Universidade Federal de São Carlos
Centro de Ciências Exatas e de Tecnologia
Departamento de Estatística

Classificação de caracteres japoneses cursivos utilizando modelos de aprendizado profundo

Fernando Borges

Trabalho de Conclusão de Curso

Universidade Federal de São Carlos
Centro de Ciências Exatas e de Tecnologia
Departamento de Estatística

Classificação de caracteres japoneses cursivos utilizando modelos de aprendizado profundo

Fernando Borges
Orientador: Prof. Dr. Thiago Rodrigo Ramos

Trabalho de Conclusão de Curso apresentado como parte dos requisitos para obtenção do título de Bacharel em Estatística.

São Carlos
Julho de 2025

Federal University of São Carlos
Exact and Technology Sciences Center
Department of Statistics

Cursive japanese characters classification using deep learning models

Fernando Borges
Advisor: Prof. Dr. Thiago Rodrigo Ramos

Bachelors dissertation submitted to the Department of Statistics, Federal University of São Carlos - DEs-UFSCar, in partial fulfillment of the requirements for the degree of Bachelor in Statistics.

São Carlos
July 2025

Fernando Borges

Classificação de caracteres japoneses cursivos utilizando modelos de aprendizado profundo

Este exemplar corresponde à redação final do trabalho de conclusão de curso devidamente corrigido e defendido por Fernando Borges e aprovado pela banca examinadora.

Aprovado em 01 de julho de 2025.

Banca Examinadora:

- Prof. Dr. Thiago Rodrigo Ramos
- Prof. Dr. Rafael Izbicki
- Profª. Drª. Andressa Cerqueira

Resumo

Com os recentes avanços na área de aprendizado de máquina, a classificação de imagens se tornou essencial em diversos ramos da ciência. Na linguística, as tarefas de decifração, transcrição e preservação de documentos antigos estão sendo beneficiadas pela aplicação de modelos de aprendizado profundo. O estudo de caracteres japoneses cursivos, por exemplo, envolve um tipo de escrita que se distingue muito de sua versão moderna, de tal forma que apenas uma pequena parte dos falantes da língua japonesa é capaz de compreendê-la fluentemente. Além disso, devido à forma como o silabário japonês evoluiu ao longo de sua história, a aplicação de métodos de aprendizado profundo é apropriada para a tarefa de classificação dessas escrituras japonesas antigas, tendo em vista que existem dezenas de variantes para cada caractere.

Assim, este trabalho teve como objetivo desenvolver um modelo de classificação que fosse capaz de predizer caracteres japoneses cursivos em escrituras antigas. Para isso, foram ajustados quatro métodos tradicionais de classificação – regressão logística, *k-nearest neighbors* (Cover e Hart (1967)), *support vector machines* (Cortes e Vapnik (1995)) e *boosting* (Freund e Schapire (1997)) – e três arquiteturas de um método de aprendizado profundo – redes neurais convolucionais (LeCun *et al.* (1989)) – em conjuntos de dados relacionados a caracteres cursivos manuscritos.

Como resultado final, o modelo que obteve os melhores resultados foi uma rede neural convolucional, com acurácia de 92.9% na classificação das 49 classes que compõem o silabário *hiragana*. Além disso, foi desenvolvido um modelo U-Net capaz de extrair caracteres de novas páginas de textos manuscritos. No entanto, ao aplicar o classificador ajustado em dados reais, a acurácia caiu para 69.1%, o que evidencia os desafios associados à variabilidade dos estilos de escrita e à presença de ruído em imagens digitalizadas.

Palavras-chave: *aprendizado profundo, classificação, japonês, redes neurais convolucionais, segmentação de imagens.*

Abstract

With recent advances in machine learning, image classification has become essential in various fields of science. In linguistics, tasks such as deciphering, transcribing, and preserving ancient documents are benefited from the application of deep learning models. The study of cursive Japanese characters, for example, involves a type of writing that differs significantly from its modern version, to the extent that only a small portion of Japanese speakers can understand it fluently. Furthermore, due to the way the Japanese syllabary has evolved throughout history, the application of deep learning methods is appropriate for the task of classifying these ancient Japanese writings, given that there are dozens of variants for each character.

Thus, this work aimed to develop a classification model capable of predicting cursive Japanese characters in ancient scripts. To achieve this, four traditional classification methods – logistic regression, k-nearest neighbors (Cover e Hart (1967)), support vector machines (Cortes e Vapnik (1995)) and boosting (Freund e Schapire (1997)) – were adjusted, as well as three architectures of a deep learning method – convolutional neural networks (LeCun *et al.* (1989)) – using datasets related to handwritten cursive characters.

As a final result, the model that achieved the best performance was a convolutional neural network, with an accuracy of 92.9% in classifying the 49 classes that make up the *hiragana* syllabary. In addition, a U-Net model was developed to extract characters from new pages of handwritten texts. However, when applying the trained classifier to real data, the accuracy dropped to 69.1%, highlighting the challenges associated with the variability of writing styles and the presence of noise in scanned images.

Keywords: *classification, convolutional neural networks, deep learning, image segmentation, Japanese.*

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Com o crescimento acelerado da área de aprendizado de máquina nos últimos anos e por meio da vasta aplicabilidade que os algoritmos modernos proporcionam, a classificação de imagens tem se tornado uma tarefa cada vez mais relevante em todos os campos da ciência (Li *et al.*, 2021; Khan *et al.*, 2020). Na linguística (Sommerschield *et al.*, 2023), por exemplo, a análise de documentos antigos envolve diversos aspectos intrínsecos a cada idioma que podem diferir de sua versão moderna, como mudanças na forma de escrita e na ortografia. Nesse contexto, os métodos de aprendizado de máquina se tornam ferramentas que podem auxiliar e acelerar o processo de registro desses documentos por meio da classificação de grafemas, isto é, a unidade mínima de cada sistema de escrita, como letras ou caracteres.

No âmbito de divergências na forma de escrita dentro de um mesmo idioma, a língua japonesa se apresenta como um caso interessante. Historicamente, a predominância da forma atual de escrita é uma consequência das políticas de modernização desde o início do período Meiji (1868-1912), em que uma série de reformas linguísticas foram estabelecidas na tentativa de padronizar a língua escrita com a falada. Como uma das medidas de simplificação do idioma, a escrita cursiva, atualmente chamada de *kuzushiji*, teve seu ensino removido do currículo das escolas e, com isso, grande parte da população japonesa atual possui dificuldades em compreender escrituras antigas (Twine, 1983; Takashiro, 2013).

1.1 O sistema de escrita japonês

O sistema de escrita japonês é dividido em três tipos de caracteres: *hiragana*, *katakana* e *kanji*. Os dois primeiros, assim como o conceito de alfabeto em outros idiomas, representam os sons silábicos que são utilizados na língua falada. Enquanto o *hiragana* é usado para flexionar verbos, conectivos gramaticais e outras palavras nativas do japonês, o *katakana* é usualmente usado em palavras emprestadas de outros idiomas ou para efeitos de ênfase, como onomatopeias. O *kanji*, por sua vez, é um sistema de caracteres emprestados da língua chinesa cuja função principal é, além dos sons silábicos, representar alguma ideia ou significado. Esse sistema integra uma parte fundamental da escrita japonesa e é utilizado em substantivos, verbos e adjetivos, conjuntamente com o *hiragana*.

O surgimento dos silabários *hiragana* e *katakana* ocorreu por meio de um processo gradual de simplificação e abreviação do *man'yōgana* desde os últimos anos do período Nara (710-794). O *man'yōgana*, uma forma antiga de escrita que utilizava caracteres chineses para representar sons do japonês sem o significado intrínseco ao ideograma, permitia que um mesmo som pudesse ser representado por caracteres chineses distintos, isto é, não existia um padrão em como representar os sons do japonês e a escolha dos caracteres ficava a critério do próprio autor. Enquanto o *hiragana* surgiu da forma cursiva do *man'yōgana*, o *katakana* faz uso de partes do *man'yōgana* para servir como uma forma de escrita taquigráfica e, ainda assim, o *hiragana* e o *katakana* não foram criados como substitutos imediatos ao *man'yōgana* – as três formas de escrita coexistiram por séculos (Frellesvig (2010)).

Apesar da padronização atual da escrita por esses dois silabários, houve diversas formas variantes que foram utilizadas ao mesmo tempo para representar os mesmos sons do *hiragana* e *katakana*. Como ilustrado na Figura 1.1, essas variantes são atualmente chamadas de *hentaigana* e caíram em desuso após reformas ortográficas que, desde 1900, restringiram o *hiragana* e o *katakana* a um único caractere por som representado.

1.2 Objetivos e organização do trabalho

O objetivo principal deste trabalho consiste em analisar a performance de modelos de classificação treinados com conjuntos de dados relacionados a caracteres japoneses cursivos – mais especificamente, escrituras cursivas em *hiragana* e suas variantes históricas. Uma análise comparativa será realizada utilizando modelos profundos (redes neurais con-

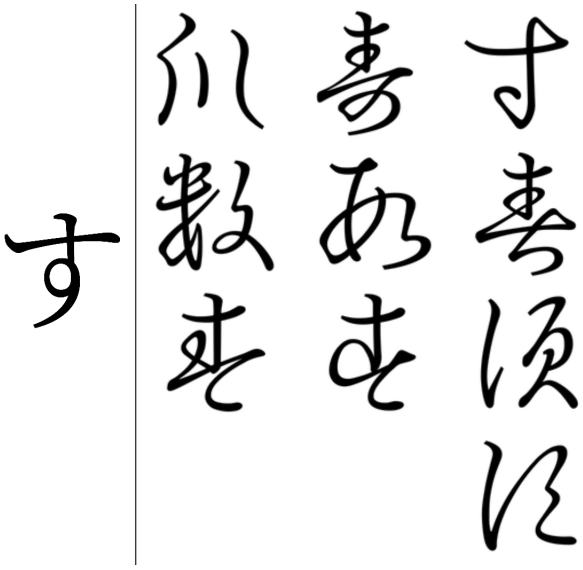


Figura 1.1: *Hentaigana* para o caractere “す”. A primeira coluna contém sua representação moderna em *hiragana*.

volucionais (CNNs) (LeCun *et al.*, 1989) e redes neurais residuais (ResNets) (He *et al.*, 2015)) e modelos não profundos (regressão logística, *k-nearest neighbors* (KNNs) (Cover e Hart, 1967), *support vector machines* (SVMs) (Cortes e Vapnik, 1995) e *boosting* (Freund e Schapire, 1997)) ajustados nesses conjuntos de dados. Com base no melhor modelo treinado, será possível também desenvolver um algoritmo de detecção e classificação instantânea de caracteres em fotos de textos manuscritos em japonês, como ilustrado na Figura 1.2. Dessa forma, o trabalho fornecerá uma análise aprofundada da performance desses métodos na tarefa de classificação dos caracteres cursivos e um algoritmo capaz de realizar essas classificações em novos textos japoneses manuscritos, com o intuito de facilitar o acesso à literatura clássica japonesa para nativos ou estudantes do idioma.

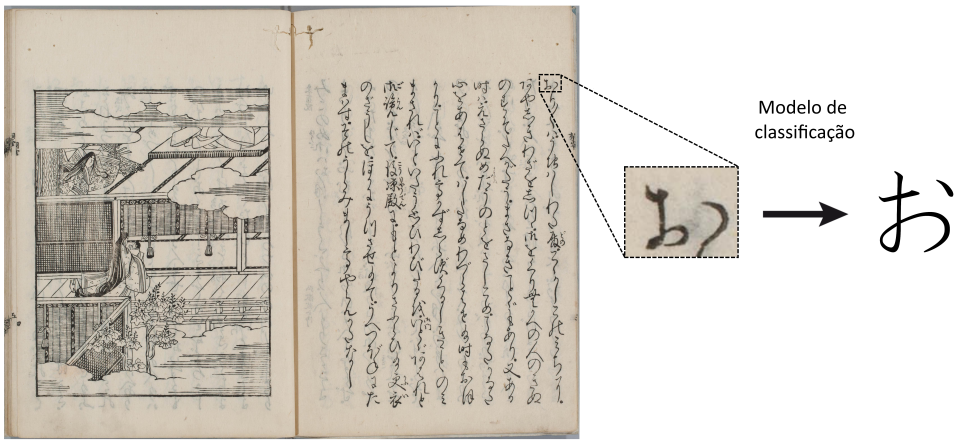


Figura 1.2: Esquema ilustrando o processo de detecção e classificação do caractere “お” em *Genjimonogatari*. Fonte: Shikibu (s.d.).

O trabalho está organizado como segue. O Capítulo 1 oferece uma breve introdução

e descrição dos principais objetivos elencados que serão abordados ao longo do trabalho. No Capítulo 2, é feita uma breve revisão de alguns conceitos de aprendizado de máquina. No Capítulo 3, os métodos de classificação tradicionais a serem utilizados na comparação com os de aprendizado profundo são revisados. No Capítulo 4, os métodos de aprendizado profundo são abordados com um maior detalhamento e, por fim, o Capítulo 5 contém a aplicação dos métodos revisados nos dados de caracteres japoneses cursivos. Além disso, o Capítulo 6 contém as considerações finais sobre o trabalho e algumas sugestões de possíveis extensões para projetos futuros relacionados ao assunto.

Capítulo 2

Aprendizado supervisionado

A maior parte dos métodos estatísticos existentes pode ser dividida em duas possíveis categorias: algoritmos de aprendizado supervisionado e não supervisionado. Problemas que envolvem a necessidade de realizar previsões de uma variável resposta Y por meio de variáveis preditoras $\boldsymbol{X} = (X_1, \ldots, X_p)$ utilizam algoritmos de aprendizado supervisionado, abrangendo métodos simples, como uma regressão linear, até abordagens mais complexas, como redes neurais artificiais. Em contrapartida, métodos de aprendizado não supervisionado são utilizados em situações nas quais não há uma variável resposta definida ou observada, isto é, deseja-se entender um comportamento entre as observações ou entre as variáveis com base nas características intrínsecas ao conjunto de dados. Essas técnicas são amplamente utilizadas em tarefas de reconhecimento de padrões e agrupamento de dados, já que não existe a necessidade de rótulos predefinidos.

Dentro do escopo de aprendizado supervisionado, os métodos são generalizados a depender do tipo de variável resposta a ser prevista – em geral, regressão para problemas envolvendo variáveis respostas quantitativas e classificação para variáveis respostas qualitativas, em que o objetivo é atribuir à nova observação uma das possíveis classes predefinidas.

2.1 Conceitos de aprendizado de máquina

2.1.1 Função de risco

Como não existe um método de regressão ou de classificação universalmente superior aos demais (Shalev-Shwartz e Ben-David (2014)), torna-se necessário uma medida de

performance que sirva de critério para a comparação de modelos. Em um problema de regressão no qual há interesse em predizer uma variável quantitativa, um modelo com bom poder preditivo é aquele cujos valores preditos estão muito próximos aos valores observados no conjunto de dados. Sendo assim, a medida usualmente utilizada é o risco quadrático (Izbicki e dos Santos (2020)),

$$R(\hat{f}) := \mathbb{E}\left[(Y - \hat{f}(\boldsymbol{X}))^2\right],$$

em que $\hat{f}(\cdot)$ é o valor predito pelo modelo estimado a partir do conjunto de treinamento.

Da mesma forma, problemas de classificação requerem uma medida de desempenho para o modelo $\hat{f}$ estimado. Nesse contexto, porém, utiliza-se a função de perda $L(y, \hat{f}(\boldsymbol{x})) = \mathbb{I}\{y \neq \hat{f}(\boldsymbol{x})\}$ (Izbicki e dos Santos (2020)) tal que o risco seja dado por

$$R(\hat{f}) := \mathbb{E}\left[\mathbb{I}\{Y \neq \hat{f}(\boldsymbol{X})\}\right] = \mathbb{P}(Y \neq \hat{f}(\boldsymbol{X})),$$

em que $\mathbb{I}\{y \neq \hat{f}(\boldsymbol{x})\}$ é uma variável indicadora que assume valor 1 quando a predição difere do valor observado para a variável resposta e 0 caso contrário.

2.1.2 *Data Splitting*

Como comentado na Seção 2.1.1, é fundamental que as observações utilizadas no cálculo da função de risco não tenham sido utilizadas na estimação de $\hat{f}$, pois o modelo com melhor desempenho estará bem ajustado aos dados da amostra, mas com um baixo poder de generalização. Para evitar esse cenário conhecido como *overfitting*, o conjunto de dados original pode ser dividido em duas partes, treinamento e validação, tal que a primeira seja utilizada exclusivamente para o ajuste dos modelos e a segunda seja utilizada apenas para as avaliações de desempenho. Em geral, é definida uma porcentagem do conjunto de dados que será utilizada para compor o conjunto de treinamento e realizada uma amostragem aleatória para seleção dessas observações.

Semelhantemente, o conjunto de validação também pode ser dividido para evitar que o estimador do risco na validação seja otimista demais, dado que o modelo escolhido por esse critério pode sofrer de *overfitting* em situações nas quais muitos modelos estejam sendo comparados no mesmo conjunto de validação. Essa nova divisão gera um conjunto de teste, responsável por fornecer uma estimativa final do desempenho do modelo selecionado em

novos dados, uma abordagem que reflete melhor sua capacidade de generalização (Izbicki e dos Santos (2020)).

Assim, enquanto o conjunto de validação auxilia na escolha do modelo, o conjunto de teste é utilizado para avaliar seu desempenho real que, eventualmente, pode ser comparado com o desempenho de modelos ajustados por outros métodos. A divisão de uma amostra $(\boldsymbol{X}_1, Y_1), \ldots, (\boldsymbol{X}_n, Y_n)$ em conjuntos de treino, validação e teste pode ser representada da seguinte forma:

$$\underbrace{(\boldsymbol{X}_1, Y_1), (\boldsymbol{X}_2, Y_2), \ldots, (\boldsymbol{X}_s, Y_s)}_{\text{Treinamento (70\%)}}, \quad \underbrace{(\boldsymbol{X}_{s+1}, Y_{s+1}), \ldots, (\boldsymbol{X}_t, Y_t)}_{\text{Validação (20\%)}}, \quad \underbrace{(\boldsymbol{X}_{t+1}, Y_{t+1}), \ldots, (\boldsymbol{X}_n, Y_n)}_{\text{Teste (10\%)}},$$

em que as observações de cada conjunto são selecionadas aleatoriamente e a porcentagem da divisão é escolhida arbitrariamente.

2.1.3 Balanço entre viés e variância

O valor esperado do erro quadrático médio no conjunto de teste para uma nova observação $\boldsymbol{x}_0$ pode ser decomposto em três quantidades: (i) a variância intrínseca à variável resposta que não depende de $\hat{f}$, (ii) o quadrado do viés de $\hat{f}(\boldsymbol{x}_0)$ e (iii) a variância de $\hat{f}(\boldsymbol{x}_0)$. Assim,

$$\mathbb{E}\left[(Y - \hat{f}(\boldsymbol{X}))^2 \mid \boldsymbol{X} = \boldsymbol{x}_0\right] = \underbrace{\mathbb{V}\left[Y \mid \boldsymbol{X} = \boldsymbol{x}_0\right]}_{(i)} + \underbrace{[\text{Viés}(\hat{f}(\boldsymbol{x}_0)]^2}_{(ii)} + \underbrace{\mathbb{V}[\hat{f}(\boldsymbol{x}_0)]}_{(iii)}. \tag{2.1}$$

Pela Equação (2.1), percebe-se que, como (i) é um elemento que não depende de $\hat{f}$, métodos cuja estimação é muito sensível ao conjunto de treinamento, isto é, métodos que terão uma alta variação entre diferentes estimações de f, tendem a possuir um viés relativamente baixo quando comparados a outros métodos menos flexíveis. Dessa forma, pode-se dizer que há um *tradeoff* a ser considerado na obtenção de um modelo com baixo risco em que, quanto mais complexo (ou flexível) for um determinado modelo, menor será seu viés às custas de uma maior variância de $\hat{f}$.

2.1.4 Outros critérios de avaliação

Além da função de risco estimada por meio do conjunto de teste, existem outras métricas relevantes a serem consideradas na avaliação do desempenho de um modelo.

A matriz de confusão, em específico, é uma tabela que descreve, de forma resumida, a performance do modelo com relação às classes preditas e verdadeiras em quantidades de verdadeiros positivos (VP), falsos positivos (FP), verdadeiros negativos (VN) e falsos negativos (FN), conforme representado na Tabela 2.1.

Tabela 2.1: Matriz de confusão.

Valor predito	**Valor verdadeiro**	
	Y=0	Y=1
Y=0	VN	FN
Y=1	FP	VP

A partir da matriz de confusão, diversas métricas podem ser calculadas para avaliar o desempenho do modelo. As métricas utilizadas neste trabalho são:

- **Acurácia**: proporção de predições que foram classificadas corretamente em relação ao número total de predições. É calculada por $\frac{VP+VN}{VP+FP+VN+FN}$;
- **Precisão**: proporção de predições positivas que foram classificadas corretamente em relação ao número total de predições positivas. É calculada por $\frac{VP}{VP+FP}$.
- ***Recall***: proporção de predições positivas que foram classificadas corretamente em relação ao número total de valores positivos verdadeiros, isto é, a soma dos verdadeiros positivos e falsos negativos. Mede a capacidade do modelo em detectar, com sucesso, as observações que são realmente positivas. É calculada por $\frac{VP}{VP+FN}$.
- ***F1-Score***: média harmônica entre as métricas *recall* e precisão. É calculada por $2 \times \frac{\text{precisão} \times recall}{\text{precisão} + recall}$.

Capítulo 3

Métodos de classificação

Pela natureza dos dados a serem estudados, o trabalho seguirá como um problema de classificação. Nesta seção, será apresentada uma revisão sobre as metodologias relacionadas à classificação, conceitos que foram explorados em James *et al.* (2014) e que servirão de base para a análise comparativa a ser realizada.

3.1 Representação de imagens

Como os dados a serem utilizados neste trabalho são imagens, cada uma delas pode ser representada por uma matriz de tamanho $m \times m$, que, por sua vez, pode ser organizada em um vetor unidimensional de tamanho m^2. Os pixels, originalmente codificados em uma escala de cinza com valores inteiros de 0 a 255, foram padronizados para o intervalo contínuo $[0, 1]$, de tal forma que cada componente do vetor pudesse ser interpretado como uma variável aleatória contínua associada à intensidade de cor de um ponto específico da imagem.

Dessa forma, o espaço de entrada nesse problema é de alta dimensionalidade e cada imagem $\boldsymbol{X}_i \in \mathbb{R}^{m \times m}$ está associada a uma classe Y_i, que representa o caractere manuscrito correspondente. A representação vetorial será utilizada em todos os métodos de classificação estudados ao longo desta seção.

3.2 Regressão logística

A regressão logística é um método paramétrico que visa modelar uma variável resposta dicotômica Y por meio de funções lineares nas covariáveis explicativas $\boldsymbol{X} = (X_1, \ldots, X_p)$.

Para isso, estima-se a probabilidade de Y pertencer a uma das duas classes, isto é, a probabilidade de sucesso $\mathbb{P}(Y = 1 \mid \boldsymbol{X})$. Como a probabilidade real deve estar limitada entre 0 e 1, uma possível estratégia a ser empregada é a modelagem dessa probabilidade por meio de uma função que gere saídas entre esses valores, como a função logística $f(x) = \frac{e^x}{1+e^x}$ (James *et al.* (2014)). Assim,

$$\mathbb{P}(Y = 1 \mid \boldsymbol{X}, \boldsymbol{\beta}) = \frac{e^{\beta_0 + \sum_{i=1}^{p} \beta_i X_i}}{1 + e^{\beta_0 + \sum_{i=1}^{p} \beta_i X_i}}$$

é o modelo utilizado para regressão logística, em que β_0 é uma constante e β_i é um coeficiente de regressão, $i = 1, \ldots, p$.

Dado que esses coeficientes de regressão são desconhecidos, utiliza-se um conjunto de treinamento para estimá-los a partir do método da máxima verossimilhança (Hastie *et al.* (2009)). Assim, para uma amostra i.i.d. $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$,

$$\begin{aligned}
\mathcal{L}(y; \boldsymbol{x}, \boldsymbol{\beta}) &= \prod_{i=1}^{n} (\mathbb{P}(y_i = 1 \mid \boldsymbol{x}, \boldsymbol{\beta}))^{y_i} (1 - \mathbb{P}(y_i = 1 \mid \boldsymbol{x}, \boldsymbol{\beta}))^{1-y_i} \\
&= \prod_{i=1}^{n} \left(\frac{e^{\beta_0 + \sum_{i=1}^{p} \beta_i x_i}}{1 + e^{\beta_0 + \sum_{i=1}^{p} \beta_i x_i}} \right)^{y_i} \left(\frac{1}{1 + e^{\beta_0 + \sum_{i=1}^{p} \beta_i x_i}} \right)^{1-y_i}
\end{aligned}$$

é a função de verossimilhança para o modelo de regressão logística e

$$l(y; \boldsymbol{x}, \boldsymbol{\beta}) = \sum_{i=1}^{n} \left[y_i \log \left(\frac{e^{\beta_0 + \sum_{i=1}^{p} \beta_i x_i}}{1 + e^{\beta_0 + \sum_{i=1}^{p} \beta_i x_i}} \right) + (1 - y_i) \log \left(\frac{1}{1 + e^{\beta_0 + \sum_{i=1}^{p} \beta_i x_i}} \right) \right]$$

é a função de log-verossimilhança que deve ser maximizada em $\boldsymbol{\beta}$ por meio de métodos numéricos (Izbicki e dos Santos (2020)).

Para os casos em que a variável resposta assume $K > 2$ categorias, a regressão logística pode ser generalizada por meio da definição de $K - 1$ regressões logísticas diferentes. de tal forma que uma classe de referência arbitrária seja comparada com todas as demais. Definida a K-ésima categoria como referência, a probabilidade de Y pertencer à classe j, $j \in \{1, \ldots, K-1\}$, é dada por

$$\mathbb{P}(Y = j \mid \boldsymbol{X}, \boldsymbol{\beta}_j) = \frac{e^{\beta_{j0} + \sum_{i=1}^{p} \beta_{ji} X_i}}{1 + \sum_{k=1}^{K-1} e^{\beta_{k0} + \sum_{i=1}^{p} \beta_{ki} X_i}},$$

enquanto a probabilidade de Y pertencer à classe de referência K é dada por

$$\mathbb{P}(Y = K \mid \boldsymbol{X}, \boldsymbol{\beta}_j) = \frac{1}{1 + \sum_{k=1}^{K-1} e^{\beta_{k0} + \sum_{i=1}^{p} \beta_{ki} X_i}}.$$

Semelhantemente ao caso dicotômico, segue que os coeficientes de regressão podem ser estimados pela maximização da função de log-verossimilhança utilizando métodos numéricos.

