PROCESADO DE AUDIO Y VOZ

MEMORIA AMPLIACIÓN P4

CLASIFICACIÓN Y VERIFICACIÓN DE LOCUTOR MEDIANTE DEEP NEURAL NETWORK

Pablo Marin
Artur Hellin

ÍNDICE

1. Introducción

Para esta ampliación se ha utilizado y modificado el repositorio de pytorch: https://github.com/santi-pdp/pav_spkid_pytorch

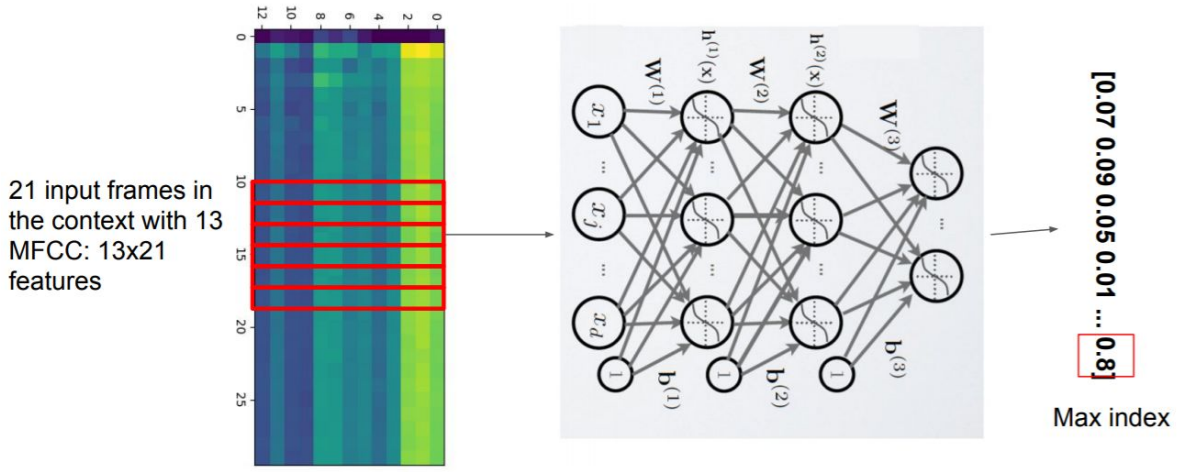
En él entrenaremos una red neuronal para, teniendo como input las características de audio previamente extraídas de unas tramas de voz (en este caso utilizaremos LPCC (cepstrum LPC), ya que fueron las características que mejor funcionaron en estas tareas usando gmm), realizar una tarea de clasificación y otra de verificación de locutor.

2. La red neuronal

Utilizaremos una red neuronal con la siguiente arquitectura:

```
model = nn.Sequential(nn.Linear(cfg['input_dim'] * cfg['in_frames'],
                                cfg['hsize']),
                      nn.ReLU(),
                      nn.Linear(cfg['hsize'], cfg['hsize']),
                      nn.ReLU(),
                      nn.Linear(cfg['hsize'], cfg['hsize']),
                      nn.ReLU(),
                      nn.Linear(cfg['hsize'], cfg['num_spks']),
                      nn.LogSoftmax(dim=1))
```

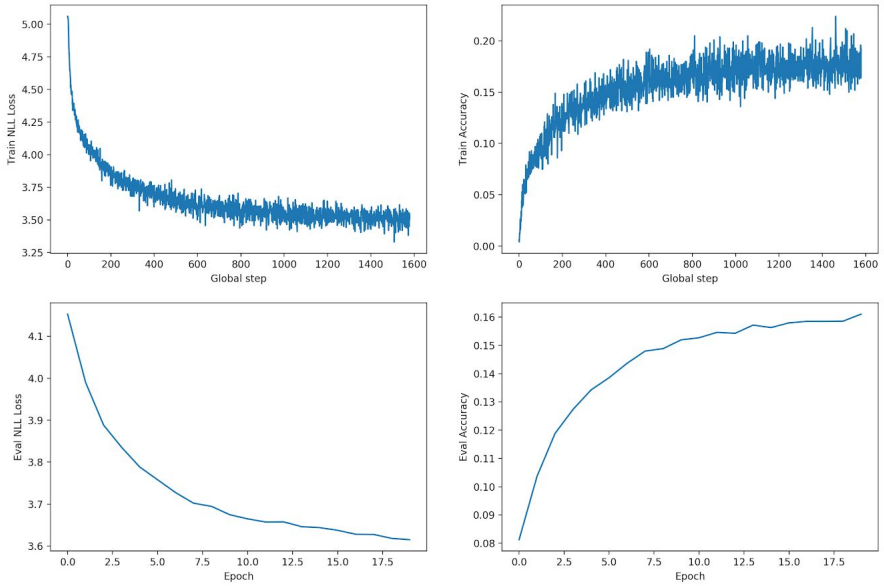
La capa de entrada tendrá tantas inputs como coeficientes LPCC tenga cada frame multiplicado por el número de frames que se le suministra (en la siguiente Figura se puede ver un ejemplo con 21 frames de 13 coefs/frame)



La última capa tendrá tantas neuronas como locutores haya en la base de datos, y cada neurona se activará proporcionalmente a la probabilidad de que la entrada de la red se identifique con el locutor asociado a la misma.

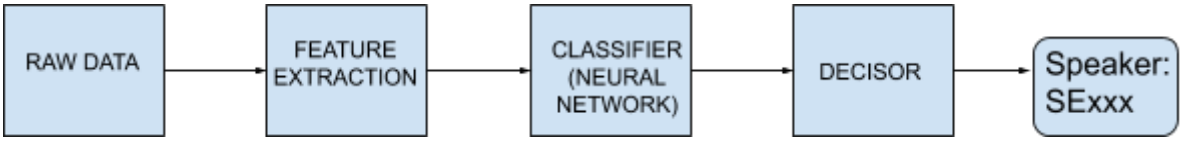
Después de varias pruebas (no tantas como nos hubiese gustado ya que cada entrenamiento lleva un tiempo considerable) hemos obtenido unos resultados interesantes con la siguiente arquitectura:

In frames = 25
Hidden Layer Size = 30
Batch Size = 1000
Patience = 10
Learning Rate = 0.001
Epoch = 20



3. Clasificación

La tarea de clasificación ha resultado sencilla dado que venía ya implementada. El sistema de clasificación es el siguiente:



Tan solo se han tenido que modificar ciertos parámetros referentes a la arquitectura de la red (vistos anteriormente) y crear el script *class_score* para evaluar los resultados usando las funciones confusion_matrix y classification_report de la librería *sklearn*:

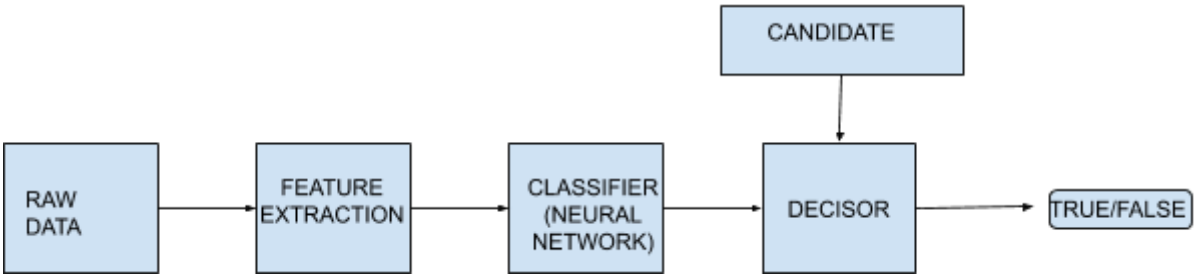
	precision	recall	f1-score	support
accuracy	-	-	0.94	785
macro avg	0.95	0.94	0.94	785
weighted avg	0.95	0.94	0.94	785

En el anexo se muestra el desglose por locutor.

4. Verificación

Esta tarea ha llevado un poco más de trabajo. Para empezar hemos creado el script *verify.py* en el que, utilzando los mismos ficheros que en la implementación con gmm, realizamos la verificación de un usuario que alega ser un locutor.

La implementación ha sido sencilla algorítmicamente hablando, ya que para realizar la verificación simplemente comprobamos si el locutor que predice la red neuronal coincide con el locutor que el candidato alega ser. El esquema del sistema sería el siguiente:



El script *verify.py* crea un fichero verify_res.log, con el formato:

Nombre_Real_Locutor Locutor_Predicho Candidato

```
63   BLOCK08/SES082/SA082S21 SES082  SES082
64   BLOCK08/SES082/SA082S24 SES082  SES082
65   BLOCK08/SES082/SA082S28 SES082  SES082
66   BLOCK08/SES083/SA083S14 SES207  SES083
67   BLOCK08/SES083/SA083S15 SES083  SES083
68   BLOCK08/SES083/SA083S24 SES277  SES083
```

Este formato nos ayuda a la hora de calcular el coste de nuestro sistema, basado en la misma definición que para el sistema basado en gmm:

$$C = \frac{1}{\text{mín}(\rho, 1-\rho)} \left(\rho \cdot p_m + (1-\rho) \cdot p_{fa}\right) \cdot 100$$

Donde:

p_m	porcentaje de fallos del sistema (bloqueo) frente a usuarios legítimos
p_{fa}	porcentaje de fallos del sistema (acceso) frente a impostores
ρ	parámetro de definición del objetivo (*target*). En este caso, se ha elegido $\rho = 0.01$, que penaliza mucho los accesos sin autorización (una falsa alarma cuenta como 99 bloqueos a usuarios legítimos).

Consideramos un fallo por bloqueo siempre que la identidad del locutor sea la misma que la del candidato pero la red no identifique correctamente al locutor, y consideramos un fallo por acceso siempre que el candidato no sea el verdadero locutor pero la red identifique erróneamente al locutor como el candidato.

De esta manera obtenemos el computo del coste de nuestro sistema:

False Negative Ratio: 0.048
False Positive Ratio: 0.001
Cost (rho = 0.01): 14.7

5. Conclusiones

Una vez finalizada la implementación del sistema con deep learning, no podemos evitar realizar unas comparaciones entre este método y el método basado en gmm.

El método clásico, las gmm, ofrece una muy buena aproximación al problema tanto de clasificación como de detección. La principal diferencia que hemos apreciado, pero, ha sido la cantidad de código necesario para una u otra implementación, siendo la última, con deep learning, la más simple.

Por contra, con el método clásico se puede ser 100% consciente de que está sucediendo, analizando las gmm asociadas a cada locutor. Esto se pierde al utilizar deep learning, ya que el ajuste de los pesos de la red no te dan la misma información (se podría llegar a entender, pero no es para nada intuitiva) y es simplemente de los resultados de donde podemos sacar conclusiones.

Por último, comentar que la realización de estas prácticas nos han llevado a comprender de una manera mucho mas tangible toda esta teoría que puede llegar a ser un poco abstracta si no se pone una aplicación clara delante. Consideramos que hemos aprendido mucho, mejorando en el camino nuestra habilidad como programadores.

Anexo: Evaluación del clasificador para cada locutor

speaker	precision	recall	f1-score	support
000	1.00	1.00	1.00	5
001	1.00	0.60	0.75	5
002	1.00	1.00	1.00	5
003	1.00	1.00	1.00	5
004	1.00	1.00	1.00	5
005	1.00	1.00	1.00	5
006	1.00	0.60	0.75	5
007	0.83	1.00	0.91	5
008	1.00	1.00	1.00	5
009	1.00	1.00	1.00	5
010	1.00	1.00	1.00	5
011	1.00	1.00	1.00	5
012	1.00	1.00	1.00	5
013	1.00	1.00	1.00	5
014	1.00	1.00	1.00	5
015	1.00	1.00	1.00	5
016	1.00	1.00	1.00	5
017	1.00	1.00	1.00	5
019	1.00	1.00	1.00	5
020	1.00	0.80	0.89	5
021	0.83	1.00	0.91	5
022	1.00	1.00	1.00	5
023	1.00	1.00	1.00	5
024	1.00	1.00	1.00	5
025	1.00	0.60	0.75	5
026	1.00	0.80	0.89	5
027	1.00	1.00	1.00	5
029	1.00	1.00	1.00	5
040	1.00	1.00	1.00	5
042	1.00	1.00	1.00	5
043	1.00	0.80	0.89	5
044	0.83	1.00	0.91	5
045	1.00	1.00	1.00	5
046	1.00	0.80	0.89	5
047	1.00	1.00	1.00	5
048	1.00	1.00	1.00	5
049	1.00	1.00	1.00	5
060	1.00	1.00	1.00	5
061	1.00	1.00	1.00	5
062	1.00	1.00	1.00	5
063	1.00	1.00	1.00	5
064	1.00	1.00	1.00	5
065	1.00	1.00	1.00	5
066	1.00	1.00	1.00	5

