Reconhecimento de escrita com redes convolucionais em aplicação web

Guilherme Zanini Moreira
Ciência da Computação
Instituto Federal Catarinense
Videira, Santa Catarina
gzaninimoreira@gmail.com

***Resumo*—O presente trabalho apresenta o reconhecimento de escrita utilizando a arquitetura de redes neurais convolucionais LeNet-5. O conjunto de dados para dígitos MNIST foi utilizado. Uma aplicação *web* utilizando a biblioteca TesnorFlow.js foi construída para detectar a escrita e converter em formato digital.**

***Palavras-chave*—reconhecimento, escrita, tensorflow**

I. Introdução

A tecnologia dos dias de hoje permitiu o armazenamento de informações em máquinas, assim, facilitando a organização e acesso aos dados. Entretanto, uma vez que essa tecnologia não estava presente no passado, muitos dados foram perdidos pois eram armazenados em maneiras tradicionais de armazenar dados, como a escrita. Atualmente, *software* para reconhecimento de escrita são utilizados para reconhecer a escrita e armazenar estes em máquinas, garantindo a facilidade de acessos aos mesmos, como também mais segurança aos dados [1]. Utilizando redes neurais convolucionais (RNC), em [2] é apresentado um reconhecedor de dígitos simples, jé em [3] é apresentado um sistema que realiza a leitura da escrita em cheques. O objetivo do presente trabalho é desenvolver uma aplicação web capaz de reconhecer dígitos utilizando redes neurais convolucionais. Para isso, foi utilizado a arquitetura LetNet-5 e a biblioteca TensorFlow.js para a aplicação web.

II. Redes neurais convolucionais

As redes neurais convolucionais (RNC) são um tipo de redes neural artificial especializadas em processamento de dados com uma topologia em grade (*grid*). Alguns exemplos de dados que podem ser processados pelas RNC são dados de série temporal, as quais podem ser vistos como uma grade de uma dimensão e dados de imagem, que podem ser considerados como uma grade 2D de *pixels* [4]. Em imagens, as RNC são capaz de reconhecer formas bidimensionais com um alto grau de invariância à translação, dimensionamento, inclinação e outras formas de distorção [5]. Ainda de acordo com [5], a capacidade de realizar essas tarefas é realizada através de aprendizado supervisionado de uma rede neural artificial cuja estrutura inclui as seguintes as seguintes etapas:

- Extração de características: cada neurônio recebe seu peso sináptico de um campo receptivo local (janela composta por por *pixels*) localizado na camada anterior. Uma vez que um recurso foi extraído, sua localização exata se torna menos importante, desde que sua posição em relação a outras características seja aproximadamente preservada.
- Mapeamento de características: cada camada computacional da rede é composta de múltiplos mapas de características, em que a forma de cada mapa de característica é um plano dentro do qual os neurônios individuais são obrigados a compartilhar o mesmo conjunto de pesos sinápticos, assim, reduzindo o número de parâmetros livres e garantindo a invariância à translação.
- Subamostragem (*subsampling*): Cada camada convolucional é seguida por uma camada computacional que realiza a média local e subamostragem, em que a resolução do mapa de características é reduzido. Esta operação tem o efeito de reduzir a sensibilidade da saída do mapa de características para translações e outras formas de distorção.

A camada de convolução é o principal "bloco" das RNC, essa camada realiza operações matemáticas para mesclar dois conjuntos de informações. Essa camada é aplicada nos dados de entrada com um filtro para produzir os mapas de características e também é a primeira camada utilizada para extrair as características das imagens. Após uma operação de convolução, normalmente é executado a operação de subamostragem com a camada *pooling*. Esse tipo de operação agrupa as camadas para reduzir a amostra de cada mapa de características de forma independente, reduzindo a altura e largura, mas mantendo a profundidade. O mais comum tipo de *pooling* é o *Max-Pooling*, que apenas leva o valor máximo no espaço de *pool* [1]. Na arquitetura *LeNet-5* é utilizado o *Average Pooling*, o qual faz a média dos valores do espaço de *pool*.

A ideia de convolução seguida por subamostragem é inspirada na noção de células "simples" seguidas por células "complexas". Com as sucessivas camadas computacionais alternando entre convolução e subamostragem, obtêm-se o efeito "bipiramidal". Ou seja, a cada convolucional ou subamostragem camada, o número de mapas de características é aumentado enquanto a resolução espacial é reduzida, em comparação com a camada anterior correspondente [5].

III. ARQUITETURA LENET-5

A LeNet-5 é uma arquitetura de redes neural convolucional de aprendizagem baseada em gradiente e aplicada pela primeira vez com sucesso no reconhecimento digital de caracteres manuscritos. A camada de entrada é uma imagem digital escrita à mão de $0 \sim 9$ com um tamanho de 32×32 pixels, e sua camada de saída tem 10 nós correspondentes a números de $0 \sim 9$. Além das camadas de entrada e saída, geralmente LeNet-5 inclui seis camadas, que são três camadas convolucionais, duas camadas de *pooling* e uma totalmente camada conectada. O tamanho do kernel convolucional é definido como 5×5 na camada convolucional e no kernel na camada de *pooling* é definido como 2×2. A camada de conexão completa reduz o número de neurônios de 120 para 84 para reduzir o treinamento do parâmetro [6]. A Figura 1 representa a arquitetura de LeNet-5, contendo apenas as camadas.

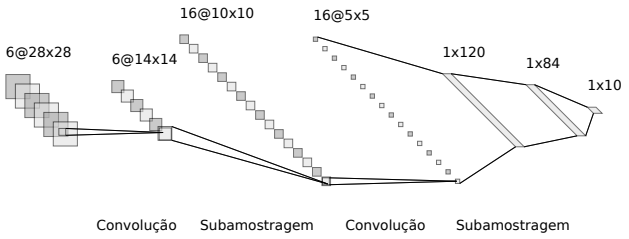

Fig. 1. Arquitetura da LeNet-5
Fonte: Adaptado de [7]

IV. TENSORFLOW.JS

De acordo com [8], o aprendizado de máquina tem proporcionado o melhoramento de software existentes, como também proporcionou novas aplicações. Entretanto, as bibliotecas até então existentes que auxiliam a criação de aplicações utilizando aprendizado de máquina refletem as "raízes" da indústria e da acadêmica, assim, a grande maioria dessas bibliotecas são endereçadas para serem desenvolvidas em linguagens de programação como Python ou C++.

Ainda de acordo com [8], há uma grande comunidade de desenvolvedores JavaScript (JS), dessa forma, o desenvolvimento de uma biblioteca para essa linguagem de programação pode capacitar a comunidade de JS com o aprendizado de máquina e auxiliar a uma nova classe de aplicativos.

Algumas das dificuldades enfrentadas no desenvolvimento de aplicações de aprendizado de máquina com JS é relacionado ao desempenho, uma vez que o JS é uma linguagem interpretada e que não tem a mesma velocidade de linguagens compiladas como o Java e C++. Outra dificuldade do JS é relacionado a segurança, uma vez que aplicações no navegador não têm acesso direito a GPU, onde normalmente acontece a computação numérica de sistemas de aprendizagem profunda [8].

Uma das aplicações que o TensorFlow.js proporcionou é a integração entre vários componentes de hardware do dispositivo, como a câmera, o microfone e o acelerômetro no navegador com modelos de aprendizado de máquina [8]. O TensorFlow.js proporcionou a expensão de modelos de aprendizado de máquina para o navegador, mas também criou plataformas que permitem a integração entre bibliotecas existentes como o TensorFlow e Keras.

Para a conversão de um modelo desenvolvido em TensorFlow ou Keras para TensorFlow.js, o usuário executa um script em Python que converte o formato existente para o formato da web. O modelo gerado é um modelo otimizado "podando" operações de treino.

V. METODOLOGIA

Conforme apresentado na Seção II, as redes neurais convolucionais (RNC) vem sendo amplamente utilizados para o processamento de imagens. Atualmente, várias aplicações permitem a escrita em dispositivos móveis, por exemplo, ou a conversão de imagem com escrita para formato digital. O navegador possibilita a obtenção de dados em tempo real de sensores como a câmera e o acelerômetro, assim, através de uma biblioteca focada do desenvolvimento de modelos de aplicações de aprendizado de máquina utilizando JavaScript (JS) é possível o desenvolvimento de aplicações com integração entre vários componentes de hardware conforme apresentado na Seção IV. Deste modo, é utilizado a biblioteca TensorFlow.js para a conversão de um modelo de rede neural convolucional desenvolvido com a biblioteca Keras, bem como utilizando o TensorFlow.js para carregar o modelo em uma aplicação web.

