

# CLASIFICACIÓN DE GÉNEROS MUSICALES

- Andrés Felipe Uribe García
- Juan Felipe Ortiz Trillos
- Orlando Alberto Moncada Rodríguez

# CONTEXTO E IDEA DEL PROYECTO

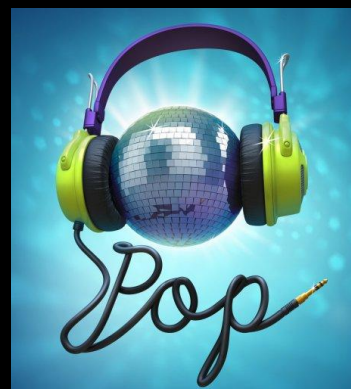
Expertos han estado tratando por mucho tiempo de:

- Entender el sonido y encontrar las diferencias de un sonido y otro.
- Como poder visualizar el sonido.
- Que hace que un tono sea diferente de otro.

Por esto, hemos decidido tratar de dar una solución a estas premisas.



# CONJUNTO DE DATOS

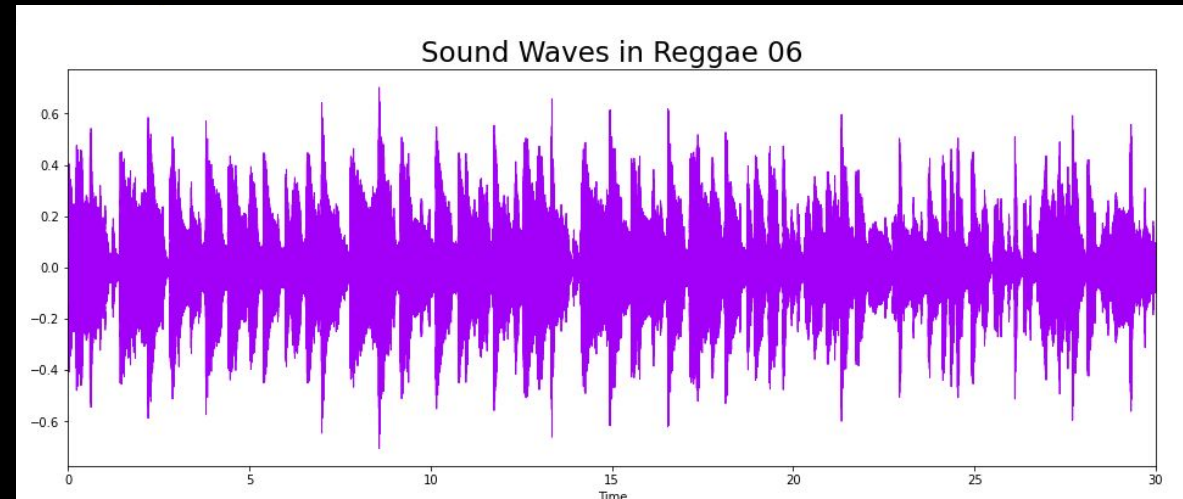




# ELEMENTOS DATASET

- Archivos de audio de 3 y 30 segundos de cada género musical.
- Imágenes de las ondas de sonido de cada género musical.
- 2 CSV's con información acerca de los archivos de audio.

	filename	length	chroma_stft_mean	chroma_stft_var	rms_mean
0	blues.00000.wav	661794	0.350088	0.088757	0.130228
1	blues.00001.wav	661794	0.340914	0.094980	0.095948
2	blues.00002.wav	661794	0.363637	0.085275	0.175570
3	blues.00003.wav	661794	0.404785	0.093999	0.141093
4	blues.00004.wav	661794	0.308526	0.087841	0.091529



## ▶ converting extracted\_features to Pandas dataframe

[Mostrar código](#)



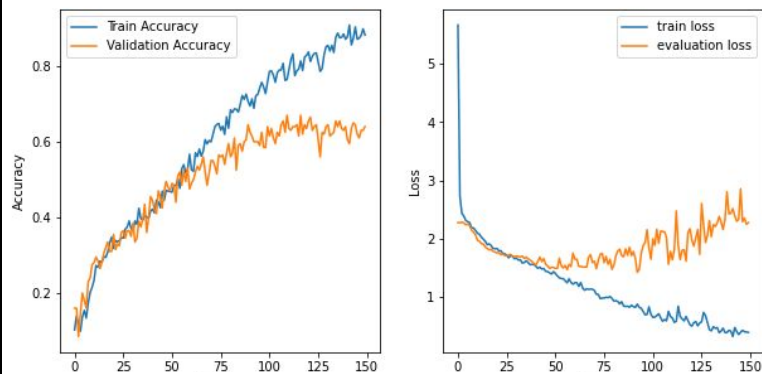
	feature	class
0	[-113.57065, 121.57179, -19.168142, 42.36642, ...	blues
1	[-207.5017, 123.991264, 8.955127, 35.87765, 2....	blues
2	[-90.722595, 140.4463, -29.09389, 31.684334, -...	blues
3	[-199.5442, 150.09091, 5.6626782, 26.85908, 1....	blues
4	[-160.3377, 126.219635, -35.58781, 22.148071, ...	blues

## [8] Cantidad de canciones en cada clase

[Mostrar código](#)

```
country      100
reggae       100
rock          100
disco         100
hiphop        100
metal         100
classical     100
pop           100
blues         100
jazz          99
Name: class, dtype: int64
```

SE CONVIRTIÓ  
EL AUDIO A UN  
DATAFRAME



Model: "sequential\_28"

Layer (type)	Output Shape	Param #
dense_127 (Dense)	(None, 1024)	41984
dropout_99 (Dropout)	(None, 1024)	0
dense_128 (Dense)	(None, 512)	524800
dropout_100 (Dropout)	(None, 512)	0
dense_129 (Dense)	(None, 256)	131328
dropout_101 (Dropout)	(None, 256)	0
dense_130 (Dense)	(None, 128)	32896
dropout_102 (Dropout)	(None, 128)	0
dense_131 (Dense)	(None, 64)	8256
dropout_103 (Dropout)	(None, 64)	0
dense_132 (Dense)	(None, 32)	2080
dropout_104 (Dropout)	(None, 32)	0
dense_133 (Dense)	(None, 10)	330

