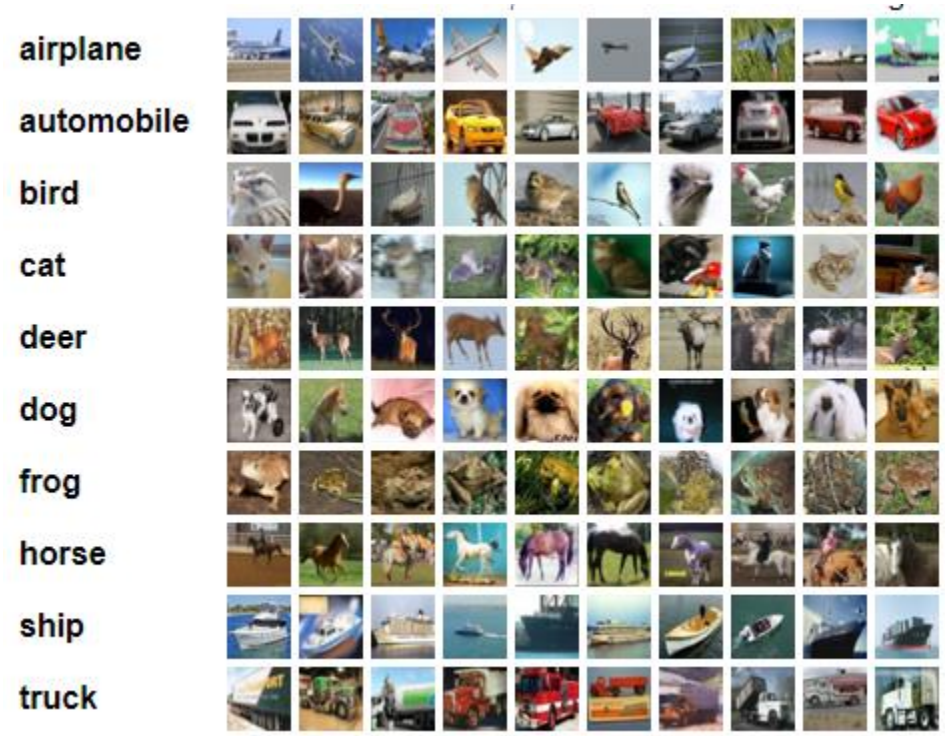
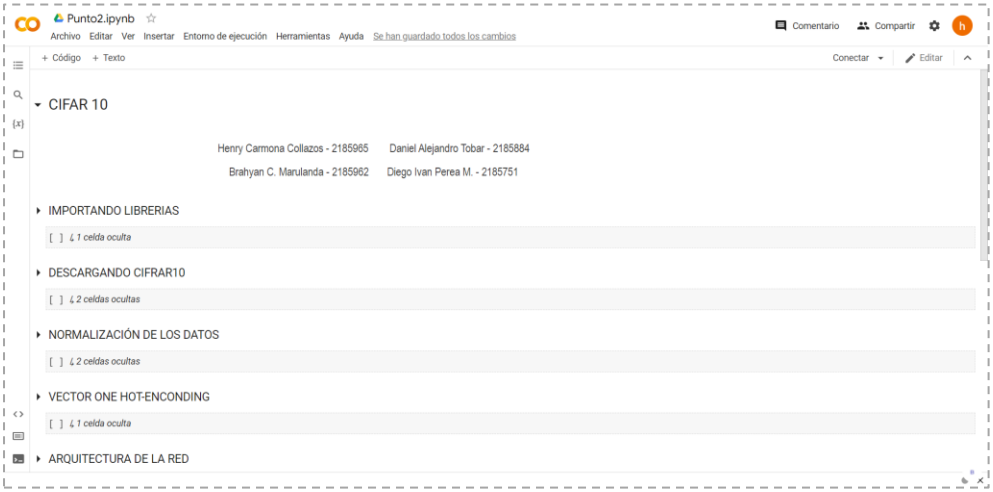


Ilustración 1. Las 10 Clases disponibles en el Dataset.

Es importante saber que las clases son completamente excluyentes. No hay solapamiento entre los automóviles y los camiones. "Automóvil" incluye berlinas, todoterrenos y cosas de ese tipo. "Camión" incluye sólo los camiones grandes. Tampoco incluye las camionetas.

