Universidad Autónoma de Querétaro

Facultad de Ingeniería

División de Investigación y Posgrado

Reporte 10

Regularización de sobreajuste

Maestría en Ciencias en Inteligencia Artificial

Optativa de especialidad II - Deep Learning

Aldo Cervantes Marquez

Expediente: 262775

Profesor: Dr. Sebastián Salazar Colores

Santiago de Querétaro, Querétaro, México

Semestre 2022-2

22 de Noviembre de 2022

Índice

1. Introducción

En la presente práctica, se utilizaran técnicas de sobreajuste de entrenamiento para evitar que el modelo se memorice los datos de entrenamiento y al momento de validar no se obtenga un valor tan bajo. Se utilizará la red LeNet5 como modelo para disminuir el sobreajuste y por lo tanto observar si una red tan sencilla es capaz de poder aumentar su exactitud del modelo. Todo esto con la base de datos de perros y gatos.

2. Marco Teórico

2.1. Base de datos

La base de datos consta de imágenes que contienen perros y gatos, de diferentes razas en diferentes posiciones, paisajes, etc. Por lo que todas las imagenes contienen una dimensión diferente (véase Figura 1).

Figura 1: Imágenes de la base de datos.

2.2. Sobreajuste

Es un efecto al sobreentrenar un algoritmo de aprendizaje [1] con los datos de entrenamiento. Teniendo el problema de que no puede generalizar el problema y cuando lleguen nuevos datos para validación y test, habrá malos resultados (véase Figura 2).

Otro indicador se encuentra en las perdidas del modelo al momento de entrenar y validar, pues al ir ajustándose el modelo a los datos de entrenamiento, puede ir incrementando la perdida de la validación (véase Figura 3) [2].

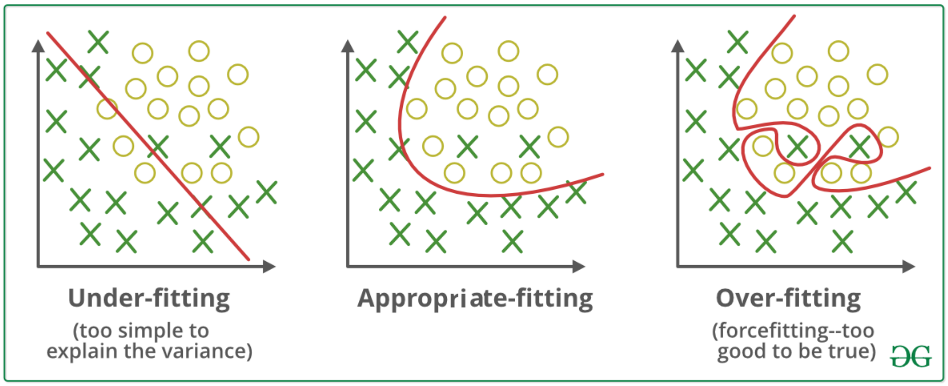


Figura 2: Representación del sobreajuste.

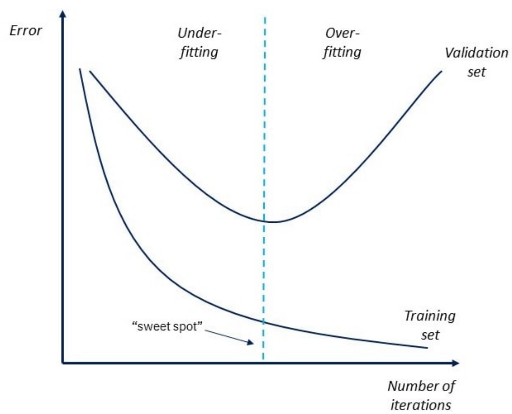


Figura 3: Representación de la perdida en función de las épocas del modelo con sobreajuste.

Por lo que se requieren técnicas que eviten estos comportamientos como se muestran a continuación:

2.2.1. Regularizadores L1 y L2

Esta técnica de regularización consiste en minimizar la complejidad y se obtienen modelos más simples que tienden a generalizar mejor [3].

Partiendo de la función de coste J como un error cuadrático medio MSE tenemos que:

$$J = MSE \qquad J = MSE + (\alpha C) \tag{1}$$

Donde:

- α es un hiperparámetro que indica la importancia de la regularización.
- C es la medida de complejidad del modelo.

El regularizador L_1 Lasso la complejidad C se mide como la media del valor absoluto de los coeficientes del modelo. Esto se puede aplicar a regresiones lineales, polinómicas, regresión logística, redes neuronales, máquinas de vectores de soporte, etc. Se define como:

$$L_1 = C = \frac{1}{N}\sum_{j=1}^{N}|\omega_i| \tag{2}$$

Lasso nos va a servir de ayuda cuando sospechemos que varios de los atributos de entrada (features) sean irrelevantes. Al usar Lasso, estamos fomentando que la solución sea poco densa. Es decir, favorecemos que algunos de los coeficientes acaben valiendo 0. Esto puede ser útil para descubrir cuáles de los atributos de entrada son relevantes y, en general, para obtener un modelo que generalice mejor. Lasso nos puede ayudar, en este sentido, a hacer la selección de atributos de entrada. Lasso funciona mejor cuando los atributos no están muy correlados entre ellos.

Por otro lado se tiene la regularización L_2 Ridge medida en la que los coeficientes son más pequeños. Minimizando el efecto de la correlación entre los atributos de los datos. Se define como:

$$L_2 = C = \frac{1}{2N}\sum_{j=1}^{N}\omega_i^2 \tag{3}$$

Finalmente se tiene la regularización ElasticNet L_1L_2. Combinando las dos caracteristicas con otro hiperparámetro r que permite indicar la importancia de cada uno de los regularizadores, esta dada por:

$$L_1L_2 = C = rL_1 + (1-r)L_2 \tag{4}$$

Teniendo como principal característica cuando se tienen un gran numero de atributos. Teniendo algunos irrelevantes y otros si tienen correlación.

2.2.2. Dropout

Consiste en remover de manera aleatoria y temporalmente unidades de neuronas de las capas ocultas de la red en una probabilidad dada p. Al tener una p baja $p(< 0.4)$ el error del modelo aumentará por lo que requerirá de más épocas para poder aprender y si es alto, se observará el sobreajuste ($overfitting$).

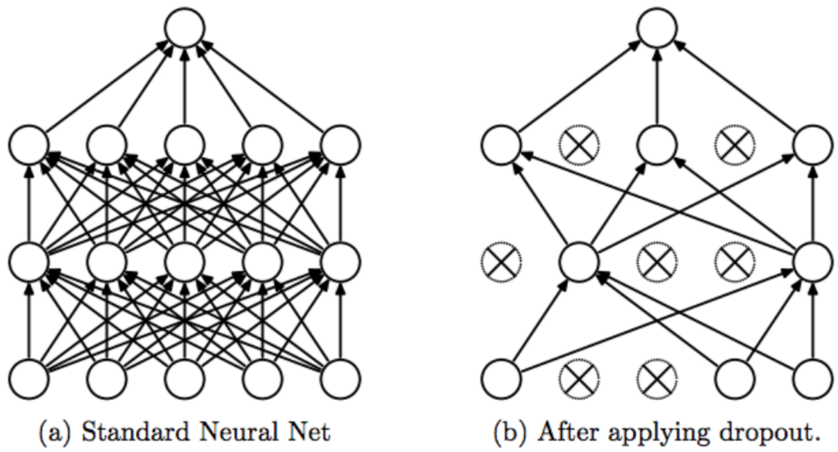


Figura 4: Representación de la técnica de Dropout.

2.2.3. BatchNormalization

Consiste en la normalización de lotes de información (batch) y básicamente lo que hace es normalizar las entradas y reescalar a la salida de cada capa. Existiendo los siguientes, tomando en cuenta el batch como $B = [x_1...m]$ y los hiperparámetros γ, β [4].

Mini-batch:

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{5}$$

Mini-Batch con varianza:

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \tag{6}$$

Normalización:

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{7}$$

Escalación y cambio:

$$y_i = \gamma\hat{x}_i + \beta \equiv BN_{\gamma,\beta}(x_i) \tag{8}$$

2.2.4. Data Augmentation

Consiste en el incremento de los datos para poder obtener mas datos de entrenamiento, existen muchas técnicas que *keras* nos incluye como son [5, 6]:

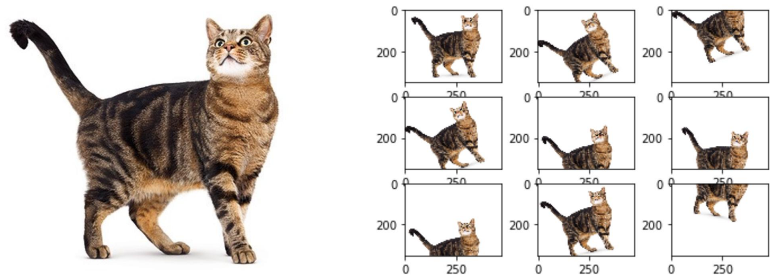


Figura 5: Representación de la técnica de Dropout.

- class CategoryEncoding: A preprocessing layer which encodes integer features.
- class CenterCrop: A preprocessing layer which crops images.
- class Discretization: A preprocessing layer which buckets continuous features by ranges.
- class HashedCrossing: A preprocessing layer which crosses features using the "hashing trick".
- class Hashing: A preprocessing layer which hashes and bins categorical features.
- class IntegerLookup: A preprocessing layer which maps integer features to contiguous ranges.
- class Normalization: A preprocessing layer which normalizes continuous features.
- class PreprocessingLayer: Base class for Preprocessing Layers.
- class RandomContrast: A preprocessing layer which randomly adjusts contrast during training.
- class RandomCrop: A preprocessing layer which randomly crops images during training.
- class RandomFlip: A preprocessing layer which randomly flips images during training.
- class RandomHeight: A preprocessing layer which randomly varies image height during training.
- class RandomRotation: A preprocessing layer which randomly rotates images during training.
- class RandomTranslation: A preprocessing layer which randomly translates images during training.
- class RandomWidth: A preprocessing layer which randomly varies image width during training.
- class RandomZoom: A preprocessing layer which randomly zooms images during training.
- class Rescaling: A preprocessing layer which rescales input values to a new range.
- class Resizing: A preprocessing layer which resizes images.
- class StringLookup: A preprocessing layer which maps string features to integer indices.
- class TextVectorization: A preprocessing layer which maps text features to integer sequences.

2.3. Convoluciones

Las convoluciones constan de kernels que permiten modificar la imagen píxel a píxel, generalmente dichos kernels representan una matriz de valores, los cuales interactúan con la cantidad de datos en igual forma para realizar la convolución (véase Figura 6) [7, 8].

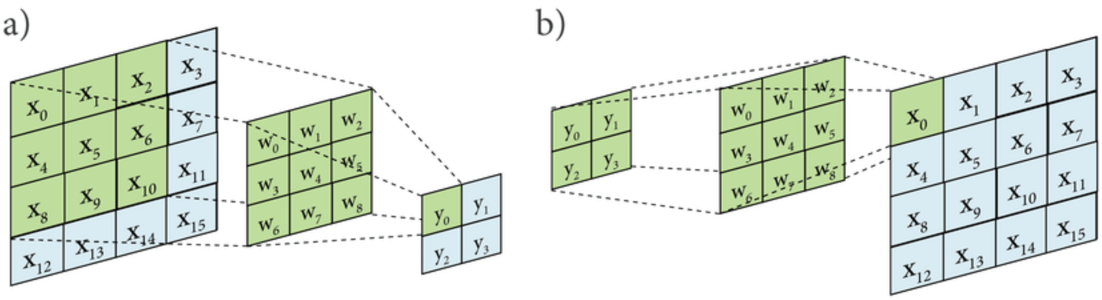


Figura 6: Paleta cúbica de colores RGB.

2.4. Pooling

Como sabemos, las convoluciones permiten resaltar partes de la imagen que nos podrían interesar, siempre conservando prácticamente en su totalidad la imagen. Por otro lado, la capa de pooling nos asegura que los patrones detectados en la capa convolucional se mantengan [9].
Además de que las capas de pooling no requieren de ningún parámetro de aprendizaje. Existen principalmente 3 tipos de Pooling: el maxpool, el minpool y el averagepool. El primero indica que el minímo de los valores de la sección de la matriz de pooling sera el seleccionado como resultado, mismo caso para el máximo y para el promedio del conjunto de valores (véase Figura 7).

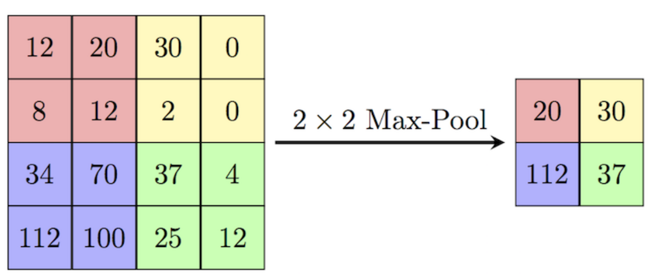


Figura 7: Ejemplo de Maxpool.

2.5. Red Convolucional

Consiste en en algoritmo que al igual que una red neuronal, asigna y actualiza pesos a los valores de la función y por lo tanto se optimizan los valores de los kernels (convoluciones) para poder reconocer patrones de siluetas, curvas, lineas, rostros, etc [10].

2.5.1. Red LeNet-5

Consiste en un arreglo de 7 capas propuesta por Yann LeCun en 1998, tiene como principal objetivo, resolver problemas con imágenes. Dichas capas se van alternando entre convoluciones y Pooling, en donde las capas de convolución son resultado de la multipliación de un kernel por la imagen, por lo que, dichos valores de la matriz del kernel, serán definidas como pesos de la red, el punto es ir disminuyendo el tamaño de la imagen y aplanarla para que pueda entrar a una red neuronal (véase Figura 8).

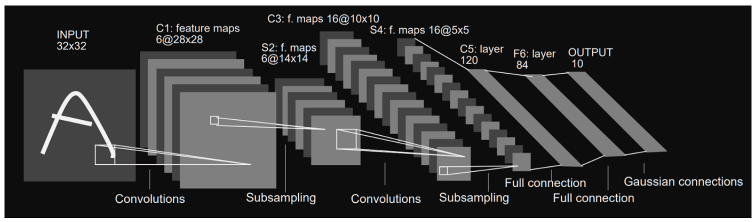


Figura 8: Estructura de red convolucional LeNet5.

3. Justificación

El uso de regularizadores es muy útil para poder incrementar el porcentaje de exactitud de nuestras pruebas y tener más fiabilidad de las mismas, sin la necesidad de tener que depender del resultado del entrenamiento. Además de tener la posibilidad de poder almacenar los resultlados del modelo y sus pesos

4. Resultados

Se observó que el uso de los regularizadores disminuye abruptamente el porcentaje de exactitud del entrenamiento, evitando el sobreajuste y por lo tanto, mostrando de una manera más realista el desempeño de la red. Se observó que algunos regularizadores usados de manera individual funcionan de una mejor manera que todos juntos, así como también su aplicación en sus capas.

Tabla 1: Resultados de LeNet5.

Prueba	Características	Optimizador	Épocas	Batch	% de entrenamiento	% de validación	% de test
1	LeNet5 básica	SGD	80	64	100	62.25	59.625
2	LeNet5 con L_1L_2 en las capas *flatten*	SGD	120	64	100	59.38	58.75
3	LeNet5 con Dropout en $c_4(0.3), c_5(0.5), c_6(0.6)$	SGD	120	64	62	52	52.875
4	LeNet5 con BatchNormalization	SGD	80	64	100	59.38	59.625
5	LeNet5 con Data Augmentation RandomCrop	SGD	80	64	98.04	59.25	58.75
6	LeNet5 con: • con L_1L_2 en las capas *flatten* • *Dropout* en todas las capas • BatchNormalization en todas las capas • RandomCrop • Reducción de factor de aprendizaje • EarlyStopper	Adam	138	32	50	50	50

5. Conclusiones

1. El uso de la red LeNet5 ya se puede considerar como un poco obsoleta para su uso en imagenes complejas y debido a su redimensionamiento a una imagen de baja resolución como 32x32x1.

2. El uso de regularizadores permite disminuir el porcentaje de exactitud del entrenamiento y por lo tanto, también hacen dar cuenta que será necesaria una red más compleja.

3. El uso del aumento de imagenes genera datos sinteticos de una manera fiable y útil para poder tener más datos de entrenamiento en casos en los que se cuenten con pocos datos.

4. La red LeNet5 debe ser utilizada para únicamente imagenes en blanco y negro (escala de grises), puesto que el uso de un solo filtro pudo haber afectado los resultados del desempeño de la red.

5. El uso de redes más complejas pueden ir mejorando el desempeño del modelo, sin embargo, es mejor comenzar con un modelo sencillo como este y continuar abstrayendo el problema.

6. Tambien es posible observar el sobreajuste en las funciones de perdida cuando se cruzan y toman rumbos distintos, puesto que si decrece la perdida del entrenamiento, tambien deberia disminuir tambien la perdida de la validación, no incrementar.

Referencias

[1] A. Rubiales, "¿qué es underfitting y overfitting?— medium." https://rubialesalberto.medium.com/qu%C3%A9-es-underfitting-y-overfitting-c73d51ffd3f9. (Accessed on 10/22/2022).

[2] "What is overfitting? — ibm." https://www.ibm.com/cloud/learn/overfitting. (Accessed on 11/22/2022).

[3] "Regularización lasso l1, ridge l2 y elasticnet - iartificial.net." https://www.iartificial.net/regularizacion-lasso-l1-ridge-l2-y-elasticnet/. (Accessed on 11/22/2022).

[4] "Implementing batch normalization in python — by tracy chang — towards data science." https://towardsdatascience.com/implementing-batch-normalization-in-python-a044b0369567. (Accessed on 11/22/2022).

[5] "Image augmentation for deep learning with keras - machinelearningmastery.com." https://machinelearningmastery.com/image-augmentation-deep-learning-keras/. (Accessed on 11/22/2022).

[6] "Module: tf.keras.layers.experimental.preprocessing — tensorflow v2.11.0." https://www.tensorflow.org/api_docs/python/tf/keras/layers/experimental/preprocessing. (Accessed on 11/22/2022).

[7] "2d convolution using python & numpy — by samrat sahoo — analytics vidhya — medium." https://medium.com/analytics-vidhya/2d-convolution-using-python-numpy-43442ff5f381. (Accessed on 09/17/2022).

[8] "5.2. imágenes rgb — introducción a la programación." https://cupi2-ip.github.io/IPBook/nivel4/seccion4-4.html. (Accessed on 09/17/2022).

[9] "Cómo crear red convolucional en keras - ander fernández." https://anderfernandez.com/blog/que-es-una-red-neuronal-convolucional-y-como-crearlaen-keras/. (Accessed on 09/26/2022).

[10] "Intro a las redes neuronales convolucionales — by bootcamp ai — medium." https://bootcampai.medium.com/redes-neuronales-convolucionales-5e0ce960caf8. (Accessed on 09/26/2022).

P10 eliminación de sobre ajuste

November 21, 2022

1 Eliminación de sobreajuste

Se utilizará la base de datos de perros y gatos.

[2]:
```
## Primeramente se descarga la base de datos que será la de perros y gatos
import cv2
import numpy as np
import os
import zipfile
from matplotlib import image

files=zipfile.ZipFile('cats_and_dogs_small.zip','r')
files.extractall('')

x_dog=[]
x_cat=[]
```

[3]:
```
from PIL import Image
x_size=32
y_size=32

for name in files.namelist():
    if '/dogs/' in name and '.jpg' in name:
        a=cv2.imread(name)
        a=cv2.resize(a,(x_size,y_size)) # Dimensión de la imagen
        img = cv2.cvtColor(a, cv2.COLOR_BGR2RGB)
        #img2=img.resize(200,200) # Mobilenet (224,224,3)
        x_dog.append(img)

    elif '/cats/' in name and '.jpg' in name:
        a=cv2.imread(name)
        a=cv2.resize(a,(x_size,y_size)) # Dimensión de la imagen
        img = cv2.cvtColor(a, cv2.COLOR_BGR2RGB)
        x_cat.append(img)
print(len(x_dog),len(x_cat))
x_dog=np.stack(x_dog,axis=0)
x_cat=np.stack(x_cat,axis=0)
```

```
2000 2000
```

1.1 Normalización y One Hot

[4]:
```
print(type(x_dog),x_dog.shape)
print(type(x_cat),x_cat.shape)
print('Valores minimos y maximos sin normalizar')
print(x_dog.min(),x_dog.max())
print(x_cat.min(),x_cat.max())
x_dog=x_dog.astype('float32')
x,y,z,w=x_dog.shape
y_dog=np.zeros((x,1),dtype=int)
x_cat=x_cat.astype('float32')
x,y,z,w=x_cat.shape
y_cat=np.ones((x,1),dtype=int)
x_dog/=255#(x_dog/127.5)-1#x_dog/=255
x_cat/=255#(x_cat/127.5)-1#x_cat/=255
## Conjunto combinado de perros y gatos
x_comb=np.vstack((x_dog,x_cat))
y_comb=np.vstack((y_dog,y_cat))
print(x_comb.ndim,x_comb.shape)
print('Valores minimos y maximos normalizados')
print(x_comb.min(),x_comb.max())
#print(y_dog)


### ONE HOT
from keras.utils import to_categorical
y_dog_oh=to_categorical(y_dog,y_dog.max()+2)
y_cat_oh=to_categorical(y_cat,y_cat.max()+1)
print(y_comb[3455])
y_comb_oh=to_categorical(y_comb,y_comb.max()+1)
print(y_comb_oh[3455])
print(type(y_comb_oh),y_comb_oh.shape)
#print(y_cat_oh.shape)
#print(y_dog_oh.shape)
#print(y_dog_oh)
#print(y_cat_oh)
```

```
<class 'numpy.ndarray'> (2000, 32, 32, 3)
<class 'numpy.ndarray'> (2000, 32, 32, 3)
Valores minimos y maximos sin normalizar
0 255
0 255
4 (4000, 32, 32, 3)
Valores minimos y maximos normalizados
0.0 1.0
[1]
[0. 1.]
<class 'numpy.ndarray'> (4000, 2)
```

