Classificação de sons urbanos utilizando CNN

Túlio Barroso Volpato
INATEL
Santa Rita do Sapucai
Brasil
tulio.volpato9@gmail.com

***Abstract*— O monitoramento acústico é um componente vital para a segurança e gestão de Cidades Inteligentes. Este trabalho apresenta o desenvolvimento de um classificador de áudio baseado em Redes Neurais Convolucionais (CNN) utilizando o dataset UrbanSound8K. A abordagem converte sinais de áudio bruto em representações visuais (Log-Mel Spectrograms) para extração de características. O pipeline desenvolvido inclui pré-processamento padronizado, extração de espectrogramas log-Mel, técnicas de aumento de dados, regularização e organização estratificada dos folds. A acurácia evoluiu de 11,9% no modelo baseline para aproximadamente 63% no modelo final, evidenciando a eficiência das melhorias aplicadas.**

***Keywords*— classificação de áudio, cidades inteligentes, deep learning, CNN, log-Mel spectrogram.**

Introdução

O desenvolvimento de cidades inteligentes (Smart Cities) depende da capacidade de monitorar o ambiente urbano em tempo real para otimizar serviços e garantir a segurança pública. Embora o uso de câmeras de vídeo seja amplamente difundido, o monitoramento acústico apresenta-se como uma ferramenta complementar poderosa. Diferente das câmeras, sensores sonoros operam eficazmente em condições de baixa visibilidade e detectam eventos fora do campo de visão direto, como disparos de armas, gritos ou falhas mecânicas, permitindo uma resposta rápida a situações críticas.

A Classificação de Sons Ambientais (*ESC -Environmental Sound Classification*) impõe desafios técnicos distintos do reconhecimento de fala ou música. O ambiente acústico urbano é caracterizado por sons não estacionários (que mudam bruscamente), alta presença de ruído de fundo e sobreposição de fontes sonoras. Métodos tradicionais de processamento de sinais frequentemente falham ao tentar capturar essa complexidade, especialmente quando a relação sinal-ruído é baixa ou quando há variabilidade nos dispositivos de gravação.

Apesar dos avanços recentes com o uso de Aprendizado Profundo (Deep Learning), uma lacuna científica persiste na aplicação prática dessas tecnologias. A maioria das soluções de estado da arte baseia-se em modelos massivos e pré-treinados (como VGGish [1] ou YamNet [2]), que exigem alto poder computacional. Essas arquiteturas são inadequadas para cenários de computação de borda (*edge computing*), onde sensores *IoT* (Internet das Coisas) possuem restrições severas de bateria, memória e processamento. Além disso, poucos trabalhos detalham o impacto de cada etapa do pipeline de processamento, da normalização ao aumento de dados, na construção de modelos leves e robustos para datasets heterogêneos como o UrbanSound8K.

Diante desse cenário, o objetivo principal deste trabalho é desenvolver e validar um sistema de classificação de sons urbanos eficientes, utilizando Redes Neurais Convolucionais. Os objetivos específicos incluem:

- Projetar um pipeline de processamento que transforme áudio bruto em espectrogramas Log-Mel [3], capturando padrões de tempo e frequência robustos;
- Analisar o impacto de técnicas de pré-processamento, aumento de dados (*data augmentation*)[4] e ponderação de classes na generalização do modelo;
- Propor uma arquitetura de CNN leve, viável para futuros sistemas embarcados;
- Quantificar os ganhos de desempenho comparando um modelo baseline com a versão otimizada.

As principais contribuições deste artigo são:

- A implementação de um pipeline de classificação leve, documentado e reprodutível;
- Uma avaliação experimental detalhada que demonstra a evolução da acurácia de 11,9% (modelo inicial) para aproximadamente 63% (modelo final) através de melhorias incrementais;
- A proposta de uma arquitetura eficiente capaz de lidar com a variabilidade acústica urbana sem a necessidade de hardware de alto desempenho.

Com o avanço do Aprendizado Profundo, arquiteturas capazes de aprender diretamente de representações espectrais tornaram-se o padrão. Modelos robustos e pré-treinados em grandes bases de dados (como o AudioSet)[5] dominam o estado da arte atual. O VGGish, uma adaptação da rede VGG [6] para áudio, e o YamNet, baseado na eficiente MobileNet [7], são exemplos amplamente utilizados que alcançam acurácias elevadas (frequentemente acima de 75%).

Outra vertente de sucesso são as PANNs (*Pretrained Audio Neural Networks*) [8], que utilizam arquiteturas profundas para atingir o topo dos benchmarks. Além das CNNs puras, modelos híbridos como as CRNNs (*Convolutional Recurrent Neural Networks*) combinam convoluções com camadas de memória (LSTM ou GRU)[9] para capturar a evolução temporal do som.

Apesar do alto desempenho, modelos modernos apresentam uma desvantagem crítica para o contexto de Cidades Inteligentes: o custo computacional dessas arquiteturas exige milhões de parâmetros e hardware dedicado (GPUs) para inferência, o que é inviável para sensores de baixo custo e baixo consumo de energia instalados em postes ou edifícios.

A Tabela 1 apresenta um comparativo entre abordagens relevantes na literatura utilizando o dataset UrbanSound8K.

Tabela 1 - Comparação de trabalhos no URBANSOUND8K.

Trabalho / Ref	Método / Arquitetura	Acurácia Típica	Foco Principal
Salamon & Bello	SVM + MFCC	~ 68%	Baseline do Dataset
Piczak	CNN + Log-Mel	~ 73%	CNN Simples
Hershey et al.	CNN Profunda	> 75%	Alta Acurácia / Transfer Learning
Este trabalho	CNN + Augmentation	63%	Eficiência / Edge Computing

Observa-se que, enquanto modelos massivos ultrapassam 80% de acurácia, modelos treinados "do zero" (*from scratch*) em arquiteturas mais simples tendem a oscilar entre 70%, dependendo fortemente das técnicas de regularização.

Muitos trabalhos focam exclusivamente na métrica final de acurácia, omitindo detalhes cruciais sobre o pré-processamento e a variabilidade entre os folds do dataset, o que dificulta a reprodução em cenários reais.

Metodologia

Esta seção descreve o processo de desenvolvimento do classificador de sons urbanos, abrangendo desde a preparação dos dados até a arquitetura da rede neural e as estratégias de treinamento. O pipeline foi projetado para ser reprodutível e computacionalmente eficiente, visando aplicações em sistemas de monitoramento urbano.

A metodologia segue um fluxo sequencial de processamento, o sistema recebe arquivos de áudio brutos, padroniza suas dimensões temporais, converte-os em representações visuais (espectrogramas) e utiliza esses dados para treinar uma CNN. O desenvolvimento ocorreu de forma iterativa, partindo de um modelo simplificado até a versão final otimizada.

O estudo utilizou o dataset UrbanSound8K, composto por 8.732 clipes de áudio classificados em 10 ruídos urbanos conforme a tabela 2. Os dados são distribuídos originalmente em 10 pastas (folds).

Tabela 2 - Classes do UrbanSound8K

ID	Classe (Nome Original)	Tipo de Som Predominante
0	Air Conditioner	Estacionário / Mecânico
1	Car Horn	Impulsivo / Alerta
2	Children Playing	Voz / Não-estacionário
3	Dog Bark	Impulsivo / Animal
4	Drilling	Mecânico / Repetitivo
5	Engine Idling	Estacionário / Baixa Frequência
6	Gun Shot	Impulsivo / Explosivo
7	Jackhammer	Mecânico / Repetitivo
8	Siren	Variável / Alerta
9	Street Music	Complexo / Harmônico

Para garantir uma avaliação rigorosa e evitar o vazamento de dados (*data leakage)*[10], adotou-se a seguinte estratégia de divisão:

- Conjunto de Teste (fold 10): Os dados desta pasta foram isolados e utilizados estritamente para a avaliação final do modelo, simulando dados nunca vistos.
- Conjunto de Treino e Validação (folds 1-9): Os dados das nove pastas restantes foram utilizados para o aprendizado da rede.

