

Modelos pre-entrenados

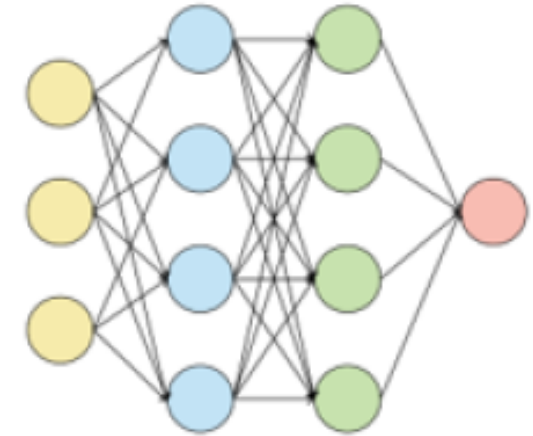
BASE CONVOLUCIONAL (extracción de características)



Convolutional Layer

Pooling and Flattening

CLASIFICADOR




Artificial Neural Network

ImageNet

- BBDD utilizada para reconocimiento de objetos en imágenes.
- Contiene 14 millones de imágenes etiquetadas con nombres de objetos de más de 20.000 categorías.
- **ILSVRC**
(Desafío de Reconocimiento Visual a Gran Escala de ImageNet)
 - ▣ ~1.2 millones de imágenes de entrenamiento.
 - ▣ 50.000 imágenes de validación 100.000 imágenes de prueba




Redes pre-entrenadas en Keras

- Las siguientes redes pre-entrenadas pueden ser consideradas como las capas convolucionales base.
- Se utilizan estas redes y se ajusta un clasificador (ANN):
 - ▣ VGG16 
 - ▣ Inception V3
 - ▣ Xception
 - ▣ ResNet50
 - ▣ MobileNet

Red neuronal convolucional con 16 capas propuesto por K. Simonyan y A. Zisserman de la Universidad de Oxford

El modelo se presentó al Desafío de Reconocimiento Visual a Gran Escala de ImageNet (ILSVRC) en 2014.

Redes pre-entrenadas en Keras

- Las siguientes redes pre-entrenadas pueden ser consideradas como las capas convolucionales base.
- Se utilizan estas redes y se ajusta un clasificador (ANN):
 - ▣ VGG16
 - ▣ Inception V3
 - ▣ Xception
 - ▣ ResNet50 
 - ▣ MobileNet

Red neuronal convolucional entrenada con el conjunto de datos de ImageNet que tiene 50 capas y que ganó el primer puesto en el ILSVRC en 2015.

VGG16 - Carga del modelo

```
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.applications.vgg16 import preprocess_input
```

```
# Descargue manualmente el modelo y colóquelo en el directorio
# C:\Users\nombre de usuario\.keras\models
# vgg16_weights_tf_dim_ordering_tf_kernels.h5
# vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
```

```
# Cargar modelo e imprimir
```

```
model = VGG16()
```

```
print(model.summary())
```



```
predictions (Dense)              (None, 1000)
=====
Total params: 138,357,544
```

Cargando la imagen a reconocer

Cargar una imagen de prueba

```
image = load_img("tigre.jpg", target_size=(224, 224))
```

Tamaño
224 x 224

Cargando la imagen a reconocer

Cargar una imagen de prueba

```
image = load_img("tigre.jpg", target_size=(224, 224))
```

Convertir a matriz

```
image = img_to_array(image)
```



Tamaño
224 x 224 x 3

Cargando la imagen a reconocer

Cargar una imagen de prueba

```
image = load_img("tigre.jpg", target_size=(224, 224))
```

Convertir a matriz

```
image = img_to_array(image)
```

Reformar en 4D



```
image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
```

Tamaño
1 x 224 x 224 x 3

La imagen a reconocer

Cargar una imagen de prueba

```
image = load_img("tigre.jpg", target_size=(224, 224))
```

Convertir a matriz

```
image = img_to_array(image)
```

Reformar en 4D

```
image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
```

Imagen de preproceso

```
image = preprocess_input(image) 
```

Las imágenes se convierten de RGB a BGR y, a continuación, cada canal de color se centra en cero con respecto al conjunto de datos de ImageNet, sin escalar.

Resultado de la predicción

Predicción de la red

```
predict_result = model.predict(image)
```



Resultado de la capa de salida
Tamaño: 1 x 1000

Resultado de la predicción

```
# Predicción de la red
```

```
predict_result = model.predict(image)
```

```
# Resultados de predicción de análisis
```

```
label = decode_predictions(predict_result) ←
```

Lista con los 5 mejores resultados

Índi ▲	Tipo	Tamaño	Valor
0	tuple	3	('n02129604', 'tiger', 0.92024356)
1	tuple	3	('n02123159', 'tiger_cat', 0.07646653)
2	tuple	3	('n02128925', 'jaguar', 0.0027824175)
3	tuple	3	('n02127052', 'lynx', 0.00025741145)
4	tuple	3	('n02128385', 'leopard', 0.0001807782)

Resultado de la predicción

```
# Predicción de la red
```

```
predict_result = model.predict(image)
```

```
# Resultados de predicción de análisis
```

```
label = decode_predictions(predict_result)
```

```
# Imprima las tres categorías con mayor probabilidad
```

```
for idx in range(0, 3):
```

```
    print ("Categoría:% s Probabilidad:% 0.4f"% (label[0][idx][1],  
                                                  label[0][idx][2]))
```

```
Categoría:tiger Probabilidad: 0.9202  
Categoría:tiger_cat Probabilidad: 0.0765  
Categoría:jaguar Probabilidad: 0.0028
```



Ajuste fino (fine-tuning) de modelos pre-entrenados

- Se busca reusar la base convolucional y mejorar la respuesta del clasificador en una tarea específica.
- Pasos para efectuar el **ajuste fino**
 - ▣ Añadir un clasificador (RNA) sobre un sistema preentrenado.
 - ▣ Fijar la base convolucional y entrenar la red.
 - ▣ Entrenar conjuntamente el clasificador añadido y la base convolucional.

Consultar Capítulo 8 del libro “The Deep Learning with Keras Workshop. An Interactive Approach to Understanding Deep Learning with Keras (2020)”

VGG16_cars_flowers.ipynb