Reconhecimento óptico de caracteres numéricos

Felipe Lemos[1], Fernando Veizaga[1], Gabriel Siqueira[1], Maria Izamara Coutinho[1], Pedro Santos[1], Vinicius Nascimento[1]

[1]Departamento de Computação – Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG)
Av. Amazonas, 7675 - Nova Gameleira, Belo Horizonte - MG, 30510-000

Abstract. *Recent technological advances have resulted in several works in the area of artificial and computational intelligence. Within the scope of this line of research, the present work deals with the problem of optical recognition of numeric characters whose data is made available by MNIST. For this purpose, classification strategies that are quite famous in the field of exact sciences were applied. In the end, they are descriptions that define how good the solution is, making comparisons between different methods possible.*

Resumo. *Os avanços tecnológicos recentes impulsionaram diversos trabalhos na área de inteligência artificial e computacional. No âmbito dessa linha de pesquisa, o presente trabalho trata do problema de reconhecimento óptico de caracteres numéricos cujo os dados são disponibilizados pelo MNIST. Para esse propósito, aplicou-se estratégias de classificação de bastante notoriedade no campo das exatas. Ao final, são apresentadas métricas que definem o quão bom está a solução fazendo ser possível a comparação entre diferentes métodos.*

1. Introdução

No que tange aos avanços tecnológicos vistos nos últimos anos tais como a evolução do processamento computacional, a melhoria na ciência de dados e no reconhecimento de padrões, juntamente da grande disponibilização de informações, geram um desafio para aqueles que trabalham em áreas relacionadas. O reconhecimento de padrões é uma ferramenta de suma importância interdisciplinar, de forma que, máquinas posssam observar o ambiente, aprender a distinguir padrões de interesse e tomar decisões confiáveis e razoáveis sobre as categorias desses padrões [de Sousa 2016]. Um projeto de reconhecimento de imagem normalmente envolve [de Sá 2000]: (i) extração de característica dos objetos que devem ser classificados; (ii) a seleção de características mais discriminativas e (iii) a construção de um classificador. Um dos conceitos dessa esfera é o reconhecimento de escrita manual.

O reconhecimento de escrita manual é uma aplicação de software de grande demanda na atualidade. O reconhecimento de assianturas escritas à mão a partir de documentos digitalizados, informações monetárias escritas em cheques [Gonçalves 2017], além do reconhecimento de cartas, reuniões e atas que nos auxiliam a verificar o passado, como as cartas de Allan Kardec.

Para a realização dessa tarefa, utiliza-se algoritmos de classficação que podem ser definidos como algoritmos de aprendizagem supervisionada onde o objetivo é prever uma classe associada com uma variável de entrada contendo determinados atributos [Éliton Fontana 2020]. A importância dos algoritmos de classificação na ciência de dados e aprendizado de máquina é significativa, trabalhos como: (i) Spontaneous Handwriting Recognition and Classification [Toselli et al. 2004] e (ii) A Review of Various Handwriting Recognition Methos [Rosyda and Purboyo 2018], utilizam esses artíficios para explicar os propósitos dos trabalhos de reconhecimento de escritas.

Dada a notoriedade da área, o Professor Marco Cristo da Universidade Federal do Amazonas (UFAM) propôs o problema de reconhecer 10 dígitos, 0 à 9, escritos a mão e digitalizados. Artefatos, tais como: (i) Weka, (ii) J48 e (iii) Naive Bayes, já foram utilizados para resolver o problema, portanto, este trabalho pretende verificar a ação de outros algoritmos, (iv) Decision Tree, (v) Random Forest (vi) CNN. O objetivo é mostrar como o tratamento da entrada pode ou não afetar o desempenho dos classificadores no reconhecimento de caracteres escritos à mão baseado nas métricas de desempenho. Nesse sentido, o seguinte artigo está dividido em seis seções. A primeira seção aborda a metodologia utilizada para a solução da proposta do trabalho, por meio da especificação dos arquivos e da explicação da rotina de código. Na segunda seção, será feita uma análise dos dados que serão utilizados como entrada para o nosso sistema. Na terceira e quarta seções, cada um dos classificadores serão detalhados e bem como seus resultados de forma a comparar as eficiências. A quinta seção fecha o trabalho, retomando todos os aspectos importantes do artigo.

2. Metodologia

Para o reconhecimento de padrões numéricos utilizou-se das bibliotecas da linguagem Python, dentre estas, se destaca a biblioteca *Scikit Learn*. Esta é uma biblioteca de machine learning de código aberto que suporta aprendizados supervisionados e não supervisionados. Ela também provê várias ferramentas para ajustes de modelo, preprocessamento de dados, seleção e avaliação de modelos, dentre outros [Scikit 2023]. O procedimento de avaliação percorre 4 etapas das quais serão discorridas nessa seção. Para a análise da escrita, utilizou-se a base de dados "mnist_avaliacao.csv". Essa base contém 2050 amostras, uma em cada linha. São ao todo 784 valores por linha que representam as cores, em escala de cinza, de uma matriz 28x28. O último elemento consiste em um número de 0 à 9 indicando qual a classe que pertence aquela matriz. Um exemplo do arquivo pode ser representado pela figura 1.

2.1. Classificação sem filtragem

A primeira etapa se trata da classificação utilizando uma base de dados e sem a utilização de filtros. A biblioteca tem consigo objetos relacionados a cada um dos classificadores que formam a base desse trabalho, desse modo, para essa primeira etapa criam-se *folds* para os dados relacionados, o que auxilia na estimativa mais robusta do desempenho, na detecção de overffiting e na utilização eficiente dos dados, posteriormente instancia-se a árvore e cria-se um objeto classificador passando por parâmetro a árvore criada. Para a criação, o seguinte conjunto de parâmetros são utilizados:

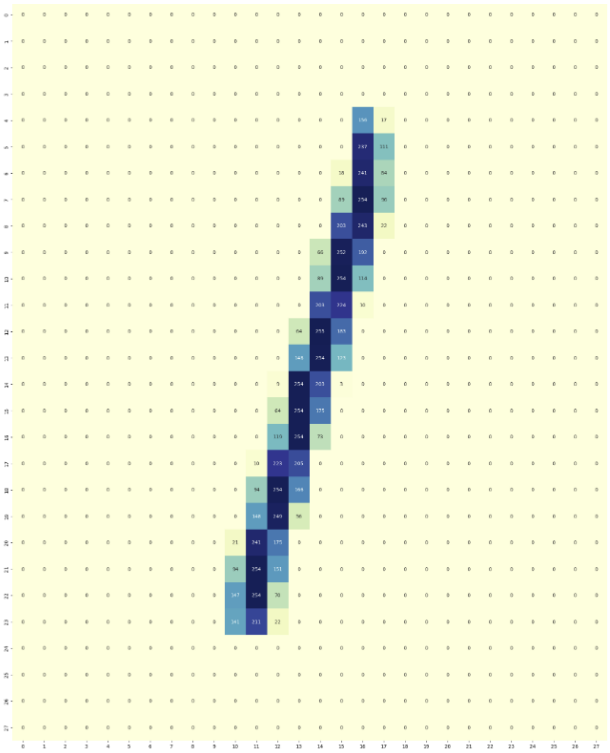

Figura 1. Representação do número 1; dados sem filtragem

Parâmetro	Valor
k	5
seed	1
folds para validação	3
threshold para validação	2

Tabela 1. Instância dos folds

Em seguida, realiza-se a avaliação utilizando a biblioteca *optuna* que, por meio do parâmetro *study*, instancia testes de otimização de modo que possam ser extraídos, os melhores *study* por *fold*. Para realizar a otimização da função objetivo que se trata da comparação entre os dados classificados e a classe que é necessária atingir, utiliza-se os seguintes parâmetros:

Parâmetro	Valor
trials	10
seed	1
startup trials	3

Tabela 2. Instância da otimização do optuna

Um *Trial* representa o processo de avaliação de uma função objetivo, o número de *trials* simboliza a quantidade de ensaios por cada teste. Para usar as árvores instânciadas o *optuna* se utiliza do algoritmo *TPESampler* (Estimador Parzen estruturado em árvore), um amostrador baseado em amostragem independente, ou

seja, determina valores de um único parâmetro sem considerar qualquer relação entre os parâmetros [Optuna 2018] Extraindo os resultados dos ensaios realizados pelo *optuna*, o resultado é apresentado na seção 4. Todas as etapas subsequêntes irão repetir a rotina descrita acima, porém para dados filtrados.

2.2. Classificação com filtragem

Nas etapas 2, 3 e 4, são formulados diferentes tipos de filtragem. Na etapa 2, o *infogain* é proposto como um filtro de entrada, portanto, para que isso seja possível, se faz necessário o uso do cálculo da entropia:

$$E(S) = \sum_{i=1}^{c} -\frac{feature}{total} \log_2(\frac{feature}{total}) \tag{1}$$

Por meio do cálculo da entropia, infere-se a expressividade das linhas e colunas da matrizes. O processo de ganho de informação sugere que as primeira 4 linhas, as primeiras 4 colunas, as últimas 4 linhas e as últimas 4 colunas não são tão expressivas na determinação do dígito. Isso acontece porque as amostras coletadas não fizeram bom uso do espaço disponível tornando os pixels dos cantos irrelevantes. Poderia ser feito um corte mais interessante também arredondando as bordas da matriz, mas isso implicaria em uma matriz deformada. Então essa ideia não será considerada. O plano consiste em apenas fazer as eliminações das linhas e colunas sugeridas, ocorre uma redução de 384 atributos. Antes desse filtro, o total era de 784 (28x28). Com a eliminação, esse número cai para 400 (20x20). É uma redução aproximada de 49% dos atributos relevantes e ainda mantém a matriz quadrada.

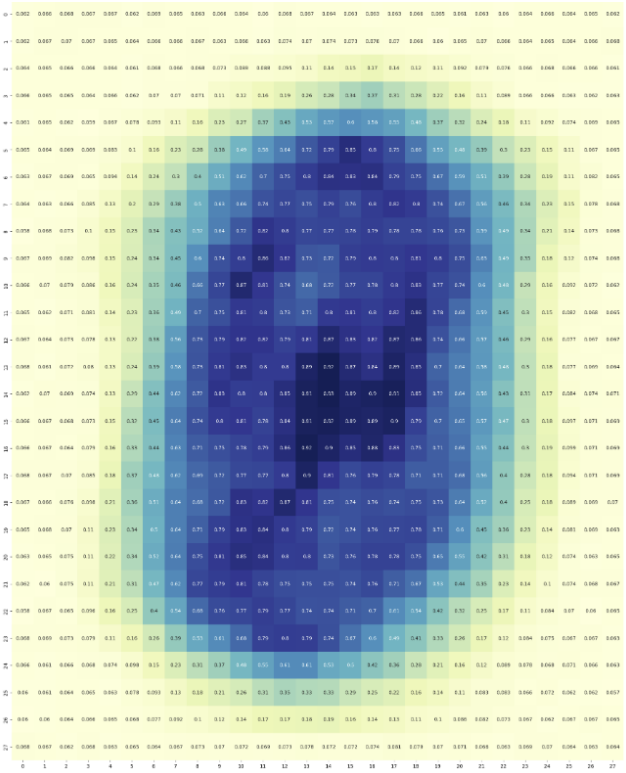

Figura 2. Ganho de informação

Na terceira etapa, uma das *features* é correlacionada como filtro de entrada, ou seja, escolhe-se um número entre 1 e 2050 e desenha-se uma nova matriz apenas com os valores de referência passados. Essa é uma maneira eficiente de tornar os dados binários, isto é, todo número menor que 128 é tratado como 0, caso contrário é tratado como 1. O resultado disso está em tratar as bordas que eram mais claras.

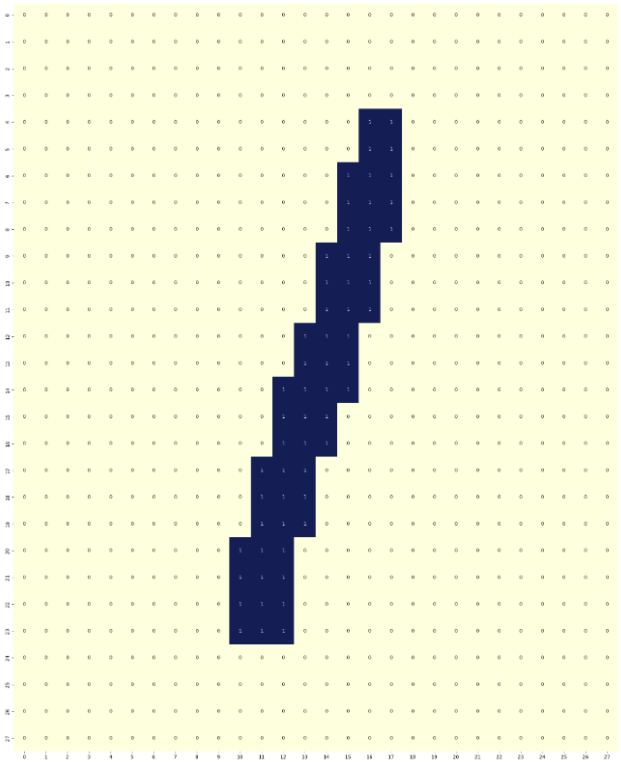

Figura 3. Representação binária

Como última etapa, realiza-se a mistura dos dois filtros de modo a ter os benefícios de ambos. Os resultados são apresentados na seção 4 deste artigo.

3. Classsificadores

Problemas de classificação ocorrem com frequência, talvez até mais do que problemas de regressão. Alguns exemplos incluem:

1. Uma pessoa chega ao pronto-socorro com um conjunto de sintomas que podem ser atribuídos a uma de três condições médicas. Qual das três condições o indivíduo possui?
2. Um serviço bancário on-line deve ser capaz de determinar se uma transação realizada no site é fraudulenta ou não, com base no endereço IP do usuário, histórico de transações anteriores e assim por diante.
3. Com base nos dados da sequência de ADN de vários pacientes com e sem uma determinada doença, um biólogo gostaria de descobrir quais as mutações do ADN que são deletérias (causadoras de doenças) e quais as que não o são.

Nos cenários de classificação temos um conjunto de observações de treinamento (x_1, y_1),...,(x_n, y_n) que podemos usar para construir um classificador. Queremos que nosso classificador tenha um bom desempenho não apenas nos dados

de treinamento, mas também nas observações de teste que não foram usadas para treinar o classificador. [James et al. 2021]

Considerando a proposta do trabalho focada no reconhecimento de imagens e dos conhecimentos do grupo a respeito da disciplina, os métodos selecionados para o treinamento e classificação das imagens do banco de dados MNIST foram os modelos de Árvores de Decisão; seu aprimoramento, a Random Forest; e o modelo com grande foco em reconhecimento de imagens, as Redes Neurais Convolucionais (CNNs).

3.1. Decision tree

Árvores de decisão são modelos de aprendizado de máquina que replicam a estrutura de decisão em forma de árvore, sendo modelos versáteis e interpretáveis que tomam decisões com base em testes hierárquicos em atributos, eficazes principalmente para problemas de classificação e regressão. No procedimento de análise implementado, será utilizado a árvore de decisão do tipo de classificação. [James et al. 2021]

O diferencial da árvore de classificação está no fato de que ela é usada para prever uma resposta qualitativa em vez de quantitativa, como ocorre em árvores de regressão. Para uma árvore de classificação, prevemos que cada observação pertence à classe mais comum entre as observações de treinamento na região à qual ela pertence. Na interpretação dos resultados de uma árvore de classificação, frequentemente estamos interessados não apenas na previsão de classe correspondente a uma região de nó terminal específica, mas também nas proporções de classe entre as observações de treinamento que se enquadram nessa região. [James et al. 2021]

As árvores de regressão e classificação têm um sabor muito diferente das abordagens mais clássicas de regressão e classificação. Em particular, a regressão linear assume um modelo da forma

$$f(x) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j \tag{2}$$

enquanto as árvores de regressão assumem um modelo da forma

$$f(x) = \sum_{m=1}^{M} c_m . 1_{(X \in R_m)} \tag{3}$$

onde R1,...,RM representam uma partição do espaço de recursos.

Os desempenhos relativos das abordagens clássica e baseada em árvore podem ser avaliados estimando o erro de teste, usando validação cruzada ou abordagem de conjunto de validação. No entanto, outras considerações além do simples erro de teste podem ser consideradas na seleção de um método de aprendizagem estatística; por exemplo, em certos ambientes, a previsão usando uma árvore pode ser preferida por uma questão de interpretabilidade e visualização. [James et al. 2021]

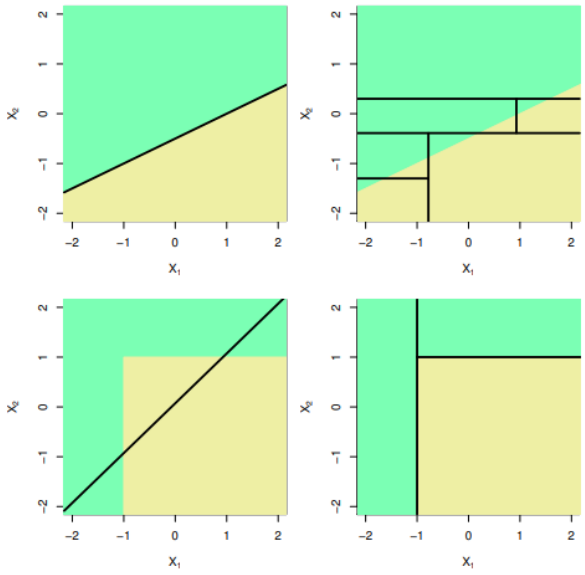

Figura 4. Limite de decisão linear

No exemplo acima, temos:

- Linha superior: um exemplo de classificação bidimensional em que o verdadeiro limite de decisão é linear e é indicado pelas regiões sombreadas. Uma abordagem clássica que assume um limite linear (esquerda) terá desempenho superior a uma árvore de decisão que realiza divisões paralelas aos eixos (direita).
- Linha Inferior: Aqui o verdadeiro limite de decisão não é linear. Aqui, um modelo linear é incapaz de capturar o verdadeiro limite de decisão (esquerda), enquanto uma árvore de decisão é bem-sucedida (direita).

As árvores de decisão para regressão e classificação têm várias vantagens sobre as abordagens mais clássicas, como:

- São muito fáceis de explicar às pessoas, sendo ainda mais fáceis de se explicar do que regressão linear;
- Podem ser exibidas graficamente e são facilmente interpretadas mesmo por um não especialista;
- As árvores podem lidar facilmente com preditores qualitativos sem a necessidade de criar variáveis fictícias.

Apesar disso, também apresentam certas desvantagens:

- Geralmente não têm o mesmo nível de precisão preditiva que algumas das outras abordagens de regressão e classificação vistas neste livro;
- Podem ser muito pouco robustas. Em outras palavras, uma pequena alteração nos dados pode causar uma grande alteração na árvore estimada final.

3.2. Random forest

Um *ensemble method* é uma abordagem que combina muitos modelos simples de *building blocks* para obter um modelo único e potencialmente muito poderoso. Esses

modelos simples de blocos de construção são às vezes conhecidos como *weak learners*, pois podem levar a previsões medíocres por si próprios. *Random Forest* é um *ensemble method* para o qual o bloco de construção simples é uma regressão ou uma árvore de classificação. [James et al. 2021]

As florestas aleatórias proporcionam uma melhoria em relação às *bagged trees* por meio de pequenos ajustes que decorrelacionam as árvores. Assim como no bagging, construímos uma série de árvores de decisão em amostras de treinamento *bootstraped*. Mas ao construir essas árvores de decisão, cada vez que uma divisão em uma árvore é considerada, uma amostra aleatória de m preditores é escolhida como candidata à divisão do conjunto completo de p preditores. [James et al. 2021]

A divisão pode usar apenas um desses m preditores. Uma nova amostra de m preditores é obtida em cada divisão e normalmente escolhemos

$$m \approx \sqrt{p}$$

ou seja, o número de preditores considerados em cada divisão é aproximadamente igual à raiz quadrada do número total de preditores. [James et al. 2021]

Em outras palavras, ao construir uma floresta aleatória, a cada divisão da árvore, o algoritmo não pode sequer considerar a maioria dos preditores disponíveis. A principal diferença entre *bagging* e florestas aleatórias é a escolha do tamanho m do subconjunto do preditor. Por exemplo, se uma floresta aleatória for construída usando

$$m = p$$

então isso equivale simplesmente ao *bagging*. [James et al. 2021]

3.3. CNN

As redes neurais se recuperaram por volta de 2010, com grandes sucessos na classificação de imagens. Naquela época, enormes bancos de dados de imagens rotuladas estavam sendo acumulados, com um número cada vez maior de classes, como a CIFAR100. [James et al. 2021]

As Redes Neurais Convolucionais (CNNs), ou *Convolutional Neural Networks* em inglês, são um tipo especializado de arquitetura de redes neurais projetadas para processar dados com estrutura de grade, como imagens. As CNNs imitam, até certo ponto, como os humanos classificam as imagens, reconhecendo características ou padrões específicos em qualquer lugar da imagem que distinguem cada classe específica de objeto. [James et al. 2021]

A rede primeiro identifica características de baixo nível na imagem de entrada, como pequenas bordas, manchas de cor e similares. Esses recursos de baixo nível são então combinados para formar recursos de nível superior, como partes dos ouvidos, olhos e assim por diante. Eventualmente, a presença ou ausência destas características de nível superior contribui para a probabilidade de qualquer classe de saída. [James et al. 2021]

Para conseguir realizar esse processo, demonstrado pela figura 5, combina dois tipos especializados de camadas ocultas, chamadas camadas de convolução e

camadas de *pooling*. As camadas de convolução procuram instâncias de pequenos padrões na imagem, enquanto as camadas de *pooling* reduzem a resolução deles para selecionar um subconjunto proeminente. Para alcançar resultados de última geração, as arquiteturas de redes neurais contemporâneas fazem uso de muitas camadas de convolução e *pooling*. [James et al. 2021]

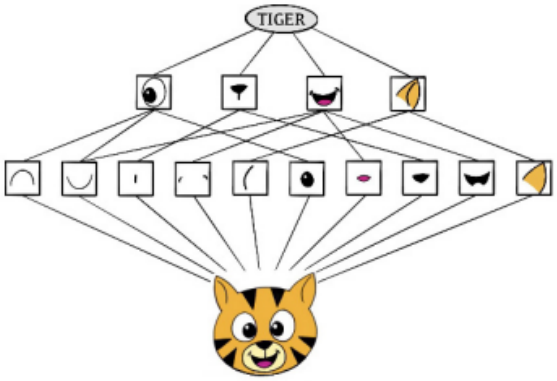

Figura 5. Esquema mostrando como uma rede neural convolucional classifica a imagem de um tigre. A rede capta a imagem e identifica características locais. Em seguida, combina os recursos locais para criar recursos compostos, que neste exemplo incluem olhos e ouvidos. Esses recursos compostos são usados para gerar o rótulo "tigre".

3.4. Camadas de convolução

Uma camada de convolução, representada pela figura 6, é composta por um grande número de filtros de convolução, cada um dos quais é um modelo que determina se uma característica local específica está presente em uma imagem. Um filtro de convolução depende de uma operação muito simples, chamada convolução, que basicamente equivale a multiplicar repetidamente os elementos da matriz e depois adicionar os resultados. [James et al. 2021]

3.5. Camadas de *Pooling*

Uma camada de *pooling* fornece uma maneira de condensar uma imagem grande em uma imagem de resumo menor. Embora existam várias maneiras possíveis de realizar o *pooling*, a operação de *pooling* máximo resume cada bloco de pixels 2×2 não sobreposto em uma imagem usando o valor máximo no bloco. Isto reduz o tamanho da imagem por um fator de dois em cada direção e também fornece alguma invariância de localização: ou seja, enquanto houver um valor grande em um dos quatro pixels do bloco, todo o bloco é registrado como um valor grande na imagem reduzida. [James et al. 2021]

Aqui está um exemplo simples de *pooling* máximo:

$$\text{Max pool} \begin{bmatrix} 1 & 2 & 5 & 3 \\ 3 & 0 & 1 & 2 \\ 2 & 1 & 3 & 4 \\ 1 & 1 & 2 & 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 3 & 5 \\ 2 & 4 \end{bmatrix} \tag{4}$$

