Image classification of Kuzushiji ideograms with Convolutional Neural Networks

Lucimara R. Martins, Mário Muramatsu Junior, Adriane B. S. Serapião

[1]Universidade Estadual Paulista (UNESP)
Instituto de Geociências e Ciências Exatas (IGCE)
Departamento de Estatística, Matemática Aplicada e Computação (DEMAC)
Rio Claro – SP – Brasil

lucimara.martins@gmail.com, junior_muramatsu@hotmail.com,
adriane.serapiao@unesp.br

Abstract. *This paper evaluates Deep Learning and Machine Learning algorithms in the classification of image datasets of Japanese-language ideogram, written in cursive Kuzushiji style, which is in disuse in Japan but has been widely used for centuries in manuscripts. There is a challenge to build a model for automatically transcribing ancient Kuzushiji into modern Japanese characters, in order to retrieve historical documents. This work aims to contribute with the purpose of identifying Kuzushiji ideograms in their variant cursive styles, using Convolutional Neural Networks. The results were compared with a baseline work and other Machine Learning classification algorithms, obtaining better results than the baseline.*

1. Introdução

Desde 1603, o Japão viveu sob o domínio do xogunato Tokugawa que manteve uma política de isolamento do país em relação ao ocidente. Durante a década de 1850, tal isolamento foi suspenso após pressão americana para abertura dos portos e da economia japonesa ao mercado internacional.

A Restauração consolidada pelo Imperador Meiji foi o processo de derrubada do xogunato e restabelecimento do poder da família imperial japonesa. Durante a restauração Meiji os líderes japoneses implementaram diversas reformas para por fim aos privilégios de classes. Uma modernização importante foi a do sistema educacional elaborada de forma a garantir a integração da população com o projeto do novo governo. Eliminou-se as diferenças de dialetos e exclui-se o ensino da letra cursiva Kuzushiji do currículo oficial. Sendo assim, a maioria do povo japonês é incapaz de ler documentos de mais de 150 anos atrás, documentos estes repletos de sua cultura e valores de seus ancestrais. Embora haja uma preocupação de preservação da história e cultura através da digitalização de documentos históricos, todo esse conhecimento é restrito a uma pequena porção da população que ainda aprendeu o Kuzushiji com seus antepassados.

Atualmente, há um projeto, o *The Kuzushiji Project*[1], que vem desenvolvendo e treinando aplicativos para o aprendizado do Kusushiji [Hashimoto et al. 2017] e, embora esses artigos encorajem pesquisas para desenvolvimento de modelos para classificação

[1]Disponível em: http://www.digitalhumanities.org/dhq/vol/11/1/000281/000281.html

desses caracteres, foram encontrados poucas aplicações sobre este tema. Existem disponíveis conjuntos de dados públicos de ideogramas Kuszushiji, criados pelo Instituto Nacional de Literatura Japonesa (*National Institute of Japanese Literature* – NIJL) a partir de documentos históricos orientais. Atualmente, há três conjuntos de dados baseados no Kuzushiji[2], compostos por imagens de caracteres escaneados de 35 livros clássicos impressos no século 18 e pré-processados.

O conjunto mais básico (Kuzushiji-MNIST ou KMNIST) é composto apenas por imagens em tons de cinza e resolução 28×28, de 10 classes de caracteres que correspondem aos ideogramas representativos dos romajis *o, ki,su,tsu,na,ha,ma,ya,re* e da partícula *o (wo)* (Figura 1a).

Um segundo conjunto de dados, Kuzushiji-49, contém 49 ideogramas, sendo 48 símbolos correspondentes ao hiragana e uma marca de interação (Figura 1b). No entanto, esse conjunto de dados é não balanceado devido às diferentes frequências de ocorrência nos livros. Por fim, há o conjunto de dados contendo 3832 *kanjis*, denominado Kuzushiji-Kanji. Neste último, o número de amostras por classe varia de centenas a apenas uma e, por isso, é tido como um conjunto de dados criado mais para experimentos do que para reconhecimento e classificação dos caracteres.