speaker	precision	recall	f1-score	support
067	1.00	1.00	1.00	5
068	1.00	1.00	1.00	5
069	1.00	1.00	1.00	5
080	0.50	0.60	0.55	5
081	1.00	1.00	1.00	5
082	1.00	1.00	1.00	5
083	1.00	1.00	1.00	5
084	1.00	1.00	1.00	5
085	1.00	1.00	1.00	5
086	0.62	1.00	0.77	5
087	1.00	0.80	0.89	5
088	1.00	1.00	1.00	5
089	1.00	1.00	1.00	5
097	1.00	1.00	1.00	5
098	1.00	1.00	1.00	5
099	1.00	1.00	1.00	5
100	1.00	1.00	1.00	5
101	1.00	0.60	0.75	5
102	0.71	1.00	0.83	5
103	1.00	1.00	1.00	5
104	0.83	1.00	0.91	5
105	1.00	1.00	1.00	5
106	1.00	1.00	1.00	5
107	1.00	1.00	1.00	5
108	1.00	1.00	1.00	5
109	1.00	0.80	0.89	5
117	1.00	1.00	1.00	5
118	0.83	1.00	0.91	5
119	0.83	1.00	0.91	5
120	1.00	1.00	1.00	5
121	1.00	1.00	1.00	5
122	1.00	1.00	1.00	5
123	1.00	0.80	0.89	5
124	1.00	0.80	0.89	5
125	1.00	1.00	1.00	5
126	1.00	0.80	0.89	5
127	1.00	0.60	0.75	5
128	0.80	0.80	0.80	5
129	1.00	1.00	1.00	5
130	1.00	0.40	0.57	5
131	1.00	1.00	1.00	5
132	1.00	1.00	1.00	5
133	1.00	1.00	1.00	5
140	1.00	0.80	0.89	5
141	1.00	1.00	1.00	5
142	1.00	1.00	1.00	5
143	1.00	1.00	1.00	5
144	1.00	1.00	1.00	5
145	1.00	1.00	1.00	5
146	0.71	1.00	0.83	5
147	1.00	1.00	1.00	5

speaker	precision	recall	f1-score	support
148	1.00	1.00	1.00	5
149	1.00	1.00	1.00	5
157	1.00	1.00	1.00	5
158	0.83	1.00	0.91	5
159	0.62	1.00	0.77	5
160	1.00	1.00	1.00	5
161	0.83	1.00	0.91	5
162	1.00	1.00	1.00	5
163	1.00	0.60	0.75	5
164	1.00	1.00	1.00	5
165	0.71	1.00	0.83	5
166	1.00	1.00	1.00	5
167	1.00	1.00	1.00	5
168	1.00	1.00	1.00	5
169	1.00	1.00	1.00	5
170	0.83	1.00	0.91	5
171	1.00	1.00	1.00	5
172	1.00	1.00	1.00	5
173	1.00	1.00	1.00	5
175	1.00	0.60	0.75	5
177	0.80	0.80	0.80	5
180	1.00	0.80	0.89	5
182	0.83	1.00	0.91	5
183	1.00	1.00	1.00	5
184	1.00	1.00	1.00	5
185	1.00	0.20	0.33	5
186	1.00	1.00	1.00	5
187	0.83	1.00	0.91	5
188	1.00	1.00	1.00	5
189	1.00	1.00	1.00	5
200	0.80	0.80	0.80	5
201	0.83	1.00	0.91	5
202	1.00	1.00	1.00	5
203	1.00	1.00	1.00	5
204	1.00	1.00	1.00	5
205	0.83	1.00	0.91	5
206	0.71	1.00	0.83	5
207	1.00	0.80	0.89	5
209	0.71	1.00	0.83	5
220	1.00	1.00	1.00	5
221	1.00	1.00	1.00	5
223	1.00	1.00	1.00	5
225	1.00	1.00	1.00	5
226	1.00	0.80	0.89	5
227	1.00	1.00	1.00	5
257	1.00	1.00	1.00	5
258	1.00	1.00	1.00	5
259	1.00	0.80	0.89	5
270	1.00	1.00	1.00	5
271	1.00	0.60	0.75	5
272	1.00	1.00	1.00	5
273	1.00	1.00	1.00	5

274	1.00	1.00	1.00	5
275	1.00	1.00	1.00	5
276	0.62	1.00	0.77	5
277	0.71	1.00	0.83	5
278	0.80	0.80	0.80	5
279	1.00	0.80	0.89	5
290	0.71	1.00	0.83	5
291	0.62	1.00	0.77	5
292	0.80	0.80	0.80	5
294	1.00	0.60	0.75	5
	precision	recall	f1-score	support
accuracy			0.94	785
macro avg	0.95	0.94	0.94	785
weighted avg	0.95	0.94	0.94	785