Total params: 741,674  
 Trainable params: 741,674  
 Non-trainable params: 0

# CREACIÓN RED NEURONAL

# SE REALIZA UN TESTEO Y SE REVISA EL ACCURACY

Evaluate

[Mostrar código](#)

```
[2.2418131828308105, 0.6549999713897705]
```



Prueba

prueba:

[Mostrar código](#)

```
(1, 40)
```

```
[1]
```

```
['classical']
```

```
/content/drive/MyDrive/Proyecto/Data/genres_original/classical/classical.00020.wav
```



# TRANSFER LEARNING

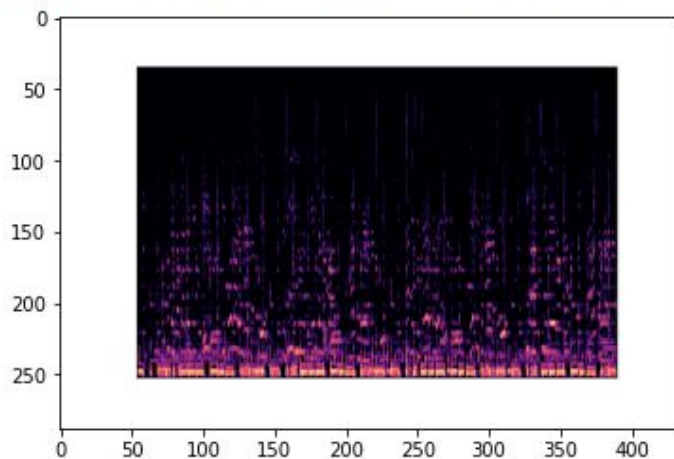
- En este apartado se decidió, por el momento utilizar la red **Xception y MobileNet** debido a su eficacia y procederemos a mostrar el proceso realizado hasta llegar al resultado del entrenamiento



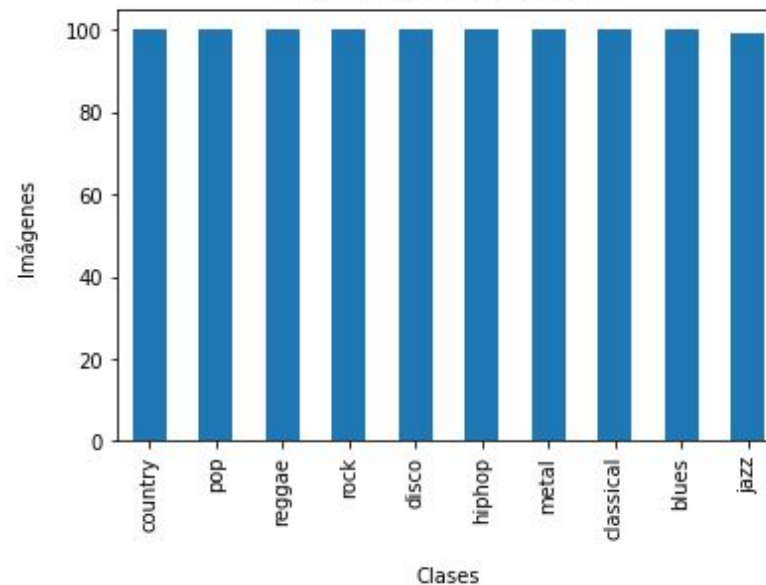
## Visualización de una imagen aleatoria

[Mostrar código](#)

```
reggae/reggae00020.png  
<matplotlib.image.AxesImage at 0x7f593013e8d0>
```



Cantidad imágenes por clase



# IMÁGENES Y DATAFRAME

# SE GENERA UN TRAIN Y TEST GENERATOR Y SE AGREGAN CAPAS A LA RED

Se divide el DF para entrenamiento y testeo

[Mostrar código](#)

```
train = 799 test = 200
```

train generator

[Mostrar código](#)

```
Found 799 validated image filenames belonging to 10 classes.
```

test generator

[Mostrar código](#)

```
Found 200 validated image filenames belonging to 10 classes.
```

## Summary

[Mostrar código](#)

Model: "sequential\_1"

Layer (type)	Output Shape	Param #
exception (Functional)	(None, 11, 7, 2048)	20861480
flatten (Flatten)	(None, 157696)	0
dense_8 (Dense)	(None, 256)	40370432
dense_9 (Dense)	(None, 128)	32896
dense_10 (Dense)	(None, 64)	8256
dense_7 (Dense)	(None, 10)	650

Total params: 61,273,714

Trainable params: 40,412,234

Non-trainable params: 20,861,480

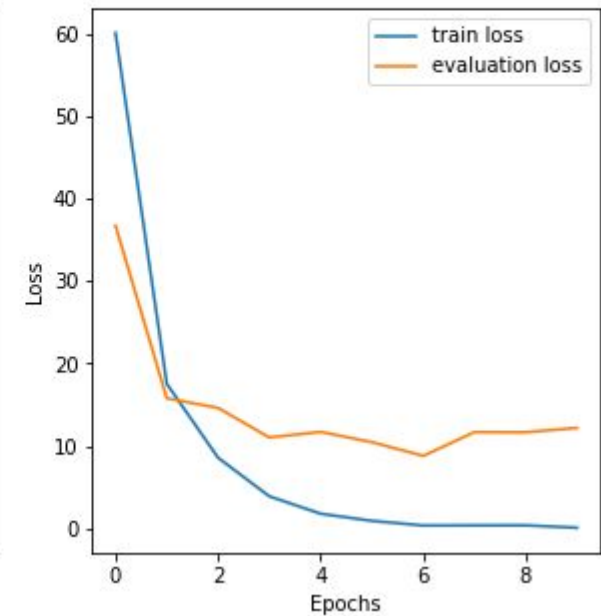
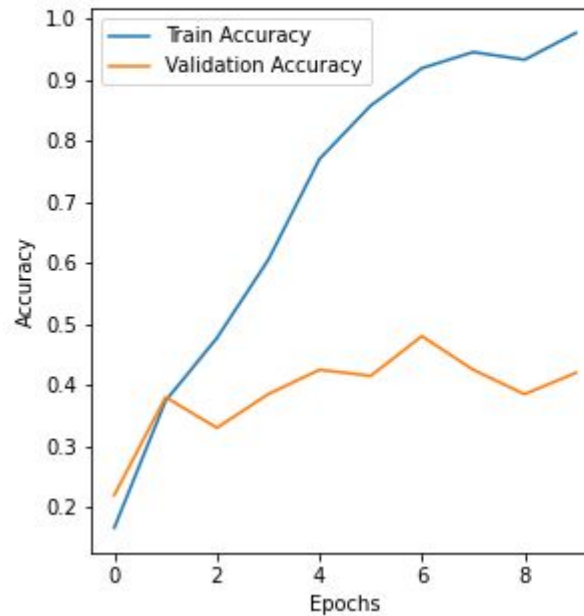
number of layers to train: 8

