Projeto 2: Reconhecimento de Faces

Trabalho de Inteligência Computacional Aplicada

Agosto de 2025

Resumo

Este relatório apresenta um resumo abrangente das atividades desenvolvidas no projeto de reconhecimento de faces, cobrindo todas as etapas entre a preparação das imagens do conjunto Yale A e a avaliação de múltiplos classificadores. A metodologia baseou-se em um pipeline modular implementado em Python, com foco na avaliação comparativa de modelos lineares e não lineares sob diferentes transformações de pré-processamento. Para cada atividade foram adotados protocolos estatísticos rigorosos de seleção de hiperparâmetros via *random search*, estratificação do conjunto de dados e repetições independentes de treino e teste. Os resultados englobam métricas de acurácia, precisão, recall e F1-score, além dos tempos médios de treinamento e predição. Também são discutidos os efeitos da aplicação da Análise de Componentes Principais (PCA) como rotação e como redução de dimensionalidade, bem como o impacto da transformação Box–Cox seguida de padronização.

1 Introdução

O objetivo deste projeto é avaliar o impacto de diferentes cadeias de pré-processamento e de modelos de classificação no problema de reconhecimento de faces, utilizando o conjunto de imagens Yale A. As atividades foram estruturadas de modo incremental. Inicialmente, avalia-se o desempenho de quatro classificadores (Mínimos Quadrados, PL, MLP de uma camada oculta e MLP de duas camadas ocultas) diretamente sobre os vetores de pixels das imagens vetorizadas. Em seguida, investiga-se a aplicação da Análise de Componentes Principais (PCA) tanto como rotação do espaço quanto como redução de dimensionalidade, e a transformação Box–Cox sobre os componentes reduzidos. Cada atividade é acompanhada de seleção de hiperparâmetros, treinamento e repetições para estimar a distribuição de desempenho dos modelos. O presente relatório consolida a metodologia e analisa os resultados obtidos.[1]

[1] As escolhas de pré-processamento, protocolos de busca e justificativas de modelagem estão documentadas em detalhes no relatório técnico fornecido com o projeto.

2 Metodologia Geral

2.1 Conjunto de dados e pré-processamento

Todas as imagens do conjunto Yale A são lidas da estrutura de diretório plana e convertidas para tons de cinza pela média dos canais. Para permitir experimentos computacionalmente factíveis, as imagens são reamostradas por vizinho mais próximo para diferentes resoluções quadradas (20×20, 30×30 e 40×40), concatenando-se os pixels linha a linha para formar vetores de dimensão $d = h \times w$. Esta etapa constitui a Atividade 1, cujo objetivo é quantificar o compromisso entre dimensão do vetor de atributos, custo computacional e desempenho dos modelos . A partir do gráfico "tempo × escala", escolhe-se uma resolução alvo (40×40) para as demais atividades. As imagens podem ser normalizadas de quatro maneiras: nenhuma normalização, *z-score*, min–max em [0, 1] ou min–max em [-1, 1]; esta escolha torna-se um hiperparâmetro nas atividades 2, 4 e 6.

2.2 Modelos de classificação

Foram comparados quatro classificadores:

1. **Mínimos Quadrados (MQ).** Utiliza solução analítica da regressão linear multiclasse com termo de viés e regularização ℓ_2. O único hiperparâmetro é o peso de penalização $\lambda \in \{0, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$.

2. **Perceptron Logístico (PL).** É a regressão logística multiclasse treinada por descida em lote com perda de entropia cruzada e regularização ℓ_2. O espaço de busca inclui taxa de aprendizado $\eta \in \{5 \times 10^{-3}, 10^{-2}, 2 \times 10^{-2}\}$, número de épocas $\in \{100, 200, 300\}$, penalização $\lambda \in \{0, 10^{-4}, 10^{-3}\}$, otimização {SGD, Momentum, Nesterov, RMSProp, Adam} e, quando aplicável, normalização em {nenhuma, *z-score*, min–max [0,1], min–max [-1,1]}. O espaço total de combinações nas atividades com normalização é de 540 possibilidades, das quais 60 são amostradas via *random search* para seleção.

