Modelos pre-entrenados, Transferencia de Aprendizaje y Finetuning

Modelos Preentrenados

- Entrenar un modelo requiere
  - Tiempo/uso de cpu
  - Optimización de hiperparámetros
  - Preparación de datos
- Usar un modelo preentrenado
  - + Fácil y rápido
  - - Arquitectura fija
  - - Dominio fijo

```
from keras.applications.mobilenet import
MobileNet,preprocess_input
model = MobileNet(weights='imagenet')
from keras.preprocessing import image
img = image.load_img(image_path, target_size=(224, 224))
img = image.img_to_array(img)
img = np.expand_dims(img, axis=0)
img =.preprocess_input(img)
class=model.predict_classes(img)
```

Transferencia de Aprendizaje

- Motivación
  - Tengo un modelo
    - para problema **X**
  - Quiero resolver
    - problema **Y**
- Modelo de Gatos y Perros
  - No sirve para Mariposas vs Abejas

conv1

conv2

conv3

fc1

fc2

gato

perro

conv1

conv2

conv3

fc1

fc2

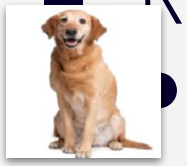
:´(

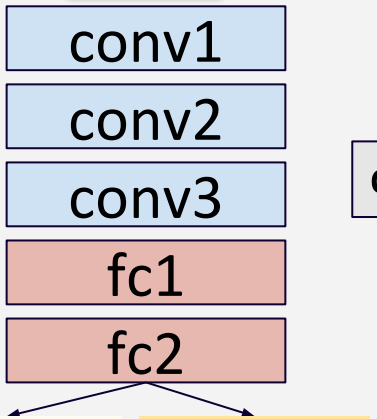
;O

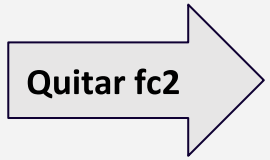
Transferencia de Aprendizaje

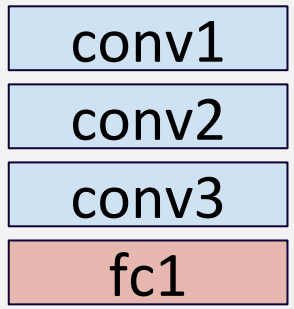
- ¿Empiezo desde 0? No
  - **Reusar Filtros Convolucionales**
    - Relativamente independientes del dominio
  - Reemplazar última capa Dense o GlobalAveragePooling
    - Reentrenar red
      - **Congelar** capas anteriores

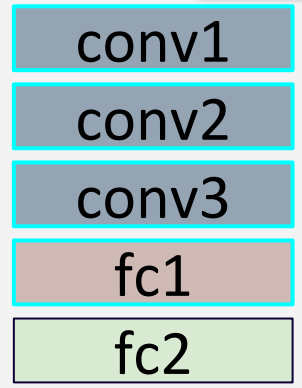
- **Congelar**
  - No entrenar los pesos

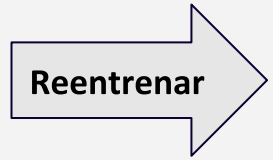


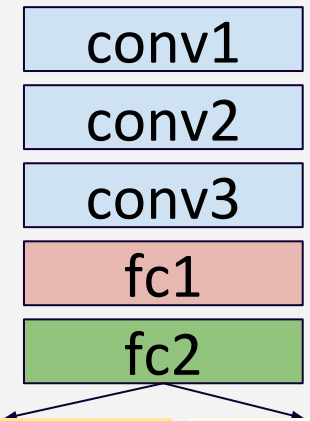












Transferencia de Aprendizaje (kaggle,notebook)

```
base_model=MobileNet(input_shape=(32,32,3),weights='imagenet',include_top=Fals
e)
for layer in base_model.layers:
   layer.trainable=False # capas “congeladas” no se entrenan

# Utilizar salida del modelo como entrada a capa Dense de 128
output = Dense(128,activation='relu')(base_model.output)
# Nueva capa de salida
output = Dense(classes,activation='softmax')(output)
model=Model(inputs=base_model.input,outputs=output)

#Entrenar con nuevos datos
model.compile(optimizer='Adam',loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x,y,epochs=20)
```

Finetuning

- Reentrenar todo el modelo
  - Más tiempo de entrenamiento
  - Resultados ligeramente mejores (1% a 5%)
- Código: Igual que el anterior
  - Pero sin congelar pesos