Universidade Tecnológica Federal do Paraná – UTFPR
Departamento Acadêmico de Eletrônica – DAELN
Bacharelado em Engenharia Eletrônica
Disciplina: ELTD4 - Reconhecimento de Padrões e Aprendizado de Máquina
Semestre: 2024.1
Prof: Gustavo B. Borba

# RELATÓRIO
## Realtime Math Solver

Alunos:
André Vaz Igarashi / 2023660
Pablo Rodrigues Sene / 2090848

Set.2024

## 1 Objetivo

O objetivo deste estudo é desenvolver um solucionador de equações em tempo real, utilizando um modelo treinado para reconhecimento de números e operadores matemáticos.

## 2 Fundamentação Teórica

Este estudo foi dividido em duas principais etapas: o treinamento do modelo para reconhecimento de números e operadores matemáticos, e a implementação desse reconhecimento em tempo real para resolver equações matemáticas.

### 2.1 Geração de Equações

Em tarefas de aprendizado de máquina supervisionadas, é essencial dispor de dados "rotulados", ou seja, dados em que a entrada e a saída correspondentes são conhecidas. Com base nessa premissa, foi implementado um método para a criação de equações utilizando imagens de números e operadores matemáticos escritos à mão, uma vez que o objetivo final do projeto é reconhecer e resolver essas equações. As equações geradas serão utilizadas para avaliar o desempenho final do modelo.

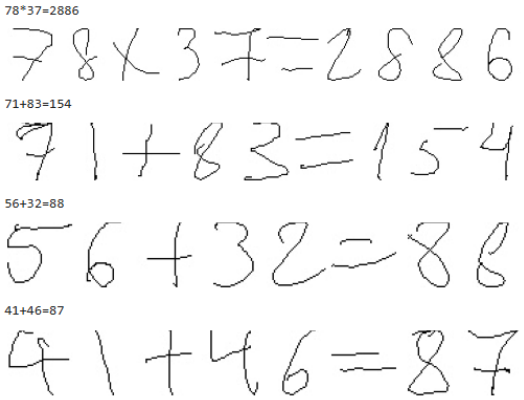


*Figura 1: Equações Geradas*

As equações geradas contemplam os números de 0 à 9, e os operadores de soma, multiplicação, e subtração. Elas podem ou não ter o sinal de igualdade.

### 2.2 Treinamento

Para a realização do treinamento do modelo, uma série de bibliotecas foi utilizada, dentre elas estão **tensorflow** e **open-cv**, que são amplamente utilizadas para o desenvolvimento de projetos do segmento de visão computacional.

Em todas as interações com os dados de imagens no decorrer do desenvolvimento do código para treinamento e avaliação do modelo, foram aplicadas uma série de processamentos de imagens, com a finalidade de tornar resultado do modelo ainda melhor. Técnicas como binarização, detecção de contornos, e *padding*, foram utilizadas em todo o processo.

Inicialmente, para reconhecimento dos caracteres, foi considerado o uso do *Tesseract OCR*, que é um software de reconhecimento de caracteres de código aberto, que já foi mantido pela Google. Porém, não foram obtidos bons resultados quando os caracteres eram escritos à mão, então outra abordagem foi utilizada para resolver o problema.

### 2.3 Redes Neurais Convolucionais (CNNs)

Redes Neurais Convolucionais (CNNs) são um tipo de rede neural profunda amplamente utilizada para reconhecimento de padrões em imagens. A principal vantagem das CNNs é sua habilidade de capturar padrões espaciais e hierárquicos em dados de imagem, por meio de camadas convolucionais que aplicam filtros para extrair características como bordas, texturas e formas [1]. Esses filtros são treinados de forma supervisionada, aprendendo a identificar características específicas que ajudam no reconhecimento e classificação de imagens [2].

### 2.4 Max Pooling

O Max Pooling é uma técnica utilizada nas CNNs para reduzir a dimensionalidade dos dados e diminuir a quantidade de parâmetros na rede, evitando o overfitting. Durante o processo, é aplicada uma janela deslizante que seleciona o valor máximo em regiões específicas da imagem, preservando características importantes como bordas e contornos [3]. Esta técnica não apenas reduz a complexidade computacional, mas também melhora a generalização do modelo [4].

### 2.5 Algoritmos de Otimização

Algoritmos de otimização, como o Gradiente Descendente Estocástico (SGD) e o Adam, são utilizados para ajustar os pesos da rede neural, minimizando a função de perda. O algoritmo Adam, por exemplo, combina as vantagens do SGD com a adaptação de taxas de aprendizado, proporcionando uma convergência mais rápida e eficiente [5]. A escolha do algoritmo de otimização pode ter um impacto significativo no desempenho do modelo e na velocidade de treinamento [6].

### 2.6 Funções de Ativação

As funções de ativação, como a ReLU (Rectified Linear Unit), são essenciais nas redes neurais, pois introduzem não-linearidade ao modelo, permitindo que a rede aprenda

e represente relações complexas nos dados [7]. A ReLU é particularmente popular por sua simplicidade e eficácia em evitar o problema do desaparecimento do gradiente, um problema comum em redes profundas [8].

### 2.7 Métricas de Avaliação do Modelo

A avaliação do desempenho do modelo é realizada utilizando métricas como acurácia, precisão, recall e F1-score. A acurácia mede a proporção de previsões corretas, enquanto precisão e recall oferecem uma visão mais detalhada sobre a performance do modelo em diferentes classes [9]. O F1-score é a média harmônica entre precisão e recall, proporcionando um equilíbrio entre esses dois aspectos, especialmente útil em problemas de classificação desbalanceada [10].

## 3 Processamento em tempo real

O processamento em tempo real de equações matemáticas é realizado através de uma sequência de etapas, cada uma desempenhando um papel crucial para garantir a precisão do reconhecimento e solução da equação. A seguir, detalhamos cada submódulo utilizado no processamento.

### 3.1 Carregamento do Modelo

O primeiro passo no processamento em tempo real é carregar um modelo de aprendizado profundo treinado para o reconhecimento de caracteres manuscritos. Utilizando o modelo desenvolvido com a biblioteca TensorFlow, que é capaz de classificar caracteres como números (0-9), operadores matemáticos (‘+‘, ‘-‘, ‘*‘, ‘=‘), entre outros.

### 3.2 Binarização da Imagem

A binarização da imagem é uma etapa fundamental no pré-processamento, transformando a imagem capturada pela câmera em um formato que destaca os caracteres manuscritos. Isso é feito convertendo a imagem para escala de cinza e aplicando um limiar adaptativo para criar uma imagem binária invertida, onde os caracteres são representados em branco sobre um fundo preto.

### 3.3 Detecção de Contornos

Após a binarização, o próximo passo é identificar contornos na imagem, que representam os diferentes caracteres matemáticos. Utilizamos o algoritmo de Canny para detectar bordas na imagem binarizada, seguido pela função ‘detect_contours‘ que refina esses contornos e elimina aqueles que não correspondem aos tamanhos mínimos esperados, reduzindo ruídos.

### 3.4 Redimensionamento e Preenchimento

Para padronizar a entrada ao modelo, cada contorno detectado é redimensionado e preenchido para um tamanho fixo, garantindo que o modelo receba uma entrada consistente. A função ‘resize_pad‘ é responsável por redimensionar a imagem de cada contorno e adicionar bordas de preenchimento conforme necessário para manter a proporção original.

### 3.5 Predição de Caracteres

Com as imagens pré-processadas, cada caractere é passado pelo modelo para prever qual classe ele representa. A função 'preprocess_webcam_image' gerencia esse fluxo, desde a captura da imagem até a montagem da equação completa a partir das previsões individuais de cada caractere.

### 3.6 Resolução da Equação

Depois que a equação é detectada e composta, utilizamos a biblioteca SymPy para resolver a equação matematicamente. A função 'solve_equation' interpreta a equação detectada, resolve-a simbolicamente, e retorna o resultado.

### 3.7 Captura de Vídeo e Exibição em Tempo Real

Por fim, o sistema captura o feed de vídeo em tempo real usando OpenCV, processa cada quadro para detecção e solução de equações, e exibe os resultados na tela. As equações detectadas são exibidas diretamente na janela de vídeo com seus resultados correspondentes, proporcionando uma interface interativa e em tempo real para o usuário.

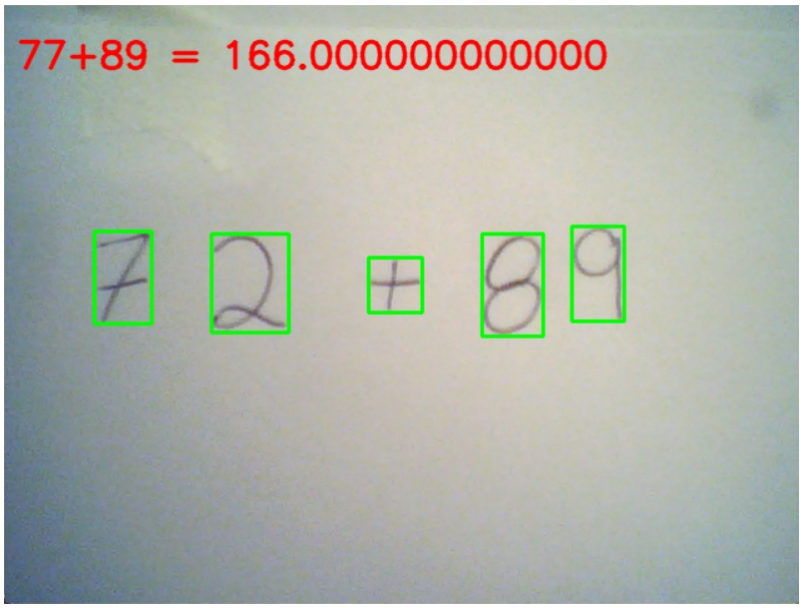


*Figura 2: Exemplo de detecção encontrados em uma câmera em tempo real.*

## Referências

[1] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.

[2] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

[3] Scherer, D., Müller, A., & Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. In *International Conference on Artificial Neural Networks* (pp. 92-101).

[4] Zeiler, M. D., & Fergus, R. (2013). Visualizing and understanding convolutional networks. *arXiv preprint arXiv:1311.2901*.

[5] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[6] Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

[7] Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (pp. 315-323).

[8] Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning* (pp. 807-814).

[9] Powers, D. M. (2011). Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. *Journal of Machine Learning Technologies*, 2(1), 37-63.

[10] Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4), 427-437.

[**?**] [**?**]