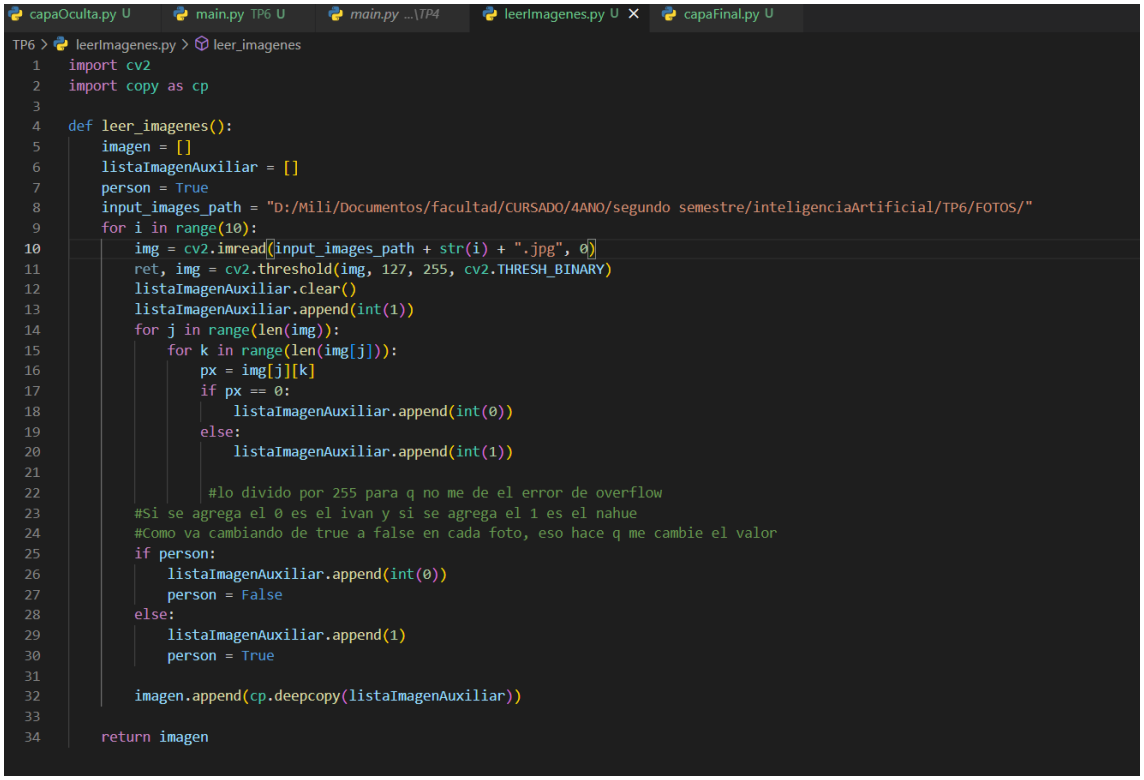
TRABAJO PRACTICO N°6

En este trabajo tomamos 5 fotos de una persona A y 5 fotos de una persona B de 80 ancho*96 alto.

Cada imagen se toma como “Entrada” las cuales son tomadas como 7680 entradas, y además le agregamos el vías =1 y la salida real que puede ser 0 (persona A) o 1 (persona B). Es decir, en total va a contar de 7682 entradas.

CÓDIGO:

leerImagenes.py



```python
import cv2
import copy as cp

def leer_imagenes():
    imagen = []
    listaImagenAuxiliar = []
    person = True
    input_images_path = "D:/Mili/Documentos/facultad/CURSADO/4ANO/segundo semestre/inteligenciaArtificial/TP6/FOTOS/"
    for i in range(10):
        img = cv2.imread(input_images_path + str(i) + ".jpg", 0)
        ret, img = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)
        listaImagenAuxiliar.clear()
        listaImagenAuxiliar.append(int(1))
        for j in range(len(img)):
            for k in range(len(img[j])):
                px = img[j][k]
                if px == 0:
                    listaImagenAuxiliar.append(int(0))
                else:
                    listaImagenAuxiliar.append(int(1))

                #lo divido por 255 para q no me de el error de overflow
        #Si se agrega el 0 es el ivan y si se agrega el 1 es el nahue
        #Como va cambiando de true a false en cada foto, eso hace q me cambie el valor
        if person:
            listaImagenAuxiliar.append(int(0))
            person = False
        else:
            listaImagenAuxiliar.append(1)
            person = True

        imagen.append(cp.deepcopy(listaImagenAuxiliar))

    return imagen
```

main.py

```python
from distutils.dir_util import copy_tree
import random
from capaOculta import *
from capaFinal import *
import matplotlib.pyplot as plt
from leerImagenes import *
def main():
    iteraciones = 0
    contador = []
    tabla_imagen = leer_imagenes()
    cantCO = int(input("Cantidad de neuronas para la capa oculta: "))
    entradas = len(tabla_imagen[0])-1 #7681 entradas
    print(len(tabla_imagen))
    entradasCO = cantCO* entradas

    listaPesosCO = []
    listaPesosCF = []


    for i in range(entradasCO):
        rCO = random.uniform(-1, 1)
        listaPesosCO.append(rCO)

    division = [listaPesosCO[i:i+entradas] for i in range(0, len(listaPesosCO), entradas)] #[[7681], [7681],...]

    for i in range(cantCO+1):
        rCF = random.uniform(-1, 1)
        listaPesosCF.append(rCF)

    listaSalidaReal = []
    lista_errores = [ [] for i in range(len(tabla_imagen))]
```

```python
while iteraciones != 100:
    iteraciones += 1
    contadorError = 0
    contador.append(iteraciones)
    print(f"----------------------ITERACION N°{iteraciones}----------------------")
    for fila in tabla_imagen:
        cortada = fila[:-1] #posicion 0 - 7681
        salidaDeseada = fila[-1] #posicion 7682
        listaSalidaReal.append(1)
        for capaoculta in division:
            salidaRealCO = CapaOculta().neuronaCapaOculta(cortada, capaoculta)
            listaSalidaReal.append(salidaRealCO)

        deltaFinal, nuevosValores, error = CapaFinal().neuronaCapaFinal(listaSalidaReal, listaPesosCF, salidaDeseada)
        lista_errores[contadorError].append(error)
        contadorError += 1



        listaPesosCF.clear()
        listaPesosCF = nuevosValores


        nuevosPesosCO = []
        lr=0.5
        for peso in range(len(division)):
            Soc_genaral =listaSalidaReal[peso]*(1-listaSalidaReal[peso])*deltaFinal
            for c in range(len(cortada)):
                delta_general = lr*cortada[c]*Soc_genaral
                w_general = division[peso][c] + delta_general
                nuevosPesosCO.append(w_general)
        listaSalidaReal.clear()
        division.clear()
        division = [nuevosPesosCO[i:i+entradas] for i in range(0, len(nuevosPesosCO), entradas)]
```

```
for i in range(len(lista_errores)):
    plt.plot(contador, lista_errores[i], label = f"Error r{i}")
plt.title("ERRORES")
plt.legend()
plt.show()
plt.savefig("grafico_errores.png")
```

capaOculta.py

```
import math
import numpy as np
class CapaOculta():
    def neuronaCapaOculta(self,filaXOR, listaPesos ):
        x = np.multiply(filaXOR, listaPesos)
        sumatoria = sum(x)
        salidaReal = 1 / (1 + (np.exp(-sumatoria)))
        #print(f"Salida real {salidaReal}")
        return salidaReal
```

capaFinal.py

```
capaFinal.py > CapaFinal > neuronaCapaFinal
import math
import numpy as np
class CapaFinal():
    def neuronaCapaFinal(self, salidaRealCO, listaPesosCF, sd):

        x = 0
        for i in range(len(listaPesosCF)):
            #print(listaPesosCF[i])
            x += (listaPesosCF[i]*salidaRealCO[i])



        salidaReal = 1 / (1 + (np.exp(-x)))
        error = sd - salidaReal
        deltaFinal = salidaReal*(1-salidaReal)*error

        nuevosPesos = []
        lr = 0.5


        for i in range(len(salidaRealCO)):
            delta_w2 = lr*salidaRealCO[i]*deltaFinal
            w2 = listaPesosCF[i] + delta_w2
            nuevosPesos.append(w2)
        #print("\nNeurona 4")
        print(f"SALIDA REAL CAPA FINAL: {salidaReal}")

        return deltaFinal, nuevosPesos, error
```

→ Seleccionamos 100 neuronas para la capa oculta y 100 iteraciones

```
cultad/CURSADO/4ANO/segundo semestre/inteligenciaArtificial/TP6/main.py"
Cantidad de neuronas para la capa oculta: 100
```

Podemos observar cómo aprende nuestro programa, ya que vemos como, intercaladamente, los valores de la salida real de la última neurona se acercan a los valores de su salida deseada (0 y 1)

```
----------------------ITERACION N°100----------------------
SALIDA REAL CAPA FINAL: 0.040342716289636 95
SALIDA REAL CAPA FINAL: 0.9611465510084598
SALIDA REAL CAPA FINAL: 0.022896427039027792
SALIDA REAL CAPA FINAL: 0.952568804213462
SALIDA REAL CAPA FINAL: 0.04933881150902368
SALIDA REAL CAPA FINAL: 0.9593744577372428
SALIDA REAL CAPA FINAL: 0.022608969663310993
SALIDA REAL CAPA FINAL: 0.9813127215285182
SALIDA REAL CAPA FINAL: 0.03612158702962904
SALIDA REAL CAPA FINAL: 0.9785605628727786
```

GRAFICO DE ERRORES