Para estimular a utilização do conjunto de dados Kuzushiji, o KMNIST foi criado com um formato similar ao do conjunto de dados MNIST (*Modified National Institute of Standards and Technology*) [Lecun et al. 1998], restringindo-se o número de ideogramas para 10, que é o mesmo número de classes do MNIST, e com o mesmo número de amostras. O MNIST é um grande banco de dados de dígitos manuscritos comumente usado para o treinamento de vários sistemas de processamento de imagens. É composto por imagens representativas dos algarismos de 0 a 9, escritos por estudantes do ensino médio americano. Embora o problema de classificação do MNIST esteja efetivamente resolvido, principalmente pelo uso de métodos de *Deep Learning* (DL) [Goodfellow et al. 2016], os quais obtiveram um alto grau de acurácia na identificação dos dígitos (acima de 99,7% de acerto) [Ciresan et al. 2012], o conhecimento disponibilizado na literatura científica sobre na resolução desse problema pode ser usado como base para aplicação em outros problemas de imagens similares, como é o caso do conjunto de dados Kuzushiji.

Dentre as abordagens de DL nas tarefas de visão computacional e processamento de imagens, as redes neurais convolucionais (*Convolutional Neural Networks* – CNN) [Rawat and Wang 2017] têm mostrado resultados bastante efetivos na classificação de imagens, sendo usadas em diversos tipos de aplicações. É o modelo mais comumente utilizado em DL, sendo uma versão regularizada de redes neurais artificiais (RNA) do tipo perceptron multicamadas (*Multi-Layer Perceptron* – MLP) [Haykin 2009].

MNIST foi exaustivamente explorado com CNN. O conjunto Kuzushiji é mais difícil que o MNIST, pois não possui padrão comum de escrita e inclui dezenas de estilos e formatos para um mesmo caracter. Entretanto, não há na literatura uma quantidade relevante de trabalhos que tratem esse problema. O artigo de [Clanuwat et al. 2018] introduziu o problema de classificação do Kuzushij e apresentou resultados de classificações de referência usando o algoritmo K vizinhos mais próximos (K*-Nearest Neighbors* – KNN) [Fix et al. 1951], CNN e redes residuais (*Residual Nets – ResNet*) [He et al. 2016].

[2]Disponível em: https://github.com/rois-codh/kmnist

Figura 1. Ideogramas do Kuzushiji.

(a) Romajis (10 ideogramas) em diferentes estilos cursivos

(b) Hiragana (48 ideogramas)

Fonte: Imagens adaptadas de a) [Clanuwat et al. 2018] e b) Wikipedia[3].

No presente trabalho, avaliamos o desempenho de modelos propostos de CNN, KNN e MLP na classificação dos dois primeiros conjuntos de dados de Kuzushiji e comparamos com o trabalho de referência. Também foi explorado o uso de extração de características através da Análise de Componentes Principais (*Principal Component Analysis* – PCA) [Pearson 1901] para os métodos de Aprendizado de Máquina (AM). Esperamos contribuir na melhoria dos modelos para esta tarefa.

O artigo é composto da seguinte forma. A Seção 2 destaca o trabalho correlato que serviu como referência para a comparação dos resultados da presente pesquisa. A Seção 3 descreve os procedimentos metodológicos adotados nos experimentos. Os resultados são apresentados na Seção 4. Por fim, a Seção 5 apresenta as conclusões e os trabalhos futuros.

2. Trabalho correlato

Em [Clanuwat et al. 2018] é apresentado uma comparação dos resultados de algumas classificações já realizadas para os conjuntos de dados KMNIST e Kuzushiji-49 com os resultados para o conjunto de dados MNIST tradicional. Basicamente, resumem-se a aplicações do algoritmo KNN, de uma CNN simples e de diferentes abordagens de *ResNet* nas mesmas configurações para os três conjuntos de dados. A Tabela 1 expressa os resultados obtidos em termos de acurácia.

Tabela 1. Classificações de referência para os conjuntos KMNIST e Kuzushiji-49 [Clanuwat et al. 2018].

Método	**MNIST**	**KMNIST**	**Kuzushiji-49**
4-Nearest Neighbour baseline	97,14%	91,56%	86,01%
Keras simple CNN benchmark	99,06%	95,12%	89,25%
PreActResNet-18	99,56%	97,82%	96,64%
PreActResNet-18 + Input Mixup	99,54%	98,41%	97,04%
PreActResNet-18 + Manifold Mixup	99,54%	98,83%	97,33%

O KNN foi implementado considerando 4 vizinhos e distância euclidiana. A CNN consistia de duas camadas de convolução, a primeira composta por 32 filtros e a segunda por 64 filtros, ambas com função de ativação ReLU e tamanho dos filtros 3×3. Na sequência. seguia-se uma camada de *MaxPooling* de tamanho 2×2, um *dropout* de 25%, linearização das saídas, camada completamente conectada com 128 neurônios e função de ativação ReLU, mais um *dropout* de 50% e outra camada completamente conectada com 10 (ou 49) neurônios com a função *softmax*. Foram utilizados o otimizador Adadelta, 12 épocas e um *batch* de tamanho 128 para o treinamento.

