Práctica 5 - TDS

Entrenamiento y testeo de la red neuronal

Práctica 5

Están las instrucciones en el apartado 5.1, pero comentadas.
Se dan los ficheros **train.mat** y **validation.mat**

OBJETIVOS

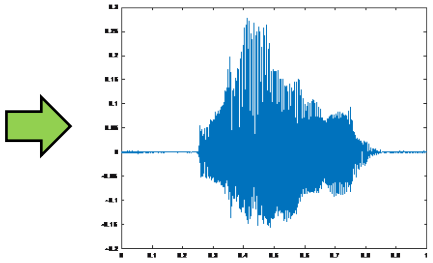
1. Extraeremos las características de toda nuestra base de datos. Para ello, primero construiremos los almacenes de datos necesarios para entrenar la red: *training* y *validation*.
2. A partir de la función Extrae_Características, programada en la práctica 4, extraeremos las características de ambos conjuntos para entrenar la red.
3. Entrenaremos la red y experimentaremos la influencia que los diferentes hiperparámetros tienen en la precisión obtenida.
4. Testearemos la red con nuevos audios

Importante:

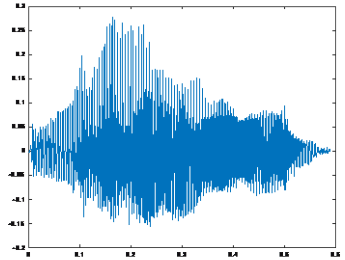
- Se proporciona el fichero P5_alumno.mlx para realizar la práctica.
- Este **fichero se debe entregar antes de la siguiente sesión junto con su impresión digital en pdf** (se puede hacer desde Matlab). Se valorará que el fichero tenga comentarios explicativos
- Recordad que tenéis que tener instalada la "Audio Toolbox" y la "Deep Learning Toolbox" y la versión de Matlab debe ser 2021 o superior.

Esquema del clasificador

La **Base de Datos** es la construida por vuestros audios en castellano

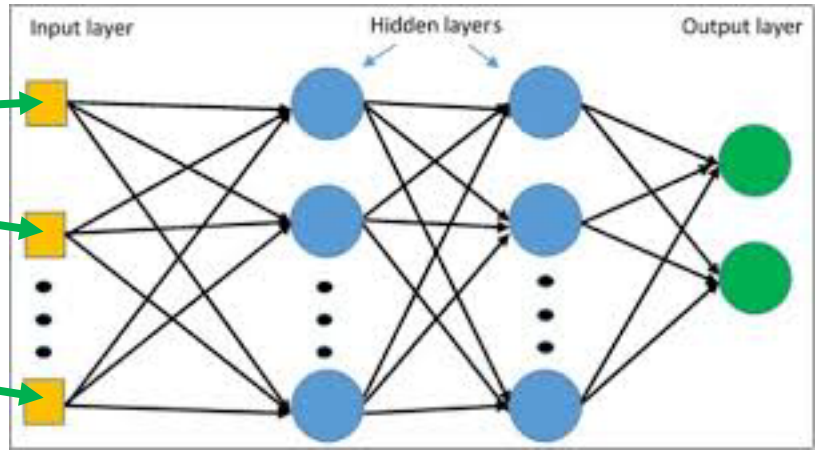


Voice Activity Detector



Extrae_Caracteristicas.m

$$\begin{bmatrix} feature_1 \\ feature_2 \\ \vdots \\ feature_{85} \end{bmatrix}$$



