Proyecto de Ingeniería: Avatar 2.0

- Bryan Gpe. Parada Medina A01332773

Asesores: Dr. Martín Rogelio Bustamante Bello

Ing. Sergio Alberto Navarro Tuch

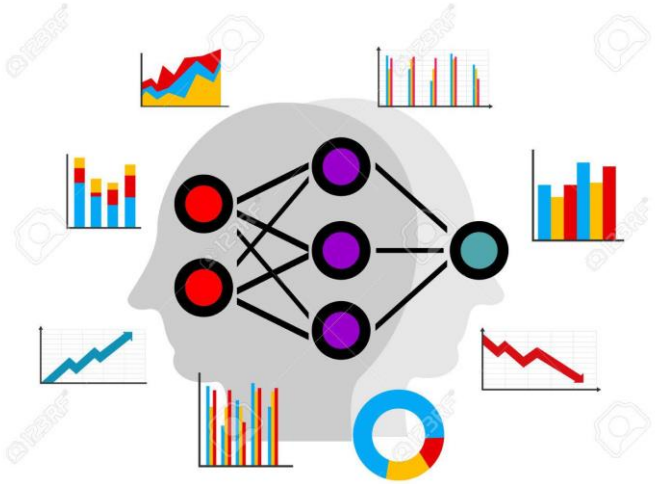
Indice

Objetivo General

- Generar una implementación de inteligencia artificial para una interfaz capaz de **detectar las emociones** de un usuario y que sea capaz de **responder** para cambiar las expresiones emocionales del mismo.

Objetivos Específicos

- Generar una metodología de entrenamiento para una red neuronal artificial programada en Python para **registrar la interacción** de emociones con un usuario a partir del impacto generado por varias animaciones y videos.



- Probar la metodología realizada con una nueva base de datos en función a la Escala SAM para determinar el rendimiento del modelo generado.

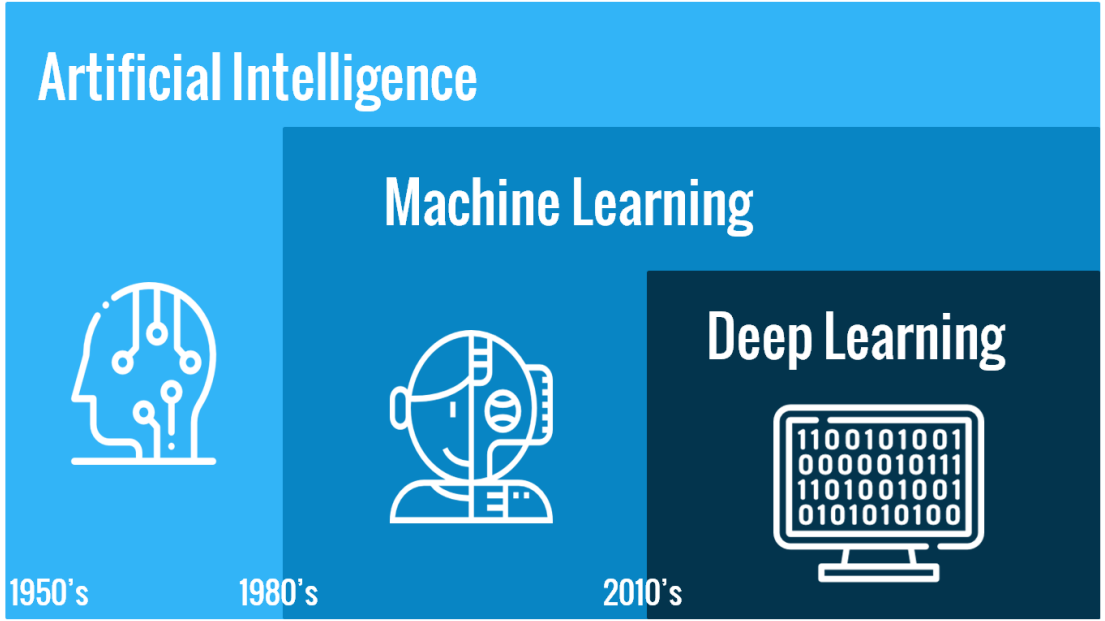
Inteligencia artificial

La Inteligencia artificial es el **campo de la informática** que se centra en la creación de programas y mecanismos que pretende suministrar a los ordenadores de capacidades de pensamiento. Tomando como base **el cerebro humano** y cómo se generan las redes neuronales que nos hacen pensar, razonar y actuar. Analizando grandes cantidades de datos **identificar patrones**, tendencias y, por lo tanto, formular predicciones de forma automática, con rapidez y precisión.

Referencia: Pagina Web de Sales Force
<https://www.salesforce.com/mx/blog/2017/6/Que-es-la-inteligencia-artificial.html>

IA

- Machine Learning: Algoritmos para muestrear grandes cantidades de datos
- Deep Learning: Modelo para evaluar diferentes ejemplos y modificar instrucciones

