INFORME PRUEBA FINAL TRATAMIENTO DE DATOS

ALUMNO: Ing. David Alejandro Vilatuña CH.

MATERIA: Tratamiento de datos.

MAESTRIA: Ciberseguridad

FECHA: 04/07/2023

TEMA:

- Clasificador de carnes

OBJETIVO:

- Crear un repositorio de GIThub (publico) en el que se va a subir un jupyter notebook y un archivo README.md
- Obtener un clasificador de imágenes de forma que dada una nueva imagen se pueda obtener la clase correspondiente
- Se pide obtener las matrices de confusión del modelo, la matriz de confusión del error en training y la de test.

NOMENCLATURA:

- PANDAS: es una biblioteca ampliamente utilizada para administrar datos tabulares, manipulación de datos y análisis basados en NumPy.
- AMBIENTE VIRTUAL: Es una herramienta para crear entornos Python aislados
- IPYKERNEL: Este paquete proporciona el kernel de IPython para Jupyter
- SCIKIT-LEARN: conocido como sklearn, es una biblioteca robusta de aprendizaje automático de Python de código abierto
- KERAS: API de redes neuronales más utilizadas para el desarrollo y las pruebas de redes neuronales
- TENSORFLOW: librería de código libre para Machine Learning (ML)
- OPENPYXL: biblioteca de Python para leer/escribir archivos Excel 2010 xlsx/xlsm/xltx/xltm
- MATPLOTLI: biblioteca completa para crear visualizaciones estáticas, animadas e interactivas en Python
- MATH: Este módulo proporciona acceso a las funciones matemáticas definidas por el estándar C

MARCO TEORICO

FUNCIONAMIENTO DE LAS REDES NEURONALES CONVOLUCIONARIAS

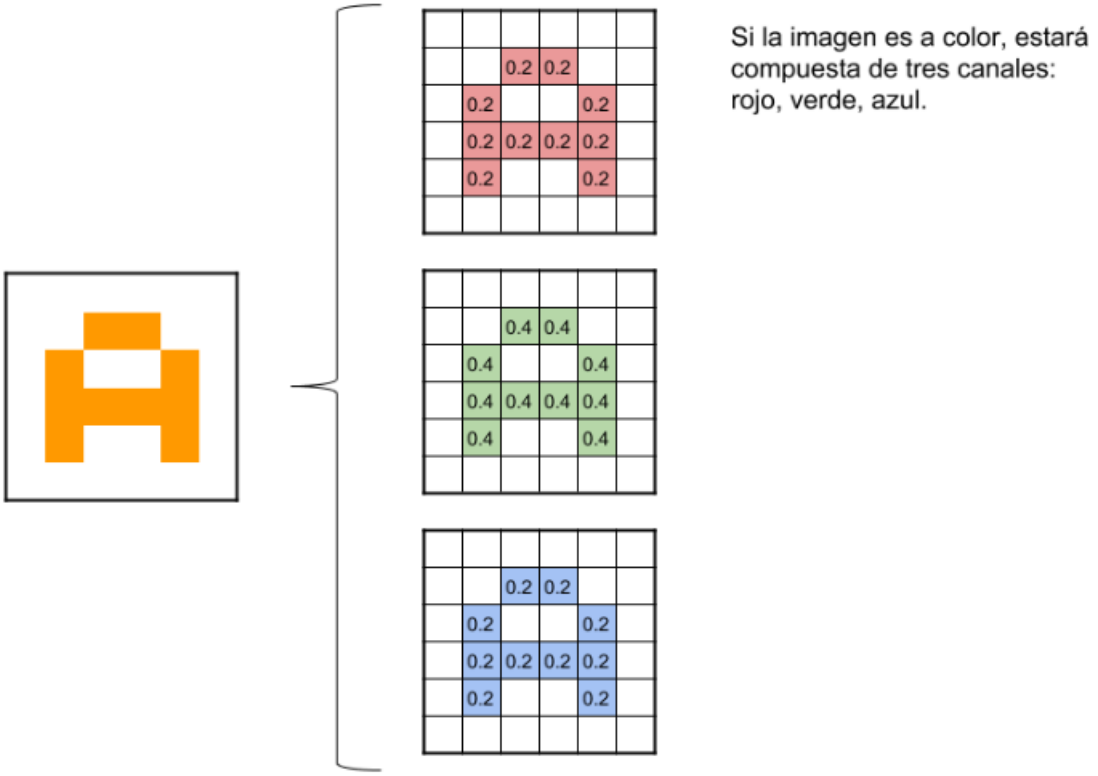
Estas redes, son las que se relacionan con la visión por ordenador. Estas son algoritmos utilizados para el Aprendizaje automático, para darle al ordenador la capacidad de VER. Estas CNN procesan las capas utilizando capas, que se asemejan al cortex del ojo humano,

identificando características del objeto, para ello se debe poner capas ocultas especilizadas jerárquicamente organizadas.