3.3 *K-nearest neighbors*

O *k-nearest neighbors* (KNN) é um método de classificação que se baseia na proximidade entre as observações no espaço de covariáveis (Cover e Hart (1967)). Diferentemente de métodos paramétricos, como a regressão logística discutida na Seção 3.2, o KNN não assume uma forma funcional para modelar a relação entre a variável resposta Y e as covariáveis explicativas $\boldsymbol{X} = (X_1, \ldots, X_p)$. A ideia é que as observações semelhantes no espaço de covariáveis, isto é, observações próximas de acordo com alguma medida de similaridade, tendem a pertencer à mesma classe. Dessa forma, deseja-se estimar o classificador $f(\boldsymbol{x}_0)$ por meio da variável resposta observada das k observações mais próximas a uma nova observação $\boldsymbol{x}_0$.

Em um contexto de classificação,

$$\hat{f}(\boldsymbol{x}_0) = \text{moda}\left\{y_i : i \in \mathcal{N}_{\boldsymbol{x}_0}\right\}$$

é o estimador do classificador para $\boldsymbol{x}_0$, em que $\mathcal{N}_{\boldsymbol{x}_0}$ é o conjunto nas k observações mais próximas a $\boldsymbol{x}_0$ de acordo com alguma medida de similaridade, como a distância euclidiana. Essencialmente, esse estimador pode ser também definido como a classe j que retorna a maior probabilidade condicional estimada para $\boldsymbol{x}_0$ (James *et al.* (2014)), isto é,

$$\mathbb{P}(Y = j \mid \boldsymbol{X} = \boldsymbol{x}_0) = \frac{1}{k} \sum_{i \in \mathcal{N}_{\boldsymbol{x}_0}} \mathbb{I}\{y_i = j\},$$

em que k é o hiperparâmetro definido para o número de vizinhos a ser considerado.

A escolha de um valor muito pequeno de k tende a resultar em um modelo altamente flexível, mas também muito sensível a ruídos nos dados, o que pode levar ao *overfitting*. Por outro lado, valores muito grandes promovem maior suavização na fronteira de decisão. Na prática, esse valor é escolhido por validação cruzada de tal forma que o k selecionado

minimize o erro de classificação médio no conjunto de validação.

3.4 *Support vector machines*

Support vector machine (SVM) é um método de classificação que alcança ótimos resultados por meio da generalização de um classificador de margem máxima, uma abordagem que busca separar dois grupos de observações em um espaço de covariáveis p por meio de um hiperplano de dimensão $p-1$ (Cortes e Vapnik (1995)). Os pontos utilizados para definir essas margens são chamados de vetores de suporte.

A equação que define o hiperplano para o cenário de dimensão p é dada por

$$f(\boldsymbol{X}) = \beta_0 + \sum_{i=1}^{p} \beta_i X_i = 0, \tag{3.1}$$

de tal forma que, dado $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_p)$, qualquer ponto $\boldsymbol{X} = (X_1, \ldots, X_p)$ pertença ao hiperplano se a Equação (3.1) for satisfeita. Assim, um classificador intuitivo que faz uso da divisão dos dados em dois grupos pelo hiperplano e que atribui a cada observação uma classe $j \in \{-1, 1\}$ tem a seguinte forma:

$$Y = \begin{cases} 1, & \text{se } f(\boldsymbol{X}) \geq 0, \\ -1, & \text{se } f(\boldsymbol{X}) < 0. \end{cases}$$

Como existem inúmeras configurações de hiperplano capazes de separar os dados perfeitamente, o classificador de margem máxima escolhe o hiperplano que maximiza a distância mínima entre as observações de cada classe e o próprio hiperplano, isto é, a margem máxima M, como ilustrado na Figura 3.1. Assim, para uma amostra $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$, isso equivale a buscar o hiperplano com coeficientes $\boldsymbol{\beta}$ tais que

$$\begin{aligned} &\boldsymbol{\beta} = \arg\max_{\boldsymbol{\beta}} M \\ &\text{sujeito a} \quad \sum_{j=1}^{p} \beta_i^2 = 1 \quad \text{e} \qquad (3.2) \\ &y_i \left(\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} \right) \geq 1, \quad \text{para todo } i = 1, \ldots, n. \qquad (3.3) \end{aligned}$$

A restrição pela Equação (3.3) impõe que todas as observações estejam corretamente

classificadas e que, em conjunto com a restrição pela Equação (3.2), estejam a pelo menos uma distância M da margem (James *et al.* (2014)). Em diversas situações reais, entretanto, os dados não são perfeitamente separáveis por um hiperplano. Para lidar com esses casos, há uma generalização do conceito discutido, por meio de *support vector classifiers* (SVC), de tal forma que o classificador permita algumas observações estarem do lado "errado" do hiperplano em prol de abandonar a suposição de dados linearmente separáveis. Essa abordagem equivale à busca do hiperplano com coeficientes $\boldsymbol{\beta}$ tais que

$$\begin{aligned}
&\boldsymbol{\beta} = \arg\max_{\boldsymbol{\beta}} M \\
&\text{sujeito a} \quad \sum_{j=1}^{p} \beta_i^2 = 1 \quad \text{e} \\
&y_i \left(\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} \right) \geq M(1 - \epsilon_i), \quad \text{para todo } i = 1, \ldots, n,
\end{aligned}$$

em que $\epsilon_i > 0$ e $\sum_{i=1}^{n} \epsilon_i \leq C$. C é um hiperparâmetro que, quanto maior for seu valor, maior será a margem obtida, já que haverá uma maior tolerância a observações do lado "errado" do hiperplano e, consequentemente, obtém-se um modelo com um maior viés em prol de uma menor variância.

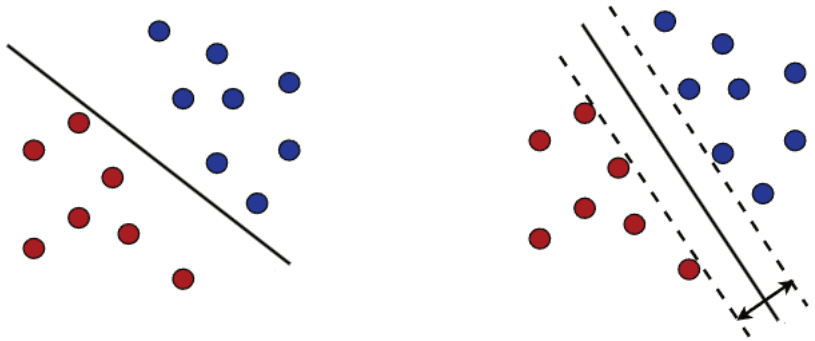


Figura 3.1: Dois possíveis hiperplanos que separam duas classes. O hiperplano da direita representa aquele que maximiza a margem. Fonte: Mohri *et al.* (2012).

A ideia central de um SVM, entretanto, é ser uma extensão do SVC para a obtenção de divisões mais complexas que hiperplanos, por meio do uso de kernels. A introdução de funções de kernel possibilita a projeção dos dados em espaços de maior dimensão nos quais um hiperplano pode separar classes que, em seu espaço original, não seriam linearmente separáveis.

O SVC pode ser representado na forma

$$f(\boldsymbol{x}) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle \boldsymbol{x}, \boldsymbol{x}_i \rangle,$$

em que $\langle \boldsymbol{x}_i, \boldsymbol{x}_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}$ é o produto interno a ser calculado para todos os pares de observações $(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$ pelo qual os coeficientes a_i são estimados (James *et al.* (2014)). Assim, pode-se generalizar o produto interno da Equação (3.4) por meio de um kernel genérico $K(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$, isto é,

$$f(\boldsymbol{x}) = \beta_0 + \sum_{i \in S} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i), \tag{3.4}$$

em que S é o conjunto que contém todos os vetores de suporte (Izbicki e dos Santos (2020)).

Para os casos em que a variável resposta assume $K > 2$ categorias, a abordagem *one-versus-one*, que funciona como uma extensão do SVM, pode ser utilizada. Essa abordagem constrói $\binom{K}{2}$ SVMs, isto é, um classificador para cada par de classes, e a classificação final de uma nova observação $\boldsymbol{x}_0$ é feita por meio da atribuição da classe mais frequente entre as $\binom{K}{2}$ predições.

3.5 Árvores de classificação

A árvore de classificação se trata de um método não paramétrico com alta interpretabilidade dos dados. A predição para uma observação no conjunto de treinamento se dá por meio da classe mais frequente dentro da região em que essa observação se situa, isto é, cria-se uma partição do espaço das covariáveis $\boldsymbol{X} = (X_1, \ldots, X_p)$ em m regiões disjuntas $R_1, \ldots, R_m$ tal que a predição da variável resposta qualitativa Y é dada por

$$\hat{f}(\boldsymbol{x}_0) = \text{moda}\{y_i : \boldsymbol{x}_i \in R_m\}$$

para uma observação do conjunto de treinamento $\boldsymbol{x}_0$ que pertence à região R_m (James *et al.* (2014)). A ideia central do método é que, como ilustrado na Figura 3.2, essa partição ocorra até a obtenção de uma folha, isto é, um nó terminal que não faz mais divisões e que representa uma classificação às observações que chegam até ele.

Essas divisões binárias do espaço de covariáveis são feitas por meio do índice de Gini,

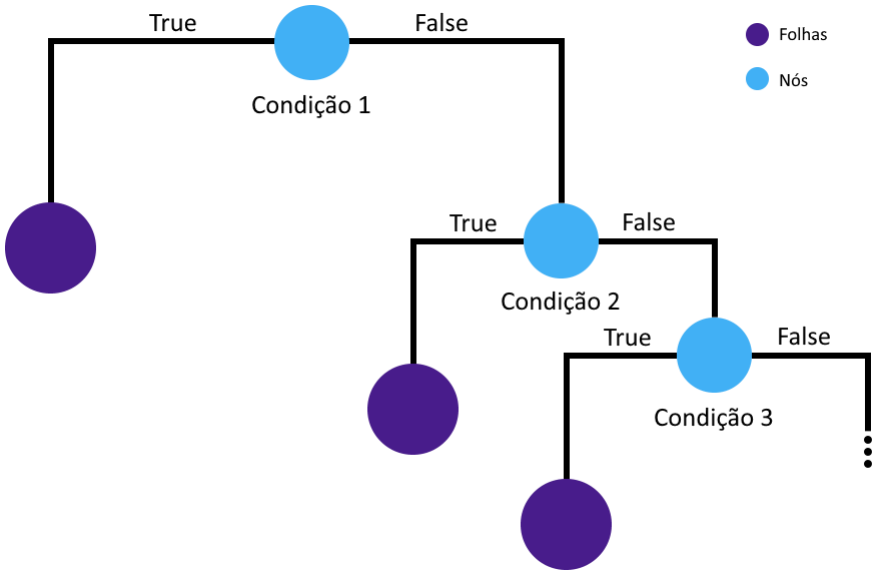


Figura 3.2: Representação de uma árvore de classificação.

que calcula a variância total sobre todas as K classes do conjunto de dados. Esse critério mede a "pureza" das folhas da árvore, isto é, quanto menor for o valor calculado, maior será a quantidade de observações pertencentes à mesma classe. É dado por

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

em que $\hat{p}_{mk}$ representa a proporção de observações na região R_m que são da k-ésima classe (James *et al.* (2014)).

Além disso, existe uma etapa em que a árvore é simplificada para evitar *overfitting*, chamada de poda. A poda consiste em remover ramificações da árvore que contribuem pouco para a redução do erro de classificação, resultando em um modelo mais simples e generalizável. Em geral, a proporção de erros no conjunto de validação é usada como estimativa do risco para esse processo (Izbicki e dos Santos (2020)).

3.6 *Boosting*

Em um contexto de classificação, o *boosting* (Freund e Schapire, 1997) é uma técnica que consiste na combinação sequencial de B diferentes classificadores fracos em um único classificador mais poderoso $f(\boldsymbol{X})$. A ideia é que, para uma série de iterações, $\hat{f}$ seja atualizado de tal forma que pesos maiores sejam atribuídos às observações classificadas erroneamente. Assim, o classificador estimado é melhorado iterativamente com a correção dos erros cometidos pelo modelo anterior.

Usualmente, árvores de classificação são utilizadas como classificadores fracos, de tal forma que as árvores cresçam sequencialmente – cada árvore utiliza informações obtidas pelas árvores ajustadas em iterações anteriores. A profundidade dessas árvores é controlada pelo hiperparâmetro d, enquanto o hiperparâmetro α representa a taxa de aprendizado que controla a contribuição de cada árvore sequencialmente (James *et al.* (2014)).

Considere que Y seja uma variável resposta dicotômica e que $\boldsymbol{X} = (X_1, \ldots, X_p)$ sejam as covariáveis do conjunto de dados. De Izbicki e dos Santos (2020), para uma amostra de treinamento $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)$, uma versão do algoritmo para obtenção do estimador é dada pelos seguintes passos:

1. Inicializa-se os pesos $w_1 = \cdots = w_m = \frac{1}{m}$.
2. Para $b = 1, \ldots, B$:
   (a) Ajusta-se um classificador $\hat{f}_b(\boldsymbol{x})$ para o conjunto de treinamento $\boldsymbol{x}$ de tamanho m com os pesos $w_1, \ldots, w_m$. Caso seja uma árvore de classificação, define-se que ela tenha d divisões.
   (b) Calcula-se o erro ponderado $err_b = \frac{\sum_{i=1}^{m} w_i \mathbb{I}\{y_i \neq \hat{f}_b(\boldsymbol{x}_i)\}}{\sum_{i=1}^{m} w_i}$.
   (c) Calcula-se $\alpha_b = \frac{1}{2} \log\left(\frac{1-err_b}{err_b}\right)$.
   (d) Atualiza-se $w_i \leftarrow w_i \exp(\alpha_b \mathbb{I}\{y_i \neq \hat{f}_b(\boldsymbol{x}_i)\})$ para $i = 1, \ldots, m$.
3. Retorna-se o modelo final $\hat{f}(\boldsymbol{x}) = \text{sinal}\left(\sum_{b=1}^{B} \alpha_b \hat{f}_b(\boldsymbol{x})\right)$.