O áudio bruto possui variações de duração e taxa de amostragem que impedem a alimentação direta em uma CNN. O pré-processamento padronizou todos os arquivos através dos seguintes passos:

- Reamostragem: Conversão para uma taxa de amostragem fixa de 22.050 Hz e redução para canal mono.
- Duração Fixa: Ajuste de todos os clipes para exatamente 4 segundos. Áudios mais curtos receberam preenchimento com silêncio (*padding*), e áudios mais longos foram cortados, resultando em um vetor de entrada fixo com N = 88.200 amostras.

A extração de características converte o sinal temporal em uma representação tempo-frequência. O processo matemático adotado consiste em quatro etapas:

- Transformada de Fourier de Curto Prazo (STFT): O sinal y[n] é segmentado em janelas temporais e transformado para o domínio da frequência.
- Escala Mel: O espectro de potência é filtrado por um banco de 64 filtros triangulares que imitam a percepção auditiva humana (mais sensível a baixas frequências).
- Escala Logarítmica: Aplica-se o logaritmo para converter a energia em Decibéis (dB), gerando o Log-Mel Spectrogram, uma vez que a percepção de volume humano é logarítmica.
- Normalização Min-Max: Para facilitar a convergência da rede neural, os valores do espectrograma foram normalizados para o intervalo [0, 1].

O tensor resultante possui dimensão (64, 345, 1), tratado pela rede como uma imagem de canal único.

Para mitigar o overfitting causado pela escassez de dados em algumas classes, foram aplicadas técnicas de Data Augmentation de forma dinâmica (*on-the-fly*) durante o carregamento dos dados de treino:

- Injeção de Ruído: Adição de ruído branco aleatório ao sinal, forçando a rede a ignorar o fundo e focar no padrão sonoro principal.
- Ganho Aleatório: Variação da amplitude do sinal entre 80% e 120%, simulando diferentes distâncias da fonte sonora.

A arquitetura proposta prioriza a eficiência computacional. Trata-se de uma CNN com três blocos de convolução sequenciais, onde cada bloco aplica extração de características, normalização e redução de dimensionalidade conforme mostrado na tabela 3.

Tabela 3 - Arquitetura da CNN proposta.

Camada	Tipo	Configuração / Filtros	Saída (Shape)
Entrada	Input	Imagem 64x345x1	(64, 345, 1)
Bloco 1	Conv2D	32 filtros, kernel 3x3, L2 reg.	(64, 345, 32)
	BatchNormalization	-	-
	ReLU + MaxPool	Pool size (2,2)	(32, 172, 32)
Bloco 2	Conv2D	64 filtros, kernel 3x3, L2 reg.	(32, 172, 64)
	BatchNormalization	-	-
	ReLU + MaxPool	Pool size (2,2)	(16, 86, 64)
Bloco 3	Conv2D	128 filtros, kernel 3x3, L2 reg.	(16, 86, 128)
	BatchNormalization	-	-
	ReLU + MaxPool	Pool size (2,2)	(8, 43, 128)
Classificador	GlobalAvgPool	-	(128)
	Dense	128 neurônios, ReLU	(128)
	Dropout	Taxa 0.5	(128)
Saída	Dense (Softmax)	10 neurônios (Classes)	(10)

O desenvolvimento seguiu uma abordagem comparativa entre duas versões:

- Modelo Baseline: Versão inicial treinada sem normalização, sem data augmentation e sem pesos de classe. Utilizou-se uma arquitetura simples e treinamento por épocas fixas. Este modelo serviu como referência base.
- Modelo Final: Incorporou o pipeline completo descrito acima. O treinamento utilizou o otimizador Adam e a função de perda Sparse Categorical Crossentropy. Para lidar com o desbalanceamento do dataset, foram calculados Pesos de Classe (*Class Weights*), penalizando mais severamente os erros nas classes minoritárias.