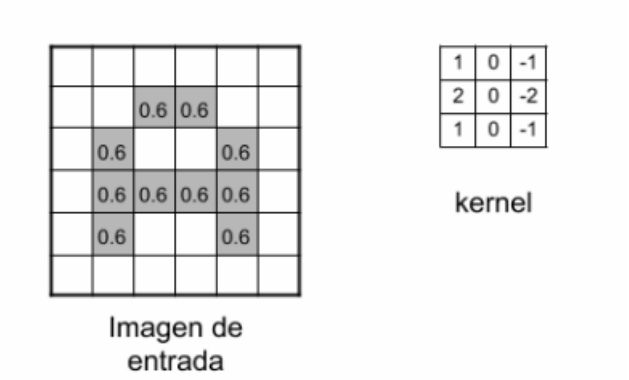
Para esto es importante tener en cuenta que se requieren muchas imágenes para que pueda captar características de las clasificaciones que se requieran entrenar.

El proceso de trabajo es que la red toma como entrada pixeles de una imagen, en el caso de las que tenemos en el ejercicio tomamos unas fotos de 384 x 216= 82944 pixeles y con una capa de color 3, total 248,832 neuronas de entrada.

Antes de alimentar la red debemos normalizar la misma, dividiendo el valor para 255, esto pone el valor en un rango de 0 a 1.

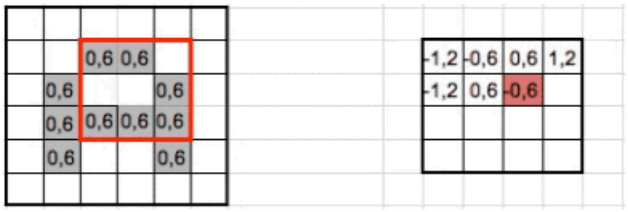


Luego se va a realizar las convoluciones que consiste en tomar grupos de pixeles cercanos de la imagen de entrada y operarlos matematicamenre con una pequeña matriz que se llama kernel, esta matriz recorre todas las neuronas de entrada de izquierda a derecha y de arriba abajo, con ello se genera una nueva matriz de salida.

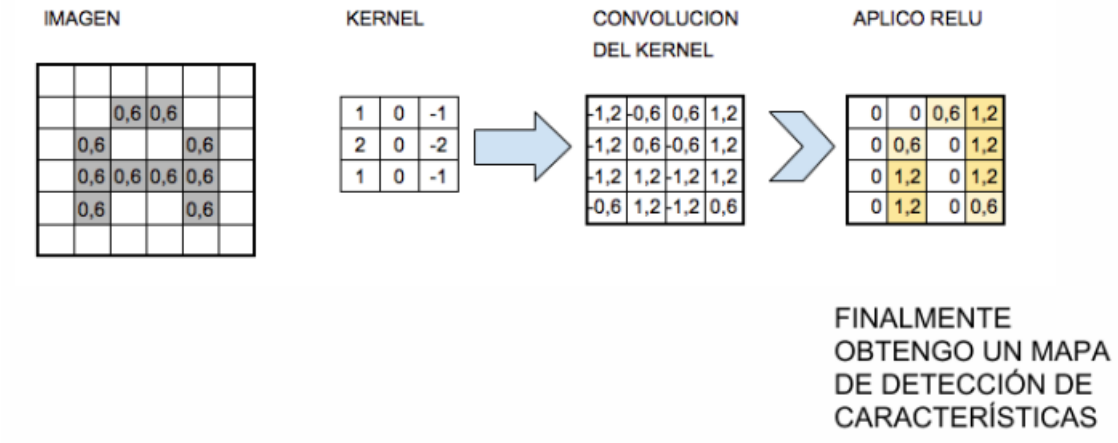


El kernel tomará inicialmente valores aleatorios(1) y se irán ajustando mediante backpropagation. (1)Una mejora es hacer que siga una distribución normal siguiendo simetrías, pero sus valores son aleatorios.

El proceso no ocupa un solo kernel sino que es el conjunto de filtros.

