Urban Sound Classification

Aprendizagem Computacional II

30.novembro.2025

Francesco Renna

U.PORTO
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Trabalho realizado por:
Beatriz Pereira up202304769
Carolina Leite up202307856
Lara Gonçalves up202307857

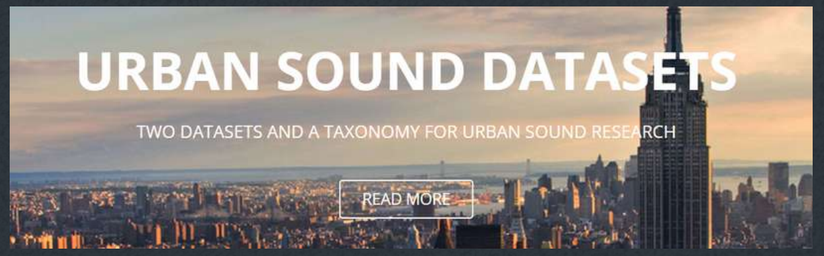
Índice

01

Dataset e Tratamento de Dados

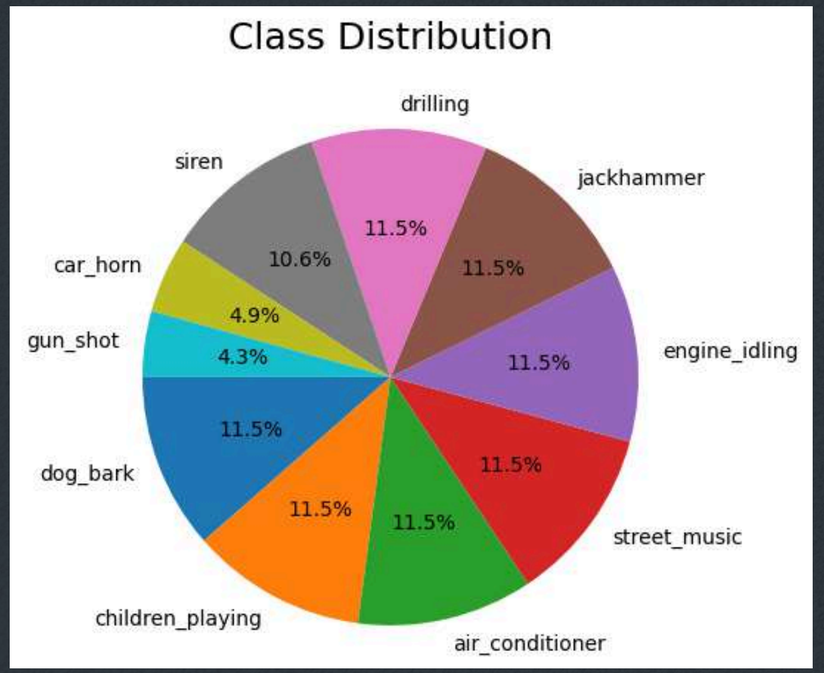
UrbanSound8K - visão geral

- Dataset amplamente utilizado para tarefas de classificação de áudio em ambientes urbanos e reais.



Características

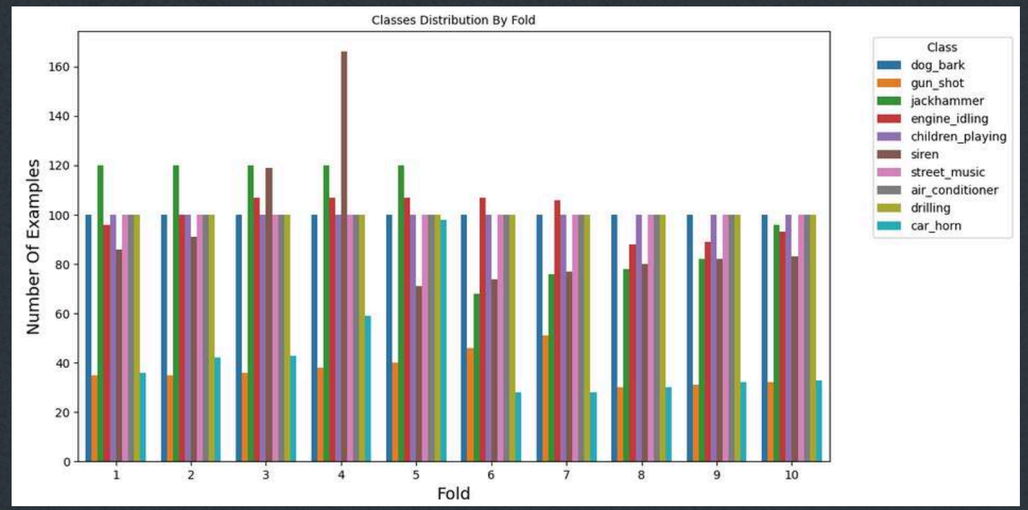
Número total de amostras	8 732 ficheiros de áudio
Formato	.wav
Duração	≤ 4 seg
Número de classes	10
Folds	10

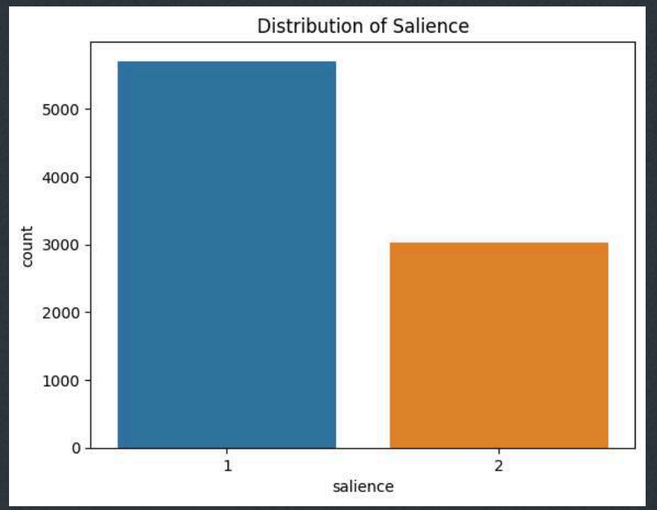


UrbanSound8K - visão geral

- Para além dos áudios, possui também um ficheiro **.csv** com metadados

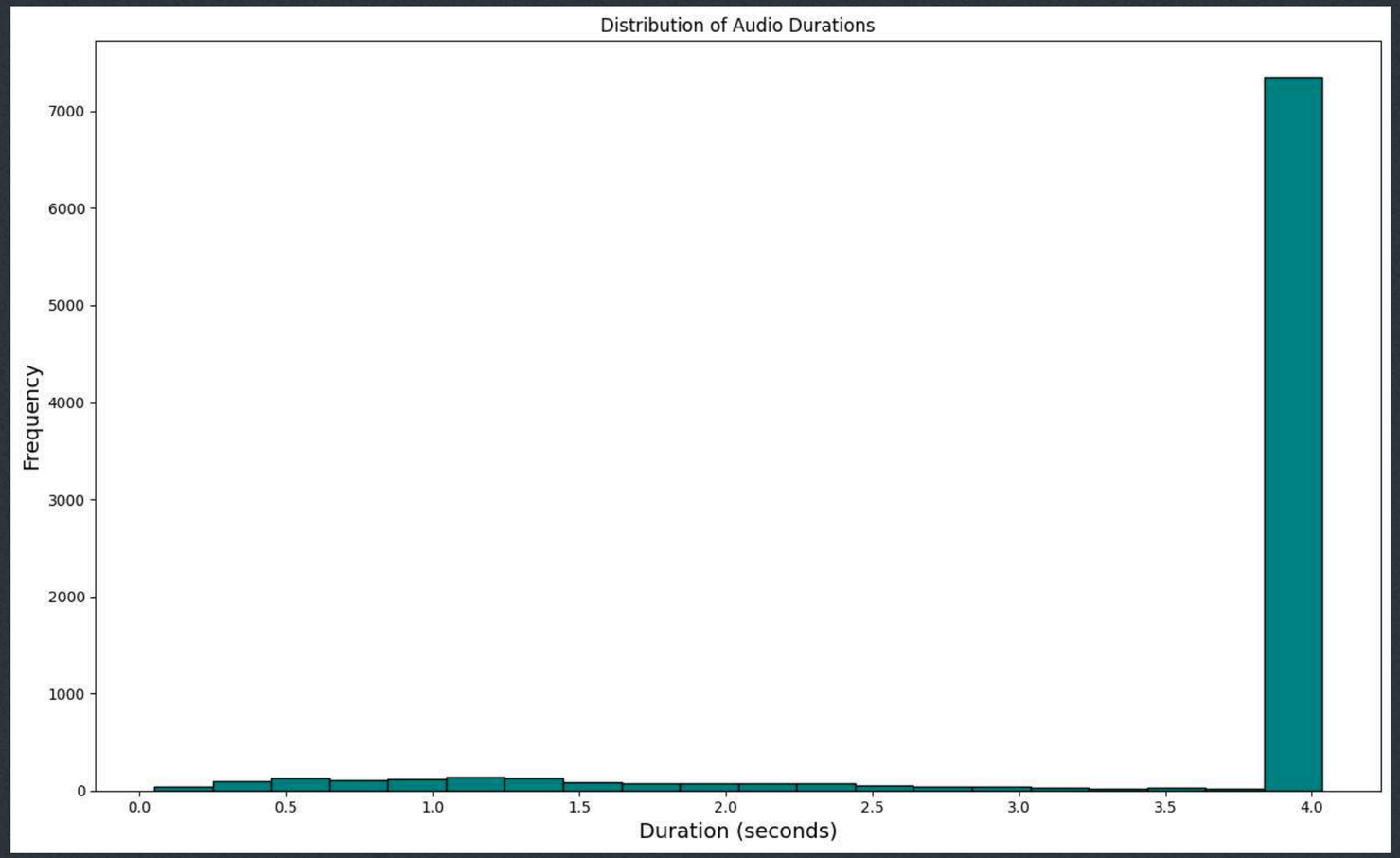
	slice_file_name	fsID	start	end	salience	fold	classID	class
0	100032-3-0-0.wav	100032	0.0	0.317551	1	5	3	dog_bark
1	100263-2-0-117.wav	100263	58.5	62.500000	1	5	2	children_playing
2	100263-2-0-121.wav	100263	60.5	64.500000	1	5	2	children_playing
3	100263-2-0-126.wav	100263	63.0	67.000000	1	5	2	children_playing
4	100263-2-0-137.wav	100263	68.5	72.500000	1	5	2	children_playing