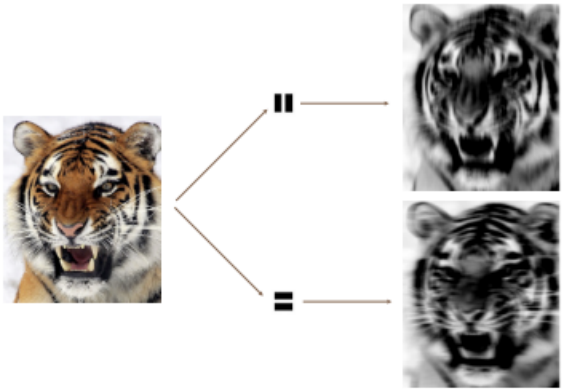

Figura 6. Os filtros de convolução encontram características locais em uma imagem, como bordas e formas pequenas. Começamos com a imagem do tigre mostrada à esquerda e aplicamos os dois pequenos filtros de convolução no meio. As imagens convolvidas destacam áreas da imagem original onde são encontrados detalhes semelhantes aos filtros. Especificamente, a imagem convoluta superior destaca as listras verticais do tigre, enquanto a imagem convoluta inferior destaca as listras horizontais do tigre. Podemos pensar na imagem original como a camada de entrada em uma rede neural convolucional e nas imagens convoluídas como as unidades na primeira camada oculta

4. Resultados e Conclusão

O problema do reconhecimento óptico de caracteres vem sofrendo muito destaque na sociedade atual, e com a ajuda da inteligência artificial, a busca por uma solução boa e satisfatória para esse problema tem sido cada vez mais procurada.

A proposta deste trabalho tem como base explorar três diferentes modelos de inteligência artificial, e estudar sua efetividade no reconhecimento óptico de dígitos de 0 a 9. Ao longo deste trabalho, os modelos e suas utilizações, assim como o tratamento dos dados utilizados por eles foram discutidos e detalhados separadamente. Por fim, esta seção tem como objetivo levantar uma comparação objetiva entre os três modelos considerados neste trabalho, fazendo isso a partir da análise dos resultados de cada modelo em relação as métricas consideradas, dando um enfoque maior para a acurácia e Macro F1.

O primeiro modelo apresentado neste trabalho foi o modelo de Árvore de D. Nesse sentido, para execução em 5 folds no contexto da árvore de decisão, foi obtida uma melhora geral nos indicadores Macro F1 e acurácia de acordo com a tabela 5. Com a inclusão dos tratamentos nos dados de entrada houve um aumento expressivo nos resultados tanto de acurácia quanto de Macro F1. Esse comprtamento apresentado pelo modelo é algo esperado, uma vez que estamos tratando de um modelo mais simplificado.

Bem como o experimento para árvore de decisão, o experimento para *random forest* também apresenta variações de resultado quando se aplica os processos de filtragem. Como a intenção desse classificador é melhorar o procedimento anterior, de fato, os valores se apresentam melhores e mais bem definidos como apresenta as tabelas 3 e 4.

Além da melhoria perante as árvores de decisão, as filtragem aprimoram minimamente os resultados sendo necessário utilizar-se de processos estatísticos tais como, p-valor, testes: (i) T, (ii) Chi-Quadrado, dentre outros, para verificar, de modo acertivo, essa melhoria.

Por fim, o algoritmo CNN apresentou comportamento um pouco diferente dos outros dois. A precisão e Macro F1 foram bem próximas de 1 para todos os casos. Acredita-se que seria necessário fazer mais folds com uma amostra maior para talvez assim trazer a tona alguma diferença expressiva nos valores obtidos. Como não foi o caso, não foi possível chegar em uma afirmação.

Com todos os resultados estabelecidos, esse artigo realiza uma comparação final com base na tabela 6.

Referências

de Sousa, R. M. (2016). Inteligência computacional aplicada ao controle externo da administração pública: aplicações da classificação de padrões utilizando redes neurais artificiais. *Revista TCU*.

de Sá, M. (2000). Reconhecimento de padrões. *Universidade do Porto*.

Gonçalves, L. (2017). Reconhecimento de escrita manual com redes neurais convolucionais. *Medium*.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2021). *An Introduction to Statistical Learning With Applications in R*. Springer, 2nd edition.

Optuna (2018). optuna.samplers.basesample. *Optuna*.

Rosyda, S. S. and Purboyo, T. W. (2018). A review of various handwriting recognition methods. *International Journal of Applied Engineering Research*.

Scikit (2023). Getting started. *Scikit Learn*.

Toselli, A. H., Juan, A., and Vidal, E. (2004). Spontaneous handwriting recognition and classification. *IEEE*.

Éliton Fontana (2020). Introdução aos algoritmos de aprendizagem supervisionada. *Universidade Federal do Paraná - UFPR*.

Modelo	Números	Métricas		
		F1	Precision	Recall
Sem filtro	0	0.923	0.9	0.947
	1	0.886	0.83	0.951
	2	0.654	0.694	0.618
	3	0.667	0.65	0.684
	4	0.794	0.771	0.818
	5	0.603	0.594	0.613
	6	0.791	0.773	0.81
	7	0.813	0.841	0.787
	8	0.675	0.7	0.651
	9	0.709	0.737	0.683
Ganho de informação	0	0.96	0.973	0.947
	1	0.864	0.809	0.927
	2	0.796	0.854	0.745
	3	0.581	0.521	0.658
	4	0.812	0.778	0.848
	5	0.656	0.667	0.645
	6	0.851	0.822	0.881
	7	0.773	0.829	0.723
	8	0.675	0.765	0.605
	9	0.667	0.651	0.683
Binário	0	0.878	0.818	0.947
	1	0.907	0.867	0.951
	2	0.694	0.791	0.618
	3	0.568	0.5	0.658
	4	0.687	0.676	0.697
	5	0.625	0.606	0.645
	6	0.81	0.81	0.81
	7	0.8	0.837	0.766
	8	0.667	0.683	0.651
	9	0.64	0.706	0.585
Mix	0	0.947	0.947	0.947
	1	0.876	0.812	0.951
	2	0.731	0.776	0.691
	3	0.636	0.56	0.737
	4	0.716	0.706	0.727
	5	0.667	0.656	0.677
	6	0.85	0.895	0.81
	7	0.822	0.86	0.787
	8	0.683	0.718	0.651
	9	0.759	0.789	0.732

Tabela 3. Tabela com métricas para Random Forest (Fold 1; O número 3 está em vermelho afim de demonstrar como a filtragem pode piorar o resultado)

Modelo	Fold	Métricas	
		Macro F1	Accuracy
Sem filtro	1	0.7514	0.7530
	2	0.7908	0.7921
	3	0.7423	0.7457
	4	0.7065	0.7163
	5	0.6843	0.6973
Ganho de informação	1	0.7633	0.7652
	2	0.7865	0.7921
	3	0.8040	0.8068
	4	0.7373	0.7506
	5	0.6800	0.6876
Binário	1	0.7274	0.7310
	2	0.7808	0.7823
	3	0.7488	0.7555
	4	0.7893	0.7970
	5	0.7574	0.7627
Mix	1	0.7688	0.7701
	2	0.7949	0.7970
	3	0.7949	0.7970
	4	0.7827	0.7897
	5	0.7122	0.7167

Tabela 4. Macro F1 e Acurácia por fold do modelo de Random Forest

Modelo	Fold	Métricas	
		Macro F1	Accuracy
Sem filtro	1	0.6523	0.6552
	2	0.7050	0.7090
	3	0.6725	0.6748
	4	0.7075	0.7163
	5	0.7125	0.7191
Ganho de informação	1	0.6509	0.6552
	2	0.6987	0.7017
	3	0.6809	0.6845
	4	0.6908	0.6992
	5	0.6903	0.6997
Binário	1	0.7157	0.7188
	2	0.7412	0.7457
	3	0.7135	0.7139
	4	0.7106	0.7139
	5	0.7069	0.7167
Mix	1	0.7422	0.7457
	2	0.7354	0.7359
	3	0.7214	0.7237
	4	0.7175	0.7188
	5	0.7111	0.7167

Tabela 5. Macro F1 e Acurácia por fold do modelo de Decision Tree

Classificador	Modelo	Fold	Accuracy
Decision tree	Mix	1	0.7457
		2	0.7359
		3	0.7237
		4	0.7188
		5	0.7167
Random forest	Mix	1	0.7701
		2	0.7970
		3	0.7970
		4	0.7897
		5	0.7167
CNN	Dropout		0.9984

Tabela 6. Comparativo entre todos os valores de acurácia (Em vermelho está o valor máximo, além disso, escolheu-se o Mix e o Dropout por apresentarem melhores resultados gerais)