

Avaliação 1 (Trabalho) de Redes Neurais - NES

prof. Eduardo Adame

17 de setembro de 2025

Avaliação 1 (Trabalho) de Redes Neurais - NES

Formato: individual. **Peso:** 35% da nota da disciplina. **Entrega:** relatório + código + apresentação oral com demonstração do código.

1) Objetivo

Consolidar os tópicos vistos até o momento - revisão de ML/Gradient Descent, redes neurais, retropropagação, Keras, regularização (L2/Dropout) e introdução a CNNs — em um mini-projeto aplicado, com ênfase na **clareza de raciocínio**, **experimentação controlada** e **comunicação técnica** (apresentação + visualização/interação ao vivo).

2) Entregáveis

1. **Relatório curto (máx. 4 páginas)** em PDF (figuras numeradas, legendas claras, referências). Sugestão: escreva em Markdown/LaTeX e exporte para PDF, o template do NeurIPS é um bom padrão.
 2. **Repositório** (zip ou link) com:
 - `notebooks/` (ou `src/`) contendo o código.
 - `README.md` com instruções de execução **reprodutíveis**.
 - `requirements.txt` e/ou `pyproject.yml`.
 - Pasta `figures/` com as imagens usadas no relatório.
 3. **Apresentação oral (7–10 min) + demonstração interativa** (Streamlit/Gradio/Jupyter/Marimo) executável localmente (e/ou no Google Colab).
-

3) Regras e restrições (para facilitar correção e foco)

- **Dados:** use **um** dos temas/datasets listados na Seção 7 (cada tema é exclusivo; escolha via planilha de inscrição). Caso queira propor variação **simples** (subconjunto/recorte), peça aprovação.
 - **Orçamento computacional:** limite de ≤ 2 horas de treino total em CPU/GPU padrão (otimize batch size, early stopping, etc.).
 - **Arquiteturas:** máximo de **1 modelo MLP** e **1 modelo CNN** por tema (comparação direta).
 - **Parâmetros:** ≤ 1 milhão de parâmetros por modelo.
 - **Épocas:** ≤ 50 (use early stopping/paciência).
 - **Sementes:** fixe `seed=42` (ou outro de sua escolha, mas **documente**). Reporte média \pm desvio de **3 rodadas** quando possível.
 - **Proibido:** usar modelos pré-treinados/pesados (e.g., ResNet50 pretrain), AutoML, ou pipelines automáticos que “resolvam sozinho”. **Permitido:** camadas básicas Keras, callbacks, Data Augmentation simples, e bibliotecas de visualização.
 - **IA generativa:** pode ser usada **apenas** para refatorar/esclarecer trechos, **não** para gerar o projeto/relatório por completo. Você deve **explicar** cada decisão e **defender** seu código.
-

4) Estrutura sugerida do relatório (máx. 4 pág.)

1. **Título e autoria** (nome, professor, curso, tema/dataset), essencialmente um cabeçalho.
2. **Pergunta/hipótese:** o que você quer demonstrar? (ex.: “Dropout reduz overfitting no dataset X”).
3. **Dados:** descrição breve do dataset (origem, número de exemplos, classes, divisão treino/val/teste, pré-processamento). Inclua **2–3 figuras** ilustrativas (análise exploratória de dados).
4. **Modelos:** descreva **MLP** (camadas, ativação) e **CNN** (conv/pool, kernel, filtros), justificando escolhas.
5. **Treinamento:** otimizador, taxa de aprendizado, **GD/SGD** escolhido, função de perda, **regularização** (L2/Dropout), **early stopping**.
6. **Experimentos:** desenho do experimento, métricas (acurácia, F1, etc.), protocolo de repetição (3 sementes), tabelas e **curvas de treino/val**.
7. **Visualização/Interação:** descreva e mostre sua demo (ver §6).
8. **Resultados:** tabela comparando MLP vs CNN (se possível com \pm desvio), discussão de **overfitting/underfitting**.
9. **Análise de erros:** matriz de confusão, exemplos mal classificados, possíveis causas.
10. **Conclusões e próximos passos.**
11. **Reprodutibilidade:** como executar (comando único), versões, tempo total.
12. **Apêndice:** código essencial (blocos principais) + link para repositório.

Checklist de figuras obrigatórias: (a) exemplos do dataset, (b) curvas de perda/acurácia treino vs validação, (c) matriz de confusão no teste, (d) pelo menos **uma** técnica interpretável (Grad-CAM, mapas de ativação ou filtros aprendidos) **para CNN**.

5) Rubrica de avaliação (35% da nota da disciplina)

Nota total do trabalho = 10 pontos (peso 0,70 na média final)

- **Apresentação + sustentação oral (6,0 pts)**
 - Clareza e estrutura (2)
 - Demonstração interativa fluida (1,5)
 - Defesa técnica (responde por quê/como, limitações) (2,5)
- **Relatório (2,5 pts)**
 - Escrita objetiva, figuras legíveis, método reproduzível (1,5)
 - Correção técnica (modelos, treino, métricas) (1,0)
 - Discussão crítica e análise de erros (1,0)
- **Código + reprodutibilidade (1,0 pt)**
 - Organização do repo, README, rodar com 1 comando (0,5)
 - Sementes fixas, resultados consistentes (0,5)
- **Interação/visualização (0,5 pt)**
 - App simples (Streamlit/Gradio/Jupyter widgets/Marimo) com inputs e saídas informativas (0,6)
 - Visual explicativa (Grad-CAM/ativação/filtros) (0,4)

Bônus de +0,5: utilizar \LaTeX para redigir o relatório. Uma referência é meu curso.

6) Requisito de Interação/Visualização

Implemente **uma** das opções abaixo (ou equivalente), integrada ao seu modelo treinado: - **Streamlit/Gradio:** upload de imagem + botão “Prever”, exibindo probabilidade por classe, **Grad-CAM** (ou saliency) e explicação curta. - **Jupyter widgets:** sliders para taxa de dropout e L2, reexecutando **apenas inferência** em um lote fixo e mostrando efeito nas ativações. - **Painel de métricas:** seleção de classe para filtrar matriz de confusão e listar top-5 erros com thumbnails. - **Notebook Interativo:** o marimo possibilita reatividade entre todas as células de um notebook. Usá-lo com widgets, por si só, é uma interação.

Inclua **print** da interface no relatório e garanta execução com `streamlit run app.py` (ou `python app.py`).

7) Temas/Datasets (15 vagas — um por aluno)

Escolha **um** tema. Todos incluem: (i) MLP baseline vs CNN, (ii) regularização L2/Dropout, (iii) análise de overfitting, (iv) visual interpretável.

1. **MNIST vs ruído**: MNIST com e sem ruído gaussiano (σ escolhido). Pergunta: a CNN é mais robusta que MLP? Visualizar filtros iniciais vs tardios.
2. **Fashion-MNIST data aug**: impacto de `RandomFlip/Rotation` na generalização; compare com aumentar Dropout.
3. **KMNIST (kana)**: classes mais confusas; Grad-CAM em amostras ambíguas.
4. **CIFAR-10 (subconjunto 5 classes)**: efeito de kernel 3×3 vs 5×5 e pooling; limite de parâmetros $\leq 500k$.
5. **Cats vs Dogs (subset 10k)**: data cleaning leve (remover corrompidas) e comparação de normalização $[0,1]$ vs padronização por canal.
6. **EuroSAT (RGB, subset 10k)**: avaliar invariância a rotações; CNN pequena com 2–3 blocos conv.
7. **ASL Alphabet (subset)**: distinguir gestos similares; ablação de Dropout (0/0,3/0,5).
8. **MedMNIST — OrganMNIST**: diagnóstico de overfitting; early stopping e L2.
9. **Leaf Classification (folhas)**: RGB→tons de cinza: ajuda ou atrapalha? Mostrar mapas de ativação.
10. **Chest X-ray (Pneumonia, subset)**: balanceamento por classe (class weights) e impacto no F1.
11. **CIFAR-10 com canais removidos**: treinar apenas com 1 canal (R ou G ou B). Como a CNN se adapta?
12. **QuickDraw (subset de 10 classes)**: linhas finas; conv 1×1 vs 3×3 .
13. **Sign Language Digits (RGB)**: normalização por imagem vs por dataset.
14. **Garbage Classification (6 classes, subset)**: robustez a background; recorte central vs resize simples.
15. **Traffic Signs (subset GTSRB)**: sensibilidade a blur; comparar data aug vs Dropout.

Observação: se algum dataset for muito grande, **use subconjunto estratificado** (p.ex., 2–10k imagens) e documente o critério de amostragem. Todos são facilmente obtidos via `tensorflow_datasets` ou links públicos.

8) Pipeline mínimo esperado (Keras)

1. **Carregamento**: TFDS/Keras utils, split treino/val/teste fixo.
2. **Preprocessamento**: resize, normalização e (se aplicável) data augmentation.
3. **Modelos**:

- **MLP:** Flatten → Dense → ReLU → Dropout → Dense(num_classes)
 - **CNN:** Conv → ReLU → MaxPool (2-3 blocos) → Flatten/GlobalAvgPool → Dense + L2/Dropout
4. **Treino:** Adam (ou SGD), EarlyStopping/ReduceLRonPlateau, ModelCheckpoint.
 5. **Métricas:** acurácia, F1 (macro), matriz de confusão; repetir com 3 seeds.
 6. **Visualização:** curvas, Grad-CAM (ou saliency) para 3 exemplos representativos.
 7. **Interação:** app simples conforme §6.

PS: Você pode usar PyTorch ou JaX.

9) Cronograma

- **Inscrição no tema:** até **19/09/2025**.
 - **Checkpoint:** **24/09/2025** — depois da aula conversaremos sobre o trabalho
 - **Entrega (PDF + código):** **01/10/2025** (15h00).
 - **Apresentações:** **01/10/2025** (15h00) (ordem sorteada).
-

10) Template de repositório sugerido

```
midterm-nn-
├── README.md
├── requirements.txt
├── data/                # vazio ou script de download
├── notebooks/
│   ├── 00_exploracao.ipynb
│   ├── 01_mlp_baseline.ipynb
│   ├── 02_cnn_model.ipynb
│   └── 03_gradcam_e_erros.ipynb
├── src/
│   ├── data.py          # carregamento/preprocessamento
│   ├── models.py        # arquiteturas MLP e CNN
│   ├── train.py         # treino e avaliação
│   ├── viz.py           # curvas, matriz de confusão, grad-cam
│   └── app.py           # streamlit/gradio
└── figures/
```

README mínimo:

```
pip install -r requirements.txt
python -m pip install tensorflow tensorflow-datasets streamlit gradio scikit-learn matplotlib
# Treino
```

```
python src/train.py --model cnn --epochs 30 --seed 42
# App
streamlit run src/app.py
```

11) Entrega

Submeta no Classroom: **PDF** (até 4 páginas) + **zip** do repositório (e link) até **01/10/2025 15:00**. Testaremos o app durante a sua apresentação.

Dúvidas ou adaptações (acessibilidade, recursos): fale comigo até **28/09/2025**.