Foram utilizados três callbacks para controle do treinamento:

- ModelCheckpoint para salvar o melhor modelo;
- ReduceLROnPlateau para ajuste fino da taxa de aprendizado;
- EarlyStopping para interromper o treinamento caso a perda de validação não melhore após 10 épocas.

Experimentos e Resultados

Para assegurar a reprodutibilidade da divisão estratificada dos dados e da inicialização dos pesos, fixou-se a semente aleatória (*random seed*) em 42. O modelo foi treinado com um tamanho de lote (*batch size*) de 32 amostras, utilizando o otimizador Adam com taxa de aprendizado inicial (*learning rate*) de 1* 10-3. O ciclo de treinamento foi configurado para um máximo de 60 épocas, controlado por callbacks de interrupção antecipada (*Early Stopping*) com paciência de 10 épocas, evitando o desperdício de recursos computacionais após a convergência do modelo.

Os experimentos foram conduzidos em ambiente Python, utilizando a biblioteca TensorFlow/Keras. O treinamento foi realizado em CPU, o que impôs limitações no tamanho da rede e no volume de dados processados por época.

O projeto estabeleceu inicialmente uma meta ambiciosa de atingir 75% de acurácia, alinhada com o estado da arte de modelos massivos. O desenvolvimento ocorreu de forma iterativa para perseguir este objetivo dentro das restrições de hardware.

A Tabela 4 apresenta a comparação entre o ponto de partida e o modelo final.

Tabela 4 - Evolução de desempenho.

Modelo	Características Principais	Acurácia	Macro F1
Baseline	CNN rasa, s/ norm., s/ aug., s/ pesos	11,95%	0,12
Versão Final	CNN profunda, Log-Mel norm., *Data Aug.*, Pesos de Classe	63,08%	0,63

O salto de desempenho de ~12% para ~63% demonstra a eficácia crítica do pipeline de pré-processamento. Embora o resultado final tenha ficado abaixo da meta estabelecida de 75%, ele representa um sucesso técnico significativo para uma arquitetura leve treinada do zero, superando amplamente o desempenho aleatório e validando a metodologia proposta.

As curvas de aprendizado apresentadas na Fig. 1 ilustram a estabilidade do modelo final. Observa-se que a perda de treinamento (*training loss*) decresce suavemente, enquanto a perda de validação estabiliza, indicando que as técnicas de regularização (Dropout de 0.5 e regularização L2) foram eficazes em controlar o overfitting. A convergência da acurácia de treino e validação para valores próximos a 90% durante o treinamento contrasta com os 63% no teste, evidenciando a disparidade acústica natural entre os folds do UrbanSound8K, onde o fold 10 apresenta características distintas dos demais.

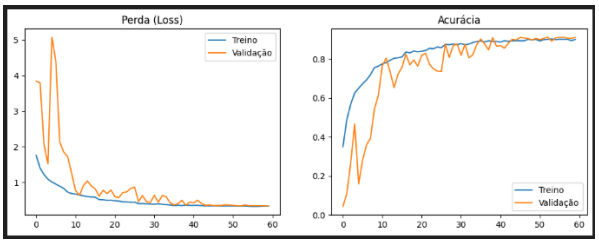
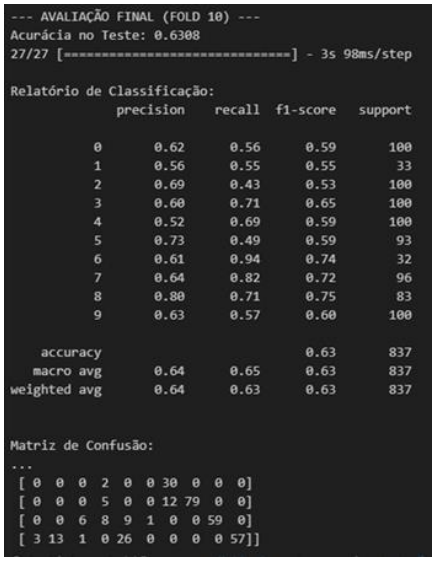


Figura 1 – Curva de aprendizado.

