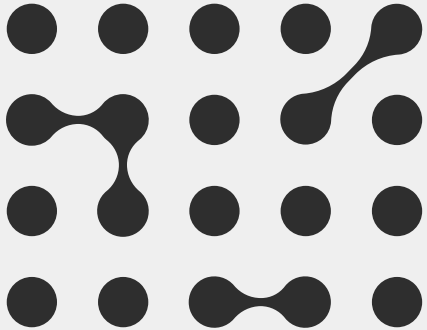
Proyecto Final

Statistical Learning II

Edgar Sabán 19012631



Feed-forward neural network

Dectectar si una persona está corriendo o caminando con base a datos del acelerómetro y giroscopio de un iPhone

Información general

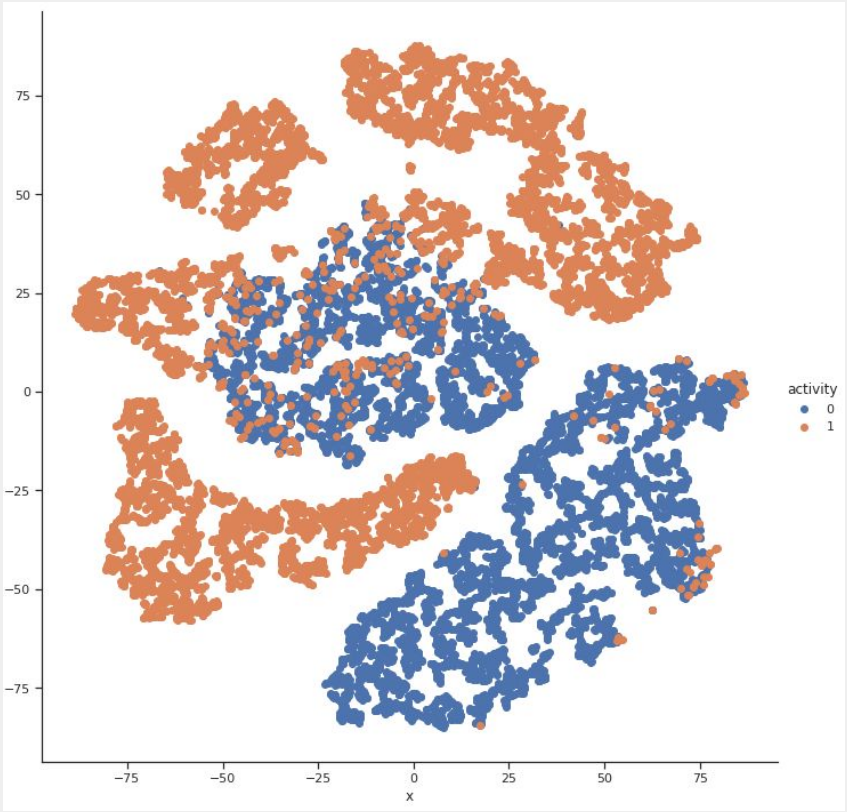
Dataset

- 88,588 observaciones
- 7 variables: acceleration x, y, z; gyro x, y, z; wrist
- 1 output: 0 walking 1 running