1.2 Division de datos 60 20 20

[5]:
```
xx,yy,ww,zz=x_cat.shape

x_train=np.vstack((x_cat[:int(xx*0.6),:,:,1:2],x_dog[:int(xx*0.6),:,:,1:2]))
y_train=np.vstack((y_cat_oh[:int(xx*0.6),:],y_dog_oh[:int(xx*0.6),:]))
x_val=np.vstack((x_cat[int(xx*0.6):int(xx*0.8),:,:,1:2],x_dog[int(xx*0.6):
 ↪int(xx*0.8),:,:,1:2]))
y_val=np.vstack((y_cat_oh[int(xx*0.6):int(xx*0.8),:],y_dog_oh[int(xx*0.6):
 ↪int(xx*0.8),:]))
x_test=np.vstack((x_cat[int(xx*0.8):,:,:,1:2],x_dog[int(xx*0.8):,:,:,1:2]))
y_test=np.vstack((y_cat_oh[int(xx*0.8):,:],y_dog_oh[int(xx*0.8):,:]))

print(x_test.min(),x_test.max())
```

```
0.0 1.0
```

1.3 Red LeNet-5

[5]:
```
import tensorflow as tf
from keras.models import Model, load_model
#from tensorflow.keras.applications.resnet50 import preprocess_input,␣
 ↪decode_predictions
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Flatten,Dropout,Input
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping,␣
 ↪ReduceLROnPlateau

lr_reduce = ReduceLROnPlateau(monitor='val_accuracy', factor=0.6, patience=8,␣
 ↪verbose=1, mode='max', min_lr=5e-5)
checkpoint =  ModelCheckpoint('vgg16_finetune.h5', monitor= 'val_accuracy',␣
 ↪mode= 'max', save_best_only = True, verbose= 1)
earlystopper = EarlyStopping(monitor = 'val_loss', min_delta = 0, patience = 10,␣
 ↪verbose = 1, restore_best_weights = True)

#"""

#"""

model=Sequential()
model.add(tf.keras.layers.
 ↪Conv2D(6,(5,5),input_shape=(x_size,y_size,1),activation='tanh',padding='valid',strides=1))␣
 ↪#C1

model.add(tf.keras.layers.AveragePooling2D(pool_size=(2,2))) #S2

model.add(tf.keras.layers.
 ↪Conv2D(16,(5,5),activation='tanh',padding='valid',strides=1)) #c3
```

```
model.add(tf.keras.layers.AveragePooling2D(pool_size=(2,2))) #s4
model.add(tf.keras.layers.Flatten())

model.add(Dense(120,activation='tanh')) #c5
model.add(Dense(84,activation='tanh')) #c6

from keras.layers import Layer
from keras import backend as K

class RBFLayer(Layer):
    def __init__(self, units, gamma, **kwargs):
        super(RBFLayer, self).__init__(**kwargs)
        self.units = units
        self.gamma = K.cast_to_floatx(gamma)

    def build(self, input_shape):
        self.mu = self.add_weight(name='mu',
                                  shape=(int(input_shape[1]), self.units),
                                  initializer='uniform',
                                  trainable=True)
        super(RBFLayer, self).build(input_shape)

    def call(self, inputs):
        diff = K.expand_dims(inputs) - self.mu
        l2 = K.sum(K.pow(diff,2), axis=1)
        res = K.exp(-1 * self.gamma * l2)
        return res

    def compute_output_shape(self, input_shape):
        return (input_shape[0], self.units)

model.add(RBFLayer(2,0.5)) #c7

model.compile(loss='categorical_crossentropy',optimizer=tf.keras.optimizers.
 ↪SGD(learning_rate=0.25),metrics=['accuracy'])
hist_1=model.fit(x_train,y_train,verbose=1,␣
 ↪batch_size=64,epochs=80,validation_data=(x_val,y_val))
```

```
Epoch 1/80
38/38 [==============================] - 3s 71ms/step - loss: 0.6930 - accuracy:
0.5063 - val_loss: 0.6925 - val_accuracy: 0.5150
Epoch 2/80
38/38 [==============================] - 2s 58ms/step - loss: 0.6916 - accuracy:
0.5367 - val_loss: 0.6893 - val_accuracy: 0.5437
Epoch 3/80
38/38 [==============================] - 2s 60ms/step - loss: 0.6883 - accuracy:
0.5542 - val_loss: 0.6863 - val_accuracy: 0.5638
```

```
Epoch 4/80
38/38 [==============================] - 2s 53ms/step - loss: 0.6834 - accuracy:
0.5646 - val_loss: 0.6921 - val_accuracy: 0.5175
Epoch 5/80
38/38 [==============================] - 2s 55ms/step - loss: 0.6818 - accuracy:
0.5537 - val_loss: 0.6846 - val_accuracy: 0.5425
Epoch 6/80
38/38 [==============================] - 2s 55ms/step - loss: 0.6776 - accuracy:
0.5750 - val_loss: 0.6920 - val_accuracy: 0.5288
Epoch 7/80
38/38 [==============================] - 2s 55ms/step - loss: 0.6760 - accuracy:
0.5858 - val_loss: 0.6864 - val_accuracy: 0.5537
Epoch 8/80
38/38 [==============================] - 2s 51ms/step - loss: 0.6767 - accuracy:
0.5688 - val_loss: 0.6889 - val_accuracy: 0.5475
Epoch 9/80
38/38 [==============================] - 2s 46ms/step - loss: 0.6759 - accuracy:
0.5733 - val_loss: 0.6913 - val_accuracy: 0.5462
Epoch 10/80
38/38 [==============================] - 2s 43ms/step - loss: 0.6741 - accuracy:
0.5750 - val_loss: 0.6922 - val_accuracy: 0.5350
Epoch 11/80
38/38 [==============================] - 2s 48ms/step - loss: 0.6731 - accuracy:
0.5833 - val_loss: 0.6896 - val_accuracy: 0.5763
Epoch 12/80
38/38 [==============================] - 2s 45ms/step - loss: 0.6738 - accuracy:
0.5846 - val_loss: 0.6959 - val_accuracy: 0.5362
Epoch 13/80
38/38 [==============================] - 2s 49ms/step - loss: 0.6714 - accuracy:
0.5796 - val_loss: 0.6881 - val_accuracy: 0.5512
Epoch 14/80
38/38 [==============================] - 2s 50ms/step - loss: 0.6710 - accuracy:
0.5792 - val_loss: 0.6994 - val_accuracy: 0.5325
Epoch 15/80
38/38 [==============================] - 2s 47ms/step - loss: 0.6710 - accuracy:
0.5921 - val_loss: 0.6925 - val_accuracy: 0.5362
Epoch 16/80
38/38 [==============================] - 2s 50ms/step - loss: 0.6703 - accuracy:
0.5863 - val_loss: 0.6932 - val_accuracy: 0.5625
Epoch 17/80
38/38 [==============================] - 2s 49ms/step - loss: 0.6661 - accuracy:
0.5987 - val_loss: 0.6976 - val_accuracy: 0.5400
Epoch 18/80
38/38 [==============================] - 2s 53ms/step - loss: 0.6657 - accuracy:
0.5950 - val_loss: 0.6916 - val_accuracy: 0.5113
Epoch 19/80
38/38 [==============================] - 2s 45ms/step - loss: 0.6660 - accuracy:
0.5858 - val_loss: 0.7013 - val_accuracy: 0.5213
```

```
Epoch 20/80
38/38 [==============================] - 2s 45ms/step - loss: 0.6628 - accuracy:
0.5962 - val_loss: 0.6891 - val_accuracy: 0.5575
Epoch 21/80
38/38 [==============================] - 2s 48ms/step - loss: 0.6615 - accuracy:
0.5987 - val_loss: 0.6861 - val_accuracy: 0.5775
Epoch 22/80
38/38 [==============================] - 2s 53ms/step - loss: 0.6598 - accuracy:
0.6050 - val_loss: 0.6814 - val_accuracy: 0.5575
Epoch 23/80
38/38 [==============================] - 2s 56ms/step - loss: 0.6571 - accuracy:
0.6075 - val_loss: 0.6847 - val_accuracy: 0.5763
Epoch 24/80
38/38 [==============================] - 2s 53ms/step - loss: 0.6579 - accuracy:
0.6046 - val_loss: 0.6829 - val_accuracy: 0.5800
Epoch 25/80
38/38 [==============================] - 2s 56ms/step - loss: 0.6516 - accuracy:
0.6142 - val_loss: 0.6850 - val_accuracy: 0.5913
Epoch 26/80
38/38 [==============================] - 2s 56ms/step - loss: 0.6457 - accuracy:
0.6292 - val_loss: 0.6828 - val_accuracy: 0.5525
Epoch 27/80
38/38 [==============================] - 2s 48ms/step - loss: 0.6459 - accuracy:
0.6167 - val_loss: 0.7054 - val_accuracy: 0.5387
Epoch 28/80
38/38 [==============================] - 2s 55ms/step - loss: 0.6416 - accuracy:
0.6313 - val_loss: 0.6972 - val_accuracy: 0.5688
Epoch 29/80
38/38 [==============================] - 2s 54ms/step - loss: 0.6379 - accuracy:
0.6317 - val_loss: 0.6718 - val_accuracy: 0.5888
Epoch 30/80
38/38 [==============================] - 2s 51ms/step - loss: 0.6367 - accuracy:
0.6263 - val_loss: 0.6731 - val_accuracy: 0.5987
Epoch 31/80
38/38 [==============================] - 2s 52ms/step - loss: 0.6368 - accuracy:
0.6267 - val_loss: 0.6863 - val_accuracy: 0.5638
Epoch 32/80
38/38 [==============================] - 2s 61ms/step - loss: 0.6349 - accuracy:
0.6325 - val_loss: 0.6650 - val_accuracy: 0.6125
Epoch 33/80
38/38 [==============================] - 2s 54ms/step - loss: 0.6271 - accuracy:
0.6363 - val_loss: 0.6768 - val_accuracy: 0.5962
Epoch 34/80
38/38 [==============================] - 2s 57ms/step - loss: 0.6301 - accuracy:
0.6279 - val_loss: 0.6881 - val_accuracy: 0.5900
Epoch 35/80
38/38 [==============================] - 2s 58ms/step - loss: 0.6287 - accuracy:
0.6421 - val_loss: 0.6680 - val_accuracy: 0.5987
```

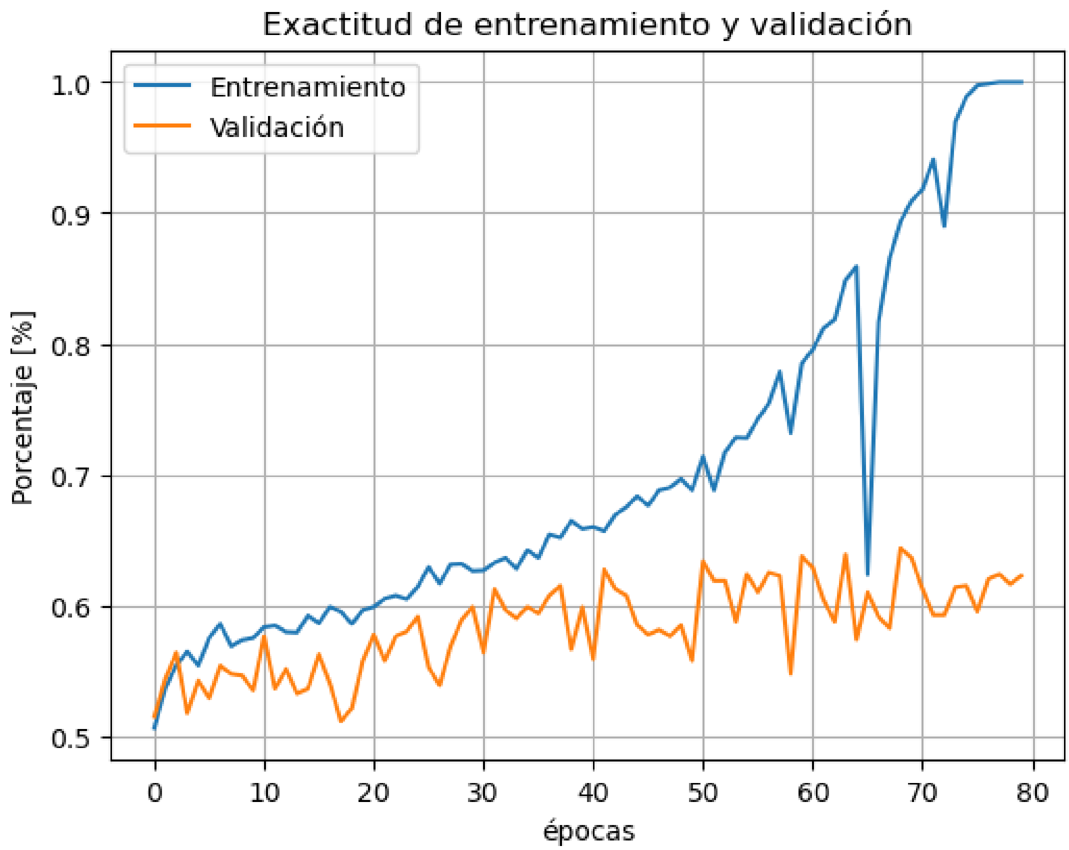
```
Epoch 36/80
38/38 [==============================] - 2s 55ms/step - loss: 0.6319 - accuracy:
0.6363 - val_loss: 0.6739 - val_accuracy: 0.5938
Epoch 37/80
38/38 [==============================] - 2s 58ms/step - loss: 0.6182 - accuracy:
0.6542 - val_loss: 0.6757 - val_accuracy: 0.6075
Epoch 38/80
38/38 [==============================] - 2s 56ms/step - loss: 0.6249 - accuracy:
0.6517 - val_loss: 0.6752 - val_accuracy: 0.6150
Epoch 39/80
38/38 [==============================] - 2s 53ms/step - loss: 0.6164 - accuracy:
0.6646 - val_loss: 0.6824 - val_accuracy: 0.5663
Epoch 40/80
38/38 [==============================] - 2s 55ms/step - loss: 0.6135 - accuracy:
0.6583 - val_loss: 0.6717 - val_accuracy: 0.5987
Epoch 41/80
38/38 [==============================] - 2s 53ms/step - loss: 0.6061 - accuracy:
0.6600 - val_loss: 0.7221 - val_accuracy: 0.5587
Epoch 42/80
38/38 [==============================] - 2s 41ms/step - loss: 0.6080 - accuracy:
0.6567 - val_loss: 0.6651 - val_accuracy: 0.6275
Epoch 43/80
38/38 [==============================] - 2s 44ms/step - loss: 0.5990 - accuracy:
0.6692 - val_loss: 0.6575 - val_accuracy: 0.6125
Epoch 44/80
38/38 [==============================] - 2s 51ms/step - loss: 0.5981 - accuracy:
0.6750 - val_loss: 0.6931 - val_accuracy: 0.6075
Epoch 45/80
38/38 [==============================] - 2s 52ms/step - loss: 0.5906 - accuracy:
0.6833 - val_loss: 0.6767 - val_accuracy: 0.5850
Epoch 46/80
38/38 [==============================] - 2s 55ms/step - loss: 0.5973 - accuracy:
0.6762 - val_loss: 0.7112 - val_accuracy: 0.5775
Epoch 47/80
38/38 [==============================] - 2s 53ms/step - loss: 0.5833 - accuracy:
0.6879 - val_loss: 0.7275 - val_accuracy: 0.5813
Epoch 48/80
38/38 [==============================] - 2s 57ms/step - loss: 0.5757 - accuracy:
0.6900 - val_loss: 0.7209 - val_accuracy: 0.5763
Epoch 49/80
38/38 [==============================] - 2s 52ms/step - loss: 0.5770 - accuracy:
0.6967 - val_loss: 0.7552 - val_accuracy: 0.5850
Epoch 50/80
38/38 [==============================] - 2s 56ms/step - loss: 0.5734 - accuracy:
0.6879 - val_loss: 0.7411 - val_accuracy: 0.5575
Epoch 51/80
38/38 [==============================] - 2s 54ms/step - loss: 0.5580 - accuracy:
0.7138 - val_loss: 0.6524 - val_accuracy: 0.6338
```

```
Epoch 52/80
38/38 [==============================] - 2s 56ms/step - loss: 0.5986 - accuracy:
0.6879 - val_loss: 0.6717 - val_accuracy: 0.6187
Epoch 53/80
38/38 [==============================] - 2s 54ms/step - loss: 0.5423 - accuracy:
0.7171 - val_loss: 0.6901 - val_accuracy: 0.6187
Epoch 54/80
38/38 [==============================] - 2s 55ms/step - loss: 0.5296 - accuracy:
0.7283 - val_loss: 0.7934 - val_accuracy: 0.5875
Epoch 55/80
38/38 [==============================] - 2s 50ms/step - loss: 0.5457 - accuracy:
0.7279 - val_loss: 0.6977 - val_accuracy: 0.6237
Epoch 56/80
38/38 [==============================] - 2s 55ms/step - loss: 0.5063 - accuracy:
0.7425 - val_loss: 0.7346 - val_accuracy: 0.6100
Epoch 57/80
38/38 [==============================] - 2s 53ms/step - loss: 0.4928 - accuracy:
0.7542 - val_loss: 0.7602 - val_accuracy: 0.6250
Epoch 58/80
38/38 [==============================] - 2s 58ms/step - loss: 0.4786 - accuracy:
0.7788 - val_loss: 0.7551 - val_accuracy: 0.6225
Epoch 59/80
38/38 [==============================] - 2s 56ms/step - loss: 0.5706 - accuracy:
0.7317 - val_loss: 0.8334 - val_accuracy: 0.5475
Epoch 60/80
38/38 [==============================] - 2s 55ms/step - loss: 0.4750 - accuracy:
0.7850 - val_loss: 0.7002 - val_accuracy: 0.6375
Epoch 61/80
38/38 [==============================] - 2s 58ms/step - loss: 0.4318 - accuracy:
0.7954 - val_loss: 0.7570 - val_accuracy: 0.6288
Epoch 62/80
38/38 [==============================] - 2s 57ms/step - loss: 0.4048 - accuracy:
0.8121 - val_loss: 0.8416 - val_accuracy: 0.6037
Epoch 63/80
38/38 [==============================] - 2s 59ms/step - loss: 0.4112 - accuracy:
0.8183 - val_loss: 0.8887 - val_accuracy: 0.5875
Epoch 64/80
38/38 [==============================] - 2s 58ms/step - loss: 0.3540 - accuracy:
0.8487 - val_loss: 0.7774 - val_accuracy: 0.6388
Epoch 65/80
38/38 [==============================] - 2s 61ms/step - loss: 0.3343 - accuracy:
0.8592 - val_loss: 1.0568 - val_accuracy: 0.5738
Epoch 66/80
38/38 [==============================] - 2s 55ms/step - loss: 0.9778 - accuracy:
0.6233 - val_loss: 0.7428 - val_accuracy: 0.6100
Epoch 67/80
38/38 [==============================] - 2s 57ms/step - loss: 0.4110 - accuracy:
0.8167 - val_loss: 0.8173 - val_accuracy: 0.5913
```

```
Epoch 68/80
38/38 [==============================] - 2s 61ms/step - loss: 0.3097 - accuracy:
0.8650 - val_loss: 0.9714 - val_accuracy: 0.5825
Epoch 69/80
38/38 [==============================] - 2s 57ms/step - loss: 0.2620 - accuracy:
0.8933 - val_loss: 0.9960 - val_accuracy: 0.6438
Epoch 70/80
38/38 [==============================] - 2s 51ms/step - loss: 0.2380 - accuracy:
0.9092 - val_loss: 0.9471 - val_accuracy: 0.6363
Epoch 71/80
38/38 [==============================] - 2s 56ms/step - loss: 0.2197 - accuracy:
0.9179 - val_loss: 1.0612 - val_accuracy: 0.6125
Epoch 72/80
38/38 [==============================] - 2s 55ms/step - loss: 0.1726 - accuracy:
0.9408 - val_loss: 1.3412 - val_accuracy: 0.5925
Epoch 73/80
38/38 [==============================] - 2s 58ms/step - loss: 0.3085 - accuracy:
0.8900 - val_loss: 1.0773 - val_accuracy: 0.5925
Epoch 74/80
38/38 [==============================] - 2s 57ms/step - loss: 0.1078 - accuracy:
0.9696 - val_loss: 1.1497 - val_accuracy: 0.6137
Epoch 75/80
38/38 [==============================] - 2s 54ms/step - loss: 0.0642 - accuracy:
0.9887 - val_loss: 1.2386 - val_accuracy: 0.6150
Epoch 76/80
38/38 [==============================] - 2s 56ms/step - loss: 0.0382 - accuracy:
0.9975 - val_loss: 1.3674 - val_accuracy: 0.5950
Epoch 77/80
38/38 [==============================] - 2s 55ms/step - loss: 0.0269 - accuracy:
0.9987 - val_loss: 1.3315 - val_accuracy: 0.6200
Epoch 78/80
38/38 [==============================] - 2s 53ms/step - loss: 0.0186 - accuracy:
1.0000 - val_loss: 1.3965 - val_accuracy: 0.6237
Epoch 79/80
38/38 [==============================] - 2s 53ms/step - loss: 0.0133 - accuracy:
1.0000 - val_loss: 1.4764 - val_accuracy: 0.6162
Epoch 80/80
38/38 [==============================] - 2s 57ms/step - loss: 0.0105 - accuracy:
1.0000 - val_loss: 1.4847 - val_accuracy: 0.6225
```

[6]:
```
type(hist_1.history['loss'])
import matplotlib.pyplot as plt
plt.plot(hist_1.history['accuracy'])
plt.plot(hist_1.history['val_accuracy'])
plt.title('Exactitud de entrenamiento y validación')
plt.xlabel('épocas')
plt.ylabel('Porcentaje [%]')
```

```
plt.legend(['Entrenamiento','Validación'])
plt.grid()
```



1.4 Test de al red LeNet5 v1

[7]:
```
pred=model.predict(x_test)
pred=np.argmax(pred,axis=1)
y1=np.argmax(y_test,axis=1)

#label=np.argmax(yp_oh)
exactitud_test=0
for a in range(len(pred)):
    if pred[a]==y1[a]:
        exactitud_test+=1
print('exactitud de la prueba= ',100*exactitud_test/len(pred),'%')
```

```
25/25 [==============================] - 0s 2ms/step
exactitud de la prueba=  59.625 %
```