A análise detalhada por classe, visualizada na Matriz de Confusão demonstrada na figura 2 revela comportamentos distintos do classificador:

- Acertos Notáveis: O modelo obteve excelente desempenho na classe "Tiro de Arma de Fogo" (Gunshot, Classe 6), com *recall* de 94 %. Este som possui características impulsivas e transitórias muito distintas no espectrograma, facilitando a extração de features pela CNN.
- Confusões Esperadas: As maiores taxas de erro ocorreram entre classes com assinaturas espectrais mecânicas e contínuas semelhantes. Houve confusão mútua entre "Ar Condicionado" (Air Conditioner, Classe 0) e "Motor em Marcha Lenta" (Engine ldling, Classe 5). Ambos são sons estacionários de baixa frequência, difíceis de distinguir sem uma resolução temporal mais fina ou contexto adicional.



```
--- AVALIAÇÃO FINAL (FOLD 10) ---
Acurácia no Teste: 0.6308
27/27 [==============================] - 3s 98ms/step

Relatório de Classificação:
              precision    recall  f1-score   support

           0       0.62      0.56      0.59       100
           1       0.56      0.55      0.55        33
           2       0.69      0.43      0.53       100
           3       0.60      0.71      0.65       100
           4       0.52      0.69      0.59       100
           5       0.73      0.49      0.59        93
           6       0.61      0.94      0.74        32
           7       0.64      0.82      0.72        96
           8       0.80      0.71      0.75        83
           9       0.63      0.57      0.60       100

    accuracy                           0.63       837
   macro avg       0.64      0.65      0.63       837
weighted avg       0.64      0.63      0.63       837


Matriz de Confusão:
...
 [ 0  0  0  2  0  0 30  0  0  0]
 [ 0  0  0  5  0  0 12 79  0  0]
 [ 0  0  6  8  9  1  0  0 59  0]
 [ 3 13  1  0 26  0  0  0  0 57]]
```

Figura 2 - Matriz de confusão.

Para uma avaliação mais rigorosa da robustez do classificador, considerou-se, além da acurácia global, o F1-Score (*Macro-Average*). Esta métrica é crítica em datasets com desbalanceamento, pois atribui o mesmo peso a todas as classes, independentemente do número de amostras disponíveis. O modelo final atingiu um Macro-F1 de 0,63, um valor extremamente próximo à acurácia, o que demonstra estatisticamente que a rede neural aprendeu a generalizar de forma parcialmente equilibrada entre as 10 categorias, sem apresentar viés significativo em direção às classes majoritárias.

Conclusão

Este trabalho apresentou o desenvolvimento integral de um sistema de classificação de sons urbanos baseado em Redes Neurais Convolucionais aplicadas a espectrogramas Log-Mel. A abordagem foi projetada para ser computacionalmente eficiente e reprodutível, visando aplicações em sensores de borda para Cidades Inteligentes.

Os experimentos validaram a eficácia da metodologia proposta, partindo de um modelo baseline com desempenho aleatório (acurácia de 11,95%), a implementação de um pipeline robusto de pré-processamento e Data Augmentation elevou a acurácia para 63,08% no conjunto de teste. Este resultado confirma que redes leves, quando alimentadas com dados tratados adequadamente, são capazes de aprender padrões complexos em áudios não estacionários e ruidosos.

As principais contribuições deste estudo incluem:

A implementação de um pipeline transparente e documentado, abrangendo desde a normalização do sinal até o treinamento da CNN;

A proposta de uma arquitetura de rede leve, otimizada para ambientes com restrição de hardware (treinamento em CPU);

A demonstração quantitativa do impacto de técnicas de regularização (pesos de classe e aumento de dados) na mitigação do overfitting em datasets desbalanceados;

A validação da viabilidade técnica do monitoramento acústico automático como ferramenta complementar à vigilância visual em ambientes urbanos.

Ao contrário de modelos baseados em Transformers ou CNNs profundas que exigem GPUs dedicadas, a arquitetura desenvolvida pode ser embarcada em:

Microcontroladores e SoCs de baixo custo (ex: Raspberry Pi, ESP32-S3);

Sensores acústicos autônomos integrados a postes de iluminação inteligente;

Sistemas de segurança patrimonial com processamento local.

Essa capacidade de processamento local reduz a latência (crítica para detecção de tiros ou acidentes) e diminui a necessidade de largura de banda para envio de áudio bruto para a nuvem, endereçando também preocupações com a privacidade dos cidadãos.

Trabalhos Futuros

Para superar as limitações identificadas e expandir a aplicabilidade do sistema, propõem-se as seguintes direções de pesquisa:

Validação Robusta: Execução de validação cruzada completa (10-fold) e testes em datasets externos para confirmar a generalização do modelo em diferentes domínios acústicos;

Transfer Learning: Investigação de arquiteturas pré-treinadas no estado da arte (como YAMNet, VGGish ou PANNs) para avaliar o compromisso entre ganho de acurácia e custo computacional;

Augmentation Avançado: Implementação de técnicas de aumento de dados no domínio da frequência, como SpecAugment [11] (mascaramento de tempo/frequência) e Pitch Shifting[12];

Inferência em Tempo Real: Desenvolvimento de um módulo de deploy otimizado (via TensorFlow Lite ou quantização) para testes práticos em dispositivos IoT embarcados (ex: Raspberry Pi);

Fusão Multimodal: Integração do classificador de áudio com dados de outros sensores urbanos (câmeras, sensores de poluição) para aumentar a confiabilidade na detecção de eventos críticos.

References

[1] S. Hershey et al., "CNN architectures for large-scale audio classification," in 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, 2017, pp. 131-135.

[2] D. Plakal et al., "YAMNet," TensorFlow Hub, 2020. [Online]. Available: https://tfhub.dev/google/yamnet/1.

[3] S. S. Stevens, J. Volkmann, and E. B. Newman, "A scale for the measurement of the psychological magnitude pitch," Journal of the Acoustical Society of America, vol. 8, no. 3, pp. 185–190, 1937.

[4] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," IEEE Signal Processing Letters, vol. 24, no. 3, pp. 279–283, Mar. 2017.

[5] J. F. Gemmeke et al., "Audio Set: An ontology and human-labeled dataset for audio events," in 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, 2017, pp. 776–780.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in International Conference on Learning Representations (ICLR), San Diego, CA, 2015.

[7] A. G. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, arXiv:1704.04861. [Online]. Available: https://arxiv.org/abs/1704.04861.

[8] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "PANNs: Large-Scale Pretrained Audio Neural Networks for Audio Pattern Recognition," IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 28, pp. 2880–2894, 2020.

[9] Z. Zhang, S. Xu, S. Zhang, T. Qiao, and S. Cao, "Attention based convolutional recurrent neural network for environmental sound classification," Neurocomputing, vol. 407, pp. 396–406, 2020.

[10] S. Kaufman, S. Rosset, C. Perlich, and O. Stitelman, "Leakage in data mining: Formulation, detection, and avoidance," ACM Transactions on Knowledge Discovery from Data (TKDD), vol. 6, no. 4, pp. 1-21, Dec. 2012.

[11] D. S. Park et al., "SpecAugment: A simple data augmentation method for automatic speech recognition," in Proc. Interspeech 2019, Graz, Austria, Sep. 2019, pp. 2613–2617.

[12] J. Laroche and M. Dolson, "Improved phase vocoder time-scale modification of audio," IEEE Transactions on Speech and Audio Processing, vol. 7, no. 3, pp. 323–332, May 1999.

[13] D. Snyder, G. Chen, and D. Povey, "MUSAN: A music, speech, and noise corpus," 2015, arXiv:1510.08484. [Online]. Available: https://arxiv.org/abs/1510.08484.

[14] B. McFee et al., "librosa: Audio and Music Signal Analysis in Python," Proc. 14th Python in Science Conf., 2015, pp. 18–25.