# GRÁFICA DEL MODELO XCEPTION

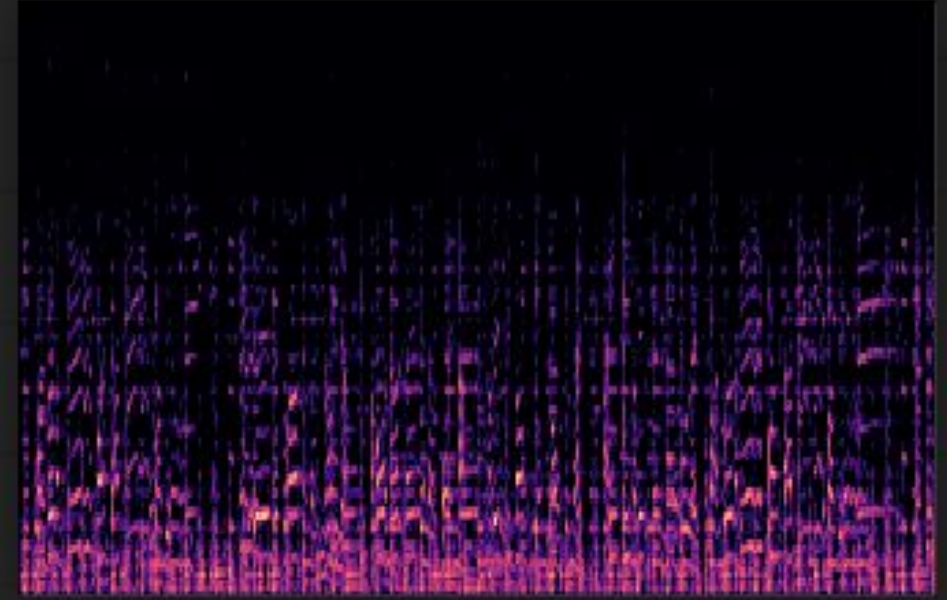
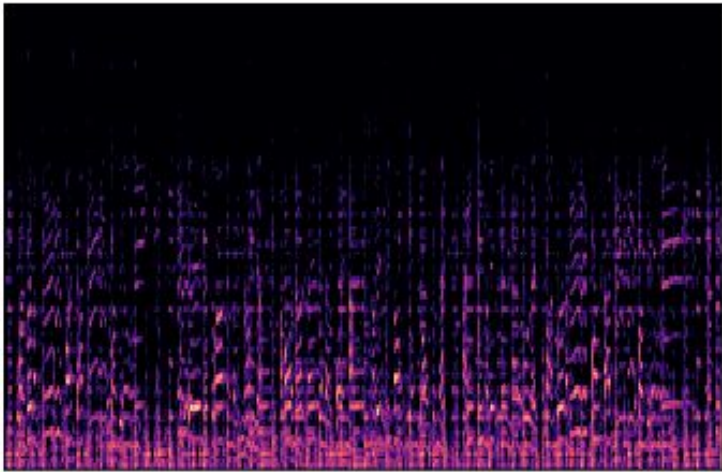
- Acá Podemos ver que ya llegó a un punto en que está sobre aprendiendo pues la diferencia entre el train accuracy y el validation accuracy cada vez aumenta más

Gráfica Xception

[Mostrar código](#)



Como se puede observar se decidió realizar un corte a las imágenes para que se puedan procesar de una mejor manera.



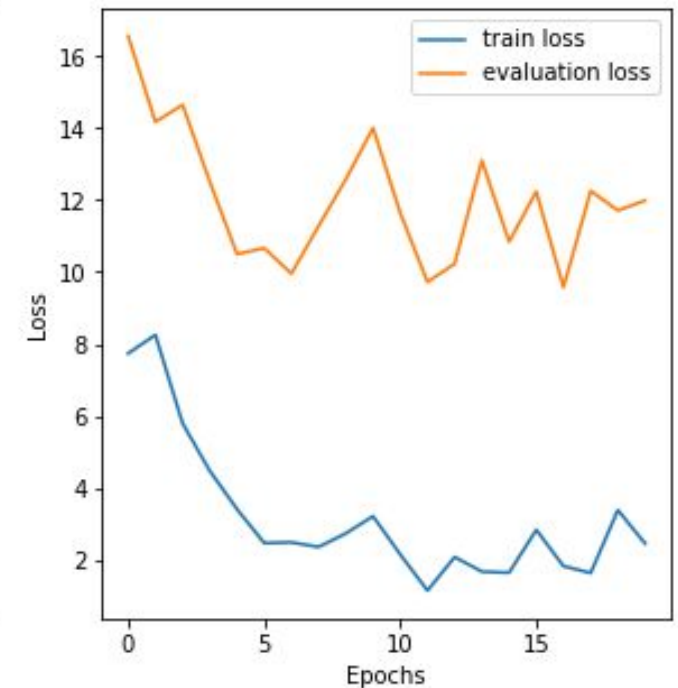
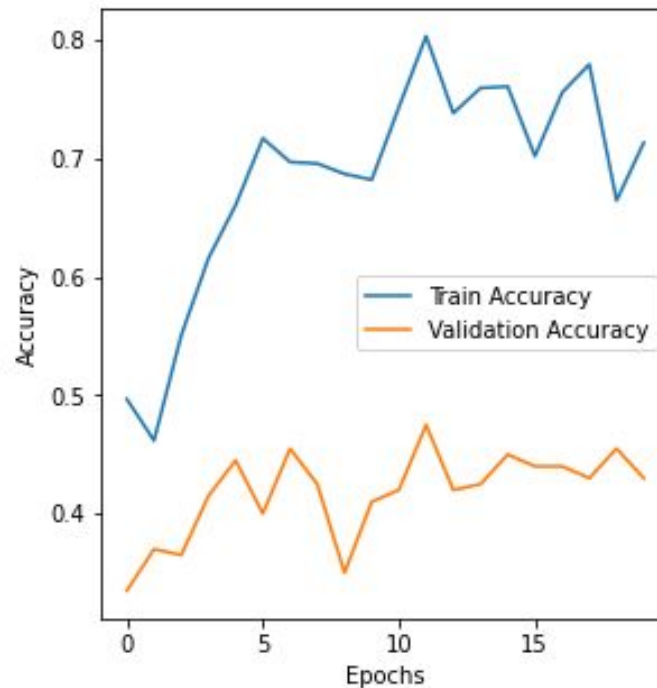


