



Universidad Tecnológica De Panamá
Facultad De Ingeniería De Sistemas Computacionales
Lic. en Ing. de Sistemas y Computación



Memoria de Trabajo para la Asignatura:
Tópicos I – Visión Artificial

Laboratorio N° 8- Aprendizaje Automático en Visión Artificial

Integrantes

Charles Chuez 8-960-2188

Grupo

1IL141

Profesor

José Carlos Rangel Ortiz

II Semestre, 2024

Introducción

Dentro de los algoritmos que utilizan redes neuronales se pueden citar las Redes Neuronales Convolucionales (CNN, por sus siglas en inglés) las cuales se utilizan principalmente en el ámbito de la visión artificial, en problemas de localización, detección y clasificación de objetos, personas y lugares.

En este laboratorio se presentará una forma de construir modelos de clasificación utilizando una CNN. La diferencia radicará en que durante esta parte del laboratorio el modelo será entrenado utilizando datos (imágenes) que se encuentran localmente en nuestra computadora, o dicho de otra manera, un dataset local y también mediante una librería especializada en el trabajo con redes neuronales profundas. El laboratorio muestra todo el procedimiento y permite también evaluar el modelo generado utilizando una imagen externa a las disponibles en el dataset.

De igual manera se presenta la forma para la utilización de un modelo que ha sido previamente entrenado con un dataset de gran tamaño y con arquitecturas de múltiples capas.

Resultados

Parte 1 y 2

Luego de la ejecución del código fuente aproximadamente ¿qué tiempo tomo para producir y guardar el modelo?

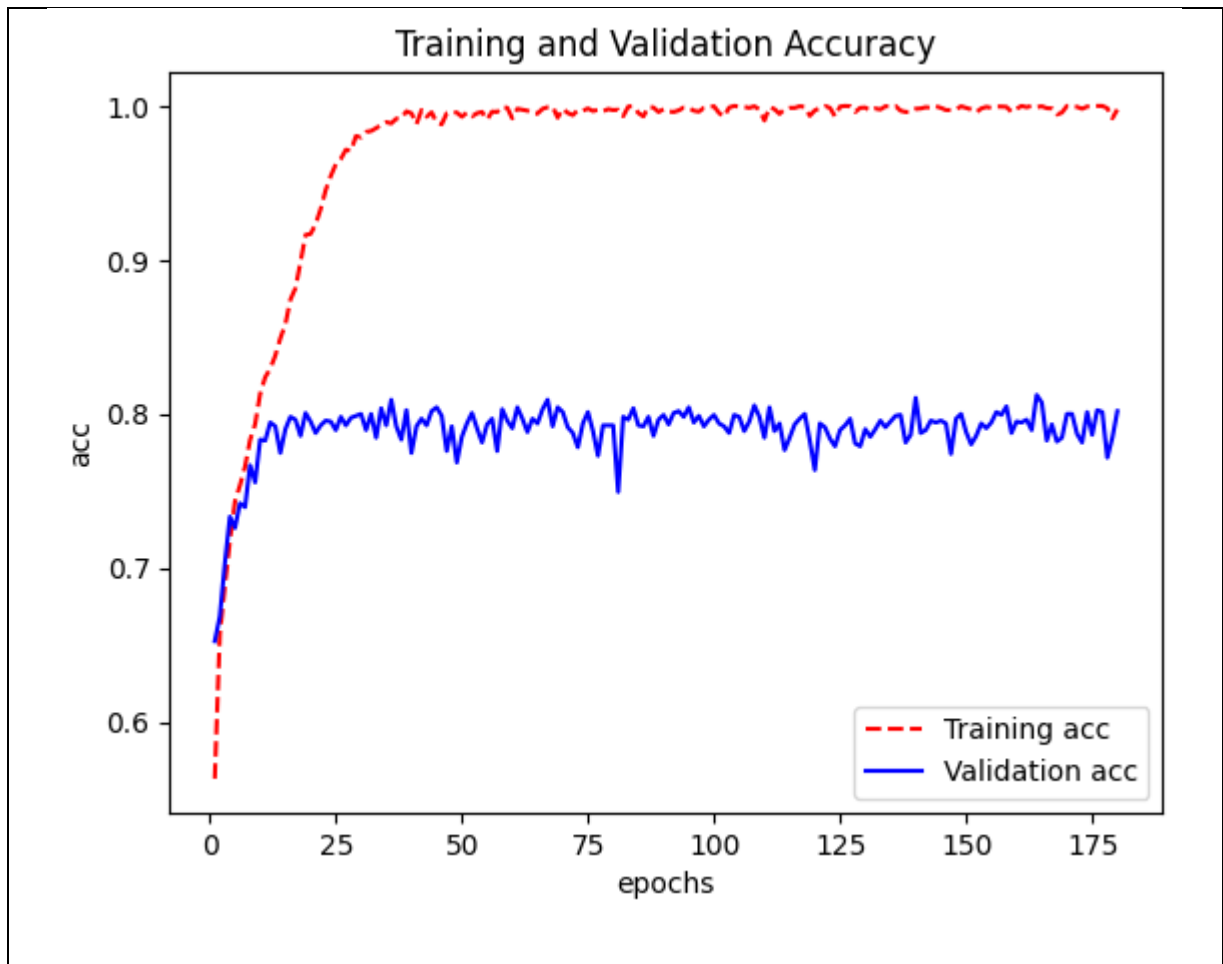
```
# In[14]:
history = model.fit(
    train_generator,
    steps_per_epoch = steps_per_epoch,
    epochs = 180,
    #epochs = 70,
    validation_data = validation_generator,
    validation_steps = validation_steps,
    verbose = 2
)

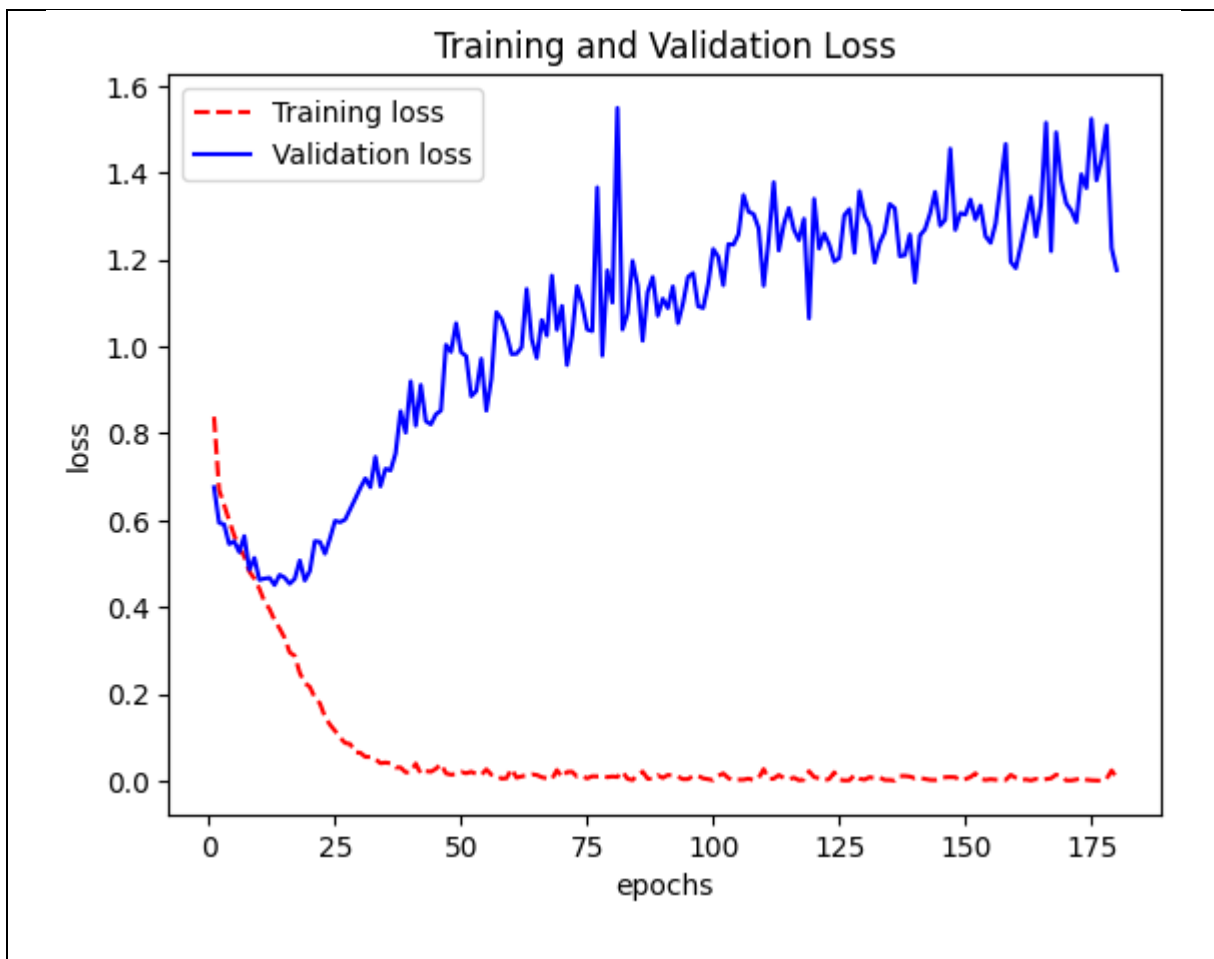
# Guardar el modelo entrenado
model.save('pets_categorical.h5')
```

[39] ✓ 104m 48.5s

... Epoch 1/180
486/486 - 36s - 73ms/step - acc: 0.5632 - loss: 0.8382 - val_acc: 0.6525 - val_loss: 0.6757
Epoch 2/180
486/486 - 33s - 69ms/step - acc: 0.6570 - loss: 0.6696 - val_acc: 0.6685 - val_loss: 0.5941

Presente una captura de la consola o jupyter notebook luego de la ejecución de este código, con las gráficas de accuracy producidas.





¿Cuál fue el accuracy y el loss obtenido por este modelo para train, validation y test? Puede presentar su respuesta completando la siguiente tabla.

SubGrupo	Accuracy	Loss
Train	0.9977366328239441	0.006117422133684158
Validation	0.8018518686294556	1.175984501838684
Test	0.7985	1.2808

Una vez entrenado el modelo, pruebe dicho modelo con el código de la Parte 2 del laboratorio, utilice 5 imágenes diferentes y que no estén presentes en el dataset y evalúe si el modelo es capaz de identificar correctamente el animal de la foto. Presente la evidencia de la ejecución de este código con los 5 elementos diferentes, así como también la clase producida por el modelo.

```
python src/lab8/CNN\ Test\ Categorical.py
~/git/portafolio_vision_artificial

(portafolio_vision_artificial)
# phoenix @ LAPTOP-VIKMPDAB in ~/git/portafolio_vision_artificial on git:main x portafolio_vision_artificial [18:13:05]
$ python src/lab8/CNN\ Test\ Categorical.py
2024-11-12 18:13:11.655962: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:477] Unable to register cuFFT factory: Attempting to register factory for plugin cuFFT which has already been registered
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
E0000 00:00:1731453191.678897 237406 cuda_dnn.cc:8310] Unable to register cuDNN factory: Attempting to register factory for plugin cuDNN when one has already been registered
E0000 00:00:1731453191.685258 237406 cuda_blas.cc:1418] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has already been registered
2024-11-12 18:13:11.709274: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance mode. To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
W0000 00:00:1731453194.192867 237406 gpu_device.cc:2344] Cannot dlopen some GPU libraries. Please make sure the missing libraries mentioned above are installed properly if you want to use GPU. Follow the guide at https://www.tensorflow.org/install/gpu for how to download and setup the required libraries for your platform.
Skipping registering GPU devices...
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. 'model.compile_metrics' will be empty until you train or evaluate the model.
1/1 [0. 1. 0.] 0s 82ms/step
1
dogs
/home/phoenix/git/portafolio_vision_artificial/src/lab8/CNN Test Categorical.py:29: UserWarning: FigureCanvasAgg is non-interactive, and thus cannot be shown
plt.show()
which: no node in (/home/phoenix/git/portafolio_vision_artificial/bin:/opt/cuda/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/opt/cuda/bin:/opt/cuda/nsight_compute/bin:/var/lib/flatpak/exports/bin:/usr/lib/jvm/default/bin:/usr/bin/site_perl:/usr/bin/vendor_perl:/usr/bin/core_perl:/usr/lib/rustup/bin)
* Running on local URL: http://127.0.0.1:7860

To create a public link, set 'share=True' in 'launch()'.
```

Imagen 1

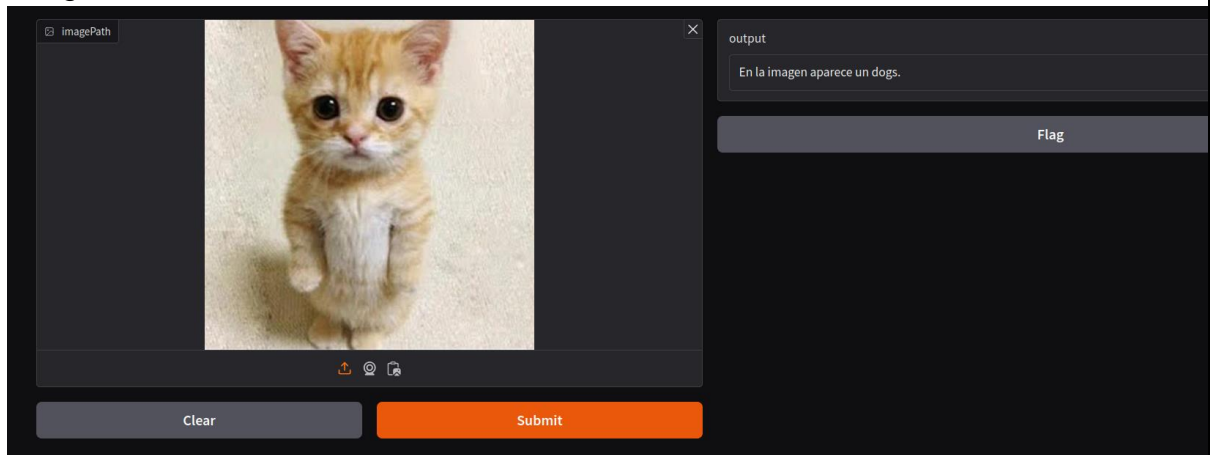


Imagen 2

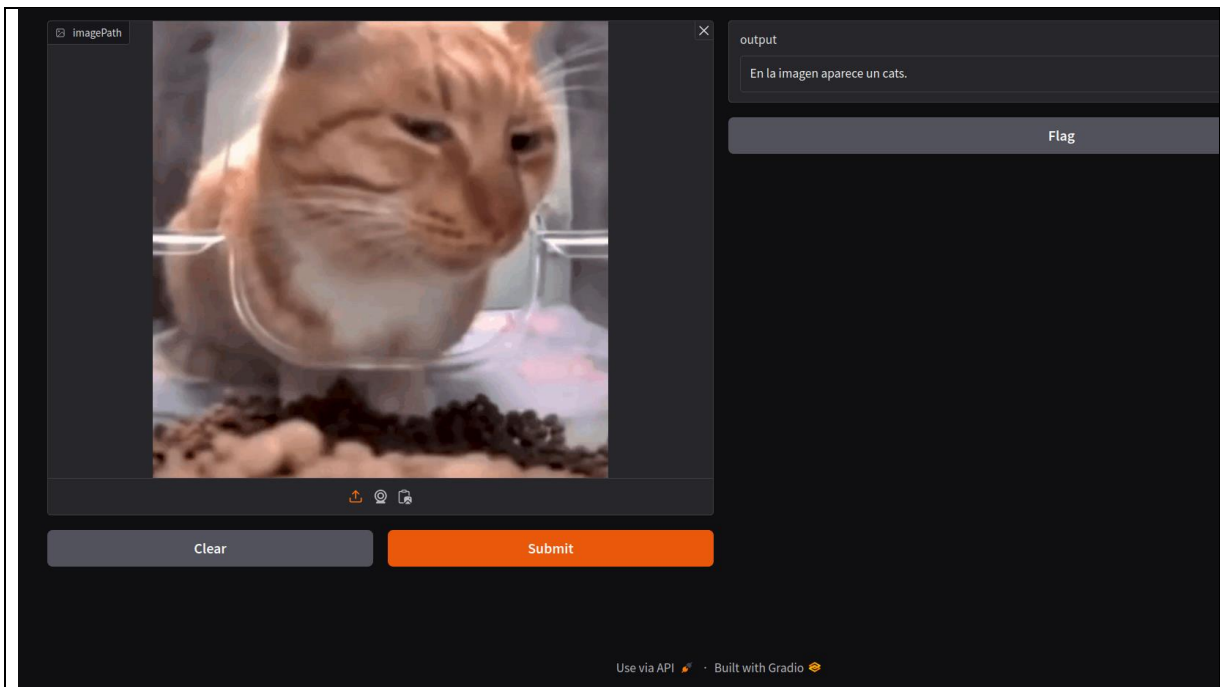


Imagen 3