2 1) Red LeNet5 con regularización mediante L1oL2

l1 o l2

[8]:
```
import tensorflow as tf
from keras.models import Model, load_model
#from tensorflow.keras.applications.resnet50 import preprocess_input,␣
↪decode_predictions
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Flatten,Dropout,Input
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping,␣
↪ReduceLROnPlateau
from tensorflow.keras import regularizers
lr_reduce = ReduceLROnPlateau(monitor='val_accuracy', factor=0.6, patience=8,␣
↪verbose=1, mode='max', min_lr=5e-5)
checkpoint =  ModelCheckpoint('vgg16_finetune.h5', monitor= 'val_accuracy',␣
↪mode= 'max', save_best_only = True, verbose= 1)
earlystopper = EarlyStopping(monitor = 'val_loss', min_delta = 0, patience = 10,␣
↪verbose = 1, restore_best_weights = True)

#"""

#"""
k_r=tf.keras.regularizers.L1L2(l1=1e-5,l2=1e-4)
b_r=tf.keras.regularizers.L2(l2=1e-4)
a_r=tf.keras.regularizers.L2(l2=1e-5)
model=Sequential()
model.add(tf.keras.layers.
↪Conv2D(6,(5,5),input_shape=(x_size,y_size,1),activation='tanh',padding='valid',strides=1))␣
↪#C1

model.add(tf.keras.layers.AveragePooling2D(pool_size=(2,2))) #S2

model.add(tf.keras.layers.
↪Conv2D(16,(5,5),activation='tanh',padding='valid',strides=1)) #c3

model.add(tf.keras.layers.AveragePooling2D(pool_size=(2,2))) #s4
model.add(tf.keras.layers.Flatten())

model.
↪add(Dense(120,activation='tanh',kernel_regularizer=k_r,bias_regularizer=b_r,activity_regulari
↪#c5
model.
↪add(Dense(84,activation='tanh',kernel_regularizer=k_r,bias_regularizer=b_r,activity_regulariz
↪#c6

from keras.layers import Layer
```

```
from keras import backend as K

class RBFLayer(Layer):
    def __init__(self, units, gamma, **kwargs):
        super(RBFLayer, self).__init__(**kwargs)
        self.units = units
        self.gamma = K.cast_to_floatx(gamma)

    def build(self, input_shape):
        self.mu = self.add_weight(name='mu',
                                  shape=(int(input_shape[1]), self.units),
                                  initializer='uniform',
                                  trainable=True)
        super(RBFLayer, self).build(input_shape)

    def call(self, inputs):
        diff = K.expand_dims(inputs) - self.mu
        l2 = K.sum(K.pow(diff,2), axis=1)
        res = K.exp(-1 * self.gamma * l2)
        return res

    def compute_output_shape(self, input_shape):
        return (input_shape[0], self.units)

model.add(RBFLayer(2,0.5)) #c7

model.compile(loss='categorical_crossentropy',optimizer=tf.keras.optimizers.
 ↪SGD(learning_rate=0.23),metrics=['accuracy'])
hist_2=model.fit(x_train,y_train,verbose=1,␣
 ↪batch_size=64,epochs=120,validation_data=(x_val,y_val))
```

```
Epoch 1/120
38/38 [==============================] - 3s 71ms/step - loss: 0.7564 - accuracy:
0.5088 - val_loss: 0.7535 - val_accuracy: 0.5063
Epoch 2/120
38/38 [==============================] - 2s 55ms/step - loss: 0.7526 - accuracy:
0.5371 - val_loss: 0.7547 - val_accuracy: 0.5138
Epoch 3/120
38/38 [==============================] - 2s 52ms/step - loss: 0.7499 - accuracy:
0.5437 - val_loss: 0.7486 - val_accuracy: 0.5400
Epoch 4/120
38/38 [==============================] - 2s 57ms/step - loss: 0.7467 - accuracy:
0.5512 - val_loss: 0.7482 - val_accuracy: 0.5450
Epoch 5/120
38/38 [==============================] - 2s 55ms/step - loss: 0.7428 - accuracy:
0.5592 - val_loss: 0.7492 - val_accuracy: 0.5550
Epoch 6/120
38/38 [==============================] - 2s 54ms/step - loss: 0.7429 - accuracy:
```

```
0.5663 - val_loss: 0.7493 - val_accuracy: 0.5537
Epoch 7/120
38/38 [==============================] - 2s 52ms/step - loss: 0.7404 - accuracy:
0.5654 - val_loss: 0.7511 - val_accuracy: 0.5213
Epoch 8/120
38/38 [==============================] - 2s 54ms/step - loss: 0.7388 - accuracy:
0.5800 - val_loss: 0.7573 - val_accuracy: 0.5362
Epoch 9/120
38/38 [==============================] - 2s 54ms/step - loss: 0.7385 - accuracy:
0.5742 - val_loss: 0.7483 - val_accuracy: 0.5450
Epoch 10/120
38/38 [==============================] - 2s 53ms/step - loss: 0.7364 - accuracy:
0.5692 - val_loss: 0.7506 - val_accuracy: 0.5525
Epoch 11/120
38/38 [==============================] - 2s 53ms/step - loss: 0.7354 - accuracy:
0.5729 - val_loss: 0.7546 - val_accuracy: 0.5475
Epoch 12/120
38/38 [==============================] - 2s 56ms/step - loss: 0.7337 - accuracy:
0.5717 - val_loss: 0.7519 - val_accuracy: 0.5800
Epoch 13/120
38/38 [==============================] - 2s 52ms/step - loss: 0.7332 - accuracy:
0.5742 - val_loss: 0.7529 - val_accuracy: 0.5625
Epoch 14/120
38/38 [==============================] - 2s 52ms/step - loss: 0.7317 - accuracy:
0.5700 - val_loss: 0.7662 - val_accuracy: 0.5325
Epoch 15/120
38/38 [==============================] - 2s 53ms/step - loss: 0.7323 - accuracy:
0.5771 - val_loss: 0.7541 - val_accuracy: 0.5575
Epoch 16/120
38/38 [==============================] - 2s 52ms/step - loss: 0.7303 - accuracy:
0.5792 - val_loss: 0.7515 - val_accuracy: 0.5475
Epoch 17/120
38/38 [==============================] - 2s 59ms/step - loss: 0.7283 - accuracy:
0.5788 - val_loss: 0.7562 - val_accuracy: 0.5550
Epoch 18/120
38/38 [==============================] - 2s 54ms/step - loss: 0.7303 - accuracy:
0.5792 - val_loss: 0.7578 - val_accuracy: 0.5250
Epoch 19/120
38/38 [==============================] - 2s 54ms/step - loss: 0.7260 - accuracy:
0.5967 - val_loss: 0.7492 - val_accuracy: 0.5550
Epoch 20/120
38/38 [==============================] - 2s 54ms/step - loss: 0.7260 - accuracy:
0.5946 - val_loss: 0.7517 - val_accuracy: 0.5525
Epoch 21/120
38/38 [==============================] - 2s 54ms/step - loss: 0.7262 - accuracy:
0.5871 - val_loss: 0.7533 - val_accuracy: 0.5750
Epoch 22/120
38/38 [==============================] - 2s 54ms/step - loss: 0.7237 - accuracy:
```

```
0.5804 - val_loss: 0.7547 - val_accuracy: 0.5462
Epoch 23/120
38/38 [==============================] - 2s 55ms/step - loss: 0.7230 - accuracy:
0.5938 - val_loss: 0.7574 - val_accuracy: 0.5375
Epoch 24/120
38/38 [==============================] - 2s 57ms/step - loss: 0.7226 - accuracy:
0.5992 - val_loss: 0.7584 - val_accuracy: 0.5250
Epoch 25/120
38/38 [==============================] - 2s 54ms/step - loss: 0.7210 - accuracy:
0.5992 - val_loss: 0.7709 - val_accuracy: 0.5200
Epoch 26/120
38/38 [==============================] - 2s 53ms/step - loss: 0.7188 - accuracy:
0.5908 - val_loss: 0.7888 - val_accuracy: 0.5362
Epoch 27/120
38/38 [==============================] - 2s 52ms/step - loss: 0.7175 - accuracy:
0.6000 - val_loss: 0.7586 - val_accuracy: 0.5425
Epoch 28/120
38/38 [==============================] - 2s 51ms/step - loss: 0.7184 - accuracy:
0.5992 - val_loss: 0.7563 - val_accuracy: 0.5288
Epoch 29/120
38/38 [==============================] - 2s 57ms/step - loss: 0.7162 - accuracy:
0.5938 - val_loss: 0.7886 - val_accuracy: 0.5263
Epoch 30/120
38/38 [==============================] - 2s 54ms/step - loss: 0.7153 - accuracy:
0.5946 - val_loss: 0.7460 - val_accuracy: 0.5813
Epoch 31/120
38/38 [==============================] - 2s 54ms/step - loss: 0.7098 - accuracy:
0.6075 - val_loss: 0.7902 - val_accuracy: 0.5375
Epoch 32/120
38/38 [==============================] - 2s 57ms/step - loss: 0.7099 - accuracy:
0.6021 - val_loss: 0.7550 - val_accuracy: 0.5663
Epoch 33/120
38/38 [==============================] - 2s 53ms/step - loss: 0.7094 - accuracy:
0.6067 - val_loss: 0.7431 - val_accuracy: 0.5713
Epoch 34/120
38/38 [==============================] - 2s 55ms/step - loss: 0.7052 - accuracy:
0.6142 - val_loss: 0.7488 - val_accuracy: 0.5625
Epoch 35/120
38/38 [==============================] - 2s 55ms/step - loss: 0.7028 - accuracy:
0.6196 - val_loss: 0.7621 - val_accuracy: 0.5587
Epoch 36/120
38/38 [==============================] - 2s 54ms/step - loss: 0.7004 - accuracy:
0.6208 - val_loss: 0.7492 - val_accuracy: 0.5875
Epoch 37/120
38/38 [==============================] - 2s 58ms/step - loss: 0.6990 - accuracy:
0.6229 - val_loss: 0.7477 - val_accuracy: 0.5650
Epoch 38/120
38/38 [==============================] - 2s 61ms/step - loss: 0.6956 - accuracy:
```

```
0.6242 - val_loss: 0.7892 - val_accuracy: 0.5263
Epoch 39/120
38/38 [==============================] - 2s 52ms/step - loss: 0.6997 - accuracy:
0.6162 - val_loss: 0.7581 - val_accuracy: 0.5688
Epoch 40/120
38/38 [==============================] - 2s 54ms/step - loss: 0.6855 - accuracy:
0.6408 - val_loss: 0.7506 - val_accuracy: 0.5575
Epoch 41/120
38/38 [==============================] - 2s 54ms/step - loss: 0.6867 - accuracy:
0.6375 - val_loss: 0.7421 - val_accuracy: 0.5975
Epoch 42/120
38/38 [==============================] - 2s 53ms/step - loss: 0.6873 - accuracy:
0.6388 - val_loss: 0.7176 - val_accuracy: 0.6062
Epoch 43/120
38/38 [==============================] - 2s 53ms/step - loss: 0.6704 - accuracy:
0.6546 - val_loss: 0.7492 - val_accuracy: 0.5900
Epoch 44/120
38/38 [==============================] - 2s 57ms/step - loss: 0.6666 - accuracy:
0.6525 - val_loss: 0.7202 - val_accuracy: 0.6175
Epoch 45/120
38/38 [==============================] - 2s 53ms/step - loss: 0.6805 - accuracy:
0.6533 - val_loss: 0.7177 - val_accuracy: 0.6125
Epoch 46/120
38/38 [==============================] - 2s 55ms/step - loss: 0.6652 - accuracy:
0.6521 - val_loss: 0.7126 - val_accuracy: 0.6200
Epoch 47/120
38/38 [==============================] - 2s 57ms/step - loss: 0.6586 - accuracy:
0.6608 - val_loss: 0.7485 - val_accuracy: 0.5775
Epoch 48/120
38/38 [==============================] - 2s 56ms/step - loss: 0.6611 - accuracy:
0.6625 - val_loss: 0.7326 - val_accuracy: 0.5938
Epoch 49/120
38/38 [==============================] - 2s 53ms/step - loss: 0.6499 - accuracy:
0.6771 - val_loss: 0.7380 - val_accuracy: 0.6012
Epoch 50/120
38/38 [==============================] - 2s 54ms/step - loss: 0.6422 - accuracy:
0.6792 - val_loss: 0.7344 - val_accuracy: 0.5913
Epoch 51/120
38/38 [==============================] - 2s 51ms/step - loss: 0.6354 - accuracy:
0.6992 - val_loss: 0.7460 - val_accuracy: 0.6112
Epoch 52/120
38/38 [==============================] - 2s 55ms/step - loss: 0.6366 - accuracy:
0.6808 - val_loss: 0.7399 - val_accuracy: 0.6037
Epoch 53/120
38/38 [==============================] - 2s 56ms/step - loss: 0.6314 - accuracy:
0.6954 - val_loss: 0.7421 - val_accuracy: 0.6075
Epoch 54/120
38/38 [==============================] - 2s 52ms/step - loss: 0.6250 - accuracy:
```

```
0.7063 - val_loss: 0.7205 - val_accuracy: 0.6112
Epoch 55/120
38/38 [==============================] - 2s 53ms/step - loss: 0.6125 - accuracy:
0.7008 - val_loss: 0.7352 - val_accuracy: 0.6025
Epoch 56/120
38/38 [==============================] - 2s 57ms/step - loss: 0.6176 - accuracy:
0.7154 - val_loss: 0.7781 - val_accuracy: 0.6075
Epoch 57/120
38/38 [==============================] - 2s 52ms/step - loss: 0.6080 - accuracy:
0.7158 - val_loss: 0.7957 - val_accuracy: 0.5987
Epoch 58/120
38/38 [==============================] - 2s 55ms/step - loss: 0.5968 - accuracy:
0.7200 - val_loss: 0.7884 - val_accuracy: 0.5875
Epoch 59/120
38/38 [==============================] - 2s 55ms/step - loss: 0.5869 - accuracy:
0.7342 - val_loss: 0.8338 - val_accuracy: 0.5825
Epoch 60/120
38/38 [==============================] - 2s 55ms/step - loss: 0.6089 - accuracy:
0.7117 - val_loss: 0.7538 - val_accuracy: 0.6125
Epoch 61/120
38/38 [==============================] - 2s 55ms/step - loss: 0.5621 - accuracy:
0.7483 - val_loss: 0.8078 - val_accuracy: 0.5863
Epoch 62/120
38/38 [==============================] - 2s 53ms/step - loss: 0.5694 - accuracy:
0.7379 - val_loss: 0.7869 - val_accuracy: 0.6187
Epoch 63/120
38/38 [==============================] - 2s 54ms/step - loss: 0.5575 - accuracy:
0.7538 - val_loss: 0.7697 - val_accuracy: 0.6338
Epoch 64/120
38/38 [==============================] - 2s 53ms/step - loss: 0.5335 - accuracy:
0.7717 - val_loss: 0.8061 - val_accuracy: 0.6125
Epoch 65/120
38/38 [==============================] - 2s 54ms/step - loss: 0.5353 - accuracy:
0.7713 - val_loss: 0.9101 - val_accuracy: 0.5925
Epoch 66/120
38/38 [==============================] - 2s 58ms/step - loss: 0.5140 - accuracy:
0.7808 - val_loss: 0.7729 - val_accuracy: 0.6263
Epoch 67/120
38/38 [==============================] - 2s 53ms/step - loss: 0.5048 - accuracy:
0.7742 - val_loss: 0.8514 - val_accuracy: 0.6237
Epoch 68/120
38/38 [==============================] - 2s 54ms/step - loss: 0.4799 - accuracy:
0.8104 - val_loss: 0.8769 - val_accuracy: 0.5962
Epoch 69/120
38/38 [==============================] - 2s 53ms/step - loss: 0.4751 - accuracy:
0.8046 - val_loss: 0.8688 - val_accuracy: 0.6062
Epoch 70/120
38/38 [==============================] - 2s 55ms/step - loss: 0.4397 - accuracy:
```

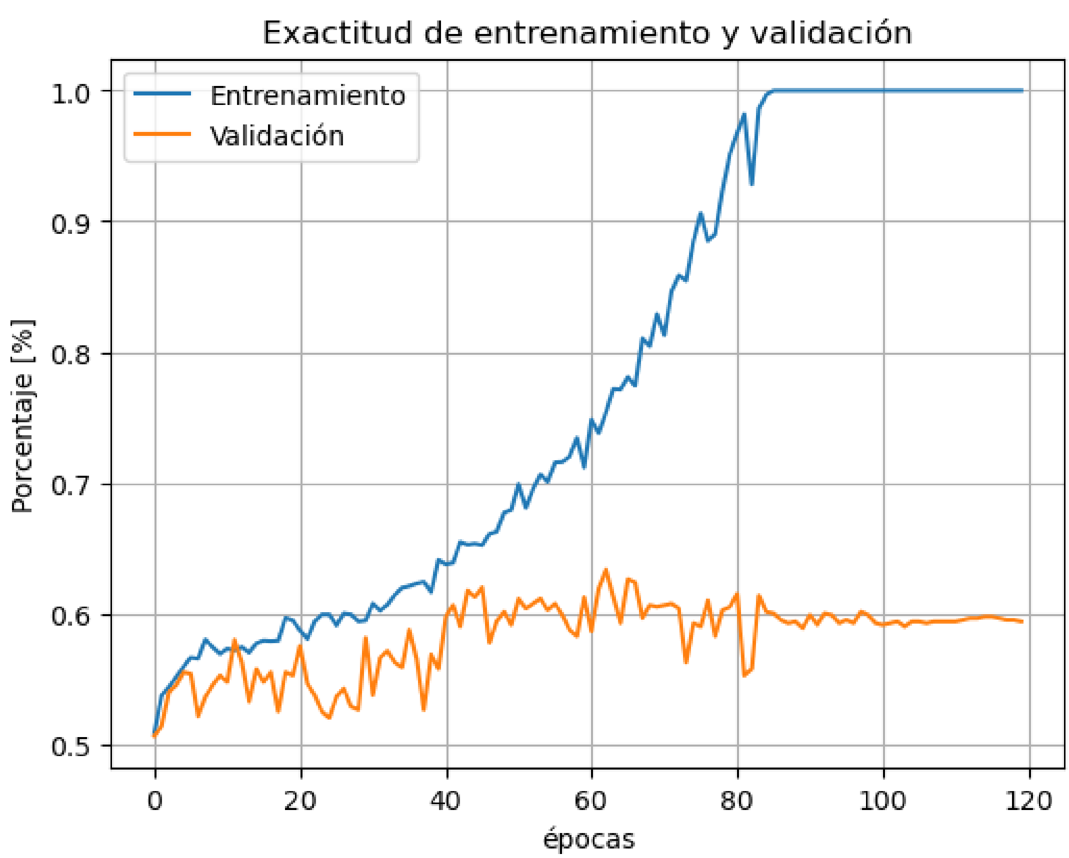
```
0.8292 - val_loss: 0.9179 - val_accuracy: 0.6050
Epoch 71/120
38/38 [==============================] - 2s 57ms/step - loss: 0.4641 - accuracy:
0.8129 - val_loss: 0.8935 - val_accuracy: 0.6062
Epoch 72/120
38/38 [==============================] - 2s 57ms/step - loss: 0.3899 - accuracy:
0.8462 - val_loss: 0.9802 - val_accuracy: 0.6075
Epoch 73/120
38/38 [==============================] - 2s 54ms/step - loss: 0.3803 - accuracy:
0.8587 - val_loss: 1.0266 - val_accuracy: 0.6037
Epoch 74/120
38/38 [==============================] - 2s 54ms/step - loss: 0.3891 - accuracy:
0.8546 - val_loss: 1.1803 - val_accuracy: 0.5625
Epoch 75/120
38/38 [==============================] - 2s 55ms/step - loss: 0.3465 - accuracy:
0.8846 - val_loss: 1.2253 - val_accuracy: 0.5925
Epoch 76/120
38/38 [==============================] - 2s 60ms/step - loss: 0.2966 - accuracy:
0.9062 - val_loss: 1.3348 - val_accuracy: 0.5900
Epoch 77/120
38/38 [==============================] - 2s 64ms/step - loss: 0.3419 - accuracy:
0.8850 - val_loss: 1.1808 - val_accuracy: 0.6100
Epoch 78/120
38/38 [==============================] - 2s 55ms/step - loss: 0.3174 - accuracy:
0.8904 - val_loss: 1.3240 - val_accuracy: 0.5825
Epoch 79/120
38/38 [==============================] - 2s 63ms/step - loss: 0.2509 - accuracy:
0.9237 - val_loss: 1.3487 - val_accuracy: 0.6025
Epoch 80/120
38/38 [==============================] - 2s 59ms/step - loss: 0.1983 - accuracy:
0.9513 - val_loss: 1.3461 - val_accuracy: 0.6050
Epoch 81/120
38/38 [==============================] - 2s 51ms/step - loss: 0.1603 - accuracy:
0.9675 - val_loss: 1.4385 - val_accuracy: 0.6150
Epoch 82/120
38/38 [==============================] - 2s 54ms/step - loss: 0.1321 - accuracy:
0.9821 - val_loss: 1.7675 - val_accuracy: 0.5525
Epoch 83/120
38/38 [==============================] - 2s 58ms/step - loss: 0.2361 - accuracy:
0.9283 - val_loss: 1.5848 - val_accuracy: 0.5575
Epoch 84/120
38/38 [==============================] - 2s 56ms/step - loss: 0.1228 - accuracy:
0.9862 - val_loss: 1.3888 - val_accuracy: 0.6137
Epoch 85/120
38/38 [==============================] - 2s 60ms/step - loss: 0.0923 - accuracy:
0.9967 - val_loss: 1.4930 - val_accuracy: 0.6012
Epoch 86/120
38/38 [==============================] - 2s 62ms/step - loss: 0.0772 - accuracy:
```

```
1.0000 - val_loss: 1.6005 - val_accuracy: 0.6000
Epoch 87/120
38/38 [==============================] - 2s 61ms/step - loss: 0.0726 - accuracy:
1.0000 - val_loss: 1.5837 - val_accuracy: 0.5950
Epoch 88/120
38/38 [==============================] - 2s 64ms/step - loss: 0.0696 - accuracy:
1.0000 - val_loss: 1.6472 - val_accuracy: 0.5925
Epoch 89/120
38/38 [==============================] - 3599s 97s/step - loss: 0.0681 -
accuracy: 1.0000 - val_loss: 1.6897 - val_accuracy: 0.5938
Epoch 90/120
38/38 [==============================] - 3s 67ms/step - loss: 0.0669 - accuracy:
1.0000 - val_loss: 1.7287 - val_accuracy: 0.5888
Epoch 91/120
38/38 [==============================] - 3s 67ms/step - loss: 0.0658 - accuracy:
1.0000 - val_loss: 1.7681 - val_accuracy: 0.5987
Epoch 92/120
38/38 [==============================] - 2s 65ms/step - loss: 0.0650 - accuracy:
1.0000 - val_loss: 1.7874 - val_accuracy: 0.5913
Epoch 93/120
38/38 [==============================] - 2s 59ms/step - loss: 0.0642 - accuracy:
1.0000 - val_loss: 1.8215 - val_accuracy: 0.6000
Epoch 94/120
38/38 [==============================] - 2s 56ms/step - loss: 0.0636 - accuracy:
1.0000 - val_loss: 1.8431 - val_accuracy: 0.5987
Epoch 95/120
38/38 [==============================] - 2s 55ms/step - loss: 0.0631 - accuracy:
1.0000 - val_loss: 1.8722 - val_accuracy: 0.5925
Epoch 96/120
38/38 [==============================] - 2s 64ms/step - loss: 0.0626 - accuracy:
1.0000 - val_loss: 1.8845 - val_accuracy: 0.5950
Epoch 97/120
38/38 [==============================] - 34s 905ms/step - loss: 0.0621 -
accuracy: 1.0000 - val_loss: 1.9033 - val_accuracy: 0.5925
Epoch 98/120
38/38 [==============================] - 2s 63ms/step - loss: 0.0617 - accuracy:
1.0000 - val_loss: 1.9307 - val_accuracy: 0.6012
Epoch 99/120
38/38 [==============================] - 2s 61ms/step - loss: 0.0613 - accuracy:
1.0000 - val_loss: 1.9421 - val_accuracy: 0.5987
Epoch 100/120
38/38 [==============================] - 3s 68ms/step - loss: 0.0609 - accuracy:
1.0000 - val_loss: 1.9677 - val_accuracy: 0.5925
Epoch 101/120
38/38 [==============================] - 2s 44ms/step - loss: 0.0605 - accuracy:
1.0000 - val_loss: 1.9803 - val_accuracy: 0.5913
Epoch 102/120
38/38 [==============================] - 2s 44ms/step - loss: 0.0602 - accuracy:
```

```
1.0000 - val_loss: 1.9894 - val_accuracy: 0.5925
Epoch 103/120
38/38 [==============================] - 2s 43ms/step - loss: 0.0599 - accuracy:
1.0000 - val_loss: 2.0127 - val_accuracy: 0.5938
Epoch 104/120
38/38 [==============================] - 2s 46ms/step - loss: 0.0595 - accuracy:
1.0000 - val_loss: 2.0277 - val_accuracy: 0.5900
Epoch 105/120
38/38 [==============================] - 2s 41ms/step - loss: 0.0592 - accuracy:
1.0000 - val_loss: 2.0362 - val_accuracy: 0.5938
Epoch 106/120
38/38 [==============================] - 2s 46ms/step - loss: 0.0589 - accuracy:
1.0000 - val_loss: 2.0486 - val_accuracy: 0.5938
Epoch 107/120
38/38 [==============================] - 2s 44ms/step - loss: 0.0586 - accuracy:
1.0000 - val_loss: 2.0630 - val_accuracy: 0.5925
Epoch 108/120
38/38 [==============================] - 2s 51ms/step - loss: 0.0583 - accuracy:
1.0000 - val_loss: 2.0725 - val_accuracy: 0.5938
Epoch 109/120
38/38 [==============================] - 2s 44ms/step - loss: 0.0580 - accuracy:
1.0000 - val_loss: 2.0844 - val_accuracy: 0.5938
Epoch 110/120
38/38 [==============================] - 2s 50ms/step - loss: 0.0578 - accuracy:
1.0000 - val_loss: 2.1003 - val_accuracy: 0.5938
Epoch 111/120
38/38 [==============================] - 2s 56ms/step - loss: 0.0575 - accuracy:
1.0000 - val_loss: 2.1042 - val_accuracy: 0.5938
Epoch 112/120
38/38 [==============================] - 2s 45ms/step - loss: 0.0572 - accuracy:
1.0000 - val_loss: 2.1151 - val_accuracy: 0.5950
Epoch 113/120
38/38 [==============================] - 2s 46ms/step - loss: 0.0570 - accuracy:
1.0000 - val_loss: 2.1294 - val_accuracy: 0.5962
Epoch 114/120
38/38 [==============================] - 2s 48ms/step - loss: 0.0567 - accuracy:
1.0000 - val_loss: 2.1402 - val_accuracy: 0.5962
Epoch 115/120
38/38 [==============================] - 2s 43ms/step - loss: 0.0564 - accuracy:
1.0000 - val_loss: 2.1437 - val_accuracy: 0.5975
Epoch 116/120
38/38 [==============================] - 2s 47ms/step - loss: 0.0562 - accuracy:
1.0000 - val_loss: 2.1603 - val_accuracy: 0.5975
Epoch 117/120
38/38 [==============================] - 2s 45ms/step - loss: 0.0559 - accuracy:
1.0000 - val_loss: 2.1665 - val_accuracy: 0.5962
Epoch 118/120
38/38 [==============================] - 2s 45ms/step - loss: 0.0557 - accuracy:
```

```
1.0000 - val_loss: 2.1736 - val_accuracy: 0.5950
Epoch 119/120
38/38 [==============================] - 2s 53ms/step - loss: 0.0554 - accuracy:
1.0000 - val_loss: 2.1805 - val_accuracy: 0.5950
Epoch 120/120
38/38 [==============================] - 2s 49ms/step - loss: 0.0552 - accuracy:
1.0000 - val_loss: 2.1901 - val_accuracy: 0.5938
```

[9]:
```
type(hist_2.history['loss'])
import matplotlib.pyplot as plt
plt.plot(hist_2.history['accuracy'])
plt.plot(hist_2.history['val_accuracy'])
plt.title('Exactitud de entrenamiento y validación')
plt.xlabel('épocas')
plt.ylabel('Porcentaje [%]')
plt.legend(['Entrenamiento','Validación'])
plt.grid()
```



[10]:
```
pred=model.predict(x_test)
pred=np.argmax(pred,axis=1)
```

```
y1=np.argmax(y_test,axis=1)

#label=np.argmax(yp_oh)
exactitud_test=0
for a in range(len(pred)):
    if pred[a]==y1[a]:
        exactitud_test+=1
print('exactitud de la prueba= ',100*exactitud_test/len(pred),'%')
```

```
25/25 [==============================] - 0s 1ms/step
exactitud de la prueba=  58.75 %
```

3 2) Regularización con Dropout

Uso de Dropout para capas flatten y convolucionales.

[11]:
```
import tensorflow as tf
from keras.models import Model, load_model
#from tensorflow.keras.applications.resnet50 import preprocess_input,␣
↪decode_predictions
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Flatten,Dropout,Input
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping,␣
↪ReduceLROnPlateau

lr_reduce = ReduceLROnPlateau(monitor='val_accuracy', factor=0.6, patience=8,␣
↪verbose=1, mode='max', min_lr=5e-5)
checkpoint =  ModelCheckpoint('vgg16_finetune.h5', monitor= 'val_accuracy',␣
↪mode= 'max', save_best_only = True, verbose= 1)
earlystopper = EarlyStopping(monitor = 'val_loss', min_delta = 0, patience = 10,␣
↪verbose = 1, restore_best_weights = True)

#"""

#"""

model=Sequential()
model.add(tf.keras.layers.
↪Conv2D(6,(5,5),input_shape=(x_size,y_size,1),activation='tanh',padding='valid',strides=1))␣
↪#C1

model.add(tf.keras.layers.AveragePooling2D(pool_size=(2,2))) #S2

model.add(tf.keras.layers.
↪Conv2D(16,(5,5),activation='tanh',padding='valid',strides=1)) #c3
model.add(Dropout(0.3))
model.add(tf.keras.layers.AveragePooling2D(pool_size=(2,2))) #s4
```

```
model.add(tf.keras.layers.Flatten())

model.add(Dense(120,activation='tanh')) #c5
model.add(Dropout(0.5))
model.add(Dense(84,activation='tanh')) #c6
model.add(Dropout(0.6))
from keras.layers import Layer
from keras import backend as K

class RBFLayer(Layer):
    def __init__(self, units, gamma, **kwargs):
        super(RBFLayer, self).__init__(**kwargs)
        self.units = units
        self.gamma = K.cast_to_floatx(gamma)

    def build(self, input_shape):
        self.mu = self.add_weight(name='mu',
                                  shape=(int(input_shape[1]), self.units),
                                  initializer='uniform',
                                  trainable=True)
        super(RBFLayer, self).build(input_shape)

    def call(self, inputs):
        diff = K.expand_dims(inputs) - self.mu
        l2 = K.sum(K.pow(diff,2), axis=1)
        res = K.exp(-1 * self.gamma * l2)
        return res

    def compute_output_shape(self, input_shape):
        return (input_shape[0], self.units)

model.add(RBFLayer(2,0.5)) #c7

model.compile(loss='categorical_crossentropy',optimizer=tf.keras.optimizers.
 ↪SGD(learning_rate=0.25),metrics=['accuracy'])
hist_3=model.fit(x_train,y_train,verbose=1,␣
 ↪batch_size=64,epochs=120,validation_data=(x_val,y_val))
```

```
Epoch 1/120
38/38 [==============================] - 3s 62ms/step - loss: 0.6964 - accuracy:
0.4971 - val_loss: 0.6926 - val_accuracy: 0.5013
Epoch 2/120
38/38 [==============================] - 2s 45ms/step - loss: 0.6958 - accuracy:
0.5108 - val_loss: 0.6927 - val_accuracy: 0.5000
Epoch 3/120
38/38 [==============================] - 2s 46ms/step - loss: 0.6924 - accuracy:
0.5188 - val_loss: 0.6958 - val_accuracy: 0.5050
Epoch 4/120
```

```
38/38 [==============================] - 2s 44ms/step - loss: 0.6923 - accuracy:
0.5179 - val_loss: 0.6889 - val_accuracy: 0.5638
Epoch 5/120
38/38 [==============================] - 2s 48ms/step - loss: 0.6921 - accuracy:
0.5396 - val_loss: 0.6957 - val_accuracy: 0.4950
Epoch 6/120
38/38 [==============================] - 2s 46ms/step - loss: 0.6904 - accuracy:
0.5350 - val_loss: 0.6921 - val_accuracy: 0.5238
Epoch 7/120
38/38 [==============================] - 2s 48ms/step - loss: 0.6881 - accuracy:
0.5471 - val_loss: 0.6917 - val_accuracy: 0.5188
Epoch 8/120
38/38 [==============================] - 2s 44ms/step - loss: 0.6933 - accuracy:
0.5242 - val_loss: 0.6895 - val_accuracy: 0.5550
Epoch 9/120
38/38 [==============================] - 2s 47ms/step - loss: 0.6888 - accuracy:
0.5429 - val_loss: 0.6901 - val_accuracy: 0.5288
Epoch 10/120
38/38 [==============================] - 2s 51ms/step - loss: 0.6862 - accuracy:
0.5713 - val_loss: 0.6879 - val_accuracy: 0.5387
Epoch 11/120
38/38 [==============================] - 2s 61ms/step - loss: 0.6874 - accuracy:
0.5483 - val_loss: 0.6849 - val_accuracy: 0.5625
Epoch 12/120
38/38 [==============================] - 2s 50ms/step - loss: 0.6844 - accuracy:
0.5562 - val_loss: 0.6939 - val_accuracy: 0.5425
Epoch 13/120
38/38 [==============================] - 2s 56ms/step - loss: 0.6839 - accuracy:
0.5654 - val_loss: 0.6876 - val_accuracy: 0.5575
Epoch 14/120
38/38 [==============================] - 2s 59ms/step - loss: 0.6809 - accuracy:
0.5604 - val_loss: 0.7033 - val_accuracy: 0.5238
Epoch 15/120
38/38 [==============================] - 2s 56ms/step - loss: 0.6868 - accuracy:
0.5537 - val_loss: 0.6947 - val_accuracy: 0.5188
Epoch 16/120
38/38 [==============================] - 2s 56ms/step - loss: 0.6848 - accuracy:
0.5558 - val_loss: 0.6923 - val_accuracy: 0.5325
Epoch 17/120
38/38 [==============================] - 2s 51ms/step - loss: 0.6804 - accuracy:
0.5654 - val_loss: 0.6895 - val_accuracy: 0.5487
Epoch 18/120
38/38 [==============================] - 2s 55ms/step - loss: 0.6829 - accuracy:
0.5733 - val_loss: 0.6928 - val_accuracy: 0.5525
Epoch 19/120
38/38 [==============================] - 2s 54ms/step - loss: 0.6831 - accuracy:
0.5742 - val_loss: 0.7072 - val_accuracy: 0.5213
Epoch 20/120
```

```
38/38 [==============================] - 2s 53ms/step - loss: 0.6823 - accuracy:
0.5646 - val_loss: 0.6861 - val_accuracy: 0.5512
Epoch 21/120
38/38 [==============================] - 2s 52ms/step - loss: 0.6820 - accuracy:
0.5717 - val_loss: 0.6910 - val_accuracy: 0.5638
Epoch 22/120
38/38 [==============================] - 2s 54ms/step - loss: 0.6820 - accuracy:
0.5675 - val_loss: 0.6957 - val_accuracy: 0.5450
Epoch 23/120
38/38 [==============================] - 2s 53ms/step - loss: 0.6812 - accuracy:
0.5775 - val_loss: 0.6904 - val_accuracy: 0.5462
Epoch 24/120
38/38 [==============================] - 2s 54ms/step - loss: 0.6827 - accuracy:
0.5604 - val_loss: 0.6971 - val_accuracy: 0.5350
Epoch 25/120
38/38 [==============================] - 2s 53ms/step - loss: 0.6828 - accuracy:
0.5788 - val_loss: 0.6891 - val_accuracy: 0.5638
Epoch 26/120
38/38 [==============================] - 2s 55ms/step - loss: 0.6803 - accuracy:
0.5692 - val_loss: 0.6963 - val_accuracy: 0.5238
Epoch 27/120
38/38 [==============================] - 2s 52ms/step - loss: 0.6855 - accuracy:
0.5658 - val_loss: 0.6913 - val_accuracy: 0.5312
Epoch 28/120
38/38 [==============================] - 2s 56ms/step - loss: 0.6832 - accuracy:
0.5571 - val_loss: 0.6913 - val_accuracy: 0.5487
Epoch 29/120
38/38 [==============================] - 2s 56ms/step - loss: 0.6783 - accuracy:
0.5658 - val_loss: 0.6990 - val_accuracy: 0.5213
Epoch 30/120
38/38 [==============================] - 2s 57ms/step - loss: 0.6803 - accuracy:
0.5783 - val_loss: 0.6910 - val_accuracy: 0.5750
Epoch 31/120
38/38 [==============================] - 2s 55ms/step - loss: 0.6806 - accuracy:
0.5763 - val_loss: 0.6887 - val_accuracy: 0.5650
Epoch 32/120
38/38 [==============================] - 2s 52ms/step - loss: 0.6785 - accuracy:
0.5729 - val_loss: 0.6905 - val_accuracy: 0.5675
Epoch 33/120
38/38 [==============================] - 2s 54ms/step - loss: 0.6803 - accuracy:
0.5717 - val_loss: 0.6917 - val_accuracy: 0.5900
Epoch 34/120
38/38 [==============================] - 2s 49ms/step - loss: 0.6791 - accuracy:
0.5725 - val_loss: 0.6892 - val_accuracy: 0.5638
Epoch 35/120
38/38 [==============================] - 2s 48ms/step - loss: 0.6778 - accuracy:
0.5788 - val_loss: 0.6919 - val_accuracy: 0.5550
Epoch 36/120
```

```
38/38 [==============================] - 2s 53ms/step - loss: 0.6783 - accuracy:
0.5717 - val_loss: 0.6914 - val_accuracy: 0.5813
Epoch 37/120
38/38 [==============================] - 2s 48ms/step - loss: 0.6802 - accuracy:
0.5671 - val_loss: 0.6903 - val_accuracy: 0.5562
Epoch 38/120
38/38 [==============================] - 2s 49ms/step - loss: 0.6758 - accuracy:
0.5733 - val_loss: 0.6972 - val_accuracy: 0.5638
Epoch 39/120
38/38 [==============================] - 2s 53ms/step - loss: 0.6754 - accuracy:
0.5717 - val_loss: 0.6962 - val_accuracy: 0.5475
Epoch 40/120
38/38 [==============================] - 2s 54ms/step - loss: 0.6756 - accuracy:
0.5800 - val_loss: 0.6995 - val_accuracy: 0.5562
Epoch 41/120
38/38 [==============================] - 2s 54ms/step - loss: 0.6801 - accuracy:
0.5717 - val_loss: 0.6859 - val_accuracy: 0.5888
Epoch 42/120
38/38 [==============================] - 2s 52ms/step - loss: 0.6764 - accuracy:
0.5833 - val_loss: 0.7044 - val_accuracy: 0.5562
Epoch 43/120
38/38 [==============================] - 2s 48ms/step - loss: 0.6783 - accuracy:
0.5821 - val_loss: 0.7103 - val_accuracy: 0.5675
Epoch 44/120
38/38 [==============================] - 2s 58ms/step - loss: 0.6746 - accuracy:
0.5938 - val_loss: 0.6979 - val_accuracy: 0.5400
Epoch 45/120
38/38 [==============================] - 2s 54ms/step - loss: 0.6759 - accuracy:
0.5779 - val_loss: 0.6925 - val_accuracy: 0.5525
Epoch 46/120
38/38 [==============================] - 2s 55ms/step - loss: 0.6805 - accuracy:
0.5775 - val_loss: 0.6932 - val_accuracy: 0.5625
Epoch 47/120
38/38 [==============================] - 2s 56ms/step - loss: 0.6748 - accuracy:
0.5858 - val_loss: 0.6929 - val_accuracy: 0.5700
Epoch 48/120
38/38 [==============================] - 2s 52ms/step - loss: 0.6762 - accuracy:
0.5729 - val_loss: 0.6902 - val_accuracy: 0.5475
Epoch 49/120
38/38 [==============================] - 2s 51ms/step - loss: 0.6762 - accuracy:
0.5700 - val_loss: 0.6985 - val_accuracy: 0.5562
Epoch 50/120
38/38 [==============================] - 2s 56ms/step - loss: 0.6745 - accuracy:
0.5879 - val_loss: 0.7154 - val_accuracy: 0.5175
Epoch 51/120
38/38 [==============================] - 2s 53ms/step - loss: 0.6770 - accuracy:
0.5771 - val_loss: 0.6995 - val_accuracy: 0.5638
Epoch 52/120
```

```
38/38 [==============================] - 2s 58ms/step - loss: 0.6707 - accuracy:
0.5875 - val_loss: 0.7116 - val_accuracy: 0.5450
Epoch 53/120
38/38 [==============================] - 2s 56ms/step - loss: 0.6723 - accuracy:
0.5896 - val_loss: 0.6905 - val_accuracy: 0.5612
Epoch 54/120
38/38 [==============================] - 2s 57ms/step - loss: 0.6770 - accuracy:
0.5767 - val_loss: 0.6988 - val_accuracy: 0.5612
Epoch 55/120
38/38 [==============================] - 2s 54ms/step - loss: 0.6693 - accuracy:
0.5933 - val_loss: 0.6952 - val_accuracy: 0.5525
Epoch 56/120
38/38 [==============================] - 2s 53ms/step - loss: 0.6722 - accuracy:
0.5917 - val_loss: 0.7114 - val_accuracy: 0.5838
Epoch 57/120
38/38 [==============================] - 2s 55ms/step - loss: 0.6693 - accuracy:
0.5817 - val_loss: 0.7044 - val_accuracy: 0.5800
Epoch 58/120
38/38 [==============================] - 2s 56ms/step - loss: 0.6701 - accuracy:
0.5921 - val_loss: 0.6929 - val_accuracy: 0.5275
Epoch 59/120
38/38 [==============================] - 2s 56ms/step - loss: 0.6760 - accuracy:
0.5738 - val_loss: 0.6963 - val_accuracy: 0.5625
Epoch 60/120
38/38 [==============================] - 2s 55ms/step - loss: 0.6663 - accuracy:
0.5925 - val_loss: 0.7099 - val_accuracy: 0.5663
Epoch 61/120
38/38 [==============================] - 2s 54ms/step - loss: 0.6777 - accuracy:
0.5767 - val_loss: 0.6983 - val_accuracy: 0.5525
Epoch 62/120
38/38 [==============================] - 2s 53ms/step - loss: 0.6673 - accuracy:
0.6042 - val_loss: 0.7046 - val_accuracy: 0.5100
Epoch 63/120
38/38 [==============================] - 2s 54ms/step - loss: 0.6734 - accuracy:
0.5825 - val_loss: 0.6942 - val_accuracy: 0.5525
Epoch 64/120
38/38 [==============================] - 2s 54ms/step - loss: 0.6683 - accuracy:
0.5921 - val_loss: 0.7072 - val_accuracy: 0.5225
Epoch 65/120
38/38 [==============================] - 2s 54ms/step - loss: 0.6701 - accuracy:
0.5846 - val_loss: 0.7100 - val_accuracy: 0.5663
Epoch 66/120
38/38 [==============================] - 2s 56ms/step - loss: 0.6724 - accuracy:
0.5950 - val_loss: 0.7043 - val_accuracy: 0.5100
Epoch 67/120
38/38 [==============================] - 2s 58ms/step - loss: 0.6719 - accuracy:
0.5950 - val_loss: 0.7003 - val_accuracy: 0.5487
Epoch 68/120
```

```
38/38 [==============================] - 2s 54ms/step - loss: 0.6661 - accuracy:
0.5942 - val_loss: 0.7047 - val_accuracy: 0.5487
Epoch 69/120
38/38 [==============================] - 2s 54ms/step - loss: 0.6658 - accuracy:
0.6037 - val_loss: 0.6992 - val_accuracy: 0.5350
Epoch 70/120
38/38 [==============================] - 2s 53ms/step - loss: 0.6683 - accuracy:
0.5962 - val_loss: 0.7189 - val_accuracy: 0.5300
Epoch 71/120
38/38 [==============================] - 2s 55ms/step - loss: 0.6708 - accuracy:
0.5933 - val_loss: 0.6967 - val_accuracy: 0.5387
Epoch 72/120
38/38 [==============================] - 2s 54ms/step - loss: 0.6694 - accuracy:
0.6025 - val_loss: 0.7004 - val_accuracy: 0.5412
Epoch 73/120
38/38 [==============================] - 2s 58ms/step - loss: 0.6665 - accuracy:
0.5987 - val_loss: 0.7072 - val_accuracy: 0.5288
Epoch 74/120
38/38 [==============================] - 2s 53ms/step - loss: 0.6673 - accuracy:
0.5942 - val_loss: 0.7035 - val_accuracy: 0.5475
Epoch 75/120
38/38 [==============================] - 2s 54ms/step - loss: 0.6644 - accuracy:
0.5992 - val_loss: 0.7099 - val_accuracy: 0.5525
Epoch 76/120
38/38 [==============================] - 2s 54ms/step - loss: 0.6739 - accuracy:
0.5896 - val_loss: 0.6928 - val_accuracy: 0.5500
Epoch 77/120
38/38 [==============================] - 2s 55ms/step - loss: 0.6605 - accuracy:
0.6183 - val_loss: 0.7057 - val_accuracy: 0.5487
Epoch 78/120
38/38 [==============================] - 2s 56ms/step - loss: 0.6622 - accuracy:
0.6075 - val_loss: 0.7087 - val_accuracy: 0.5462
Epoch 79/120
38/38 [==============================] - 2s 55ms/step - loss: 0.6622 - accuracy:
0.6104 - val_loss: 0.7073 - val_accuracy: 0.5213
Epoch 80/120
38/38 [==============================] - 2s 53ms/step - loss: 0.6650 - accuracy:
0.6033 - val_loss: 0.7038 - val_accuracy: 0.5150
Epoch 81/120
38/38 [==============================] - 2s 53ms/step - loss: 0.6691 - accuracy:
0.5983 - val_loss: 0.6996 - val_accuracy: 0.5350
Epoch 82/120
38/38 [==============================] - 2s 54ms/step - loss: 0.6660 - accuracy:
0.6004 - val_loss: 0.6945 - val_accuracy: 0.5500
Epoch 83/120
38/38 [==============================] - 2s 57ms/step - loss: 0.6622 - accuracy:
0.5954 - val_loss: 0.7009 - val_accuracy: 0.5450
Epoch 84/120
```