3. **Rede MLP com uma camada oculta (MLP-1H).** Varia a largura da camada em $\{16, 32, 64, 128, 256, 512\}$, a função de ativação em {tanh, sigmoide, ReLU, Leaky–ReLU, ReLU6, Swish}, a taxa de aprendizado em $\{5 \times 10^{-3}, 10^{-2}, 2 \times 10^{-2}\}$, o número de épocas em $\{150, 200, 300\}$, a penalização $\ell_2 \in \{0, 10^{-4}, 10^{-3}\}$, o otimizador no mesmo conjunto do PL e um parâmetro de *clipping* de gradiente em $\{2, 5, 10\}$. Quando a normalização faz parte da busca, o espaço discreto com normalização contém 58 320 combinações.

4. **Rede MLP com duas camadas ocultas (MLP-2H).** Amplia a busca para dois vetores de largura (cada um com 6 opções) e mantém

as demais variações da MLP-1H, elevando o espaço para 349 920 combinações. O orçamento de *random search* de 60 amostras é mantido em todas as atividades para explorar regiões promissoras do espaço de forma estocástica.

As redes utilizam inicialização de He, cálculo de *softmax* com o truque *log-sum-exp* e *clipping* dos gradientes para garantir estabilidade numérica, principalmente em combinações de learning rate elevadas e funções de ativação não saturantes.

Vale ressaltar que o *clipping* elemento a elemento dos gradientes foi introduzido justamente para evitar explosões de gradiente nos primeiros ciclos de treinamento. Em arquiteturas com ativação ReLU ou Swish e taxas de aprendizagem relativamente altas, os gradientes podem crescer rapidamente antes que a rotação da PCA reduza sua variância. Limitar o valor máximo desses gradientes impede picos abruptos e reduz a variabilidade entre as repetições, contribuindo para treinos mais estáveis.

2.3 Protocolo experimental

Cada atividade segue o mesmo protocolo básico: a base é dividida estratificadamente por sujeito em 80% para treino e 20% para teste. Para cada modelo e cenário experimental, realiza-se uma busca aleatória de hiperparâmetros com 60 amostras. Cada combinação candidata é reavaliada em 10 repetições independentes; o conjunto de hiperparâmetros vencedor é então submetido a 50 repetições adicionais, totalizando 650 treinos/predições por modelo e cenário. As métricas relatadas são a acurácia média, desvio-padrão, mínimo, máximo e mediana, além das médias de precisão, recall, F1-score, tempo de treino e tempo de predição. Esse desenho estatístico proporciona estimativas robustas de desempenho e permite comparar modelos e pré-processamentos de forma confiável. É importante frisar que as métricas de precisão, recall e F1 são calculadas na forma de *macro-médias*: cada uma é calculada separadamente para cada classe a partir da matriz de confusão multiclasse e, em seguida, promediada sem ponderação. Esta escolha equilibra a contribuição de cada sujeito e evita que classes mais frequentes dominem a avaliação.

Além disso, sempre que há uso de PCA no pipeline, a transformação é **ajustada exclusivamente** com os dados de treino e **aplicada antes de qualquer normalização subsequente**. Essa ordem é mantida em todas as atividades envolvendo PCA para evitar vazamento de informação e garantir que a geometria seja consistente entre treino e teste.

3 Atividades 1 e 2 – Conjunto Original

3.1 Metodologia

Nas Atividades 1 e 2 trabalham-se as imagens vetorizadas sem redução de dimensionalidade. Na Atividade 1 são avaliadas três escalas quadradas 20×20, 30×30 e 40×40. O gráfico *tempo × escala* apresentado na Figura 1 compara o tempo médio de treinamento dos quatro classificadores em cada tamanho de imagem. Observa-se que a resolução 40×40 oferece o melhor compromisso entre desempenho e tempo de treino, especialmente para o PL e as MLPs, que exibem tempos relativamente baixos nesse formato. Assim, adota-se 40×40 como escala padrão para as demais atividades. Na Atividade 2, para a escala escolhida, selecionam-se os melhores hiperparâmetros para cada classificador considerando também a normalização. A normalização é avaliada como parte do espaço de busca e inclui as quatro opções descritas anteriormente.

