Clasificación de sonidos respiratorios

Sistemas inteligentes

Definición del problema

El objetivo es desarrollar un modelo de clasificación basado en una red neuronal convolucional tiene como objetivo que a partir de las distintas características extraídas de múltiples audios respiratorios en formato .WAV sea capaz mediante algoritmos de clasificación determinar si un sonido respiratorio presenta alguna enfermedad o está sano.

Conjunto de datos

El modelo será entrenado y probado con dos distintos subconjuntos de datos del conjunto de datos llamado “Respiratory Sound Database”, proveído por Kaggle y conformado por novecientas veinte grabaciones de longitud variable entre un rango de diez y noventa segundos, donde se incluyen sonidos limpios de respiración y grabaciones ruidosas que simular condiciones de la vida real.

Estado del arte

Redes convolucionales

Las redes convolucionales tienen sus orígenes en la década de los ochenta, pero han sido recientemente adaptadas como un método de elección para varias tareas de clasificación de objetos, comúnmente de imágenes pero también en clasificación de sonidos ambientales o géneros (Piczak, 2015, 1).

Representación gráfica

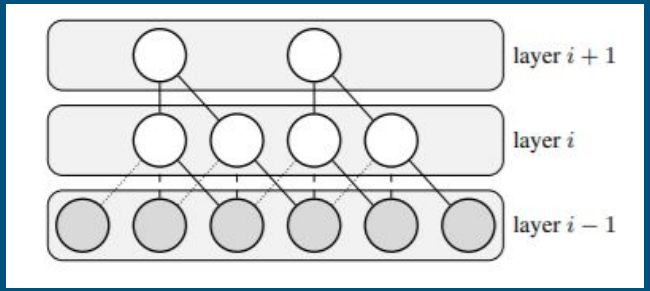

Figura 1. Cada unidad oculta, en lugar de estar conectada a todas las entradas provenientes de la capa anterior, se limita a procesar solo una pequeña parte de todo el espacio de entrada.

Clasificación de sonidos urbanos

Uno de los primeros experimentos en probar la eficacia de las redes neuronales convolucionales en tareas relacionadas a la clasificación de sonidos se realizó sobre un conjunto de datos de sonidos ambientales, este experimento muestra que un modelo convolucional alcanza un nivel similar al de otros métodos de aprendizaje de características.

Matriz de confusión

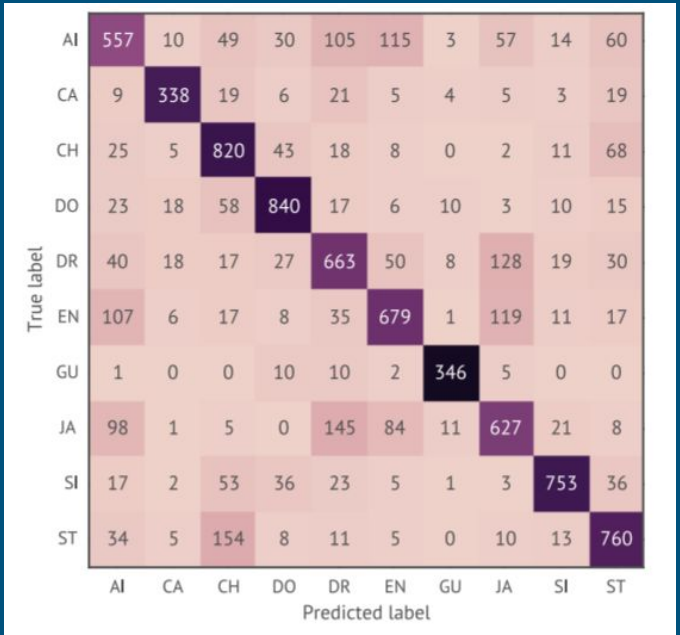

Figura 2. Los modelos basados en una red neuronal convolucional funcionaron mejor que las implementaciones respectivas que utilizan características diseñadas manualmente.

Clasificación musical en Spotify

Aplica técnicas de aprendizaje automático sobre los datos que la plataforma puede recopilar como la música que otros usuarios con un gusto similar escuchan, pero la parte importante es la extracción de características numéricas sobre contenido de las canciones, para descubrir si una canción es feliz o triste, el tempo que es la velocidad en la que una pieza musical es reproducida (Zimmer, 2020) u otras que se enfoquen en los elementos acústicos (Lazzaro, 2019).