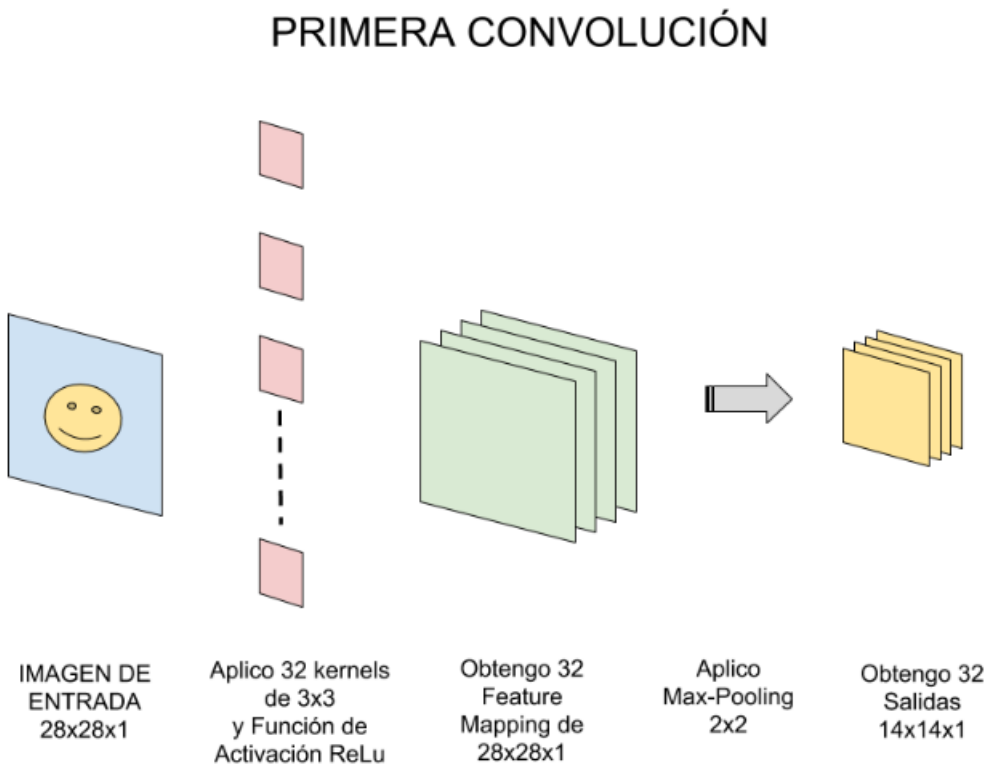


Aquí vemos al kernel realizando el producto matricial con la imagen de entrada y desplazando de a 1 pixel de izquierda a derecha y de arriba-abajo y va generando una nueva matriz que compone al mapa de features



La imagen realiza una convolución con un kernel y aplica la función de activación, en este caso ReLu

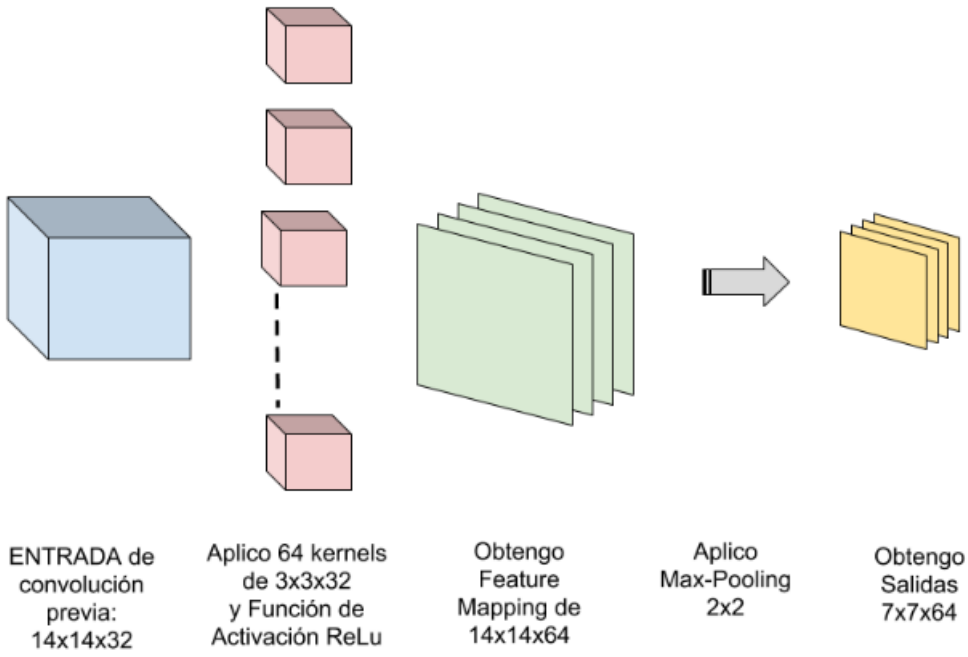
Con todos los procesos descritos la siguiente imagen explicaría de manera la manera que trabaja.



La primera convolución, consiste de una entrada, un conjunto de filtros, un mapa de características un subsampling, que con una imagen de1 solo color se tiene

1)Entrada: Imagen	2)Aplico Kernel	3)Obtengo Feature Mapping	4)Aplico Max-Pooling	5)Obtengo "Salida" de la Convolución
28x28x1	32 filtros de 3×3	28x28x32	de 2×2	14x14x32

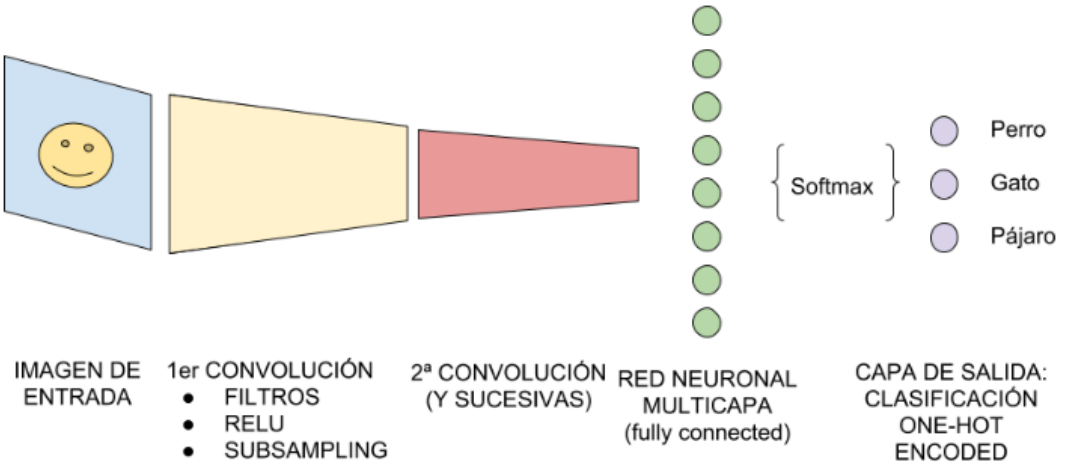
En esta primera convolución el modelo observa características primitivas como líneas y curvas, con mas de estas capas el modelo ya conocera mas formas complejas

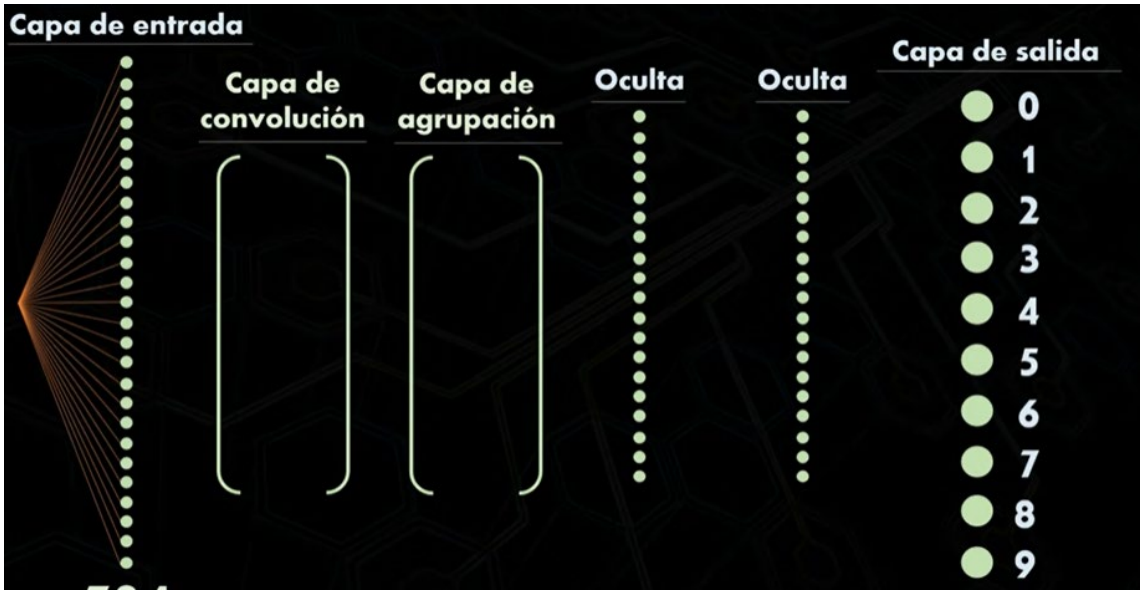


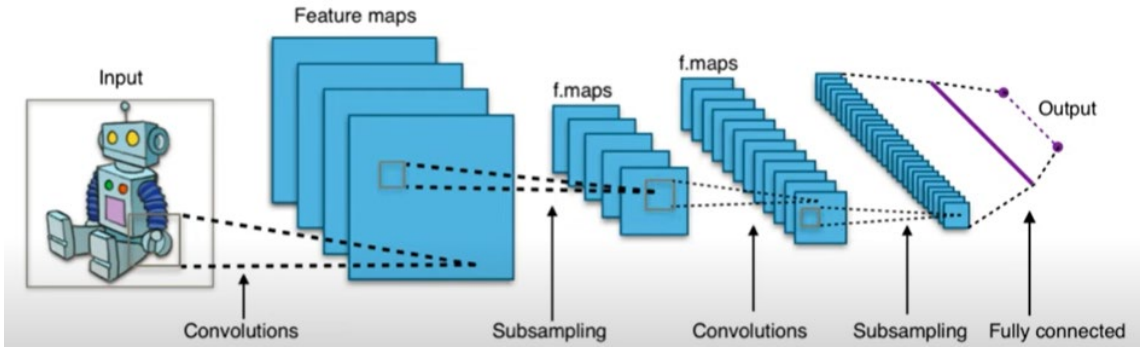
La segunda

1)Entrada: Imagen	2)Aplico Kernel	3)Obtengo Feature Mapping	4)Aplico Max-Pooling	5)Obtengo “Salida” de la Convolución
14x14x32	64 filtros de 3×3	14x14x64	de 2×2	7x7x64

La forma de la red cnn se la indica en el siguiente grafico.







Es importante comentar que la sunción Softmax conecta contra la capa de salida final que tendría las neuronas correspondientes con las clases que estamos clasificando.

Las salidas del entrenamiento tendrán un formato conocido como one-hot-encoding, que pasa a ser una variación representada por números binarios.

La arquitectura básica de las cnn son:

Entrada: Serán los pixeles de la imagen. Serán alto, ancho y profundidad será 1 sólo color o 3 para Red,Green,Blue.

Capa De Convolución: procesará la salida de neuronas que están conectadas en "regiones locales" de entrada (es decir pixeles cercanos), calculando el producto escalar entre sus pesos (valor de pixel) y una pequeña región a la que están conectados en el volumen de entrada. Aquí usaremos por ejemplo 32 filtros o la cantidad que decidamos y ese será el volumen de salida.

"CAPA RELU" aplicará la función de activación en los elementos de la matriz.

POOL ó SUBSAMPLING: Hará una reducción en las dimensiones alto y ancho, pero se mantiene la profundidad.

CAPA "TRADICIONAL" red de neuronas feedforward que conectará con la última capa de subsampling y finalizará con la cantidad de neuronas que queremos clasificar.

DESARROLLO

Para este desarrollo, vamos a realizar una Convolutional Neural Network con Keras y Tensorflow, con el lenguaje de programación Python, para el reconocimiento de imágenes.

En base a las imágenes entregadas tenemos 2 una carpeta de Train y otra de Test,

Lo que se utilizara para el desarrollo de la aplicación son:

- Python
- Notebook Jupyter
- Anaconda
- Keras y Tensorflow

Instrucciones de instalación:

python -m virtualenv venv #preparamos el ambiente virtual

.\venv\Scripts\activate

pip install pandas # instalamos pandas

pip install ipykernel # instalamos ipykernel #

pip install scikit-learn

pip install keras

pip install tensorflow

pip install openpyxl

pip freeze > requirements.txt

pip install matplotli

pip install math

```
import numpy as np
import os
import re
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import keras
from keras.utils import to_categorical
from keras.models import Sequential,Model
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.layers.normalization import batch_normalization
from keras.layers.activation import LeakyReLU
```

[41] ✓ 0.6s Python

Declaramos todas las librerías a utilizar

```
dirname_train = os.path.join(os.getcwd(), 'train')
imgpath_train = dirname_train + os.sep

images_train = []
directories_train = []
dircount_train = []
prevRoot_train=''
cant_train=0

print("leyendo imagenes de entrenamiento: ",imgpath_train)

for root, dirnames, filenames in os.walk(imgpath_train):
    for filename in filenames:
        if re.search("\.(jpg|jpeg|png|bmp|tiff)$", filename):
            cant_train=cant_train+1
            filepath = os.path.join(root, filename)
            image = plt.imread(filepath)
            images_train.append(image)
            b = "Leyendo..." + str(cant_train)
            print (b, end="\r")
            if prevRoot_train !=root:
```

```
            print (b, end="\r")
            if prevRoot_train !=root:
                print(root, cant_train)
                prevRoot_train=root
                directories_train.append(root)
                dircount_train.append(cant_train)
                cant_train=0
    dircount_train.append(cant_train)

    dircount_train = dircount_train[1:]
    dircount_train[0]=dircount_train[0]+1
    print('Directorios leidos:',len(directories_train))
    print("Imagenes en cada directorio", dircount_train)
    print('suma Total de imagenes en subdirs:',sum(dircount_train))

    dirname_test = os.path.join(os.getcwd(), 'test')
    imgpath_test = dirname_test + os.sep

    images_test = []
    directories_test = []
    dircount_test = []
    prevRoot_test=''
    cant_test=0

    print("leyendo imagenes de testeo: ",imgpath_test)

    for root, dirnames, filenames in os.walk(imgpath_test):
```

PROBLEMAS 6 SALIDA CONSOLA DE DEPURACIÓN TERMINAL JUPYTER

```python
print('leyendo imagenes de testeo: ',imgpath_test)

for root, dirnames, filenames in os.walk(imgpath_test):
    for filename in filenames:
        if re.search("\.(jpg|jpeg|png|bmp|tiff)$", filename):
            cant_test=cant_test+1
            filepath = os.path.join(root, filename)
            image = plt.imread(filepath)
            images_test.append(image)
            b = "Leyendo..." + str(cant_test)
            print (b, end="\r")
            if prevRoot_test !=root:
                print(root, cant_test)
                prevRoot_test=root
                directories_test.append(root)
                dircount_test.append(cant_test)
                cant_test=0
dircount_test.append(cant_test)

dircount_test = dircount_test[1:]
dircount_test[0]=dircount_test[0]+1
print('Directorios leidos:',len(directories_test))
print("Imagenes en cada directorio", dircount_test)
print('suma Total de imagenes en subdirs:',sum(dircount_test))
```

4] ✓ 11.6s Python

```
leyendo imagenes de entrenamiento:  c:\Users\vilas\Documents\TRATAMIENTO_DATOS\prueba_final_modulo\proyecto_final_modulo\train\
c:\Users\vilas\Documents\TRATAMIENTO_DATOS\prueba_final_modulo\proyecto_final_modulo\train\CLASS_02 1
c:\Users\vilas\Documents\TRATAMIENTO_DATOS\prueba_final_modulo\proyecto_final_modulo\train\CLASS_03 62
c:\Users\vilas\Documents\TRATAMIENTO_DATOS\prueba_final_modulo\proyecto_final_modulo\train\CLASS_04 213
c:\Users\vilas\Documents\TRATAMIENTO_DATOS\prueba_final_modulo\proyecto_final_modulo\train\CLASS_05 105
c:\Users\vilas\Documents\TRATAMIENTO_DATOS\prueba_final_modulo\proyecto_final_modulo\train\CLASS_06 949
c:\Users\vilas\Documents\TRATAMIENTO_DATOS\prueba_final_modulo\proyecto_final_modulo\train\CLASS_07 37
c:\Users\vilas\Documents\TRATAMIENTO_DATOS\prueba_final_modulo\proyecto_final_modulo\train\CLASS_08 204
Directorios leidos: 7
Imagenes en cada directorio [63, 213, 105, 949, 37, 204, 62]
suma Total de imagenes en subdirs: 1633
leyendo imagenes de testeo:  c:\Users\vilas\Documents\TRATAMIENTO_DATOS\prueba_final_modulo\proyecto_final_modulo\test\
c:\Users\vilas\Documents\TRATAMIENTO_DATOS\prueba_final_modulo\proyecto_final_modulo\test\CLASS_01 1
c:\Users\vilas\Documents\TRATAMIENTO_DATOS\prueba_final_modulo\proyecto_final_modulo\test\CLASS_02 1
c:\Users\vilas\Documents\TRATAMIENTO_DATOS\prueba_final_modulo\proyecto_final_modulo\test\CLASS_03 48
c:\Users\vilas\Documents\TRATAMIENTO_DATOS\prueba_final_modulo\proyecto_final_modulo\test\CLASS_04 97
c:\Users\vilas\Documents\TRATAMIENTO_DATOS\prueba_final_modulo\proyecto_final_modulo\test\CLASS_05 45
c:\Users\vilas\Documents\TRATAMIENTO_DATOS\prueba_final_modulo\proyecto_final_modulo\test\CLASS_06 459
c:\Users\vilas\Documents\TRATAMIENTO_DATOS\prueba_final_modulo\proyecto_final_modulo\test\CLASS_07 19
c:\Users\vilas\Documents\TRATAMIENTO_DATOS\prueba_final_modulo\proyecto_final_modulo\test\CLASS_08 114
Directorios leidos: 8
```

Se realiza la lectura de las imágenes en el disco

```python
labels_train=[]
indice_train=0
for cantidad in dircount_train:
    for i in range(cantidad):
        labels_train.append(indice_train)
    indice_train=indice_train+1
print("Cantidad etiquetas creadas: ",len(labels_train))

clases_carne_train=[]
indice_train=0
for directorio in directories_train:
    name = directorio.split(os.sep)
    print(indice_train , name[len(name)-1])
    clases_carne_train.append(name[len(name)-1])
    indice_train=indice_train+1

y_train = np.array(labels_train)
X_train = np.array(images_train, dtype=np.uint8) #convierto de lista a numpy

# Find the unique numbers from the train labels
classes_train = np.unique(y_train)
nClasses_train = len(classes_train)
print('Total number of outputs : ', nClasses_train)
print('Output classes : ', classes_train)
```

Realizamos la lectura de las categorías de las imágenes en el disco

```
Cantidad etiquetas creadas:  1633
0 CLASS_02
1 CLASS_03
2 CLASS_04
3 CLASS_05
4 CLASS_06
5 CLASS_07
6 CLASS_08
Total number of outputs :  7
Output classes :  [0 1 2 3 4 5 6]
Cantidad etiquetas creadas:  810
0 CLASS_01
1 CLASS_02
2 CLASS_03
3 CLASS_04
4 CLASS_05
5 CLASS_06
6 CLASS_07
7 CLASS_08
Total number of outputs :  8
Output classes :  [0 1 2 3 4 5 6 7]
```

```
    labels_test=[]
    indice_test=0
    for cantidad in dircount_test:
        for i in range(cantidad):
            labels_test.append(indice_test)
        indice_test=indice_test+1
    print("Cantidad etiquetas creadas: ",len(labels_test))

    clases_carne_test=[]
    indice_test=0
    for directorio in directories_test:
        name = directorio.split(os.sep)
        print(indice_test , name[len(name)-1])
        clases_carne_test.append(name[len(name)-1])
        indice_test=indice_test+1

    y_test = np.array(labels_test)
    X_test = np.array(images_test, dtype=np.uint8) #convierto de lista a numpy

    # Find the unique numbers from the train labels
    classes_test = np.unique(y_test)
    nClasses_test = len(classes_test)
    print('Total number of outputs : ', nClasses_test)
    print('Output classes : ', classes_test)
[45]  ✓ 0.7s                                                    Python
```