Esse algoritmo, conhecido como AdaBoost (Freund e Schapire (1997)), garante que o modelo final $\hat{f}$ seja uma combinação ponderada dos classificadores fracos, em que cada um deles contribui ao modelo de acordo com sua proporção de erro. A função sinal no último passo do algoritmo faz com que a saída seja uma classificação binária, já que se trata de uma variável resposta dicotômica.

Um número de classificadores fracos B muito pequeno pode resultar em um modelo com baixa capacidade preditiva, enquanto valores muito grandes podem levar ao *overfitting*, especialmente se combinados com árvores muito profundas ou uma taxa de aprendizado alta. Em geral, B é escolhido por meio de validação cruzada, de tal forma que o desempenho preditivo em dados não utilizados no treinamento seja maximizado.

Além do AdaBoost, existem outras variantes do *boosting* desenvolvidas para melhorar o desempenho do algoritmo em diferentes cenários. O XGBoost (Chen e Guestrin (2016)),

por exemplo, um algoritmo muito popular por sua capacidade de lidar eficientemente com grandes volumes de dados, contrói as árvores de classificação sequenciais que minimizem a função de perda por meio de um gradiente descendente.

Capítulo 4

Aprendizado profundo

Uma rede neural é um modelo que pode ser descrito como um grafo direcionado cujos nós e arestas representam, respectivamente, os neurônios – unidades de computação do modelo – e as conexões entre os mesmos (Shalev-Shwartz e Ben-David (2014)). É um conceito relativamente antigo que, com os avanços computacionais ao longo das últimas décadas, ganhou cada vez mais relevância e se estabeleceu como um alicerce da área de aprendizado profundo.

A ideia de uma rede neural é, como o próprio nome informa, se basear na estrutura do cérebro humano para realizar computações complexas em tarefas de regressão ou classificação. Esse conceito engloba uma grande classe de métodos distintos com finalidades específicas, mas a primeira etapa deste trabalho abordará apenas dois deles: redes neurais multicamadas (ou *percepton* multicamadas) e redes neurais convolucionais.

4.1 Redes neurais multicamadas

As redes neurais multicamadas (*multilayer perceptrons*, em inglês) são uma extensão da ideia de redes neurais de camada única, isto é, modelos que possuem uma única camada oculta, capazes de capturar relações não lineares nos dados. Como ilustrado na Figura 4.1, a representação em grafo de uma rede neural é estruturada por meio de uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Em um contexto de classificação que envolva K classes, o modelo recebe as covariáveis $\boldsymbol{X} = (X_1, \ldots, X_p)$, isto é, p unidades na camada de entrada, e constrói um estimador não linear para cada uma das K unidades na camada de saída (Hastie *et al.* (2009)). Dessa forma, K funções de predição são estimadas, uma para cada possível classe que Y pode assumir, em que

cada função traz a probabilidade estimada associada à classe respectiva ser atribuída à variável resposta.

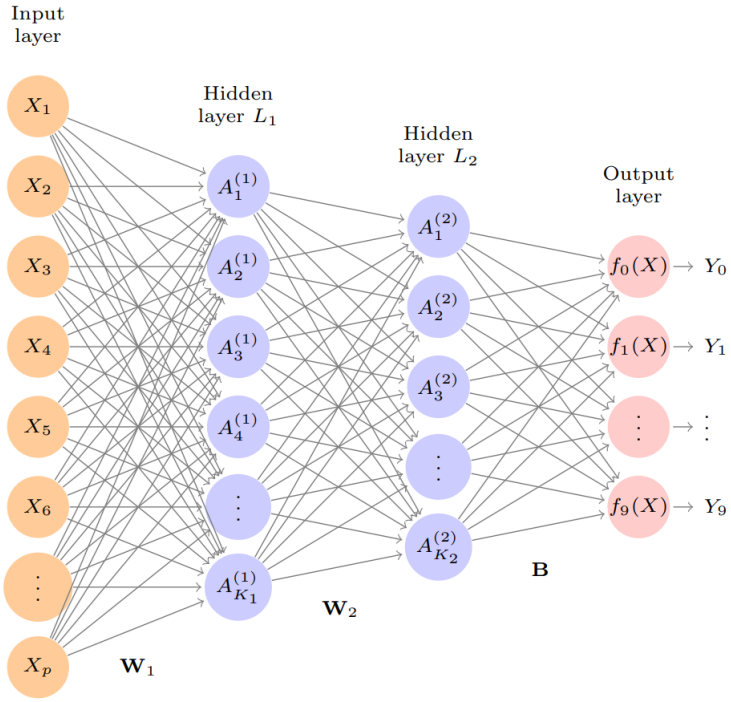


Figura 4.1: Representação de uma rede neural com duas camadas ocultas e múltiplas saídas. Fonte: James *et al.* (2014).

Com base na arquitetura da Figura 4.1, deseja-se estimar a probabilidade de Y assumir cada uma das $K = 10$ diferentes classes. A função $f_m(\boldsymbol{X})$ pode ser escrita na forma

$$f_m(\boldsymbol{X}) = \mathbb{P}(Y = m \mid \boldsymbol{X}) = \frac{e^{Z_m}}{\sum_{\ell=0}^{K-1} e^{Z_\ell}}, \tag{4.1}$$

tal que

$$\begin{aligned} Z_m &= \beta_{m0} + \sum_{\ell=1}^{K_2} \beta_{m\ell} g\left(w_{\ell 0}^{(2)} + \sum_{k=1}^{K_1} w_{\ell k}^{(2)} g\left(w_{k0}^{(1)} + \sum_{j=1}^{p} w_{kj}^{(1)} X_j\right)\right) \\ &= \beta_{m0} + \sum_{\ell=1}^{K_2} \beta_{m\ell} g\left(w_{\ell 0}^{(2)} + \sum_{k=1}^{K_1} w_{\ell k}^{(2)} A_k^{(1)}\right) \\ &= \beta_{m0} + \sum_{\ell=1}^{K_2} \beta_{m\ell} A_\ell^{(2)}, \end{aligned} \tag{4.2}$$

em que g é uma função de ativação não linear a ser definida e os parâmetros $\beta_{m\ell}$ e $w_{\ell k}$ devem ser estimados a partir dos dados para $m = 0, \ldots, K-1$ (James *et al.* (2014)). O índice superescrito na equação denota a camada oculta na rede e os valores K_1 e K_2 são, respectivamente, as quantidades de neurônios que compõem a primeira e a segunda

camada. O modelo é construído por meio de uma série de transformações complexas nas quais cada camada se alimenta da saída da camada antecessora. Essa estrutura é chamada de *feed-forward*, tendo em vista que a informação flui em uma única direção ao longo de sua arquitetura, isto é, não há um ciclo retroativo – pode-se pensar em um grafo direcionado acíclico.

Observa-se que a Equação (4.1) utiliza uma função *softmax* na camada de saída para garantir que as saídas se comportem como probabilidades e somem 1. Usualmente, utiliza-se a função de ativação ReLU (*Rectified Linear Unit*) nas demais camadas por sua simplicidade computacional e capacidade de introduzir não linearidade ao modelo, ao mesmo tempo que mitiga alguns problemas presentes em outras funções de ativação. A função ReLU é dada por

$$g(z) = \begin{cases} 0, & \text{se } z < 0, \\ z, & \text{caso contrário.} \end{cases}$$

Além disso, no contexto de redes neurais, existe também o conceito de *dropout*, uma técnica de regularização utilizada para evitar o *overfitting* dos modelos. Durante o treinamento, o *dropout* desativa aleatoriamente uma porção dos neurônios em cada iteração, de tal forma que os neurônios não se tornem muito dependentes uns dos outros e o modelo aprenda representações mais generalizáveis. A fração de unidades a serem desativadas em cada camada é controlada por um hiperparâmetro p, chamado de taxa de *dropout*, que define a probabilidade de um neurônio aleatório ser temporariamente removido.

Como comentado anteriormente, o treinamento de uma rede neural consiste em encontrar estimativas para os parâmetros $\beta_{m\ell}$ e $w_{\ell k}$ da Equação (4.2). Isso pode ser realizado por meio da minimização de uma função objetivo, como a entropia cruzada (Izbicki e dos Santos (2020)), que é dada por

$$\mathrm{CE}(f_0, \ldots, f_{K-1}) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{k=0}^{K-1} \mathbb{I}(y_i = k) \log(f_k(\boldsymbol{x}_i)).$$

Um possível método numérico de otimização a ser utilizado nessa estimação é o gradiente descendente (Hastie *et al.* (2009)). Esse método consiste em atualizar os parâmetros em pequenas porções, de tal forma que o treinamento não seja realizado em todo o conjunto de treinamento de uma só vez e, dessa forma, permita que o processo de otimização seja mais computacionalmente viável. Há também algumas variantes desse método, como o gradiente descendente estocástico (*stochastic gradient descent*, em inglês), em que os

parâmetros são atualizados com base em subconjuntos aleatórios dos dados (Bottou (2010)), e o gradiente descendente com momento, que incorpora uma média móvel dos gradientes para acelerar a convergência ao mínimo global (Qian (1999)).

Apesar dos pontos positivos, os métodos de otimização citados atualizam todos os parâmetros do modelo a cada iteração e dependem de uma taxa de aprendizado fixa, isto é, um parâmetro que define o tamanho do passo a ser tomado em direção ao ponto de mínimo. Os otimizadores que ajustam essa taxa de aprendizado para cada parâmetro do modelo iterativamente garantem uma convergência ainda mais rápida, já que cada parâmetro é atualizado individualmente com sua respectiva taxa de aprendizado – otimizadores como Adam e RMSprop são alguns desses métodos adaptativos (Goodfellow *et al.* (2016)).

4.2 Redes neurais convolucionais

Essencialmente, as redes neurais convolucionais (*convolutional neural networks* (CNNs), em inglês) são *perceptrons*, como vistos na Seção 4.1, com algumas etapas adicionais. Esses modelos possuem uma camada de convolução e se apresentam como uma evolução das redes neurais multicamadas para tarefas de classificação de imagens (James *et al.* (2014)). A etapa de convolução aplica filtros (ou kernels) à imagem de entrada para extrair características locais, como bordas, texturas ou padrões específicos. Assim, a arquitetura de uma CNN pode ser resumida em três aspectos: (i) mapeamento de características, (ii) camadas de *pooling* e (iii) camadas totalmente conectadas (*fully connected layers*, em inglês).

No mapeamento de características, identifica-se padrões locais na imagem por meio de filtros que são aprendidos no treinamento do modelo. Esse mecanismo ocorre na camada de convolução, isto é, uma camada que será composta por um grande número desses filtros, em que cada um deles evidencia alguma característica local. A partir dessa camada, o mapeamento é realizado pela aplicação desses filtros na imagem original, o que resulta em um mapa de características (*feature map*, em inglês) para cada filtro aplicado. Conforme a rede se aprofunda, esses *feature maps* capturam hierarquicamente características cada vez mais abstratas e complexas, de tal forma que a presença ou ausência desses contribua na probabilidade estimada de Y assumir cada uma das possíveis classes (James *et al.* (2014)).

No intuito de reduzir a complexidade computacional, cada neurônio na camada de

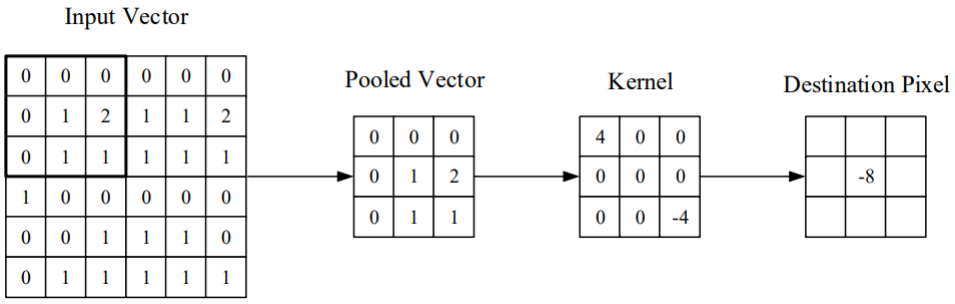


Figura 4.2: Representação de uma camada de convolução. Fonte: O'Shea e Nash (2015).

convolução está conectado a apenas uma pequena região da imagem de entrada (O'Shea e Nash (2015)). Como ilustrado na Figura 4.2, o elemento central do filtro 3×3 é posto sobre a imagem de entrada e substituído por uma soma ponderada dos pixels pertencentes à região. Existem três hiperparâmetros principais nessa camada que controlam essa redução de complexidade:

- **Profundidade:** define o número de filtros e influencia a capacidade de extração dos padrões;
- ***Stride*:** determina o deslocamento do filtro sobre a entrada, isto é, quanto maior for o *stride* definido, menor será a dimensão do *feature map* para um kernel (filtro) específico;
- ***Padding*:** adiciona bordas artificiais à entrada para controlar a dimensão da saída. Faz-se possível, por meio desse hiperparâmetro, preservar a dimensão original da imagem mesmo com um *stride* maior do que 1.

A etapa de *pooling* visa reduzir a dimensionalidade da representação da imagem convolucionada, de tal forma que a quantidade de parâmetros a serem estimados diminua e, consequentemente, a complexidade do modelo também. Essa camada opera nos *feature maps* gerados pela camada de convolução e, usualmente, utiliza-se a operação *max-pooling* que condensa a informação contida em blocos 2×2 de pixels em um único valor – o máximo entre os valores do bloco.