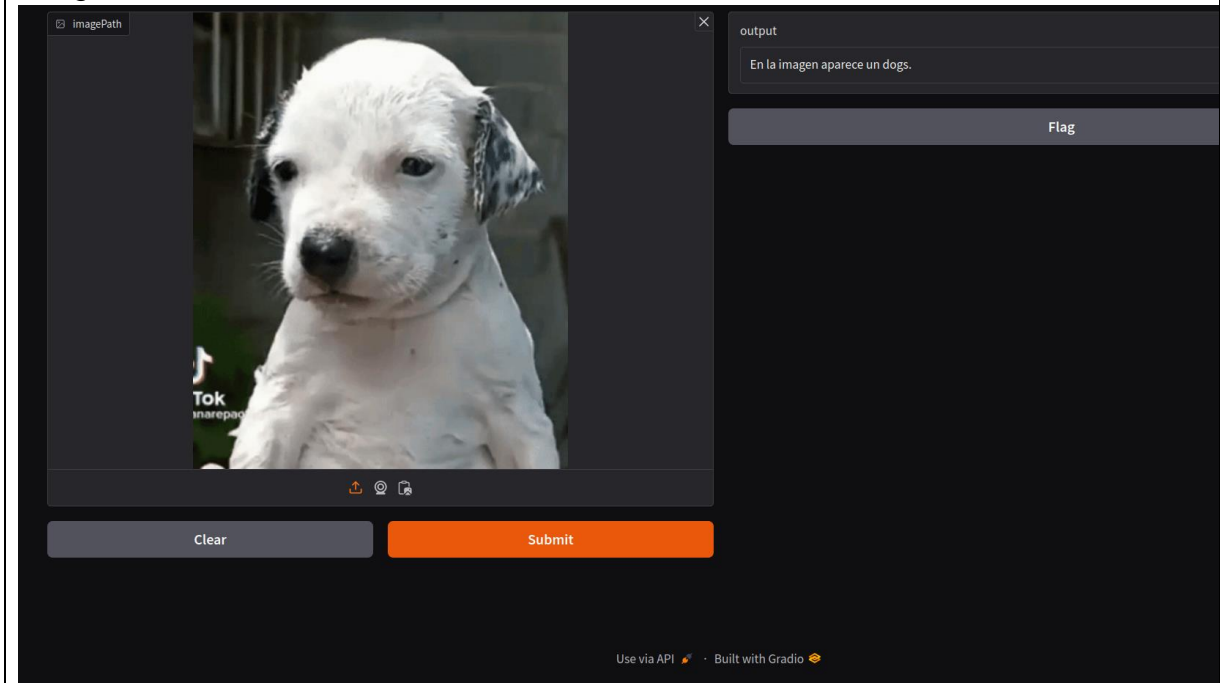


Imagen 4

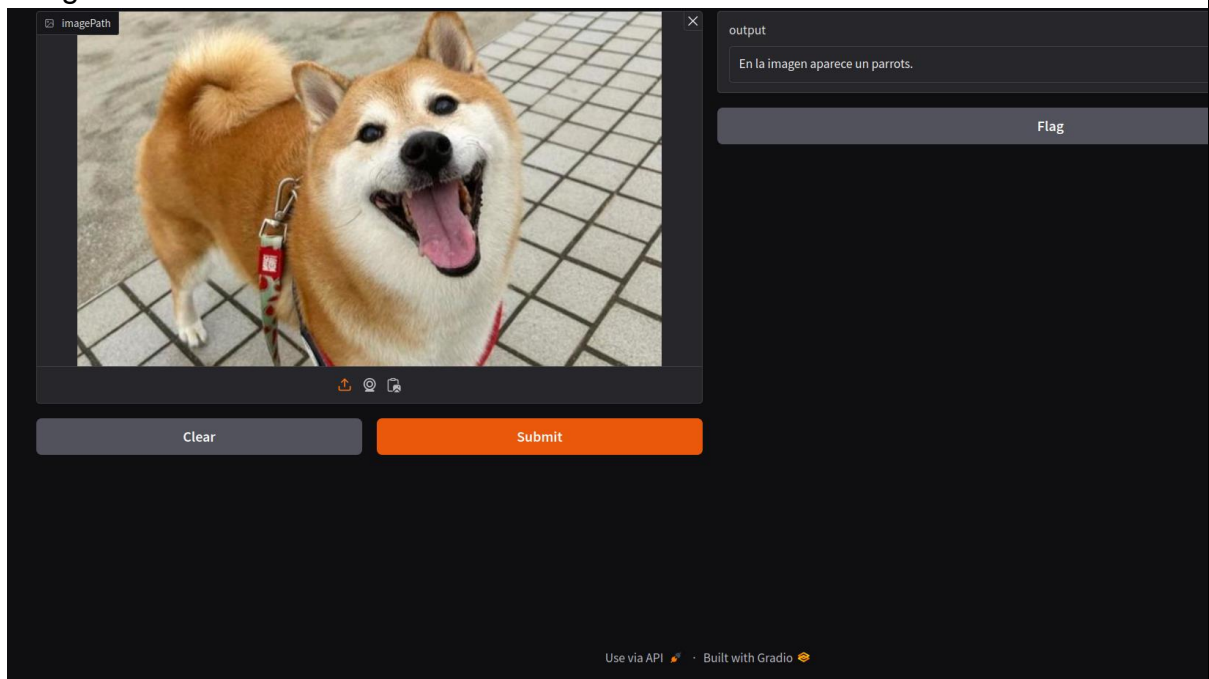
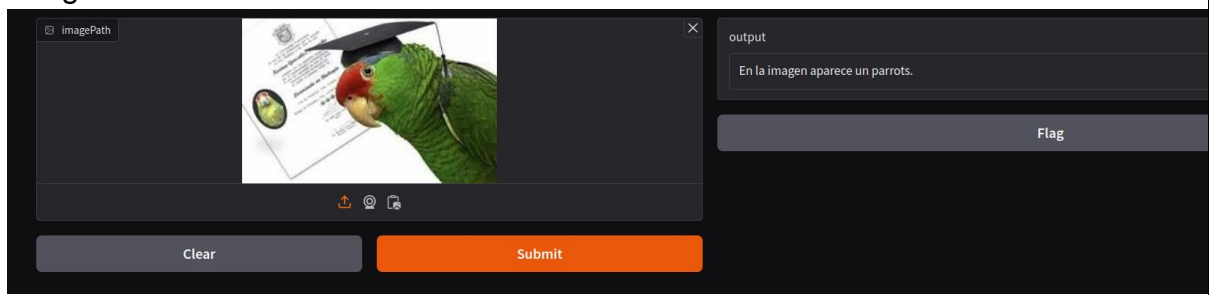


Imagen 5



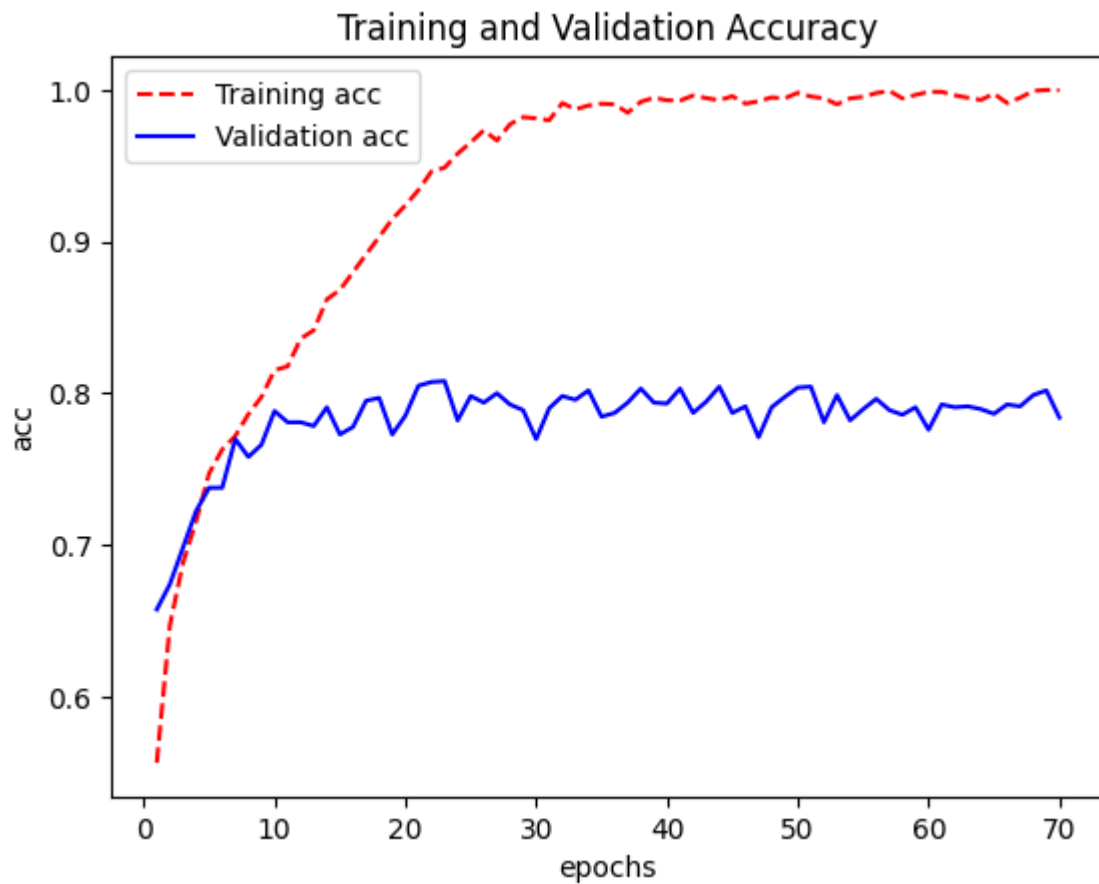
Modifique el código de entrenamiento, utilizando ahora un epochs=70.
¿Aproximadamente cuánto tiempo tarda en el ejecutar?*

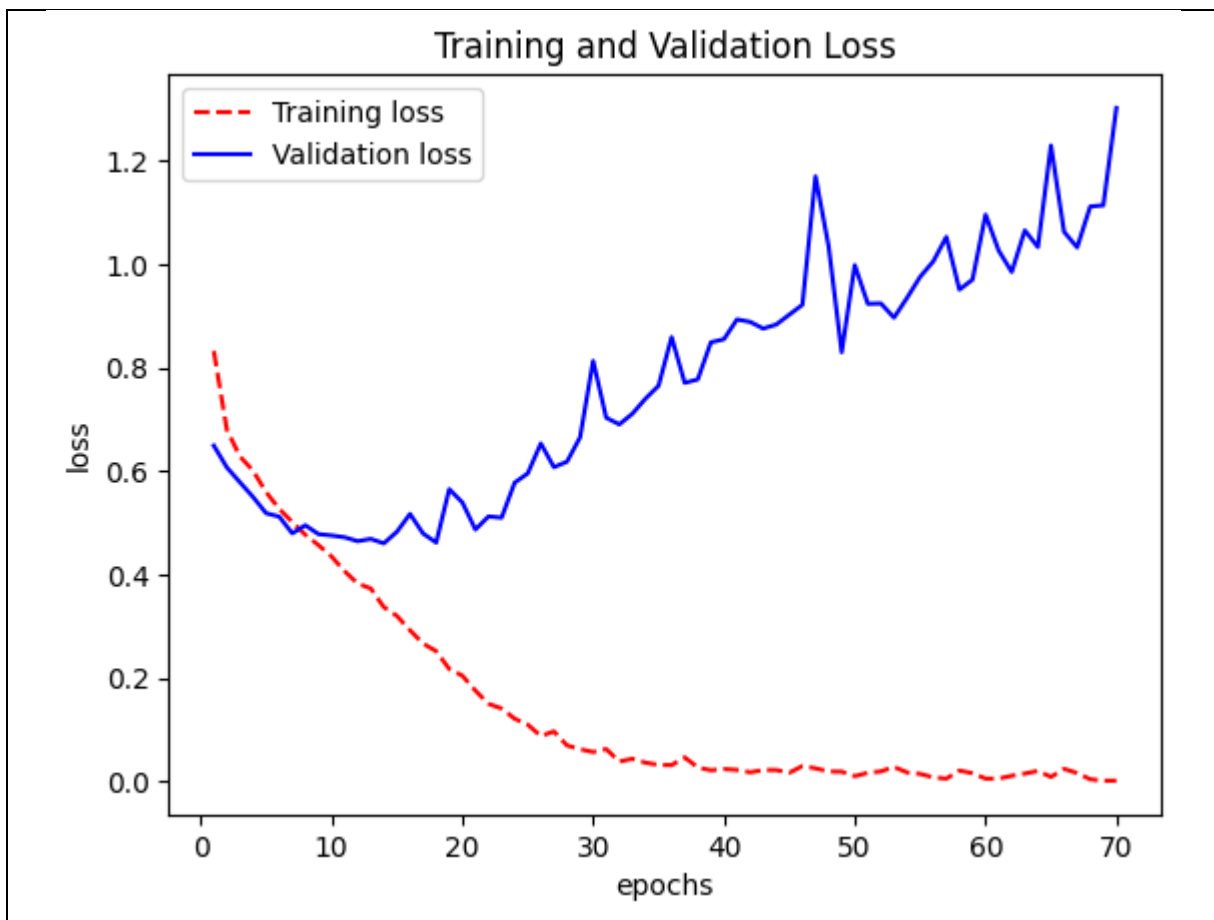
```
# In[14]:
history = model.fit(
    train_generator,
    steps_per_epoch = steps_per_epoch,
    #epochs = 180,
    epochs = 70,
    validation_data = validation_generator,
    validation_steps = validation_steps,
    verbose = 2
)

# Guardar el modelo entrenado
model.save('pets_categorical.h5')
```

[55] ✓ 47m 57.8s

... Epoch 57/70
486/486 - 39s - 79ms/step - acc: 0.9994 - loss: 0.0057 - val_acc: 0.7889 - val_loss: 1.0528
Epoch 58/70
486/486 - 39s - 79ms/step - acc: 0.9944 - loss: 0.0213 - val_acc: 0.7858 - val_loss: 0.9510





¿Cuál fue el accuracy y el loss obtenidos por este modelo para train, validation y test, luego de esta modificación? Puede presentar su respuesta completando la siguiente tabla.

SubGrupo	Accuracy	Loss
Train	0.9997942447662354	0.0018445615423843265
Validation	0.7839506268501282	1.3020528554916382
Test	0.7744	1.3216

Una vez entrenado el modelo utilizando 70 épocas, pruebe dicho modelo con el código de la Parte 2 del laboratorio, utilice 5 imágenes diferentes y que no estén presentes en el dataset y evalúe si el modelo es capaz de identificar correctamente el animal de la foto. Presente la evidencia de la ejecución de este código con los 5 elementos diferentes así como también la clase producida por el modelo, puede usar las mismas imágenes que utilizó para la Pregunta 4.

```
python src/lab8/CNN\ Test\ Categorical.py
~/git/portfolio_vision_artificial

(portafolio_vision_artificial)
# phoenix @ LAPTOP-VIKMPDAB in ~/git/portfolio_vision_artificial on git:main x portfolio_vision_artificial [20:00:00]
$ python src/lab8/CNN\ Test\ Categorical.py
2024-11-12 20:00:02.479678: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:477] Unable to register cuFFT factory: Attempting to register factory for plugin
ady been registered
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
E0000 00:00:1731459602.500274 364133 cuda_dnn.cc:8310] Unable to register cuDNN factory: Attempting to register factory for plugin cuDNN when one has already been
E0000 00:00:1731459602.506027 364133 cuda_blas.cc:1418] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has already been
2024-11-12 20:00:02.527138: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
W0000 00:00:1731459604.933904 364133 gpu_device.cc:2344] Cannot dlopen some GPU libraries. Please make sure the missing libraries mentioned above are installed properly
use GPU. Follow the guide at https://www.tensorflow.org/install/gpu for how to download and setup the required libraries for your platform.
Skipping registering GPU devices...
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. 'model.compile_metrics' will be empty until you train or evaluate the model.
1/1 [0. 1. 0.] 0s 83ms/step
1
dogs
which: no node in (/home/phoenix/git/portafolio_vision_artificial/bin:/opt/cuda/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/opt/cuda/bin:/opt/cuda/nsight_compute:
n:/var/lib/flatpak/exports/bin:/usr/lib/jvm/default/bin:/usr/bin/site_perl:/usr/bin/vendor_perl:/usr/bin/core_perl:/usr/lib/rustup/bin)
* Running on local URL: http://127.0.0.1:7860

To create a public link, set 'share=True' in 'launch()'.
```

Imagen 1

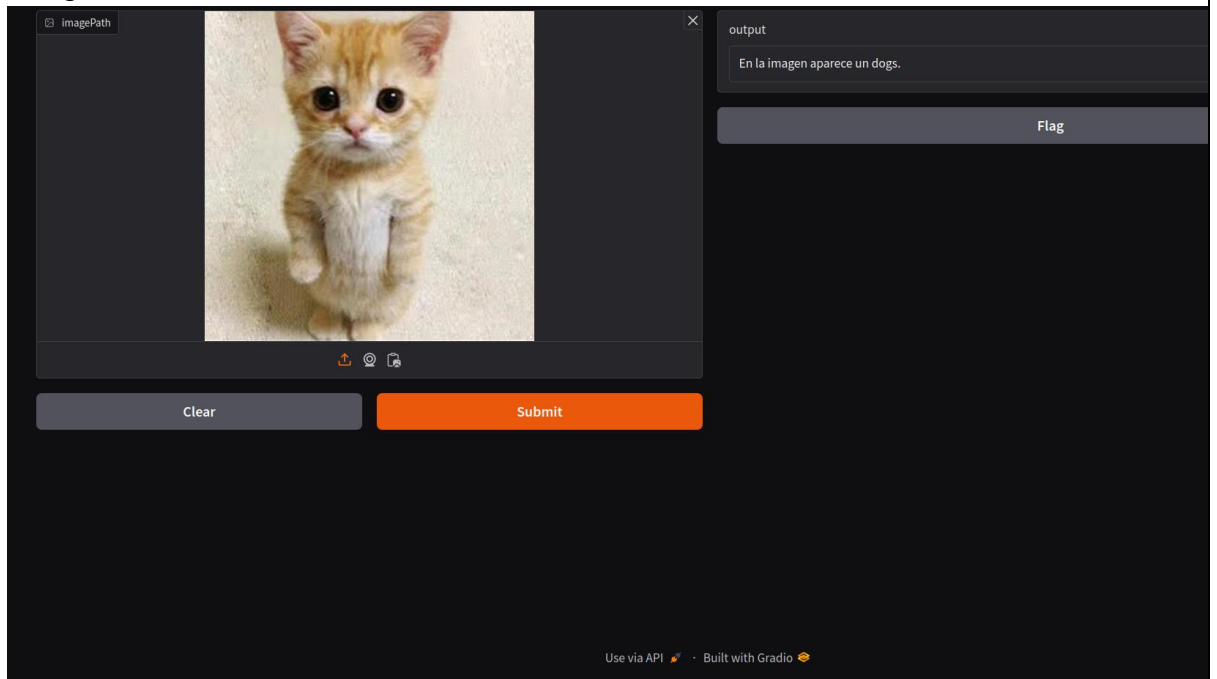



Imagen 2

imagePath



output

En la imagen aparece un dogs.

Flag


Clear

Submit

Use via API · Built with Gradio

Imagen 3

imagePath



output

En la imagen aparece un dogs.

Flag

Clear

Submit

Use via API · Built with Gradio

Imagen 4

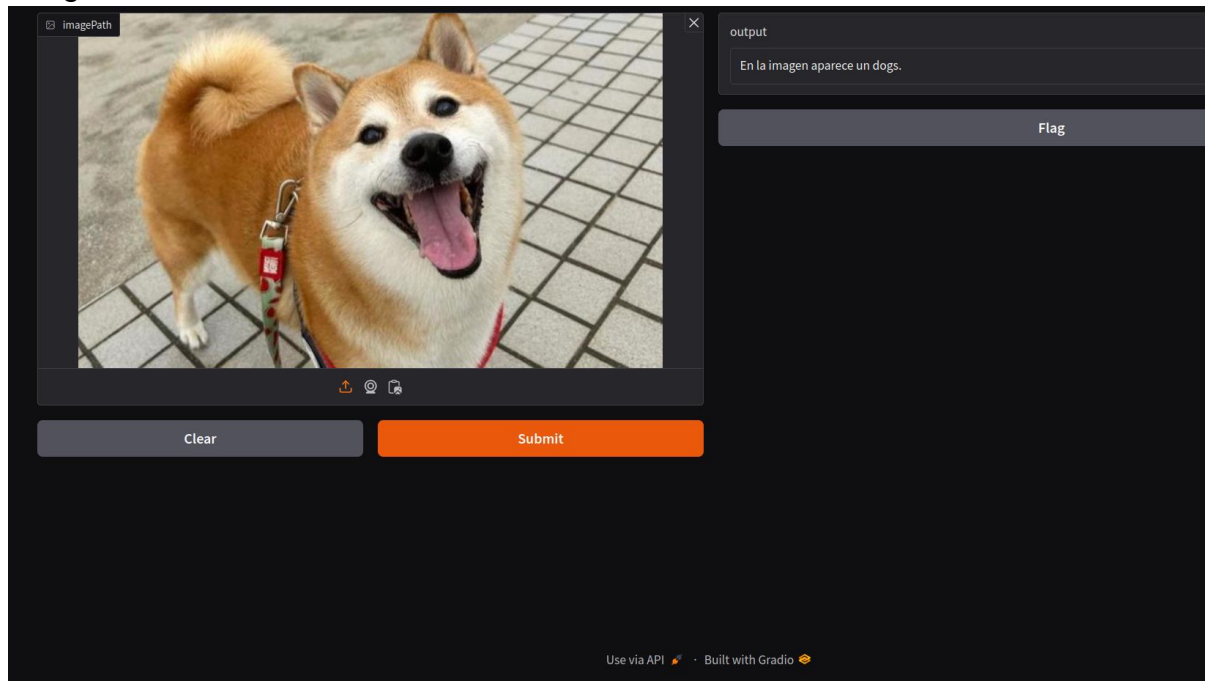
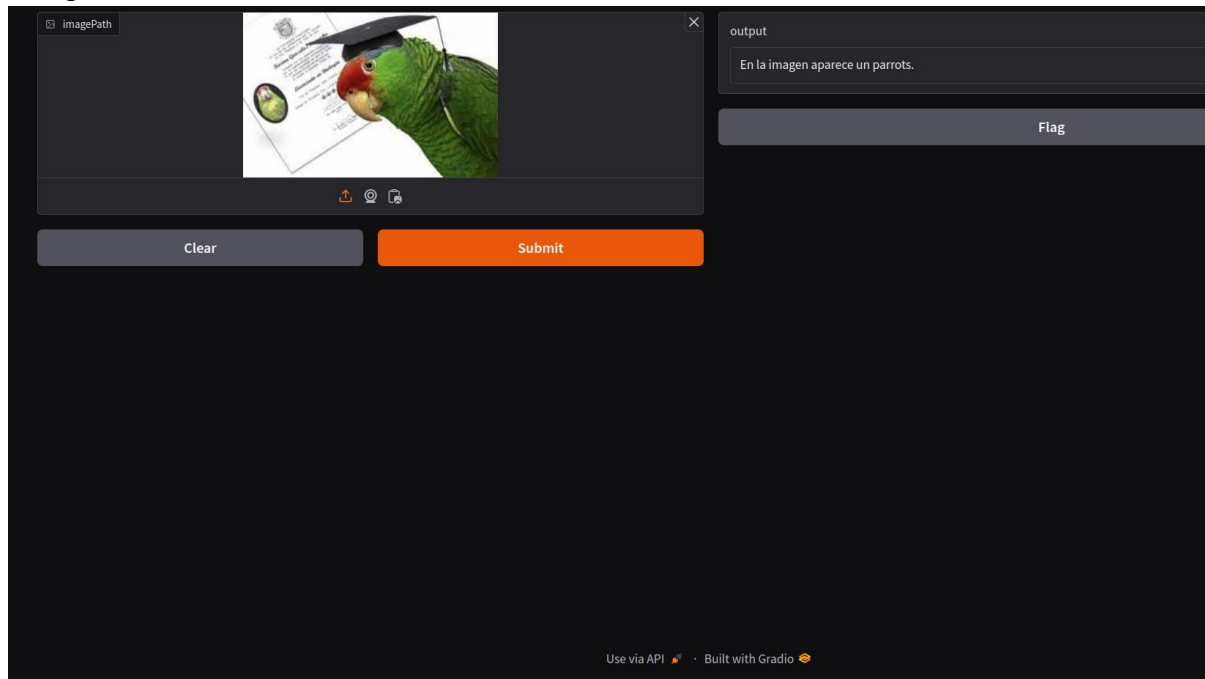


Imagen 5



Luego de la ejecución de ambos modelos. ¿Cuál obtuvo mejores resultados cuando se le mostraron sus imágenes?

Ambas están igual ya que en la primera me detecta un perro como un perico y en la segunda no me detecta ningún gato.

Captura de pantalla de ejecución con menos épocas

```
# In[14]:
history = model.fit(
    train_generator,
    steps_per_epoch = steps_per_epoch,
    epochs = 180,
    #epochs = 70,
    validation_data = validation_generator,
    validation_steps = validation_steps,
    verbose = 2
)

# Guardar el modelo entrenado
model.save('pets_categorical.h5')
```

[39] ✓ 104m 48.5s

... Epoch 1/180
486/486 - 36s - 73ms/step - acc: 0.5632 - loss: 0.8382 - val_acc: 0.6525 - val_loss: 0.6757
Epoch 2/180
486/486 - 33s - 69ms/step - acc: 0.6570 - loss: 0.6696 - val_acc: 0.6685 - val_loss: 0.5941

Captura de pantalla de ejecución con más épocas

```
# In[14]:
history = model.fit(
    train_generator,
    steps_per_epoch = steps_per_epoch,
    #epochs = 180,
    epochs = 70,
    validation_data = validation_generator,
    validation_steps = validation_steps,
    verbose = 2
)

# Guardar el modelo entrenado
model.save('pets_categorical.h5')
```

[55] ✓ 47m 57.8s

... Epoch 57/70
486/486 - 39s - 79ms/step - acc: 0.9994 - loss: 0.0057 - val_acc: 0.7889 - val_loss: 1.0528
Epoch 58/70
486/486 - 39s - 79ms/step - acc: 0.9944 - loss: 0.0213 - val_acc: 0.7858 - val_loss: 0.9510

Parte 4

capturas de pantalla de la ejecución de su clasificador en tiempo real. RESNET50

Label: lab coat, Prob: 14.61%



Label: coffee mug, Prob: 52.44%



Collage 1


Collage 2

Collage 3

Gradio Collage 1

Gradio Collage 2

Gradio Collage 3



Conclusiones y Comentarios de los Estudiantes

Siempre quise saber como entrenar una IA y en este laboratorio pude resolver esa duda, ahora sé por qué se necesita mucho poder computacional para hacerlo.

Bibliografía

- [1] A. Fernández Villán, Mastering OpenCV 4 with Python: a practical guide covering topics from image processing, augmented reality to deep learning with OpenCV 4 and Python 3.7. Mastering Open Source Computer Vision four with Python. Birmingham: Packt Publishing, 2019. [Online]. Available: <https://cds.cern.ch/record/2674578>
- [2] F. Chollet, Deep Learning with Python. Manning Publications Company, 2017. [Online]. Available: <https://books.google.com.pa/books?id=Yo3CAQAACAAJ>
- [3] J. Torres, Python deep learning: Introduccion practica con Keras y TensorFlow 2. Marcombo, 2020. [Online]. Available: <https://books.google.com.pa/books?id=ooqqzQEACAAJ>
- [4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," CoRR, vol. abs/1512.03385, 2015. [Online]. Available: <http://arxiv.org/abs/1512.03385>
- [5] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," arXiv preprint arXiv:1408.5093, 2014.
- [6] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li, "Imagenet large scale visual recognition challenge," CoRR, vol. abs/1409.0575, 2014. [Online]. Available: <http://arxiv.org/abs/1409.0575>
- [7] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," CoRR, vol. abs/1804.02767, 2018. [Online]. Available: <http://arxiv.org/abs/1804.02767>
- [8] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," CoRR, vol. abs/1405.0312, 2014. [Online]. Available: <http://arxiv.org/abs/1405.0312>