se prepara el set de datos

```
#Mezclar todo y crear los grupos de entrenamiento y testing
train_X,train_test_X,train_Y,train_test_Y = train_test_split(X_train,y_train,test_size=0.2)

print('Training data shape : ', train_X.shape, train_Y.shape)
print('Testing data shape : ',train_test_X.shape, train_test_Y.shape)
#normlizamos los valores
train_X = train_X.astype('float32')
train_X1=train_X.astype('float32')
train_test_X = train_test_X.astype('float32')
train_X = train_X / 255.
train_X1 = train_X / 255.
train_test_X = train_test_X / 255.

# Change the labels from categorical to one-hot encoding
train_Y_one_hot = to_categorical(train_Y)
train_test_Y_one_hot = to_categorical(train_test_Y)

# Display the change for category label using one-hot encoding
print('Original label 0:', train_Y[0])
print('Original label 1:', train_Y[1])
print('After conversion to one-hot 0:', train_Y_one_hot[0])
print('After conversion to one-hot 1:', train_Y_one_hot[1])

#train_X,valid_X,train_label,valid_label = train_test_split(train_X, train_Y_one_hot,test_size=0.001, random_state=13)
train_X,train_valid_X,train_label,train_valid_label = train_test_split(train_X, train_Y_one_hot, test_size=0.3, random_state=13)
```

```
train_X,train_valid_X,train_label,train_valid_label = train_test_split(train_X, train_Y_one_hot, test_size=0.3,

print(train_X.shape,"\n",train_X1.shape,"\n",train_valid_X.shape,"\n",train_label.shape,"\n",train_valid_label.shape)
```

✓ 5.4s Python

```
Training data shape :  (1306, 216, 384, 3) (1306,)
Testing data shape :  (327, 216, 384, 3) (327,)
Original label 0: 5
Original label 1: 1
After conversion to one-hot 0: [0. 0. 0. 0. 0. 1. 0.]
After conversion to one-hot 1: [0. 1. 0. 0. 0. 0. 0.]
(914, 216, 384, 3)
 (1306, 216, 384, 3)
 (392, 216, 384, 3)
 (914, 7)
 (392, 7)
```

Preparamos las redes de las capas intermedias

```
INIT_LR = 1e-3
epochs = 6
batch_size = 64
carne_model = Sequential()
carne_model.add(Conv2D(32, kernel_size=(3, 3),activation='linear',padding='same',input_shape=(216,384,3))) ##32  384,216
carne_model.add(LeakyReLU(alpha=0.1))
carne_model.add(MaxPooling2D((2, 2),padding='same')) # tamaño de la matriz de agrupacion
#----------------------------------------
carne_model.add(Conv2D(64, kernel_size=(3, 3),activation='linear',padding='same')) ##32  384,216
carne_model.add(LeakyReLU(alpha=0.1))
carne_model.add(MaxPooling2D((2, 2),padding='same')) # tamaño de la matriz de agrupacion
carne_model.add(Conv2D(128, kernel_size=(3, 3),activation='linear',padding='same')) ##32  384,216
carne_model.add(LeakyReLU(alpha=0.1))
carne_model.add(MaxPooling2D((2, 2),padding='same')) # tamaño de la matriz de agrupacion
carne_model.add(Conv2D(256, kernel_size=(3, 3),activation='linear',padding='same')) ##32  384,216
carne_model.add(LeakyReLU(alpha=0.1))
carne_model.add(MaxPooling2D((2, 2),padding='same')) # tamaño de la matriz de agrupacion
carne_model.add(Conv2D(256, kernel_size=(3, 3),activation='linear',padding='same')) ##32  384,216
carne_model.add(LeakyReLU(alpha=0.1))
carne_model.add(MaxPooling2D((2, 2),padding='same')) # tamaño de la matriz de agrupacion
#----------------------------------------------------
carne_model.add(Dropout(0.5))
carne_model.add(Flatten())
carne_model.add(Dense(256, activation='linear'))  #32 -> 256
carne_model.add(Dense(128, activation='linear'))  #32 -> 256
```

```
carne_model.add(LeakyReLU(alpha=0.1))
carne_model.add(Dropout(0.5))
carne_model.add(Dense(nClasses_train, activation='softmax'))  #nClasses_train
carne_model.summary()
carne_model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.Adagrad(learning_rate=INIT_LR,
decay=INIT_LR / 100),metrics=['accuracy'])
```

[48] ✓ 0.8s Pyth

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_5 (Conv2D)           (None, 216, 384, 32)      896

 leaky_re_lu_6 (LeakyReLU)   (None, 216, 384, 32)      0

 max_pooling2d_5 (MaxPooling  (None, 108, 192, 32)     0
 2D)

 conv2d_6 (Conv2D)           (None, 108, 192, 64)      18496

 leaky_re_lu_7 (LeakyReLU)   (None, 108, 192, 64)      0
```

```
leaky_re_lu_7 (LeakyReLU)   (None, 108, 192, 64)      0

max_pooling2d_6 (MaxPooling  (None, 54, 96, 64)       0
2D)

conv2d_7 (Conv2D)           (None, 54, 96, 128)       73856

leaky_re_lu_8 (LeakyReLU)   (None, 54, 96, 128)       0

max_pooling2d_7 (MaxPooling  (None, 27, 48, 128)      0
2D)