Por fim, a camada totalmente conectada é responsável por combinar as características extraídas pelas camadas convolucionais e de *pooling* para realizar a classificação final. Essa camada recebe os *feature maps* achatados, isto é, transformados em um vetor unidimensional na etapa de *flatten*, e os processa por meio de uma ou mais camadas densas, em que cada neurônio estará conectado a todos os neurônios da camada anterior. Em

problemas de classificação, assim como na Seção 4.1, a última camada densa utiliza uma função de ativação *softmax* para produzir as probabilidades de Y assumir cada uma das possíveis classes (James *et al.* (2014)).

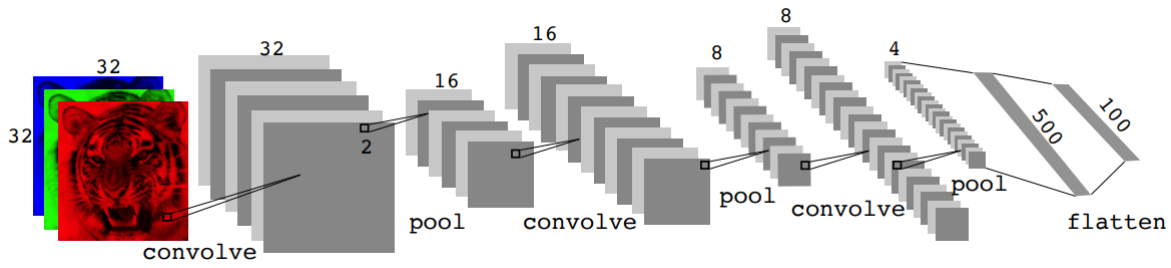


Figura 4.3: Representação de uma rede neural convolucional para classificação de imagens coloridas de dimensão 32×32. Fonte: James *et al.* (2014).

Como representado na Figura 4.3, uma arquitetura típica de CNN para classificação de imagens consiste em uma sequência de camadas convolucionais seguidas por camadas de *pooling*, que são então seguidas por uma ou mais camadas totalmente conectadas. Geralmente, recomenda-se que camadas de convolução grandes sejam divididas em camadas menores para reduzir a complexidade computacional envolvida e, além disso, recomenda-se também a utilização de duas camadas de convolução antes de cada camada de *pooling*, tendo em vista que isso permite a identificação de padrões cada vez mais complexos (O'Shea e Nash (2015)).

4.3 Redes neurais residuais

As redes neurais residuais (ResNets ou, em inglês, *residual neural networks*) foram propostas com o objetivo de mitigar o problema da degradação de desempenho em redes neurais muito profundas (He *et al.* (2015)). À medida que o número de camadas aumenta, torna-se mais difícil treinar a rede de forma eficiente, já que os gradientes tendem a se tornar muito pequenos ou muito grandes ao longo do caminho pelas camadas. Ainda com a utilização de funções de ativação específicas, como a ReLU, e algumas técnicas de normalização, redes muito profundas tendem a apresentar um desempenho inferior ao de redes mais rasas, o que contraria a ideia de que redes mais profundas seriam mais versáteis (He *et al.* (2015)).

A inovação das ResNets é a introdução de conexões de atalho (*skip connections*, em inglês) – essas conexões permitem que o fluxo de informação seja transferido diretamente de uma camada para outra não adjacente, o que facilita a atualização dos parâmetros em

camadas profundas e, consequentemente, o treinamento da rede como um todo. Um bloco residual recebe uma entrada $\boldsymbol{X} = (X_1, \ldots, X_p)$ e, ao invés de aprender uma transformação direta $\mathcal{H}(\boldsymbol{X})$, aprende-se uma função residual $\mathcal{F}(\boldsymbol{X}) := \mathcal{H}(\boldsymbol{X}) - \boldsymbol{X}$, de modo que a saída da unidade seja expressa como:

$$Y = \mathcal{F}(\boldsymbol{X}, W_i) + \boldsymbol{X}, \tag{4.3}$$

em que W_i representa a matriz de pesos a serem ajustados no bloco residual da i-ésima camada.

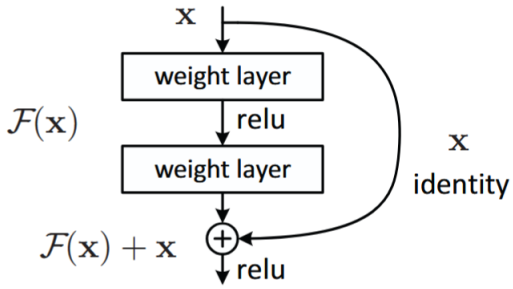


Figura 4.4: Esquema ilustrando o processo de um bloco residual. Fonte: He *et al.* (2015).

Como ilustrado na Figura 4.4 e pela Equação (4.3), o bloco residual adiciona à entrada original $\boldsymbol{X}$ o resultado da transformação não linear aprendida $\mathcal{F}(\boldsymbol{X})$, isto é, a rede aprende por meio das correções que são adicionadas à entrada original.

Com relação à performance em dados reais, as redes neurais residuais demonstraram resultados superiores quando comparadas a arquiteturas convolucionais tradicionais, especialmente em conjuntos de dados mais complexos e com maior variabilidade entre as classes. Nos experimentos conduzidos em He *et al.* (2015), observou-se que as ResNets conseguem manter uma acurácia elevada em arquiteturas com altas profundidades, o que indica que as conexões residuais facilitam a otimização de redes profundas.

4.4 U-Net

U-Net (Ronneberger *et al.* (2015)) é uma arquitetura de rede neural convolucional desenvolvida inicialmente para tarefas de segmentação de imagens biomédicas, especificamente para o desafio do ISBI (*International Symposium on Biomedical Imaging*) de segmentação de estruturas celulares, realizado em 2015. A nomenclatura deriva de sua estrutura com formato em "U", que é composta por um caminho de *encoding* seguido por um caminho de *decoding* com conexões de atalho entre camadas de mesmo nível, assim

como em redes neurais residuais, para que os mapas de características gerados no *encoder* sejam passados diretamente ao *decoder*.

O *encoder* é responsável por capturar o contexto da imagem de tal forma que a dimensão dela seja reduzida, por meio de camadas convolucionais seguidas de operações de *max-pooling*. Por sua vez, o *decoder* tem a função de restaurar a resolução da imagem original e refinar os detalhes locais a partir da informação capturada nas etapas anteriores.

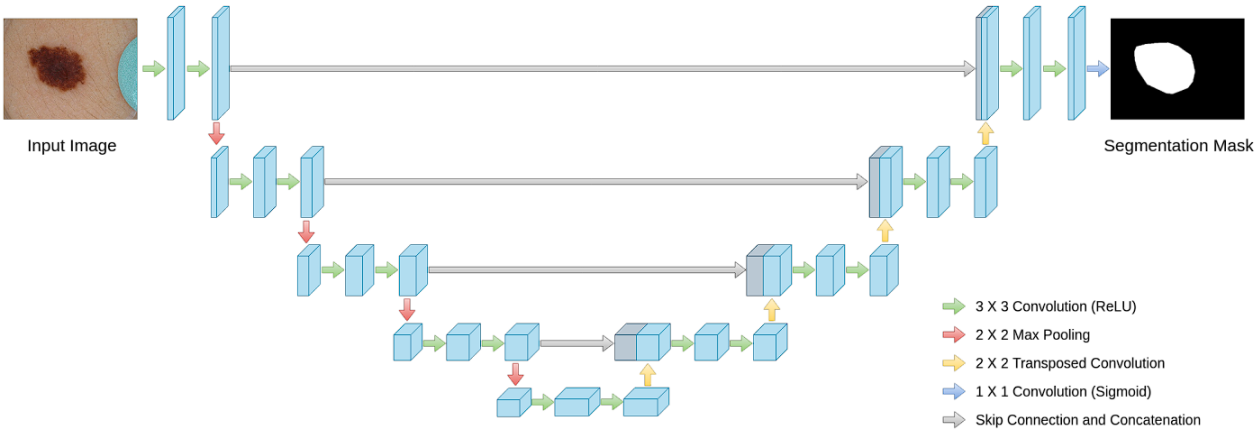


Figura 4.5: Esquema que ilustra o formato de "U" observado na arquitetura U-Net. Fonte: Majidifard *et al.* (2020).

Observa-se que, pela Figura 4.5, a arquitetura apresenta uma simetria, com o lado esquerdo correspondendo ao caminho de *encoding* e o lado direito ao caminho de *decoding*. As conexões horizontais entre camadas de mesmo nível atuam como canais de informação direta entre o processo de análise e o de síntese, o que permite à rede preservar detalhes espaciais importantes ao longo do processo.

Dessa forma, a inclusão dessas conexões de atalho garante que informações espaciais relevantes – como bordas e contornos finos – não se percam durante o processo de redução de dimensionalidade. Nos experimentos em Ronneberger *et al.* (2015), a arquitetura demonstrou desempenho competitivo em diferentes desafios de segmentação e superou diversas abordagens que utilizavam CNNs tradicionais. Ainda em casos com um número limitado de imagens rotuladas, graças a essa utilização eficiente das informações espaciais recuperadas pelo *decoder*, foram obtidos modelos muito capazes em generalização.

Capítulo 5

Aplicação

5.1 Conjuntos de dados

Com o intuito de fomentar o desenvolvimento de recursos voltados à leitura facilitada de caracteres cursivos, a organização *Center for Open Data in the Humanities* (CODH) disponibilizou um enorme repositório de escrituras e livros japoneses pré-modernos digitalizados na competição *Kuzushiji Challenge*. Esse repositório é público e pode ser acessado em `https://codh.rois.ac.jp/char-shape/`. Cada obra literária presente nesse repositório possui diversos escaneamentos que permitem a extração de cada um dos caracteres presentes em seus textos. Dessa forma, muitas ferramentas foram criadas com a finalidade de tornar a leitura de textos com caracteres cursivos acessível à população japonesa, como o aplicativo *miwo*, capaz de transcrever textos antigos em tempo real com o modelo RURI (瑠璃), desenvolvido por Tarin Clanuwat para a competição (Clanuwat, 2022).

Inspirados no famoso conjunto de dados MNIST que contém imagens de dígitos manuscritos, um grupo de pesquisadores da área de aprendizado de máquina disponibilizou um pré-processamento do conjunto de dados da competição dividido em três partes: (i) Kuzushiji-MNIST, um conjunto balanceado que contém imagens de 10 classes de caracteres; (ii) Kuzushiji-49, um conjunto desbalanceado que contém imagens de todos os 49 caracteres do silabário moderno japonês *hiragana*; (iii) Kuzushiji-Kanji, um conjunto desbalanceado que contém imagens de 3823 ideogramas chineses distintos (Clanuwat *et al.*, 2018).

5.1.1 MNIST

MNIST (*Modified National Institute of Standards and Technology database*) (LeCun *et al.*, 1998) é um banco de dados de dígitos manuscritos, sendo um dos mais conhecidos na área de aprendizado de máquina. O conjunto contém 70000 imagens de dígitos manuscritos e é balanceado entre as 10 classes correspondentes aos dígitos de 0 a 9, isto é, cada classe desse conjunto possui 7000 amostras.

As imagens são monocromáticas e todas possuem a mesma dimensão de 28×28 pixels. A cor de cada pixel é representada por um valor entre 0 e 255, de tal forma que o pixel será branco quanto mais próximo estiver de 0 e será preto quanto mais próximo estiver de 255, como ilustrado na Figura 5.1. Para o contexto do projeto, esses valores foram padronizados entre 0 e 1.

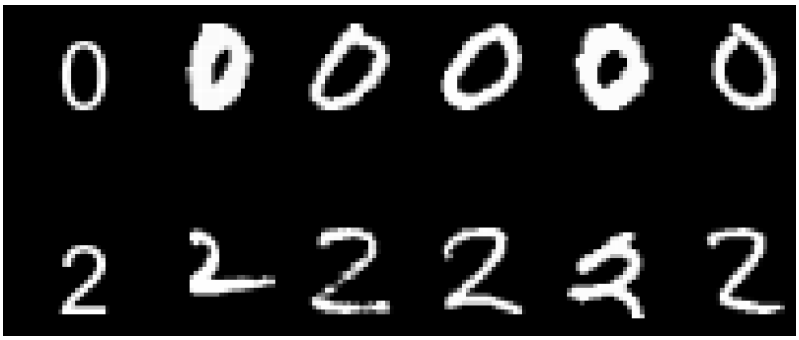


Figura 5.1: Cinco exemplos de imagens presentes no conjunto MNIST para os dígitos 0 e 2. A primeira coluna contém a representação moderna dos dígitos.

A ideia central de utilizar esse conjunto de dados é comparar o desempenho dos métodos no MNIST com o desempenho dos mesmos métodos no Kuzushiji-MNIST, tendo em vista que o primeiro já foi extensivamente explorado na literatura e, dessa forma, estabelece-se como um ponto de referência para avaliar a dificuldade intrínseca aos dados de caracteres japoneses cursivos.

5.1.2 Kuzushiji-49

O conjunto de dados Kuzushiji-49 (Clanuwat *et al.* (2018)) abrange completamente o silabário *hiragana* e contém 270912 imagens manuscritas dos 49 caracteres que o compõe. É um conjunto desbalanceado cujas frequências dos caracteres refletem, de certa forma, suas aparições em manuscritos antigos, isto é, caracteres menos utilizados possuem uma menor quantidade de imagens digitalizadas.

Observa-se que, pela Tabela 5.2, a quantidade de imagens por caractere varia entre

Tabela 5.1: Número de amostras por caractere em *hiragana* no Kuzushiji-49.

Caractere	Amostras	Caractere	Amostras	Caractere	Amostras
あ (a)	7000	ち (chi)	2983	む (mu)	1998
い (i)	7000	つ (tsu)	7000	め (me)	3946
う (u)	7000	て (te)	7000	も (mo)	7000
え (e)	903	と (to)	7000	や (ya)	7000
お (o)	7000	な (na)	7000	ゆ (yu)	1858
か (ka)	7000	に (ni)	7000	よ (yo)	7000
き (ki)	7000	ぬ (nu)	2399	ら (ra)	7000
く (ku)	7000	ね (ne)	2850	り (ri)	7000
け (ke)	5481	の (no)	7000	る (ru)	7000
こ (ko)	7000	は (ha)	7000	れ (re)	7000
さ (sa)	7000	ひ (hi)	5968	ろ (ro)	2487
し (shi)	7000	ふ (fu)	7000	わ (wa)	2787
す (su)	7000	へ (he)	7000	ゐ (i)	485
せ (se)	4843	ほ (ho)	2317	ゑ (e)	456
そ (so)	4496	ま (ma)	7000	を (wo)	7000
た (ta)	7000	み (mi)	3558	ん (n)	7000
				ゝ	4097

7000 e 456. Apesar desse desbalanceamento, todas as imagens são monocromáticas e possuem a mesma dimensão de 28 × 28 pixels, ou seja, estão no padrão utilizado no conjunto de dados MNIST, como ilustrado na Figura 5.2.

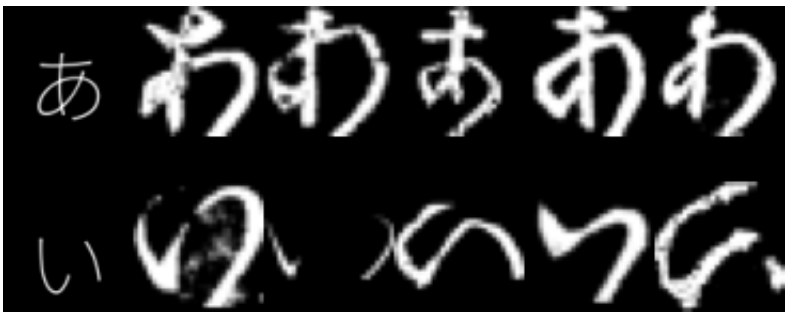


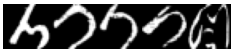
Figura 5.2: Cinco exemplos de imagens presentes no conjunto Kuzushiji-49 para os caracteres “あ” (a) e “ゝ” (i). A primeira coluna contém a representação moderna dos caracteres.

5.1.3 Kuzushiji-MNIST

O Kuzushiji-MNIST (Clanuwat *et al.* (2018)) é um conjunto de dados balanceado que, por ser baseado no MNIST, limita o conjunto Kuzushiji-49 a apenas 10 classes, isto é, 10 caracteres do silabário *hiragana*. Todas as classes desse conjunto possuem 7000 amostras, em que as imagens são monocromáticas e todas possuem a mesma dimensão de 28 × 28 pixels. Além disso, pela Tabela 5.2 que contém alguns exemplos dos caracteres

que compõem o conjunto, verifica-se a presença de suas variantes históricas para cada uma das 10 classes.

Tabela 5.2: Número de amostras e cinco exemplos de imagens para cada caractere em *hiragana* do Kuzushiji-MNIST.

Caractere	Amostras	Exemplos
お (o)	7000	
き (ki)	7000	
す (su)	7000	
つ (tsu)	7000	
な (na)	7000	
は (ha)	7000	
ま (ma)	7000	
や (ya)	7000	
れ (re)	7000	
を (wo)	7000	

5.1.4 Competição *Kuzushiji Challenge*

A competição *Kuzushiji Challenge*, além de ter disponibilizado as digitalizações das páginas de manuscritos antigos, também forneceu as caixas de delimitação para cada caractere contido nas páginas e seus rótulos verdadeiros. O conjunto de dados contém 3606 imagens de páginas de diferentes tamanhos provenientes de livros distintos, o que garante uma alta variabilidade dos dados com relação à forma de escrita, tamanho e coloração.

A Figura 5.3 ilustra os dados que foram disponibilizados pela competição. Para cada página do conjunto de dados, foi disponibilizado um mapeamento que contém as *bounding boxes* que delimitam cada caractere presente na página e o respectivo rótulo verdadeiro, isto é, o caractere que está contido em cada caixa.

Além disso, observa-se que, pela Figura 5.4, esse conjunto de dados é desbalanceado com relação a esses caracteres pertencentes às páginas digitalizadas, tendo em vista que esse valor depende da frequência natural de utilização dos caracteres nas escrituras antigas.

Figura 5.3: À esquerda, a imagem original da página digitalizada presente no conjunto de dados. À direita, a mesma página com suas caixas de delimitação, em vermelho, para cada caractere presente na página.

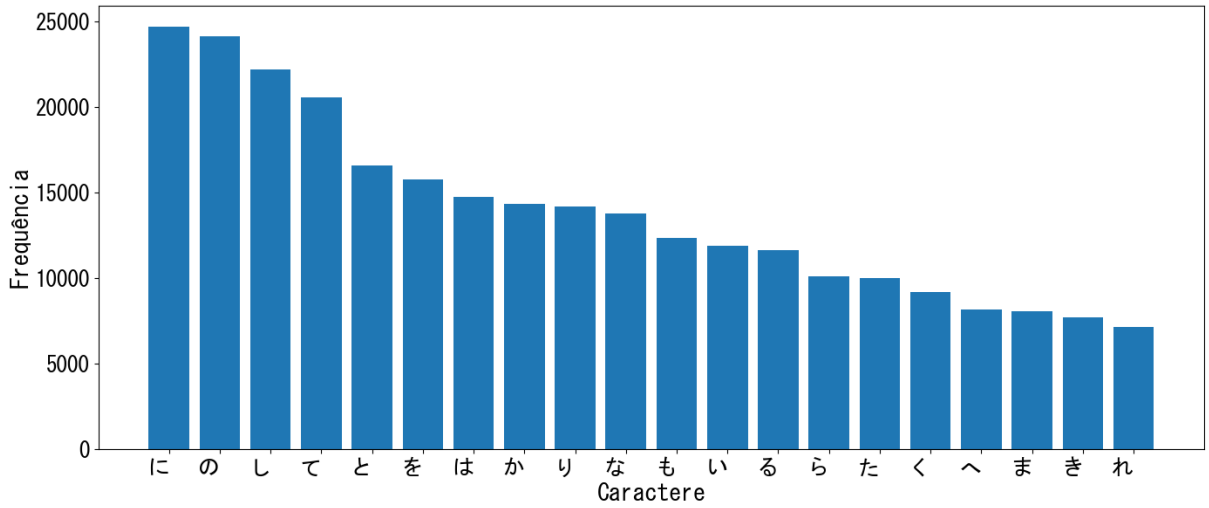


Figura 5.4: Histograma para os 20 caracteres mais frequentes observados no conjunto de dados da competição *Kuzushiji Challenge*.

5.2 Resultados

Como comentado na Seção 1.2, um dos objetivos deste trabalho é comparar a performance de modelos de aprendizado profundo com métodos de classificação tradicionais. Três arquiteturas de CNN foram treinadas para a classificação do conjunto de dados Kuzushiji-MNIST e, com base nas métricas da Seção 2.1.4, a arquitetura que obteve o melhor desempenho foi escolhida para ser comparada com os demais métodos. Todos os códigos utilizados neste trabalho estão disponíveis no repositório do autor no GitHub: `https://github.com/DCCXXVII/ufscar-tcc`.

As divisões do MNIST e Kuzushiji-MNIST em conjuntos de treino, validação e teste foram dadas da seguinte forma:

$$\underbrace{(\boldsymbol{X}_1, Y_1), (\boldsymbol{X}_2, Y_2), \ldots, (\boldsymbol{X}_s, Y_s)}_{\text{Treinamento (85.7\%)}}, \quad \underbrace{(\boldsymbol{X}_{s+1}, Y_{s+1}), \ldots, (\boldsymbol{X}_t, Y_t)}_{\text{Validação (2.86\%)}}, \quad \underbrace{(\boldsymbol{X}_{t+1}, Y_{t+1}), \ldots, (\boldsymbol{X}_n, Y_n)}_{\text{Teste (11.44\%)}},$$

isto é, um conjunto de treinamento com 6000 amostras por classe, um conjunto de teste com 800 amostras por classe e um conjunto de validação com 200 amostras por classe.

As três arquiteturas para o método de aprendizado profundo foram adaptadas de trabalhos aplicados ao conjunto de dados MNIST, dada a similaridade do problema de classificação entre dígitos manuscritos e caracteres cursivos japoneses. A escolha dessas arquiteturas foi motivada pelos bons desempenhos reportados, como em LeCun *et al.* (1989) e Wang *et al.* (2020), sendo a LeNet uma das mais tradicionais para esse tipo de tarefa.

A primeira arquitetura, ilustrada na Figura 5.5, possui 2 camadas convolucionais com, respectivamente, 32 e 64 filtros de convolução 3×3, 2 camadas de *max-pooling* com filtros 2×2 de *stride* 2 e 2 camadas totalmente conectadas. Após cada camada convolucional, a função de ativação ReLU foi aplicada e o otimizador Adam foi utilizado.

A segunda arquitetura foi definida para servir de comparação com a primeira arquitetura. Ela possui a mesma estrutura, mas foi utilizado um *dropout* com taxa de 0.25 após a primeira camada de *max-pooling* e, após a segunda camada, um *dropout* com taxa de 0.5, no intuito de reduzir o *overfitting* do modelo.

A terceira, ilustrada na Figura 5.6, foi inspirada na arquitetura LeNet (Lecun *et al.* (1998)), criada para tarefas de classificação de dígitos e caracteres manuscritos. Ela possui 2 camadas convolucionais com, respectivamente, 6 e 16 filtros de convolução 5×5,

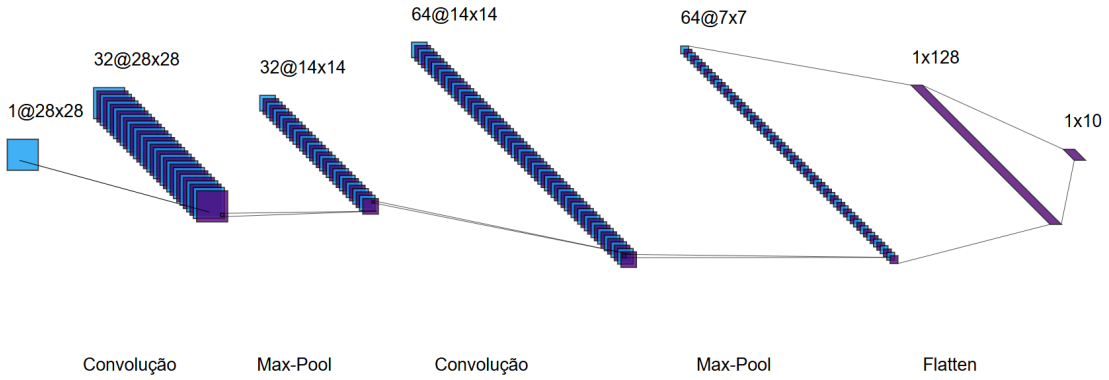


Figura 5.5: Representação da primeira arquitetura de CNN.

2 camadas de *max-pooling* com filtros 2×2 de stride 2 e 3 camadas totalmente conectadas. Após cada camada convolucional, a função de ativação ReLU foi aplicada, e o otimizador Adam foi utilizado.

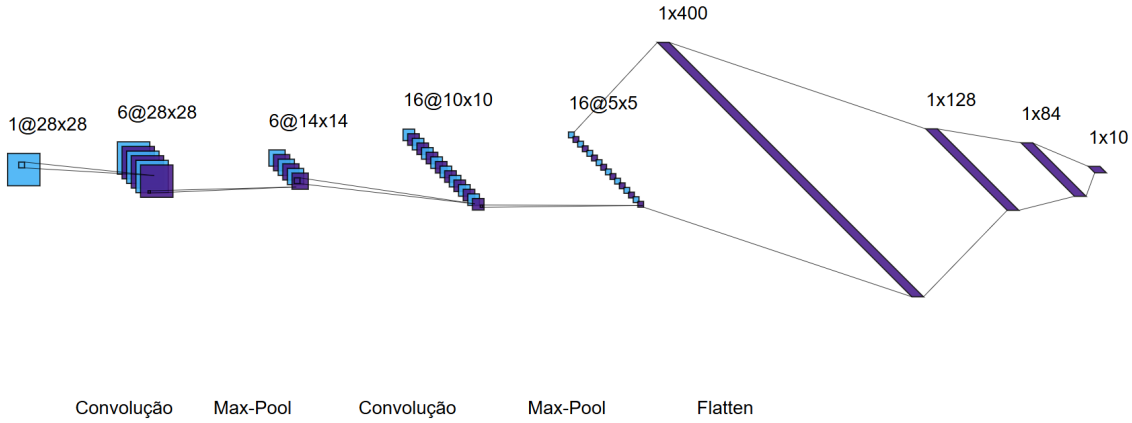


Figura 5.6: Representação da terceira arquitetura de CNN.

Para a comparação com os métodos de classificação tradicionais, os modelos de regressão logística, KNN, SVM e *boosting* foram ajustados por meio da biblioteca scikit-learn. A estimação dos parâmetros de cada modelo foi realizada separadamente para o MNIST e o Kuzushiji-MNIST, de tal forma que o número de observações por classe no conjunto de treinamento fosse o mesmo em ambos, isto é, 6000 amostras por classe. O modelo de regressão logística foi treinado com regularização L2 e o número de vizinhos dos KNNs, os hiperparâmetros do *boosting* e a função de kernel para o SVM foram escolhidos por meio de validação cruzada.

As análises e o desenvolvimento das três arquiteturas de CNN foram realizados por meio da biblioteca PyTorch. Pelos resultados dispostos nas Tabelas 5.3 e 5.4, a segunda arquitetura de CNN obteve o melhor desempenho no conjunto de teste. Observa-se que, quando comparada à primeira arquitetura, houve um impacto positivo da aplicação de duas camadas de *dropout* na segunda arquitetura, o que provavelmente evitou alguns

Tabela 5.3: Métricas calculadas no conjunto de teste do MNIST.

Método	**MNIST**				**Tempo de execução**
	Acurácia	**Precisão**	***Recall***	***F1-Score***	
Regressão Logística	0.917	0.916	0.916	0.916	5s
KNN	0.971	0.972	0.971	0.971	10s
SVM	0.982	0.982	0.982	0.982	173s
Boosting	0.983	0.983	0.983	0.983	277s
1ª CNN	0.990	0.990	0.990	0.990	72s
2ª CNN	**0.994**	**0.994**	**0.994**	**0.994**	91s
3ª CNN	0.991	0.991	0.991	0.991	95s

Tabela 5.4: Métricas calculadas no conjunto de teste do Kuzushiji-MNIST.

Método	**Kuzushiji-MNIST**				**Tempo de execução**
	Acurácia	**Precisão**	***Recall***	***F1-Score***	
Regressão Logística	0.699	0.701	0.690	0.691	5s
KNN	0.921	0.923	0.921	0.921	11s
SVM	0.929	0.930	0.929	0.929	396s
Boosting	0.931	0.931	0.931	0.931	304s
1ª CNN	0.930	0.932	0.930	0.930	75s
2ª CNN	**0.957**	**0.957**	**0.957**	**0.957**	165s
3ª CNN	0.934	0.934	0.934	0.934	81s

efeitos de *overfitting*.

Além disso, observa-se que a tarefa de classificação se tornou consideravelmente mais desafiadora no conjunto de dados Kuzushiji-MNIST quando comparada ao MNIST, algo que já era esperado devido à maior complexidade das formas do *hiragana* e suas variações históricas. Entre os métodos utilizados, a rede neural convolucional obteve o melhor desempenho com uma acurácia de 99.4% no MNIST e 95.7% no Kuzushiji-MNIST. A Tabela 5.5 mostra a acurácia e o *F1-Score* da segunda arquitetura de CNN a nível dos caracteres e, por meio dela, verifica-se que o desempenho no conjunto de teste foi satisfatório em todas as 10 classes.

Esse resultado reforça a alta capacidade das redes neurais convolucionais em tarefas de classificação de imagens e nos dá indícios de que esse método seria adequado para realizar predições de escrituras cursivas em *hiragana* em tempo real. Além disso, destaca-se que, mesmo com a presença das variantes históricas desses caracteres, a performance dos algoritmos avaliados foi muito satisfatória, com exceção à regressão logística, que obteve bons resultados no MNIST mas uma baixa acurácia no Kuzushiji-MNIST.

Além disso, como o objetivo principal é criar um modelo que seja capaz de classificar caracteres japoneses em novas páginas de textos manuscritos, os mesmos métodos foram ajustados para o conjunto de dados Kuzushiji-49, que contém imagens de todos os carac-

Tabela 5.5: Desempenho da segunda arquitetura de CNN treinada pelo conjunto Kuzushiji-MNIST no conjunto de teste.

Caractere	**Acurácia**	***F1-Score***
お (o)	0.958	0.959
き (ki)	0.942	0.956
す (su)	0.906	0.931
つ (tsu)	0.982	0.965
な (na)	0.943	0.944
は (ha)	0.961	0.963
ま (ma)	0.979	0.956
や (ya)	0.969	0.973
れ (re)	0.978	0.966
を (wo)	0.962	0.966
Média	0.957	0.957

teres do silabário *hiragana* e suas variantes. O conjunto foi dividido em treino, validação e teste nas mesmas proporções que os conjuntos MNIST e Kuzushiji-MNIST, mas, tendo em vista que o conjunto é desbalanceado, as proporções entre classes foram mantidas.

Tabela 5.6: Métricas calculadas no conjunto de teste do Kuzushiji-49.

Método	**Kuzushiji-49**				**Tempo de execução**
	Acurácia	**Precisão**	***Recall***	***F1-Score***	
Regressão Logística	0.567	0.571	0.567	0.564	84s
KNN	0.855	0.862	0.837	0.845	156s
SVM	–	–	–	–	–
Boosting	0.883	0.889	0.872	0.879	–
1ª CNN	0.898	0.900	0.898	0.898	101s
2ª CNN	**0.929**	**0.927**	**0.920**	**0.922**	338s
3ª CNN	0.897	0.899	0.897	0.897	237s

Pela Tabela 5.6, verifica-se que, assim como as performances observadas para as bases MNIST e Kuzushiji-MNIST, a segunda arquitetura de CNN obteve as melhores performances em todas as métricas. O *boosting* ajustado obteve performance próxima à primeira e à terceira arquitetura de CNN, enquanto os demais métodos – com exceção ao SVM, que foi excluído por ultrapassar um limite de 30 minutos – obtiveram performances inferiores. Dessa forma, o melhor modelo ajustado para o conjunto de dados referente aos 49 caracteres do *hiragana* foi a segunda arquitetura de rede neural convolucional definida, que alcançou uma acurácia de 92.9% e um *F1-Score* de 92.2% no conjunto de teste.

5.2.1 Extração de caracteres via U-Net

Após a seleção da segunda arquitetura de CNN para classificação de caracteres em *hiragana*, a próxima etapa consiste na aplicação de um modelo U-Net para extrair caracteres de novas páginas de manuscritos japoneses, isto é, imagens que não foram utilizadas no treinamento do modelo de classificação. Para isso, foi utilizada uma arquitetura que combina elementos das implementações clássicas do U-Net, descritas na Seção 4.4, com a incorporação de alguns componentes de uma ResNet com 18 camadas como *encoder*. As páginas disponibilizadas na competição *Kuzushiji Challenge*, que contêm as caixas de delimitação (*bounding boxes*, em inglês) para todos os caracteres das páginas, foram utilizadas para treinamento e validação do modelo.

O modelo produz duas máscaras binárias sobre a nova página que detectam duas classes como saída: a região que contém caracteres e os centros dos caracteres. Com a informação de ambos, aplica-se um k-médias para que os pixels da região que contém caracteres sejam atribuídos aos centros estimados mais próximos e, com base nos pixels que compõem cada agrupamento, calculam-se as caixas de delimitação.

Inicialmente, com o intuito de manter uma consistência entre o conjunto de dados no treinamento do modelo e os novos dados, as imagens foram processadas e formatadas para um tamanho padronizado 400×400, de tal forma que a proporção fosse preservada por meio de um preenchimento com zeros nas bordas. Além disso, como ilustrado na Figura 5.7, foi aplicado um filtro que normaliza a intensidade dos pixels com base na mediana e no desvio padrão dos pixels que compõem a imagem, com o intuito de melhorar a robustez do modelo às variações de iluminação e contraste.

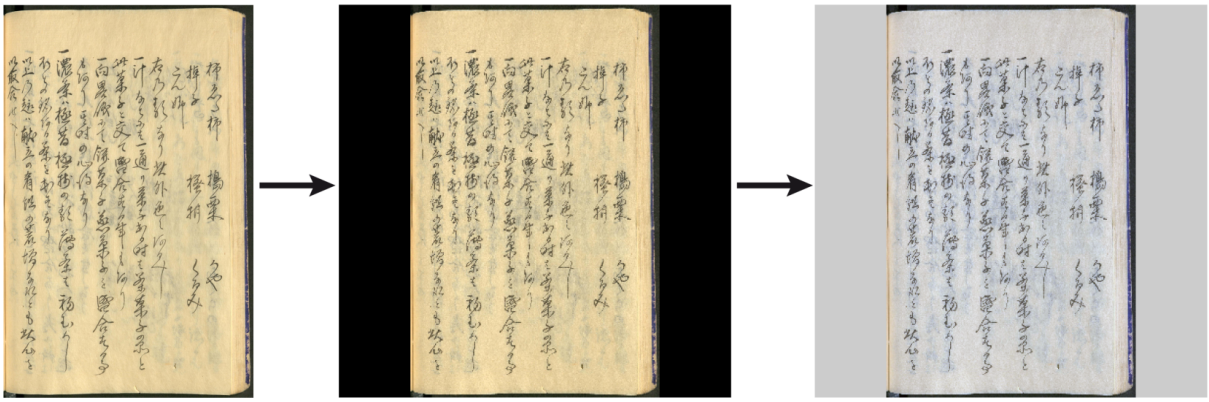


Figura 5.7: Três imagens que representam o pré-processamento de uma página. A página original está disposta à esquerda, enquanto a formatação para o padrão 400×400 e a normalização da iluminação estão dispostas, respectivamente, ao centro e à direita.

O processo de extração dos centros é realizado diretamente pelo modelo U-Net, que

aprende a destacar as regiões centrais de cada caractere como pontos. Esses centros identificados são utilizados como centroides para o k-médias, que associa os pixels da região de caractere ao centroide mais próximo. Assim, para cada conjunto de pixels associado a um centro, é calculada uma caixa delimitadora para o caractere identificado com base nos pixels pertencentes ao *cluster*, como ilustrado na Figura 5.9.

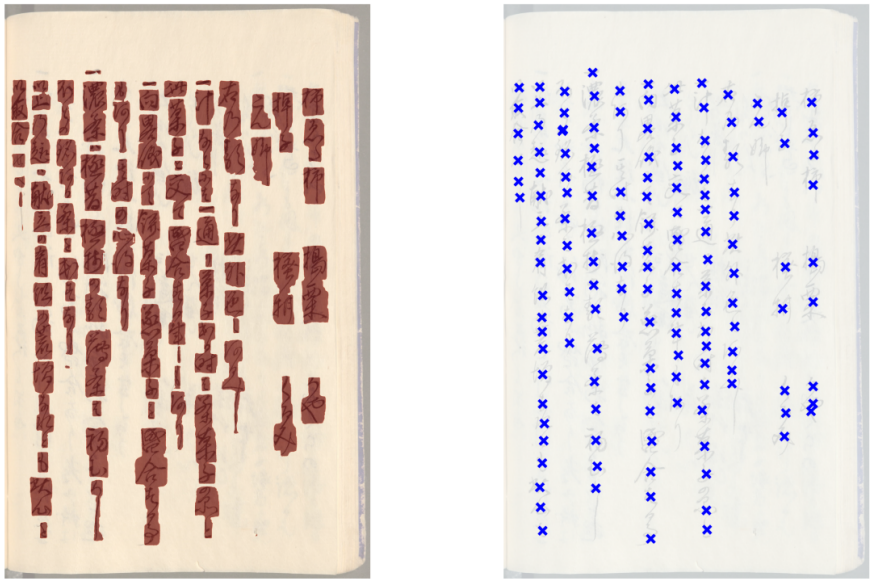


Figura 5.8: Regiões estimadas pelo modelo U-Net. À esquerda, região da imagem que contém caracteres. À direita, estimativa dos centroides encontrados na página, isto é, posição dos caracteres.

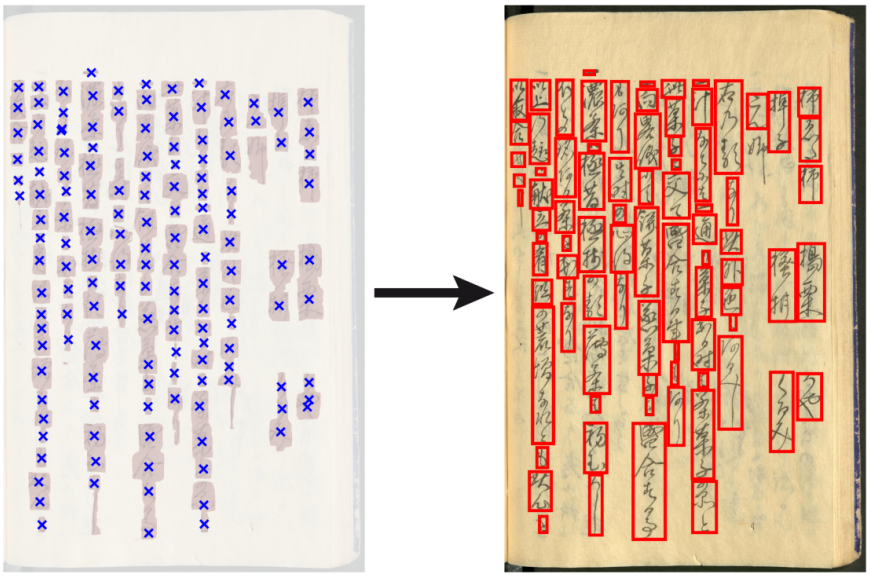


Figura 5.9: À esquerda, regiões da imagem que contém caracteres e sua estimativa dos centroides. À direita, *bounding boxes* estimadas após o agrupamento pelo k-médias para cada centroide.

Com o intuito de avaliar a qualidade das caixas de delimitação geradas, uma métrica de similaridade, conhecida como coeficiente de Dice, foi utilizada para medir a sobre-

posição entre as caixas de delimitação preditas e as reais. Essa métrica quantifica o grau de interseção entre dois conjuntos, que é calculada como o dobro da área de interseção dividido pela soma das áreas individuais, de forma que valores mais próximos de 1 indicam maior similaridade entre as regiões comparadas. O modelo obteve um coeficiente de Dice médio de 95.1% sobre o conjunto de validação e, sendo assim, tínhamos indícios de que as regiões detectadas pelo modelo eram, em geral, adequadas em relação aos caracteres reais.

5.2.2 Classificação de novas páginas de texto

Após a extração dos caracteres de novas páginas de texto, o objetivo principal seria classificá-los com base na rede neural ajustada pelo Kuzushiji-49. Como a base de dados está no formato MNIST, isto é, imagens monocromáticas com tamanho 28×28, faz-se necessário, inicialmente, transformar os caracteres extraídos para o formato esperado. Além disso, todo o processo de avaliação do modelo considera apenas os 49 caracteres do *hiragana*, não sendo possível aplicá-lo em novas páginas não rotuladas – para isso, seria necessário ajustar também um modelo de classificação para os 3823 ideogramas chineses observados no Kuzushiji-Kanji, que não é o foco deste trabalho.

O filtro aplicado no processo de transformação das imagens extraídas para o formato MNIST consiste em duas etapas principais: realce de contraste e remoção de ruído. Na etapa de realce, a imagem original é combinada com uma versão borrada dela mesma, de forma que os traços finos dos caracteres sejam realçados e as sombras ou variações de iluminação sejam reduzidas. Em seguida, aplica-se um algoritmo de remoção de ruído que suaviza algumas regiões da imagem de forma que as bordas e detalhes importantes sejam preservados. Por fim, a imagem é binarizada com base em um *threshold* de intensidade dos pixels, que foi definido como 180 com base em validação cruzada sobre o conjunto de caracteres extraídos. Como ilustrado na Figura 5.10, todos os pixels com intensidade abaixo desse valor foram convertidos para preto (0) e, caso contrário, para branco (255), a fim de adequar o formato final das imagens ao padrão monocromático do MNIST.

Observa-se que, pela Tabela 5.7, os resultados apresentados revelam um decréscimo expressivo de desempenho na aplicação do modelo de classificação treinado no conjunto Kuzushiji-49 diretamente aos caracteres extraídos de novas páginas. Esse fenômeno pode ser atribuído a um desvio de distribuição (*distribution shift*, em inglês) entre os dados de treinamento e os dados extraídos, isto é, as imagens dos caracteres extraídos podem

Figura 5.10: Processo de filtragem da imagem do caractere extraído para o formato MNIST.

ter variações significativas em termos de estilo de escrita, espessura de traço e ruído comparadas às imagens do conjunto Kuzushiji-49, o que afeta a generalização do modelo ajustado. Além disso, a própria forma como os dados extraídos foram tratados para o formato MNIST pode ter influenciado nesse desvio de distribuição observado.

A acurácia geral obtida pela rede neural ajustada foi de 69.1% e o *F1-Score* obtido foi de 68%, uma performance muito abaixo do esperado quando comparada ao desempenho no conjunto de teste observado na Tabela 5.6. Sendo assim, a utilização de um modelo ajustado pelo Kuzushiji-49 para gerar predições em novas páginas de texto não foi satisfatória e, considerando o desvio de distribuição, uma abordagem lógica seria que o próprio modelo fosse ajustado pelos caracteres extraídos.

Tabela 5.7: Desempenho da segunda arquitetura de CNN treinada pelo conjunto Kuzushiji-49 em 10 classes do *hiragana* presentes no conjunto de teste e no conjunto de caracteres extraídos.

Caractere	**Conjunto de teste**		**Conjunto extraído**	
	Acurácia	***F1-Score***	**Acurácia**	***F1-Score***
お (o)	0.926	0.941	0.772	0.543
き (ki)	0.949	0.931	0.744	0.603
す (su)	0.925	0.915	0.471	0.543
つ (tsu)	0.927	0.945	0.623	0.497
な (na)	0.889	0.883	0.834	0.589
は (ha)	0.976	0.928	0.565	0.705
ま (ma)	0.856	0.906	0.457	0.523
や (ya)	0.927	0.931	0.655	0.375
れ (re)	0.920	0.943	0.778	0.741
を (wo)	0.946	0.950	0.900	0.773
Média	0.923	0.927	0.675	0.586

Apesar do desempenho geral não satisfatório, observa-se que a performance do modelo ajustado pelo Kuzushiji-49 nos caracteres que compõem o Kuzushiji-MNIST é muito boa para o conjunto de teste, com uma acurácia média de 0.923. Ainda que esse modelo tenha sido ajustado para classificar todas as 49 classes do *hiragana*, observa-se que, quando comparado à performance do modelo ajustado pelo Kuzushiji-MNIST na Tabela 5.5,

sua performance ainda é relativamente muito próxima para alguns caracteres. Assim, há indícios de que a inclusão de mais classes no modelo, por si só, não seria um fator determinante para a queda de desempenho observada nos dados extraídos.

Além disso, observa-se que, por exemplo, para uma das predições do caractere “お” (o) na Figura 5.11, a rede neural convolucional detectou algumas características semelhantes com o caractere “な” (na). Assim, da mesma forma que a presença das variantes históricas (*hentaigana*) pode influenciar negativamente o poder preditivo do modelo, diversos aspectos das representações modernas do *hiragana* impactam nosso modelo, como formatos ou traços parecidos.

Figura 5.11: Exemplo de predição incorreta para o caractere “お”.

Capítulo 6

Considerações finais

Na primeira etapa do trabalho, foram analisados três modelos de classificação tradicionais aplicados a dados de caracteres japoneses cursivos e foi realizada uma comparação de seus desempenhos com um modelo de aprendizado profundo. Ambos os métodos de classificação tradicionais, abordados no Capítulo 3, e métodos de aprendizado profundo, abordados no Capítulo 4, foram avaliados por diferentes métricas de desempenho previamente discutidas na Seção 2.1.4, como acurácia, precisão, *recall* e *F1-Score*.

Para um conjunto de dados balanceado de 10 caracteres cursivos manuscritos do silabário *hiragana*, observou-se que as redes neurais convolucionais (CNNs) apresentaram as melhores performances, com uma acurácia de 95.7%. Verificou-se também, por meio da aplicação desses métodos em dados balanceados de dígitos manuscritos, que há uma maior dificuldade intrínseca aos dados de caracteres japoneses devido às formas complexas do *hiragana* – a melhor arquitetura de CNN para esses dados de caracteres obteve uma acurácia de 99.4% nos dados de dígitos manuscritos. Além disso, na seleção da melhor CNN a ser comparada com os métodos tradicionais, foram avaliadas três possíveis arquiteturas e destaca-se que, nos dados de caracteres japoneses cursivos, os SVMs e os modelos de *boosting* alcançaram uma performance muito próxima de duas das arquiteturas de CNN avaliadas.

Semelhantemente, para o conjunto de dados desbalanceado com todos os 49 caracteres do silabário *hiragana*, a segunda arquitetura de CNN definida obteve as melhores performances com relação às métricas observadas. Com 92.9% de acurácia e 92.2% de *F1-Score*, a rede neural ajustada com a segunda arquitetura obteve um desempenho próximo ao que fora observado nos outros métodos para o conjunto balanceado de 10 caracteres, o que demonstrou uma robustez do modelo mesmo em cenários de alta variabilidade e

desbalanceamento nas classes, já que manteve um bom nível de generalização.

A segunda etapa do trabalho consistiu no processo de extração e classificação de caracteres provenientes de novas páginas de textos manuscritos em japonês. Com base no conjunto de dados da competição *Kuzushiji Challenge*, foi possível ajustar um modelo U-Net que fosse capaz de detectar os centroides dos caracteres pertencentes à nova página e aplicar um método de agrupamento nas regiões de pixels classificados como caractere. Dessa forma, as caixas de delimitação obtidas alcançaram um coeficiente de Dice médio de 95.1%, o que nos deu indícios de que esse modelo alcançava uma performance suficiente para o trabalho.

Após a extração dos caracteres, foi aplicado um processamento para que os mesmos estivessem no formato MNIST, tendo em vista que o modelo de classificação foi ajustado pelo conjunto Kuzushiji-49. Apesar da ótima performance do modelo no conjunto de teste do Kuzushiji-49, seu desempenho em dados reais extraídos de páginas manuscritas não foi satisfatório, tendo em vista que obteve uma acurácia média de 69.1%. Em geral, essa queda pode ter ocorrido devido às diferenças entre o processamento utilizado para converter as imagens no formato MNIST e o processamento que foi utilizado para a geração do conjunto Kuzushiji-49.

6.1 Abordagens exploradas e possíveis extensões

Uma possível melhoria para o trabalho pode envolver a experimentação com outras arquiteturas profundas, além das CNNs convencionais que foram utilizadas. Modelos mais modernos, como a própria ResNet que foi utilizada no modelo de extração, poderiam potencialmente melhorar a capacidade de generalização do classificador. Além disso, a aplicação de *autoencoders* poderia ser explorada para a redução de dimensionalidade e representações mais robustas dos caracteres, tendo em vista a presença de ruídos nas imagens extraídas.

Adicionalmente, como a base Kuzushiji-49 cobre apenas os 49 caracteres do silabário *hiragana*, a utilização do conjunto Kuzushiji-Kanji, que contém milhares de ideogramas chineses (*kanji*) e suas variantes, permitiria treinar modelos mais abrangentes e mais representativos do domínio real dos documentos digitalizados. Se um modelo satisfatório ajustado com esses dados fosse obtido, seria possível lidar efetivamente com novas páginas de texto não rotuladas que contenham caracteres fora do *hiragana*. Além disso, apesar do

algoritmo U-Net ter sido utilizado para a etapa de extração dos caracteres, o algoritmo CenterNet (Duan *et al.*, 2019) também foi implementado, mas os resultados obtidos não foram satisfatórios para a etapa de extração do trabalho.

É possível também que o modelo tenha sofrido *overfitting* ao conjunto de teste do Kuzushiji-49, especialmente devido ao alto desempenho alcançado anteriormente, e uma abordagem para mitigar esse problema seria realizar um *fine-tuning* diretamente nos dados reais extraídos. Além disso, seria interessante a inclusão de variáveis que incorporem informações de contexto linguístico, como, por exemplo, a frequência condicional de ocorrência dos caracteres – ou seja, a probabilidade do próximo caractere, dado o atual. Isso poderia ajudar o classificador a lidar melhor com casos ambíguos, especialmente quando as imagens extraídas estiverem ruidosas ou apresentarem formatos semelhantes a mais de uma classe, de tal forma que funcione como uma regularização semântica sobre as predições do modelo.

Referências Bibliográficas

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. Em *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*, páginas 177–186. Springer.

Chen, T. e Guestrin, C. (2016). Xgboost: A scalable tree boosting system. Em *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, página 785–794. ACM.

Clanuwat, T. (2022). miwo: App for ai kuzushiji recognition. Disponível em: http://codh.rois.ac.jp/miwo/.

Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K. e Ha, D. (2018). Deep learning for classical japanese literature. *CoRR*.

Cortes, C. e Vapnik, V. (1995). Support-vector networks. *Mach. Learn.*, **20**(3), 273–297.

Cover, T. e Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, **13**(1), 21–27.

Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q. e Tian, Q. (2019). Centernet: Keypoint triplets for object detection.

Frellesvig, B. (2010). A history of the japanese language.

Freund, Y. e Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, **55**(1), 119–139.

Goodfellow, I., Bengio, Y. e Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Hastie, T., Tibshirani, R. e Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics. Springer. ISBN 9780387848846.

He, K., Zhang, X., Ren, S. e Sun, J. (2015). Deep residual learning for image recognition.

Izbicki, R. e dos Santos, T. M. (2020). *Aprendizado de máquina: uma abordagem estatística*. ISBN 978-65-00-02410-4.

James, G., Witten, D., Hastie, T. e Tibshirani, R. (2014). *An Introduction to Statistical Learning: with Applications in R*. Springer Texts in Statistics. Springer New York. ISBN 9781461471370.

Khan, A., Sohail, A., Zahoora, U. e Qureshi, A. S. (2020). A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, **53**(8), 5455–5516.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. e Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, **1**(4), 541–551.

Lecun, Y., Bottou, L., Bengio, Y. e Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, **86**(11), 2278–2324.

LeCun, Y., Cortes, C. e Burges, C. (1998). MNIST handwritten digit database. Disponível em: https://yann.lecun.com/exdb/mnist.

Li, H., Yue, X., Wang, Z., Wang, W., Tomiyama, H. e Meng, L. (2021). A survey of convolutional neural networks —from software to hardware and the applications in measurement. *Measurement: Sensors*, **18**.

Majidifard, H., Adu-Gyamfi, Y. e Buttlar, W. G. (2020). Deep machine learning approach to develop a new asphalt pavement condition index. *Construction and Building Materials*, **247**, 118513.

Mohri, M., Rostamizadeh, A. e Talwalkar, A. (2012). *Foundations of Machine Learning*. Adaptive Computation and Machine Learning series. MIT Press. ISBN 9780262018258.

O'Shea, K. e Nash, R. (2015). An introduction to convolutional neural networks.

Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural networks*, **12**(1), 145–151.

Ronneberger, O., Fischer, P. e Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation.

Shalev-Shwartz, S. e Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, USA. ISBN 1107057132.

Shikibu, M. (s.d.). Genjimonogatari (源氏物語). Disponível em: http://codh.rois.ac.jp/pmjt/book/200003803/.

Sommerschield, T., Assael, Y., Pavlopoulos, J., Stefanak, V., Senior, A., Dyer, C., Bodel, J., Prag, J., Androutsopoulos, I. e de Freitas, N. (2023). Machine Learning for Ancient Languages: A Survey. *Computational Linguistics*, **49**(3), 703–747.

Takashiro, K. (2013). Notation of the japanese syllabary seen in the textbook of the meiji first year. *The bulletin of Jissen Women's Junior College*, **34**, 109–119.

Twine, N. (1983). Toward Simplicity: script reform movements in the Meiji period. *Monumenta Nipponica*, **38**(2), 115–132.

Wang, Y., Li, F., Sun, H., Li, W., Zhong, C., Wu, X., Wang, H. e Wang, P. (2020). Improvement of mnist image recognition based on cnn. Em *IOP conference series: earth and environmental science*, volume 428, página 012097. IOP Publishing.