

Transfer learning para clasificación de razas

Introducción

Anteriormente se realizó un clasificador de perros y gatos, se entrenaron diversos modelos basados en la arquitectura VGG, sin embargo, llevar a cabo la mejora de éstos modelos suele ser una tarea muy exhaustiva, ya sea que se adopte cualquiera de los siguientes paradigmas

- Baby Panda: utilizar un solo modelo e ir optimizándolo hasta alcanzar el desempeño deseado.
- Caviar: crear una gran variedad de modelos, medir su desempeño y acotar los que han presentado mejor desempeño.

Cualquiera de estos dos paradigmas que se adopte (en especial la del Caviar), el costo computacional suele ser muy alto para modelos más complejos.

Aprendizaje transferido

El aprendizaje transferido es una técnica muy útil para llevar a cabo proyectos más complejos ya que disminuye mucho la carga del entrenamiento del modelo, y por lo tanto, el costo computacional. Ésta técnica consiste en utilizar un modelo preentrenado para ser utilizado, alterado o complementado para aplicaciones que suelen ser más complejas que la habituales.

TensorFlow Hub

TensorFlow Hub es un sitio que contiene una grna variedad de modelos que han sido preentrenados con diferentes bases de datos y propósitos.



Modelos

Busca modelos entrenados de la comunidad de TensorFlow en [TFHub.dev](#)

BERT

Utiliza BERT para tareas de PLN, incluidas la clasificación de texto y la respuesta a preguntas.

[Consulta el modelo](#)

Detección de objetos

Utiliza el modelo Faster R-CNN Inception ResNet V2 640x640 para detectar objetos en imágenes.

[Consulta el modelo](#)

Transferencia de estilo

Transfiere el estilo de una imagen a otra con el modelo de transferencia de estilo de imagen.

[Consulta el modelo](#)

Clasificador de alimentos en el dispositivo

Usa este modelo de TFLite para clasificar fotos de comida en un dispositivo móvil.

[Consulta el modelo](#)

Clasificador de Gatos y Perros



Este proyecto se enfoca en la clasificación de perros y gatos por su tipo de raza, el cual da seguimiento al proyecto anterior que consistía simplemente en saber si era gato o perro.

Ahora la siguiente Red Neural da el paso siguiente que es clasificar los perros y gatos por su tipo de raza, lo cual tiene aplicaciones inmediatas al trabajar con una interfaz digital en el ámbito de la veterinaria o comercio de mascotas.


Red Neuronal ResNet50

Para llevar a cabo la tarea antes mencionada se utilizará la red neuronal ResNet50, la cual ha sido preentrenada para clasificar diversos objetos.


TensorFlow > API > TensorFlow Core v2.9.1 > Python



¿Te resultó útil?  

tf.keras.applications.resnet50.ResNet50

 View source on GitHub

Instantiates the ResNet50 architecture.

 View aliases

```
tf.keras.applications.resnet50.ResNet50(  
    include_top=True,  
    weights='imagenet',  
    input_tensor=None,  
    input_shape=None,  
    pooling=None,  
    classes=1000,  
    **kwargs  
)
```

La precisión de este modelo puede resumirse en la siguiente imagen.

Available models

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
Xception	88	79.0%	94.5%	22.9M	81	109.4	8.1
VGG16	528	71.3%	90.1%	138.4M	16	69.5	4.2
VGG19	549	71.3%	90.0%	143.7M	19	84.8	4.4
ResNet50	98	74.9%	92.1%	25.6M	107	58.2	4.6
ResNet50V2	98	76.0%	93.0%	25.6M	103	45.6	4.4
ResNet101	171	76.4%	92.8%	44.7M	209	89.6	5.2
ResNet101V2	171	77.2%	93.8%	44.7M	205	72.7	5.4
ResNet152	232	76.6%	93.1%	60.4M	311	127.4	6.5
ResNet152V2	232	78.0%	94.2%	60.4M	307	107.5	6.6
InceptionV3	92	77.9%	93.7%	23.9M	189	42.2	6.9
InceptionResNetV2	215	80.3%	95.3%	55.9M	449	130.2	10.0
MobileNet	16	70.4%	89.5%	4.3M	55	22.6	3.4
MobileNetV2	14	71.3%	90.1%	3.5M	105	25.9	3.8
DenseNet121	33	75.0%	92.3%	8.1M	242	77.1	5.4
DenseNet169	57	76.2%	93.2%	14.3M	338	96.4	6.3
DenseNet201	80	77.3%	93.6%	20.2M	402	127.2	6.7
NASNetMobile	23	74.4%	91.9%	5.3M	389	27.0	6.7
NASNetLarge	343	82.5%	96.0%	88.9M	533	344.5	20.0
EfficientNetB0	29	77.1%	93.3%	5.3M	132	46.0	4.9
EfficientNetB1	31	79.1%	94.4%	7.9M	186	60.2	5.6
EfficientNetB2	36	80.1%	94.9%	9.2M	186	80.8	6.5