Tratamento de dados

- Nem todos os áudios têm 4 segundos.



Que estratégia usar?

Zeropadding

Ideal para manter a integridade dos áudios

Tratamento de dados

Limpeza dos Dados

Valores nulos

Colunas com tipo object

Valores duplicados

Normalização

Tipos de dados das colunas

02

Escolha dos Classificadores

Porque é que escolhemos estes classificadores?

CNN - Convolutional Neural Networks

- Excelente para padrões espectrais (imagem)
- Aprende filtros robustos
- Estado da arte baseado em espectogramas
- **Componente espacial**

RNN - Recurrent Neural Networks

- Modela o áudio como sequência
- Capta dependências temporais
- Adequado para sons com ritmo/evolução
- **Componente temporal**

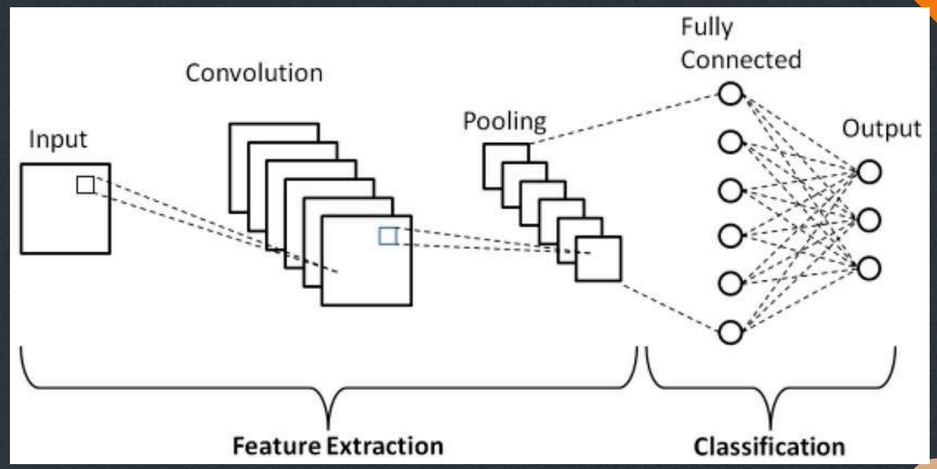
03

Modelo CNN

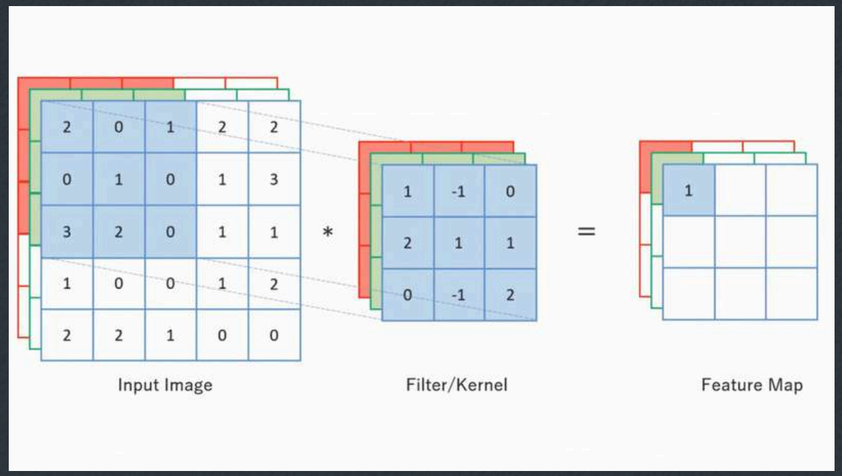
CNN - Convolutional Neural Networks

O que é?

- Redes neuronais especializadas em processar dados com estrutura de grelha, como imagens (ou espectrogramas de áudio).
- Utilizam operações de convolução (filtros) para detetar padrões locais e hierárquicos, independentemente da sua posição.



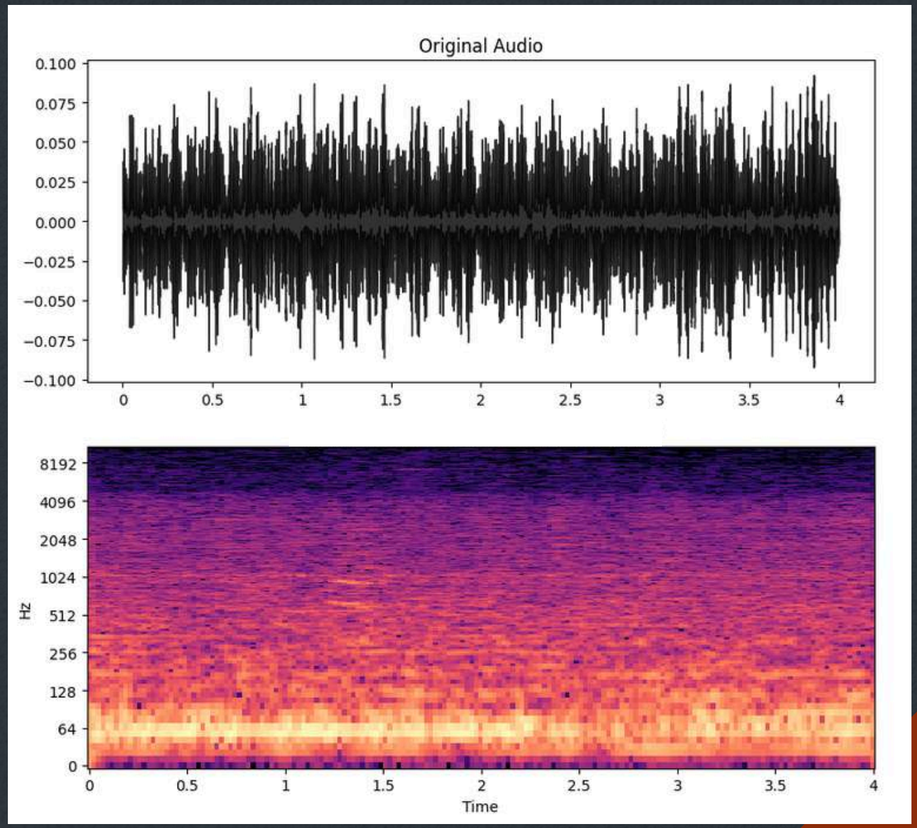
- Componentes fundamentais para a extração de características:
  - Camadas de Convolução (Extração de features).
  - Camadas de Pooling (Redução de dimensionalidade).