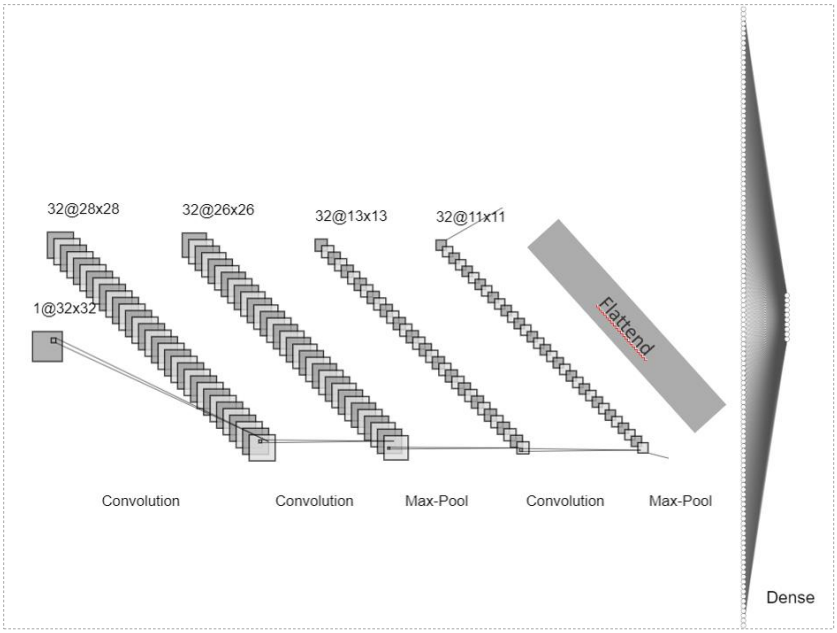
b) Entrene una red neuronal convolucional en Tensorflow-Keras que resuelva este problema

Para el entrenamiento de la red neuronal se hace uso del siguiente [Google Colab]



c) Represente gráficamente la red neuronal convolucional por usted usada

El grafico presentado contiene inicialmente una imagen de 32x32, la cual pasa a realizar una convolución con 32 filtros o kernels de 5x5, obteniendo como resultado una imagen a la salida de 28x28, así mismo, se realiza otra convolución con 32 kernels de tamaño 3x3 lo cual da una imagen de 26x26 a la salida. Luego se realiza un Max Pooling de 2x2, reduciendo la imagen a una de 13x13. Después, se realiza una convolución más, con 32 filtros de tamaño 3x3, obteniendo imágenes de 11x11. Seguido de ello, se tiene otro Max Pooling de 2x2, el cual da una imagen de 5x5, la cual pasara a realizar un Dropout del 30% y un Flatten, el cual entrega la información de forma aplanada de forma que pueda ser procesada por 2 capas densas, una de 512 y otra de 256 neuronas, para finalmente pasar a la capa de salida de 10 neuronas, debido al numero de clases.



d) Calcule la cantidad de parámetros por cada una de las capas de la red neuronal convolucional utilizada

Para el cálculo de los parámetros de las convoluciones se multiplica el tamaño del kernel por la cantidad de filtros por la cantidad de imágenes sumado con el bias.

Cálculo de parámetros para la primera convolución:

$$Parametro_1 = 5 * 5 * 32 * 3 + 32$$

$$Parametro_1 = 2432$$

Cálculo de parámetros la segunda convolución:

$$Parametro_2 = 3 * 3 * 32 * 32 + 32$$

$$Parametro_2 = 9248$$

Cálculo de parámetros para la tercera convolución:

$$Parametro_3 = 3 * 3 * 32 * 32 + 32$$

$$Parametro_3 = 9248$$

Cálculo de parámetros para la primera capa densa:

Para la capa densa se requiere inicialmente, calcular el valor de salida del Flattend, el cual corresponde al tamaño de imagen de salida del ultimo Max-Pool multiplicado por la cantidad de imágenes.

$$Salida_{Flattend} = (5 * 5 * 32) = 800$$

$$Parametro_4 = 512 * 800 + 512$$

$$Parametro_4 = 410112$$

Cálculo de parámetros para la segunda capa densa:

$$Parametro_5 = 512 * 256 + 256$$

$$Parametro_5 = 131328$$

Cálculo de parámetros para la capa de salida:

$$Parametro_{salida} = 256 * 10 + 10$$

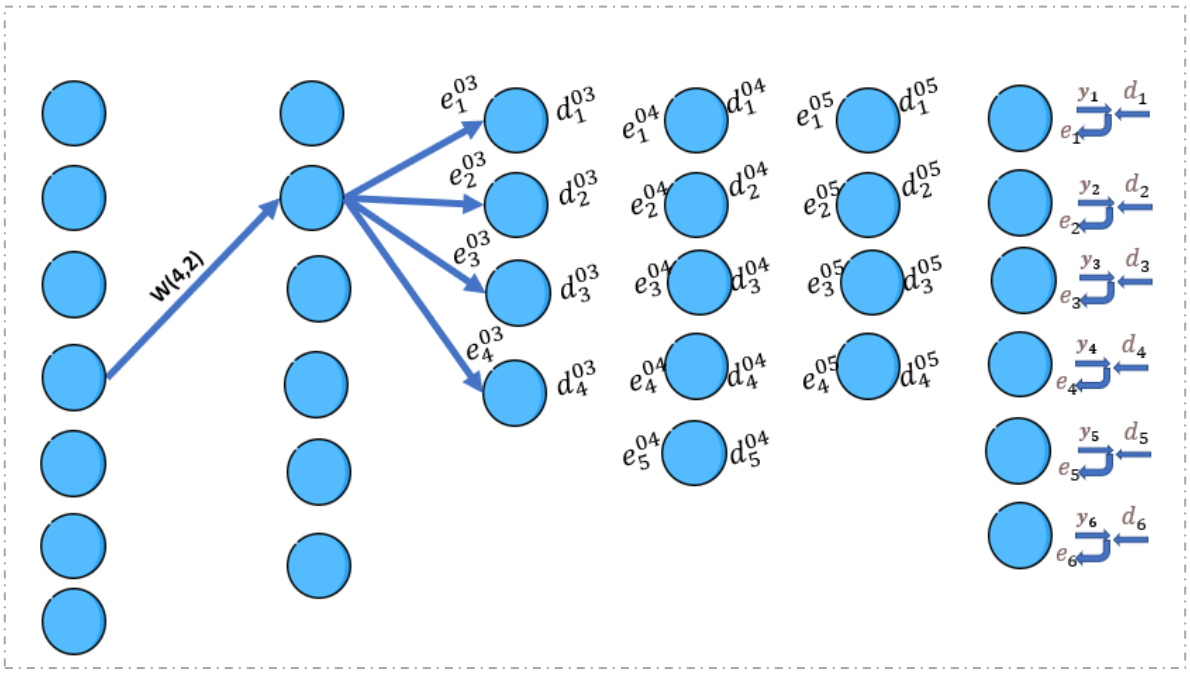
$$Parametro_{salida} = 2570$$

Cálculo del total de parámetros:

$$Parametro_{total} = Parametro_1 + Parametro_2 + Parametro_3 + Parametro_4 + Parametro_5 + Parametro_{salida}$$

$$Parametro_{total} = 564.938$$

Inciso 3. Se tiene una red neuronal con 6 capas de procesamiento y la siguiente cantidad de neuronas [7 6 4 5 4 6] y las siguientes funciones de activación [lineal, tangente-sigmoidal, simoidal, tangente-sigmoidal, lineal, cúbica], Muestre la expresión para entrenar el peso w(4,2) ubicado entre la primera y segunda capa considerando las funciones de activación definidas.



Hallazgo de error de la capa de salida

$$e^{06} = y - y'$$

$$\delta^{06} = e^{06} fact(neta^{06})$$

$$e^{05} = w^{05}\delta^{06}$$

$$\delta^{05} = e^{05} fact(neta^{05})$$

$$e^{04} = w^{04}\delta^{05}$$

$$\delta^{04} = e^{04} fact(neta^{04})$$

$$e^{03} = w^{03}\delta^{05}$$

$$\delta^{03} = e^{03} fact(neta^{03})$$

Conclusión de expresión:

$$e_2^{02} = \sum_{i=1}^{4} w_{i2}^{02}\delta_i^{03}$$

$$\delta_2^{02} = e_2^{02} fact_2(neta_2^{02})$$

$$w_{24}^{01}(t+1) = w_{24}^{01}(t) + \propto \delta_2^{02} y_2^{01}$$

Inciso 4. Se tiene una red con 4 entradas, 4 neuronas ocultas tangentes sigmoidales y 4 neuronas de salidas sigmoidales. Asuma un valor de razón de aprendizaje igual a 0.60. Encuentre la actualización del valor del bias de la primer neurona de la capa oculta y del peso w(2,4) de la capa de salida.

$$W^o = \begin{bmatrix} 2.3 & 2.0 & 0.9 & 1.2 \\ 0.7 & -0.4 & 0.5 & 1.0 \\ -1.0 & 1.1 & -0.3 & 0.8 \\ 1.2 & 1.5 & 2.5 & -0.3 \end{bmatrix} \quad W^s = \begin{bmatrix} 1.3 & -1.1 & -0.6 & 1.2 \\ 0.7 & 1.4 & 2.0 & 1.0 \\ -0.4 & 1.6 & 0.8 & 1.5 \\ 0.8 & 1.5 & 0.5 & -0.3 \end{bmatrix}$$

$$Bco = [-1.0 \quad 0.6 \quad 3.3 \quad -1.0] \qquad Bcs = [0.4 \quad -0.7 \quad 1.2 \quad 0.9]$$

$$X = [\,3.0, -2.0, 1.0, -1.5] \qquad D = [\,0.5, 1.0, 0.5, 1.0]$$

$$neta^0 = w^0 * x + b_{co}$$

$$y^0 = \tanh\,(neta^0)$$

$$neta^s = w^s * y^0 + b_{cs}$$

$$y^s = \text{sigmod}\,(neta^s)$$

$$e^s = d - y^s$$

$$\delta^s = e^s fact^s(neta^s)$$

$$e_2^0 = \sum_{i=1}^{2} w_{is}^s \delta^s$$

$$\delta_2^0 = e_2^{02} fact_2(neta_2^0)$$

$$B_2^0(t+1) = B_2^0(t) + \delta_2^0$$

$$w_{24}^s(t+1) = w_{24}^s(t) + \propto \delta_4^s y_2^s$$

$$b_{24}^s(t+1) = b_{24}^s(t) + \propto \delta_4^s y_2^s$$

$$w_{24}^s(t+1) = 0.5$$

$$b_{24}^s(t+1) = 0.2$$

Inciso 5. Realice una breve investigación sobre una arquitectura de red convolucional denominada EfficientNet.

EfficientNet

Las redes neuronales convolucionales (ConvNets) se desarrollan comúnmente con un presupuesto de recursos fijo y luego se amplían para una mejor precisión si hay más recursos disponibles. En aras de ser un poco más ambiciosas estas redes, se ha diseñado una nueva red de referencia y es escalada a una familia de modelos, llamados EfficientNets, que logran una precisión y eficiencia mucho mejores que las ConvNet anteriores. En particular, el EfficientNet-B7 logra una precisión top-1 del 84,3% en ImageNet, al tiempo que es 8,4 veces más pequeño y 6,1 veces más rápido en inferencia que el mejor ConvNet existente. Las EfficientNets también se transfieren bien y logran una precisión de última generación en CIFAR-100 (91.7%), Flowers (98.8%) y otros 3 conjuntos de datos de aprendizaje de transferencia, con un orden de magnitud menos parámetros [5].

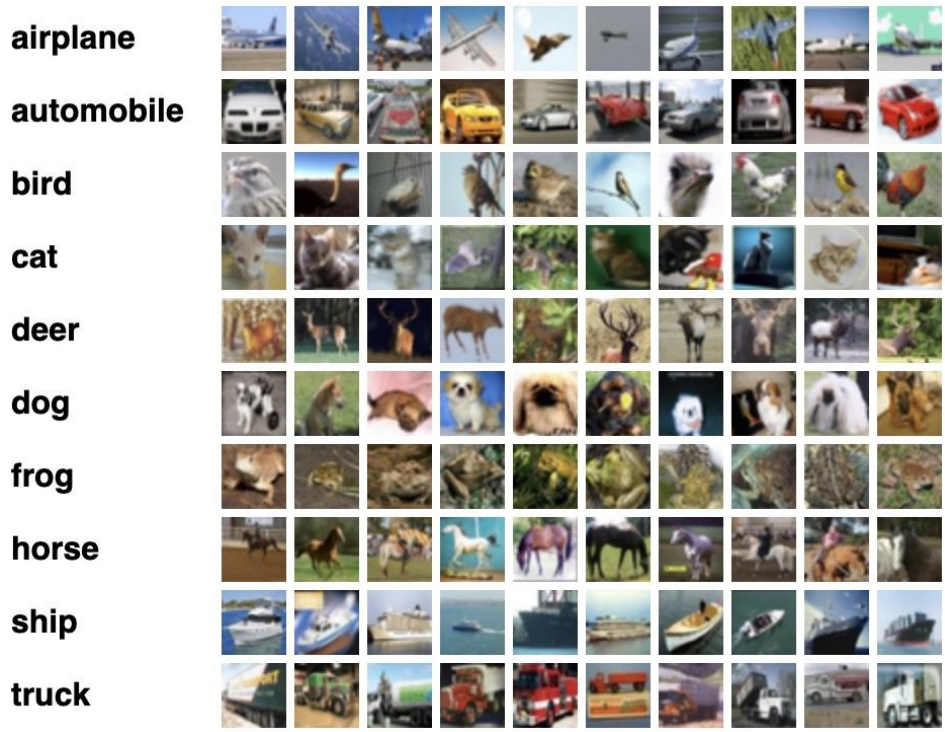


Ilustración 2. CIFAR 100 Dataset. Tomada de [6]

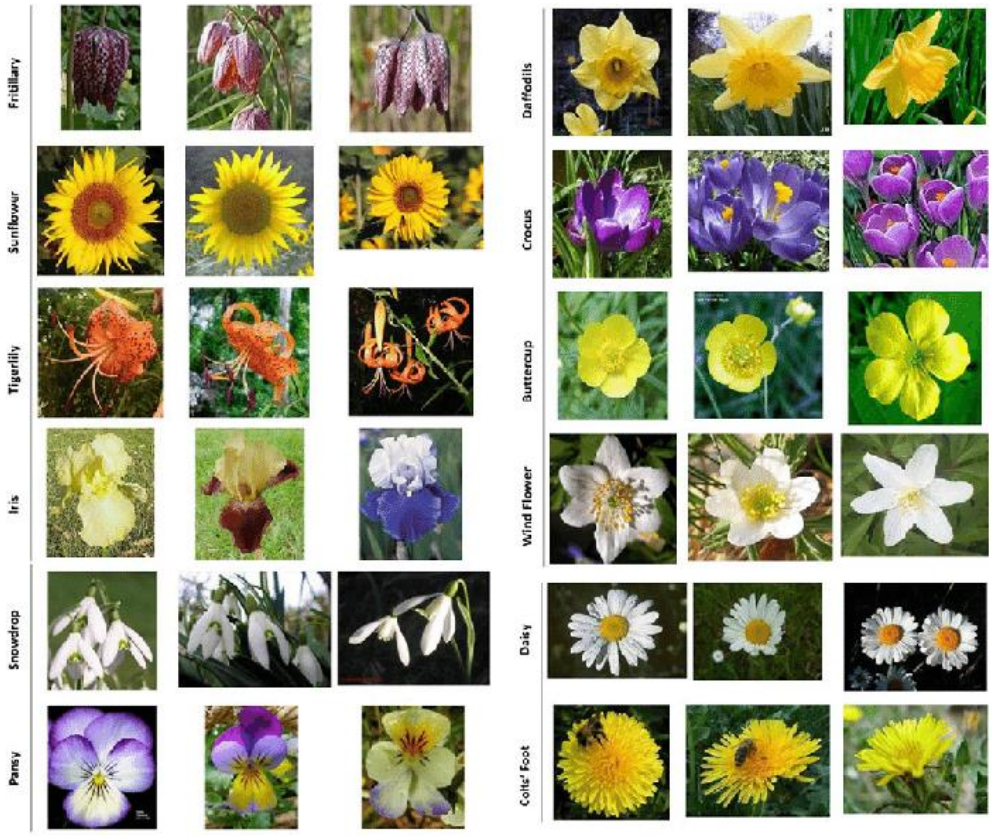


Ilustración 3. Flowers Dataset.

EfficientNet es una arquitectura de red neuronal convolucional y un método de escalado que escala uniformemente todas las dimensiones de profundidad/anchura/resolución utilizando un coeficiente compuesto. A diferencia de la práctica convencional que escala arbitrariamente estos factores, el método de escalado de EfficientNet escala uniformemente la anchura, la profundidad y la resolución de la red con un conjunto de coeficientes de escalados fijos. Por ejemplo, si se desea usar 2^N veces más recursos computacionales, simplemente es posible aumentar la profundidad de la red por α^N, ancho por β^N, y el tamaño de la imagen por γ^N, donde α,β, γ son coeficientes constantes determinados por una pequeña búsqueda de cuadrícula en el modelo pequeño original. EfficientNet utiliza un coeficiente compuesto Φ para escalar uniformemente el ancho, la profundidad y la resolución de la red de una manera basada en principios [7].

La eficacia del escalado del modelo también depende en gran medida de la red de referencia. Así pues, para mejorar aún más el rendimiento, también ha sido desarrollada una nueva red de referencia realizando una búsqueda de arquitectura neuronal mediante el Framework AutoML MNAS, que optimiza tanto la precisión como la eficiencia (FLOPS). La arquitectura resultante utiliza la convolución móvil de cuello de botella invertido (MBConv), similar a MobileNetV2 y MnasNet, pero es ligeramente más grande debido a un mayor presupuesto de FLOP. Los investigadores continuaron escalando el modelo de línea de base para obtener un conjunto de modelos EfficientNets:

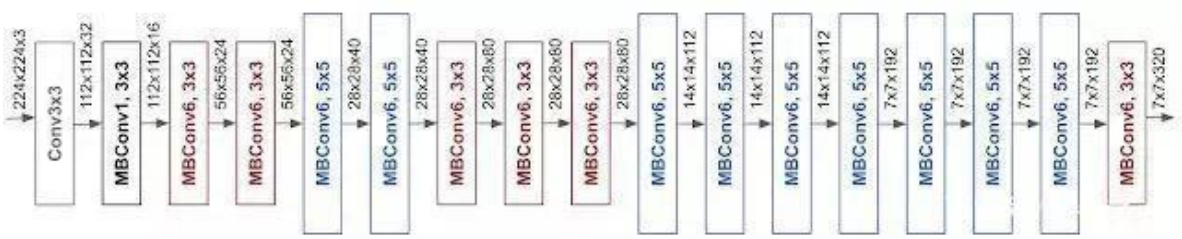


Ilustración 4. El modelo básico EfficientNet-B0 tiene una arquitectura simple y limpia, lo que facilita su expansión y generalización.

Rendimiento EfficientNet con Dataset ImageNet: Los investigadores compararon EfficientNets y los modelos CNN existentes en el conjunto de datos ImageNet. El modelo EfficientNet es más preciso y eficiente que el modelo CNN existente, y su cantidad de parámetros y FLOPS se han reducido en un orden de magnitud. Por ejemplo, en el modo de alta precisión, EfficientNet-B7 ha alcanzado la mejor precisión actual del 84,4% entre el 1 y el 97,1% entre los 5 primeros en ImageNet. La velocidad de inferencia de la CPU es 6,1 veces mayor que la de Gpipe, y el tamaño de este último es 8.4 veces de EfficientNet-B7. En comparación con el ResNet-50 ampliamente utilizado, EfficientNet-B4 utiliza FLOPS similares para lograr una precisión de primer nivel que es un 6,3% más alta que ResNet-50 (ResNet-50 76,3%, EfficientNet-B4 82,6%).

Ilustración 5. ImageNet Dataset: es una base de datos de imágenes organizada según la jerarquía de WordNet (actualmente solo los sustantivos), en la que cada nodo de la jerarquía está representado por cientos y miles de imágenes. El proyecto ha sido fundamental en el avance de la visión por computadora y la investigación de aprendizaje profundo. Los datos están disponibles de forma gratuita para los investigadores para uso no comercial [8].

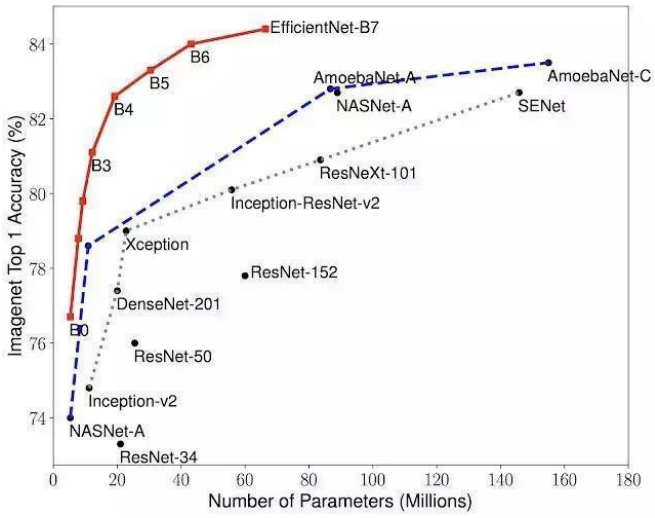


Ilustración 6. Comparación entre el tamaño del modelo y la precisión.

Explicación de la Imagen: EfficientNet-B0 es un modelo de línea de base desarrollado a través de AutoML MNAS, y Efficient-B1 a B7 son redes obtenidas después de extender el modelo de línea de base. EfficientNet es significativamente mejor que otras CNN. Específicamente, EfficientNet-B7 ha logrado nuevos resultados de SOTA: 84,4% de precisión entre 1 y 97,1% entre los 5 primeros, y su tamaño es mucho más pequeño que el modelo óptimo anterior de CNN GPipe (el último tamaño del modelo es EfficientNet- 8.4 veces de B7), la velocidad es 6.1 veces de GPipe. El parámetro de EfficientNet-B1 es mucho más pequeño que el de ResNet-152, pero la velocidad es 5,7 veces mayor que la de este último.

Table 2. **EfficientNet Performance Results on ImageNet** (Russakovsky et al., 2015). All EfficientNet models are scaled from our baseline EfficientNet-B0 using different compound coefficient ϕ in Equation 3. ConvNets with similar top-1/top-5 accuracy are grouped together for efficiency comparison. Our scaled EfficientNet models consistently reduce parameters and FLOPS by an order of magnitude (up to 8.4x parameter reduction and up to 16x FLOPS reduction) than existing ConvNets.

Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPS	Ratio-to-EfficientNet
EfficientNet-B0	**76.3%**	**93.2%**	**5.3M**	**1x**	**0.39B**	**1x**
ResNet-50 (He et al., 2016)	76.0%	93.0%	26M	4.9x	4.1B	11x
DenseNet-169 (Huang et al., 2017)	76.2%	93.2%	14M	2.6x	3.5B	8.9x
EfficientNet-B1	**78.8%**	**94.4%**	**7.8M**	**1x**	**0.70B**	**1x**
ResNet-152 (He et al., 2016)	77.8%	93.8%	60M	7.6x	11B	16x
DenseNet-264 (Huang et al., 2017)	77.9%	93.9%	34M	4.3x	6.0B	8.6x
Inception-v3 (Szegedy et al., 2016)	78.8%	94.4%	24M	3.0x	5.7B	8.1x
Xception (Chollet, 2017)	79.0%	94.5%	23M	3.0x	8.4B	12x
EfficientNet-B2	**79.8%**	**94.9%**	**9.2M**	**1x**	**1.0B**	**1x**
Inception-v4 (Szegedy et al., 2017)	80.0%	95.0%	48M	5.2x	13B	13x
Inception-resnet-v2 (Szegedy et al., 2017)	80.1%	95.1%	56M	6.1x	13B	13x
EfficientNet-B3	**81.1%**	**95.5%**	**12M**	**1x**	**1.8B**	**1x**
ResNeXt-101 (Xie et al., 2017)	80.9%	95.6%	84M	7.0x	32B	18x
PolyNet (Zhang et al., 2017)	81.3%	95.8%	92M	7.7x	35B	19x
EfficientNet-B4	**82.6%**	**96.3%**	**19M**	**1x**	**4.2B**	**1x**
SENet (Hu et al., 2018)	82.7%	96.2%	146M	7.7x	42B	10x
NASNet-A (Zoph et al., 2018)	82.7%	96.2%	89M	4.7x	24B	5.7x
AmoebaNet-A (Real et al., 2019)	82.8%	96.1%	87M	4.6x	23B	5.5x
PNASNet (Liu et al., 2018)	82.9%	96.2%	86M	4.5x	23B	6.0x
EfficientNet-B5	**83.3%**	**96.7%**	**30M**	**1x**	**9.9B**	**1x**
AmoebaNet-C (Cubuk et al., 2019)	83.5%	96.5%	155M	5.2x	41B	4.1x
EfficientNet-B6	**84.0%**	**96.9%**	**43M**	**1x**	**19B**	**1x**
EfficientNet-B7	**84.4%**	**97.1%**	**66M**	**1x**	**37B**	**1x**
GPipe (Huang et al., 2018)	84.3%	97.0%	557M	8.4x	-	-

We omit ensemble and multi-crop models (Hu et al., 2018), or models pretrained on 3.5B Instagram images (Mahajan et al., 2018).

Ilustración 7. El rendimiento de EfficientNet en ImageNet. Tomado de [9].

Aunque las EfficientNets funcionan bien en ImageNet, para ser más útiles, deberían tener la capacidad de migrar a otros conjuntos de datos. Los investigadores de Google evaluaron EfficientNets en 8 conjuntos de datos de aprendizaje de migración de uso común. Los resultados muestran que EfficientNets ha alcanzado la tasa de precisión óptima actual en 5 de estos conjuntos de datos, y la cantidad de parámetros se reduce considerablemente, lo que demuestra que EfficientNets tiene buenas capacidades de migración [10].

EfficientNets puede mejorar significativamente la eficiencia del modelo, y los investigadores de Google esperan que EfficientNets se pueda utilizar como una nueva base para futuras tareas de visión por computadora. Por lo tanto, los investigadores abrieron el modelo EfficientNet cuyo código en Open Source y script de entrenamiento de TPU (Unidad de Procesamiento Tensorial), se encuentra en https://github.com/tensorflow/tpu/tree/master/models/official/efficientnet y https://cloud.google.com/tpu/.

Además de ello, se tiene información en TensorFlow [11], donde se adecua el módulo `tf.keras.applications.efficientnet` y se tiene información de cada una de las Funciones, así:

EfficientNetB0(...): Instantiates the EfficientNetB0 architecture.
EfficientNetB1(...): Instantiates the EfficientNetB1 architecture.
EfficientNetB2(...): Instantiates the EfficientNetB2 architecture.
EfficientNetB3(...): Instantiates the EfficientNetB3 architecture.
EfficientNetB4(...): Instantiates the EfficientNetB4 architecture.
EfficientNetB5(...): Instantiates the EfficientNetB5 architecture.
EfficientNetB6(...): Instantiates the EfficientNetB6 architecture.
EfficientNetB7(...): Instantiates the EfficientNetB7 architecture.
decode_predictions(...): Decodes the prediction of an ImageNet model.
preprocess_input(...): A placeholder method for backward compatibility.

Cada una de ellas es discriminada en Keras. Por ejemplo, `EfficientNetB0` es una función que devuelve un modelo de clasificación de imágenes Keras, opcionalmente cargado con pesas previamente entrenadas en ImageNet. Los argumentos explicados de la función y las demás funciones con sus respectivos argumentos se encuentran en la página web de Keras: https://keras.io/api/applications/efficientnet/.

```
tf.keras.applications.EfficientNetB0(
    include_top=True,
    weights="imagenet",
    input_tensor=None,
    input_shape=None,
    pooling=None,
    classes=1000,
    classifier_activation="softmax",
    **kwargs
)
```

Iniciso 6. En el siguiente enlace se resuelve un problema de clasificación de imágenes usando una red neuronal convolucional https://poloclub.github.io/cnn-explainer/

a) Describa el ejercicio de clasificación realizado

El problema de clasificación de imágenes de CNN Explainer, consta de 10 imágenes en color de tamaño 64x64, las cuales se ubican en 10 clases. Es posible añadir nuevas imágenes a la red, para comprobar su funcionamiento. Las clases que se encuentran en el data set son las siguientes: Lifeboat, Ladybug, Pizza, Bell Pepper, School Bus, Koala, Espresso, Red Panda, Orange y Sport Car.

b) Represente gráficamente la red neuronal convolucional usada

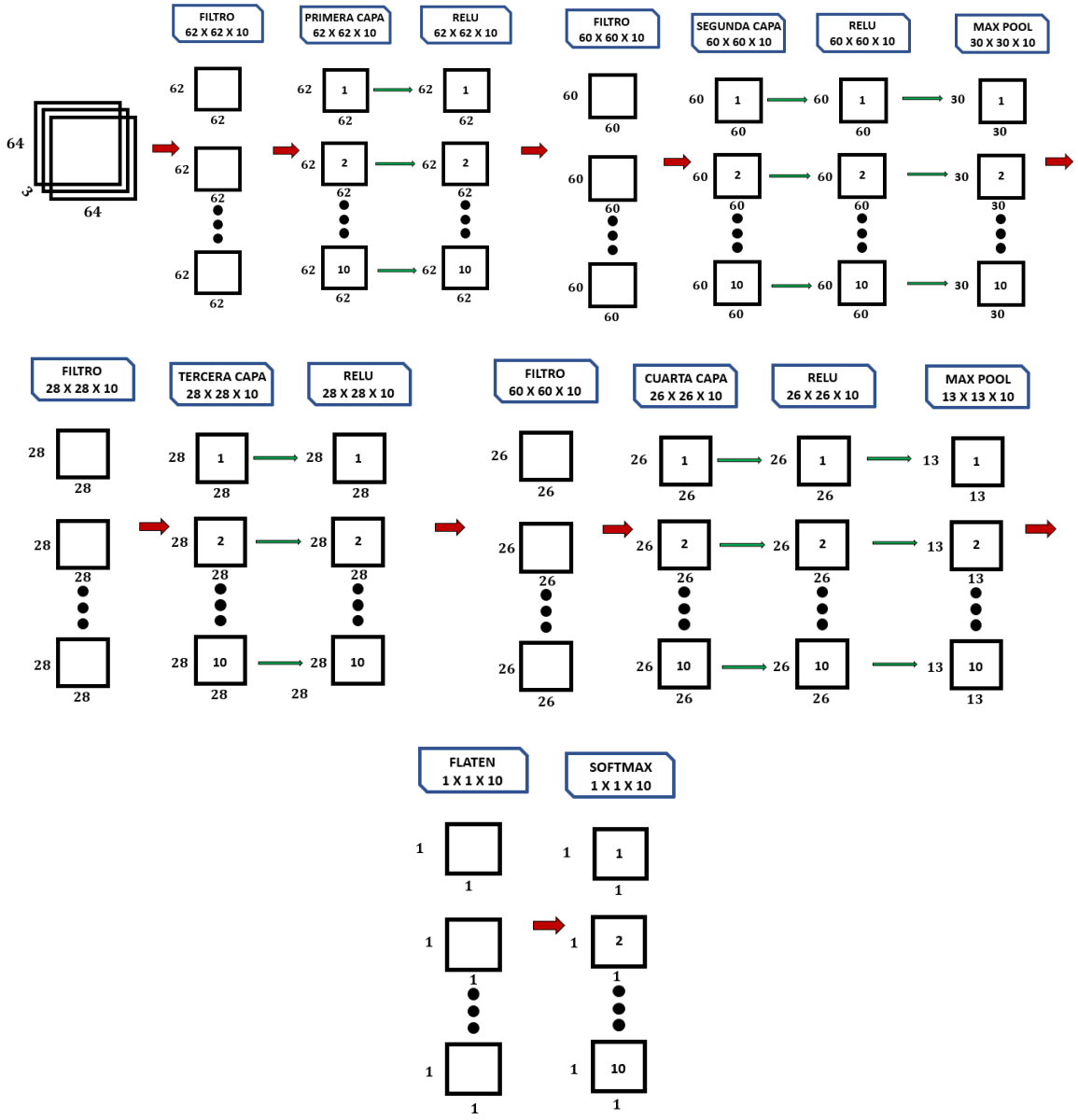


Figura . Arquitectura gráficamente de la CNN Explainer

c) Calcule la cantidad de parámetros por cada una de las capas de la red neuronal convolucional utilizada.

Cálculo de parámetros para la primera capa:

$$Parametro_1 = 3 * 3 * 10 * 3 + 10$$

$$Parametro_1 = 280$$

Cálculo de parámetros la segunda capa:

$$Parametro_2 = 3 * 3 * 10 * 10 + 10$$

$$Parametro_2 = 910$$

Cálculo de parámetros para la tercera capa:

$$Parametro_3 = 3 * 3 * 10 * 10 + 10$$

$$Parametro_3 = 910$$

Cálculo de parámetros para la cuarta capa:

$$Parametro_4 = 3 * 3 * 10 * 10 + 10$$

$$Parametro_4 = 910$$

Cálculo de parámetros para la capa de salida:

$$Parametro_{salida} = 10 * 13 * 13 * 10 + 10$$

$$Parametro_{salida} = 16910$$

Cálculo del total de parámetros:

$$Parametro_{total} = Parametro_1 + Parametro_2 + Parametro_3 + Parametro_4 + Parametro_{salida}$$

$$Parametro_{total} = 19920$$

Referencias

[1] "CIFAR-10 and CIFAR-100 datasets". http://www.cs.toronto.edu/~kriz/cifar.html (consultado oct. 07, 2022).

[2] "| del conjunto de datos CIFAR-10 Papeles con código". https://paperswithcode.com/dataset/cifar-10 (consultado oct. 07, 2022).

[3] "cifar10 | Conjuntos de datos de TensorFlow". https://www.tensorflow.org/datasets/catalog/cifar10 (consultado oct. 07, 2022).

[4] "Proyecto: Reconocimiento de Objetos de en Fotografías con CIFAR-10 - ▷ Cursos de Programación de 0 a Experto © Garantizados". https://unipython.com/proyecto-reconocimiento-de-objetos-de-en-fotografias-con-cifar-10/ (consultado oct. 07, 2022).

[5] M. Tan y Q. v. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”, *36th International Conference on Machine Learning, ICML 2019*, vol. 2019-June, pp. 10691–10700, may 2019, doi: 10.48550/arxiv.1905.11946.

[6] “CIFAR-100 Dataset | Papers With Code”. https://paperswithcode.com/dataset/cifar-100 (consultado oct. 07, 2022).

[7] “EfficientNet”. https://paperswithcode.com/method/efficientnet (consultado oct. 07, 2022).

[8] “ImageNet (en inglés)”. https://www.image-net.org/ (consultado oct. 07, 2022).

[9] “Google propone una nueva red convolucional EfficientNet: la velocidad de inferencia se incrementa en 5.1 veces y los parámetros se reducen en un 88%, lo que requiere nuestra verificación - programador clic”. https://programmerclick.com/article/62541119575/ (consultado oct. 07, 2022).

[10] “Blog de Google AI: EfficientNet: Mejora de la precisión y la eficiencia a través de AutoML y Model Scaling”. https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html (consultado oct. 07, 2022).

[11] “Module: tf.keras.applications.efficientnet | TensorFlow v2.10.0”. https://www.tensorflow.org/api_docs/python/tf/keras/applications/efficientnet (consultado oct. 07, 2022).