

Machine Learning II

Urban Sounds 8k

Carolina Proença 202306055
Eduarda Neves 202307178
Maria Morais 202304201

MLP

Começámos por implementar um MLP porque este modelo serve como uma baseline simples para o problema. O MLP permite avaliar rapidamente o impacto do pré-processamento e das features extraídas (como MFCCs e outros descritores acústicos), além de exigir menor capacidade computacional e ser fácil de treinar e interpretar. Assim, forneceu-nos um ponto de referência inicial para compreender a dificuldade da tarefa de classificação de sons urbanos.

Pré-processamento

- **Reprodutibilidade:** Fixação de Random Seeds (42) para garantir resultados consistentes.
- **Normalização de Áudio:**
 - Conversão para Mono.
 - Reamostragem para 22.050 Hz.
 - Motivação: Análise espectral revelou 95% da energia abaixo de 12.8kHz. 22kHz cumpre o Teorema Nyquist e reduz o custo computacional em 50%.
 - Normalização de Amplitude $[-1, 1]$.
- **Padronização Temporal:**
 - Duração fixa de 4 segundos.
 - Técnica: Zero-Padding (preserva o ataque do som) no final ou Truncating.

Feature Extraction

- Desafio do MLP: Não processa dados sequenciais brutos.
- Solução: Conversão em Vetores Estatísticos (Média e Desvio Padrão de cada feature)
- **Conjuntos de features:**
 - a. Simples (34 features): 13 MFCCs + Chroma (STFT) + Descritores Espectrais (RMS, ZCR)
 - b. Intermédio (38 features): Simples + Features Rítmicas (Tempogram)
 - c. Completo (106 features): Acrescido com 40 MFCCs + Chroma (CQT, CENS) + Fourier Tempogram (Real/Img).



Arquitetura do Modelo

- **Arquitetura Base:**
 - Input: Vetor de 106 features (Estatísticas 1D).
 - Hidden Layers: 3 Camadas Densas com 256 neurónios cada
 - Ativação: ReLU (Hidden) e Softmax (Output).
 - Otimizador: Adam
- **Variantes Implementadas para comparação:**
 1. Baseline: Apenas Dropout após cada camada oculta.
 2. Com Batch Normalization: Adição de camadas BN antes do Dropout para estabilizar o treino.
 3. Com L2 Regularization: Adição de penalização de pesos nas camadas densas.

Estratégias de Regularização

- **Dropout:** Desativação aleatória de neurónios. Impede que os neurónios dependam excessivamente uns dos outros.
- **L2 Regularization:** Penalização de pesos elevados. Encoraja o modelo a manter os seus parâmetros pequenos e dispersos.
- **Batch Normalization:** Normaliza os outputs da camada anterior.
- **Early Stopping:** Paragem baseada na Validation Loss (Paciência=10). Ao interromper o treino no ponto ótimo, impedimos que o modelo continue a ajustar-se excessivamente aos detalhes irrelevantes.

Treino

- **SMOTE:** A EDA revelou um desequilíbrio entre classes. Gun shot e Car horn estão sub-representadas. O SMOTE cria novos exemplos sintéticos interpolando vetores de características existentes.
- **MinMax Scaler:** Reescala as features para [0,1] para que contribuam equitativamente.

Otimização Experimental

Fase 1: Seleção de Features

- **Resultado:** O conjunto Completo (Acc: 67.34%) superou os conjuntos Simples (62.89%) e Intermédio (65.07%).
- **Decisão:** Fixar o dataset Completo.

Fase 2: Arquitetura

- Baseline vs. Batch Norm vs. L2.
- **Resultado:** Dropout + BN causou instabilidade (Acc: 65.20%). Dropout + L2 foi a melhor combinação (Acc: 68.67%).

Fase 3: Hiperparâmetros

- **Testes de capacidade e Regularização:**
 - Hidden Units: Testámos 128 e 512. Conclusão: nenhuma superou 256. A capacidade atual é a ideal.
 - L2 Lambda: Aumentar para 0.01 causou over_regularization (Acc: 61.97%). Mantivemos 0.001.
 - Dropout: 0.1, 0.2 e 0.5 não se mostraram melhores alternativas. Mantivemos 0.3.
- **Otimização do Treino**
 - Batch Size: Aumento de 32 para 128 aumentou a estabilidade do gradiente (Acc: 69.60%)
 - Learning Rate: Em vez de 0.001 testámos 0.0001. Resultou numa convergência mais lenta e precisa (Acc final: 69.72%)