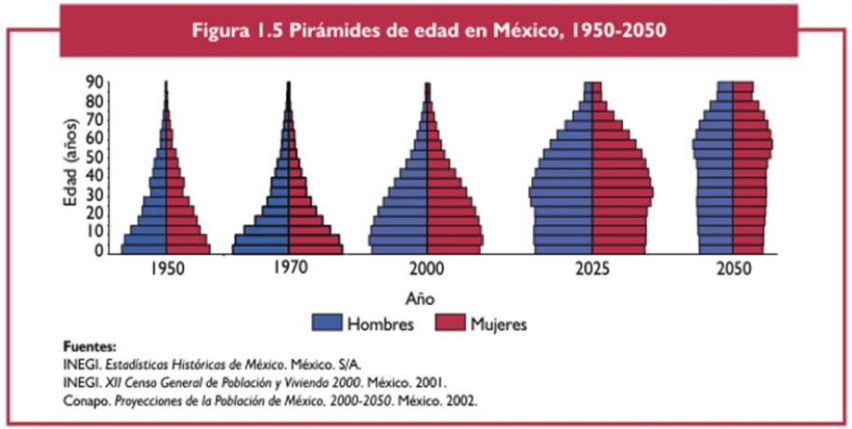


Referencia: Pagina Web de tecnología <https://www.xataka.com/robotica-e-ia/machine-learning-y-deep-learning-como-entender-las-claves-del-presente-y-futuro-de-la-inteligencia-artificial>

Referencia: Pagina web INEGI
<http://cuentame.inegi.org.mx/poblacion/habitantes.aspx?tema=P>

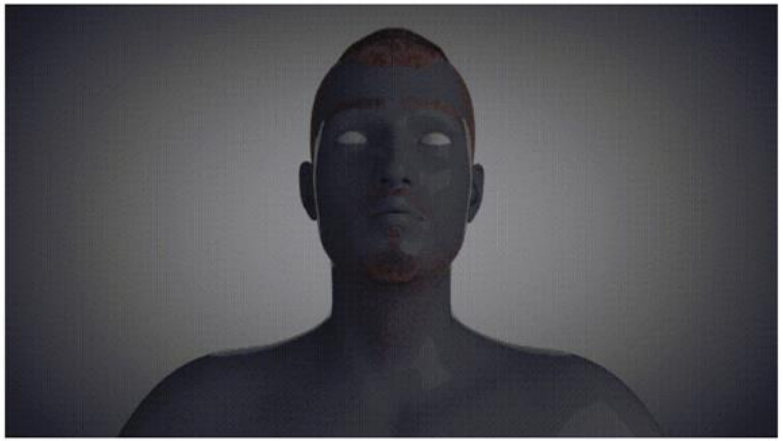
Antecedentes

- Siri
- Facebook y Google Fotos
- Piramide Poblacional
- Las emociones son estados complejos de sentimientos que influyen en nuestro desempeño diario



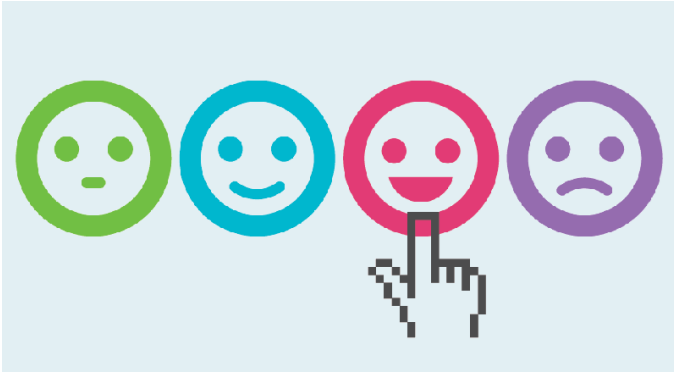
Encuestas (Base de Datos)

¿Cómo te hace sentir la siguiente animación? *



	1	2	3	4	5	6
Agrado-Desagrado	○	○	○	○	○	○
Emocionado-Calmado	○	○	○	○	○	○
Dependiente-Independiente	○	○	○	○	○	○

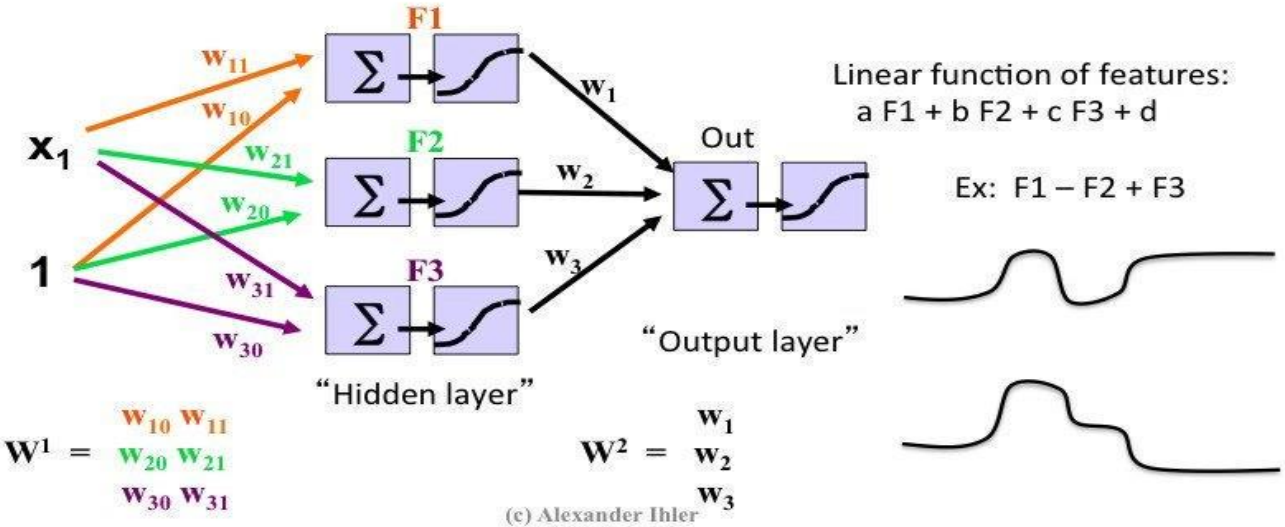
Escala SAM: Técnica de evaluación de emociones que mide directamente el placer, la excitación y el dominio asociados con la reacción afectiva de una persona ante una amplia variedad de estímulos.



Referencia: Artículo de investigación - Investigación biomédica (2017) Complex World of Neuroscience

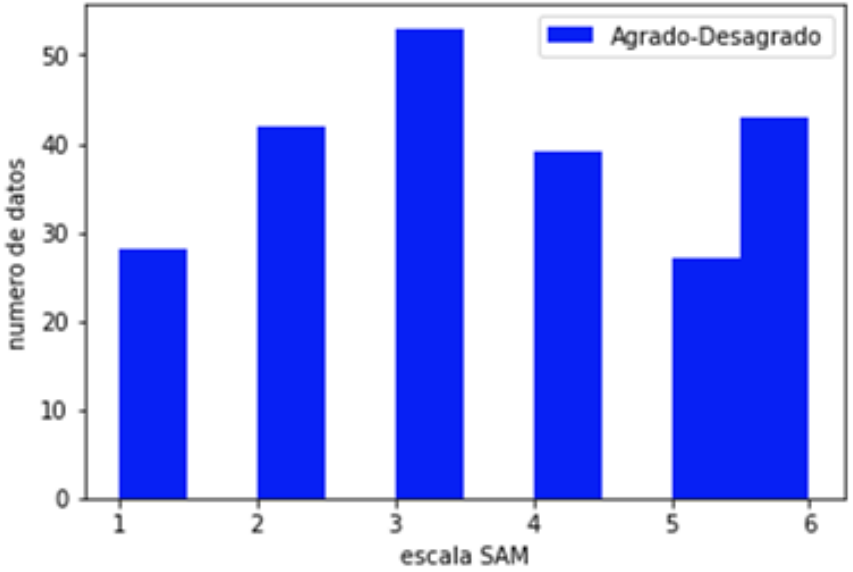
Multi-layer perceptron model

- Step functions are just perceptrons!
  - "Features" are outputs of a perceptron
  - Combination of features output of another

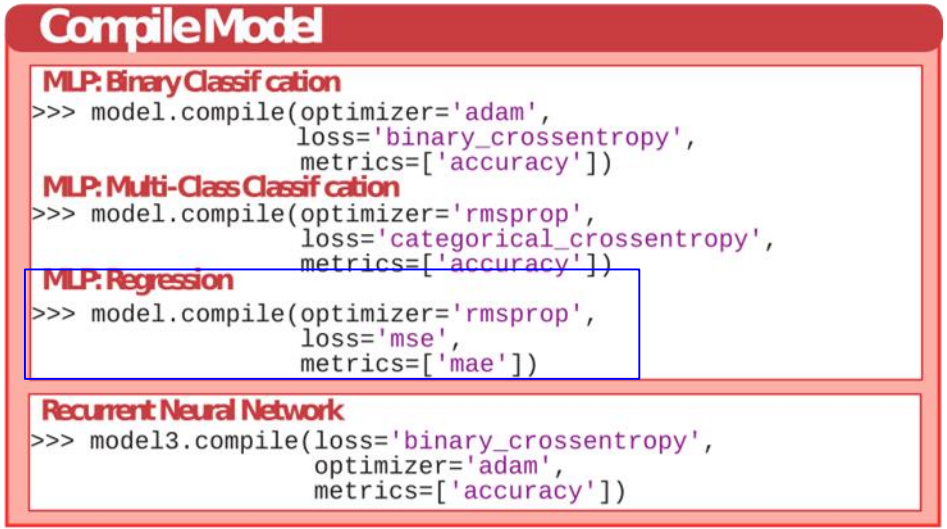


1ª fase

	Genero	Edad	Animacion	[Agrado-Desagrado]
0	1	22	1	3
1	1	22	1	3
2	1	21	1	3
3	1	22	1	2
4	1	23	1	2
5	1	21	1	3
6	1	23	1	5
7	1	23	1	2
8	0	16	1	3
9	0	22	1	4
10	1	20	1	4
11	1	22	1	5
12	1	18	1	6
13	0	21	1	4
14	0	22	1	2
15	1	18	1	4
16	0	17	1	3
17	1	21	1	2



¿Qué tipo de modelo utilizar?



Referencia: Página web oficial de la librería Keras.<https://keras.io/>

Tabla comparativa con diferentes entrenamientos

MAE	5 neuronas	10 neuronas	20 neuronas	40 neuronas
1 Capas	0.5874	0.5874	0.5777	0.5777
2 Capas	0.4433	0.4433	0.4159	0.4159
3 Capas	0.3045	0.2869	0.2779	0.2779
6 Capas	0.3045	0.2779	0.2779	0.2779

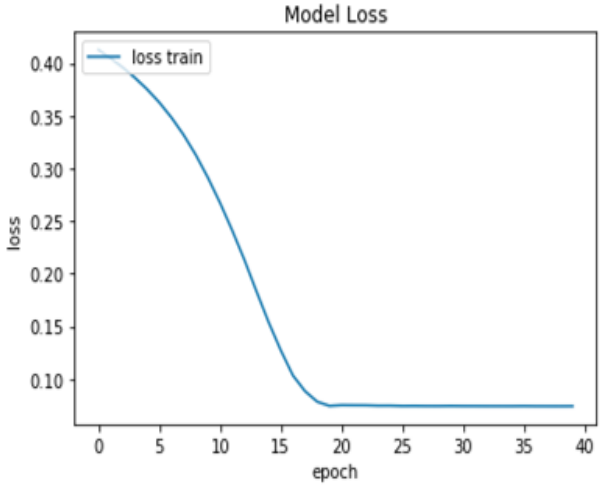
Optimizador	Mejoramiento en el modelo
‘rmsprop’	0.2779
‘sdg’	0.2779
‘adam’	0.2559

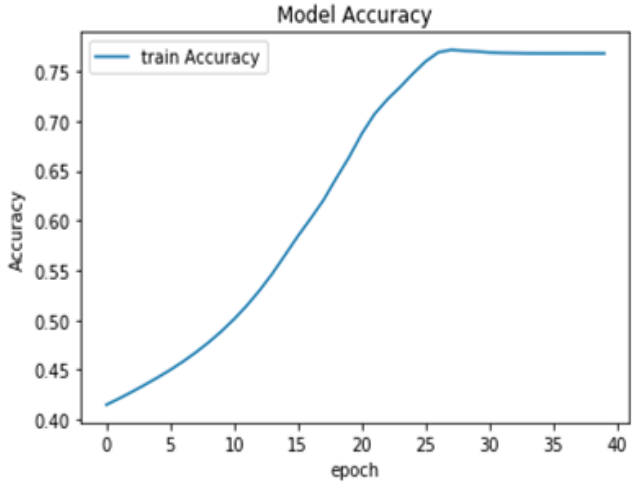
Función de Activación	Mejoramiento en el modelo
‘Sigmoid’	0.2869
‘Relu’	0.2559

Resultados de 1ª fase

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 30)	90
dense_2 (Dense)	(None, 5)	155
dense_3 (Dense)	(None, 1)	6

Total params: 251
Trainable params: 251
Non-trainable params: 0





Desarrollo 2ª fase

	Participants	Video	Edad	Genero	Consumo Alcohol	Consumo caf	Consumo Te	Horas de sueno	Nivel de Alerta	Vision	Valencia
0	1	1	31	1	0.0	0.0	0.0	8	2	0	6.96
1	2	1	24	0	0.0	0.5	1.0	8	2	1	4.96
2	3	1	19	0	0.5	0.0	0.5	7	3	1	7.15
3	4	1	24	0	0.0	1.0	0.5	8	2	1	8.17
4	5	1	24	1	0.0	0.0	0.0	8	2	1	7.00
5	6	1	23	1	0.5	0.5	0.0	8	2	0	3.65
6	7	1	31	1	1.0	0.0	0.0	8	2	1	7.12
7	8	1	22	0	0.5	0.0	1.0	8	2	0	4.23
8	9	1	25	0	0.5	0.5	1.0	8	2	1	5.05
9	10	1	31	0	1.0	1.0	0.5	7	2	0	8.99
10	11	1	27	0	0.0	1.0	0.5	7	3	0	1.00
11	12	1	37	1	0.0	1.0	0.0	8	3	1	4.01
12	13	1	24	0	1.0	1.0	1.0	8	2	0	1.97
13	14	1	27	0	0.5	0.0	1.0	7	2	1	3.99
14	15	1	22	0	1.0	0.0	0.5	6	2	0	4.05
15	16	1	28	1	0.0	0.5	1.0	7	2	1	3.64

Entrenamiento de la red neuronal

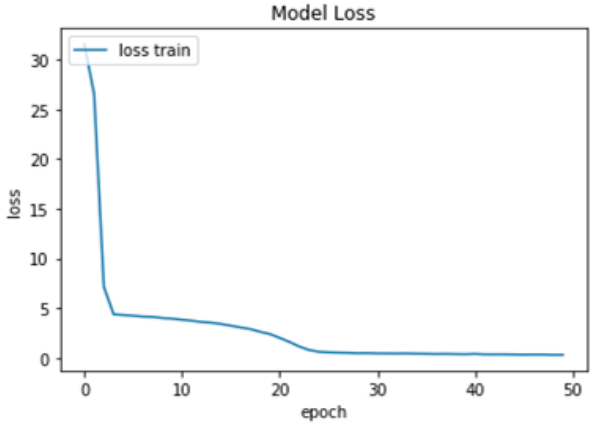
LOSS MAE	20 neuronas	40 neuronas	80 neuronas	120 neuronas
2 Capas	1.5280	1.3440	0.6297	0.5777
3 Capas	1.5280	1.3440	0.5758	0.4941
4 Capas	1.5280	1.3440	0.5452	0.4775
8 Capas	1.5280	1.3440	0.5299	0.4544

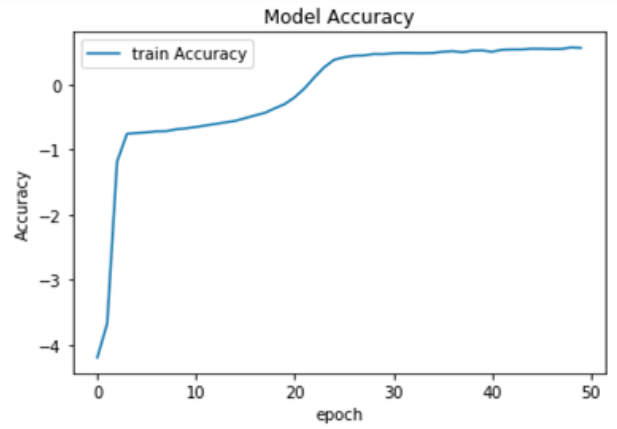
Función de Activación	Mejoramiento en el modelo
‘Sigmoid’	1.2855
‘Relu’	0.4544

Modelo de entrenamiento

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 35)	420
dense_2 (Dense)	(None, 30)	1080
dense_3 (Dense)	(None, 28)	868
dense_4 (Dense)	(None, 21)	609
dense_5 (Dense)	(None, 15)	330
dense_6 (Dense)	(None, 10)	160
dense_7 (Dense)	(None, 7)	77
dense_8 (Dense)	(None, 1)	8

Total params: 3,552
Trainable params: 3,552
Non-trainable params: 0





Resultados

	Participants	Video	Edad	Genero	Consumo Alcohol	Consumo caf	Consumo Te	Horas de sueno	Nivel de Alerta	Vision
0	1	1	31	1	0.0	0.0	0.0	8	2	0
1	2	1	24	0	0.0	0.5	1.0	8	2	1
2	3	1	19	0	0.5	0.0	0.5	7	3	1
3	4	1	24	0	0.0	1.0	0.5	8	2	1
4	5	1	24	1	0.0	0.0	0.0	8	2	1
5	6	1	23	1	0.5	0.5	0.0	8	2	0
6	7	1	31	1	1.0	0.0	0.0	8	2	1
7	8	1	22	0	0.5	0.0	1.0	8	2	0
8	9	1	25	0	0.5	0.5	1.0	8	2	1
9	10	1	31	0	1.0	1.0	0.5	7	2	0
10	11	1	27	0	0.0	1.0	0.5	7	3	0
11	12	1	37	1	0.0	1.0	0.0	8	3	1
12	13	1	24	0	1.0	1.0	1.0	8	2	0
13	14	1	27	0	0.5	0.0	1.0	7	2	1
14	15	1	22	0	1.0	0.0	0.5	6	2	0
15	16	1	28	1	0.0	0.5	1.0	7	2	1
16	17	1	25	1	0.5	1.0	1.0	7	2	0
17	18	1	29	1	0.5	0.0	1.0	8	2	0
18	19	1	27	1	0.5	0.0	1.0	7	2	1
19	20	1	25	1	0.0	0.0	0.0	8	1	1

Valencia	result
6.96	5.351799
4.96	5.111963
7.15	7.304615
8.17	2.669236
7.00	5.959931
3.65	3.228230
7.12	8.093441
4.23	4.036056
5.05	3.642924
8.99	4.599218
1.00	5.690218
4.01	5.589157
1.97	6.628652
3.99	4.426013
4.05	9.070338
3.64	6.675827
6.05	6.415900
4.23	6.609907
1.62	1.481020
6.35	2.341062

Tabla comparativa de Resultados

1ª Fase	2ª Fase
Accuracy de entrenamiento: **75%**	Accuracy de entrenamiento: **65%**
Test Score: **7%**	Test Score: **40%**
Test Accuracy: **23%**	Test Accuracy: **46%**

Conclusiones

- Se logró con éxito la realización de una metodología de entrenamiento de emociones que tiene un **buen funcionamiento** como modelo para ser utilizado en futuros proyectos.
- Conjunto de datos con una **gran dispersión** y la **falta de datos** en 1ª fase, pero ha logrado servir para el objetivo del proyecto, demostrando la potencia de las técnicas actuales de Inteligencia Artificial, logrando el objetivo de generar una metodología de inteligencia artificial en base a una red neuronal artificial.

Áreas de oportunidad

- Afectividad computacional, una área donde se trata leer pistas de lenguaje no verbal para interpretarlas como emociones.
- En campañas publicitarias, cuantificando la respuesta emocional de una persona al ver un anuncio

Dilemas éticos

- La IA debe realizarse por el bien de la humanidad y **beneficiar** al mayor número de personas pero con el objetivo de evitar la exclusión.
- El desarrollo de la IA debe evitar la violación de **privacidad** de los usuarios
- Necesidad de **control humano**: Que en todo momento sean los humanos los que decidan qué pueden hacer o no los sistemas robóticos o basados en IA.
- El implemento de tecnología de Inteligencia artificial no sea utilizada para controlar **armas de destrucción.**

Referencias

1. Pedro Ponce Cruz. (2011). Inteligencia artificial con aplicaciones en ingeniería. México: Alfaomega
2. B Geethanjali, K Adalarasu, A Hemapraba, S Pravin Kumar , R Rajasekeran. (10 de mayo del 2016). Emotion analysis using SAM (Self-Assessment Manikin) scale. Biomedical Research, 29, 7. 12 de septiembre del 2018, De Investigación Biomédica Base de datos.
3. Raiko, T., Valpola, H., y LeCun, Y. (2012). «Deep Learning Made Easier by Linear Transformations in Perceptron», en Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS-12), págs. 924–932, 2012.
4. Página web con información teórica sobre Deep Learning en español.
   <https://rubenlopezg.wordpress.com/2014/05/07/que-es-y-como-funciona-deep-learning/>
5. Página web con información acerca de un proyecto de reconocimiento de emociones.
   <https://riunet.upv.es/bitstream/handle/10251/77873/P%C3%89REZ%20-%20Emotions%20Recognition%20using%20Deep%20Learning.pdf?sequence=1>
6. Página web oficial de la librería Keras.<https://keras.io/>

Anexos

Python For Data Science *Cheat Sheet*

Keras

Learn Python for data science Interactively at www.DataCamp.com

Keras

Keras is a powerful and easy-to-use deep learning library for Theano and TensorFlow that provides a high-level neural networks API to develop and evaluate deep learning models.

A Basic Example

```
>>> import numpy as np
>>> from keras.models import Sequential
>>> from keras.layers import Dense
>>> data = np.random.random((1000,100))
>>> labels = np.random.randint(2,size=(1000,1))
>>> model = Sequential()
>>> model.add(Dense(32,
                    activation='relu',
                    input_dim=100))
>>> model.add(Dense(1, activation='sigmoid'))
>>> model.compile(optimizer='rmsprop',
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
>>> model.ft(data,labels,epochs=10,batch_size=32)
>>> predictions = model.predict(data)
```

Data

Also see NumPy, Pandas & Scikit-Learn

Your data needs to be stored as NumPy arrays or as a list of NumPy arrays. Ideally, you split the data in training and test sets, for which you can also resort to the `train_test_split` **module of** `sklearn.cross_validation`.

Keras Data Sets

```
>>> from keras.datasets import boston_housing,
                               mnist,
                               cifar10,
                               imdb
>>> (x_train,y_train),(x_test,y_test) = mnist.load_data()
>>> (x_train2,y_train2),(x_test2,y_test2) = boston_housing.load_data()
>>> (x_train3,y_train3),(x_test3,y_test3) = cifar10.load_data()
>>> (x_train4,y_train4),(x_test4,y_test4) = imdb.load_data(num_words=20000)
>>> num_classes = 10
```

Other

```
>>> from urllib.request import urlopen
>>> data = np.loadtxt(urlopen("http://archive.ics.uci.edu/
ml/machine-learning-databases/pima-indians-diabetes/
pima-indians-diabetes.data"),delimiter=",")
>>> X = data[:,0:8]
>>> y = data [:,8]
```

Model Architecture

Sequential Model

```
>>> from keras.models import Sequential
>>> model = Sequential()
>>> model2 = Sequential()
>>> model3 = Sequential()
```

Multilayer Perceptron (MLP)

Binary Classif cation

```
>>> from keras.layers import Dense
>>> model.add(Dense(12,
                    input_dim=8,
                    kernel_initializer='uniform',
                    activation='relu'))
>>> model.add(Dense(8,kernel_initializer='uniform',activation='relu'))
>>> model.add(Dense(1,kernel_initializer='uniform',activation='sigmoid'))
```

Multi-Class Classif cation

```
>>> from keras.layers import Dropout
>>> model.add(Dense(512,activation='relu',input_shape=(784,)))
>>> model.add(Dropout(0.2))
>>> model.add(Dense(512,activation='relu'))
>>> model.add(Dropout(0.2))
>>> model.add(Dense(10,activation='softmax'))
```

Regression

```
>>> model.add(Dense(64,activation='relu',input_dim=train_data.shape[1]))
>>> model.add(Dense(1))
```

Convolutional Neural Network (CNN)

```
>>> from keras.layers import Activation,Conv2D,MaxPooling2D,Flatten
>>> model2.add(Conv2D(32,(3,3),padding='same',input_shape=x_train.shape[1:]))
>>> model2.add(Activation('relu'))
>>> model2.add(Conv2D(32,(3,3)))
>>> model2.add(Activation('relu'))
>>> model2.add(MaxPooling2D(pool_size=(2,2)))
>>> model2.add(Dropout(0.25))

>>> model2.add(Conv2D(64,(3,3), padding='same'))
>>> model2.add(Activation('relu'))
>>> model2.add(Conv2D(64,(3, 3)))
>>> model2.add(Activation('relu'))
>>> model2.add(MaxPooling2D(pool_size=(2,2)))
>>> model2.add(Dropout(0.25))

>>> model2.add(Flatten())
>>> model2.add(Dense(512))
>>> model2.add(Activation('relu'))
>>> model2.add(Dropout(0.5))
>>> model2.add(Dense(num_classes))
>>> model2.add(Activation('softmax'))
```

Recurrent Neural Network (RNN)

```
>>> from keras.klayers import Embedding,LSTM
>>> model3.add(Embedding(20000,128))
>>> model3.add(LSTM(128,dropout=0.2,recurrent_dropout=0.2))
>>> model3.add(Dense(1,activation='sigmoid'))
```

Inspect Model

```
>>> model.output_shape          Model output shape
>>> model.summary()             Model summary representation
>>> model.get_confg()           Model conf guration
>>> model.get_weights()         List all weight tensors in the model
```

Compile Model

MLP: Binary Classif cation

```
>>> model.compile(optimizer='adam',
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
```

MLP: Multi-Class Classif cation

```
>>> model.compile(optimizer='rmsprop',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
```

MLP: Regression

```
>>> model.compile(optimizer='rmsprop',
                  loss='mse',
                  metrics=['mae'])
```

Recurrent Neural Network

```
>>> model3.compile(loss='binary_crossentropy',
                   optimizer='adam',
                   metrics=['accuracy'])
```

Model Training

```
>>> model3.ft(x_train4,
              y_train4,
              batch_size=32,
              epochs=15,
              verbose=1,
              validation_data=(x_test4,y_test4))
```

Evaluate Your Model's Performance

```
>>> score = model3.evaluate(x_test,
                            y_test,
                            batch_size=32)
```

Prediction

```
>>> model3.predict(x_test4, batch_size=32)
>>> model3.predict_classes(x_test4,batch_size=32)
```

Save/ Reload Models

```
>>> from keras.models import load_model
>>> model3.save('model_fle.h5')
>>> my_model = load_model('my_model.h5')
```

Model Fine-tuning

Optimization Parameters

```
>>> from keras.optimizers import RMSprop
>>> opt = RMSprop(lr=0.0001, decay=1e-6)
>>> model2.compile(loss='categorical_crossentropy',
                   optimizer=opt,
                   metrics=['accuracy'])
```

Early Stopping

```
>>> from keras.callbacks import EarlyStopping
>>> early_stopping_monitor = EarlyStopping(patience=2)
>>> model3.ft(x_train4,
              y_train4,
              batch_size=32,
              epochs=15,
              validation_data=(x_test4,y_test4),
              callbacks=[early_stopping_monitor])
```

Preprocessing

Also see NumPy & Scikit-Learn

Sequence Padding

```
>>> from keras.preprocessing import sequence
>>> x_train4 = sequence.pad_sequences(x_train4,maxlen=80)
>>> x_test4 = sequence.pad_sequences(x_test4,maxlen=80)
```

One-Hot Encoding

```
>>> from keras.utils import to_categorical
>>> Y_train = to_categorical(y_train, num_classes)
>>> Y_test = to_categorical(y_test, num_classes)
>>> Y_train3 = to_categorical(y_train3, num_classes)
>>> Y_test3 = to_categorical(y_test3, num_classes)
```

Train and Test Sets

```
>>> from sklearn.model_selection import train_test_split
>>> X_train5,X_test5,y_train5,y_test5 = train_test_split(X,
                                                         y,
                                                         test_size=0.33,
                                                         random_state=42)
```

Standardization/Normalization

```
>>> from sklearn.preprocessing import StandardScaler
>>> scaler = StandardScaler().ft(x_train2)
>>> standardized_X = scaler.transform(x_train2)
>>> standardized_X_test = scaler.transform(x_test2)
```

DataCamp
Learn Python for Data Science Interactively

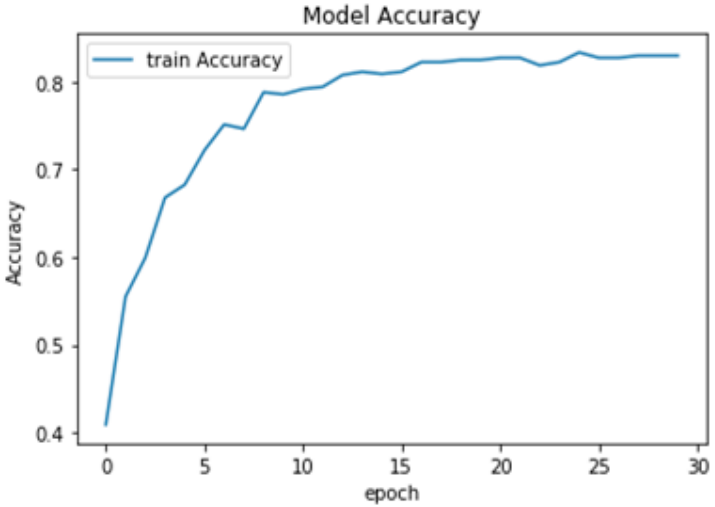
Anexos

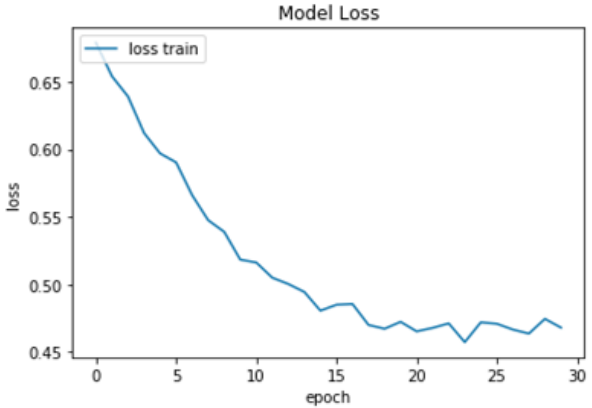
Entrenamiento de las redes neuronales pero con la ayuda de la función ‘softmax’

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 64)	256
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 36)	2340
dropout_2 (Dropout)	(None, 36)	0
dense_3 (Dense)	(None, 6)	222

Total params: 2,818
Trainable params: 2,818
Non-trainable params: 0

Anexos


Model Accuracy
train Accuracy
Accuracy
epoch
0.4
0.5
0.6
0.7
0.8
0
5
10
15
20
25
30


Model Loss
loss train
loss
epoch
0.45
0.50
0.55
0.60
0.65
0
5
10
15
20
25
30

Anexo

```
Avatar = pd.read_csv('Avatar2.0.csv')
Avatar

y = targets = labels = Avatar["[Agrado-Desagrado]"].values
y = y.reshape(-1,1)

cols = ["Animacion", "Edad"]
features = Avatar[list(cols)].values

x1 = np.array([x[0:230] for x in features])
y1 = np.array([x[0:230] for x in y])

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(features, y, test_size=0.2)

x_train = np.array(x1)
x_train = x_train/x_train.max()
y_train = np.array(y1)
y_train = y_train/y_train.max()
x_test = np.array(x1)
x_test = x_test/x_test.max()
y_test = np.array(y1)
y_test = y_test/y_test.max()
```

```
model = Sequential()
model.add(Dense(25, input_dim=2, activation='relu',
kernel_initializer="uniform"))
model.add(Dense(5, kernel_initializer="uniform", activation='relu'))
model.add(Dense(1, activation='relu', kernel_initializer="uniform"))
model.compile(loss='mse', optimizer="adam", metrics=['mae'])
model.summary()

history_object= model.fit(x_train, y_train, epochs=40, shuffle=True,
verbose=1, validation_split=0.2)


predicciones = model.predict(x_test)
print(predicciones)
predic= np.array(predicciones)*6
predic

result = np.rint(predic)
result
result = result.tolist()

pre = pd.Series(result)
Avatar['result'] = pre
Avatar['result'] = Avatar['result'].str.get(0)
Avatar
```

Anexo (Resultados Fase 1)

	Genero	Edad	Animacion	[Agrado-Desagrado]	result
0	1	22	1	3	3.0
1	1	22	1	3	3.0
2	1	21	1	3	3.0
3	1	22	1	2	3.0
4	1	23	1	2	4.0
5	1	21	1	3	3.0
6	1	23	1	5	4.0
7	1	23	1	2	4.0
8	0	16	1	3	3.0
9	0	22	1	4	3.0
10	1	20	1	4	3.0
11	1	22	1	5	3.0
12	1	18	1	6	3.0
13	0	21	1	4	3.0
14	0	22	1	2	3.0
15	1	18	1	4	3.0

Anexo resultados

	Participants	Video	Edad	Genero	Consumo Alcohol	Consumo caf	Consumo Te	Horas de sueno	Nivel de Alerta	Vision	Excitacion	result
0	1	1	31	1	0.0	0.0	0.0	8	2	0	3.92	4.257283
1	2	1	24	0	0.0	0.5	1.0	8	2	1	5.01	2.862322
2	3	1	19	0	0.5	0.0	0.5	7	3	1	4.37	4.922933
3	4	1	24	0	0.0	1.0	0.5	8	2	1	8.05	2.114147
4	5	1	24	1	0.0	0.0	0.0	8	2	1	6.96	5.718594
5	6	1	23	1	0.5	0.5	0.0	8	2	0	3.49	4.158087
6	7	1	31	1	1.0	0.0	0.0	8	2	1	2.05	4.805614
7	8	1	22	0	0.5	0.0	1.0	8	2	0	5.79	7.586703
8	9	1	25	0	0.5	0.5	1.0	8	2	1	4.04	7.496221
9	10	1	31	0	1.0	1.0	0.5	7	2	0	7.14	3.454487
10	11	1	27	0	0.0	1.0	0.5	7	3	0	8.70	9.203244
11	12	1	37	1	0.0	1.0	0.0	8	3	1	8.21	6.704191
12	13	1	24	0	1.0	1.0	1.0	8	2	0	9.00	2.928674
13	14	1	27	0	0.5	0.0	1.0	7	2	1	7.06	2.711467
14	15	1	22	0	1.0	0.0	0.5	6	2	0	6.08	2.783105
15	16	1	28	1	0.0	0.5	1.0	7	2	1	6.28	5.956354
16	17	1	25	1	0.5	1.0	1.0	7	2	0	4.99	5.056306
17	18	1	29	1	0.5	0.0	1.0	8	2	0	5.60	6.052489
18	19	1	27	1	0.5	0.0	1.0	7	2	1	7.58	3.436383

Anexo resultados

	Participants	Video	Edad	Genero	Consumo Alcohol	Consumo caf	Consumo Te	Horas de sueno	Nivel de Alerta	Vision	Dominancia	result
0	1	1	31	1	0.0	0.0	0.0	8	2	0	7.19	6.763408
1	2	1	24	0	0.0	0.5	1.0	8	2	1	3.01	5.110972
2	3	1	19	0	0.5	0.0	0.5	7	3	1	4.03	5.162420
3	4	1	24	0	0.0	1.0	0.5	8	2	1	6.08	2.172548
4	5	1	24	1	0.0	0.0	0.0	8	2	1	8.00	4.977219
5	6	1	23	1	0.5	0.5	0.0	8	2	0	4.05	4.212013
6	7	1	31	1	1.0	0.0	0.0	8	2	1	3.00	3.480520
7	8	1	22	0	0.5	0.0	1.0	8	2	0	1.19	5.183892
8	9	1	25	0	0.5	0.5	1.0	8	2	1	3.01	3.948178
9	10	1	31	0	1.0	1.0	0.5	7	2	0	7.09	4.468852
10	11	1	27	0	0.0	1.0	0.5	7	3	0	1.14	4.819095
11	12	1	37	1	0.0	1.0	0.0	8	3	1	2.97	5.873275
12	13	1	24	0	1.0	1.0	1.0	8	2	0	1.99	2.462061
13	14	1	27	0	0.5	0.0	1.0	7	2	1	7.99	5.096415
14	15	1	22	0	1.0	0.0	0.5	6	2	0	4.01	5.975973
15	16	1	28	1	0.0	0.5	1.0	7	2	1	2.83	4.686368
16	17	1	25	1	0.5	1.0	1.0	7	2	0	5.00	6.568071
17	18	1	29	1	0.5	0.0	1.0	8	2	0	5.73	4.114949
18	19	1	27	1	0.5	0.0	1.0	7	2	1	5.81	2.626477
19	20	1	25	1	0.0	0.0	0.0	8	1	1	4.68	2.364194