Nesta seção será abordado as etapas para a elaboração da aplicação web, primeiramente contruindo o modelo com o Keras, exportando para o formato TensorFlow.js e a construção da aplicação utilizando a biblioteca React. Na Figura 2 é apresentado a visão geral do método, onde após a obtenção do conjunto de dados é construído a arquitetura e o treinamento do modelo. A etapa antes da construção da aplicação web é a conversão do modelo para o formato TensorFlow.js.

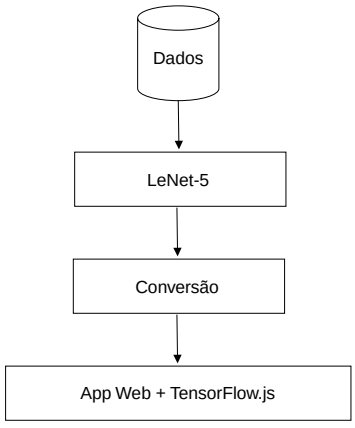

Fig. 2. Visão geral do método

- Conjunto de dados: para o treinamento e avaliação do modelo será utilizado o conjunto de dados MNIST (*Mod-*

ified National Institute of Standards and Technology), composto por 60.000 imagens para treino e 10.000 imagens para teste. As imagens do conjunto de dados possuem 28×28 pixel.

- Arquitetura: a arquitetura utilizada no trabalho é a arquitetura LeNet-5, apresentada na Seção III.
- Conversão: após o desenvolvimento e treino do modelo é realizdo a etapa de conversão, onde primeiramente é utilizado o método de salvar modelo do Keras, salvando-o em formato HDF5. A próxima etapa é converter o modelo de HDF5 para formato TensorFlow.js utilizando o script disponibilizado pela biblioteca.
- Desenvolvimento da aplicação web: a aplicação web contêm o elemento HTML *canvas*, destinado a delimitar uma área para renderização dinâmica de gráficos, onde o usuário vai escrever o dígito. Para o desenvolvimento da aplicação foi utilizado a biblioteca React.

Na Figura 3 é apresentado um subconjunto de dígitos do conjunto de treino do conjunto de dados MNIST.

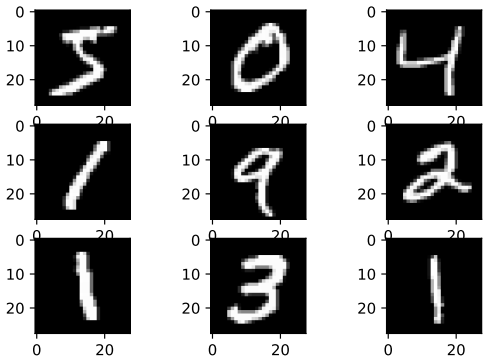

Fig. 3. Subconjunto de imagens do MNIST

VI. Resultados do modelo

Acurácia para os dados de teste: $98,84\%$

A Figura 4 apresenta a matriz de confusão para os dados de teste.

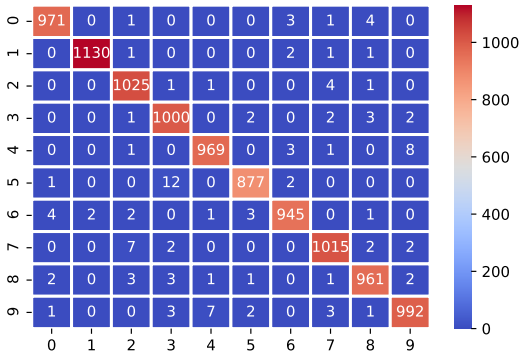

Fig. 4. Matriz de confusão

As Figuras 5 e 6 apresentam o gráfico de acurácia e função de perda do modelo em relação as épocas.

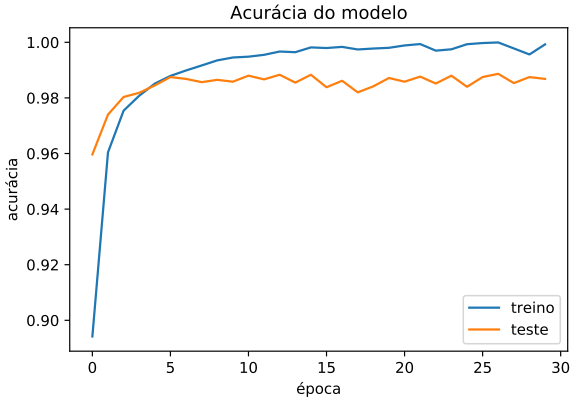

Fig. 5. Acurácia do modelo

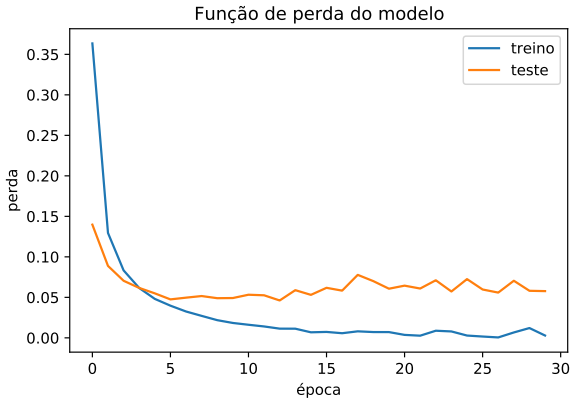

Fig. 6. Função de perda do modelo

VII. Aplicação

A aplicação foi desenvolvida utilizando a biblioteca React. A aplicação faz uma requisição a um servidor que disponibiliza os arquivos convertidos pelo TensorFlow.js.

A Figura 7 apresenta o elemento *canvas* para o usuário desenhar o dígito. A Figura 8 mostra o gráfico gerado pela aplicação com os valores preditos.

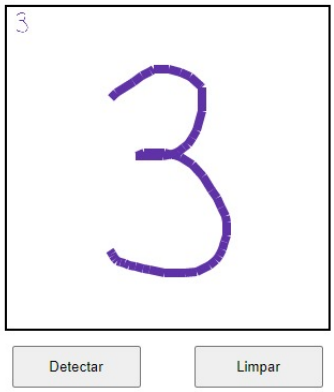

Fig. 7. Canvas da aplicação

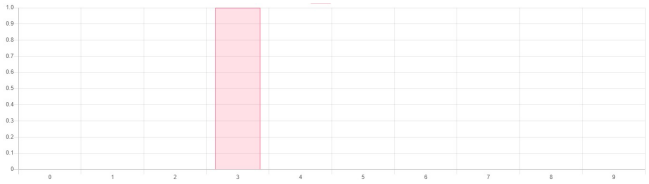

Fig. 8. Gráfico da predição do modelo

VIII. Conclusão

O objetivo da pesquisa atual é detectar a escrita através de um aplicativo web por meio de redes neurais convolucionais. Para isso, foi desenvolvido a arquitetura LeNet-5 e treinada com o conjunto de dados MNIST, o qual possuí imagens da escrita de dígitos. O modelo final da arquitetura LeNet-5 apresentou uma acurácia para os dados de teste de $98,84\%$ apesar de apresentar sobreajuste. Uma aplicação web foi desenvolvida com o fim de permitir o usuário "desenhar" o dígito em tela, para isso foi utilizado as bibliotecas TensorFlow.js para a parte de detecção da imagem gerada e o React para desenvolvimento da aplicação.

Referências

[1] Mor, Shubham Sanjay et al. HANDWRITTEN TEXT RECOGNITION: with Deep Learning and Android. 2019. Disponível em: https://www.ijeat.org/wp-content/uploads/papers/v8i3S/C11730283S19.pdf.

[2] LeCun, Y. (1989). Generalization and network design strategies. Connectionism in perspective, 19, 143-155.

[3] Yann Le Cun, L. Bottou and Y. Bengio, "Reading checks with multilayer graph transformer networks," 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing, Munich, 1997, pp. 151-154 vol.1, doi: 10.1109/ICASSP.1997.599580.

[4] GoodFellow, Ian; BENGIO, Yoshua; Courville, Aaron. Deep Learning. Cambridge: The Mit Press, 2016.

[5] Haykin, Simon. Neural Networks and Learning Machines. 3. ed. Hamilton, Ontario, Canada: Pearson, 2009.

[6] Wei, Guangfen; LI, Gang; ZHAO, Jie; HE, Aixiang. (2019). Development of a LeNet-5 Gas Identification CNN Structure for Electronic Noses. Sensors. 19. 217. 10.3390/s19010217.

[7] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

[8] Smilkov, Daniel, et al. "Tensorflow. js: Machine learning for the web and beyond." arXiv preprint arXiv:1901.05350 (2019).