# GRÁFICA DEL MODELO XCEPTION

- Acá Podemos ver que ya llegó a un punto en que está sobre aprendiendo pues la diferencia entre el train accuracy y el validation accuracy cada vez aumenta más

Gráfica Xception

[Mostrar código](#)

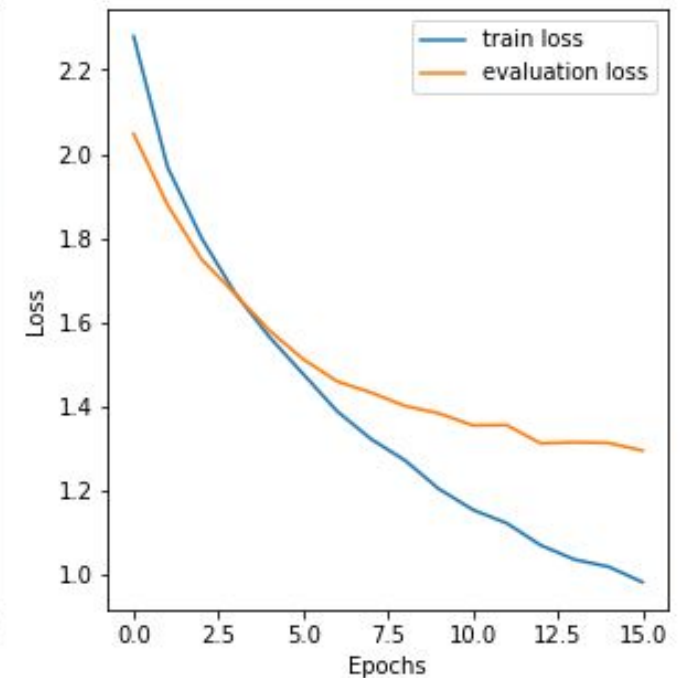
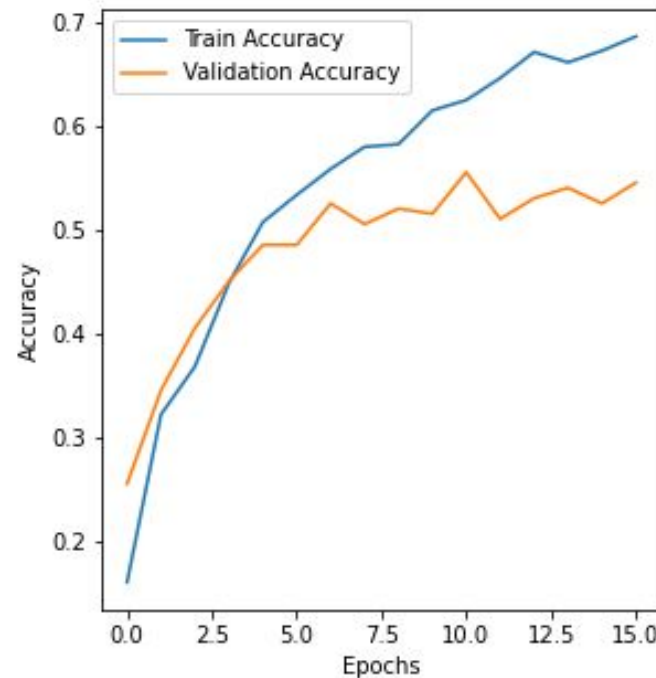


# GRÁFICA DEL MODELO MOBILENETV2

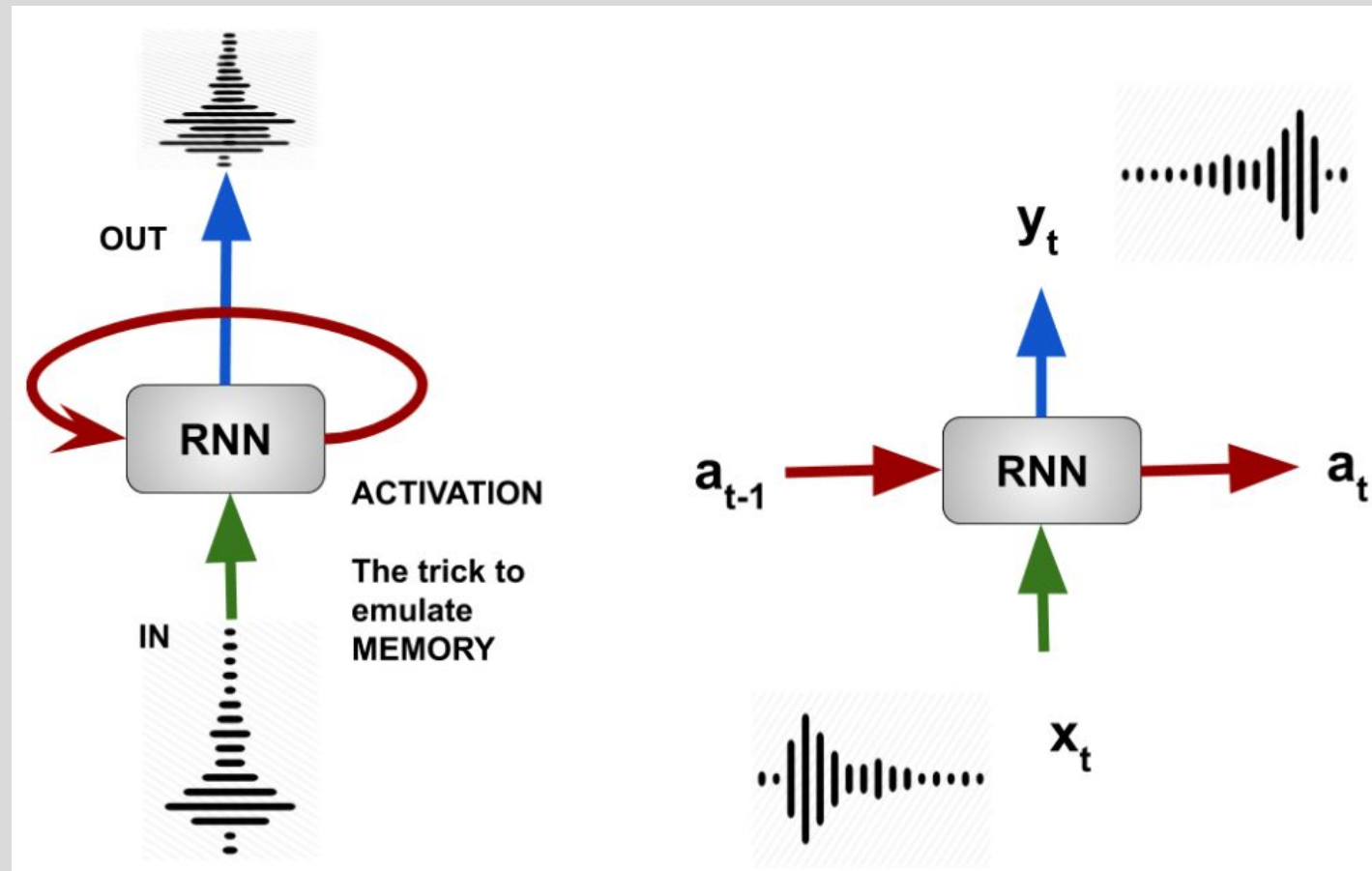
- Acá Podemos ver que ya llegó a un punto en que está sobre aprendiendo pues la diferencia entre el train accuracy y el validation accuracy cada vez aumenta más