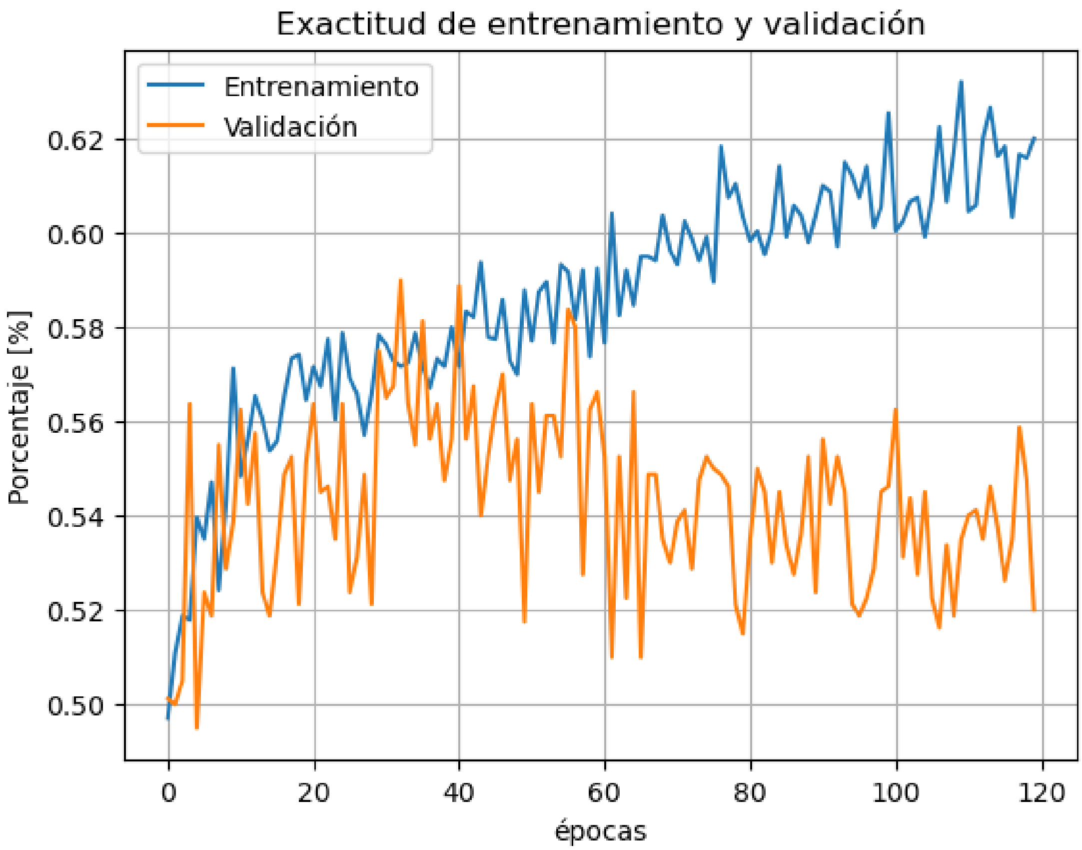
```
38/38 [==============================] - 2s 55ms/step - loss: 0.6659 - accuracy:
0.6008 - val_loss: 0.7003 - val_accuracy: 0.5300
Epoch 85/120
38/38 [==============================] - 2s 56ms/step - loss: 0.6605 - accuracy:
0.6142 - val_loss: 0.7067 - val_accuracy: 0.5450
Epoch 86/120
38/38 [==============================] - 2s 51ms/step - loss: 0.6648 - accuracy:
0.5992 - val_loss: 0.7026 - val_accuracy: 0.5337
Epoch 87/120
38/38 [==============================] - 2s 53ms/step - loss: 0.6621 - accuracy:
0.6058 - val_loss: 0.6970 - val_accuracy: 0.5275
Epoch 88/120
38/38 [==============================] - 2s 53ms/step - loss: 0.6651 - accuracy:
0.6037 - val_loss: 0.6980 - val_accuracy: 0.5362
Epoch 89/120
38/38 [==============================] - 2s 53ms/step - loss: 0.6626 - accuracy:
0.5979 - val_loss: 0.6940 - val_accuracy: 0.5525
Epoch 90/120
38/38 [==============================] - 2s 61ms/step - loss: 0.6575 - accuracy:
0.6037 - val_loss: 0.6990 - val_accuracy: 0.5238
Epoch 91/120
38/38 [==============================] - 2s 58ms/step - loss: 0.6624 - accuracy:
0.6100 - val_loss: 0.6967 - val_accuracy: 0.5562
Epoch 92/120
38/38 [==============================] - 2s 56ms/step - loss: 0.6579 - accuracy:
0.6087 - val_loss: 0.7133 - val_accuracy: 0.5425
Epoch 93/120
38/38 [==============================] - 2s 54ms/step - loss: 0.6590 - accuracy:
0.5971 - val_loss: 0.7172 - val_accuracy: 0.5525
Epoch 94/120
38/38 [==============================] - 2s 53ms/step - loss: 0.6604 - accuracy:
0.6150 - val_loss: 0.6990 - val_accuracy: 0.5450
Epoch 95/120
38/38 [==============================] - 2s 53ms/step - loss: 0.6606 - accuracy:
0.6121 - val_loss: 0.6991 - val_accuracy: 0.5213
Epoch 96/120
38/38 [==============================] - 2s 53ms/step - loss: 0.6580 - accuracy:
0.6075 - val_loss: 0.7305 - val_accuracy: 0.5188
Epoch 97/120
38/38 [==============================] - 2s 53ms/step - loss: 0.6616 - accuracy:
0.6142 - val_loss: 0.7003 - val_accuracy: 0.5225
Epoch 98/120
38/38 [==============================] - 2s 55ms/step - loss: 0.6638 - accuracy:
0.6012 - val_loss: 0.7004 - val_accuracy: 0.5288
Epoch 99/120
38/38 [==============================] - 2s 55ms/step - loss: 0.6618 - accuracy:
0.6054 - val_loss: 0.7102 - val_accuracy: 0.5450
Epoch 100/120
```

```
38/38 [==============================] - 2s 54ms/step - loss: 0.6587 - accuracy:
0.6254 - val_loss: 0.7139 - val_accuracy: 0.5462
Epoch 101/120
38/38 [==============================] - 2s 55ms/step - loss: 0.6651 - accuracy:
0.6004 - val_loss: 0.6976 - val_accuracy: 0.5625
Epoch 102/120
38/38 [==============================] - 2s 52ms/step - loss: 0.6627 - accuracy:
0.6025 - val_loss: 0.7050 - val_accuracy: 0.5312
Epoch 103/120
38/38 [==============================] - 2s 55ms/step - loss: 0.6564 - accuracy:
0.6067 - val_loss: 0.7095 - val_accuracy: 0.5437
Epoch 104/120
38/38 [==============================] - 2s 58ms/step - loss: 0.6648 - accuracy:
0.6075 - val_loss: 0.6970 - val_accuracy: 0.5275
Epoch 105/120
38/38 [==============================] - 2s 54ms/step - loss: 0.6594 - accuracy:
0.5992 - val_loss: 0.7038 - val_accuracy: 0.5450
Epoch 106/120
38/38 [==============================] - 2s 51ms/step - loss: 0.6537 - accuracy:
0.6075 - val_loss: 0.7199 - val_accuracy: 0.5225
Epoch 107/120
38/38 [==============================] - 2s 54ms/step - loss: 0.6534 - accuracy:
0.6225 - val_loss: 0.7063 - val_accuracy: 0.5163
Epoch 108/120
38/38 [==============================] - 2s 53ms/step - loss: 0.6531 - accuracy:
0.6067 - val_loss: 0.7124 - val_accuracy: 0.5337
Epoch 109/120
38/38 [==============================] - 2s 54ms/step - loss: 0.6488 - accuracy:
0.6175 - val_loss: 0.7274 - val_accuracy: 0.5188
Epoch 110/120
38/38 [==============================] - 2s 53ms/step - loss: 0.6556 - accuracy:
0.6321 - val_loss: 0.7154 - val_accuracy: 0.5350
Epoch 111/120
38/38 [==============================] - 2s 55ms/step - loss: 0.6609 - accuracy:
0.6046 - val_loss: 0.7061 - val_accuracy: 0.5400
Epoch 112/120
38/38 [==============================] - 2s 53ms/step - loss: 0.6588 - accuracy:
0.6058 - val_loss: 0.7130 - val_accuracy: 0.5412
Epoch 113/120
38/38 [==============================] - 2s 54ms/step - loss: 0.6598 - accuracy:
0.6200 - val_loss: 0.6980 - val_accuracy: 0.5350
Epoch 114/120
38/38 [==============================] - 2s 55ms/step - loss: 0.6531 - accuracy:
0.6267 - val_loss: 0.7021 - val_accuracy: 0.5462
Epoch 115/120
38/38 [==============================] - 2s 55ms/step - loss: 0.6543 - accuracy:
0.6162 - val_loss: 0.7089 - val_accuracy: 0.5375
Epoch 116/120
```

```
38/38 [==============================] - 2s 57ms/step - loss: 0.6545 - accuracy:
0.6183 - val_loss: 0.6989 - val_accuracy: 0.5263
Epoch 117/120
38/38 [==============================] - 2s 55ms/step - loss: 0.6607 - accuracy:
0.6033 - val_loss: 0.7087 - val_accuracy: 0.5350
Epoch 118/120
38/38 [==============================] - 2s 56ms/step - loss: 0.6580 - accuracy:
0.6167 - val_loss: 0.6956 - val_accuracy: 0.5587
Epoch 119/120
38/38 [==============================] - 2s 52ms/step - loss: 0.6539 - accuracy:
0.6158 - val_loss: 0.7353 - val_accuracy: 0.5475
Epoch 120/120
38/38 [==============================] - 2s 55ms/step - loss: 0.6504 - accuracy:
0.6200 - val_loss: 0.7456 - val_accuracy: 0.5200
```

[12]:

```
type(hist_3.history['loss'])
import matplotlib.pyplot as plt
plt.plot(hist_3.history['accuracy'])
plt.plot(hist_3.history['val_accuracy'])
plt.title('Exactitud de entrenamiento y validación')
plt.xlabel('épocas')
plt.ylabel('Porcentaje [%]')
plt.legend(['Entrenamiento','Validación'])
plt.grid()
```



[13]:
```
pred=model.predict(x_test)
pred=np.argmax(pred,axis=1)
y1=np.argmax(y_test,axis=1)

#label=np.argmax(yp_oh)
exactitud_test=0
for a in range(len(pred)):
    if pred[a]==y1[a]:
        exactitud_test+=1
print('exactitud de la prueba= ',100*exactitud_test/len(pred),'%')
```

```
25/25 [==============================] - 0s 1ms/step
exactitud de la prueba=  52.875 %
```

4 3) Regularización por BatchNormalization

[14]:
```
import tensorflow as tf
from keras.models import Model, load_model
```

```
#from tensorflow.keras.applications.resnet50 import preprocess_input,␣
 ↪decode_predictions
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Flatten,Dropout,Input
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping,␣
 ↪ReduceLROnPlateau

lr_reduce = ReduceLROnPlateau(monitor='val_accuracy', factor=0.6, patience=8,␣
 ↪verbose=1, mode='max', min_lr=5e-5)
checkpoint =  ModelCheckpoint('vgg16_finetune.h5', monitor= 'val_accuracy',␣
 ↪mode= 'max', save_best_only = True, verbose= 1)
earlystopper = EarlyStopping(monitor = 'val_loss', min_delta = 0, patience = 10,␣
 ↪verbose = 1, restore_best_weights = True)

#"""

#"""

model=Sequential()
model.add(tf.keras.layers.
 ↪Conv2D(6,(5,5),input_shape=(x_size,y_size,1),activation='tanh',padding='valid',strides=1))␣
 ↪#C1
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.AveragePooling2D(pool_size=(2,2))) #S2

model.add(tf.keras.layers.
 ↪Conv2D(16,(5,5),activation='tanh',padding='valid',strides=1)) #c3
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.AveragePooling2D(pool_size=(2,2))) #s4
model.add(tf.keras.layers.Flatten())

model.add(Dense(120,activation='tanh')) #c5
model.add(tf.keras.layers.BatchNormalization())
model.add(Dense(84,activation='tanh')) #c6

from keras.layers import Layer
from keras import backend as K

class RBFLayer(Layer):
    def __init__(self, units, gamma, **kwargs):
        super(RBFLayer, self).__init__(**kwargs)
        self.units = units
        self.gamma = K.cast_to_floatx(gamma)

    def build(self, input_shape):
        self.mu = self.add_weight(name='mu',
```

```
                                    shape=(int(input_shape[1]), self.units),
                                    initializer='uniform',
                                    trainable=True)
        super(RBFLayer, self).build(input_shape)

    def call(self, inputs):
        diff = K.expand_dims(inputs) - self.mu
        l2 = K.sum(K.pow(diff,2), axis=1)
        res = K.exp(-1 * self.gamma * l2)
        return res

    def compute_output_shape(self, input_shape):
        return (input_shape[0], self.units)

model.add(RBFLayer(2,0.5)) #c7

model.compile(loss='categorical_crossentropy',optimizer=tf.keras.optimizers.
 ↪SGD(learning_rate=0.25),metrics=['accuracy'])
hist_4=model.fit(x_train,y_train,verbose=1,␣
 ↪batch_size=64,epochs=80,validation_data=(x_val,y_val))
```

```
Epoch 1/80
38/38 [==============================] - 3s 56ms/step - loss: 0.7120 - accuracy:
0.5288 - val_loss: 0.7166 - val_accuracy: 0.5000
Epoch 2/80
38/38 [==============================] - 2s 46ms/step - loss: 0.6915 - accuracy:
0.5683 - val_loss: 0.7424 - val_accuracy: 0.5000
Epoch 3/80
38/38 [==============================] - 2s 51ms/step - loss: 0.6843 - accuracy:
0.5708 - val_loss: 0.7696 - val_accuracy: 0.5000
Epoch 4/80
38/38 [==============================] - 2s 55ms/step - loss: 0.6689 - accuracy:
0.5917 - val_loss: 0.6992 - val_accuracy: 0.5437
Epoch 5/80
38/38 [==============================] - 2s 58ms/step - loss: 0.6481 - accuracy:
0.6342 - val_loss: 0.7333 - val_accuracy: 0.5400
Epoch 6/80
38/38 [==============================] - 2s 49ms/step - loss: 0.6561 - accuracy:
0.6283 - val_loss: 0.7636 - val_accuracy: 0.5437
Epoch 7/80
38/38 [==============================] - 2s 61ms/step - loss: 0.6156 - accuracy:
0.6708 - val_loss: 0.8907 - val_accuracy: 0.5350
Epoch 8/80
38/38 [==============================] - 3s 67ms/step - loss: 0.6059 - accuracy:
0.6704 - val_loss: 0.9825 - val_accuracy: 0.5288
Epoch 9/80
38/38 [==============================] - 3s 71ms/step - loss: 0.5616 - accuracy:
0.7163 - val_loss: 0.8699 - val_accuracy: 0.5238
```

```
Epoch 10/80
38/38 [==============================] - 2s 60ms/step - loss: 0.5398 - accuracy:
0.7254 - val_loss: 0.9009 - val_accuracy: 0.5550
Epoch 11/80
38/38 [==============================] - 2s 64ms/step - loss: 0.5195 - accuracy:
0.7437 - val_loss: 0.9155 - val_accuracy: 0.5688
Epoch 12/80
38/38 [==============================] - 426s 12s/step - loss: 0.4524 -
accuracy: 0.7804 - val_loss: 1.0378 - val_accuracy: 0.5025
Epoch 13/80
38/38 [==============================] - 3s 76ms/step - loss: 0.4074 - accuracy:
0.8150 - val_loss: 1.0041 - val_accuracy: 0.5500
Epoch 14/80
38/38 [==============================] - 3s 71ms/step - loss: 0.3808 - accuracy:
0.8242 - val_loss: 1.0038 - val_accuracy: 0.5863
Epoch 15/80
38/38 [==============================] - 2s 59ms/step - loss: 0.2750 - accuracy:
0.8875 - val_loss: 0.9602 - val_accuracy: 0.5863
Epoch 16/80
38/38 [==============================] - 2s 61ms/step - loss: 0.2724 - accuracy:
0.8913 - val_loss: 1.3475 - val_accuracy: 0.5875
Epoch 17/80
38/38 [==============================] - 2s 52ms/step - loss: 0.2692 - accuracy:
0.8896 - val_loss: 1.1469 - val_accuracy: 0.5675
Epoch 18/80
38/38 [==============================] - 2s 55ms/step - loss: 0.2011 - accuracy:
0.9221 - val_loss: 1.1957 - val_accuracy: 0.5750
Epoch 19/80
38/38 [==============================] - 2s 58ms/step - loss: 0.1073 - accuracy:
0.9621 - val_loss: 1.5078 - val_accuracy: 0.5763
Epoch 20/80
38/38 [==============================] - 2s 55ms/step - loss: 0.1167 - accuracy:
0.9621 - val_loss: 1.3711 - val_accuracy: 0.5838
Epoch 21/80
38/38 [==============================] - 2s 49ms/step - loss: 0.0585 - accuracy:
0.9837 - val_loss: 1.3791 - val_accuracy: 0.5150
Epoch 22/80
38/38 [==============================] - 2s 49ms/step - loss: 0.0697 - accuracy:
0.9792 - val_loss: 1.6888 - val_accuracy: 0.5700
Epoch 23/80
38/38 [==============================] - 2s 44ms/step - loss: 0.0454 - accuracy:
0.9883 - val_loss: 1.5643 - val_accuracy: 0.5813
Epoch 24/80
38/38 [==============================] - 2s 47ms/step - loss: 0.0159 - accuracy:
0.9996 - val_loss: 1.4825 - val_accuracy: 0.6050
Epoch 25/80
38/38 [==============================] - 2s 47ms/step - loss: 0.0099 - accuracy:
1.0000 - val_loss: 1.4487 - val_accuracy: 0.6137
```

```
Epoch 26/80
38/38 [==============================] - 2s 45ms/step - loss: 0.0069 - accuracy:
0.9996 - val_loss: 1.5791 - val_accuracy: 0.5975
Epoch 27/80
38/38 [==============================] - 2s 46ms/step - loss: 0.0058 - accuracy:
0.9996 - val_loss: 1.5545 - val_accuracy: 0.6000
Epoch 28/80
38/38 [==============================] - 2s 46ms/step - loss: 0.0041 - accuracy:
1.0000 - val_loss: 1.5737 - val_accuracy: 0.6075
Epoch 29/80
38/38 [==============================] - 2s 46ms/step - loss: 0.0036 - accuracy:
1.0000 - val_loss: 1.5825 - val_accuracy: 0.6100
Epoch 30/80
38/38 [==============================] - 2s 46ms/step - loss: 0.0035 - accuracy:
1.0000 - val_loss: 1.6074 - val_accuracy: 0.5938
Epoch 31/80
38/38 [==============================] - 2s 52ms/step - loss: 0.0026 - accuracy:
1.0000 - val_loss: 1.6409 - val_accuracy: 0.6162
Epoch 32/80
38/38 [==============================] - 2s 48ms/step - loss: 0.0023 - accuracy:
1.0000 - val_loss: 1.6501 - val_accuracy: 0.6125
Epoch 33/80
38/38 [==============================] - 2s 46ms/step - loss: 0.0024 - accuracy:
1.0000 - val_loss: 1.6706 - val_accuracy: 0.6050
Epoch 34/80
38/38 [==============================] - 2s 45ms/step - loss: 0.0021 - accuracy:
1.0000 - val_loss: 1.6834 - val_accuracy: 0.6037
Epoch 35/80
38/38 [==============================] - 2s 51ms/step - loss: 0.0019 - accuracy:
1.0000 - val_loss: 1.7002 - val_accuracy: 0.6112
Epoch 36/80
38/38 [==============================] - 2s 47ms/step - loss: 0.0025 - accuracy:
0.9996 - val_loss: 1.9191 - val_accuracy: 0.5788
Epoch 37/80
38/38 [==============================] - 2s 52ms/step - loss: 0.0077 - accuracy:
0.9996 - val_loss: 1.7785 - val_accuracy: 0.6100
Epoch 38/80
38/38 [==============================] - 2s 50ms/step - loss: 0.0036 - accuracy:
0.9996 - val_loss: 1.8215 - val_accuracy: 0.5913
Epoch 39/80
38/38 [==============================] - 2s 50ms/step - loss: 0.0024 - accuracy:
1.0000 - val_loss: 1.8219 - val_accuracy: 0.5813
Epoch 40/80
38/38 [==============================] - 2s 49ms/step - loss: 0.0019 - accuracy:
1.0000 - val_loss: 1.8098 - val_accuracy: 0.5938
Epoch 41/80
38/38 [==============================] - 2s 50ms/step - loss: 0.0019 - accuracy:
1.0000 - val_loss: 1.8260 - val_accuracy: 0.5962
```