Os modelos utilizando *ResNet* consistiam de três abordagens distintas. A primeira, usando um modelo de redes residuais típico. A segunda, adicionando o *mixup*, uma forma de minimização de risco vicinal, que treina em exemplos virtuais construídos como a interpolação linear de dois exemplos aleatórios do conjunto de treinamento e seus rótulos, melhorando a capacidade de generalização da rede neural. A terceira, usando este modelo incorporando um regularizador denominado *Manifold Mixup*, o qual encoraja a rede neural a predizer com menos confiança nas interpolações de representações ocultas, produzindo fronteiras de decisão mais suaves. Estes modelos não foram implementados no presente artigo.

Não foram encontrados outros trabalhos correlatos na literatura sobre o conjunto de dados Kuzushiji.

3. Metodologia

Os experimentos realizados foram implementados na linguagem Python com a plataforma Keras [Chollet 2015] para a construção da CNN e com o pacote Scikit-learn para o algoritmo KNN e da rede neural MLP. A seguir, detalham-se os aspectos metodológicos adotados.

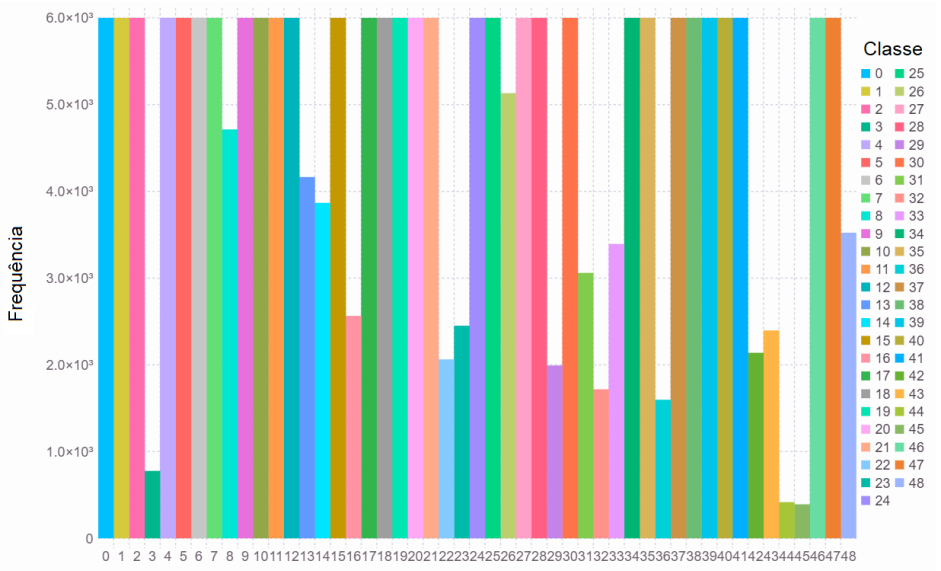
3.1. Conjuntos de dados

Dois conjuntos de dados foram utilizados nos experimentos, o KMNIST e o Kujushiji-49, a fim de comparar os resultados com o trabalho de referência [Clanuwat et al. 2018].

O KMNIST contém imagens em escala de cinza de tamanho 28×28 pixels, contendo 70.000 instâncias de 10 classes. O conjunto de dados é divididos em 60.000 instâncias de treinamento e 10.000 instâncias de teste, ambos balanceadas. Ou seja, cada uma das 10 classes contém 6000 imagens no conjunto de treinamento e 1000 imagens no conjunto de teste.

O conjunto de dados Kuzushiji-49, considerando 49 ideogramas, contém 270.912 instâncias e não é balanceado, devido à dificuldade de encontrar amostras dos ideogramas mais raros nos livros antigos. O conjunto de treinamento é composto por 232.365 instâncias e o de teste por 38.547 instâncias. No conjunto de treinamento, apenas 30 classes possuem 6000 instâncias. Há três classes com menos de 1000 imagens. O mesmo ocorre no conjunto de teste, sendo que 31 classes possuem 1000 instâncias e há duas com menos de 100 instâncias. A numeração das classes segue a ordem de leitura dos ideogramas (*a*, *ka*, *sa*, *ta*, *na*, *ha*, *ma*, *ya*, *ra*, *wa*, *n*). A Figura 2 mostra o histograma de distribuição das classes no conjunto total de imagens do Kuzushiji-49.

Figura 2. Histograma de distribuição de classes do conjunto de dados Kuzushiji-49.



Fonte: Imagem adaptada de https://www.simonwenkel.com/2018/12/18/Kuzushiji-MNIST.html.

3.2. Métodos e algoritmos de classificação

No trabalho de referência foram avaliados os desempenhos de uma CNN simples, de modelos *Resnets* e do algoritmo KNN na classificação de ideogramas Kuzushiji. Abordagens de DL para classificação de imagens realizam extração de características significativas das imagens de forma hierárquica e automaticamente, sem a necessidade explícita de uma fase anterior de extração de características. Já com os métodos de AM, como KNN e RNA, a engenharia de extração de características é feita de forma manual, com o pré-processamento das imagens para produzir entradas mais relevantes.

Assim, no presente trabalho, investigou-se o desempenho de uma CNN em relação aos métodos de AM, sem e com extração de características das imagens, comparando também os resultados com o trabalho de referência. Ressalta-se que, embora dois métodos

(CNN e KNN) sejam os mesmos do trabalho de referência, as configurações dos modelos são distintas.

3.2.1. K vizinhos mais próximos

O KNN é um algoritmo amplamente empregado para reconhecer padrões. Ele classifica uma certa amostra de acordo com as respectivas classes de seus K vizinhos mais próximos, os quais pertencem a um certo conjunto de dados. O método KNN atribui uma classe para uma dada amostra baseado na classificação majoritária dos K vizinhos mais próximos a uma dada amostra.O algoritmo calcula a distância da amostra dada para cada amostra do conjunto de dados, ordenando as amostras do conjunto do mais próximo ao de maior distância. Após a ordenação, selecionam-se os K primeiros e elege-se a classe que predomina nos vizinhos como a classe da amostra dada.

Sendo assim, é necessário otimizar tanto a quantidade de vizinhos mais próximos que serão considerados na classificação quanto o tipo de medida de distância a ser utilizada. Desta forma, diferentes testes foram conduzidos utilizando diversos tipos de medidas para o cálculo da distância (euclidiana, Camberra, Hamming e Bray-Curtis) e variando a quantidade de vizinhos, de 3 a 20 vizinhos. Os melhores resultados foram obtidos utilizando a distância Bray-Curtis e os 3 vizinhos mais próximos, os quais foram adotados como os parâmetros do algoritmo para efeito de comparação com os outros métodos.

3.2.2. Rede neural perceptron multicamadas

Uma rede neural *feedforward* do tipo MLP é um grafo acíclico direcionado, composto por uma camada de entrada, uma camada de saída e uma ou mais camadas escondidas (ou ocultas) [Haykin 2009]. Cada nó na camada é um neurônio, que pode ser considerado como a unidade básica de processamento de uma rede neural. O neurônio funciona em duas etapas: calcula a soma ponderada de suas entradas e, em seguida, aplica uma função de ativação para normalizar a soma. As funções de ativação podem ser lineares ou não lineares. Além disso, existem pesos associados a cada entrada de um neurônio. Os pesos são os parâmetros que a rede tem que aprender durante a fase de treinamento.

A camada de entrada é usada para fornecer os dados ou características de entrada para a rede. A camada de saída é a camada que realiza as previsões. A camada escondida produz funções complexas através da cascata de funções mais simples, de modo a transformar as entradas em algo que a camada de saída possa usar na predição. No treinamento da rede neural, os dados são passados pela rede, computados pelos neurônios em todas as camadas, e a saída obtida é comparada com a saída real. O erro entre as duas saídas é usado para alterar os pesos dos neurônios, de modo que o erro diminui gradualmente e a rede aprende os padrões dos dados.

A arquitetura proposta da rede MLP para os experimentos provê uma camada de entrada, com o número de neurônios igual ao número de pixels da imagem, uma única camada escondida com 500 neurônios e uma camada de saída com número de neurônios igual ao número de classes. A função de ativação ReLU foi usada nas camadas de entrada e escondida. A camada de saída usou a função de ativação *softmax*. O otimizador empre-

gado foi o Adam e o tamanho do *batch* foi determinado automaticamente. Os parâmetros foram encontrados por experimentação.

3.2.3. Rede neural convolucional

As CNNs são provavelmente o modelo de *Deep Learning* mais conhecido e utilizado atualmente. O que caracteriza esse tipo de rede é ser composto basicamente de camadas convolucionais que processam as entradas considerando campos receptivos locais. as quais funcionam como extratores de características. Adicionalmente, incluem também operações conhecidas como *pooling*, responsáveis por reduzir a dimensionalidade espacial das representações [Goodfellow et al. 2016].

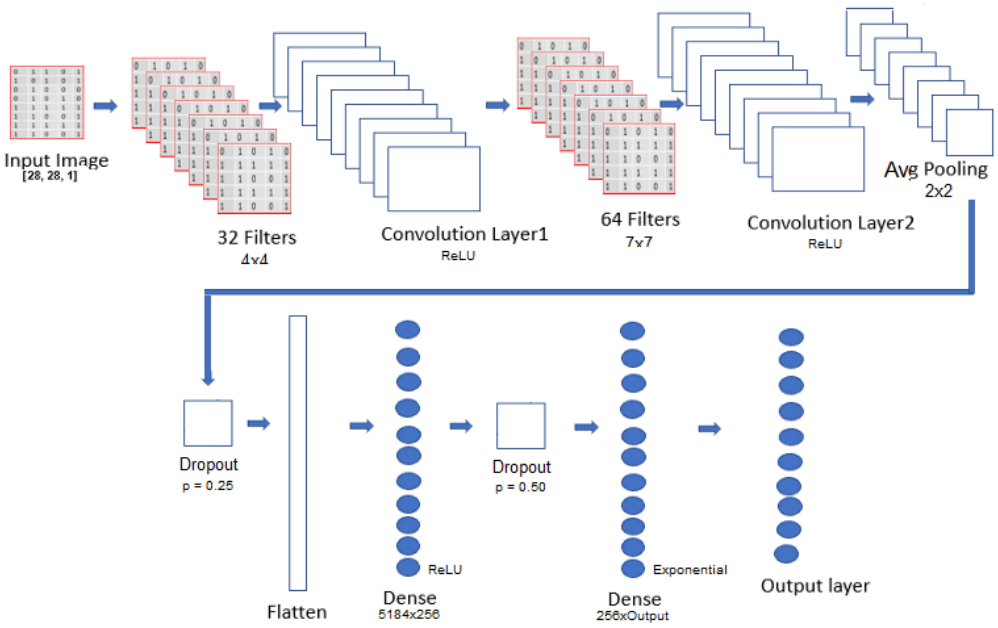
Nos dois métodos descritos anteriormente, como entrada do classificador, as imagens eram apenas vetorizadas, desprezando toda a estrutura espacial e relacionamentos entre seus pixels vizinhos. Na camada convolucional, cada neurônio é um filtro aplicado a uma imagem de entrada. Um filtro é uma matriz de pesos que faz uma combinação linear dos pixels vizinhos e, por isso, considera-se que a convolução filtra as imagens levando em conta sua estrutura bidimensional. Cada representação gerada por um filtro da camada convolucional é conhecida como "mapa de características". Estes são empilhados formando um tensor com profundidade igual ao número de filtros, o qual será a entrada para a próxima camada da rede.

A camada de *pooling* é utilizada para reduzir a dimensionalidade ao longo das camadas da rede, mas há estudos a favor de não utilizar *pooling* e sim aumentar o passo (*stride*) nas convoluções. A camada de *dropout* aleatoriamente desativa com probabilidade p a ativação de neurônios na fase de treinamento e durante o passo de propagação dos dados pela rede. Ao longo das iterações, isso dá um efeito que minimiza a variância e aumenta o viés das funções e, por isso, foi demonstrado que o *dropout* tem relação com o método de *ensemble bagging*.

Nos estudos do problema com CNN, partimos da implementação disponibilizada para os conjuntos Kuzushiji em questão que, por sinal, é idêntica a utilizada por muitos para o conjunto MNIST. Na versão proposta no presente trabalho, a rede profunda é composta por duas camadas consecutivas de convolução, uma com 32 filtros de *kernel* (4,4) e outra com 64 filtros e *kernel* (7, 7), ambas usando a função de ativação ReLU. Na sequência tem-se um *AveragePooling* de tamanho (2,2) e um *dropout* de p = 0,25. Em seguida, as saídas dessa camada são convertidas em um vetor linearizado e enviadas para uma camada densamente conectada com 256 neurônios, que utiliza a função de ativação ReLU. Essa camada possui um *dropout* com p = 0,5. Por fim, segue-se outra camada densamente conectada com 256 neurônios que usa a função de ativação exponencial. A camada de saída representa as possíveis classes (10 neurônios para KMNIST ou 49 neurônios para Kuzushiji-49). A Figura 3 exibe a arquitetura da CNN proposta.

O treinamento da CNN foi feito com o otimizador Adam, 15 épocas e um *batch* de tamanho 128. Em relação à arquitetura proposta em [Clanuwat et al. 2018], a presente arquitetura diferencia-se pelo tamanhos dos filtros nas camadas de convolução, pelo método de *pooling*, pela função de ativação da última camada densa, pelo otimizador e pelo número de épocas do treinamento. Essa configuração foi determinada através da

Figura 3. Arquitetura do modelo CNN.



Fonte: Elaborada pelos autores.

avaliação de diversas combinações de parâmetros, escolhendo-se a que proporcionou melhores resultados.

3.3. Procedimentos experimentais

Os experimentos consistiram em aplicar os classificadores CNN, KNN e MLP aos dois conjuntos de dados. Para a CNN, considerou-se 10 corridas para cada conjunto e os resultados foram expressos como as médias das métricas de avaliação do classificador. A entrada da CNN era composta pelos valores dos pixels das imagens.

Para os classificadores KNN e MLP, foram levados em conta dois cenários. O primeiro, considerando como entrada direta dos classificadores os valores dos pixels das imagens, ou seja, sem extração de características. É a mesma abordagem utilizada no trabalho de referência.

No segundo cenário, foi utilizada a PCA como ferramenta para a extração de características. PCA encontra os autovetores de uma matriz de covariância com os mais altos autovalores e usa-os para projetar os dados em um novo subespaço de dimensões iguais ou menores, reduzindo o número de características por compressão dos dados. Nos experimentos, as imagens dos conjuntos de dados foram reduzidas utilizando 100 componentes principais. Os cenários com o uso de PCA são designados como PCA+<classificador>.

Em ambos os cenários, o classificador MLP executou 10 corridas e os resultados foram expressos como as médias das métricas de avaliação. Para o KNN, que é um algoritmo estável, bastou apenas uma corrida para o classificador.

Para a validação da aprendizagem da CNN e dos modelos MLP foram retiradas 10% das amostras do conjunto de treinamento.

3.4. Métricas de avaliação

Os resultados dos experimentos foram avaliados segundo as métricas de avaliação típicas da tarefa de classificação. As métricas calculadas [Tan et al. 2016] foram: acurácia, revocação (*recall*), precisão, F1-*score*, índice *Kappa* e AUC (*Area Under Curve*).

A acurácia é a razão entre o número de previsões corretas e o número total de amostras. Precisão é o número de resultados positivos corretos dividido pelo número de resultados positivos previstos pelo classificador. Revocação é o número de resultados positivos corretos dividido pelo número de todas as amostras relevantes (todas as amostras que deveriam ter sido identificadas como positivas). Tanto para a precisão quanto para a revocação, valores mais próximos de 1 representam um melhor modelo.

F1-*score* é a média harmônica entre precisão e revocação. O intervalo para pontuação de F1 é [0, 1]. Ele informa a precisão do seu classificador (quantas instâncias ele classifica corretamente), bem como o quão robusto ele é (não perde um número significativo de instâncias). Alta precisão, mas menor revocação, oferece uma precisão extrema, mas perde um grande número de instâncias que são difíceis de classificar. Quanto maior a pontuação de F1, melhor é o desempenho do classificador.

O índice *Kappa* expressa a medida da diferença entre a concordância das instâncias de referência e a classificação automática e a probabilidade de concordância entre as instâncias de referência e a classificação aleatória. Indica a coesão das instâncias classificadas. Ele varia de -1 a 1 e, quanto mais próximo de 1, maior a precisão da classificador. AUC de um classificador é igual à probabilidade do classificador classificar um exemplo positivo (sensibilidade) escolhido aleatoriamente mais alto do que um exemplo negativo (especificidade) escolhido aleatoriamente. AUC varia no intervalo de [0, 1]. Quanto maior o valor, melhor é o desempenho do classificador.

4. Resultados

A Tabela 2 exibe os resultados da classificação do conjunto de dados KMNIST com os métodos implementados. Nota-se que a nossa versão do KNN (93.16%) superou a versão KNN do trabalho de referência (91,56%) em termos de média de acurácia e desvio padrão (D.P.), assim como também as médias de todas as outras métricas. Em relação ao algoritmo KNN de referência, nossa abordagem se diferenciou na consideração de uma quantidade menor de vizinhos, 3 ao invés de 4. Entretanto, a medida utilizada para o cálculo da distância (Bray-Curtis) entre as instâncias foi o fator que proporcionou um melhor desempenho. Quando nosso modelo teve as entradas pré-processadas com PCA, a acurácia do modelo foi ainda ligeiramente mais alta.

Tabela 2. Métricas de avaliação da classificação para o conjunto KMNIST.

Método	**Acurácia (%) ± D.P.**	***Recall***	**Precisão**	**F1**	***Kappa***	**AUC**
CNN	96,10 ± 0,208	0,961	0,961	0,961	0,957	0,98
KNN	93,16	0,932	0,933	0,932	0,924	0,96
PCA+KNN	93,85 ± 0,037	0,939	0,939	0,939	0,932	0,97
MLP	88,07 ± 1,013	0,881	0,882	0,881	0,867	0,94
PCA+MLP	90,55 ± 0,679	0,906	0,907	0,905	0,895	0,95

Os classificadores MLP e PCA+MLP perderam em todas as métricas de avaliação de todas as abordagens do trabalho de referência e das propostas no presente artigo. Apesar disso, a agregação da PCA elevou a acurácia do modelo MLP, assim como no KNN. A perda final média no treinamento do modelo MLP foi de 0,11241, ao passo que no modelo PCA+MLP foi de 0,07155.

Para a CNN, no trabalho de referência, a acurácia foi de 95,12%, enquanto que no nosso modelo foi de 96,10%. A arquitetura proposta divergiu em vários parâmetros do trabalho de referência e obteve melhor desempenho em todas as métricas de avaliação. Chegou-se aos parâmetros através de exaustivos testes com a variação dos valores. A nossa CNN refletiu melhor as características das imagens, pois utilizou filtros com tamanhos divisores da resolução da imagem. Esse foi o fator principal na melhoria de nossos resultados. A perda final média no treinamento da CNN foi de 0,02040.

No KMNIST, os ideogramas com maiores erros de classificação (Figura 4, à esquerda) foram *su*, *na*, *ha* e *ya*. Esse resultado reflete as semelhanças entre os ideogramas em questão. Entre os ideogramas que comumente eram os mais corretamente classificados (Figura 4, à direita) estão o *tsu*, *ma* e *re*.

Figura 4. Ideogramas com maiores (esquerda) e menores (direita) erros de classificação no conjunto KMNIST.

Os erros de classificação do ideograma *na* foram devidos a confusões com os ideogramas *su* e *ma*. Para o ideograma *ya*, as atribuições equivocadas na classificação foram feitas, sobretudo, com os ideogramas *na* e *wo*. Já para o ideograma *su*, a confusão estabeleceu-se com os ideogramas *ma*, *tsu* e *re*. Por fim, o ideograma *na* apresentou confusão com os ideogramas *su* e *ki*.

Uma vez determinados os melhores parâmetros para os modelos que classificaram o conjunto de dados para o KMNIST, os mesmos valores e funções foram utilizados na aprendizagem dos modelos para a classificação do conjunto de dados Kuzushiji-49. A Tabela 3 indica os resultados das métricas de avaliação dos modelos para esse conjunto.

Tabela 3. Métricas de avaliação da classificação para o conjunto Kuzushiji-49.

Método	**Acurácia (%) ± D.P.**	***Recall***	**Precisão**	**F1**	***Kappa***	**AUC**
CNN	94,55 ± 0,118	0,936	0,942	0,938	0,944	0,97
KNN	86,93	0,850	0,872	0,858	0,866	0,92
PCA+KNN	88,15 ± 0,049	0,864	0,884	0,871	0,879	0,93
MLP	68,99 ± 0,158	0,647	0,701	0,662	0,682	0,82
PCA+MLP	84,16 ± 0,204	0,828	0,834	0,83o	0,838	0,91

Para o KNN com Kuzushizi-49, a nossa implementação (86,93%) produziu uma maior taxa de acerto na classificação que o trabalho de referência (86,01%), a qual foi melhorada com a extração de características com PCA (88,15%). A rede MLP na

classificação mostrou os piores resultados (68,99% de acerto), perdendo de todos os métodos, incluindo o trabalho de referência. Porém, com a aplicação de PCA, houve um aumento considerável na taxa de acerto (84,16%) da classificação. O treinamento da rede MLP produziu uma perda final média de 0,61567 e da PCA+MLP de 0,34956. Os principais erros ocorreram na classificação do ideograma de marcação como sendo o ideograma *ku* e do ideograma *so* atribuído como *u*.

A CNN proposta produziu ganhos consideráveis de acurácia (94,55%) comparada à referência (89,25%). A perda final média no treinamento foi 0,10895. Todas as métricas da CNN foram mais altas que as métricas dos outros experimentos. Os valores obtidos nessas métricas, todos muito próximos a 1, confirmam a boa qualidade do modelo da CNN para a tarefa em questão.

Figura 5. Ideogramas com maiores erros de classificação no conjunto Kuzushiji-49.

A classe com maior erro de classificação no Kuzushiji-49 com CNN foi a referente ao ideograma de marcação, o qual foi 67 vezes confundido com o ideograma (*ku*), devido à semelhança de grafia entre eles. Além dessa, outras classes (Figura 5) foram classificadas erroneamente. Tais classes foram: *ka* (5 vezes), *ki* (6 vezes), *ku* (7 vezes), *shi* (11 vezes), *so* (14 vezes), *fu* (27 vezes) e *he* (28 vezes). Dessas, apenas o ideograma *ka* faz parte do conjunto KMNIST, ou seja, os erros mais vistos no conjunto de dados com apenas 10 ideogramas não apareceram mais ao incluir outros ideogramas e um número maior de classes (49). Ou seja, a discriminação entre as classes parece ter melhorado com a inclusão de novas instâncias.

Observar-se que houve uma pequena perda de desempenho na classificação do Kuzushiji-49 em relação ao KMNIST, mas ainda superior aos relatados na literatura [Clanuwat et al. 2018]. Isso era esperado, uma vez que o conjunto de dados Kuzushiji-49 possui número bem maior de instâncias, número muito maior de classes e é desbalanceado.

5. Conclusões

Neste estudo foi apresentada uma avaliação de desempenho de diferentes algoritmos para classificação de imagens de ideogramas da letra cursiva japonesa. Entre os algoritmos avaliados, tanto o KNN quanto a CNN propostas ultrapassaram os resultados do trabalho de referência. Mostrou-se ainda que os modelos de AM tiveram ganhos de desempenho quando a PCA foi utilizada como extrator de características para alimentar os modelos. Claramente, o pior desempenho do KNN e da rede MLP em relação à CNN deve-se a não considerar os relacionamentos entre pixels vizinhos, o que é suprido pela aplicação de filtros de convolução na CNN, os quais consideram a estrutura espacial da imagem.

Embora a CNN tenha obtido os melhores resultados nos experimentos e tenha sido também melhor que sua equivalente no trabalho de referência, seus resultados foram ainda inferiores aos dos modelos usando *PreActResNet-18* na referência. Tais modelos, além de possuírem mais camadas, têm unidades de pré-ativação que proporcionam efeito de regularização durante o treinamento.

Para trabalhos futuros, propõe-se inserir mecanismos de regularização L2 na CNN e experimentar diferentes números de unidades escondidas nas camadas densas, assim como explorar arquiteturas de redes resilientes.

Referências

Chollet, F. (2015). Keras. `https://github.com/fchollet/keras`.

Ciresan, D., Meier, U., and Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *Proceedings of the 25th IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2012)*, pages 3642–3649.

Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Yamamoto, K., Lamb, A., and Ha, D. (2018). Deep learning for classical japanese literature. *CoRR*, abs/1812.01718.

Fix, E., Hodges, J., and of Aviation Medicine, U. S. (1951). *Discriminatory Analysis: Nonparametric Discrimination, Consistency Properties*. Number v. 1-2 in Discriminatory Analysis: Nonparametric Discrimination, Consistency Properties. USAF School of Aviation Medicine.

Goodfellow, I. J., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press, Cambridge, MA, USA.

Hashimoto, Y., Iikura, Y., Hisada, Y., Kang, S., Arisawa, T., and Kobayashi-Better, D. (2017). The kuzushiji project: Developing a mobile learning application for reading early modern japanese texts. *Digital Humanities Quarterly*, 11(1):1–13.

Haykin, S. S. (2009). *Neural networks and learning machines*. Pearson Education, Upper Saddle River, NJ, third edition.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

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

Pearson, K. (1901). On lines and planes of closest fit to system of points in space. *Phil. Mag. 2*, 6:559–572.

Rawat, W. and Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *Neural Comput.*, 29(9):2352–2449.

Tan, P.-N., Steinbach, M., and Kumar, V. (2016). *Introduction to Data Mining, (Second Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.