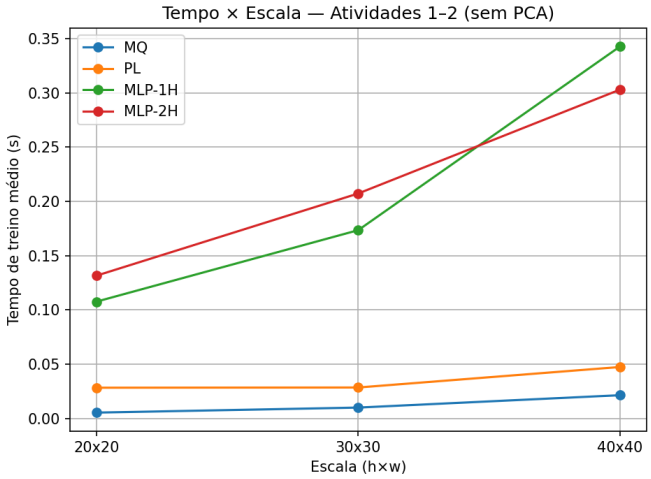

Figura 1: Comparação dos tempos médios de treinamento nas escalas 20×20, 30×30 e 40×40 para os classificadores MQ, PL, MLP-1H e MLP-2H. O eixo horizontal indica a escala e o eixo vertical o tempo médio de treino (segundos). Nota-se que a resolução 40×40 alcança menores tempos de treino para os modelos de interesse, justificando sua adoção nas etapas seguintes.

3.2 Algoritmos vencedores e configurações

A Tabela 1 resume as melhores combinações de hiperparâmetros (obtidas por random search) e as principais métricas de desempenho. Na ausência de PCA, o Perceptron Logístico apresentou a maior acurácia média (0,86) com normalização min–max [-1,1], otimização Nesterov, taxa de aprendizado 0,02, 100 épocas e regularização 10^{-4}. As MLPs alcançaram acurácias com-

paráveis (0,86), mas com custos de treino muito superiores (1,14 s para a MLP-1H e 0,61 s para a MLP-2H) em comparação ao PL (0,029 s) e ao MQ (0,036 s). O MQ, apesar de simples, atingiu acurácia média de 0,82 usando min–max [0,1] e penalização 0,1.

Tabela 1: Resultados médios das Atividades 1–2 (sem PCA). Cada valor é média de 50 repetições com desvio-padrão indicado.

Modelo	Norm	Opt	Ativação	λ_{ℓ_2}	Acurácia	Precisão (macro)
MQ	min–max [0,1]	–	–	0,10	$(0{,}8191 \pm 0{,}0500)$	0,8558
PL	min–max [-1,1]	Nesterov	–	(10^{-4})	$(0{,}8622 \pm 0{,}0500)$	0,8912
MLP-1H	min–max [0,1]	Momentum	tanh	0,00	$(0{,}8578 \pm 0{,}0500)$	0,8842
MLP-2H	min–max [-1,1]	Nesterov	tanh	(10^{-4})	$(0{,}8551 \pm 0{,}0500)$	0,8854

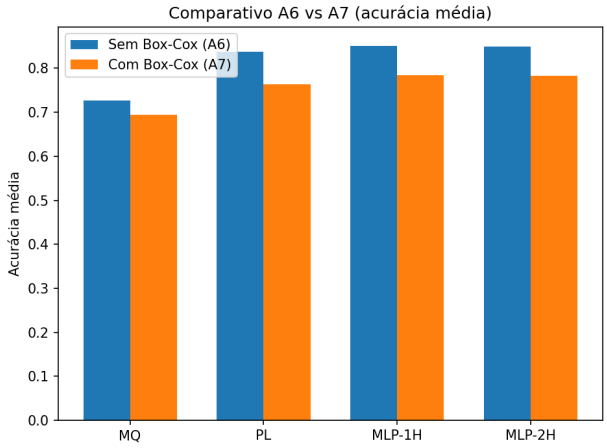

Figura 2: Comparação gráfica das acurácias médias entre as Atividades 6 (PCA, $q = 10$) e 7 (PCA + Box–Cox). Em todas as arquiteturas observa-se queda de desempenho com a aplicação da Box–Cox, reforçando que a transformação não traz benefícios neste conjunto.

3.3 Discussão dos resultados

O PL despontou como o classificador com melhor compromisso entre acurácia e tempo de treino: obteve acurácia média de 0,8622, superando levemente as MLPs (0,858) e o MQ (0,8191) ao mesmo tempo em que exigiu apenas ≈0,029 s por repetição para treinar. As MLPs apresentaram desempenho competitivo, mas com tempo de treino uma ordem de grandeza maior. Em predição, os tempos de inferência foram da ordem de dezenas de microssegundos para o MQ e o PL e de aproximadamente 0,7 ms para as MLPs;

mesmo assim o custo de treino torna o PL mais atrativo para aplicações em que a eficiência é importante. A diferença entre MLP-1H e MLP-2H foi pequena, não justificando o aumento de complexidade da segunda camada. O MQ funcionou como linha de base simples e rápida, mas sua acurácia inferior limita seu uso.

4 Atividades 3 e 4 – PCA como Rotação

4.1 Metodologia

Na Atividade 3 aplica-se a PCA como uma rotação ortogonal do espaço de atributos, mantendo todas as d componentes ($q = d$) e, portanto, sem redução de dimensionalidade. Esta rotação decorrela as variáveis e melhora o condicionamento numérico, mas não altera a informação contida nos dados. Em seguida, a Atividade 4 repete o protocolo de busca e avaliação dos classificadores utilizando as saídas da PCA como entradas. A normalização continua sendo tratada como hiperparâmetro.

4.2 Algoritmos vencedores e configurações

Os melhores hiperparâmetros após a PCA como rotação estão resumidos na Tabela 2. Verificam-se reduções expressivas nos tempos de treino para o MQ e para as MLPs: o MQ caiu de 0,036 s para aproximadamente $3{,}3 \times 10^{-4}$ s em média. As MLPs reduziram seus tempos de treino em mais de 40 MLP-2H de 0,612 s para 0,529 s. O Perceptron Logístico, em contrapartida, apresentou leve aumento de tempo (de 0,029 s para 0,040 s). Em termos de acurácia, a resposta foi assimétrica: as MLPs melhoraram ou mantiveram o desempenho (0,868 para a MLP-1H), enquanto os modelos lineares (MQ e PL) sofreram pequena queda (0,805 e 0,818, respectivamente).

Tabela 2: Resultados médios das Atividades 3–4 (PCA como rotação, $q = d$). Cada valor de acurácia indica média e desvio-padrão; são também reportadas as macro-médias de precisão, recall e F1.

Modelo	Norm	Opt	Ativação	λ_{ℓ_2}	Acurácia	Precisão (macro)	Re
MQ	sem norm	–	–	0,001	$(0{,}8049 \pm 0{,}0400)$	0,8445	
PL	z-score	Nesterov	–	0,001	$(0{,}8178 \pm 0{,}0500)$	0,8566	
MLP-1H	min–max [0,1]	Adam	Swish	0,001	$(0{,}8680 \pm 0{,}0500)$	0,8966	
MLP-2H	min–max [0,1]	Adam	tanh	0,0001	$(0{,}8542 \pm 0{,}0500)$	0,8855	

4.3 Discussão dos resultados

A rotação pela PCA gerou fortes ganhos computacionais para todos os classificadores, como previsto, devido à decorrelação e ao melhor condicionamento da Hessiana. As MLPs se beneficiaram levemente em acurácia (a MLP-1H subiu de 0,858 para 0,868) e reduziram o tempo de treino em 40 modelos lineares perderam desempenho (MQ de 0,819 para 0,805 e PL de 0,862 para 0,818), possivelmente porque a rotação altera a orientação da fronteira ótima em conjunto com as novas normalizações e reguladores selecionados. Em suma, a PCA como rotação acelera o treino e pode favorecer redes neurais, mas não garante ganhos para modelos lineares.

5 Atividade 5 – Escolha do número de componentes q

O passo seguinte consiste em determinar o número q de componentes principais necessário para preservar pelo menos 98 % da variância dos dados. A curva de variância explicada acumulada evidenciou um "joelho" pronunciado: para a escala 40×40 ($d = 1{,}600$) apenas 10 componentes já retêm mais de 98 % da variância. O valor final adotado foi $q = 10$, conforme registrado no arquivo de controle. A Figura 3 mostra a curva acumulada com a linha tracejada em 98 % e o ponto correspondente a $q = 10$. Este valor fundamenta a redução aplicada nas atividades seguintes.

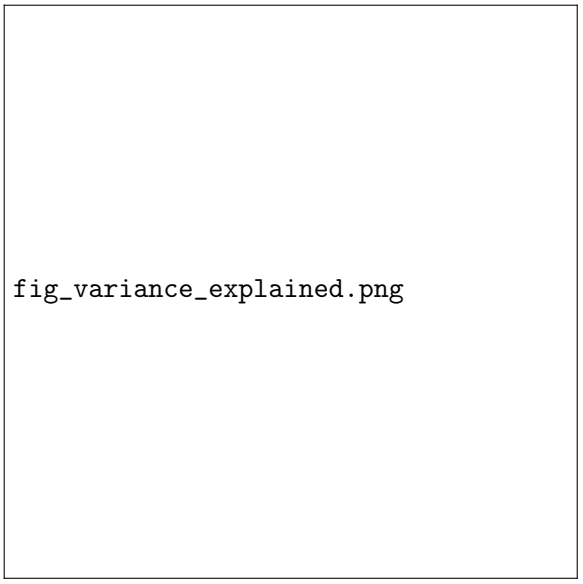

Figura 3: Curva de variância explicada acumulada. A linha tracejada marca 98 % da variância e a linha vertical indica o ponto $q = 10$.

6 Atividade 6 – PCA com redução ($q = 10$)

6.1 Metodologia

Com $q = 10$ fixado, aplica-se a PCA como redução de dimensionalidade. O pipeline adotado lê as imagens, ajusta a PCA exclusivamente sobre o conjunto de treino para obter 10 componentes e projeta os dados de treino e teste nesse subespaço. Somente após a projeção é aplicada a normalização escolhida (*z-score* ou min–max), tratada como um hiperparâmetro da busca. Essa ordem reflete o que foi implementado no código: tanto a PCA quanto os parâmetros de normalização são estimados a partir dos dados de treino, evitando vazamento de informação. Após a preparação dos atributos reduzidos procede-se à busca de hiperparâmetros para cada classificador. A redução de dimensionalidade não apenas compacta os vetores (de dimensão 1600 para 10), mas também preserva a maior parte da variância relevante, conforme demonstrado na Atividade 5.

6.2 Resultados e configurações vencedoras

A Tabela 3 apresenta os melhores hiperparâmetros e métricas para $q = 10$. Observa-se uma grande economia de tempo para as MLPs: o treino da MLP-1H caiu para 0,078s e da MLP-2H para 0,195s. O PL manteve tempos baixos (0,026 s) e acurácia média de 0,837, recuperando grande parte do desempenho perdido na PCA sem redução. A MLP-1H continuou com a melhor acurácia (0,852), embora o ganho sobre o PL seja pequeno. O MQ piorou sensivelmente, ficando com acurácia média de 0,727.

Tabela 3: Resultados médios da Atividade 6 (PCA com $q = 10$). São apresentados os hiperparâmetros vencedores, as macro-médias de precisão, recall e F1 e os tempos médios de treinamento.

Modelo	Norm	Opt	Ativação	λ_{ℓ_2}	Acurácia	Precisão (macro)	F
MQ	min–max [1,1]	–	–	0,01	$(0{,}7267 \pm 0{,}0400)$	0,7529	
PL	z-score	RMSProp	–	0,001	$(0{,}8373 \pm 0{,}0600)$	0,8750	
MLP-1H	min–max [1,1]	Adam	Swish	0,001	$(0{,}8516 \pm 0{,}0500)$	0,8889	
MLP-2H	min–max [1,1]	Nesterov	tanh	0,0001	$(0{,}8498 \pm 0{,}0500)$	0,8854	

6.3 Discussão

A redução de dimensionalidade para $q = 10$ produziu dois fenômenos complementares: um ganho computacional expressivo e uma reorganização do desempenho relativo dos modelos. As MLPs reduziram seu tempo de treino em mais de uma ordem de grandeza em relação ao cenário original, enquanto o PL manteve-se barato. O MQ foi o mais penalizado, com queda de acurácia

(0,73). O PL recuperou parte do desempenho e atingiu acurácia próxima à original, tornando-se novamente competitivo. As MLPs preservaram acurácia elevada (0,85) com custo muito menor, sendo recomendadas quando se busca o melhor desempenho absoluto.

7 Atividade 7 – PCA com Box–Cox e *z-score*

7.1 Metodologia

A sétima atividade investiga o efeito de aplicar a transformação Box–Cox seguida de padronização *z-score* no espaço de componentes principais reduzido para $q = 10$. A Box–Cox exige entradas positivas; cada componente principal é deslocado por uma constante mínima e o parâmetro λ_{BC} é ajustado por máxima verossimilhança em uma grade densa no intervalo $[-2, 2]$. Após a transformação, os dados são padronizados e a busca de hiperparâmetros é repetida, mas não se utiliza normalização adicional.

7.2 Resultados e comparação com a Atividade 6

A Tabela 4 mostra os resultados médios com Box–Cox. Comparando com a Atividade 6, nota-se queda de desempenho em todos os modelos: a acurácia do MQ cai de 0,73 para 0,69, a do PL de 0,84 para 0,76 e das MLPs de 0,85 para 0,78. Os tempos de treino aumentam ligeiramente, sobretudo nas MLPs, devido ao custo extra da transformação.

Tabela 4: Resultados médios da Atividade 7 (PCA com Box–Cox + *z-score*). Para cada modelo são mostrados os hiperparâmetros vencedores, as macro-médias de precisão, recall e F1 e os tempos médios de treinamento.

Modelo	Opt	Ativação	λ_{ℓ_2}	Acurácia	Precisão (macro)	Recall (macro)	
MQ	–	–	0,0001	$(0{,}6947 \pm 0{,}0500)$	0,7458	0,6947	
PL	Nesterov	–	0	$(0{,}7644 \pm 0{,}0600)$	0,8316	0,7644	
MLP-1H	RMSProp	sigmoide	0,0001	$(0{,}7844 \pm 0{,}0700)$	0,8523	0,7844	
MLP-2H	Nesterov	Swish	0	$(0{,}7827 \pm 0{,}0600)$	0,8475	0,7827	

Uma comparação direta entre as atividades 6 e 7 é apresentada na Tabela 5. A coluna Δ indica a diferença entre a acurácia média sem e com Box–Cox. Em todos os casos a diferença é negativa, com quedas de 3,2 p.p. no MQ e cerca de 6–7 p.p. nos demais modelos. As razões para essa degradação foram discutidas no relatório técnico: os componentes principais já são aproximadamente gaussianos e centrados, de modo que a Box–Cox (uma transformação monótona para variáveis positivas) distorce a geometria do espaço reduzido e introduz ruído desnecessário.

Tabela 5: Comparação entre as acurácias médias da Atividade 6 (base) e da Atividade 7 (Box–Cox).

Modelo	Acurácia base	Acurácia Box–Cox	Δ
MQ	0,73	0,69	-0,03
PL	0,84	0,76	-0,07
MLP-1H	0,85	0,78	-0,07
MLP-2H	0,85	0,78	-0,07

7.3 Discussão

Os resultados indicam que a combinação PCA + Box–Cox não traz benefícios para o conjunto Yale A com $q = 10$. Como os componentes principais já são quase gaussianos, a Box–Cox distorce as distribuições e reduz a discriminabilidade. A degradação se manifesta tanto na acurácia quanto no F1, e o aumento do custo de treinamento elimina qualquer vantagem potencial. Assim, para este conjunto, recomenda-se omitir a transformação Box–Cox após a PCA reduzida.

8 Conclusões

O estudo sistemático das Atividades 1–7 permite extrair algumas mensagens centrais. Em primeiro lugar, o Perceptron Logístico surge como um classificador forte e eficiente no conjunto original: obtém a maior acurácia entre os modelos lineares e é extremamente rápido. As redes MLP alcançam acurácias semelhantes, mas exigem muito mais tempo de treinamento; portanto, só são justificadas quando pequenas melhorias são críticas.

A aplicação da PCA como rotação diminui drasticamente os tempos de treino e beneficia principalmente as redes neurais. Entretanto, os modelos lineares perdem um pouco de desempenho, indicando que a rotação altera a fronteira ótima em combinação com os hiperparâmetros selecionados.

Ao se reduzir a dimensionalidade para $q = 10$, preservando 98 % da variância, observa-se uma melhoria significativa na eficiência, especialmente para as MLPs. O PL recupera parte da acurácia perdida após a rotação e mantém baixo custo, tornando-se competitivamente atraente. O MQ é penalizado pela redução e exibe acurácia bem inferior.

Por fim, a tentativa de aplicar a transformação Box–Cox seguida de padronização não apresentou ganhos; ao contrário, comprometeu o desempenho de todos os modelos. A análise detalhada mostra que, após a PCA, os dados já apresentam distribuição aproximadamente normal e centrada, deixando pouco espaço para transformações adicionais.

9 Implementação e reprodutibilidade

Todos os modelos avaliados neste projeto foram implementados em Python utilizando rotinas matriciais de baixo nível, o que possibilitou controlar a inicialização dos parâmetros, as funções de custo e a retropropagação nos treinamentos. As variações de Mínimos Quadrados, Perceptron Logístico e redes MLP de uma e duas camadas foram codificadas sem dependências de bibliotecas externas para aprendizado profundo. Nenhum resultado reportado neste relatório utiliza diretamente implementações do `scikit-learn`; eventuais chamadas a bibliotecas externas foram usadas apenas para verificação local e *não constam* nas tabelas.