Resultados finais

- Melhor configuração: L2 Regularization (lambda=0.001) + Dropout(0.3)
- Hiden Units=256, learning rate=0.0001, batch size=128
- Accuracy Média: 69.72% (Std: 4.04%)
- Pontos Fortes:
 - Gun_shot e Car_horn (F1-score > 0.80)
- Pontos Fracos:
 - Air_conditioner (F1-score: 0.52), Jackhammer (F1-score: 0.62)
- Limitações (Matriz de Confusão):
- Confusão entre Air_conditioner e Engine_idling.
- Causa: Perda de informação temporal (médias estatísticas semelhantes)
- Conclusão: O modelo distingue eficazmente sons impulsivos/distintos. Dificuldade em sons contínuos e ruidosos (Air Conditioner vs Engine Idling). O MLP atingiu o limite da representação estatística (sem tempo).

CNN

Optámos depois por implementar uma CNN, pois este tipo de rede é atualmente uma das abordagens mais eficazes para classificação de áudio curto. Ao transformar o áudio em representações tempo-frequência (por exemplo, Mel-spectrogramas), conseguimos explorar as capacidades da CNN para aprender padrões espaciais semelhantes aos das imagens, capturando variações de frequência e estrutura temporal de forma mais robusta do que o MLP. Além disso, as CNNs são modelos amplamente utilizados no estado-da-arte do UrbanSound8K, garantindo uma melhoria de desempenho e permitindo uma análise comparativa mais completa entre arquiteturas.

Pré-processamento

- Fixámos seeds para garantir resultados estáveis.
- Garantimos que o número de frames é calculado automaticamente sem truncamentos artificiais.
- Padronizámos todos os áudios para 4 segundos
- Implementámos um sistema de cache para evitar recomputar features, acelerando drasticamente o pipeline.

Escolhemos três tipos de features:

- Log-Mel Spectrograms → muito eficazes em CNNs para áudio
- MFCC + deltas + delta-deltas → destacam informação dinâmica útil em sons humanos/ambientais.
- Both (concat) → permite testar se a combinação melhora a representatividade do sinal.

Data Augmentation

- Aplicámos time stretch, pitch shift, ruído e time shift → aumentam a variabilidade temporal e harmónica dos sons
- Implementámos também SpecAugment (time/frequency masking), que força a CNN a aprender padrões mais globais



Feature	Blocos Conv	Pool / Norm / Dropout	Extração final	Dense / Saída	Observações
Mel Spectrograms	3: Conv2D (3×7) → SeparableConv2D (3×3) → SeparableConv2D (3×5)	BatchNorm + MaxPooling + Dropout (0.25→0.35→0.40)	GlobalAveragePooling2D	Dense(128)+Dropout (0.5) → Softmax	Kernels largos no 1º bloco, GAP reduz overfitting.
MFCC	3: Conv2D (3×3) → SeparableConv2D (3×3) → SeparableConv2D (3×5)	BatchNorm + MaxPooling + Dropout (0.25→0.35→0.40)	GlobalAveragePooling2D	Dense(128)+Dropout (0.5) → Softmax	Adaptado à dimensão mais compacta dos MFCCs.
Both (Mel + MFCC)	3: Conv2D assimétrica (5×7) → SeparableConv2D (3×5) → SeparableConv2D (5×7)	BatchNorm + MaxPooling + Dropout (0.25→0.35→0.40)	GlobalAveragePooling2D	Dense(128)+Dropout (0.5) → Softmax	Mais informação na frequência Mel + MFCC empilhados verticalmente

Resultados CNN

Mel spectrograms: melhor compromisso desempenho/estabilidade, accuracy $\sim 0.58 \pm 0.077$

MFCC: representação mais uniforme, ligeiramente menor desempenho, accuracy $\sim 0.54 \pm 0.061$

Both (Mel + MFCC): maior variabilidade e instabilidade, accuracy $\sim 0.51 \pm 0.18$

Tendência geral: losses seguem padrão similar; modelos individuais mais consistentes

Observação: fusão direta aumenta complexidade sem melhorar classificação; resultados alinhados com literatura de CNNs leves.

Ensemble CNN + MLP (end to end)

Objetivo: combinar CNN e MLP para explorar complementaridade de representações internas. CNN extrai embeddings hierárquicos de Mel spectrograms; MLP refina decisões com combinações não lineares.

Treino end-to-end: CNN e MLP ajustam pesos simultaneamente, sem fusão manual de embeddings.

MLP usa hiperparâmetros previamente otimizados para estabilidade e boa generalização. Pipeline simples, coerente com experimentos anteriores, aproveitando características aprendidas pela CNN.

- Desempenho abaixo da CNN isolada.
- Razão principal: MLP recebe ativação dinâmica da CNN, não embeddings estáveis → gradientes ruidosos, instabilidade e overfitting.
- Capacidade do modelo aumentada, mas dataset pequeno limita generalização.
- Interação CNN + MLP degrada performance média.
- Métricas:
- Accuracy: 0.384 ± 0.134
- Loss: 1.990 ± 0.334

DeepFool

O DeepFool é um algoritmo de ataque adversarial que tem como objetivo medir a robustez de modelos de Machine Learning, como Redes Neurais Convolucionais (CNNs) e Multi-Layer Perceptrons (MLPs).

A sua principal característica é que ele encontra a perturação mínima (o menor ruído) necessária para forçar o modelo a mudar a sua predição de uma amostra corretamente classificada.

O que faz o algoritmo?

O DeepFool opera iterativamente, movendo a amostra original (x) o mínimo possível em direção à fronteira de decisão mais próxima do classificador.

- Perturbação Mínima: O algoritmo calcula a menor perturbação (r) que, quando adicionada à imagem original (x), resulta numa nova amostra adversarial (x'), forçando o modelo a classificar x' numa classe diferente.
- Métrica de Robustez (ρ): O resultado principal é a magnitude relativa da perturbação, frequentemente representada por ρ . Um ρ mais baixo significa que o modelo é menos robusto (mais vulnerável), pois requer menos esforço para ser enganado.

Porque usar este algoritmo?

O DeepFool fornece uma quantificação rigorosa da robustez, que é essencial para avaliar a segurança e fiabilidade de um modelo:

- Avaliação da Vulnerabilidade: Permite saber exatamente o quanto perto as amostras estão das fronteiras de decisão.
- Comparação de Arquiteturas: É usado, como no nosso projeto, para comparar a robustez intrínseca de diferentes modelos (e.g., MLP vs. CNN) sob o mesmo ataque.

Em resumo, o DeepFool é uma ferramenta para testar o "limiar de falha" de um modelo.

No nosso projeto, decidimos usar este algoritmo para avaliar a robustez dos nossos modelos, isto é, avaliamos tanto para o MLP, como para o CNN.



DeepFool - Análise de Resultados (MLP)

O ataque DeepFool mostrou que pequenas perturbações são suficientes para enganar o modelo na maioria das amostras.

Estatísticas das perturbações (ρ):

- Mediana $\approx 2.08 \rightarrow$ a maioria dos exemplos requer alteração mínima
- Média $\approx 19.74 \rightarrow$ inflacionada por outliers (ex.: fold 3)
- Min: 0.34, Max: 172.77

Visualizações:

- Histograma: maioria das perturbações < 10 , pico entre 1.5–2.0
- Boxplot: concentração próxima de zero, outlier destacado

Conclusão:

- Modelo é sensível a perturbações adversariais pequenas
- Exceções mostram que algumas amostras/folds exigem perturbações maiores
- DeepFool fornece insights sobre robustez e vulnerabilidades do MLP



DeepFool - Análise de Resultados (CNN)

O modelo foi vulnerável ao DeepFool na grande maioria dos casos.

Estatísticas globais das perturbações (ρ):

- Fooling rate geral: 98.57%
- Mediana ρ : 0.007 → a maioria das amostras requer perturbação mínima
- Média ρ : 3.30 → alguns outliers elevam o valor

Distribuição das perturbações:

- Maioria muito próxima de zero
- Alguns outliers indicam classes mais robustas

Por fold:

- Fooling rate entre 90%–100%
- Folds 6 e 9 → ligeiramente mais robustos
- Folds 7 e 10 → mais vulneráveis

Conclusão:

- CNN é altamente vulnerável a pequenas alterações nos espectrogramas
- Variabilidade entre folds e classes revela que certos sons são mais estáveis
- Resultados indicam necessidade de adversarial training ou data augmentation para maior robustez



Trabalho Futuro

Para trabalho futuro, seria importante melhorar a representação temporal dos sons, já que os modelos mostraram dificuldades em distinguir classes contínuas e parecidas. Uma possível evolução é testar arquiteturas que lidem melhor com sequências, como modelos recorrentes simples (LSTM ou GRU), ou adicionar pequenos módulos de atenção para ajudar o modelo a focar nas partes mais informativas do áudio. Outra melhoria natural seria aumentar o dataset com data augmentation mais variado e mais controlado, de forma a reduzir overfitting e melhorar a robustez.