Input e Pré-processamento

Representação Espectro-Temporal

- **Abordagem de Visão Computacional:** Abandonámos a análise de sinal bruto (1D) a favor de Log-Mel Spectrograms (2D).
  - Esta transformação permite que o modelo CNN processe o som como uma imagem, identificando padrões visuais.
- **Definição dos Tensores:** Entrada padronizada em matrizes de (64, 174, 1).
  - 64: Bandas de Frequência (Eixo Y).
  - 174: Janelas Temporais Fixas (Eixo X).
- **Estabilidade Numérica:** Escala Logarítmica (dB) + Normalização Z-Score ($\mu=0, \sigma=1$).



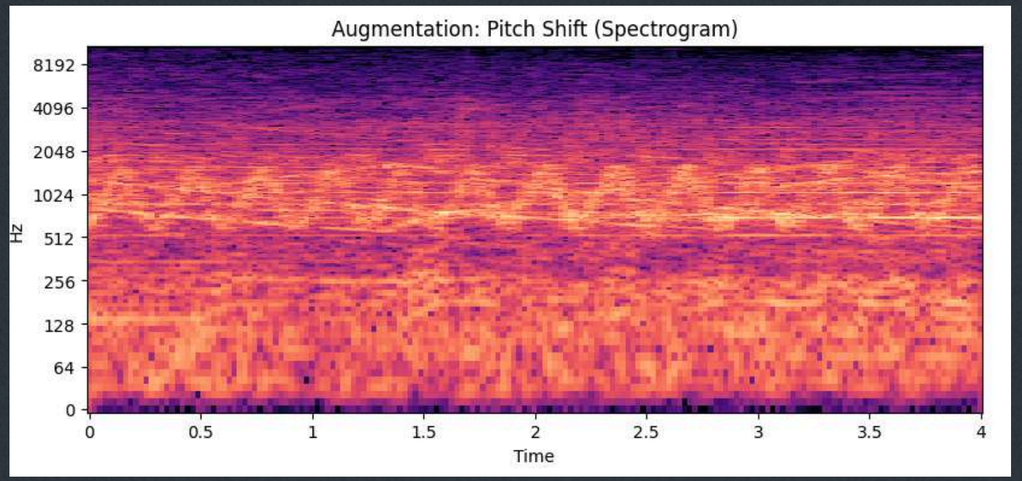
Data Augmentation

Mitigação de Escassez: O dataset UrbanSound8K é pequeno para Deep Learning (~8k amostras).

- Implementámos uma triplicação artificial do dataset para combater o risco de overfitting.

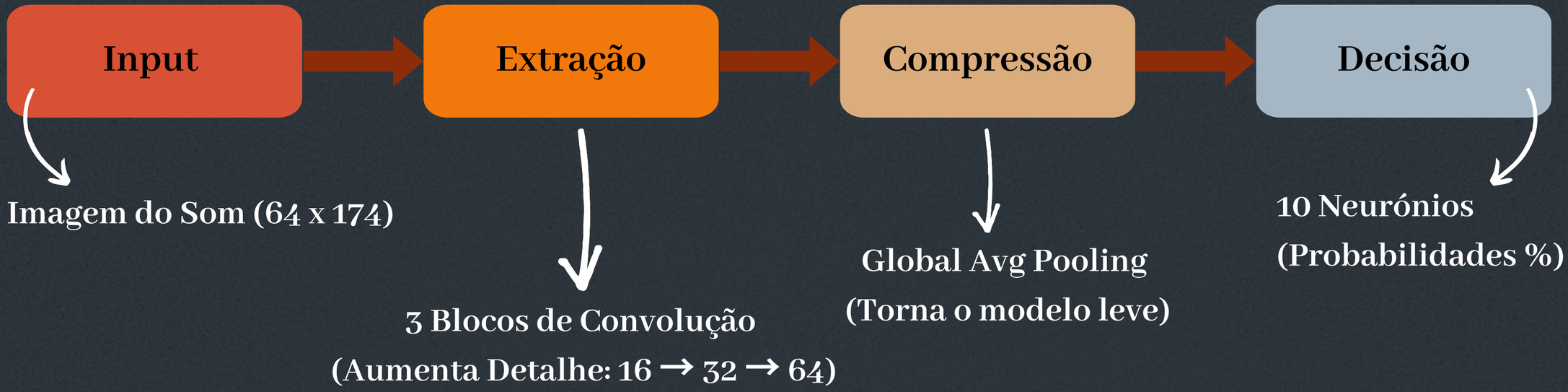
Invariância Tonal: Utilizámos Pitch Shifting (+2 e -2 semitons) para criar variantes de cada som.
Isto força a rede a reconhecer a identidade do som (ex: motor a trabalhar) independentemente da sua frequência base.

Integridade dos Folds: Garantimos rigorosamente que as amostras aumentadas permanecem no mesmo Fold do ficheiro original para evitar Data Leakage.



Arquitetura da Rede

Regularized CNN Architecture



Regularização e Otimização

Combate ao Overfitting

Global Average Pooling (GAP): Reduz parâmetros e evita overfitting.

Dropout Progressivo: Desliga neurónios aleatoriamente para robustez.

Estabilização: Estabiliza o treino e permite maior Learning Rate.

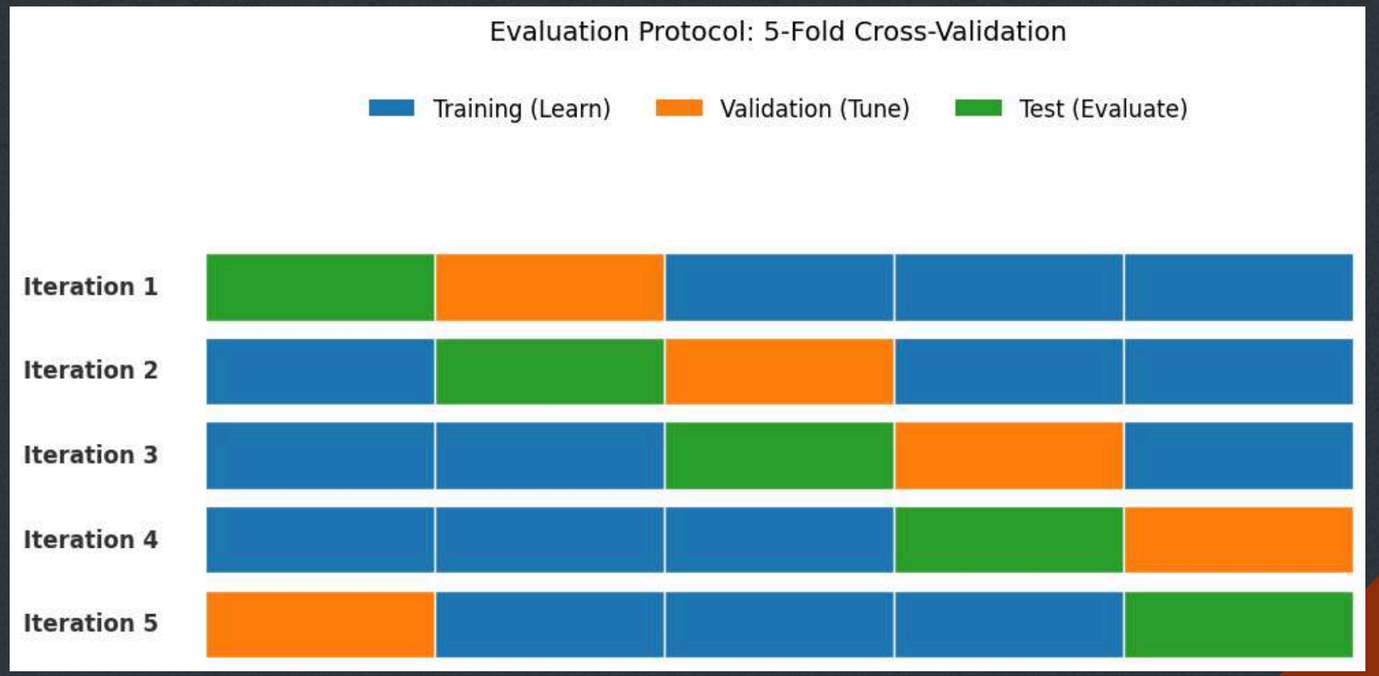
Parâmetro	Valor / Configuração
Input Shape	(64, 174, 1)
Otimizador	Adam (lr=0.001)
Batch Size	64
Dropout	0.25 - 0.4 (Progressivo)
Total Parâmetros	~28,000 (Muito Leve)
Validação	5-Fold Cross-Validation

Protocolo de Avaliação

5-Fold Cross-Validation

- **Justificação:**
  - K=5: Equilíbrio entre Robustez e Rapidez.
  - Garantia: Todas as amostras são testadas.

- **Callbacks (O "Travão de Mão"):**
  - Early Stopping: Pára se não melhorar (5 épocas).
  - ReduceLROnPlateau: Ajuste fino se estagnar.

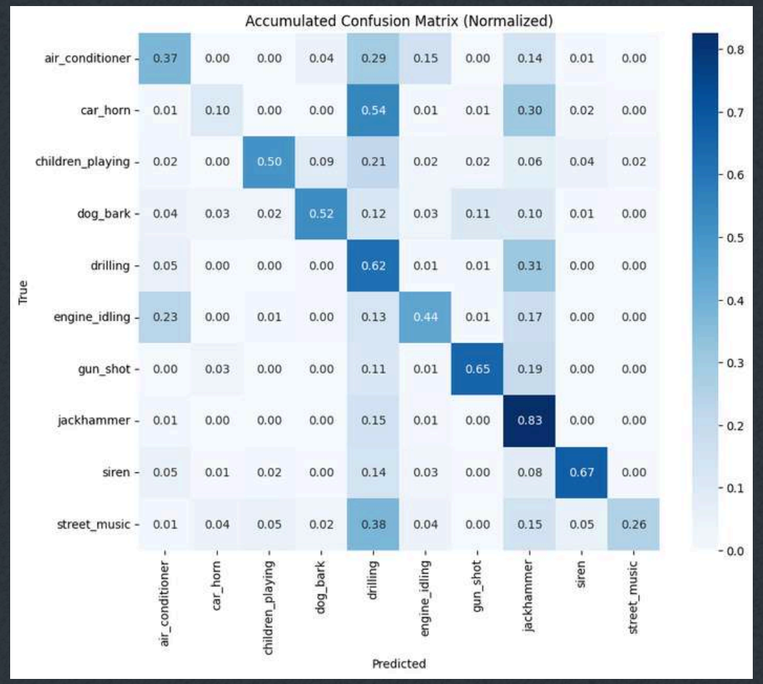


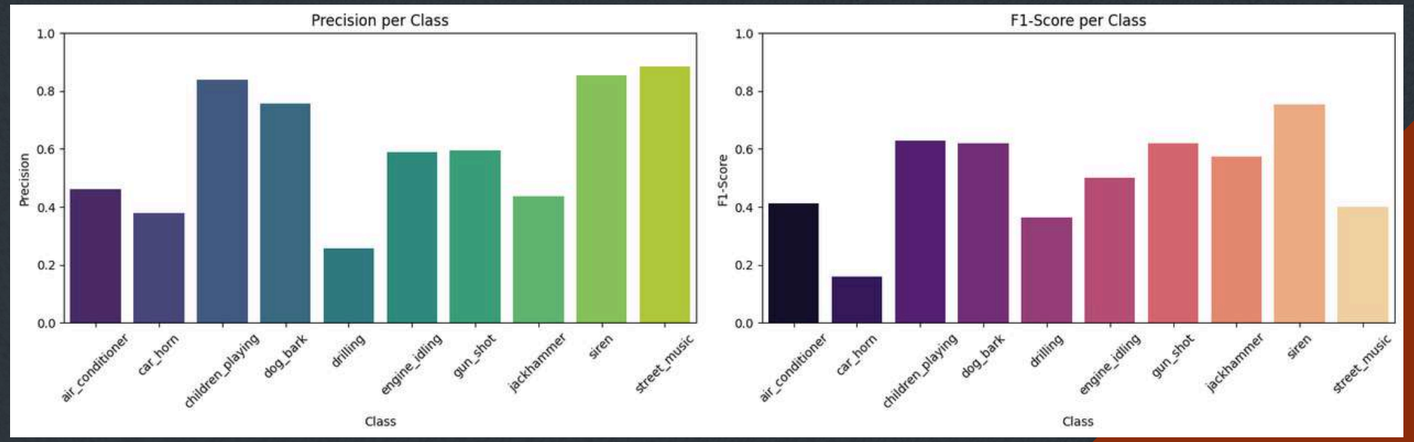
Análise de Resultados

- **Performance Global:** Acurácia média de 51.34% (± 3.6%).
  - Supera o baseline aleatório (10%) em 5x, validando a extração de features via Espectrogramas.

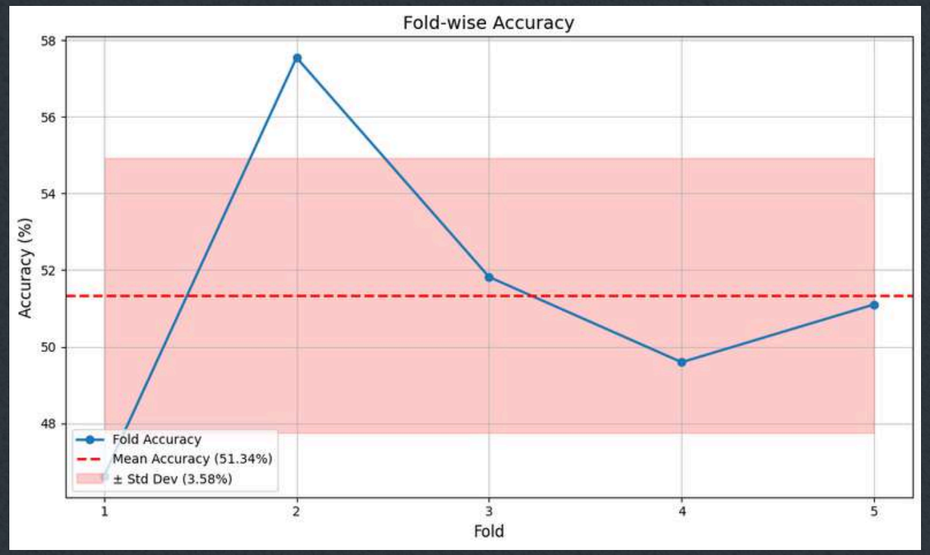
- **Estabilidade:** O desvio padrão baixo (± 3.6%) entre os 5 Folds comprova que a arquitetura é robusta e não depende de uma divisão de dados "sortuda".

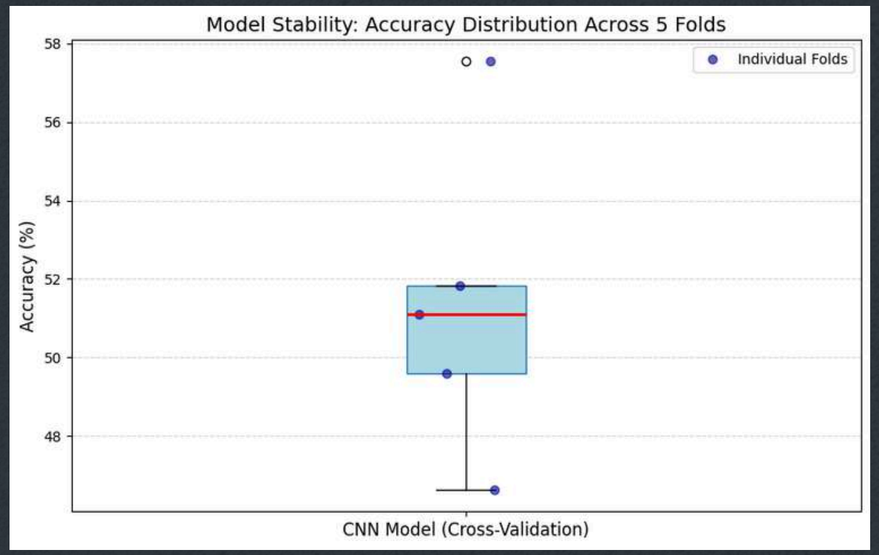
- **Análise de Erro:**
  - Melhores Classes: Gunshot e Siren (Assinaturas espectrais distintas).
  - Dificuldades: Air Conditioner vs. Engine Idling (Ruído contínuo de baixa frequência).

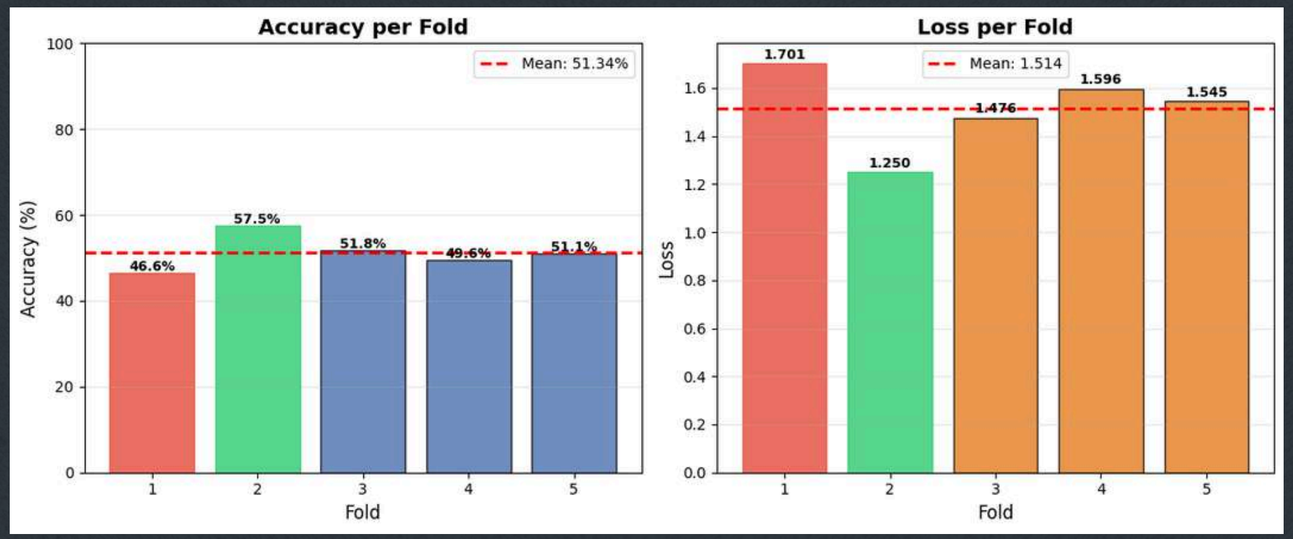




Análise de Resultados







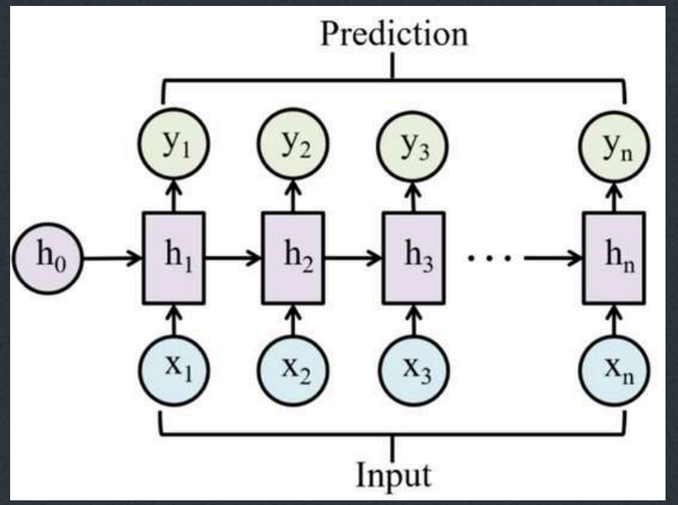
04

Modelo RNN

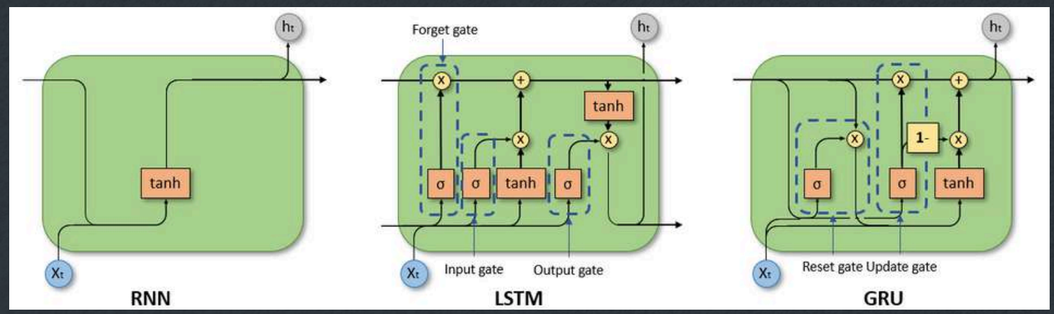
RNN - Recurrent Neural Networks

O que é?

- Redes neuronais projetadas para trabalhar com dados sequenciais
- Possuem memória interna que permite usar as informações das etapas anteriores para interpretar a etapa atual



- Variantes mais avançadas para aprimorar capacidade das RNN simples
  - LSTM
  - GRU



RNN - Recurrent Neural Networks

Pré-Processamento

- **Features:** Coeficientes cepstrais de frequência Mel (MFCC)
- **Windows:** 0.25 segundos
- **Hop size:** 0.125 segundos

- Cada audio é dividido em **33 segmentos**

	fold	classID	audio_name	mfcc_0	mfcc_1	mfcc_2	mfcc_3	mfcc_4	mfcc_5	mfcc_6	...	mfcc_50	mfcc_51	mfcc_52	mfcc_53	mfcc_54	mfcc_55	mfcc_56	mfcc_57	mfcc_58	mfcc_59
0	1	3	101415-3-0-2.wav	-453.56300	117.05657	13.528964	-14.669173	-0.444877	7.370026	-13.528390	...	-2.961487	-1.577231	-2.605381	-1.561487	-1.790304	-2.338786	-1.914598	-1.112039	-1.154101	-0.758382
1	1	3	101415-3-0-2.wav	-331.42860	165.98656	-9.015431	-39.910366	-18.887604	-5.412030	-34.820940	...	-6.170475	-2.565945	-2.863710	-1.049180	-0.990644	0.491482	-4.392341	-1.046770	-0.865129	-1.788776
2	1	3	101415-3-0-2.wav	-427.55652	144.75357	10.944602	-7.468174	-6.224545	-5.990418	-13.361083	...	-7.164290	-2.828632	-0.170411	-1.148738	-1.493358	3.177762	-2.938912	-1.548875	-0.805051	-1.845005
3	1	3	101415-3-0-2.wav	-602.04614	147.65944	10.409507	8.791451	3.895717	5.907481	4.938591	...	-8.777754	-4.251200	-1.152711	0.130814	-0.835157	3.964458	0.432533	-1.403586	-3.892276	-3.242209
4	1	3	101415-3-0-2.wav	-666.96550	138.00640	28.807978	9.505721	11.236222	10.922768	8.751998	...	-7.666019	-3.864321	-1.856921	0.440216	0.064126	1.518500	-1.048186	-0.580486	-5.227286	-4.355827

RNN - Recurrent Neural Networks

Arquitetura do Modelo

Baseada no estudo (Phan et al., 2017)

Modelo Auxiliar → ***Embeddings*** → Probabilidade de pertencer a uma **metaclasse**

MLP simples

GRU Layer Bidirecional
Regularização (Dropout)
Loss function
EarlyStopping ReduceLROnPlateu

Baseada na taxonomia do UrbanSound8K

RNN - Recurrent Neural Networks

Arquitetura do Modelo

Baseada no estudo (Phan et al., 2017)

Modelo Auxiliar

LTE - Label Tree Embedding

```python
def build_LTE_model(input_dim, embedding_dim, num_meta_classes):
    tf.keras.backend.clear_session()
    inputs = Input(shape=(input_dim,), name='Input_layer')
    embeddings = Dense(embedding_dim, activation='relu', name='Embedding_layer')(inputs)
    outputs = Dense(num_meta_classes, activation='softmax', name='Output_layer')(embeddings)
    model = Model(inputs=inputs, outputs=outputs, name="LTE_Model")
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

RNN - Recurrent Neural Networks

Arquitetura do Modelo

Baseada no estudo (Phan et al., 2017)

Modelo

```python
def create_model(input_dim, hidden_dim, output_dim, num_layers, dropout_rate = 0.5):
    tf.keras.backend.clear_session()
    inputs = tf.keras.Input(shape = (None, input_dim))  # Define the input shape
    x = inputs
    for i in range(num_layers):
        x = tf.keras.layers.Bidirectional(
            tf.keras.layers.GRU(
                hidden_dim,
                return_sequences = (i < num_layers - 1),
                kernel_regularizer = tf.keras.regularizers.l2(1e-2),
            )
        )(x)
    x = tf.keras.layers.Dropout(dropout_rate)(x)
    outputs = tf.keras.layers.Dense(output_dim ,kernel_regularizer=tf.keras.regularizers.l2(1e-2))(x)
    model = tf.keras.Model(inputs, outputs)
    model.compile(
        optimizer = tf.keras.optimizers.Adam(learning_rate=0.001),
        loss = tf.keras.losses.CategoricalCrossentropy(from_logits=True),
        metrics = ['accuracy'],
    )
    return model
```

RNN - Recurrent Neural Networks

Estratégia de Treino

Baseada no estudo (Phan et al., 2017)

Hiperparâmetro de Tunning

Hyperparameter	Value
input_dim	60
hidden_dim	256
output_dim	10
num_layers	2
dropout_rate	0.3
batch_size	64

Encontrar os melhores hiperparâmetros através de uma Grid Search

NOTA: processo muito demorado

Optámos por usar valores reportados na literatura.

RNN - Recurrent Neural Networks

Teste do Modelo

Baseada no estudo (Phan et al., 2017)

RNN - Recurrent Neural Networks

Teste do Modelo

Baseada no estudo (Phan et al., 2017)

LTE

```
input_dim_lte = 60
embedding_dim = 8
num_meta_classes = 6

train_lte = df_mfcc[df_mfcc['fold'] == 1]
x_train_lte = train_lte.iloc[:,3:]
y_train_lte = train_lte.iloc[:,1]

y_train_lte=get_meta_classes(y_train_lte).astype('int64')

y_train_lte_one_hot = tf.keras.utils.to_categorical(y_train_lte,num_classes = num_meta_classes)
LTE_model = build_LTE_model(input_dim = input_dim_lte, embedding_dim = embedding_dim, num_meta_classes = num_meta_classes)
LTE_model.fit(x_train_lte, y_train_lte_one_hot, epochs = 100, batch_size = 128, validation_split = 0.2,callbacks = [
    EarlyStopping(monitor = 'val_loss',patience = 10,restore_best_weights = True),
    ReduceLROnPlateau(monitor = 'val_loss', factor = 0.5, patience = 5, min_lr = 1e-6)
])
LTE_model.summary()

embedding_model= tf.keras.models.Model(inputs=LTE_model.input, outputs=LTE_model.layers[1].output)
```

RNN - Recurrent Neural Networks

Teste do Modelo

Baseada no estudo (Phan et al., 2017)

Treino RNN

```
input_dim = 60
hidden_dim = 256
output_dim = 10
num_layers = 2
dropout_rate = 0
batch_size = 64
max_epochs = 50

rnn_model = create_model(
    input_dim=input_dim,
    hidden_dim=hidden_dim,
    output_dim=output_dim,
    num_layers=num_layers,
    dropout_rate=dropout_rate
)
```