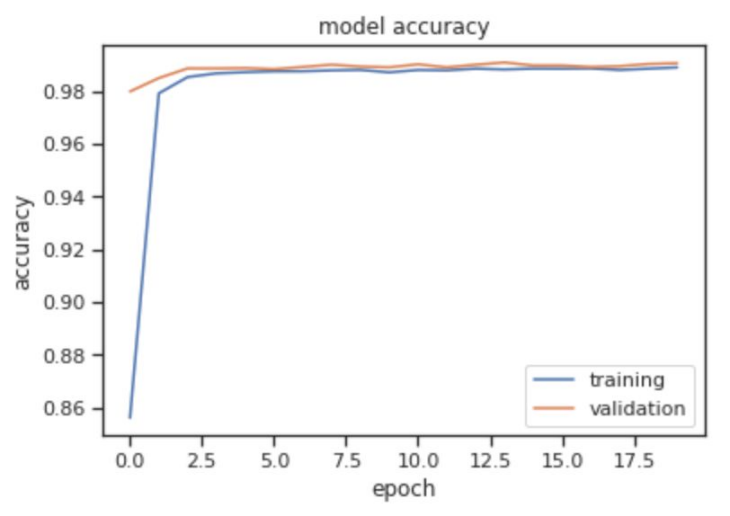
Feed-foward neural network

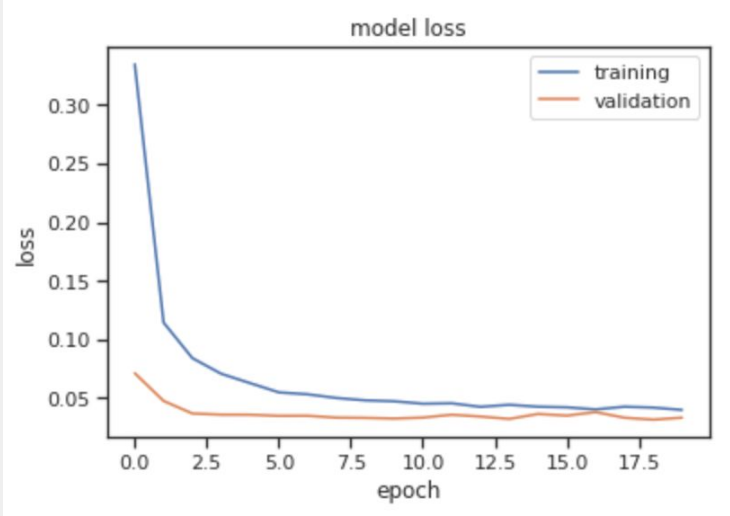
- Reducción de dimensionalidad (tsne)
- 1 capa de entrada
- 4 capas ocultas (relu)
- 1 capa de salida (sigmoid)
- loss=binary_crossentropy
- optimizer=adam
- Metric = accuracy

	date	time	username	wrist	activity	acceleration_x	acceleration_y	acceleration_z	gyro_x	gyro_y	gyro_z
0	2017-6-30	13:51:15:847724020	viktor	0	0	0.2650	-0.7814	-0.0076	-0.0590	0.0325	-2.9296
1	2017-6-30	13:51:16:246945023	viktor	0	0	0.6722	-1.1233	-0.2344	-0.1757	0.0208	0.1269
2	2017-6-30	13:51:16:446233987	viktor	0	0	0.4399	-1.4817	0.0722	-0.9105	0.1063	-2.4367
3	2017-6-30	13:51:16:646117985	viktor	0	0	0.3031	-0.8125	0.0888	0.1199	-0.4099	-2.9336
4	2017-6-30	13:51:16:846738994	viktor	0	0	0.4814	-0.9312	0.0359	0.0527	0.4379	2.4922


75
50
25
0
−25
−50
−75
−75
−50
−25
0
25
50
75
x
activity
0
1

```python
# Definiendo arquitectura de la red
def crear_red():
    model = keras.Sequential(
    [
        layers.Dense(14, input_shape = (None,7), activation="relu"),
        layers.Dense(20, activation="relu"),
        layers.Dense(10, activation="relu"),
        layers.Dense(5, activation="relu"),
        layers.Dropout(0.25),
        layers.Dense(1,  activation="sigmoid"),
    ]
    )
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model
```






```
Test loss: 0.0266
Test accuracy: 0.993
```

Predicciones

```
Y = model.predict(test_X)
Y = np.round(Y[:,0]).astype('int')
```

```
WARNING:tensorflow:Model was constructed w
(None, None, 7), dtype=tf.float32, name='c
_10_input'"), but it was called on an inpu
```

```
correct = Y == test_Y
correct.sum()
```

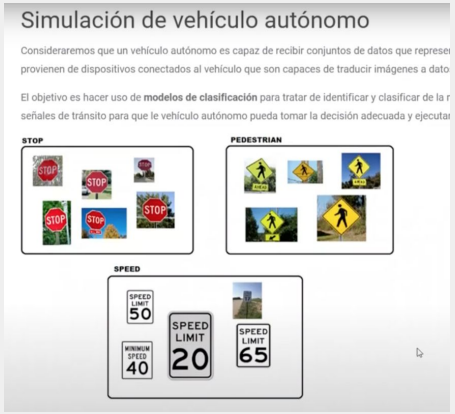
```
17593
```

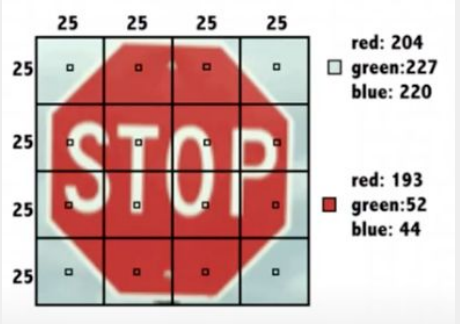
```
incorrect = Y != test_Y
incorrect.sum()
```