Gráfica Mobile

[Mostrar código](#)



Y AHORA...  
UNA RNN!



## Modelo:

- 2 capas LSTM
- 1 capa densa
- 1 dropout
- 1 capa densa de activación softmax, que en este caso sería la clasificación que se requiere.

Model: "sequential\_2"

Layer (type)	Output Shape	Param #
=====		
lstm_4 (LSTM)	(None, 130, 64)	19968
lstm_5 (LSTM)	(None, 64)	33024
dense_4 (Dense)	(None, 64)	4160
dropout_2 (Dropout)	(None, 64)	0
dense_5 (Dense)	(None, 10)	650
=====		

Total params: 57,802

Trainable params: 57,802

Non-trainable params: 0



# RESULTADOS

Como podemos observar se obtuvo una buena precisión, teniendo en cuenta que hasta el momento el mejor accuracy de los diferentes métodos era de 0,65.

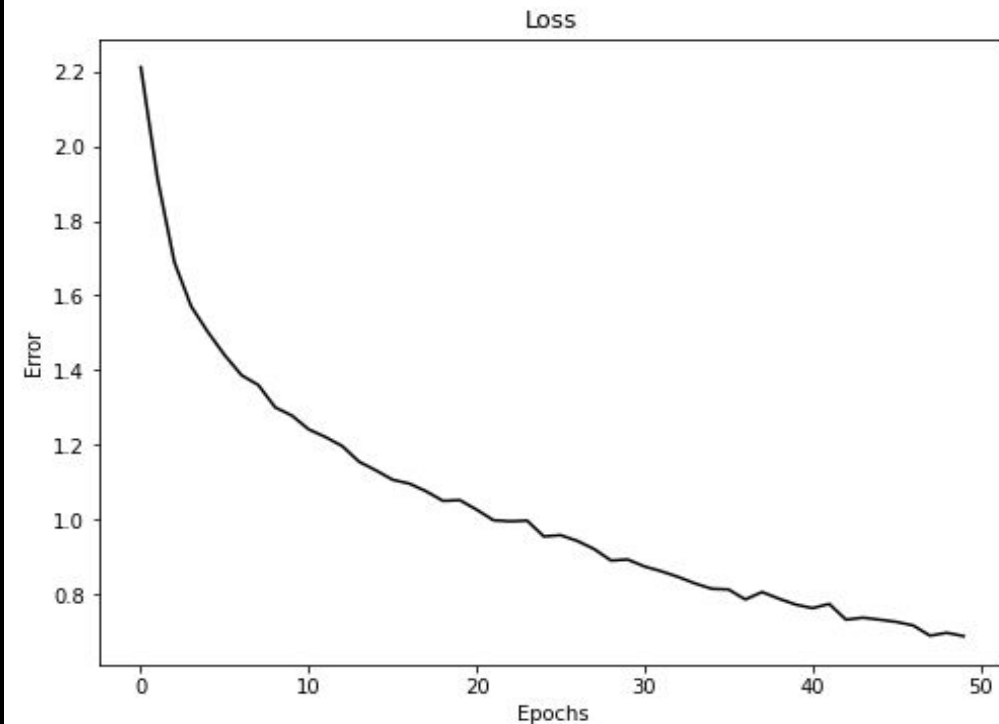
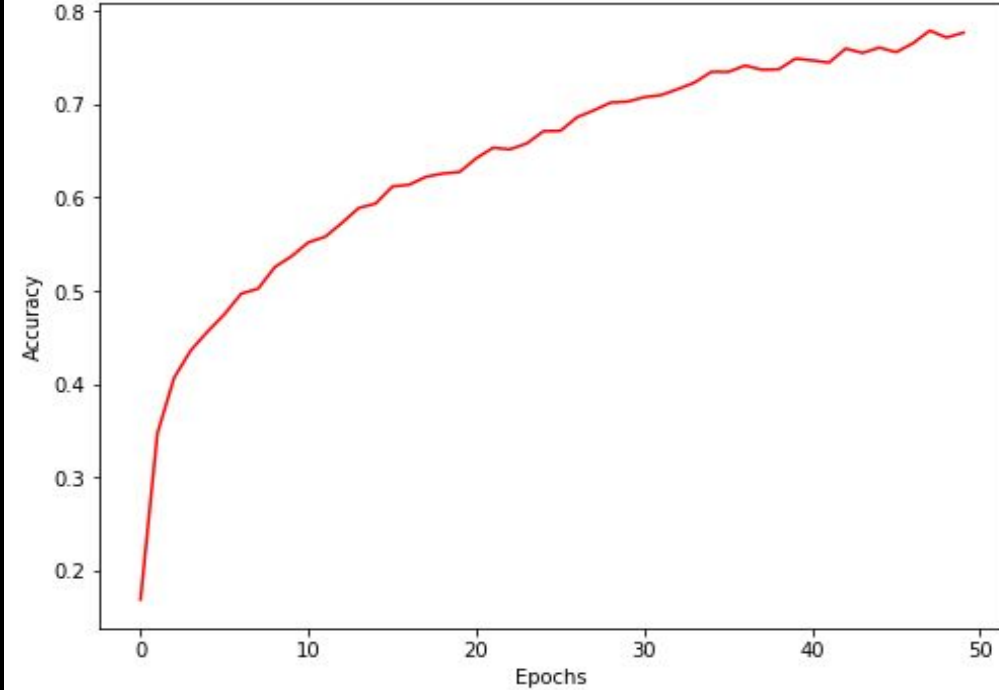
Lo cual nos indica que hasta el momento la RNN tiene mejor precisión de acuerdo a los métodos presentados en este proyecto

Evaluate

[Mostrar código](#)

71/71 - 2s - loss: 1.0359 - accuracy: 0.6634

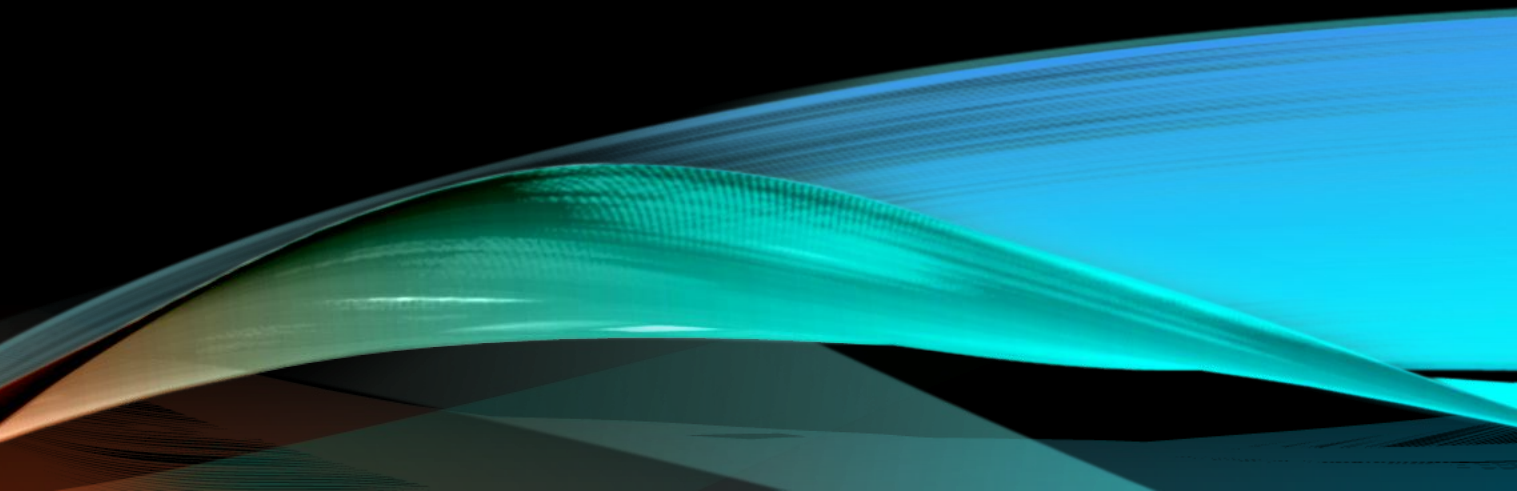
Test accuracy: 0.6634059548377991



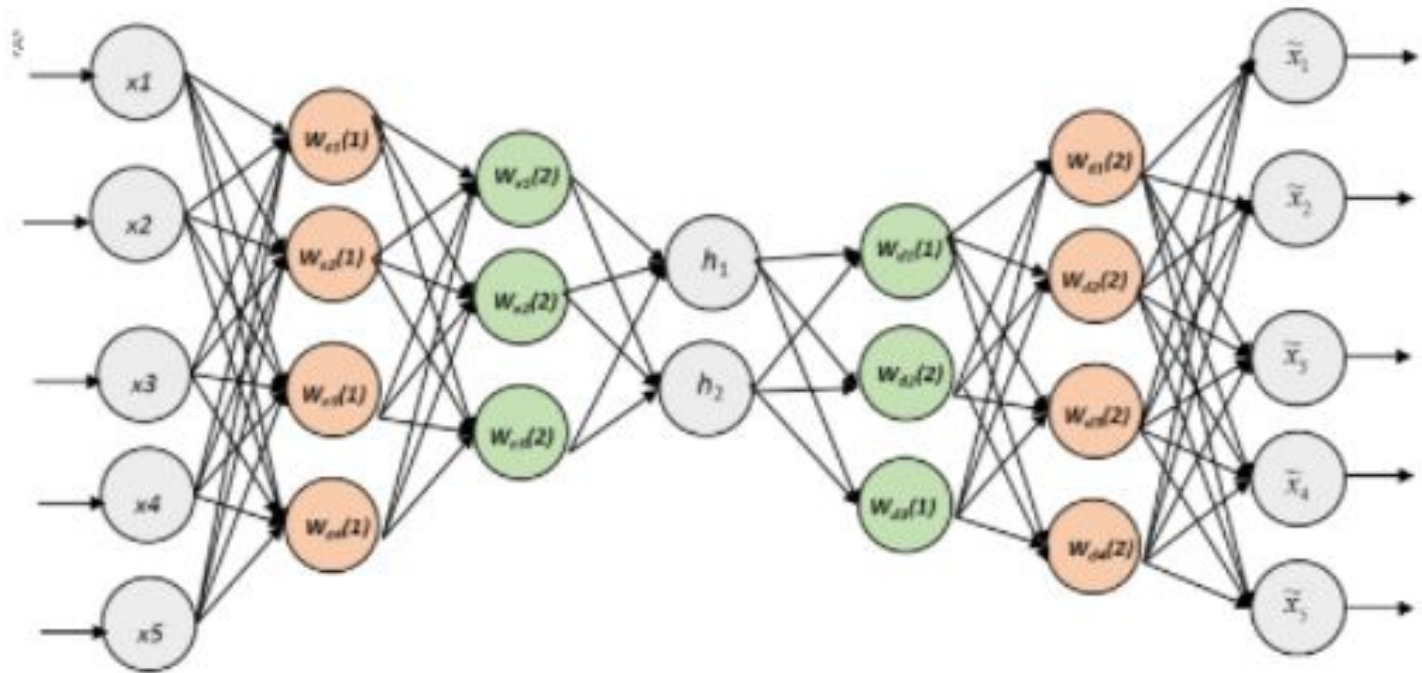
The left side of the slide features a vertical panel with a complex, abstract pattern of overlapping teal and dark green triangles and polygons, creating a sense of depth and movement.

# RETO: CREACIÓN DE MÚSICA

Se decidió utilizar los datos para realizar un autoencoder y, así, poder crear música a base de las que ya tenemos en el dataset

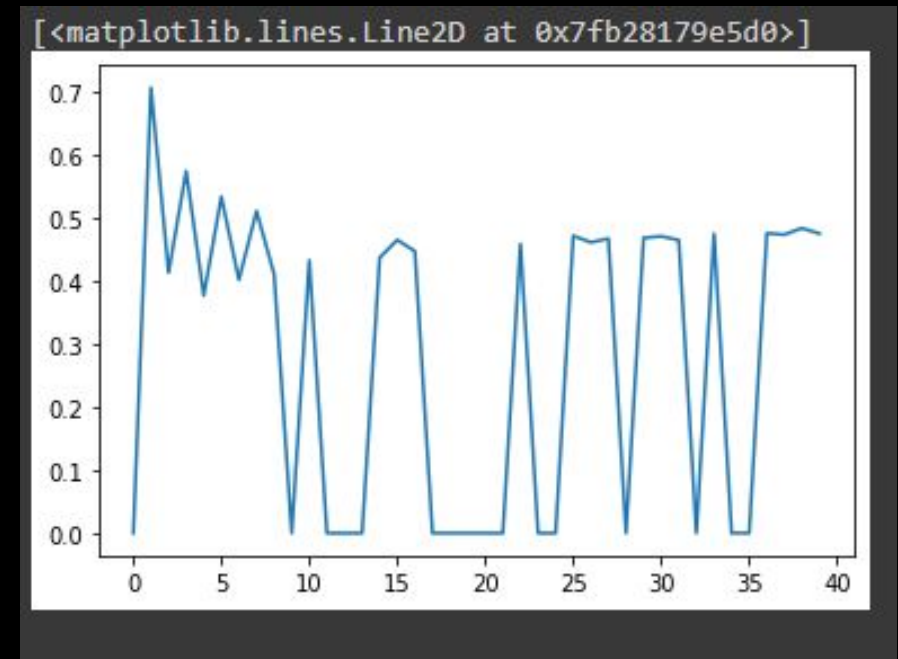
The bottom right corner of the slide features a horizontal panel with a smooth, flowing, wavy pattern in shades of blue and teal, resembling liquid or a digital signal, set against a dark background.

# AUTOENCODER



# AUTOENCODER APLICADO A MFCC

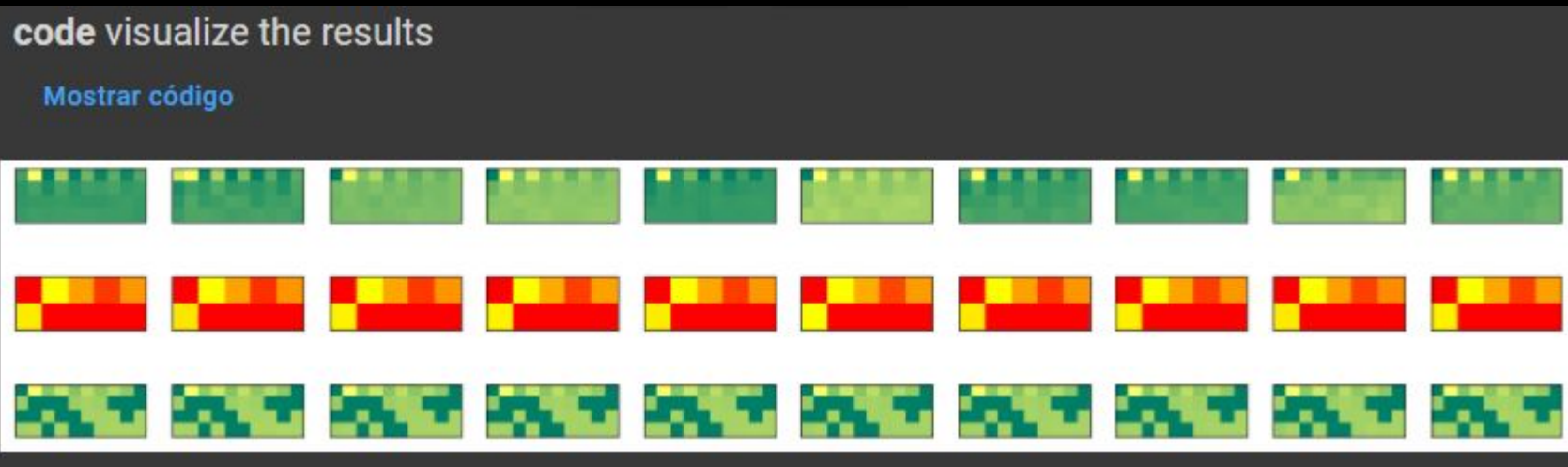
- Con este ejercicio se busca recrear los Coeficientes Cepstrales en las Frecuencias de Mel (MFCC) los cuales son coeficientes para la representación del habla basados en la percepción auditiva, esto con el fin de dar un soporte alternativo a la recreación de la música con un análisis alternativo de las pistas de audio





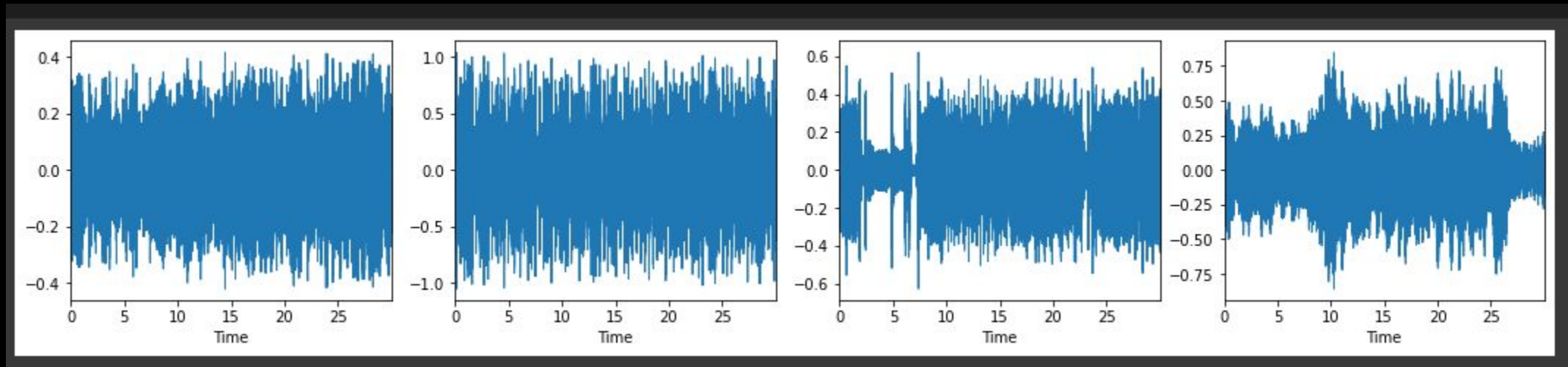
# RESULTADOS

Los resultados de la aplicación de autoencoders a MFCC no resultaron ser los esperados, esto a mostrar un estancamiento en fases tempranas del entrenamiento del modelo con resultados muy inferiores, apenas distinguibles entre pistas de audio



# DESARROLLO

- A continuación podemos ver la comparación de 2 audios antes y después de ser procesados por un autoencoder, podemos ver cómo si bien el audio se recrea en términos generales también se agrega una cantidad considerable de ruido al pasar por el modelo



# AUDIOS REAL VS GENERADO

Audio Real



Audio generado

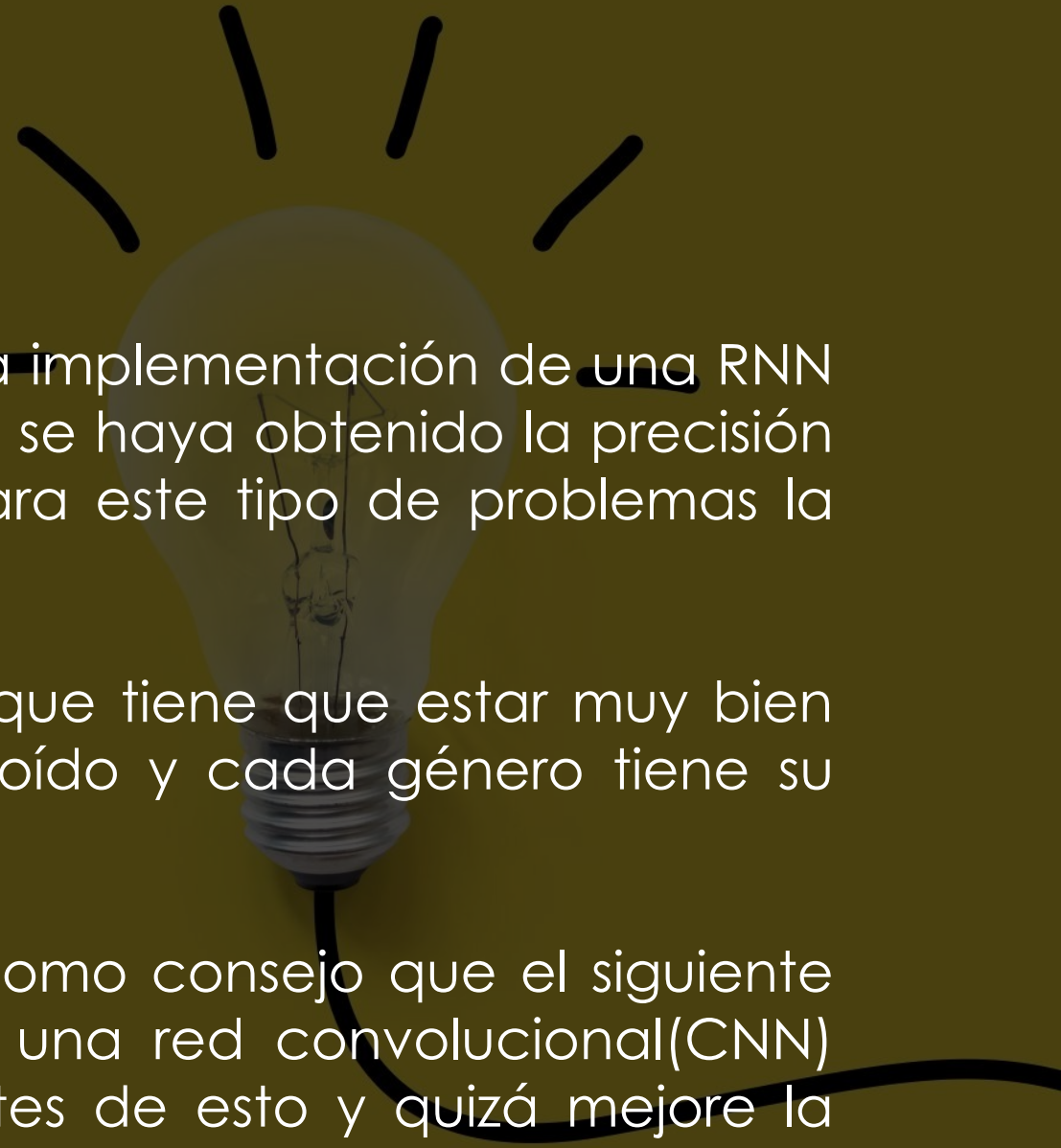


# CONCLUSIONES

Para este proyecto funcionó mucho mejor la implementación de una RNN para la clasificación de géneros aunque no se haya obtenido la precisión esperada y esto puede deberse a que para este tipo de problemas la “memoria” es bastante relevante.

La música tiene un ritmo y una secuencia que tiene que estar muy bien concatenada para que sea amigable al oído y cada género tiene su “ítem” que lo hace especial y diferente.

Este es un proyecto en marcha y se deja como consejo que el siguiente paso a seguir, sea la implementación de una red convolucional(CNN) embebida a una RNN, existen antecedentes de esto y quizá mejore la precisión con la cual se clasifica.





GRACIAS POR  
SU ATENCIÓN!