Propuesta de solución

Librerías utilizadas

- Pandas.
- Librosa.
- Matplotlib.
- Numpy.
- Tensorflow.
- Sklearn.

Extracción de características

Cada señal de audio consta de muchas características. Sin embargo, debemos extraer las características que son relevantes para el problema que estamos tratando de resolver.

- Espectrograma.
- Coeficientes cepstrales en las frecuencias de Mel.
- Espectrogramas en escala Mel.
- Frecuencias de croma.

Pre-procesamiento

Antes de dividir el conjunto de datos, se tendrá que eliminar los diagnósticos con muy poca representación, en este caso, los diagnosticados con asma e infección del tracto respiratorio inferior. Posteriormente, se van a dividir los conjuntos de entrenamiento y de pruebas, el primer conjunto va a contener alrededor del ochenta por ciento de los datos debido a que la fase de entrenamiento es la que más información requiere y el segundo será utilizado para medir la exactitud de las predicciones generadas por el modelo.

Red neuronal convolucional

Analizando los distintos modelos disponibles y basado en los trabajos científicos generados, se optó por un modelo secuencial debido a la naturaleza del problema que solo requiere una entrada y una salida correspondiente, que será generado y entrenado mediante los métodos que la librería Keras provee.

Evaluación

El objetivo del modelo es evitar estar sobre-ajustado que es cuando funciona muy bien con los datos de entrenamiento, pero, su precisión disminuye cuando se utiliza con otros datos, por lo que no tendrá una aplicación real en el mundo real. También se busca evitar que esté desajustado, que es cuando ni siquiera puede funcionar correctamente con los datos de entrenamiento por lo que no puede realizar predicciones precisas.

También debido a que es un problema de clasificación, se usará una matriz de clasificación, que es una herramienta que permite la visualización del desempeño de un algoritmo que se emplea en aprendizaje supervisado. Donde cada columna representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real.

Desarrollo

Procesamiento de conjunto de datos

La siguiente función tiene como propósito listar todos los archivos en formato .WAV de un directorio, en este caso la dirección del directorio se le asignó de forma constante. Posteriormente, se hace uso de notación de conjuntos para crear dos compresiones de listas.

```
def get_wav_files():
  audio_path = '../input/respiratory-sound-database/Respiratory_Sound_Database/Respiratory
_Sound_Database/audio_and_txt_files/'
  files = [f for f in listdir(audio_path) if isfile(join(audio_path, f))]  # Gets all files
in dir
  wav_files = [f for f in files if f.endswith('.wav')]  # Gets wav files
  wav_files = sorted(wav_files)
  return wav_files, audio_path
```

Figura 3. Función “get_wav_files()” que contiene dos comprensiones de listas y retorna una tupla de valores.

In [11]:

```
def diagnosis_data():
  diagnosis = pd.read_csv('../input/respiratory-sound-database/Respiratory_Sound_Database/
Respiratory_Sound_Database/patient_diagnosis.csv')

  wav_files, audio_path = get_wav_files()
  diag_dict = { 101 : "URTI"}
  diagnosis_list = []

  for index , row in diagnosis.iterrows():
    diag_dict[row[0]] = row[1]

  c = 0
  for f in wav_files:
    diagnosis_list.append(Diagnosis(c, diag_dict[int(f[:3])], audio_path+f))
    c+=1

  return diagnosis_list
```

Figura 4. Función “diagnosis_data()” que carga la información del conjunto de diagnósticos y la agrega en una lista que es retornada.

Extracción de características

El proceso de extracción de características para usarlas en análisis es llamada extracción de características, para este paso se tiene que definir una función llamada “audio_features()” con un argumento.

```
def audio_features(filename):
  sound, sample_rate = librosa.load(filename)
  stft = np.abs(librosa.stft(sound))

  mfccs = np.mean(librosa.feature.mfcc(y=sound, sr=sample_rate, n_mfcc=40),axis=1)
  chroma = np.mean(librosa.feature.chroma_stft(S=stft, sr=sample_rate),axis=1)
  mel = np.mean(librosa.feature.melspectrogram(sound, sr=sample_rate),axis=1)
  contrast = np.mean(librosa.feature.spectral_contrast(S=stft, sr=sample_rate),axis=1)
  tonnetz = np.mean(librosa.feature.tonnetz(y=librosa.effects.harmonic(sound), sr=sample_r
ate),axis=1)

  concat = np.concatenate((mfccs,chroma,mel,contrast,tonnetz))
  return concat
```

Figura 5. Método que extrae las características de un audio y las retorna en un eje existente.