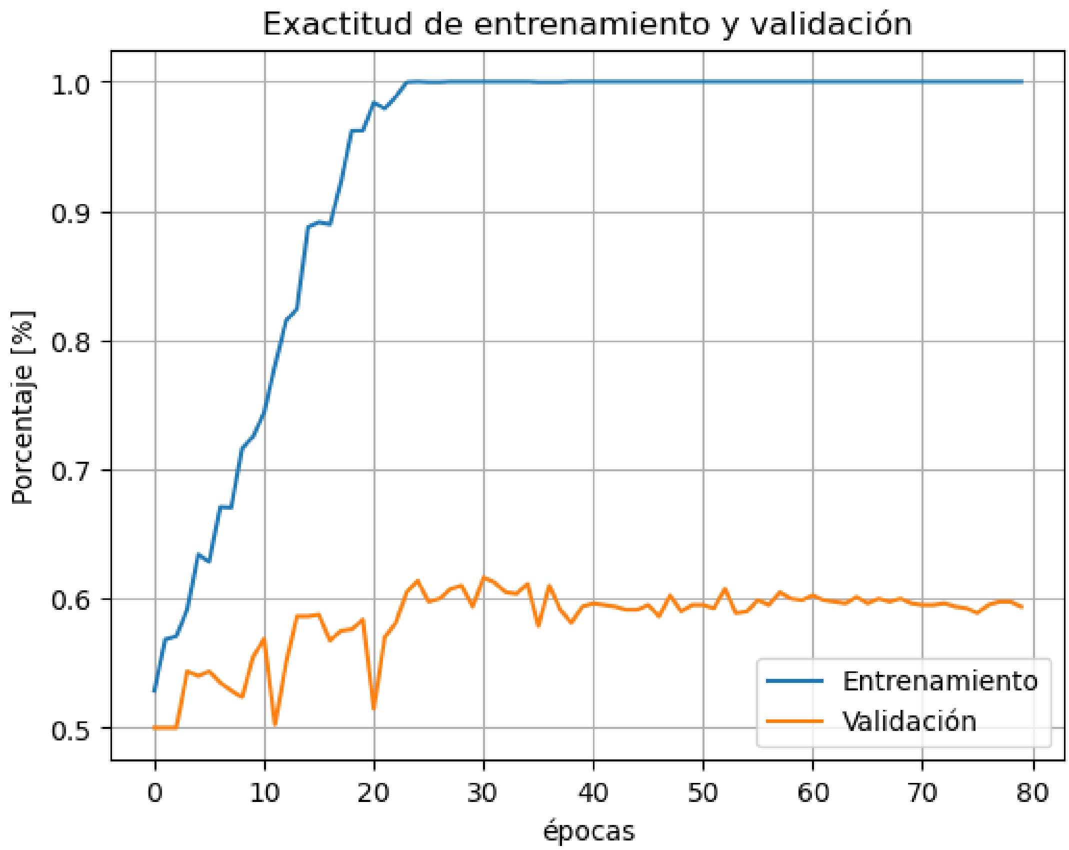
```
Epoch 42/80
38/38 [==============================] - 2s 48ms/step - loss: 0.0013 - accuracy:
1.0000 - val_loss: 1.8139 - val_accuracy: 0.5950
Epoch 43/80
38/38 [==============================] - 2s 49ms/step - loss: 0.0014 - accuracy:
1.0000 - val_loss: 1.8372 - val_accuracy: 0.5938
Epoch 44/80
38/38 [==============================] - 2s 61ms/step - loss: 0.0012 - accuracy:
1.0000 - val_loss: 1.8522 - val_accuracy: 0.5913
Epoch 45/80
38/38 [==============================] - 2s 48ms/step - loss: 0.0017 - accuracy:
1.0000 - val_loss: 1.8719 - val_accuracy: 0.5913
Epoch 46/80
38/38 [==============================] - 2s 51ms/step - loss: 0.0010 - accuracy:
1.0000 - val_loss: 1.8726 - val_accuracy: 0.5950
Epoch 47/80
38/38 [==============================] - 2s 57ms/step - loss: 0.0013 - accuracy:
1.0000 - val_loss: 1.9762 - val_accuracy: 0.5863
Epoch 48/80
38/38 [==============================] - 2s 54ms/step - loss: 0.0016 - accuracy:
1.0000 - val_loss: 1.8778 - val_accuracy: 0.6025
Epoch 49/80
38/38 [==============================] - 2s 50ms/step - loss: 9.5201e-04 -
accuracy: 1.0000 - val_loss: 1.8924 - val_accuracy: 0.5900
Epoch 50/80
38/38 [==============================] - 2s 50ms/step - loss: 9.5213e-04 -
accuracy: 1.0000 - val_loss: 1.9279 - val_accuracy: 0.5950
Epoch 51/80
38/38 [==============================] - 2s 47ms/step - loss: 8.7523e-04 -
accuracy: 1.0000 - val_loss: 1.9240 - val_accuracy: 0.5950
Epoch 52/80
38/38 [==============================] - 2s 49ms/step - loss: 8.2130e-04 -
accuracy: 1.0000 - val_loss: 1.9334 - val_accuracy: 0.5925
Epoch 53/80
38/38 [==============================] - 2s 50ms/step - loss: 8.4788e-04 -
accuracy: 1.0000 - val_loss: 1.9385 - val_accuracy: 0.6075
Epoch 54/80
38/38 [==============================] - 2s 47ms/step - loss: 0.0015 - accuracy:
1.0000 - val_loss: 1.9257 - val_accuracy: 0.5888
Epoch 55/80
38/38 [==============================] - 2s 48ms/step - loss: 9.2246e-04 -
accuracy: 1.0000 - val_loss: 1.8946 - val_accuracy: 0.5900
Epoch 56/80
38/38 [==============================] - 2s 48ms/step - loss: 7.3352e-04 -
accuracy: 1.0000 - val_loss: 1.9038 - val_accuracy: 0.5987
Epoch 57/80
38/38 [==============================] - 2s 52ms/step - loss: 6.9166e-04 -
accuracy: 1.0000 - val_loss: 1.9016 - val_accuracy: 0.5950
```

```
Epoch 58/80
38/38 [==============================] - 2s 48ms/step - loss: 8.1271e-04 -
accuracy: 1.0000 - val_loss: 1.9278 - val_accuracy: 0.6050
Epoch 59/80
38/38 [==============================] - 2s 50ms/step - loss: 6.1350e-04 -
accuracy: 1.0000 - val_loss: 1.9234 - val_accuracy: 0.6000
Epoch 60/80
38/38 [==============================] - 2s 49ms/step - loss: 5.1931e-04 -
accuracy: 1.0000 - val_loss: 1.9381 - val_accuracy: 0.5987
Epoch 61/80
38/38 [==============================] - 2s 46ms/step - loss: 6.1753e-04 -
accuracy: 1.0000 - val_loss: 1.9629 - val_accuracy: 0.6025
Epoch 62/80
38/38 [==============================] - 2s 46ms/step - loss: 5.3625e-04 -
accuracy: 1.0000 - val_loss: 1.9713 - val_accuracy: 0.5987
Epoch 63/80
38/38 [==============================] - 2s 47ms/step - loss: 6.3513e-04 -
accuracy: 1.0000 - val_loss: 1.9782 - val_accuracy: 0.5975
Epoch 64/80
38/38 [==============================] - 2s 46ms/step - loss: 6.2630e-04 -
accuracy: 1.0000 - val_loss: 1.9958 - val_accuracy: 0.5962
Epoch 65/80
38/38 [==============================] - 2s 48ms/step - loss: 6.1119e-04 -
accuracy: 1.0000 - val_loss: 1.9905 - val_accuracy: 0.6012
Epoch 66/80
38/38 [==============================] - 2s 44ms/step - loss: 5.8403e-04 -
accuracy: 1.0000 - val_loss: 1.9949 - val_accuracy: 0.5962
Epoch 67/80
38/38 [==============================] - 2s 48ms/step - loss: 5.5764e-04 -
accuracy: 1.0000 - val_loss: 1.9995 - val_accuracy: 0.6000
Epoch 68/80
38/38 [==============================] - 2s 42ms/step - loss: 7.3648e-04 -
accuracy: 1.0000 - val_loss: 1.9913 - val_accuracy: 0.5975
Epoch 69/80
38/38 [==============================] - 2s 46ms/step - loss: 6.4903e-04 -
accuracy: 1.0000 - val_loss: 1.9932 - val_accuracy: 0.6000
Epoch 70/80
38/38 [==============================] - 2s 48ms/step - loss: 4.8835e-04 -
accuracy: 1.0000 - val_loss: 2.0138 - val_accuracy: 0.5962
Epoch 71/80
38/38 [==============================] - 2s 45ms/step - loss: 4.0495e-04 -
accuracy: 1.0000 - val_loss: 2.0220 - val_accuracy: 0.5950
Epoch 72/80
38/38 [==============================] - 2s 48ms/step - loss: 7.0355e-04 -
accuracy: 1.0000 - val_loss: 2.0194 - val_accuracy: 0.5950
Epoch 73/80
38/38 [==============================] - 2s 50ms/step - loss: 4.1989e-04 -
accuracy: 1.0000 - val_loss: 2.0188 - val_accuracy: 0.5962
```

```
Epoch 74/80
38/38 [==============================] - 2s 47ms/step - loss: 4.9727e-04 -
accuracy: 1.0000 - val_loss: 2.0216 - val_accuracy: 0.5938
Epoch 75/80
38/38 [==============================] - 2s 51ms/step - loss: 4.2023e-04 -
accuracy: 1.0000 - val_loss: 2.0304 - val_accuracy: 0.5925
Epoch 76/80
38/38 [==============================] - 2s 51ms/step - loss: 5.1811e-04 -
accuracy: 1.0000 - val_loss: 2.0789 - val_accuracy: 0.5888
Epoch 77/80
38/38 [==============================] - 2s 51ms/step - loss: 5.3157e-04 -
accuracy: 1.0000 - val_loss: 2.0467 - val_accuracy: 0.5950
Epoch 78/80
38/38 [==============================] - 2s 46ms/step - loss: 4.1858e-04 -
accuracy: 1.0000 - val_loss: 2.0541 - val_accuracy: 0.5975
Epoch 79/80
38/38 [==============================] - 2s 52ms/step - loss: 4.2673e-04 -
accuracy: 1.0000 - val_loss: 2.0540 - val_accuracy: 0.5975
Epoch 80/80
38/38 [==============================] - 2s 46ms/step - loss: 3.3516e-04 -
accuracy: 1.0000 - val_loss: 2.0545 - val_accuracy: 0.5938
```

[15]:

```
type(hist_4.history['loss'])
import matplotlib.pyplot as plt
plt.plot(hist_4.history['accuracy'])
plt.plot(hist_4.history['val_accuracy'])
plt.title('Exactitud de entrenamiento y validación')
plt.xlabel('épocas')
plt.ylabel('Porcentaje [%]')
plt.legend(['Entrenamiento','Validación'])
plt.grid()
```



[16]:
```python
pred=model.predict(x_test)
pred=np.argmax(pred,axis=1)
y1=np.argmax(y_test,axis=1)

#label=np.argmax(yp_oh)
exactitud_test=0
for a in range(len(pred)):
    if pred[a]==y1[a]:
        exactitud_test+=1
print('exactitud de la prueba= ',100*exactitud_test/len(pred),'%')
```

```
25/25 [==============================] - 0s 2ms/step
exactitud de la prueba=  59.625 %
```

5 4) Regularización por data augmentation

[17]:
```python
import tensorflow as tf
from keras.models import Model, load_model
```

```
#from tensorflow.keras.applications.resnet50 import preprocess_input,␣
↪decode_predictions
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Flatten,Dropout,Input
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping,␣
↪ReduceLROnPlateau

lr_reduce = ReduceLROnPlateau(monitor='val_accuracy', factor=0.6, patience=8,␣
↪verbose=1, mode='max', min_lr=5e-5)
checkpoint =  ModelCheckpoint('vgg16_finetune.h5', monitor= 'val_accuracy',␣
↪mode= 'max', save_best_only = True, verbose= 1)
earlystopper = EarlyStopping(monitor = 'val_loss', min_delta = 0, patience = 10,␣
↪verbose = 1, restore_best_weights = True)

#"""

#"""
## Uso de RandomCrop
#tf.keras.layers.experimental.preprocessing.RandomCrop()
#tf.keras.layers.CenterCrop(x_size,y_size,1)
tf.keras.layers.RandomCrop(x_size,y_size,1)

model=Sequential()
model.add(tf.keras.layers.
↪Conv2D(6,(5,5),input_shape=(x_size,y_size,1),activation='tanh',padding='valid',strides=1))␣
↪#C1

model.add(tf.keras.layers.AveragePooling2D(pool_size=(2,2))) #S2

model.add(tf.keras.layers.
↪Conv2D(16,(5,5),activation='tanh',padding='valid',strides=1)) #c3

model.add(tf.keras.layers.AveragePooling2D(pool_size=(2,2))) #s4
model.add(tf.keras.layers.Flatten())

model.add(Dense(120,activation='tanh')) #c5
model.add(Dense(84,activation='tanh')) #c6

from keras.layers import Layer
from keras import backend as K

class RBFLayer(Layer):
    def __init__(self, units, gamma, **kwargs):
        super(RBFLayer, self).__init__(**kwargs)
        self.units = units
        self.gamma = K.cast_to_floatx(gamma)
```

```
    def build(self, input_shape):
        self.mu = self.add_weight(name='mu',
                                  shape=(int(input_shape[1]), self.units),
                                  initializer='uniform',
                                  trainable=True)
        super(RBFLayer, self).build(input_shape)

    def call(self, inputs):
        diff = K.expand_dims(inputs) - self.mu
        l2 = K.sum(K.pow(diff,2), axis=1)
        res = K.exp(-1 * self.gamma * l2)
        return res

    def compute_output_shape(self, input_shape):
        return (input_shape[0], self.units)

model.add(RBFLayer(2,0.5)) #c7

model.compile(loss='categorical_crossentropy',optimizer=tf.keras.optimizers.
 ↪SGD(learning_rate=0.25),metrics=['accuracy'])
hist_5=model.fit(x_train,y_train,verbose=1,␣
 ↪batch_size=64,epochs=80,validation_data=(x_val,y_val))
```

```
Epoch 1/80
38/38 [==============================] - 3s 62ms/step - loss: 0.6938 - accuracy:
0.5096 - val_loss: 0.6918 - val_accuracy: 0.5350
Epoch 2/80
38/38 [==============================] - 2s 53ms/step - loss: 0.6907 - accuracy:
0.5412 - val_loss: 0.6905 - val_accuracy: 0.5163
Epoch 3/80
38/38 [==============================] - 2s 57ms/step - loss: 0.6881 - accuracy:
0.5446 - val_loss: 0.6869 - val_accuracy: 0.5600
Epoch 4/80
38/38 [==============================] - 2s 56ms/step - loss: 0.6845 - accuracy:
0.5542 - val_loss: 0.6850 - val_accuracy: 0.5450
Epoch 5/80
38/38 [==============================] - 2s 51ms/step - loss: 0.6821 - accuracy:
0.5546 - val_loss: 0.6993 - val_accuracy: 0.5437
Epoch 6/80
38/38 [==============================] - 2s 56ms/step - loss: 0.6806 - accuracy:
0.5675 - val_loss: 0.6945 - val_accuracy: 0.5425
Epoch 7/80
38/38 [==============================] - 2s 54ms/step - loss: 0.6804 - accuracy:
0.5571 - val_loss: 0.6897 - val_accuracy: 0.5300
Epoch 8/80
38/38 [==============================] - 2s 56ms/step - loss: 0.6771 - accuracy:
0.5775 - val_loss: 0.7113 - val_accuracy: 0.5225
```

```
Epoch 9/80
38/38 [==============================] - 2s 55ms/step - loss: 0.6795 - accuracy:
0.5546 - val_loss: 0.6963 - val_accuracy: 0.5325
Epoch 10/80
38/38 [==============================] - 2s 53ms/step - loss: 0.6764 - accuracy:
0.5604 - val_loss: 0.6939 - val_accuracy: 0.5375
Epoch 11/80
38/38 [==============================] - 3s 68ms/step - loss: 0.6756 - accuracy:
0.5683 - val_loss: 0.6917 - val_accuracy: 0.5175
Epoch 12/80
38/38 [==============================] - 2s 51ms/step - loss: 0.6753 - accuracy:
0.5817 - val_loss: 0.6889 - val_accuracy: 0.5738
Epoch 13/80
38/38 [==============================] - 2s 50ms/step - loss: 0.6734 - accuracy:
0.5767 - val_loss: 0.6898 - val_accuracy: 0.5625
Epoch 14/80
38/38 [==============================] - 2s 54ms/step - loss: 0.6732 - accuracy:
0.5838 - val_loss: 0.6952 - val_accuracy: 0.5337
Epoch 15/80
38/38 [==============================] - 2s 55ms/step - loss: 0.6709 - accuracy:
0.5875 - val_loss: 0.7017 - val_accuracy: 0.5250
Epoch 16/80
38/38 [==============================] - 2s 53ms/step - loss: 0.6704 - accuracy:
0.5921 - val_loss: 0.6949 - val_accuracy: 0.5575
Epoch 17/80
38/38 [==============================] - 2s 52ms/step - loss: 0.6677 - accuracy:
0.5888 - val_loss: 0.6916 - val_accuracy: 0.5638
Epoch 18/80
38/38 [==============================] - 2s 55ms/step - loss: 0.6679 - accuracy:
0.5842 - val_loss: 0.7011 - val_accuracy: 0.5437
Epoch 19/80
38/38 [==============================] - 2s 51ms/step - loss: 0.6663 - accuracy:
0.5838 - val_loss: 0.6956 - val_accuracy: 0.5425
Epoch 20/80
38/38 [==============================] - 2s 55ms/step - loss: 0.6647 - accuracy:
0.5925 - val_loss: 0.7027 - val_accuracy: 0.5400
Epoch 21/80
38/38 [==============================] - 2s 53ms/step - loss: 0.6654 - accuracy:
0.5904 - val_loss: 0.6919 - val_accuracy: 0.5962
Epoch 22/80
38/38 [==============================] - 2s 50ms/step - loss: 0.6595 - accuracy:
0.5958 - val_loss: 0.6889 - val_accuracy: 0.5462
Epoch 23/80
38/38 [==============================] - 2s 53ms/step - loss: 0.6578 - accuracy:
0.5950 - val_loss: 0.6840 - val_accuracy: 0.5650
Epoch 24/80
38/38 [==============================] - 2s 56ms/step - loss: 0.6573 - accuracy:
0.6079 - val_loss: 0.6867 - val_accuracy: 0.5788
```

