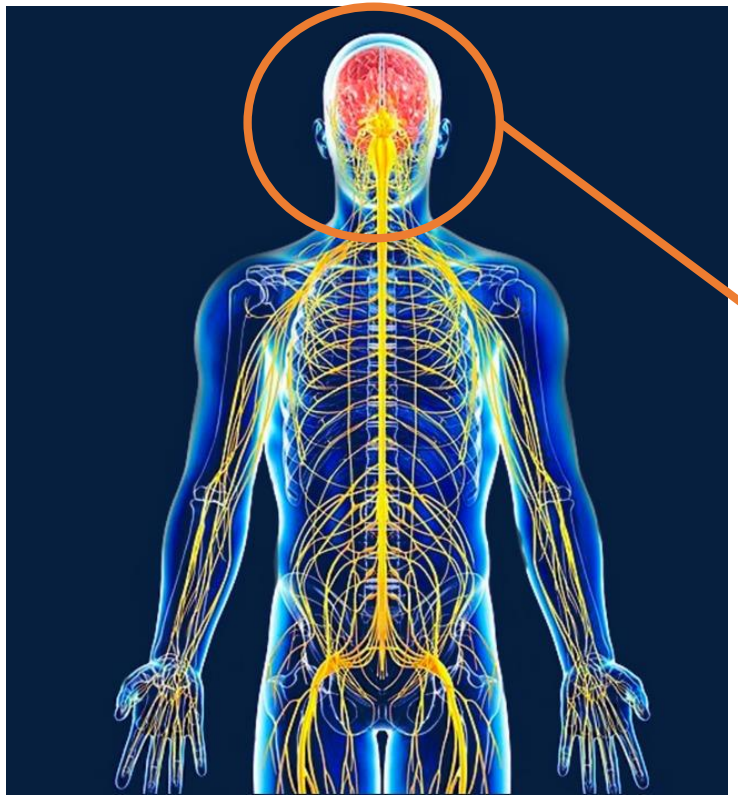
TRABAJO FINAL: RECONOCIMIENTO DE EMOCIONES

Materia Optativa: Introducción a Sistemas Inteligentes
Profesores: Mg. Rosa Macaione – Lic. Carlos Ismael Orozco
Alumno: Barboza Héctor Emanuel

Sistema Nervioso y el Cerebro



Es:

- Complejo
- No Lineal
- Paralelo

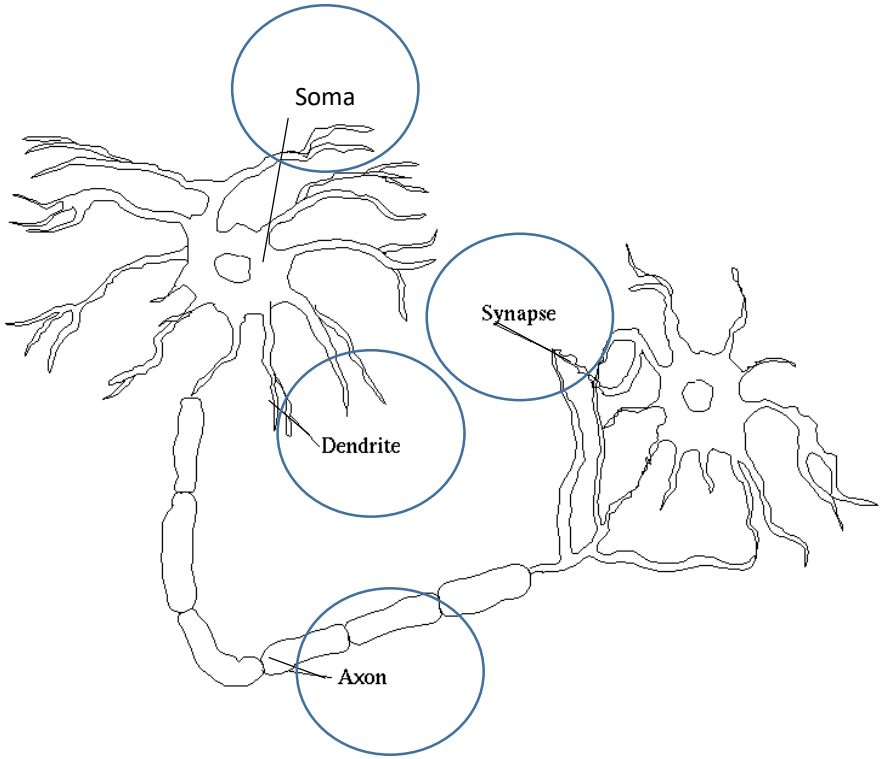
¿QUE SON LAS NEURONAS?

Comparación con Redes Neuronales Humanas

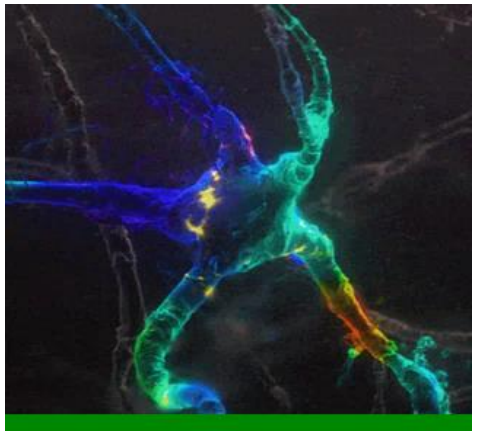
Se estima que el cerebro humano contiene mas de cien mil millones de neuronas y sinapsis en el sistema nervioso humano.

Tiene los Siguientes componentes:

- ✓ Soma o cuerpo
- ✓ Sinapsis
- ✓ Dentritas
- ✓ Axón



Redes Neuronales Artificiales (RNA) o Artificial Neural Network (ANN)



Es un modelo computacional basado en un conjunto de neuronas que emulan el funcionamiento del cerebro humano.

Es un paradigma o un nuevo sistema de procesamiento y de tratamiento de la información compuesta por neuronas Artificiales

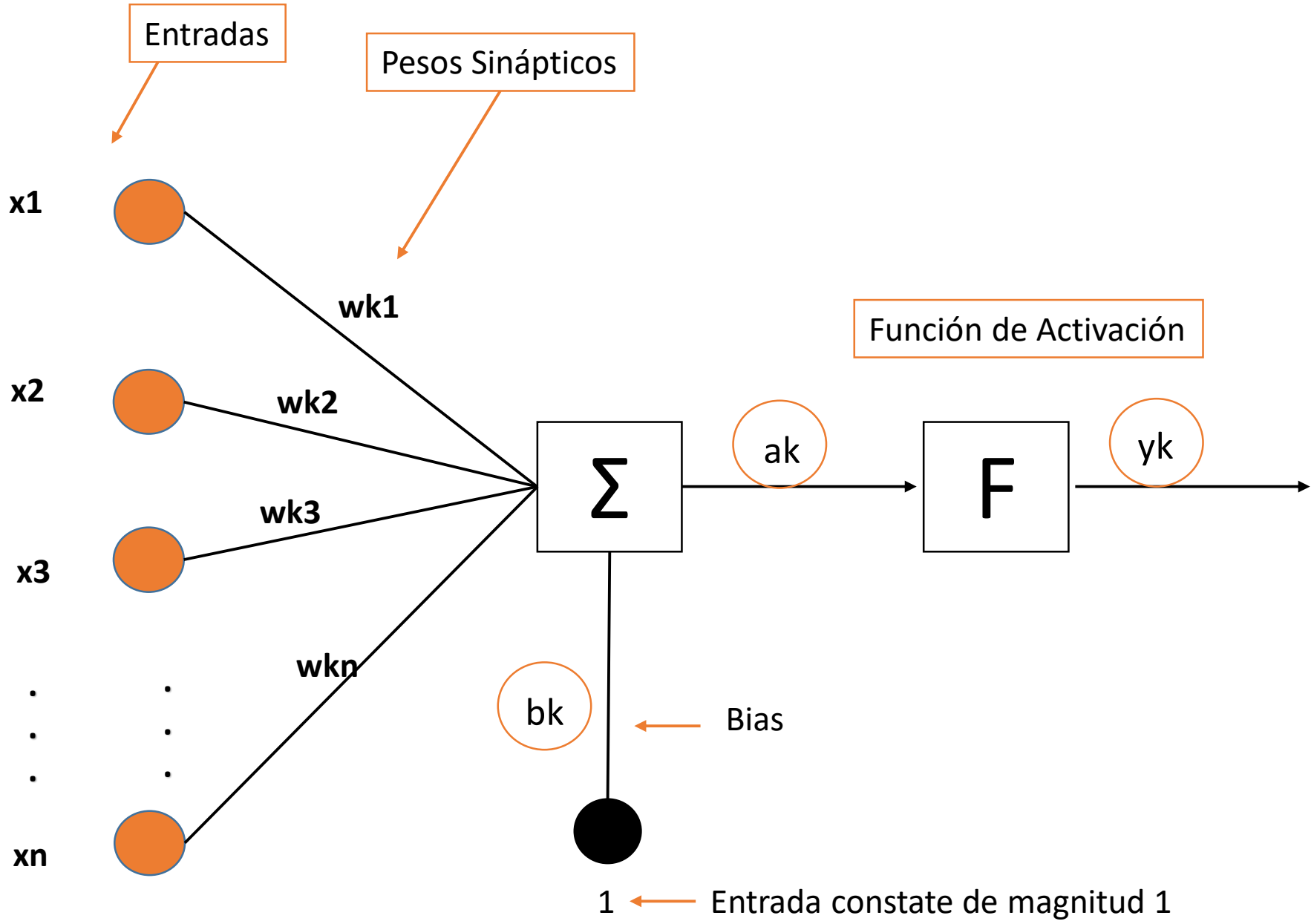
Una red neuronal es apta para resolver problemas que no tienen un algoritmo claramente definido para transformar una entrada en una salida.

Esta inspirado en el en el Sistema Neuronal o Nervioso Biológico.

1 NURONA → Es un componente relativamente Simple
Miles de Millones de NEURONAS → Forman un poderoso Sistema

Características:

- ❑ Aprendizaje Adaptable
  - Red en Entrenamiento
  - Red Entrenada
  - Conocimiento Adquirido
- ❑ Auto-Organización
- ❑ Tolerancia a Fallos
- ❑ Flexibilidad
- ❑ Tiempo Real


Entradas
Pesos Sinápticos
x1
x2
x3
xn
wk1
wk2
wk3
wkn
Función de Activación
Σ
ak
F
yk
bk
Bias
1
Entrada constate de magnitud 1

$$\text{Sk} = wk1 * x1 + wk2 * x2 + \cdots + wkn * xn$$

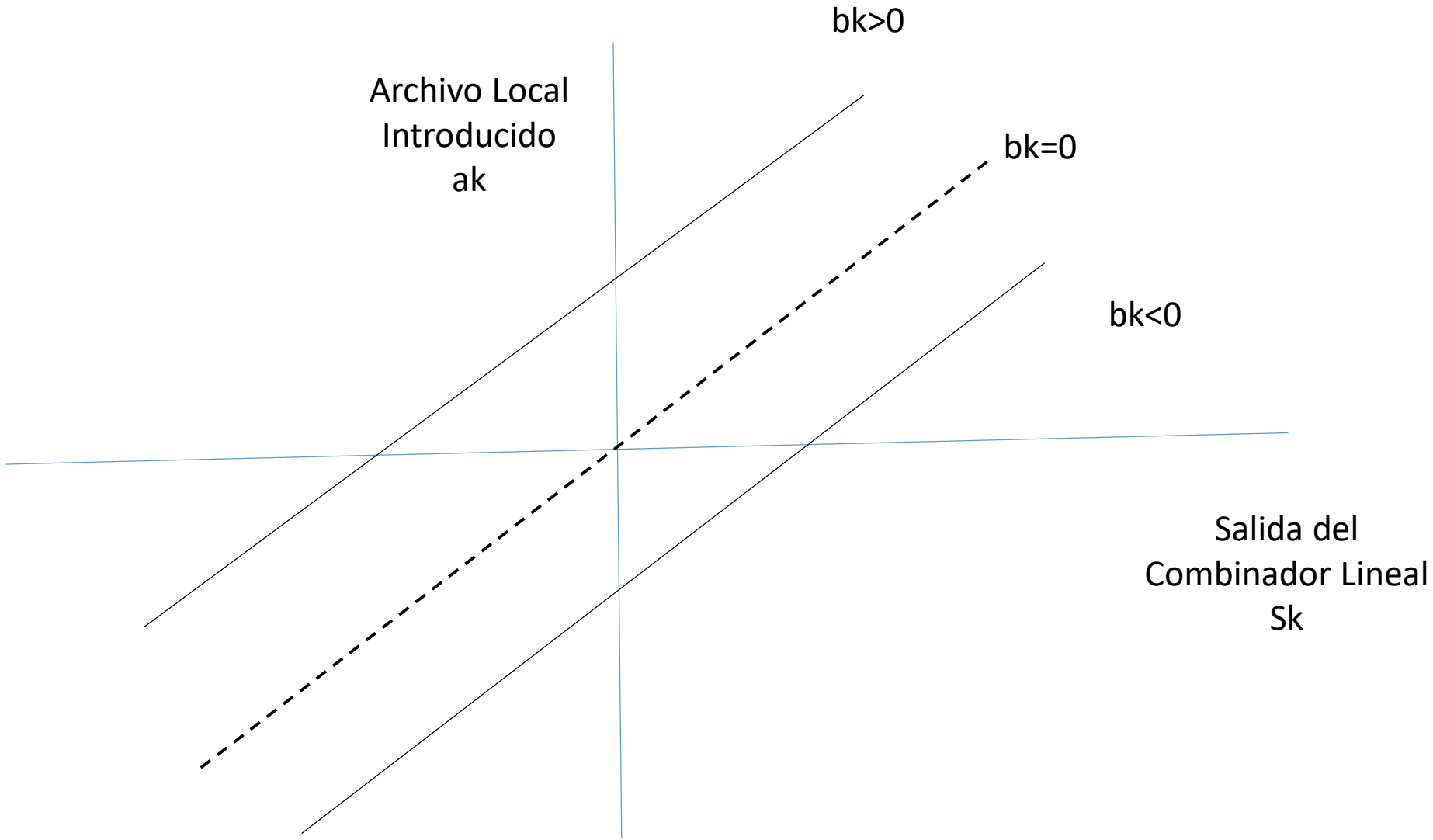
$$\text{Sk} = \sum_{i=1}^{n} wki * xi$$

$$\text{ak} = \sum_{i=1}^{n} wki * xi + bk$$

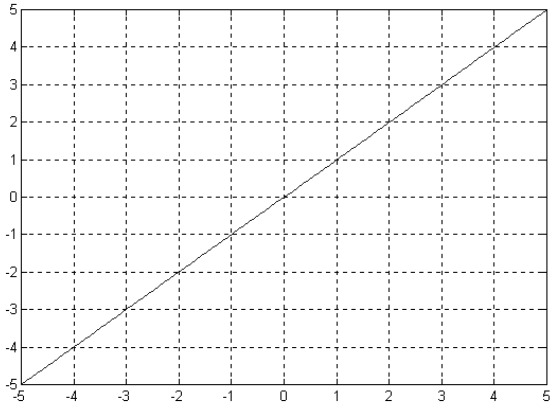
$$\text{ak} = \sum_{i=0}^{n} wki * xi$$

$$\text{ak} = Sk + bk$$

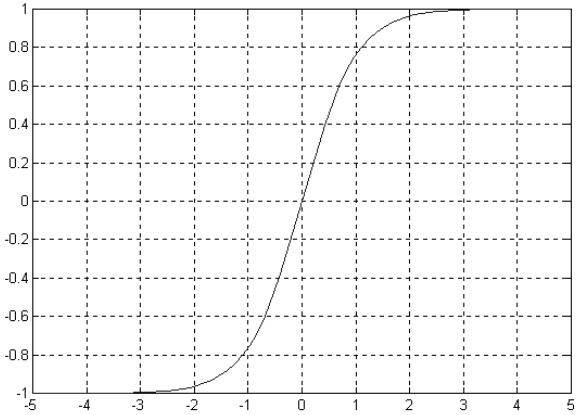
$$\text{yk} = F(ak)$$


bk>0
Archivo Local
Introducido
ak
bk=0
bk<0
Salida del
Combinador Lineal
Sk

Lineal : y=k*a

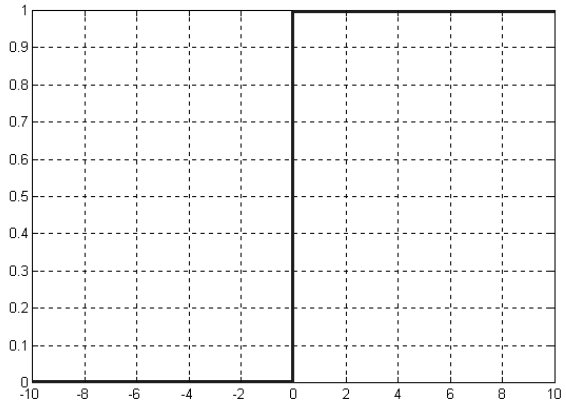


Sigmoide

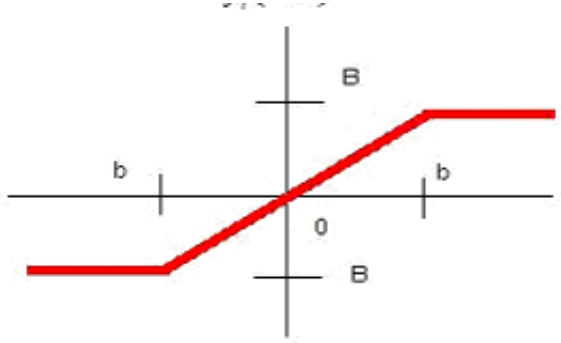


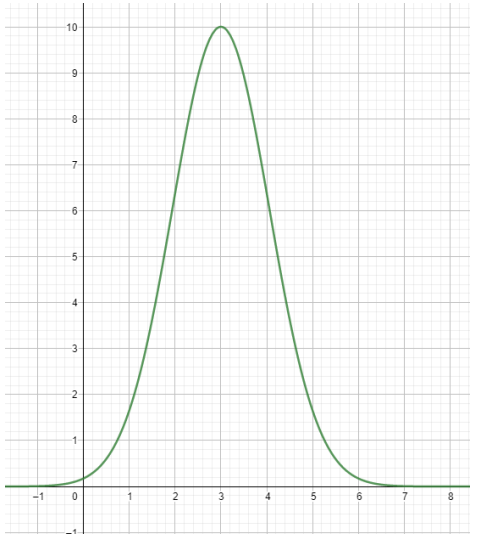
Escalón: y=0 si a<0

y=1 si a>=0



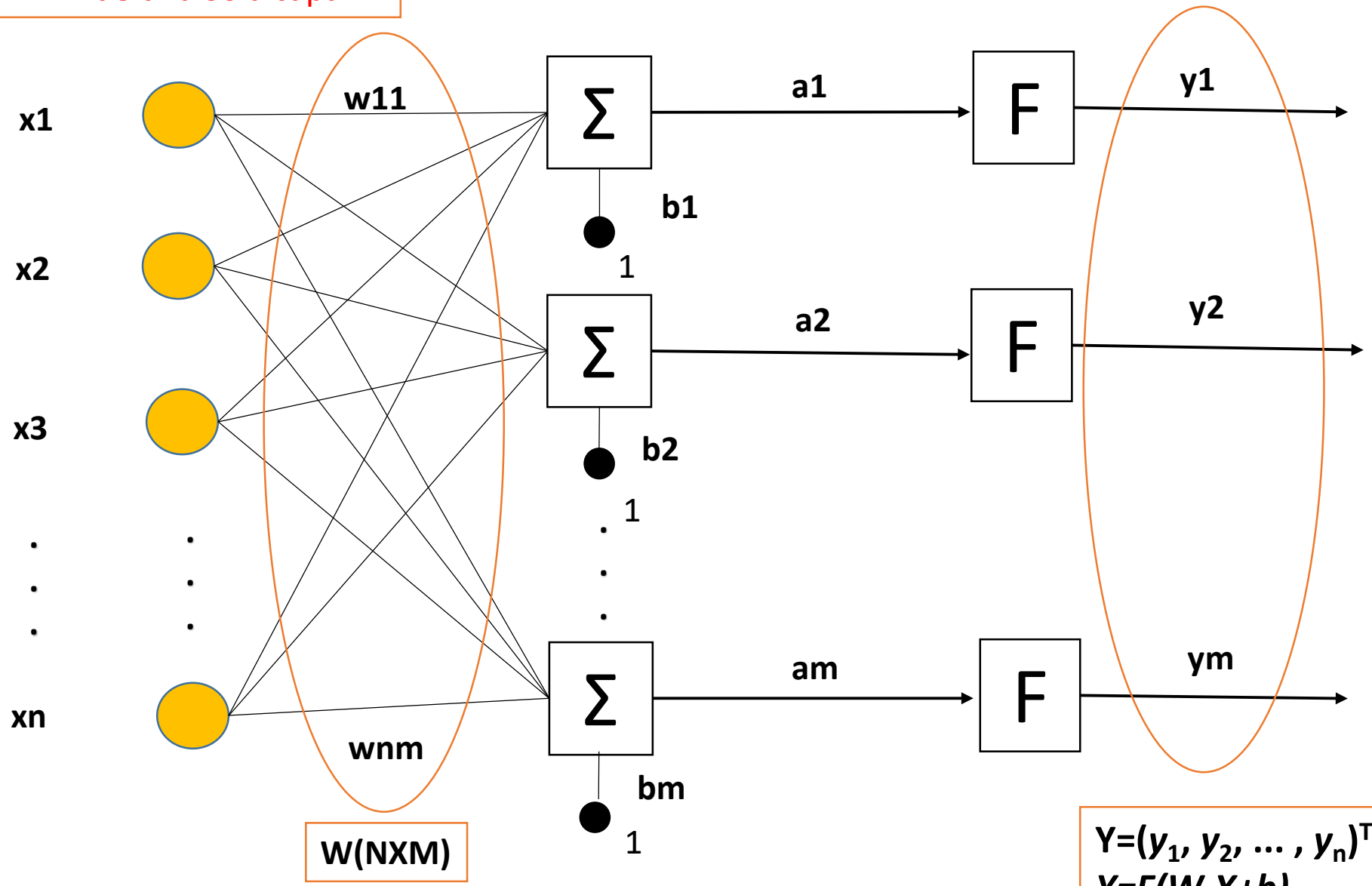
Lineal Mixta





Gaussiana

RNA de una Sola capa



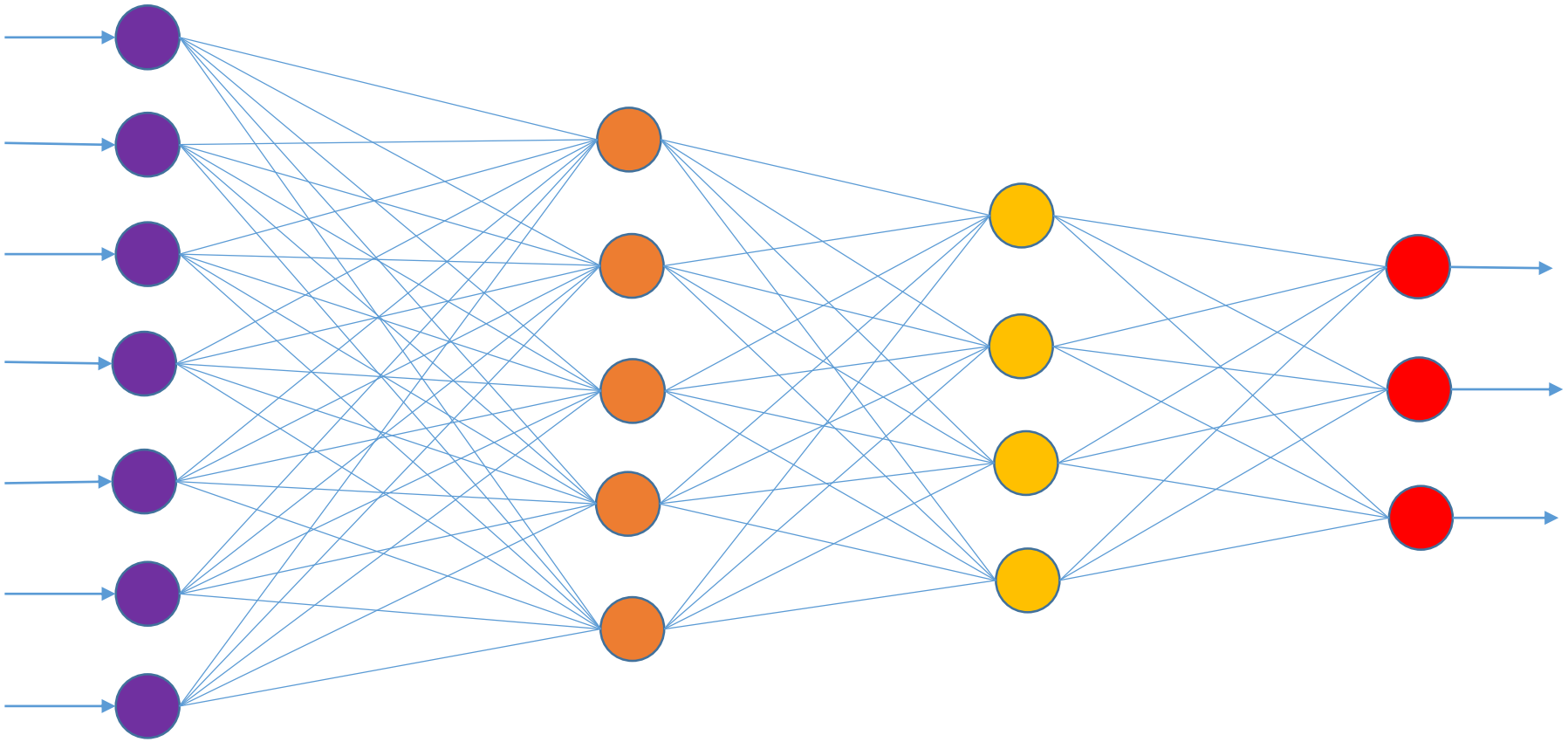
RNA Multicapa

Las neuronas se interconectan en tres tipos de capas:

Capa de entrada

Capas Ocultas

Capa de salida

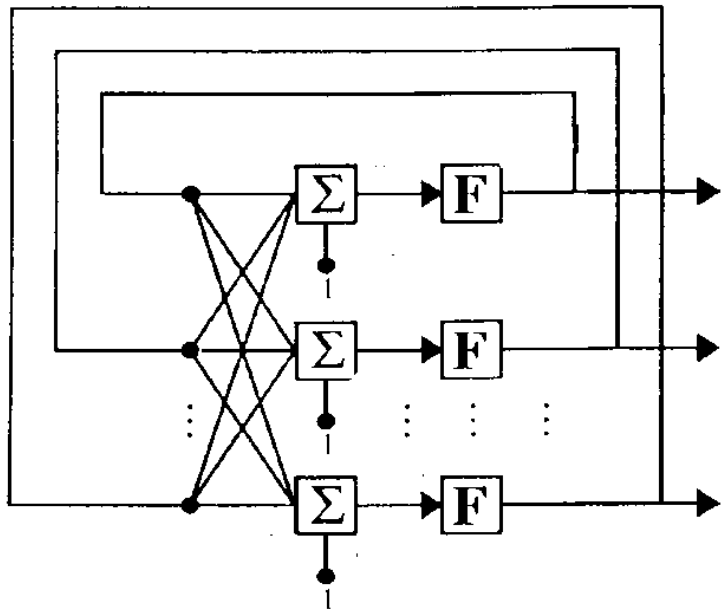


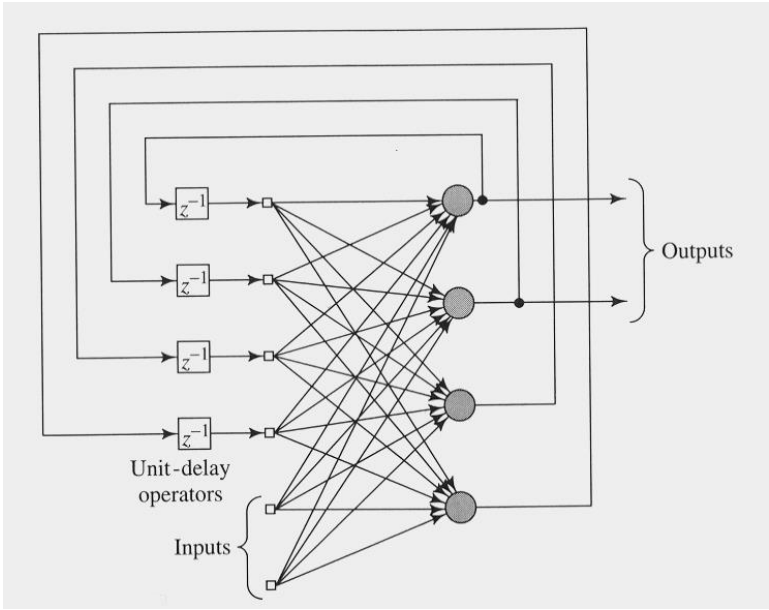
❑ ***Redes Feedforward*** *:*

Las mas conocidas son:

- ✓ Perceptron
- ✓ Adaline
- ✓ Madaline
- ✓ Backpropagation

❑ ***Redes Feedback:***





Tipo de Entrenamiento

- **Supervisado:**

En este caso un experto asesora el entrenamiento

Pares de entrenamiento: entrada - salida deseada.

Error por cada par que se utiliza para ajustar parámetros

- **No supervisado:**

Solamente conjunto de entradas.

Salidas: la agrupación o clasificación por clases

- **Reforzado o por esfuerzos:**

Premios y castigos

Armando una topología

Cuando armamos una topología de una red neuronal hay que tener en cuenta algunas cosas como por ejemplo:

- ✓ El numero de capas
- ✓ El numero de neuronas por capa
- ✓ Tipo de conexiones. Normalmente, todas las neuronas de una capa reciben señales de la capa anterior (más cercana a la entrada) y envían su salida a las neuronas de la capa posterior (más cercana a la salida de la red).
- ✓ Tenemos que definir el entrenamiento. La creación de una conexión implica que el peso de la misma pasa a tener un valor distinto de cero.
  Una conexión se destruye cuando su valor pasa a ser cero.

- ✓ Se dice que una red converge si la salida de la red es la salida deseada

Aplicaciones

Las aplicaciones mas comunes que podemos tener en su uso son

- Reconocimiento y clasificación de patrones
- Diagnostico de imágenes medicas de tejidos cancerígenos y sanos.
- Análisis de tendencias de datos
- Pronostico de ventas
- Control de Procesos Industriales
- Administración de Riesgo en ventas
- Investigación sobre clientes

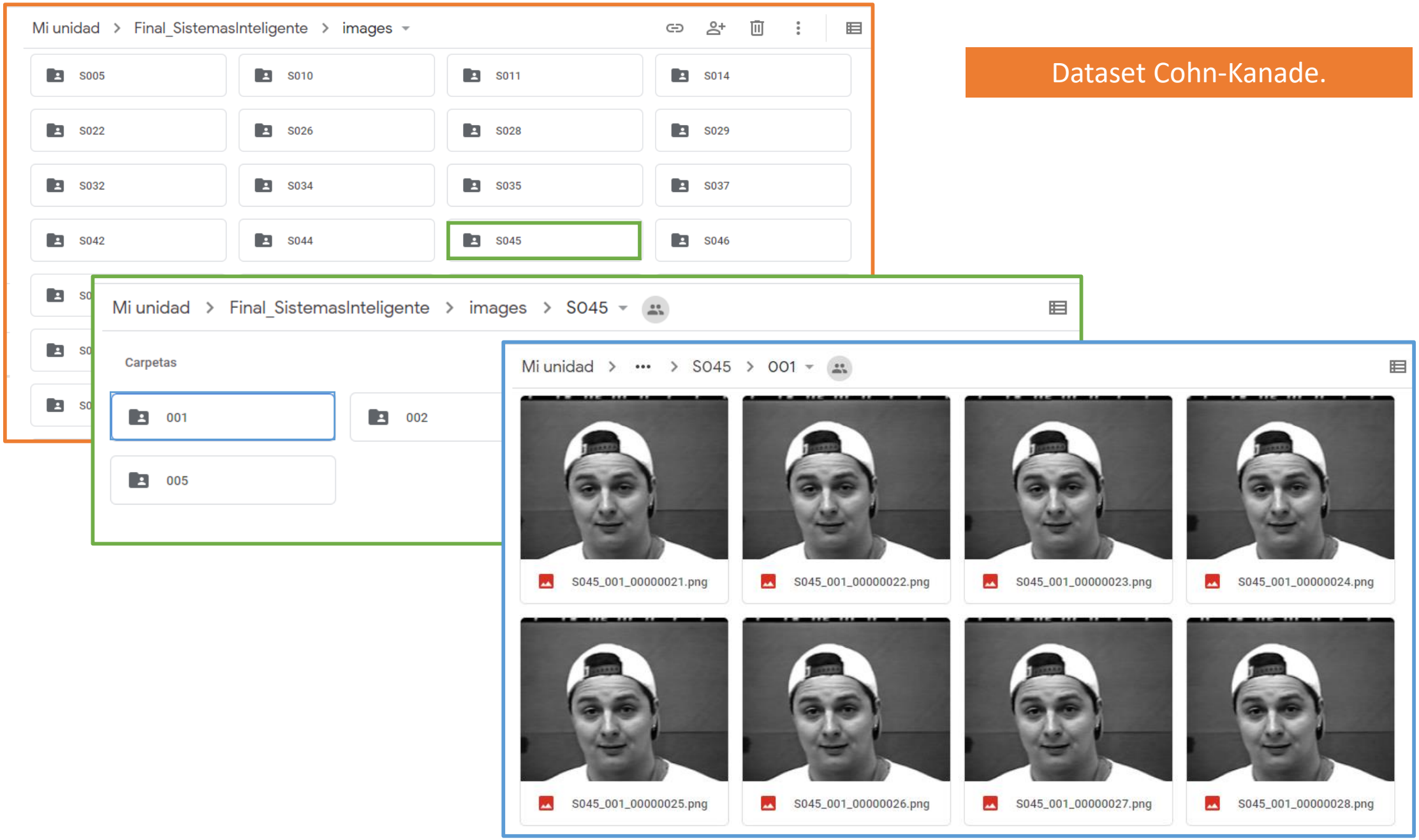
El reconocimiento automático de emociones en personas es un tema de interés en el área de visión por computadora debido a sus potenciales aplicaciones.

Tales como:

- Interacción Humano- Maquina
- Evaluación del dolor en atención Medica
- Detección de engaños
- Seguimiento de comportamiento en niños
- Detección de perdida de atención en la conducción de autos.
- Entre otras

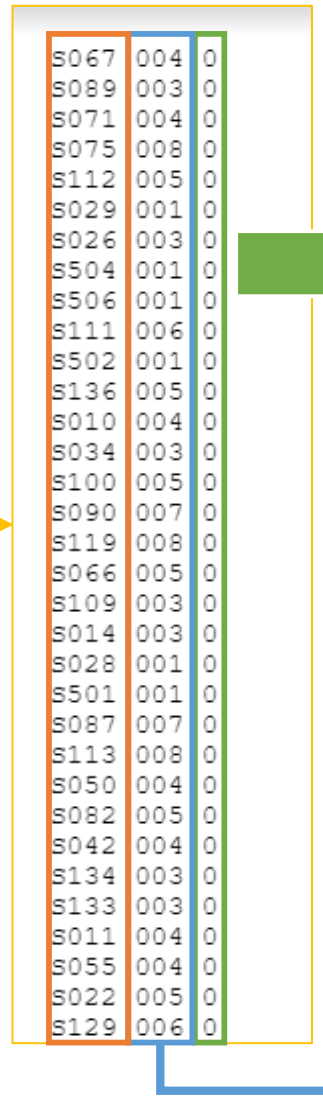
Objetivo: Desarrollar un prototipo capaz de realizar reconocimiento de emociones en imágenes utilizando un enfoque de redes neuronales tradicionales multicapa.

Dataset Cohn-Kanade.



Preparación de la Base de Datos

Archivo llamado dataset.txt



Representa una clasificación de Emociones

0. *Enojo*
1. *Desprecio*
2. *Disgusto*
3. *Miedo*
4. *Alegría*
5. *Tristeza*
6. *Sorpresa*

De las n imágenes:

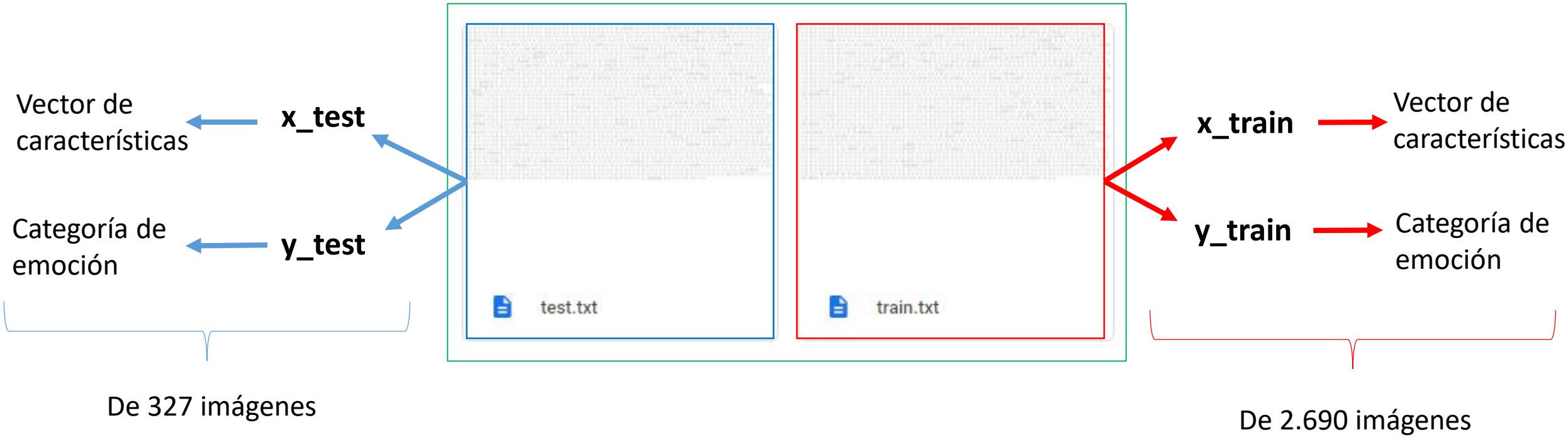
- ✓ **50% ->** La mitad de las primeras imágenes no usamos
- ✓ **50% ->** La mitad de las segundas imágenes usaremos para **TRAIN** y la penúltima imagen usamos para **TEST**

Extracción de Características de una Imagen

```
from keras.applications.vgg16 import preprocess_input
from keras.applications.vgg16 import VGG16
import cv2 as cv
from google.colab.patches import cv2_imshow
```

```
[ ]  #vamos a extraer las caracteristicas de las imagenes y las retornamos
     class Extractor():
         def __init__(self):
             self.modelo = VGG16(weights='imagenet', include_top=False)

         def extract(self, image):
             image = cv.cvtColor(image, cv.COLOR_BGR2RGB).astype(np.float32)
             image = cv.resize(image, (224, 224), interpolation = cv.INTER_LINEAR)
             image = np.array(image).reshape((1, image.shape[0], image.shape[1], image.shape[2]))
             image = preprocess_input(image)
             features = self.modelo.predict(image)
             features = features.flatten()
             return features
```

Entrenamiento Supervisado



Los archivos generados anteriormente en donde se extrajeron las características y se entreno el modelo a partir de los archivos generados se realizaron a través de dos maneras:

- Una de las primeras manera se realizo a través de **Google Colaboratory** que es un entorno de máquinas virtuales basado en Jupyter Notebooks. Se pueden correr en la nube, es posible elegir correr nuestro notebook en una CPU, GPU o en una TPU de forma gratuita.
  En nuestro caso se realizo con Python 3 y en una GPU.
  Tiempo de duración : 40 mint.

- Otra de las maneras que se realizo es a través de manera local con el entorno de desarrollo **PyCharm** Community 2019.1 en donde se realizo con Python 3 y con CPU con procesador Intel Core i7 y teniendo memoria RAM de 8 GB.
  Tiempo de duración: 2 hs y 35 mint.

Armamos Modelo de Red Neuronal Multicapa (MLP)

```
[ ]   model = Sequential()

      model.add(Dense(1470, input_dim=n, activation='relu'))
      model.add(Dense(735, activation='relu'))
      model.add(Dense(368, activation='relu'))
      #model.add(Dropout(0.3))
      model.add(Dense(92, activation='relu'))
      model.add(Dense(7, activation='sigmoid'))


      #Vamos a entrenar la red en base al Modelo definido anteriormente
      print("\nEntrenando ...")
      epocas = 100
      #lrate = 0.01
      #decay = lrate / epocas
      #sgd = SGD(lr=lrate, momentum=0.9, decay=decay, nesterov=False)
      model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

      model.fit(x_train, y_train, epochs=epocas, batch_size=20)

      #Evaluamos con el x,y tests
      score = model.evaluate(x_test, y_test)

      y_pred = model.predict(x_test)

      print('\n>>>Esto es exactitud: \n %s: %.2f%%' % (model.metrics_names[1], score[1]*100))
      print (model.predict(x_test).round())
```

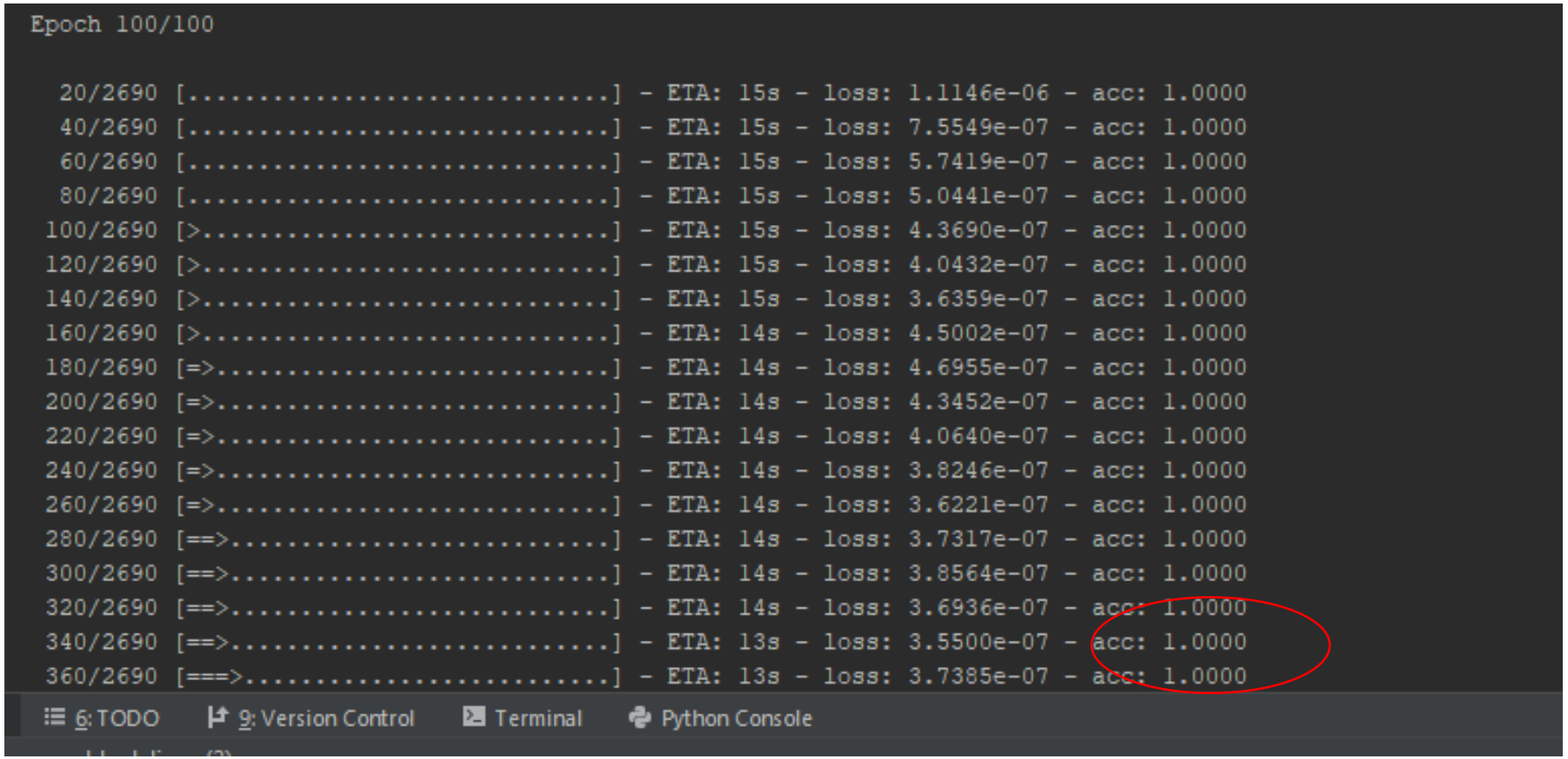
```
model = Sequential()

model.add(Dense(1470, input_dim=n, activation='relu'))
model.add(Dense(735, activation='relu'))
model.add(Dense(368, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(92, activation='relu'))
model.add(Dense(7, activation='sigmoid'))


epocas = 100
#lrate = 0.01
#decay = lrate / epocas
#sgd = SGD(lr=lrate, momentum=0.9, decay=decay, nesterov=False)

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

model.fit(x_train, y_train, epochs=epocas, batch_size=20)

score = model.evaluate(x_test, y_test)

print('esto es exactitud: \n %s: %.2f%%' % (model.metrics_names[1], score[1]*10))
```

En Google Colaboraty…

```
[120] #cargarImagenes()
      ModeloSecuencial()
```

```
Epoch 79/100
2690/2690 [==============================] - 1s 385us/step - loss: 6.8679e-07 - acc: 1.0000
Epoch 80/100
2690/2690 [==============================] - 1s 387us/step - loss: 6.1101e-07 - acc: 1.0000
Epoch 81/100
2690/2690 [==============================] - 1s 385us/step - loss: 5.3886e-07 - acc: 1.0000
Epoch 82/100
2690/2690 [==============================] - 1s 382us/step - loss: 4.9116e-07 - acc: 1.0000
Epoch 83/100
2690/2690 [==============================] - 1s 381us/step - loss: 4.5256e-07 - acc: 1.0000
Epoch 84/100
2690/2690 [==============================] - 1s 384us/step - loss: 4.1205e-07 - acc: 1.0000
Epoch 85/100
2690/2690 [==============================] - 1s 382us/step - loss: 3.8313e-07 - acc: 1.0000
Epoch 86/100
2690/2690 [==============================] - 1s 384us/step - loss: 3.6900e-07 - acc: 1.0000
Epoch 87/100
2690/2690 [==============================] - 1s 381us/step - loss: 3.5118e-07 - acc: 1.0000
Epoch 88/100
2690/2690 [==============================] - 1s 380us/step - loss: 3.0609e-07 - acc: 1.0000
Epoch 89/100
2690/2690 [==============================] - 1s 389us/step - loss: 2.7567e-07 - acc: 1.0000
Epoch 90/100
2690/2690 [==============================] - 1s 386us/step - loss: 2.6401e-07 - acc: 1.0000
Epoch 91/100
2690/2690 [==============================] - 1s 376us/step - loss: 2.6279e-07 - acc: 1.0000
Epoch 92/100
```

En Pycharm…


Epoch 100/100
20/2690 [..............................] - ETA: 15s - loss: 1.1146e-06 - acc: 1.0000
40/2690 [..............................] - ETA: 15s - loss: 7.5549e-07 - acc: 1.0000
60/2690 [..............................] - ETA: 15s - loss: 5.7419e-07 - acc: 1.0000
80/2690 [..............................] - ETA: 15s - loss: 5.0441e-07 - acc: 1.0000
100/2690 [>.............................] - ETA: 15s - loss: 4.3690e-07 - acc: 1.0000
120/2690 [>.............................] - ETA: 15s - loss: 4.0432e-07 - acc: 1.0000
140/2690 [>.............................] - ETA: 15s - loss: 3.6359e-07 - acc: 1.0000
160/2690 [>.............................] - ETA: 14s - loss: 4.5002e-07 - acc: 1.0000
180/2690 [=>............................] - ETA: 14s - loss: 4.6955e-07 - acc: 1.0000
200/2690 [=>............................] - ETA: 14s - loss: 4.3452e-07 - acc: 1.0000
220/2690 [=>............................] - ETA: 14s - loss: 4.0640e-07 - acc: 1.0000
240/2690 [=>............................] - ETA: 14s - loss: 3.8246e-07 - acc: 1.0000
260/2690 [=>............................] - ETA: 14s - loss: 3.6221e-07 - acc: 1.0000
280/2690 [==>...........................] - ETA: 14s - loss: 3.7317e-07 - acc: 1.0000
300/2690 [==>...........................] - ETA: 14s - loss: 3.8564e-07 - acc: 1.0000
320/2690 [==>...........................] - ETA: 14s - loss: 3.6936e-07 - acc: 1.0000
340/2690 [==>...........................] - ETA: 13s - loss: 3.5500e-07 - acc: 1.0000
360/2690 [===>..........................] - ETA: 13s - loss: 3.7385e-07 - acc: 1.0000
6: TODO
9: Version Control
Terminal
Python Console

Resultados

```
--------------------------------------------------
En Carpeta de sujeto --->S071
En subcarpeta--->004
Prueba--->0
Prediccion--->0
```

ENOJO

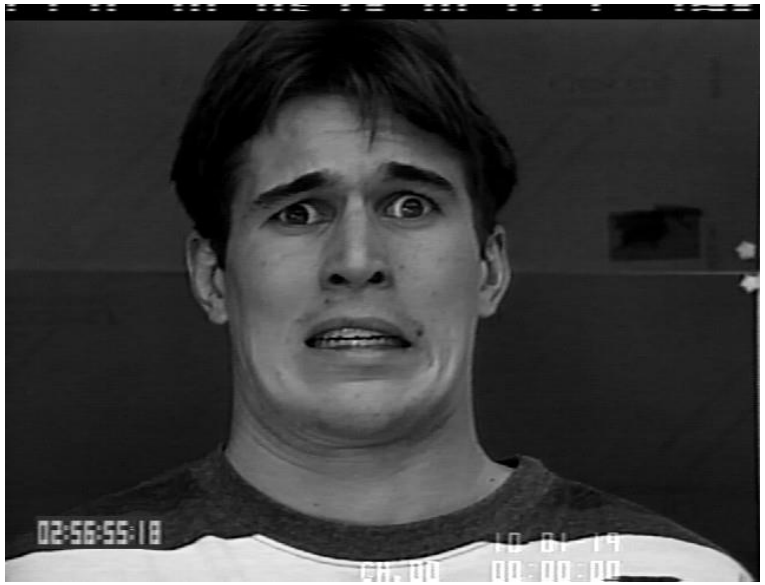
```
-----------------------------------------
En Carpeta de sujeto --->S154
En subcarpeta--->002
Prueba--->1
Prediccion--->1
```

DESPRECIO

```
----------------------------------------------------
En Carpeta de sujeto --->S067
En subcarpeta--->006
Prueba--->2
Prediccion--->2
```

DISGUSTO

```
-----------------------------------------------
En Carpeta de sujeto --->S119
En subcarpeta--->003
Prueba--->3
Prediccion--->3
```



MIEDO

```
--------------------------------------------------
En Carpeta de sujeto --->S108
En subcarpeta--->005
Prueba--->5
Prediccion--->5
```

TRISTEZA

ALEGRIA

```
----------------------------------------
En Carpeta de sujeto --->S067
En subcarpeta--->005
Prueba--->4
Prediccion--->4
```

```
----------------------------------------
En Carpeta de sujeto --->S077
En subcarpeta--->001
Prueba--->6
Prediccion--->6
```

SORPRESA

Caso en donde se realiza una mala clasificación

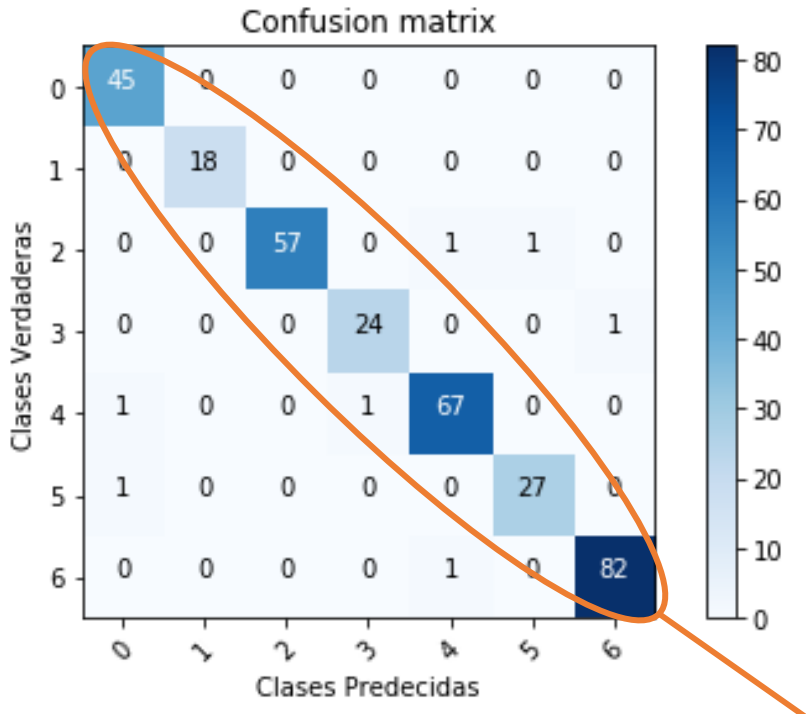
```
--------------------------------------------------
En Carpeta de sujeto --->S130
En subcarpeta--->012
Prueba--->2
Prediccion--->5
```

DISGUSTO

TRISTEZA

Evaluación del Modelo

```
>>>Esto es exactitud:
 acc: 97.86%
presicion:::--->0.9785932721712538
```




```
 32/327 [=>............................] - ETA: 1s
160/327 [=============>................] - ETA: 0s
288/327 [=========================>....] - ETA: 0s
327/327 [==============================] - 0s 802us/step

>>>Esto es presicion:
 acc: 97.55%

Process finished with exit code 0
```

$$\text{Precisión} = \frac{\text{Suma de diagonal}}{327}$$

0. *Enojo*
1. *Desprecio*
2. *Disgusto*
3. *Miedo*
4. *Alegría*
5. *Tristeza*
6. *Sorpresa*

0. *Enojo*
1. *Desprecio*
2. *Miedo*
3. *Alegría*
4. *Tristeza*
5. *Sorpresa*

```
model = Sequential()

model.add(Dense(1470, input_dim=n, activation='relu'))
model.add(Dense(735, activation='relu'))
model.add(Dense(368, activation='relu'))
#model.add(Dropout(0.3))
model.add(Dense(102, activation='relu'))
model.add(Dense(6, activation='sigmoid'))

print("\nEntrenando ...")
epocas = 100
#lrate = 0.01
#decay = lrate / epocas
#sgd = SGD(lr=lrate, momentum=0.9, decay=decay, nesterov=False)
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

model.fit(x_train

#Evaluamos con el
score = model.eva

y_pred = model.pr
```

```
Epoch 89/100
2690/2690 [==============================] - 1s 374us/step - loss: 6.0151e-07 - acc: 1.0000
Epoch 90/100
2690/2690 [==============================] - 1s 374us/step - loss: 5.7027e-07 - acc: 1.0000
Epoch 91/100
2690/2690 [==============================] - 1s 365us/step - loss: 5.5628e-07 - acc: 1.0000
Epoch 92/100
2690/2690 [==============================] - 1s 368us/step - loss: 5.3443e-07 - acc: 1.0000
Epoch 93/100
2690/2690 [==============================] - 1s 376us/step - loss: 5.0099e-07 - acc: 1.0000
Epoch 94/100
2690/2690 [==============================] - 1s 381us/step - loss: 4.8169e-07 - acc: 1.0000
Epoch 95/100
2690/2690 [==============================] - 1s 383us/step - loss: 4.7659e-07 - acc: 1.0000
Epoch 96/100
2690/2690 [==============================] - 1s 380us/step - loss: 4.2777e-07 - acc: 1.0000
Epoch 97/100
2690/2690 [==============================] - 1s 381us/step - loss: 4.0559e-07 - acc: 1.0000
Epoch 98/100
2690/2690 [==============================] - 1s 370us/step - loss: 3.9316e-07 - acc: 1.0000
Epoch 99/100
2690/2690 [==============================] - 1s 374us/step - loss: 3.6282e-07 - acc: 1.0000
Epoch 100/100
2690/2690 [==============================] - 1s 374us/step - loss: 3.5859e-07 - acc: 1.0000
327/327 [==============================] - 0s 520us/step
```

```
[ ]
for i in range (0,len(y_train),1):
  y_train[i]=int(y_train[i])

  if(y_train[i]==0):
    y_train[i]=0
  elif((y_train[i]==1) or (y_train[i]==2)):
    y_train[i]=1
  else:
    y_train[i]=y_train[i]-1

y_train=np_utils.to_categorical(y_train)

x_test=np.array(x_test)
for i in range (0,len(y_test),1):
  y_test[i]=int(y_test[i])
  if(y_test[i]==0):
    y_test[i]=0
  elif((y_test[i]==1) or (y_test[i]==2)):
    y_test[i]=1
  else:
    y_test[i]=y_test[i]-1
for i in y_test:
    print(i)
y_test=np_utils.to_categorical(y_test)
for i in y_test:
    print(i)
```

```
>>>Esto es exactitud:
 acc: 97.25%
presicion::--->0.9724770642201835
```