conv2d_8 (Conv2D)           (None, 27, 48, 256)       295168

leaky_re_lu_9 (LeakyReLU)   (None, 27, 48, 256)       0

max_pooling2d_8 (MaxPooling  (None, 14, 24, 256)      0
2D)

conv2d_9 (Conv2D)           (None, 14, 24, 256)       590080

leaky_re_lu_10 (LeakyReLU)  (None, 14, 24, 256)       0
```

```
dropout_2 (Dropout)         (None, 7, 12, 256)        0

flatten_1 (Flatten)         (None, 21504)             0

dense_4 (Dense)             (None, 256)               5505280

dense_5 (Dense)             (None, 128)               32896

dense_6 (Dense)             (None, 64)                8256

leaky_re_lu_11 (LeakyReLU)  (None, 64)                0

dropout_3 (Dropout)         (None, 64)                0

dense_7 (Dense)             (None, 7)                 455

=================================================================
Total params: 6,525,383
Trainable params: 6,525,383
Non-trainable params: 0
_________________________________________________________________
```

Entrenamos el modelo

```
carne_train_dropout = carne_model.fit(train_X, train_label, batch_size=batch_size,epochs=epochs,verbose=1,validation_data=
(train_valid_X, train_valid_label))

# guardamos la red, para reutilizarla en el futuro, sin tener que volver a entrenar
carne_model.save("carnes_mnist.h5py")
```

[49] ✓ 10m 36.3s Python

```
Epoch 1/6
15/15 [==============================] - 125s 8s/step - loss: 1.9290 - accuracy: 0.5394 - val_loss: 1.9086 - val_accuracy: 0.5689
Epoch 2/6
15/15 [==============================] - 116s 8s/step - loss: 1.8919 - accuracy: 0.5777 - val_loss: 1.8734 - val_accuracy: 0.5689
Epoch 3/6
15/15 [==============================] - 99s 7s/step - loss: 1.8588 - accuracy: 0.5678 - val_loss: 1.8364 - val_accuracy: 0.5689
Epoch 4/6
15/15 [==============================] - 99s 7s/step - loss: 1.8198 - accuracy: 0.5733 - val_loss: 1.7947 - val_accuracy: 0.5689
Epoch 5/6
15/15 [==============================] - 97s 7s/step - loss: 1.7702 - accuracy: 0.5722 - val_loss: 1.7434 - val_accuracy: 0.5689
Epoch 6/6
15/15 [==============================] - 95s 6s/step - loss: 1.7161 - accuracy: 0.5744 - val_loss: 1.6793 - val_accuracy: 0.5689

WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing 5 of 5). These functions will not be directly callable
after loading.
```

Evaluamos el modelo

```
test_eval = carne_model.evaluate(train_test_X, train_test_Y_one_hot, verbose=1)
print('Test loss:', test_eval[0])
print('Test accuracy:', test_eval[1])
```

[50] ✓ 12.1s Python

```
11/11 [==============================] - 12s 1s/step - loss: 1.6588 - accuracy: 0.5994
Test loss: 1.6588343381881714
Test accuracy: 0.5993883609771729
```

Se genera la matriz de confusión

```
y_prediccion_train=carne_model.predict(train_X1)#.round()

confusion_matrix(train_Y_one_hot.argmax(axis=1),y_prediccion_train.argmax(axis=1))
```

[52] ✓ 51.9s Pytho

```
41/41 [==============================] - 52s 1s/step

array([[  0,   0,   0,  52,   0,   0,   0],
       [  0,   0,   0, 174,   0,   0,   0],
       [  0,   0,   0,  88,   0,   0,   0],
       [  0,   0,   0, 753,   0,   0,   0],
       [  0,   0,   0,  32,   0,   0,   0],
       [  0,   0,   0, 159,   0,   0,   0],
       [  0,   0,   0,  48,   0,   0,   0]], dtype=int64)
```

CONCLUSIONES Y RECOMENDACIONES:

- Si bien la teoría de tratamiento de datos nos ha servido para profundizar en este tipo de lecciones, es importante reconocer que el reconocimiento de imágenes es un proceso mas complejo de manejo de información, Ya que en el se realizan muchos tipos de programación las cuales de manera personal me parece muy interesante pero es importante tener un conocimiento mucho mas acertado con el proceso, y manejo de librerías de los tratamientos de imágenes
- Se reconoce la complejidad de la enseñanza a modelos neuronales ya que los parámetros de información deben validarse de acuerdo a la información realizada.
- Las personas que se dedican a este campo deben formar parte de expertos en el campo de redes con lo cual se puede complicar mucho mas los datos de seguimiento para la clasificación exacta de estas redes.
- Se recomienda, realizar un modelo de acercamiento de las imágenes en clase ya que por mas que se trata de seguir la programación que se encuentra colgada en algunos sitios es importante siempre tener el direccionamiento de como funciona para que sirve y como se configura.
- A medida de lo que se entendió en el proceso se puede decir que se logro trabajar el proyecto, para mi nivel de programación pues fue un reto hacer lo que se alcanzo a realizar.