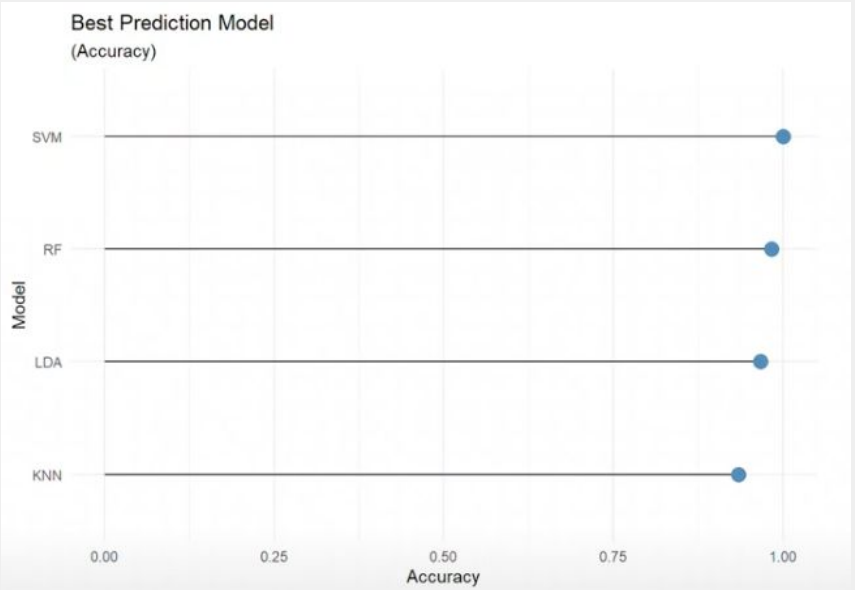
```
125
```

Red Convolucional

Clasificación de imágenes que representan señales de tránsito


Simulación de vehículo autónomo
Consideraremos que un vehículo autónomo es capaz de recibir conjuntos de datos que represe
provienen de dispositivos conectados al vehículo que son capaces de traducir imágenes a dato
El objetivo es hacer uso de modelos de clasificación para tratar de identificar y clasificar de la
señales de tránsito para que le vehículo autónomo pueda tomar la decisión adecuada y ejecutar
STOP
PEDESTRIAN
SPEED
SPEED LIMIT 50
MINIMUM SPEED 40
SPEED LIMIT 20
SPEED LIMIT 65


25
25
25
25
25
25
25
STOP
red: 204
green:227
blue: 220
red: 193
green:52
blue: 44


Best Prediction Model
(Accuracy)
Model
SVM
RF
LDA
KNN
0.00
0.25
0.50
0.75
1.00
Accuracy

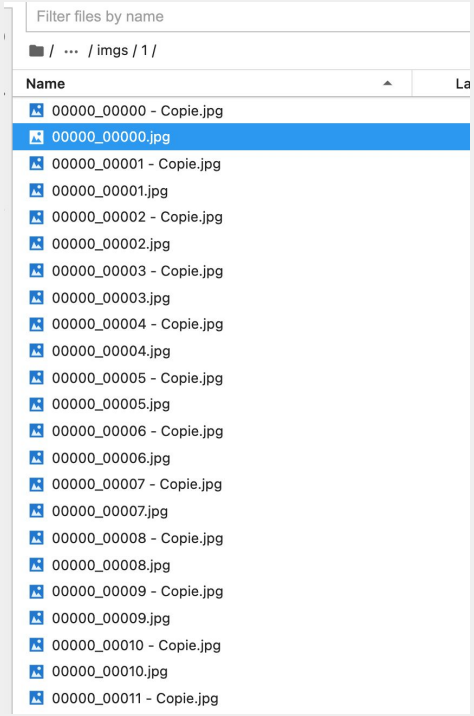
Información general

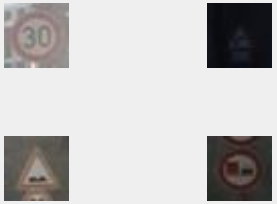
Dataset

- 58, 861 observaciones
- 32 x 32 pixels
- rgb
- (58871, 32, 32, 3)
- 30 tipos de señales

CNN

- Creación de labels - one hot encoding
- lr = 0.001 epochs=10 batch_size=128
- Capas convolucionales de 32 y 64 con kernel (3,3) (relu)
- MaxiPooling(2,2)
- Flatten()
- Capa oculta de 256 (relu)
- Capa salida 30 (softmax)





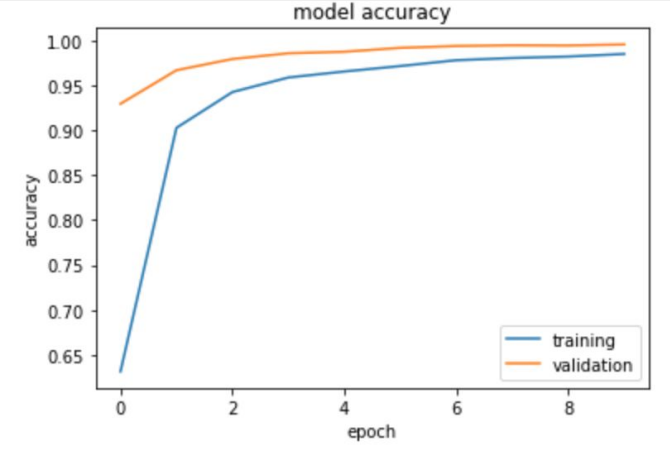

```
dataset = np.array(dataset)
dataset.shape
```

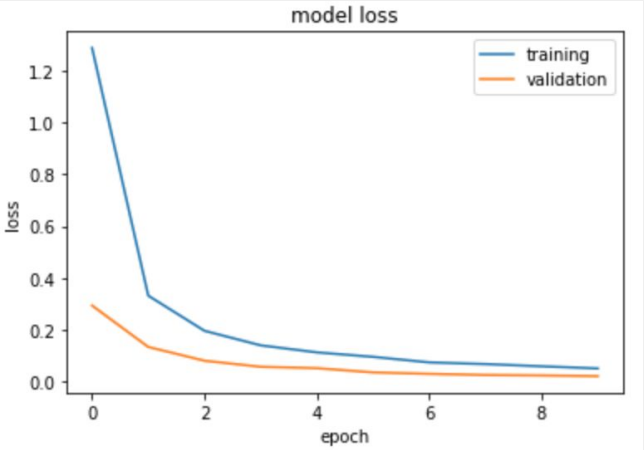
```
(58861, 32, 32, 3)
```

```
labels = np.array(labels)
labels.shape
```

```
(58861,)
```

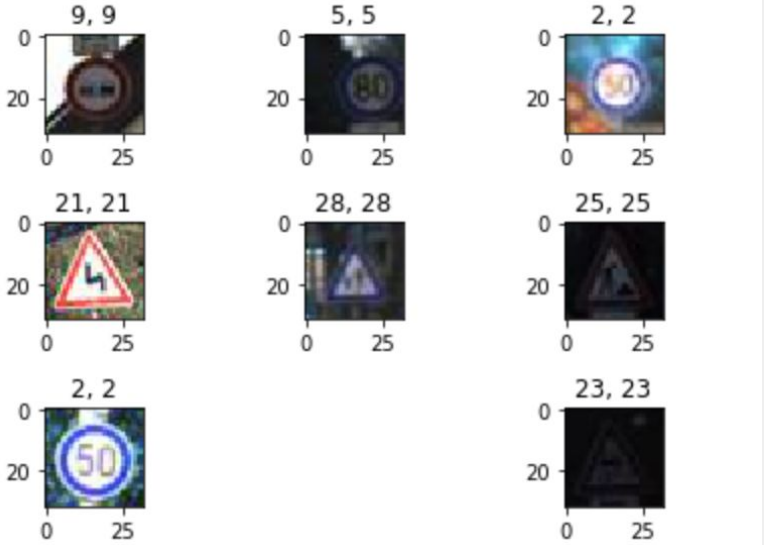
```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),activation='relu',padding='same',input_shape=(32,32,3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
model.summary()
```



