Universidade Federal do Rio Grande do Norte - CCET/DIMAp Aprendizado Profundo – DIM0446 – 2024.2

**Título do trabalho** Warlike Richard da Silva Soares

Natal, 29/01/2025.

**Resumo:** este documento tem como objetivo descrever um modelo de regressão que utiliza redes recorrentes para transformar um áudio de cantarolado em uma música tocada em piano. Para isso foram gravados os cantarolados a partir da base de dados Maestro v3 de arquivos MIDI.

# Introdução

Neste trabalho eu implementei uma IA que transforma áudios .wav de cantarolados em um outro áudio .wav com um som de piano, que visa corresponder ao cantarolado original. Para isso, o sistema transforma o áudio do cantarolado em um vetor com os embeddings extraídos do áudio, usando o modelo wav2vec. Esses embeddings passam então pelo modelo que utiliza RNN para ser treinado, gerando então um espectrograma que logo após será usado em outro modelo para gerar o áudio. Sendo assim, o sistema criado utiliza duas ferramentas para atingir o objetivo, já que o modelo puro apenas transforma os embeddings em espectrogramas.

# Fundamentação teórica

RNN, ou Rede Neural Recorrente, é um modelo de aprendizado profundo que leva em consideração a recorrência dos dados, ou seja, como as informações são dispostas ao longo do tempo, já que elas dependem das informações anteriores.

Podemos citar como exemplo o Chat GPT, já que para gerar uma resposta ele considera as palavras já geradas para gerar as próximas.

# O modelo implementado

O modelo utilizado foi o LSTM, que é uma RRN importada do keras.

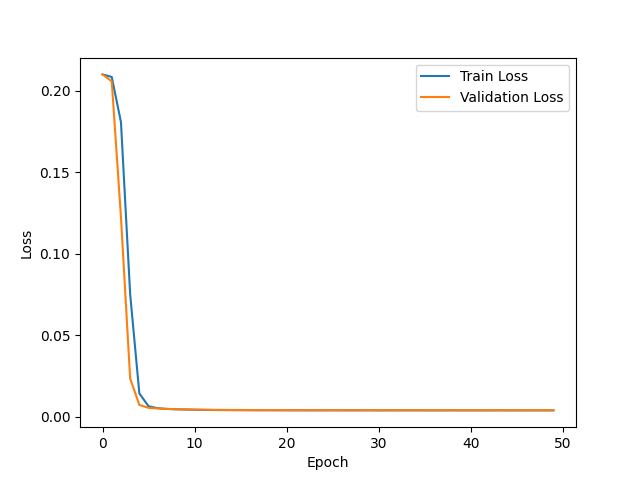
# Resultados

Para os dados de entrada (áudios .wav) foi usado o modelo wav2vec2, que extraiu os embeddings de cada áudio e os salvou em arquivos do tipo .npy.

Para os dados de saída (arquivos MIDI) primeiro foi necessário convertê-los para .wav utilizando o programa fluidsynth, para então converter os resultados para espectrogramas, utilizando a biblioteca librosa.

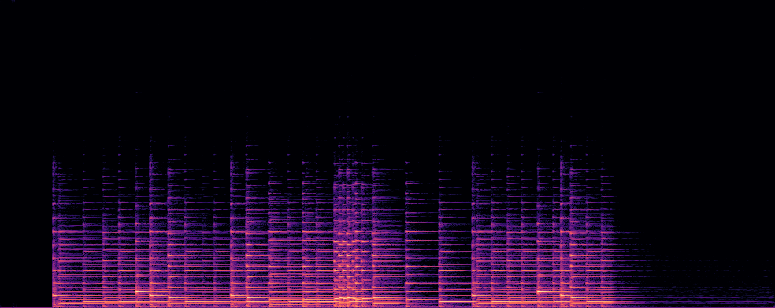
A base de dados foi dividida em 20% de validação e 80% de treino. Como o problema é de recorrência, não vi uma maneira clara de separar os dados de teste, então apenas criei novos áudios (cantarolados) para observar o resultado gerado.

Abaixo é possível ver o training loss e validation loss do modelo:

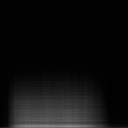


Como o problema abordado foi de regressão, não foi possível determinar a acurácia do modelo, nem criar uma matriz confusão. Foi observado, no entanto, que os espectrogramas gerados pareciam estar consistentes com os áudios, mas não foi viável determinar com precisão a qualidade do modelo pois houveram complicações ao tentar transformar esses espectrogramas em áudio.

Tentei utilizar o primeiro espectrograma da base de dados para comparar com um espectrograma gerado pelo modelo utilizando a mesma entrada de áudio, e pude observar a semelhança, o que não pode ser determinada com grande precisão devido à limitação de visualização desses dados em forma de áudio. Abaixo estão duas imagens comparando esses espectrogramas:



Espectrograma original, gerado diretamente de um dos áudios da base de dados.



Espectrograma gerado utilizando o modelo treinado.

O modelo foi treinado utilizando um processador Ryzen 5 5700x3D, localmente. Os códigos foram feitos utilizando o VS code como IDE, e os passos foram separados em diferentes arquivos.

Como utilizar o modelo: para executar e testar o programa, você pode executar o arquivo runModel.py, que em seguida irá solicitar um diretório/path para um áudio .wav. Após passado esse diretório, o modelo irá criar um espectrograma criado para representar uma música, assim como um novo arquivo de áudio gerado a partir desse espectrograma.

# Conclusão

Implementamos um modelo classificador de XXX baseado em YYYY. A base de dados utilizada foi ZZZ que foi particionada da seguinte forma: AA amostras para treinamento, BB amostras para validação e CC amostras para teste.

O modelo alcançou uma acurácia de UUU%.

Aqui estão algumas sugestões para melhoria: XXX, YYY, ZZZ

# Referências bibliográficas

https://aws.amazon.com/pt/what-is/recurrent-neural-network/