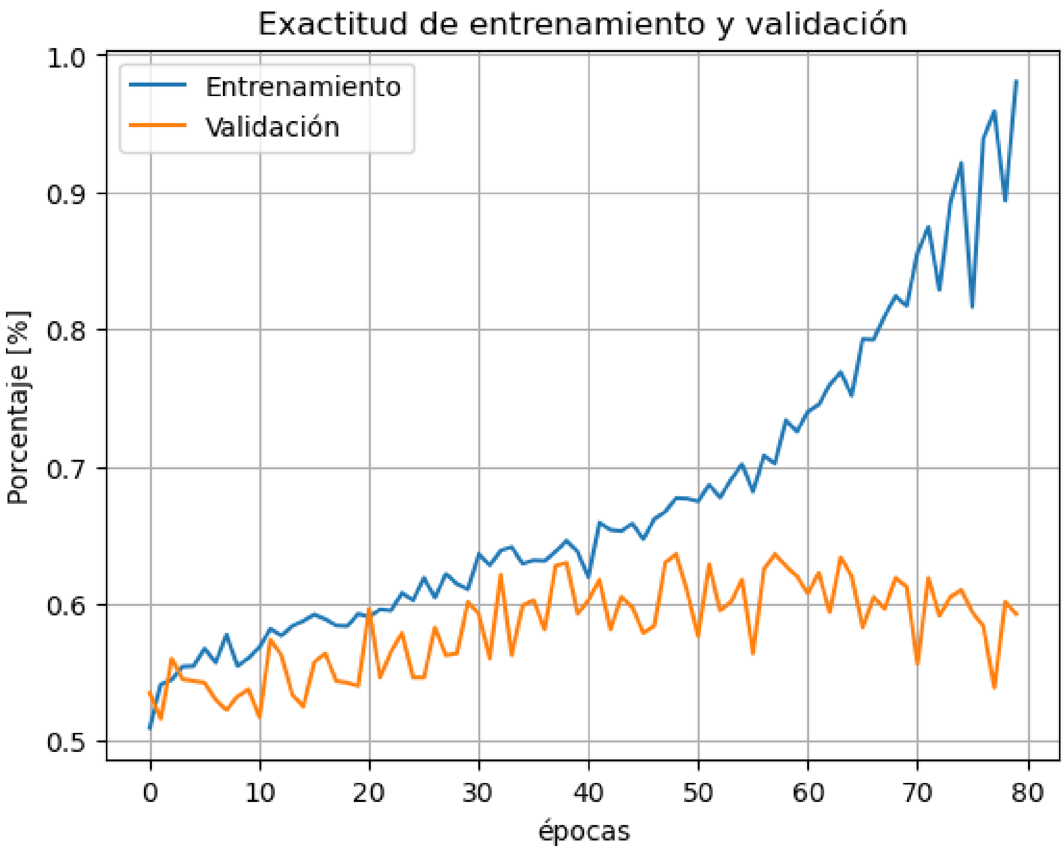
```
Epoch 25/80
38/38 [==============================] - 2s 51ms/step - loss: 0.6529 - accuracy:
0.6025 - val_loss: 0.7047 - val_accuracy: 0.5462
Epoch 26/80
38/38 [==============================] - 2s 55ms/step - loss: 0.6494 - accuracy:
0.6187 - val_loss: 0.6906 - val_accuracy: 0.5462
Epoch 27/80
38/38 [==============================] - 2s 51ms/step - loss: 0.6519 - accuracy:
0.6046 - val_loss: 0.6789 - val_accuracy: 0.5825
Epoch 28/80
38/38 [==============================] - 2s 53ms/step - loss: 0.6423 - accuracy:
0.6217 - val_loss: 0.6895 - val_accuracy: 0.5625
Epoch 29/80
38/38 [==============================] - 2s 53ms/step - loss: 0.6428 - accuracy:
0.6146 - val_loss: 0.6882 - val_accuracy: 0.5638
Epoch 30/80
38/38 [==============================] - 2s 53ms/step - loss: 0.6520 - accuracy:
0.6104 - val_loss: 0.6692 - val_accuracy: 0.6012
Epoch 31/80
38/38 [==============================] - 2s 58ms/step - loss: 0.6384 - accuracy:
0.6363 - val_loss: 0.6714 - val_accuracy: 0.5925
Epoch 32/80
38/38 [==============================] - 2s 55ms/step - loss: 0.6380 - accuracy:
0.6279 - val_loss: 0.7016 - val_accuracy: 0.5600
Epoch 33/80
38/38 [==============================] - 2s 56ms/step - loss: 0.6354 - accuracy:
0.6388 - val_loss: 0.6633 - val_accuracy: 0.6212
Epoch 34/80
38/38 [==============================] - 2s 56ms/step - loss: 0.6336 - accuracy:
0.6413 - val_loss: 0.7096 - val_accuracy: 0.5625
Epoch 35/80
38/38 [==============================] - 2s 55ms/step - loss: 0.6378 - accuracy:
0.6292 - val_loss: 0.6635 - val_accuracy: 0.5987
Epoch 36/80
38/38 [==============================] - 2s 52ms/step - loss: 0.6322 - accuracy:
0.6317 - val_loss: 0.6760 - val_accuracy: 0.6025
Epoch 37/80
38/38 [==============================] - 2s 46ms/step - loss: 0.6317 - accuracy:
0.6313 - val_loss: 0.6810 - val_accuracy: 0.5813
Epoch 38/80
38/38 [==============================] - 2s 54ms/step - loss: 0.6252 - accuracy:
0.6379 - val_loss: 0.6672 - val_accuracy: 0.6275
Epoch 39/80
38/38 [==============================] - 2s 55ms/step - loss: 0.6305 - accuracy:
0.6458 - val_loss: 0.6618 - val_accuracy: 0.6300
Epoch 40/80
38/38 [==============================] - 2s 57ms/step - loss: 0.6220 - accuracy:
0.6379 - val_loss: 0.6745 - val_accuracy: 0.5925
```

```
Epoch 41/80
38/38 [==============================] - 2s 55ms/step - loss: 0.6655 - accuracy:
0.6192 - val_loss: 0.6639 - val_accuracy: 0.6025
Epoch 42/80
38/38 [==============================] - 2s 53ms/step - loss: 0.6174 - accuracy:
0.6592 - val_loss: 0.6691 - val_accuracy: 0.6175
Epoch 43/80
38/38 [==============================] - 2s 56ms/step - loss: 0.6195 - accuracy:
0.6538 - val_loss: 0.6966 - val_accuracy: 0.5813
Epoch 44/80
38/38 [==============================] - 2s 51ms/step - loss: 0.6135 - accuracy:
0.6529 - val_loss: 0.6748 - val_accuracy: 0.6050
Epoch 45/80
38/38 [==============================] - 2s 50ms/step - loss: 0.6083 - accuracy:
0.6583 - val_loss: 0.6752 - val_accuracy: 0.5975
Epoch 46/80
38/38 [==============================] - 2s 54ms/step - loss: 0.6263 - accuracy:
0.6471 - val_loss: 0.6898 - val_accuracy: 0.5788
Epoch 47/80
38/38 [==============================] - 2s 54ms/step - loss: 0.6078 - accuracy:
0.6617 - val_loss: 0.6754 - val_accuracy: 0.5838
Epoch 48/80
38/38 [==============================] - 2s 56ms/step - loss: 0.6027 - accuracy:
0.6671 - val_loss: 0.6636 - val_accuracy: 0.6300
Epoch 49/80
38/38 [==============================] - 2s 55ms/step - loss: 0.6015 - accuracy:
0.6771 - val_loss: 0.6460 - val_accuracy: 0.6363
Epoch 50/80
38/38 [==============================] - 2s 54ms/step - loss: 0.5985 - accuracy:
0.6767 - val_loss: 0.6630 - val_accuracy: 0.6087
Epoch 51/80
38/38 [==============================] - 2s 53ms/step - loss: 0.5938 - accuracy:
0.6746 - val_loss: 0.7744 - val_accuracy: 0.5763
Epoch 52/80
38/38 [==============================] - 2s 52ms/step - loss: 0.5847 - accuracy:
0.6867 - val_loss: 0.6661 - val_accuracy: 0.6288
Epoch 53/80
38/38 [==============================] - 2s 52ms/step - loss: 0.5873 - accuracy:
0.6775 - val_loss: 0.6812 - val_accuracy: 0.5950
Epoch 54/80
38/38 [==============================] - 2s 62ms/step - loss: 0.5772 - accuracy:
0.6904 - val_loss: 0.7030 - val_accuracy: 0.6012
Epoch 55/80
38/38 [==============================] - 2s 55ms/step - loss: 0.5712 - accuracy:
0.7017 - val_loss: 0.6994 - val_accuracy: 0.6175
Epoch 56/80
38/38 [==============================] - 2s 57ms/step - loss: 0.5905 - accuracy:
0.6817 - val_loss: 0.7332 - val_accuracy: 0.5638
```

```
Epoch 57/80
38/38 [==============================] - 2s 53ms/step - loss: 0.5610 - accuracy:
0.7079 - val_loss: 0.7062 - val_accuracy: 0.6250
Epoch 58/80
38/38 [==============================] - 2s 56ms/step - loss: 0.5668 - accuracy:
0.7021 - val_loss: 0.6963 - val_accuracy: 0.6363
Epoch 59/80
38/38 [==============================] - 2s 51ms/step - loss: 0.5307 - accuracy:
0.7337 - val_loss: 0.6935 - val_accuracy: 0.6275
Epoch 60/80
38/38 [==============================] - 2s 55ms/step - loss: 0.5283 - accuracy:
0.7254 - val_loss: 0.7741 - val_accuracy: 0.6200
Epoch 61/80
38/38 [==============================] - 2s 54ms/step - loss: 0.5223 - accuracy:
0.7400 - val_loss: 0.7360 - val_accuracy: 0.6075
Epoch 62/80
38/38 [==============================] - 2s 54ms/step - loss: 0.5066 - accuracy:
0.7450 - val_loss: 0.7299 - val_accuracy: 0.6225
Epoch 63/80
38/38 [==============================] - 2s 55ms/step - loss: 0.4830 - accuracy:
0.7596 - val_loss: 0.8021 - val_accuracy: 0.5938
Epoch 64/80
38/38 [==============================] - 2s 54ms/step - loss: 0.4857 - accuracy:
0.7688 - val_loss: 0.7543 - val_accuracy: 0.6338
Epoch 65/80
38/38 [==============================] - 2s 60ms/step - loss: 0.4927 - accuracy:
0.7517 - val_loss: 0.7544 - val_accuracy: 0.6200
Epoch 66/80
38/38 [==============================] - 2s 54ms/step - loss: 0.4466 - accuracy:
0.7929 - val_loss: 0.8374 - val_accuracy: 0.5825
Epoch 67/80
38/38 [==============================] - 2s 53ms/step - loss: 0.4452 - accuracy:
0.7925 - val_loss: 0.8420 - val_accuracy: 0.6050
Epoch 68/80
38/38 [==============================] - 2s 54ms/step - loss: 0.4153 - accuracy:
0.8096 - val_loss: 0.9198 - val_accuracy: 0.5962
Epoch 69/80
38/38 [==============================] - 2s 54ms/step - loss: 0.3949 - accuracy:
0.8242 - val_loss: 0.8238 - val_accuracy: 0.6187
Epoch 70/80
38/38 [==============================] - 2s 53ms/step - loss: 0.3889 - accuracy:
0.8171 - val_loss: 0.8654 - val_accuracy: 0.6125
Epoch 71/80
38/38 [==============================] - 2s 53ms/step - loss: 0.3316 - accuracy:
0.8558 - val_loss: 1.0697 - val_accuracy: 0.5562
Epoch 72/80
38/38 [==============================] - 2s 55ms/step - loss: 0.3125 - accuracy:
0.8746 - val_loss: 0.9811 - val_accuracy: 0.6187
```

```
Epoch 73/80
38/38 [==============================] - 2s 55ms/step - loss: 0.4231 - accuracy:
0.8288 - val_loss: 0.9229 - val_accuracy: 0.5913
Epoch 74/80
38/38 [==============================] - 2s 56ms/step - loss: 0.2645 - accuracy:
0.8921 - val_loss: 0.9998 - val_accuracy: 0.6050
Epoch 75/80
38/38 [==============================] - 2s 59ms/step - loss: 0.2208 - accuracy:
0.9212 - val_loss: 1.1522 - val_accuracy: 0.6100
Epoch 76/80
38/38 [==============================] - 2s 54ms/step - loss: 0.4514 - accuracy:
0.8163 - val_loss: 1.0370 - val_accuracy: 0.5938
Epoch 77/80
38/38 [==============================] - 2s 50ms/step - loss: 0.1803 - accuracy:
0.9388 - val_loss: 1.1654 - val_accuracy: 0.5838
Epoch 78/80
38/38 [==============================] - 2s 50ms/step - loss: 0.1375 - accuracy:
0.9588 - val_loss: 1.5540 - val_accuracy: 0.5387
Epoch 79/80
38/38 [==============================] - 2s 54ms/step - loss: 0.2944 - accuracy:
0.8938 - val_loss: 1.1374 - val_accuracy: 0.6012
Epoch 80/80
38/38 [==============================] - 2s 54ms/step - loss: 0.0898 - accuracy:
0.9804 - val_loss: 1.1905 - val_accuracy: 0.5925
```

[18]:
```
type(hist_5.history['loss'])
import matplotlib.pyplot as plt
plt.plot(hist_5.history['accuracy'])
plt.plot(hist_5.history['val_accuracy'])
plt.title('Exactitud de entrenamiento y validación')
plt.xlabel('épocas')
plt.ylabel('Porcentaje [%]')
plt.legend(['Entrenamiento','Validación'])
plt.grid()
```



[19]:
```
pred=model.predict(x_test)
pred=np.argmax(pred,axis=1)
y1=np.argmax(y_test,axis=1)

#label=np.argmax(yp_oh)
exactitud_test=0
for a in range(len(pred)):
    if pred[a]==y1[a]:
        exactitud_test+=1
print('exactitud de la prueba= ',100*exactitud_test/len(pred),'%')
```

```
25/25 [==============================] - 0s 1ms/step
exactitud de la prueba=  58.75 %
```

6 5) Regularización con todos los métodos

- **L1 o norma L1 Lasso**
- **Dropout**
- **Normalización por lotes (BatchNormalization)**

- **Aumento de los datos (Data augmentation)**

Extra: También se usarán los Callbacks de reducción de factor de aprendizaje y el "EarlyStopper".

[14]:
```
import tensorflow as tf
from keras.models import Model, load_model
#from tensorflow.keras.applications.resnet50 import preprocess_input,␣
↪decode_predictions
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Flatten,Dropout,Input
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping,␣
↪ReduceLROnPlateau

lr_reduce = ReduceLROnPlateau(monitor='val_accuracy', factor=0.6, patience=8,␣
↪verbose=1, mode='max', min_lr=5e-5)
#checkpoint =  ModelCheckpoint('vgg16_finetune.h5', monitor= 'val_accuracy',␣
↪mode= 'max', save_best_only = True, verbose= 1)
earlystopper = EarlyStopping(monitor = 'val_loss', min_delta = 0, patience = 10,␣
↪verbose = 1, restore_best_weights = True)
## L1oL2
k_r=tf.keras.regularizers.L1L2(l1=1e-5,l2=1e-4)
b_r=tf.keras.regularizers.L2(l2=1e-4)
a_r=tf.keras.regularizers.L2(l2=1e-5)
#"""

#"""
# Data Augmentation con RandomCrop
tf.keras.layers.RandomCrop(x_size,y_size,1)

model=Sequential()
model.add(tf.keras.layers.
↪Conv2D(6,(5,5),input_shape=(x_size,y_size,1),activation='tanh',padding='valid',strides=1))␣
↪#C1
model.add(Dropout(0.2)) #Dropout
model.add(tf.keras.layers.BatchNormalization()) # BatchNormalization
model.add(tf.keras.layers.AveragePooling2D(pool_size=(2,2))) #S2
model.add(Dropout(0.9)) #Dropout
model.add(tf.keras.layers.BatchNormalization()) # BatchNormalization
model.add(tf.keras.layers.
↪Conv2D(16,(5,5),activation='tanh',padding='valid',strides=1)) #c3
model.add(Dropout(0.6)) #Dropout
model.add(tf.keras.layers.BatchNormalization()) # BatchNormalization
model.add(tf.keras.layers.AveragePooling2D(pool_size=(2,2))) #s4
model.add(Dropout(0.5)) #Dropout
model.add(tf.keras.layers.BatchNormalization()) # BatchNormalization
model.add(tf.keras.layers.Flatten())
```

```
model.
 ↪add(Dense(120,activation='tanh',kernel_regularizer=k_r,bias_regularizer=b_r,activity_regulari
 ↪#c5
model.add(Dropout(0.6)) #Dropout
model.add(tf.keras.layers.BatchNormalization()) # BatchNormalization
model.
 ↪add(Dense(84,activation='tanh',kernel_regularizer=k_r,bias_regularizer=b_r,activity_regulariz
 ↪#c6
model.add(Dropout(0.7)) #Dropout
model.add(tf.keras.layers.BatchNormalization()) # BatchNormalization

from keras.layers import Layer
from keras import backend as K

class RBFLayer(Layer):
    def __init__(self, units, gamma, **kwargs):
        super(RBFLayer, self).__init__(**kwargs)
        self.units = units
        self.gamma = K.cast_to_floatx(gamma)

    def build(self, input_shape):
        self.mu = self.add_weight(name='mu',
                                  shape=(int(input_shape[1]), self.units),
                                  initializer='uniform',
                                  trainable=True)
        super(RBFLayer, self).build(input_shape)

    def call(self, inputs):
        diff = K.expand_dims(inputs) - self.mu
        l2 = K.sum(K.pow(diff,2), axis=1)
        res = K.exp(-1 * self.gamma * l2)
        return res

    def compute_output_shape(self, input_shape):
        return (input_shape[0], self.units)

model.add(RBFLayer(2,0.5)) #c7

model.compile(loss='categorical_crossentropy',optimizer=tf.keras.optimizers.
 ↪Adam(learning_rate=0.25),metrics=['accuracy'])
hist_final=model.fit(x_train,y_train,verbose=1,␣
 ↪batch_size=32,epochs=150,validation_data=(x_val,y_val),callbacks=[lr_reduce,earlystopper])
```

```
Epoch 1/150
75/75 [==============================] - 6s 53ms/step - loss: 8.4081 - accuracy:
0.5013 - val_loss: 6.2067 - val_accuracy: 0.5000 - lr: 0.2500
Epoch 2/150
```

```
75/75 [==============================] - 3s 42ms/step - loss: 3.9898 - accuracy:
0.5096 - val_loss: 2.6703 - val_accuracy: 0.5000 - lr: 0.2500
Epoch 3/150
75/75 [==============================] - 2s 23ms/step - loss: 2.2438 - accuracy:
0.5029 - val_loss: 1.5981 - val_accuracy: 0.5000 - lr: 0.2500
Epoch 4/150
75/75 [==============================] - 3s 37ms/step - loss: 1.8145 - accuracy:
0.5221 - val_loss: 1.3449 - val_accuracy: 0.5000 - lr: 0.2500
Epoch 5/150
75/75 [==============================] - 3s 36ms/step - loss: 1.5869 - accuracy:
0.5075 - val_loss: 1.1648 - val_accuracy: 0.5000 - lr: 0.2500
Epoch 6/150
75/75 [==============================] - 3s 38ms/step - loss: 1.3273 - accuracy:
0.4808 - val_loss: 1.2436 - val_accuracy: 0.5000 - lr: 0.2500
Epoch 7/150
75/75 [==============================] - 4s 48ms/step - loss: 1.3153 - accuracy:
0.4946 - val_loss: 1.1519 - val_accuracy: 0.5000 - lr: 0.2500
Epoch 8/150
75/75 [==============================] - 3s 40ms/step - loss: 1.3169 - accuracy:
0.4979 - val_loss: 0.9882 - val_accuracy: 0.5000 - lr: 0.2500
Epoch 9/150
75/75 [==============================] - ETA: 0s - loss: 1.4148 - accuracy:
0.5017
Epoch 9: ReduceLROnPlateau reducing learning rate to 0.15.
75/75 [==============================] - 3s 41ms/step - loss: 1.4148 - accuracy:
0.5017 - val_loss: 2.2968 - val_accuracy: 0.5000 - lr: 0.2500
Epoch 10/150
75/75 [==============================] - 3s 36ms/step - loss: 1.5475 - accuracy:
0.5071 - val_loss: 1.0017 - val_accuracy: 0.5000 - lr: 0.1500
Epoch 11/150
75/75 [==============================] - 2s 34ms/step - loss: 0.9871 - accuracy:
0.4958 - val_loss: 0.8342 - val_accuracy: 0.5000 - lr: 0.1500
Epoch 12/150
75/75 [==============================] - 2s 28ms/step - loss: 0.8192 - accuracy:
0.5142 - val_loss: 0.7735 - val_accuracy: 0.5000 - lr: 0.1500
Epoch 13/150
75/75 [==============================] - 2s 31ms/step - loss: 0.7702 - accuracy:
0.4971 - val_loss: 0.7905 - val_accuracy: 0.5000 - lr: 0.1500
Epoch 14/150
75/75 [==============================] - 3s 36ms/step - loss: 0.7557 - accuracy:
0.4996 - val_loss: 0.8091 - val_accuracy: 0.5000 - lr: 0.1500
Epoch 15/150
75/75 [==============================] - 3s 38ms/step - loss: 0.9055 - accuracy:
0.4958 - val_loss: 0.8324 - val_accuracy: 0.5000 - lr: 0.1500
Epoch 16/150
75/75 [==============================] - 2s 26ms/step - loss: 0.8763 - accuracy:
0.5083 - val_loss: 0.8397 - val_accuracy: 0.5000 - lr: 0.1500
Epoch 17/150
```

```
74/75 [============================>.] - ETA: 0s - loss: 1.0106 - accuracy:
0.4916
Epoch 17: ReduceLROnPlateau reducing learning rate to 0.09000000357627869.
75/75 [==============================] - 2s 34ms/step - loss: 1.0093 - accuracy:
0.4917 - val_loss: 0.9981 - val_accuracy: 0.5000 - lr: 0.1500
Epoch 18/150
75/75 [==============================] - 3s 41ms/step - loss: 0.8966 - accuracy:
0.5192 - val_loss: 0.8746 - val_accuracy: 0.5000 - lr: 0.0900
Epoch 19/150
75/75 [==============================] - 3s 43ms/step - loss: 0.7787 - accuracy:
0.4988 - val_loss: 0.7295 - val_accuracy: 0.5000 - lr: 0.0900
Epoch 20/150
75/75 [==============================] - 2s 26ms/step - loss: 0.7605 - accuracy:
0.4804 - val_loss: 0.7241 - val_accuracy: 0.5000 - lr: 0.0900
Epoch 21/150
75/75 [==============================] - 2s 32ms/step - loss: 0.7354 - accuracy:
0.4967 - val_loss: 0.7234 - val_accuracy: 0.5000 - lr: 0.0900
Epoch 22/150
75/75 [==============================] - 2s 26ms/step - loss: 0.7491 - accuracy:
0.4825 - val_loss: 0.7507 - val_accuracy: 0.5000 - lr: 0.0900
Epoch 23/150
75/75 [==============================] - 2s 31ms/step - loss: 0.7584 - accuracy:
0.5046 - val_loss: 0.7708 - val_accuracy: 0.5000 - lr: 0.0900
Epoch 24/150
75/75 [==============================] - 2s 33ms/step - loss: 0.7962 - accuracy:
0.5129 - val_loss: 1.0622 - val_accuracy: 0.5000 - lr: 0.0900
Epoch 25/150
74/75 [============================>.] - ETA: 0s - loss: 0.8414 - accuracy:
0.4928
Epoch 25: ReduceLROnPlateau reducing learning rate to 0.05400000214576721.
75/75 [==============================] - 2s 28ms/step - loss: 0.8408 - accuracy:
0.4917 - val_loss: 0.7505 - val_accuracy: 0.5000 - lr: 0.0900
Epoch 26/150
75/75 [==============================] - 3s 37ms/step - loss: 0.7439 - accuracy:
0.5108 - val_loss: 0.7404 - val_accuracy: 0.5000 - lr: 0.0540
Epoch 27/150
75/75 [==============================] - 3s 47ms/step - loss: 0.7310 - accuracy:
0.5092 - val_loss: 0.7311 - val_accuracy: 0.5000 - lr: 0.0540
Epoch 28/150
75/75 [==============================] - 2s 31ms/step - loss: 0.7322 - accuracy:
0.4892 - val_loss: 0.7402 - val_accuracy: 0.5000 - lr: 0.0540
Epoch 29/150
75/75 [==============================] - 3s 38ms/step - loss: 0.7228 - accuracy:
0.4975 - val_loss: 0.7119 - val_accuracy: 0.5000 - lr: 0.0540
Epoch 30/150
75/75 [==============================] - 3s 38ms/step - loss: 0.7073 - accuracy:
0.4904 - val_loss: 0.6996 - val_accuracy: 0.5000 - lr: 0.0540
Epoch 31/150
```

```
75/75 [==============================] - 4s 54ms/step - loss: 0.7523 - accuracy:
0.5083 - val_loss: 0.8238 - val_accuracy: 0.5000 - lr: 0.0540
Epoch 32/150
75/75 [==============================] - 4s 48ms/step - loss: 0.7765 - accuracy:
0.4879 - val_loss: 0.7550 - val_accuracy: 0.5000 - lr: 0.0540
Epoch 33/150
74/75 [============================>.] - ETA: 0s - loss: 0.7344 - accuracy:
0.4945
Epoch 33: ReduceLROnPlateau reducing learning rate to 0.03240000084042549.
75/75 [==============================] - 2s 32ms/step - loss: 0.7343 - accuracy:
0.4954 - val_loss: 0.7323 - val_accuracy: 0.5000 - lr: 0.0540
Epoch 34/150
75/75 [==============================] - 2s 28ms/step - loss: 0.7233 - accuracy:
0.4900 - val_loss: 0.7059 - val_accuracy: 0.5000 - lr: 0.0324
Epoch 35/150
75/75 [==============================] - 2s 31ms/step - loss: 0.7075 - accuracy:
0.4958 - val_loss: 0.7026 - val_accuracy: 0.5000 - lr: 0.0324
Epoch 36/150
75/75 [==============================] - 2s 30ms/step - loss: 0.7112 - accuracy:
0.5171 - val_loss: 0.7490 - val_accuracy: 0.5000 - lr: 0.0324
Epoch 37/150
75/75 [==============================] - 2s 33ms/step - loss: 0.7189 - accuracy:
0.5208 - val_loss: 0.7080 - val_accuracy: 0.5000 - lr: 0.0324
Epoch 38/150
75/75 [==============================] - 3s 33ms/step - loss: 0.7060 - accuracy:
0.5033 - val_loss: 0.7015 - val_accuracy: 0.5000 - lr: 0.0324
Epoch 39/150
75/75 [==============================] - 2s 28ms/step - loss: 0.7008 - accuracy:
0.4888 - val_loss: 0.6982 - val_accuracy: 0.5000 - lr: 0.0324
Epoch 40/150
75/75 [==============================] - 2s 33ms/step - loss: 0.6985 - accuracy:
0.4867 - val_loss: 0.6952 - val_accuracy: 0.5000 - lr: 0.0324
Epoch 41/150
74/75 [============================>.] - ETA: 0s - loss: 0.6975 - accuracy:
0.5101
Epoch 41: ReduceLROnPlateau reducing learning rate to 0.019440000504255293.
75/75 [==============================] - 3s 42ms/step - loss: 0.6975 - accuracy:
0.5104 - val_loss: 0.6971 - val_accuracy: 0.5000 - lr: 0.0324
Epoch 42/150
75/75 [==============================] - 3s 37ms/step - loss: 0.6973 - accuracy:
0.4950 - val_loss: 0.6960 - val_accuracy: 0.5000 - lr: 0.0194
Epoch 43/150
75/75 [==============================] - 2s 30ms/step - loss: 0.7291 - accuracy:
0.4908 - val_loss: 0.7145 - val_accuracy: 0.5000 - lr: 0.0194
Epoch 44/150
75/75 [==============================] - 2s 30ms/step - loss: 0.7053 - accuracy:
0.4942 - val_loss: 0.7028 - val_accuracy: 0.5000 - lr: 0.0194
Epoch 45/150
```

```
75/75 [==============================] - 2s 30ms/step - loss: 0.7023 - accuracy:
0.4963 - val_loss: 0.7037 - val_accuracy: 0.5000 - lr: 0.0194
Epoch 46/150
75/75 [==============================] - 3s 44ms/step - loss: 0.6995 - accuracy:
0.4875 - val_loss: 0.6964 - val_accuracy: 0.5000 - lr: 0.0194
Epoch 47/150
75/75 [==============================] - 2s 24ms/step - loss: 0.6969 - accuracy:
0.4871 - val_loss: 0.6952 - val_accuracy: 0.5000 - lr: 0.0194
Epoch 48/150
75/75 [==============================] - 3s 45ms/step - loss: 0.6959 - accuracy:
0.4950 - val_loss: 0.6949 - val_accuracy: 0.5000 - lr: 0.0194
Epoch 49/150
74/75 [============================>.] - ETA: 0s - loss: 0.6955 - accuracy:
0.4987
Epoch 49: ReduceLROnPlateau reducing learning rate to 0.011664000526070594.
75/75 [==============================] - 3s 33ms/step - loss: 0.6955 - accuracy:
0.4979 - val_loss: 0.6946 - val_accuracy: 0.5000 - lr: 0.0194
Epoch 50/150
75/75 [==============================] - 2s 32ms/step - loss: 0.6951 - accuracy:
0.5167 - val_loss: 0.6940 - val_accuracy: 0.5000 - lr: 0.0117
Epoch 51/150
75/75 [==============================] - 2s 28ms/step - loss: 0.6959 - accuracy:
0.5167 - val_loss: 0.6963 - val_accuracy: 0.5000 - lr: 0.0117
Epoch 52/150
75/75 [==============================] - 2s 24ms/step - loss: 0.6952 - accuracy:
0.4812 - val_loss: 0.6939 - val_accuracy: 0.5000 - lr: 0.0117
Epoch 53/150
75/75 [==============================] - 2s 30ms/step - loss: 0.6936 - accuracy:
0.4929 - val_loss: 0.6936 - val_accuracy: 0.5000 - lr: 0.0117
Epoch 54/150
75/75 [==============================] - 2s 31ms/step - loss: 0.6936 - accuracy:
0.5008 - val_loss: 0.6936 - val_accuracy: 0.4863 - lr: 0.0117
Epoch 55/150
75/75 [==============================] - 2s 26ms/step - loss: 0.6935 - accuracy:
0.5050 - val_loss: 0.6934 - val_accuracy: 0.5000 - lr: 0.0117
Epoch 56/150
75/75 [==============================] - 2s 32ms/step - loss: 0.6936 - accuracy:
0.4825 - val_loss: 0.6936 - val_accuracy: 0.5000 - lr: 0.0117
Epoch 57/150
75/75 [==============================] - ETA: 0s - loss: 0.6935 - accuracy:
0.5079
Epoch 57: ReduceLROnPlateau reducing learning rate to 0.006998400203883648.
75/75 [==============================] - 3s 34ms/step - loss: 0.6935 - accuracy:
0.5079 - val_loss: 0.6936 - val_accuracy: 0.5000 - lr: 0.0117
Epoch 58/150
75/75 [==============================] - 3s 44ms/step - loss: 0.6941 - accuracy:
0.5021 - val_loss: 0.6935 - val_accuracy: 0.5000 - lr: 0.0070
Epoch 59/150
```

```
75/75 [==============================] - 3s 40ms/step - loss: 0.6933 - accuracy:
0.4929 - val_loss: 0.6933 - val_accuracy: 0.5000 - lr: 0.0070
Epoch 60/150
75/75 [==============================] - 3s 39ms/step - loss: 0.6933 - accuracy:
0.4929 - val_loss: 0.6933 - val_accuracy: 0.5000 - lr: 0.0070
Epoch 61/150
75/75 [==============================] - 3s 40ms/step - loss: 0.6934 - accuracy:
0.5042 - val_loss: 0.6933 - val_accuracy: 0.5000 - lr: 0.0070
Epoch 62/150
75/75 [==============================] - 3s 34ms/step - loss: 0.6933 - accuracy:
0.4808 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 0.0070
Epoch 63/150
75/75 [==============================] - 3s 43ms/step - loss: 0.6933 - accuracy:
0.4842 - val_loss: 0.6933 - val_accuracy: 0.5000 - lr: 0.0070
Epoch 64/150
75/75 [==============================] - 3s 40ms/step - loss: 0.6934 - accuracy:
0.5042 - val_loss: 0.6936 - val_accuracy: 0.5000 - lr: 0.0070
Epoch 65/150
75/75 [==============================] - ETA: 0s - loss: 0.6933 - accuracy:
0.4908
Epoch 65: ReduceLROnPlateau reducing learning rate to 0.004199040122330189.
75/75 [==============================] - 4s 48ms/step - loss: 0.6933 - accuracy:
0.4908 - val_loss: 0.6933 - val_accuracy: 0.5000 - lr: 0.0070
Epoch 66/150
75/75 [==============================] - 3s 42ms/step - loss: 0.6932 - accuracy:
0.4967 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 0.0042
Epoch 67/150
75/75 [==============================] - 2s 29ms/step - loss: 0.6932 - accuracy:
0.4921 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 0.0042
Epoch 68/150
75/75 [==============================] - 2s 27ms/step - loss: 0.6932 - accuracy:
0.4971 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 0.0042
Epoch 69/150
75/75 [==============================] - 2s 31ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 0.0042
Epoch 70/150
75/75 [==============================] - 2s 31ms/step - loss: 0.6932 - accuracy:
0.4942 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 0.0042
Epoch 71/150
75/75 [==============================] - 3s 38ms/step - loss: 0.6932 - accuracy:
0.5067 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 0.0042
Epoch 72/150
75/75 [==============================] - 3s 37ms/step - loss: 0.6932 - accuracy:
0.4979 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 0.0042
Epoch 73/150
74/75 [============================>.] - ETA: 0s - loss: 0.6932 - accuracy:
0.5093
Epoch 73: ReduceLROnPlateau reducing learning rate to 0.002519424073398113.
```

```
75/75 [==============================] - 4s 51ms/step - loss: 0.6932 - accuracy:
0.5104 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 0.0042
Epoch 74/150
75/75 [==============================] - 3s 40ms/step - loss: 0.6932 - accuracy:
0.5113 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 0.0025
Epoch 75/150
75/75 [==============================] - 2s 30ms/step - loss: 0.6932 - accuracy:
0.4746 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 0.0025
Epoch 76/150
75/75 [==============================] - 3s 37ms/step - loss: 0.6932 - accuracy:
0.4992 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 0.0025
Epoch 77/150
75/75 [==============================] - 2s 33ms/step - loss: 0.6932 - accuracy:
0.4967 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 0.0025
Epoch 78/150
75/75 [==============================] - 2s 33ms/step - loss: 0.6932 - accuracy:
0.4979 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 0.0025
Epoch 79/150
75/75 [==============================] - 3s 34ms/step - loss: 0.6932 - accuracy:
0.4967 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 0.0025
Epoch 80/150
75/75 [==============================] - 3s 42ms/step - loss: 0.6932 - accuracy:
0.5033 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 0.0025
Epoch 81/150
74/75 [============================>.] - ETA: 0s - loss: 0.6932 - accuracy:
0.4992
Epoch 81: ReduceLROnPlateau reducing learning rate to 0.0015116544440388678.
75/75 [==============================] - 3s 41ms/step - loss: 0.6932 - accuracy:
0.5004 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 0.0025
Epoch 82/150
75/75 [==============================] - 3s 35ms/step - loss: 0.6932 - accuracy:
0.4850 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 0.0015
Epoch 83/150
75/75 [==============================] - 3s 34ms/step - loss: 0.6932 - accuracy:
0.5063 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 0.0015
Epoch 84/150
75/75 [==============================] - 2s 30ms/step - loss: 0.6932 - accuracy:
0.4900 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 0.0015
Epoch 85/150
75/75 [==============================] - 2s 28ms/step - loss: 0.6932 - accuracy:
0.4942 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 0.0015
Epoch 86/150
75/75 [==============================] - 3s 36ms/step - loss: 0.6932 - accuracy:
0.5038 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 0.0015
Epoch 87/150
75/75 [==============================] - 3s 41ms/step - loss: 0.6932 - accuracy:
0.4908 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 0.0015
Epoch 88/150
```

```
75/75 [==============================] - 3s 39ms/step - loss: 0.6932 - accuracy:
0.4850 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 0.0015
Epoch 89/150
73/75 [============================>.] - ETA: 0s - loss: 0.6932 - accuracy:
0.4983
Epoch 89: ReduceLROnPlateau reducing learning rate to 0.0009069926803931594.
75/75 [==============================] - 2s 25ms/step - loss: 0.6932 - accuracy:
0.4996 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 0.0015
Epoch 90/150
75/75 [==============================] - 2s 31ms/step - loss: 0.6932 - accuracy:
0.4992 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 9.0699e-04
Epoch 91/150
75/75 [==============================] - 2s 26ms/step - loss: 0.6932 - accuracy:
0.4996 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 9.0699e-04
Epoch 92/150
75/75 [==============================] - 2s 28ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 9.0699e-04
Epoch 93/150
75/75 [==============================] - 2s 32ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 9.0699e-04
Epoch 94/150
75/75 [==============================] - 2s 31ms/step - loss: 0.6932 - accuracy:
0.5013 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 9.0699e-04
Epoch 95/150
75/75 [==============================] - 3s 38ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 9.0699e-04
Epoch 96/150
75/75 [==============================] - 3s 39ms/step - loss: 0.6932 - accuracy:
0.4979 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 9.0699e-04
Epoch 97/150
74/75 [============================>.] - ETA: 0s - loss: 0.6932 - accuracy:
0.4987
Epoch 97: ReduceLROnPlateau reducing learning rate to 0.0005441956222057342.
75/75 [==============================] - 3s 35ms/step - loss: 0.6932 - accuracy:
0.5004 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 9.0699e-04
Epoch 98/150
75/75 [==============================] - 2s 26ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 5.4420e-04
Epoch 99/150
75/75 [==============================] - 3s 43ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 5.4420e-04
Epoch 100/150
75/75 [==============================] - 3s 36ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 5.4420e-04
Epoch 101/150
75/75 [==============================] - 2s 28ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 5.4420e-04
Epoch 102/150
```

```
75/75 [==============================] - 3s 34ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 5.4420e-04
Epoch 103/150
75/75 [==============================] - 3s 39ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 5.4420e-04
Epoch 104/150
75/75 [==============================] - 2s 25ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 5.4420e-04
Epoch 105/150
75/75 [==============================] - ETA: 0s - loss: 0.6932 - accuracy:
0.5008
Epoch 105: ReduceLROnPlateau reducing learning rate to 0.00032651738729327917.
75/75 [==============================] - 2s 31ms/step - loss: 0.6932 - accuracy:
0.5008 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 5.4420e-04
Epoch 106/150
75/75 [==============================] - 2s 25ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 3.2652e-04
Epoch 107/150
75/75 [==============================] - 3s 33ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 3.2652e-04
Epoch 108/150
75/75 [==============================] - 2s 27ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 3.2652e-04
Epoch 109/150
75/75 [==============================] - 3s 35ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 3.2652e-04
Epoch 110/150
75/75 [==============================] - 2s 28ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 3.2652e-04
Epoch 111/150
75/75 [==============================] - 2s 33ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 3.2652e-04
Epoch 112/150
75/75 [==============================] - 2s 33ms/step - loss: 0.6932 - accuracy:
0.4992 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 3.2652e-04
Epoch 113/150
74/75 [============================>.] - ETA: 0s - loss: 0.6932 - accuracy:
0.5000
Epoch 113: ReduceLROnPlateau reducing learning rate to 0.0001959104323759675.
75/75 [==============================] - 2s 30ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 3.2652e-04
Epoch 114/150
75/75 [==============================] - 3s 34ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 1.9591e-04
Epoch 115/150
75/75 [==============================] - 4s 49ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 1.9591e-04
Epoch 116/150
```

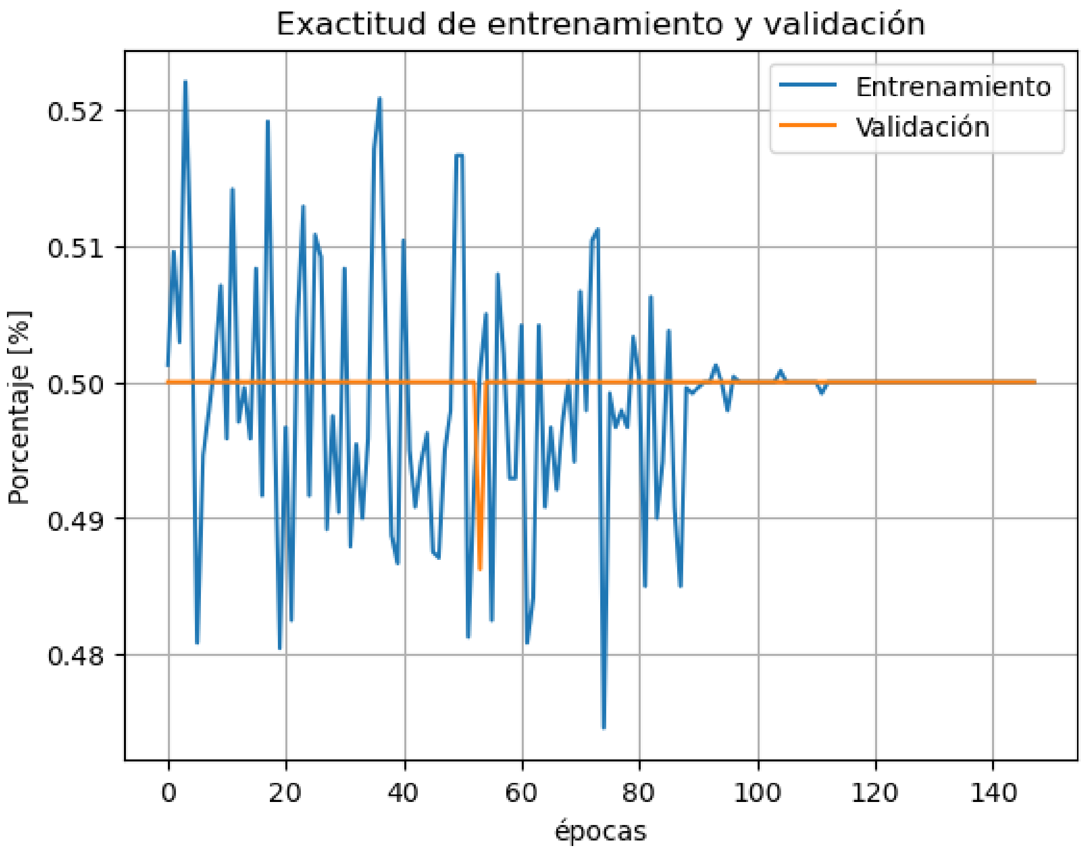
```
75/75 [==============================] - 3s 41ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 1.9591e-04
Epoch 117/150
75/75 [==============================] - 2s 31ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 1.9591e-04
Epoch 118/150
75/75 [==============================] - 2s 28ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 1.9591e-04
Epoch 119/150
75/75 [==============================] - 3s 35ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 1.9591e-04
Epoch 120/150
75/75 [==============================] - 4s 47ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 1.9591e-04
Epoch 121/150
75/75 [==============================] - ETA: 0s - loss: 0.6932 - accuracy:
0.5000
Epoch 121: ReduceLROnPlateau reducing learning rate to 0.00011754625593312084.
75/75 [==============================] - 3s 34ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 1.9591e-04
Epoch 122/150
75/75 [==============================] - 2s 29ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 1.1755e-04
Epoch 123/150
75/75 [==============================] - 3s 38ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 1.1755e-04
Epoch 124/150
75/75 [==============================] - 4s 52ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 1.1755e-04
Epoch 125/150
75/75 [==============================] - 4s 49ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 1.1755e-04
Epoch 126/150
75/75 [==============================] - 4s 47ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 1.1755e-04
Epoch 127/150
75/75 [==============================] - 3s 43ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 1.1755e-04
Epoch 128/150
75/75 [==============================] - 3s 34ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 1.1755e-04
Epoch 129/150
71/75 [===========================>..] - ETA: 0s - loss: 0.6932 - accuracy:
0.4991
Epoch 129: ReduceLROnPlateau reducing learning rate to 7.052775181364268e-05.
75/75 [==============================] - 2s 31ms/step - loss: 0.6932 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 1.1755e-04
Epoch 130/150
```

```
75/75 [==============================] - 2s 31ms/step - loss: 0.6931 - accuracy:
0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000 - lr: 7.0528e-05
Epoch 131/150
75/75 [==============================] - 3s 37ms/step - loss: 0.6931 - accuracy:
0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000 - lr: 7.0528e-05
Epoch 132/150
75/75 [==============================] - 4s 48ms/step - loss: 0.6931 - accuracy:
0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000 - lr: 7.0528e-05
Epoch 133/150
75/75 [==============================] - 3s 35ms/step - loss: 0.6931 - accuracy:
0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000 - lr: 7.0528e-05
Epoch 134/150
75/75 [==============================] - 2s 28ms/step - loss: 0.6931 - accuracy:
0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000 - lr: 7.0528e-05
Epoch 135/150
75/75 [==============================] - 4s 51ms/step - loss: 0.6931 - accuracy:
0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000 - lr: 7.0528e-05
Epoch 136/150
75/75 [==============================] - 3s 37ms/step - loss: 0.6931 - accuracy:
0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000 - lr: 7.0528e-05
Epoch 137/150
74/75 [============================>.] - ETA: 0s - loss: 0.6931 - accuracy:
0.4970
Epoch 137: ReduceLROnPlateau reducing learning rate to 5e-05.
75/75 [==============================] - 4s 50ms/step - loss: 0.6931 - accuracy:
0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000 - lr: 7.0528e-05
Epoch 138/150
75/75 [==============================] - 4s 47ms/step - loss: 0.6931 - accuracy:
0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000 - lr: 5.0000e-05
Epoch 139/150
75/75 [==============================] - 3s 44ms/step - loss: 0.6931 - accuracy:
0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000 - lr: 5.0000e-05
Epoch 140/150
75/75 [==============================] - 3s 41ms/step - loss: 0.6931 - accuracy:
0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000 - lr: 5.0000e-05
Epoch 141/150
75/75 [==============================] - 2s 27ms/step - loss: 0.6931 - accuracy:
0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000 - lr: 5.0000e-05
Epoch 142/150
75/75 [==============================] - 3s 34ms/step - loss: 0.6931 - accuracy:
0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000 - lr: 5.0000e-05
Epoch 143/150
75/75 [==============================] - 3s 46ms/step - loss: 0.6931 - accuracy:
0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000 - lr: 5.0000e-05
Epoch 144/150
75/75 [==============================] - 3s 36ms/step - loss: 0.6931 - accuracy:
0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000 - lr: 5.0000e-05
Epoch 145/150
```

```
75/75 [==============================] - 3s 46ms/step - loss: 0.6931 - accuracy:
0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000 - lr: 5.0000e-05
Epoch 146/150
75/75 [==============================] - 2s 30ms/step - loss: 0.6931 - accuracy:
0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000 - lr: 5.0000e-05
Epoch 147/150
75/75 [==============================] - 2s 32ms/step - loss: 0.6931 - accuracy:
0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000 - lr: 5.0000e-05
Epoch 148/150
75/75 [==============================] - ETA: 0s - loss: 0.6931 - accuracy:
0.5000Restoring model weights from the end of the best epoch: 138.
75/75 [==============================] - 2s 31ms/step - loss: 0.6931 - accuracy:
0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000 - lr: 5.0000e-05
Epoch 148: early stopping
```

[15]:

```
type(hist_final.history['loss'])
import matplotlib.pyplot as plt
plt.plot(hist_final.history['accuracy'])
plt.plot(hist_final.history['val_accuracy'])
plt.title('Exactitud de entrenamiento y validación')
plt.xlabel('épocas')
plt.ylabel('Porcentaje [%]')
plt.legend(['Entrenamiento','Validación'])
plt.grid()
```



[16]:
```
pred=model.predict(x_test)
pred=np.argmax(pred,axis=1)
y1=np.argmax(y_test,axis=1)

#label=np.argmax(yp_oh)
exactitud_test=0
for a in range(len(pred)):
    if pred[a]==y1[a]:
        exactitud_test+=1
print('exactitud de la prueba= ',100*exactitud_test/len(pred),'%')
```

```
25/25 [==============================] - 0s 1ms/step
exactitud de la prueba=  50.0 %
```

[ ]: