1st Summer School Physical Engineering UNI 2022-Lima,Perú

Reporte Deep Learning: Reconocimiento de Voz

Eduardo Steve Rodriguez Canales[1]

[1]Universidade de Sao Paulo
[1]edurcan@usp.br

February 28, 2022

Resumen

En este trabajo, un algoritmo basado en Deep Learning para el reconocimiento de comandos de voz es presentado. Usando una dataset conformada por un corpus de frases en inglés, una red neuronal convolucional unidimensional es entrenada para identificar las frases mediante el uso de Cross Entropy Loss para la clasificación y el algoritmo Adam para la optimización. Los resultados muestran la efectividad de nuestro modelo de red y el método de optimización usado en comparación al optimizador estándar SGD (stochastic gradient descendent).

Palabras Clave
reconocimiento de voz, red neuronal convolucional, deep learning, Cross Entropy Loss, Adam

1 Metodología

Para realizar este proyecto, se siguió la metodología propuesta en el curso de Deep Learning de la escuela de verano de Ingeniería Física de la Universidad Nacional de Ingeniería de Lima-Perú. Para entrenar y evaluar el algoritmo de reconocimiento de voz se usó el Dataset *Fluent Speech Commands* [1], este es un conjunto de datos públicos formado por 248 frases de 97 hablantes en 30046 declaraciones grabadas en 16 kHz en un único canal. El algoritmo, fue implementado usando Pytorch [2], una biblioteca de aprendizaje automático de código abierto basada en Python.

Los audios de la Database son preprocesados antes de introducirlos a la red neuronal. Primero se consirea una longitud máxima de 30000 puntos. Luego son normalizados dividiéndolos entre 4000 y haciendo un resampling de 8000 Hz, por último se establece un tamaño de Batch (lote) de 128. El reconocimiento de voz fue realizado mediante un modelo de red neuronal convolucional unidimensional. Esta operación, diseñada para capturar características de la voz, se obtiene mediante la función *Conv1d* y está descrita como

$$out(N_i, C_{out_j}) = bias(C_{out_j}) + \sum_{k=0}^{C_{in}-1} weight(C_{out_j}, k) \star input(N_i, k),$$

donde , N es el tamaño del batch, C denota el número de canales y L es la longitud de la señal. El modelo de red neuronal fue diseñado con 7 capas como se muestra en el siguiente código:

```
model = nn.Sequential(
    nn.Conv1d(1,16,100,bias=False),
    nn.BatchNorm1d(16),
    nn.ReLU(inplace=True),
    nn.MaxPool1d(4),
    nn.Conv1d(16,32,4,bias=False),
    nn.BatchNorm1d(32),
    nn.ReLU(inplace=True),
    nn.MaxPool1d(4),
    nn.Conv1d(32,64,4,bias=False),
    nn.BatchNorm1d(64),
    nn.ReLU(inplace=True),
    nn.MaxPool1d(4),
    nn.Conv1d(64,64,4,bias=False),
    nn.BatchNorm1d(64),
    nn.ReLU(inplace=True),
    nn.MaxPool1d(4),
    nn.Conv1d(64,128,4,bias=False),
    nn.BatchNorm1d(128),
    nn.ReLU(inplace=True),
    nn.MaxPool1d(4),
    nn.Conv1d(128,128,4,bias=False),
    nn.BatchNorm1d(128),
    nn.ReLU(inplace=True),
    nn.MaxPool1d(4),
    nn.Conv1d(128,256,4,bias=False),
    nn.BatchNorm1d(256),
    nn.ReLU(inplace=True),
    nn.Flatten(),
    nn.Linear(768,248)
).to(dev)
```

En la clasificación se usó el algoritmo Cross Entropy Loss, usado para problemas con varias clases, donde es necesario conocer la probabilidad de pertenencia a cada clase. Está descrito por

$$\mathcal{L} = -\sum_i t_i \log y_i \tag{1}$$

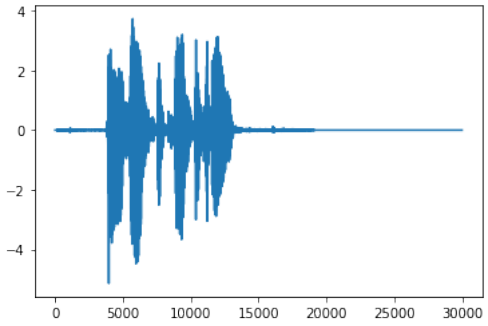

Figura 1: Ejemplo de onda de audio.

```
torch.Size([128, 1, 30000])
tensor([150,  33, 101,  15,  37, 193,  35, 245, 118, 207,  34, 208, 241,  36,
        112, 209, 119, 191, 247,  60, 173,  25,  58, 224, 194, 124,  96, 119,
        141, 201,  13,  26, 131, 101, 165,  36,  38, 242, 229,  13, 101, 150,
        141, 167, 224,  39,  95,  21, 150, 158, 210,   9, 136, 167, 187,  93,
        105,  64,  98, 166, 123, 186,  75, 235, 124,  12, 119, 124, 232,   9,
          4, 212,  96,  79,  92, 242, 145, 222, 189,  32, 152,  68, 195, 160,
        210,  18, 210,  55, 132,  21, 225, 146, 242, 138, 244, 136, 175,  68,
         75, 208, 107, 108, 205,  12, 190, 113,  64, 132, 210, 247, 206, 247,
        159, 216, 236, 233,  92, 242, 236, 232, 149, 185,  60, 151, 168,  37,
        244, 242])
```

Figura 2: Información del archivo de audio.

donde y es una probabilidad relacionada a los valores retornados y t una distribución de probabilidad relacionada con el entrenamiento.

Para la optimización se comparó dos optimizadores en la librería *Pytorch*, el *Adam* [3] y el convencional stochastic gradient descent (SGD). La clase *Adam* es uno de los optimizadores más populares con un algoritmo para la optimización basada en gradientes de primer orden de funciones objetivo estocásticas, basado en estimaciones adaptativas de momentos de orden inferior. En este proyecto se usó una tasa de aprendizaje $lr = 0{,}01$ y 10 épocas para el entrenamiento de la red.

```
crit = nn.CrossEntropyLoss()
optim = torch.optim.Adam(model.parameters(), lr=0.01)
train(model, train_dl, test_dl,crit,optim,epochs=10)
```

2 Resultados

En esta sección se muestran los resultados obtenidos en el entrenamiento de la red neuronal para el reconocimiento de voz usando los optimizadores SGD y Adam. La etapa de preprocesamiento de los archivos de audio del Data set puede ser evaluada de forma gráfica como se muestra en la Figura 1, donde se puede apreciar la onda sonora de un audio perteneciente a la frase *increase the temperature*, en esta figura se muestra la señal con 30000 puntos como se había previsto. La Figura 2 muestra información del archivo de audio, su tamaño de empaquetamiento de 128, su canal unitario y su tamaño de 30000, así como el tensor. Para la optimización del modelo se usaron dos métodos. El método de stochastic gradiente descent (SGD), como se muestra en la Figura 3 tuvo un tiempo de optimización de 10 minutos para 10 épocas con una

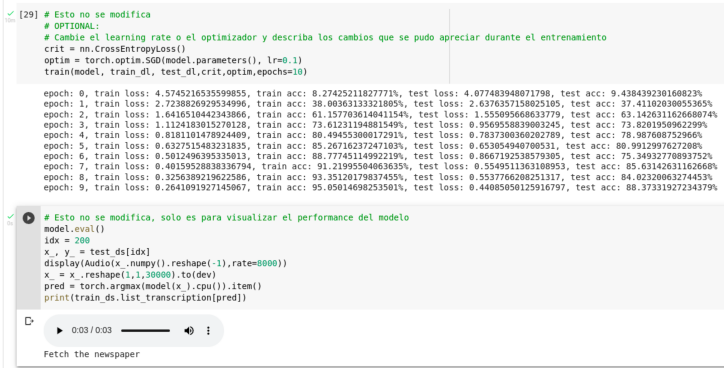

Figura 3: Método de optimización SGD.

Figura 4: Método de optimización Adam.

tasa de aprendizaje de lr=0.1. Alcanzó un desempeño en pérdida de entrenamiento fue 0.26 y precisión de entrenamiento 95.05. Luego se comprobó su funcionamiento con el audio *Fetch the newspaper*, identificando correctamente la frase. La Figura 4 muestra el método de optimización Adam. Este método tuvo un tiempo de optimización de 6 minutos para 10 épocas y una tasa de aprendizaje de lr=0.01. El método propuesto alcanzó un desempeño de pérdida de entrenamiento de 0.06 y precisión de entrenamiento de 98.05. Igual que en el método SGD se comprobó el funcionamiento con un audio, en este caso la frase *I couldn't hear anything, turn up the volume* fue identificada correctamente.

3 Discusión de resultados

Los resultados sugieren una superioridan en el método de optimización *Adam* sobre el método usado en clase SGD. El tiempo de optimización para el método *Adam* fue 4 minutos menor, incluso con una tasa de aprendizaje de 0.01. Además los parámetros de precisión de entrenamiento alcanzaron un valor de 98.05 contra 95.05 en el caso del optimizador SGD y de igual forma la pérdida de entrenamiento fue de solo 0.06 contra los 0.26 del método SGD. Un punto a destacar

es que prácticamente desde la primera época el método *Adam* superó al método SGD por lo que se podrían incluso reducir las épocas, lo que llevaría a un menor tiempo de entrenamiento, pero manteniendo un buen nivel de desempeño.

Conclusiones

En este proyecto se ha conseguido implementar una red neuronal para el reconocimiento de voz exitosamente. Para ello se usó la base de datos pública *Fluent Speech Commands* y se desarrolló un algoritmo de aprendizaje usando la biblioteca *Pytorch*. Una red convolucional unidimensional de 7 capas fue propuesta, usando el método de clasificación Cross Entropy Loss. Se comparó el desempeño de dos optimizadores el *Adam* y el SGD, mostrando el optimizador *Adam* mejores resultados de precisión y pérdida, así como un menor tiempo de entrenamiento. Trabajos futuros pueden incluir el uso de espectrogramas y redes convolucionales de dos dimensiones.

Agradecimientos

Agradezco al equipo organizador de la 1ra Escuela de Verano de Ingeniería Física de la UNI y en especial al Prof. Elvin Muñoz Vega por realizar el curso de Deep Learning.

Referencias

[1] (2021, Apr) Fluent speech commands: A dataset for spoken language understanding research. [Online]. Available: https://fluent.ai/fluent-speech-commands-a-dataset-\for-spoken-language-understanding-research/

[2] Pytorch documentation. [Online]. Available: https://pytorch.org/docs/stable/index.html

[3] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.