Importando bibliotecas a utilizar

```
In [ ]: from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input, decode_predictions  
from keras.preprocessing import image  
import numpy as np  
import matplotlib.pyplot as plt  
import random  
import pandas as pd
```

Creando el modelo ResNet50

Durante este proceso se cargan los pesos asociados a la red neuronal listo para clasificar.

```
In [ ]: model = ResNet50(weights='imagenet')
```

Función evaluar imágenes

La siguiente función lee una imagen y la clasifica.

```
In [ ]: def evaluate(img_fname):  
    """Realiza la clasificación de razas de gatos y perros mediante la red neuronal RedNet50.  
  
    Args:  
        img_fname (imagen): imagen en formato .jpg (de preferencia)  
  
    Returns:  
        lista (list) : Retorna una lista con el identificador, raza y la probabilidad asociada brindada por la red neural.  
    """  
    img = image.load_img(img_fname, target_size=(224, 224))  
    x = image.img_to_array(img)  
    x = np.expand_dims(x, axis=0)  
    x = preprocess_input(x)  
    preds = model.predict(x)  
    # print the probability and category name for the 5 categories  
    # with highest probability:  
    # print('Predicted:', decode_predictions(preds, top=5)[0])  
    plt.imshow(img)
```

```
lista = list(decode_predictions(preds)[0][0])

plt.title("Raza:" + lista[1])
plt.xticks([])
plt.yticks([])
return lista
```

Loop Clasificador de Imágenes

- Itera aleatoriamente 10 imágenes de nuestro set de prueba y las clasifica mediante la red neuronal.
- Retorna una lista con las propiedades asociadas a cada imagen (id, Raza, Probabilidad).

```
In [ ]: # Obtiene el índice aleatorio de 10 imágenes
rand_idx = [random.randint(0,12500) for i in range(10)]
# Define una lista vacía
df_pred = []
# Configura el canvas de la imagen
plt.figure(figsize=(15,7))
# Iterador for
for i in range(10):
    plt.subplot(2,5,i+1)
    # Agrega la lista retornada por la función evaluate()
    df_pred.append(evaluate("test\\" + str(rand_idx[i]) + ".jpg"))
```



Dataframe de razas y probabilidad

El siguiente dataframe muestra el resultado obtenido en el loop anterior con su probabilidad asociada.

```
In [ ]: df_pred = pd.DataFrame(df_pred, columns=["Id", "Raza", "Probabilidad"])
df_pred.set_index("Id")
```

Out[]:

	Raza	Probabilidad
Id		
n04553703	washbasin	0.206098
n02124075	Egyptian_cat	0.195711
n02110185	Siberian_husky	0.600096
n03000134	chainlink_fence	0.485013
n02099601	golden_retriever	0.931125
n02105056	groenendael	0.587192
n02088364	beagle	0.930223
n02089867	Walker_hound	0.460709
n02124075	Egyptian_cat	0.832501
n02123597	Siamese_cat	0.942830

Conclusión

El transfer learning es una herramienta bastante útil para diseñar redes con propósitos más complejos, además nos ahorra mucho el proceso de optimización y el costo computacional necesario para diseñar una red neural desde cero.

Durante este proceso se llevó a cabo una red para clasificar por raza, sin embargo, el sistema no es perfecto, pero es un gran paso en el proceso de perfeccionamiento de la red.