Todo o material necessário para reproduzir os experimentos (scripts de pré-processamento, cadernos de EDA, variações de normalização e geração de figuras) está disponível no repositório público `https://github.com/LucasJLBraz/Trabalhos_IC_Aplicada`. Os códigos relevantes encontram-se na pasta `src`: o arquivo `tc2_faces_A1_A4.py` implementa as Atividades 1–4, o `tc2_faces_A5_A6.py` trata das Atividades 5–6, e o `tc2_faces_A7.py` aplica a transformação Box–Cox após a PCA reduzida. Um script similar é utilizado para a Atividade 8, reutilizando as rotinas de carga de dados, normalização e classificação.

Os experimentos foram conduzidos em Python 3.11 com NumPy 1.24 e bibliotecas científicas afins. As sementes aleatórias globais (por exemplo, `seed`=42) e de cada repetição estão registradas nos notebooks de apoio e no arquivo de configuração do projeto. Dessa forma, qualquer leitor pode clonar o repositório, instalar as dependências descritas no `requirements.txt` em um ambiente virtual e executar os scripts para obter os mesmos resultados reportados neste relatório.

10 Atividade 8 – Detecção de intrusos

10.1 Metodologia

Esta última atividade aborda o problema de *controle de acesso*: dado um conjunto de sujeitos cadastrados, deve-se identificar se uma nova imagem pertence a um desses indivíduos ou a um intruso. Para compor a classe "intruso" foram capturadas dez fotos próprias seguindo o mesmo padrão do conjunto Yale A — fundo claro e uniforme, variações de iluminação frontal e lateral, expressões de sorriso e surpresa, olhos fechados e um enquadramento com óculos. O enunciado do projeto (TC2_PPGETI_2025.1) solicita 11 imagens de intrusos; contudo, por limitações de disponibilidade foram utilizadas apenas dez. Esse total ainda permite uma divisão estratificada 80/20: em cada repetição oito imagens de intrusos compõem o treino e duas ficam no teste, preservando a validade estatística dos experimentos. Todas essas imagens foram armazenadas na mesma pasta que as demais, com nomes

que não começam por `subject`. Essa convenção faz com que o carregador `build_face_dataset` agrupe automaticamente essas fotos em um único rótulo de "intruso", simplificando a instrumentação.

O pré-processamento segue o mesmo fluxo determinístico adotado nas atividades anteriores. Cada foto é carregada, convertida para escala de cinza e reamostrada por vizinho mais próximo para resolução 40×40, sendo então vetorizada em $\mathbb{R}^{1600}$. Em cada repetição, o conjunto de dados é particionado estratificadamente em 80 % de treino e 20 % de teste, garantindo que os dez intrusos sejam distribuídos proporcionalmente entre as partições. A redução de dimensionalidade é realizada por PCA com $q^* = 10$, valor fixado na Atividade 5 por preservar pelo menos 98 % da variância. Importante salientar que a PCA é ajustada apenas com os dados de treino em cada repetição, evitando vazamento de informação.

Após a PCA, aplica-se a transformação de Box–Cox a cada componente para tornar as distribuições mais simétricas e aproximar a normalidade. O deslocamento mínimo garante que todas as variáveis sejam estritamente positivas e o parâmetro λ_{BC} é estimado por máxima verossimilhança em uma grade densa no intervalo $[-2, 2]$; em seguida os componentes são padronizados por *z-score*. Os parâmetros da Box–Cox e da padronização são aprendidos no treino e reutilizados no teste. Esse pipeline PCA→Box–Cox→*z-score* é fixo na Atividade 8, e a busca de hiperparâmetros se concentra apenas nos classificadores.

Quatro classificadores são avaliados: Mínimos Quadrados (MQ), Perceptron Logístico (PL) e redes MLP com uma e duas camadas ocultas. O espaço de busca para cada modelo reproduz as variações definidas nas atividades anteriores, com sessenta amostras de hiperparâmetros escolhidas aleatoriamente e reavaliadas em dez particionamentos para seleção. Uma vez escolhida a melhor combinação, ela é quantificada em cinquenta repetições adicionais. Para avaliar o *controle de acesso*, considera-se "intruso" como a classe positiva e calcula-se a taxa de falsos negativos (FN/TP+FN), taxa de falsos positivos (FP/FP+TN), sensibilidade (TPR), precisão (PPV) e o F1-score da classe intruso. A acurácia global também é reportada, mas a seleção de hiperparâmetros prioriza o F_1 do intruso.

10.2 Resultados e discussão

A Tabela 6 sintetiza as métricas obtidas para cada classificador. Todas as médias e desvios-padrão foram estimados a partir de cinquenta repetições de treino e teste com partições aleatórias. Observa-se que todos os modelos atingem acurácia próxima de 98 %, mas existem diferenças sutis nas métricas específicas do intruso. O MQ apresenta a menor taxa de falsos negativos (0 %) e a maior sensibilidade (1,0), o que implica que nenhuma das dez fotos de intrusos foi erroneamente classificada como autorizada. Por outro lado, sua precisão é mais baixa (78,2 %), pois cerca de 1,8 % das imagens de

sujeitos legítimos foram rotuladas como intrusos, produzindo falsos positivos. O Perceptron Logístico (PL) exibe desempenho semelhante, com FNR de 4 % e FPR idêntica à do MQ, resultando em F_1 de 0,86. As redes MLP de uma e duas camadas mantêm acurácia alta, porém apresentam FNR de 5 % e 10 %, respectivamente, e maiores custos de treinamento. Considerando a simplicidade e o tempo de ajuste, o MQ e o PL são os modelos mais adequados para o controle de acesso neste conjunto.

Tabela 6: Resultados da Atividade 8: médias e desvios-padrão das métricas de *controle de acesso*.

Modelo	Acurácia	FNR	FPR	Sensibilidade	Precisão	F_1 intruso
MQ	$0{,}9826 \pm 0{,}0200$	0,0000	0,0182	1,0000	0,7824	0,8779
PL	$0{,}9813 \pm 0{,}0200$	0,0400	0,0178	0,9600	0,7757	0,8581
MLP-1H	$0{,}9809 \pm 0{,}0200$	0,0500	0,0178	0,9500	0,7724	0,8520
MLP-2H	$0{,}9787 \pm 0{,}0300$	0,1000	0,0178	0,9000	0,7524	0,8196

Nota: A taxa de falsos negativos (**FNR**) é definida como $\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{TP+FN}}$ e a taxa de falsos positivos (**FPR**) como $\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP+TN}}$. Os valores reportados na Tabela 6 correspondem às médias de cinquenta repetições (quatro casas decimais). O desvio-padrão da FNR para o MQ foi zero, refletindo que nenhuma repetição classificou erroneamente um intruso como autorizado; para os demais modelos os desvios foram maiores (0,136 para o PL, 0,150 para a MLP-1H e 0,224 para a MLP-2H). A FPR média variou ligeiramente entre os modelos (cerca de 0,0178–0,0182), com desvio-padrão 0,023).

Do ponto de vista computacional, os tempos médios de treinamento diferem significativamente. O MQ ajusta-se em menos de 1 ms, o PL em aproximadamente 35 ms, enquanto as MLP-1H e MLP-2H necessitam de cerca de 144 ms e 125 ms por repetição, respectivamente (valores médios derivados de 50 repetições). Esse ganho de eficiência torna o MQ particularmente atrativo quando a latência é crítica. No entanto, em aplicações de segurança nas quais o custo de um intruso aceito é elevado, pode-se preferir um modelo com FNR rigorosamente baixo, mesmo que isso aumente o número de falsos positivos. Ajustar um limiar sobre a probabilidade *softmax* da classe intruso é uma extensão possível para calibrar a FNR e a FPR conforme exigências regulatórias.

No geral, a Atividade 8 demonstra que o pipeline PCA($q = 10$)→Box–Cox→*z-score* permite treinar classificadores eficientes para detectar intrusos no conjunto Yale A. A incorporação de fotografias adicionais reproduzindo as condições originais mostra a viabilidade de estender o sistema para controle de acesso, embora ressalte-se que o uso de apenas dez imagens para a classe intrusa limita a generalização. Testes futuros com intrusos mais variados e com calibração de limiares devem complementar esta análise.

Figura 4: Exemplo de imagem intrusa concebida para a Atividade 8. A figura apresenta uma silhueta genérica em escala de cinza para ilustrar o conceito de intruso sem corresponder a nenhuma pessoa real.