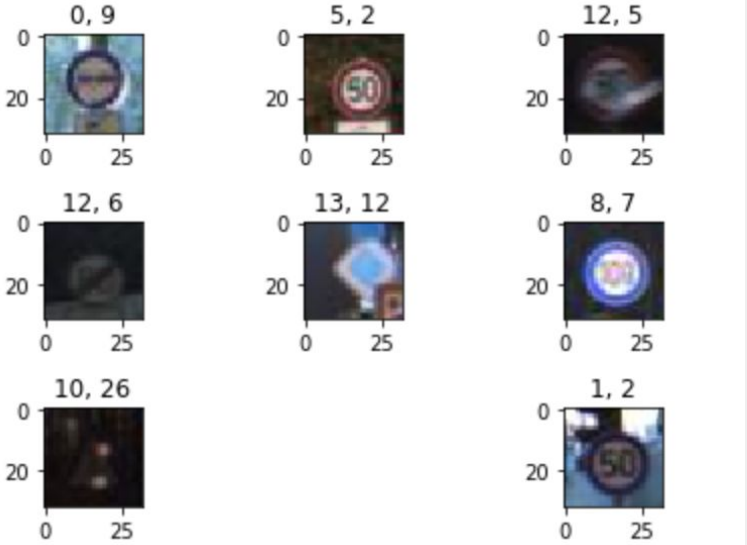



```
Test loss: 0.0193
Test accuracy: 0.995
```

11719 imágenes clasificadas correctamente



54 imágenes clasificadas incorrectamente



Red Recurrente

Predicción de las acciones de APPLE Inc.

Información general

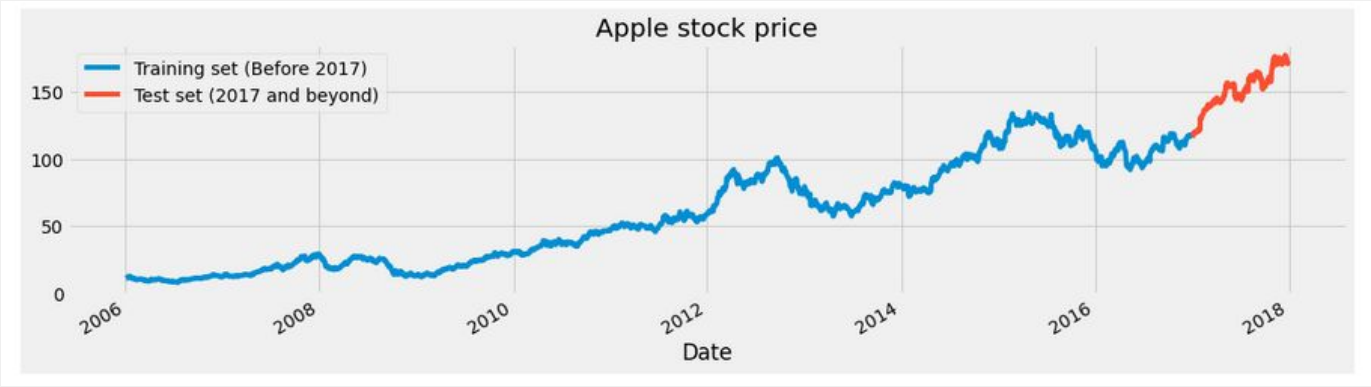
Dataset

- 2,518 observaciones
- 6 variables

RNN

- LSTM con 128 unidades
- LSTM con 64 unidades
- Capa oculta de 25 neuronas
- Capa salida 1 (prediccion)

Date	Open	High	Low	Close	Volume	Name
2006-01-03	10.34	10.68	10.32	10.68	201853036	AAPL
2006-01-04	10.73	10.85	10.64	10.71	155225609	AAPL
2006-01-05	10.69	10.70	10.54	10.63	112396081	AAPL
2006-01-06	10.75	10.96	10.65	10.90	176139334	AAPL
2006-01-09	10.96	11.03	10.82	10.86	168861224	AAPL



```
# The LSTM architecture
regressor = Sequential()
# First LSTM layer with Dropout regularisation
regressor.add(LSTM(units=50, return_sequences=True, input_shape=(60,1)))
regressor.add(Dropout(0.2))
# Second LSTM layer
regressor.add(LSTM(units=50, return_sequences=True))
regressor.add(Dropout(0.2))
# Third LSTM layer
regressor.add(LSTM(units=50, return_sequences=True))
regressor.add(Dropout(0.2))
# Fourth LSTM layer
regressor.add(LSTM(units=50))
regressor.add(Dropout(0.2))
# The output layer
regressor.add(Dense(units=1))

# Compiling the RNN
regressor.compile(optimizer='rmsprop',loss='mean_squared_error')
# Fitting to the training set
regressor.fit(X_train,y_train,epochs=50,batch_size=32)
```


Close Price History
Close Price USD ($)
140
120
100
80
60
40
20
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
Date