```
history = rnn_model.fit(
    x_train_rnn,
    y_train_rnn_one_hot,
    batch_size=batch_size,
    epochs=max_epochs,
    validation_data=(x_val_rnn, y_val_rnn_one_hot),
    callbacks=[
        EarlyStopping(
            monitor='val_loss',
            patience=10,
            restore_best_weights=True
        ),
        ReduceLROnPlateau(
            monitor='val_loss',
            factor=0.75,
            patience=5,
            min_lr=1e-6
        )
    ]
)
```

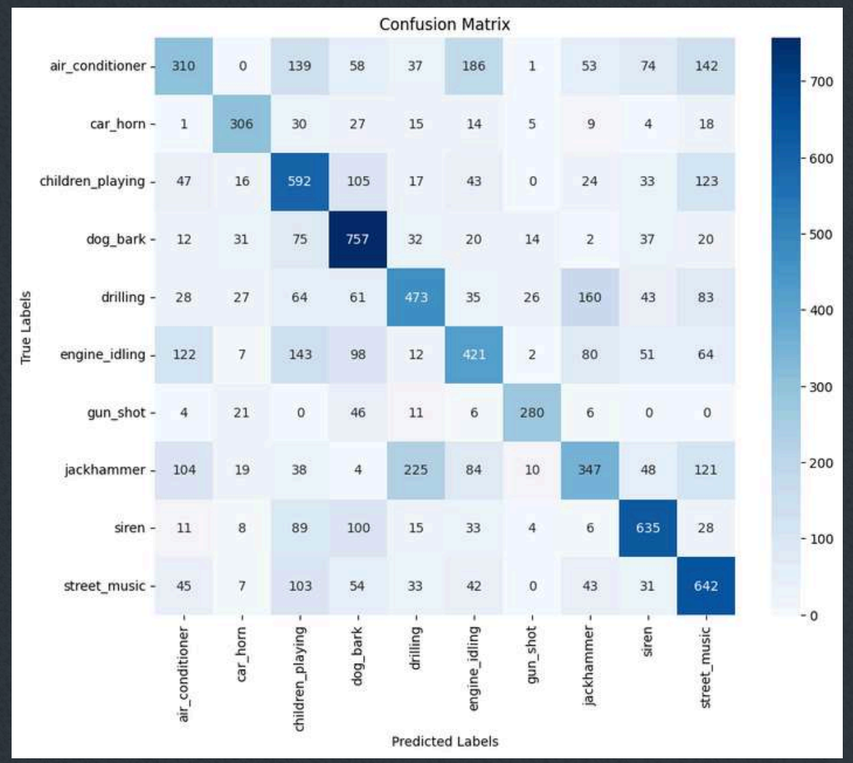
Avaliação para um fold

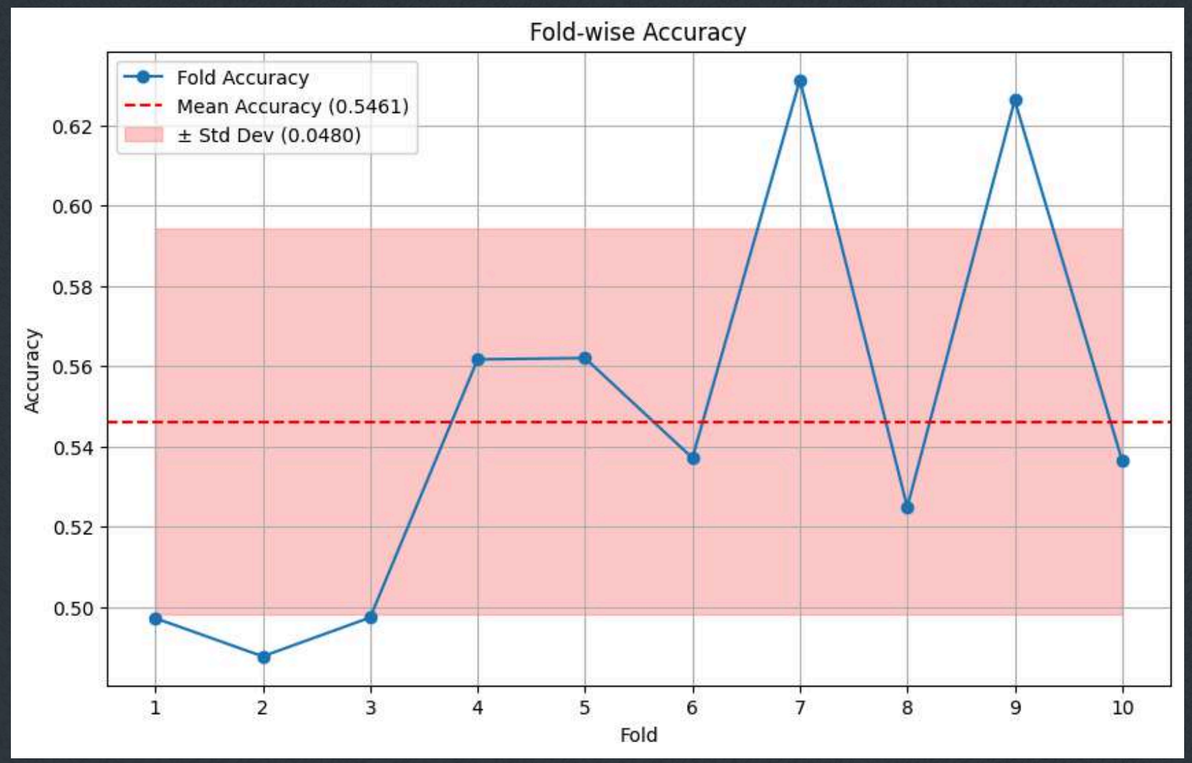
10-Fold Cross-Validation

RNN - Recurrent Neural Networks

Análise dos Resultados

*(ver resultados completos em **results_analysis**)*

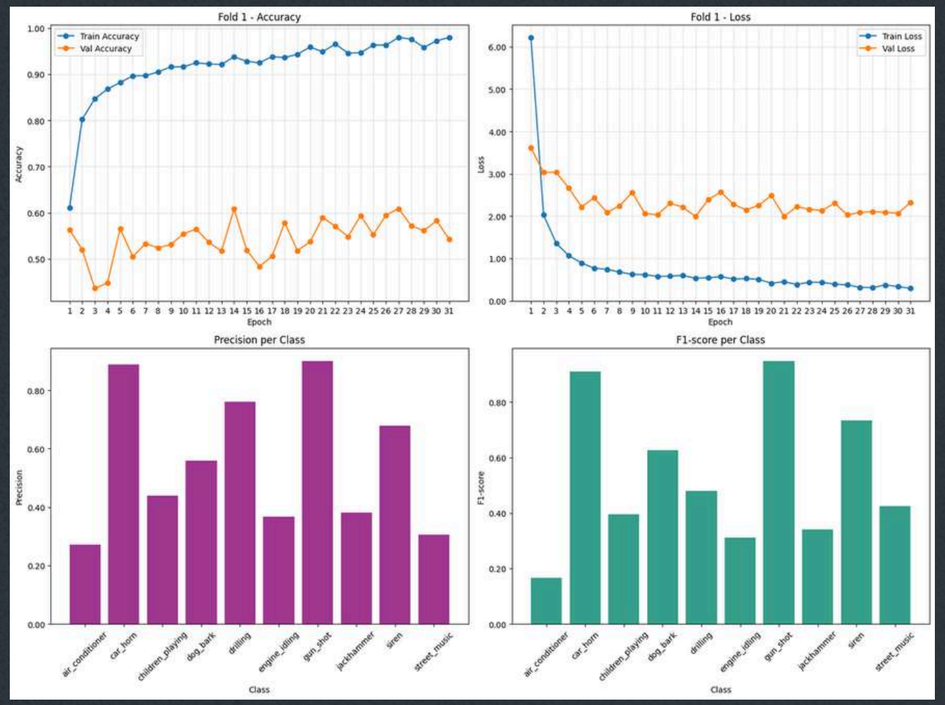




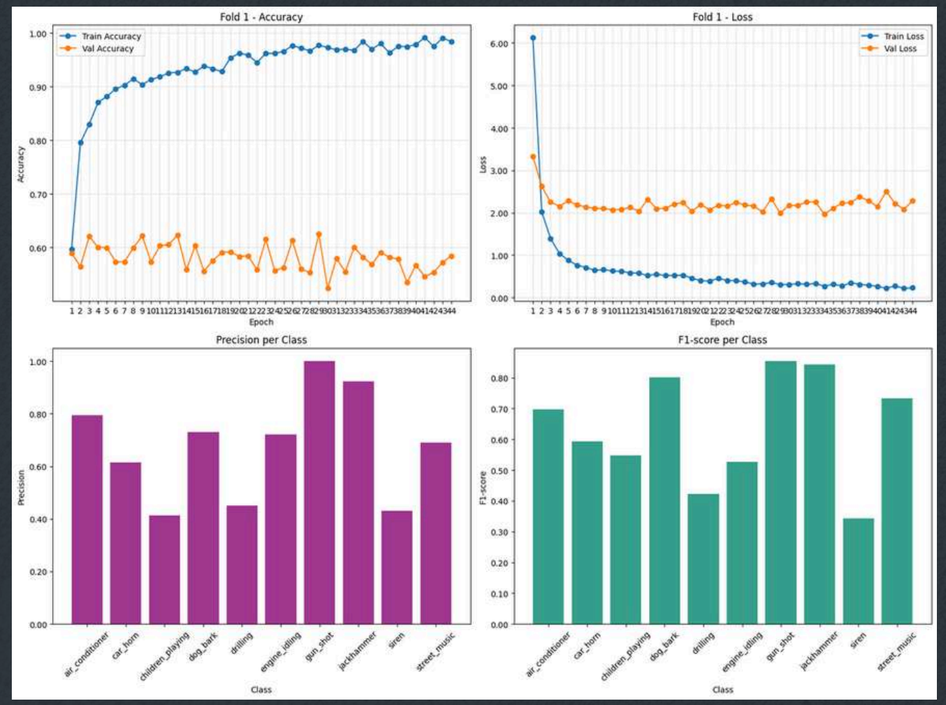
RNN - Recurrent Neural Networks

Análise dos Resultados

*(ver resultados completos em **results_analysis**)*



Fold 3



Fold 7

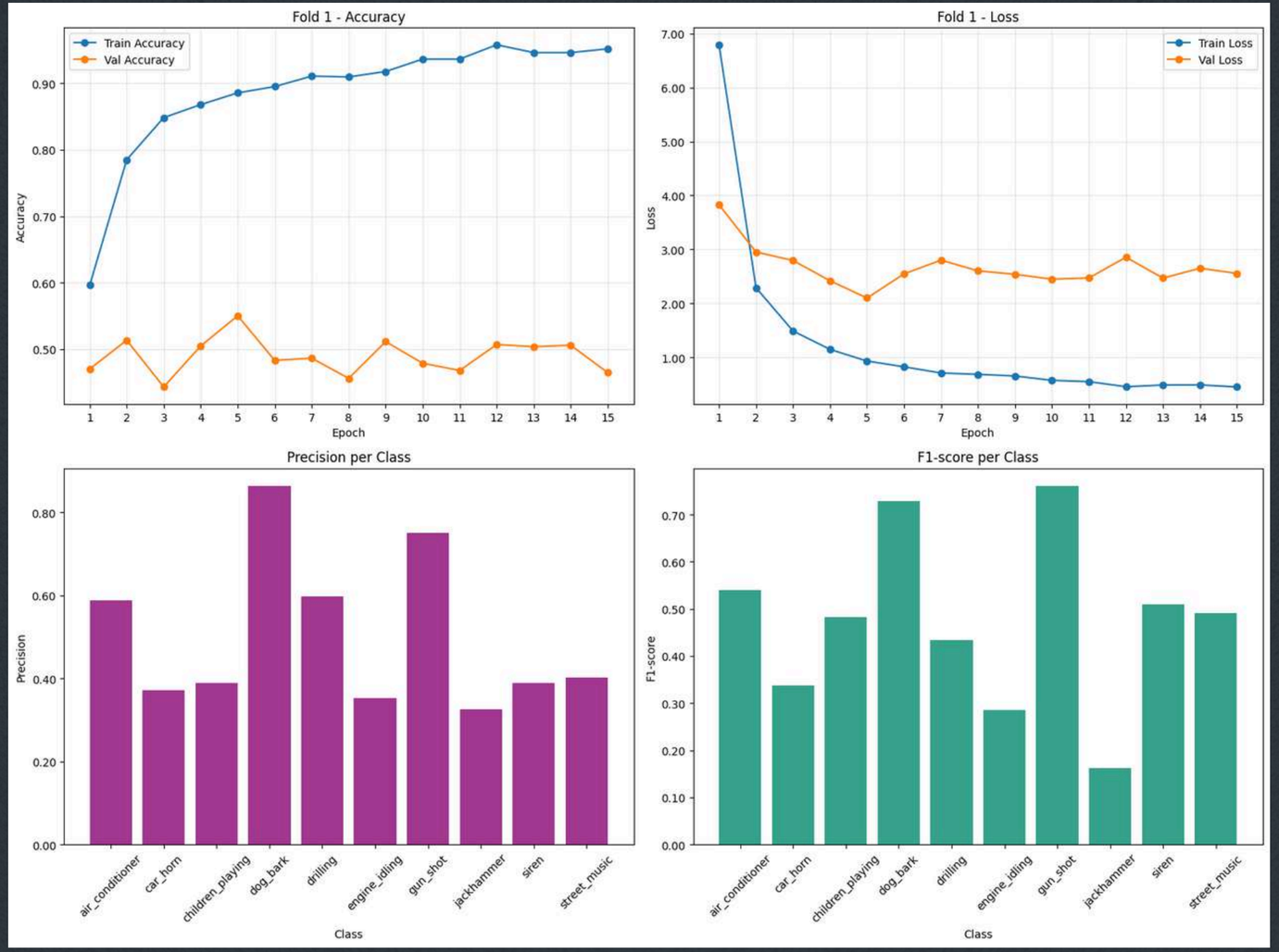
RNN - Recurrent Neural Networks

Melhorias no RNN

Component	Improvement 1	Improvement 2
Base Architecture	2 × Bidirectional GRU	2 × Bidirectional GRU (optimized dropout + L2)
Normalization	Layer Normalization after each GRU	Same Layer Normalization + feature-wise normalization in preprocessing
Attention	Simple attention (Keras Attention)	Multi-Head Attention (4 heads) + residual connection
Pooling	GlobalAveragePooling1D	GlobalAveragePooling1D
Regularization	Final dropout + moderate L2	Optimized dropout, L2 + AdamW weight decay
Data Augmentation	Basic SpecAugment (1 mask per axis, moderate probability)	Strong SpecAugment (multiple masks, time-shift, higher probability)
Mixup	Not included	Included in the training pipeline
Preprocessing	Only reshaping	Feature-wise normalization (per MFCC)
Optimizer	Standard Adam	AdamW with weight decay
Loss Function	Categorical Crossentropy (logits)	Categorical Crossentropy + label smoothing (0.1)
Training Pipeline	Direct NumPy training	`tf.data` with on-the-fly generation (SpecAugment + Mixup)
Main Goal	Improve model representational capacity	Improve robustness, generalization, and training stability
Observed Result	Slight improvement over baseline	Additional, more consistent improvement over Improvement 1

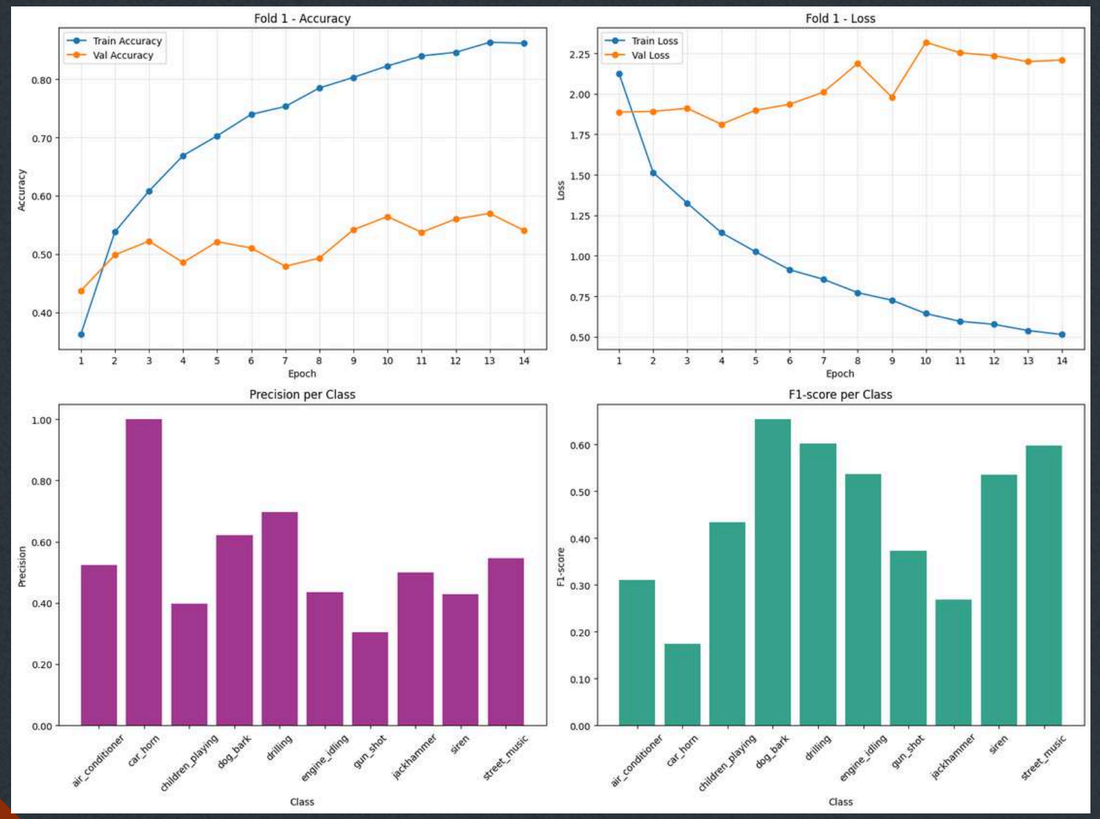
RNN - Recurrent Neural Networks

Modelo Base



RNN - Recurrent Neural Networks

Melhoria 1



Melhoria 2

05

DeepFool

DeepFool

Definição

O DeepFool é um algoritmo que encontra a menor perturbação necessária nos dados de entrada do modelo para enganar a classificação.
Pequenas alterações, muitas vezes impercetíveis no áudio, podem levar a uma classificação incorreta.

Interpretação

A robustez do modelo é avaliada através da magnitude da perturbação:

- Valor baixo → Modelo vulnerável
- Valor alto → Modelo mais robusto

Resultados

Obtivemos os seguintes valores de robustez:

- **RNN:** 0.0096365

O modelo pode ser enganado com menos de 1% de alteração das características MFCC.

- **CNN:** 0.017435

Embora um valor também baixo, é um valor superior ao RNN, o que indica uma melhor robustez.

07

Conclusão e Trabalho Futuro

Conclusão

- O tratamento rigoroso dos dados permitiu desenvolver modelos de classificação de sons urbanos com desempenho consistente: CNN e RNN.
- Os modelos apresentaram resultados razoáveis, captando padrões temporais e espectrais relevantes.
- Contudo, o DeepFool revelou baixa robustez: pequenas perturbações são suficientes para enganar o modelo.
- Concluímos que elevada accuracy não garante segurança nem fiabilidade do sistema.

Trabalho Futuro

- Aumentar a robustez através do treino adversarial, mais dados, otimização de hiperparâmetros e arquiteturas mais avançadas (ex: modelos híbridos CNN-RNN).

08

Referências

Referências

- A Review of DeepFool: a simple and accurate method to fool deep neural networks | by Adrian Morgan | Machine Intelligence and Deep Learning | Medium. (n.d.). Retrieved November 30, 2025, from https://medium.com/machine-intelligence-and-deep-learning-lab/a-review-of-deepfool-a-simple-and-accurate-method-to-fool-deep-neural-networks-b016fba9e48e
- Abdoli, S., Cardinal, P., & Lameiras Koerich, A. (2019). End-to-end environmental sound classification using a 1D convolutional neural network. Expert Systems with Applications, 136, 252–263. https://doi.org/10.1016/J.ESWA.2019.06.040
- Barua, S., Akter, T., Musa, M. A. S., & Azim, M. A. (2023). A Deep Learning Approach for Urban Sound Classification. International Journal of Computer Applications, 185(24), 8–14. https://doi.org/10.5120/ijca2023922991
- Massoudi, M., Verma, S., & Jain, R. (2021). Urban Sound Classification using CNN. Proceedings of the 6th International Conference on Inventive Computation Technologies, ICICT 2021, 583–589. https://doi.org/10.1109/ICICT50816.2021.9358621
- Mienye, I. D., Swart, T. G., Obaido, G., Mienye, I. D., Swart, T. G., & Obaido, G. (2024). Recurrent Neural Networks: A Comprehensive Review of Architectures, Variants, and Applications. Information 2024, Vol. 15, 15(9). https://doi.org/10.3390/INFO15090517
- Phan, H., Koch, P., Katzberg, F., Maass, M., Mazur, R., & Mertins, A. (n.d.). Audio Scene Classification with Deep Recurrent Neural Networks.
- Piczak, K. J. (2015). ENVIRONMENTAL SOUND CLASSIFICATION WITH CONVOLUTIONAL NEURAL NETWORKS. https://doi.org/10.5281/zenodo.12714
- Salamon, J., Jacoby, C., & Bello, J. P. (2014). A dataset and taxonomy for urban sound research. MM 2014 - Proceedings of the 2014 ACM Conference on Multimedia, 1041–1044. https://doi.org/10.1145/2647868.2655045
- Tyagi, S., Aggarwal, K., Kumar, D., & Garg, S. (n.d.). Urban Sound Classification for Audio Analysis Using Long Short-Term Memory.
- Zhao, H., Ye, Y., Shen, X., & Liu, L. (2024). 1D-CNN-based audio tampering detection using ENF signals. Scientific Reports, 14(1). https://doi.org/10.1038/s41598-024-60813-0

Trabalho disponível em:

GitHub

SCAN ME

laragoncalves1375/**Urban-Sound-Classification-...**

2 Contributors | 0 Issues | 0 Stars | 0 Forks

laragoncalves1375/Urban-Sound-Classification-Deep-Learning-Approaches-for-Audio-Recognition

Contribute to laragoncalves1375/Urban-Sound-Classification-Deep-Learning-Approaches-for-Audio-Recognition development by creating an account on GitHub.

GitHub