Prácticas 5 y 6

5.2: Normalización de características

- Una vez creados los dos set de datos, *train* y *validation*, hay que extraer las características de ambos conjuntos y almacenarlas en dos matrices, así como dos vectores con las etiquetas.
- Hemos utilizado la función *Extrae_Características* programada en la práctica 4. Este proceso es costoso por lo que una vez extraídas, deben guardarse las matrices y vectores para su posterior uso. Para la realización de la práctica **os proporcionamos directamente las matrices de características** en dos ficheros .mat:
  - **train.mat**: contiene la matriz `features_train`, que contiene la matriz de características del conjunto de entrenamiento, y el vector `labels_train`, que contiene el vector de etiquetas.
  - **validation.mat** : contiene la matriz `features_validation`, que contiene la matriz de características del conjunto de validación, y el vector `labels_validation`, que contiene el vector de etiquetas.
- Normalización de características para train.mat

$$\begin{bmatrix} & \cdots & \\ \vdots & \ddots & \vdots \\ & \cdots & \end{bmatrix} = \text{features\_train}$$ matriz de tamaño [12.808 x 85], hay que normalizarla por columnas

  - Se calcula la media M por columnas y la desviación típica S por columnas. Se obtendrán dos vectores de tamaño 1 x 85
  - Se normalizan las matrices de entrenamiento y validación con **la misma media y desviación**

$$features\_trainN = (features\_train - M)./S$$

- Normalización de características para validation.mat: ¡Se usa la media y varianza obtenida en train.mat!

5.3: Entrenamiento de la red

- En esta sección aprenderemos a definir la estructura de una red, se entrenará y se inspeccionará su funcionamiento.
- Definición de la arquitectura de red:

```
layers = [ ...
imageInputLayer([1 1 nFeatures]);
fullyConnectedLayer(n_1);
batchNormalizationLayer
reluLayer
….
softmaxLayer
classificationLayer
];
```

5.3: Entrenamiento de la red

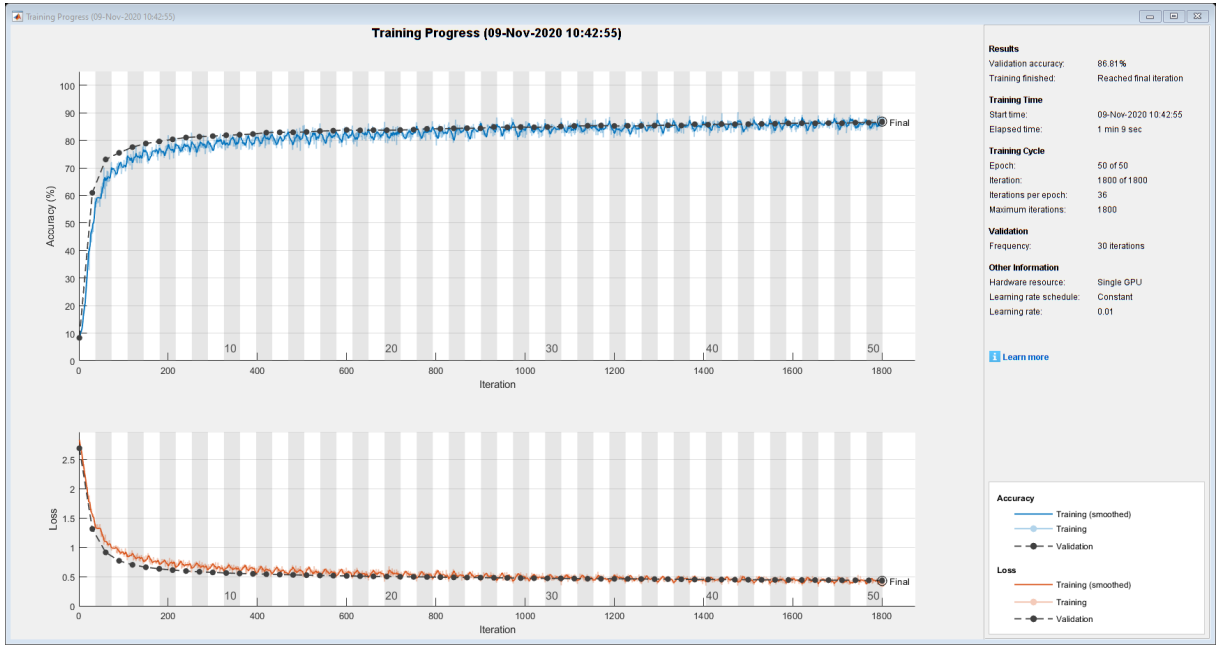
- Definición de las opciones del optimizador

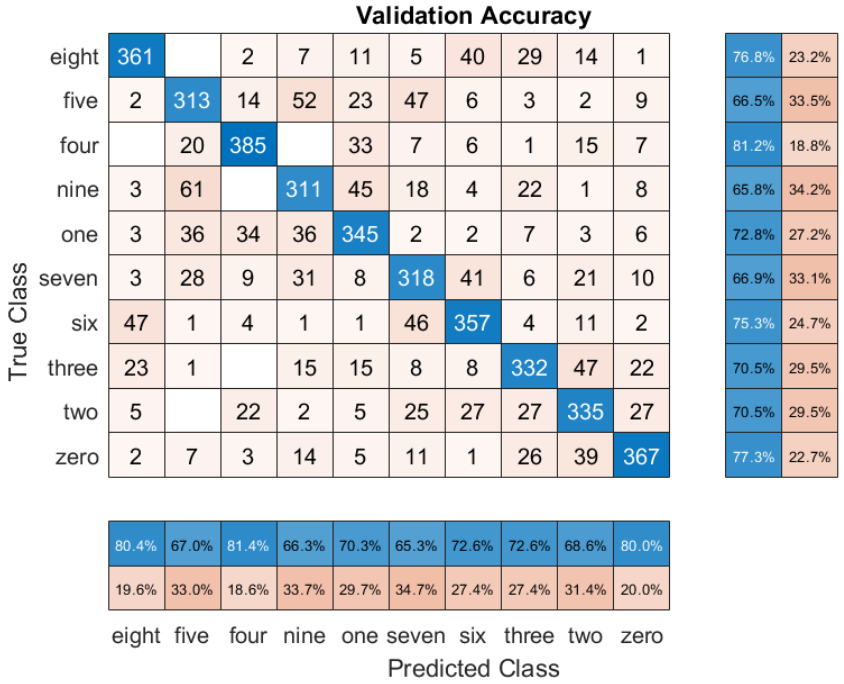
```
options = trainingOptions('sgdm', ...
'InitialLearnRate',0.01, ...
'MaxEpochs’,10, ...
'ValidationFrequency’,30,...
'ValidationData',{X_val_new,labels_validation}, ...
'MiniBatchSize',miniBatchSize, ...
'Verbose',false, ...
‘Plots','training-progress’);
```

5.3: Entrenamiento de la red

- Entrenamiento de la red

trainedNet = trainNetwork(X_train, labels_train, layers, options);





5.4: Efecto de la tasa de aprendizaje y el número de épocas

- En esta sección vamos a ver cómo afecta la tasa de aprendizaje y el número de épocas sobre al entrenamiento de la red
- Evaluaremos la *accuracy* final y el tiempo de procesado de cada nueva configuración de red (recuerda que la configuración de red también se cambia cada vez que cambiamos algún parámetro en options)

- Los ejercicios de esta sección van destinados a conseguir que la red acabe el aprendizaje de dos formas:
  - Haciendo que aprenda más rápido aumentando la tasa de aprendizaje.
  - Aumentando el número de épocas.

5.5: Efecto del tamaño del batch

- En esta sección vamos a estudiar qué influencia tiene el *batch size* en el tiempo de ejecución.
- Haremos una serie de ejercicios destinados a experimentar qué pasa con el tiempo de ejecución y la *accuracy* cuando aumentamos o disminuimos el tamaño del *batch*

5.6: Cambio en la estructura de la red

- En esta sección vamos a cambiar la estructura de la red con el fin de aumentar la *accuracy*:
  - Aumentando el número de capas para hacer una red más profunda.
  - Aumentando el número de neuronas por capa.

- Veremos los efectos de estos cambios y cómo solucionar los problemas que aparezcan.

Ejercicio 5.7: Testeo de la red

- En este ejercicio se pretende testear nuestra red entrenada (trainedNet) en la predicción de la etiqueta de un nuevo audio de entrada según el siguiente esquema: