Resolução de CAPTCHA's utilizando Redes Neurais Convolucionais

Rafael Gonçalves , Thomás Portugal

Departamento de Engenharia de Computação e Automação Industrial (DCA)
Faculdade de Engenharia Elétrica e de Computação (FEEC)
Universidade Estadual de Campinas (Unicamp)
CEP 13083-852 – Campinas, SP, Brasil

{ra186062,ra187646}@.fee.unicamp.br

Abstract – This project studies the possibility of using k-means and convolutional neural networks for solving text-based reverse Turing test mechanism, called CAPTCHA. Such mechanism is often used in network security for preventing attacks such as password guessing using bruteforce or dictionary or denial of service attacks. Our approuch was based on creating a pipeline of pre-processing, segmentation and optical character recognition. For the first part of the project we used otsu threshold, erosion and dilatation methods for denoising, then used k-means clusterization algorithm for segmentation and finally trained a convolutional neural network to classify each individual character. The output was the concatenation of the output of the network for each input character of the CAPTCHA.

Keywords – CAPTCHA, k-means, otsu threashold, convolutional neural networks, network security, optical character recognition.

1. Introdução

CAPTCHA (*Completely Automated Public Turing test to tell Computers and Humans Apart*) é um mecanismo utilizado em serviços digitais para garantir que softwares automatizados não peçam acesso em excesso ou de maneira torpe, preservando a integridade e a confiabilidade de tais serviços. Normalmente um CAPTCHA é uma imagem gerada por um computador, com caracteres distorcidos. A intenção é que seja uma tarefa relativamente fácil para uma pessoa executar, mas razoavelmente complicada, do ponto de vista computacional.

Como o CAPTCHA vem sendo extensamente usado como a primeira linha de defesa contra ataques DDoS (*Distributed Denial of Services*) e afins, saber o quanto essa ferramenta é segura no contexto atual é de suma importância. Este trabalho visa desenvolver um sistema, baseado em redes neurais convolucionais e em um algoritmo de clusterização, para a identificação dos caracteres apresentados em imagens de CAPTCHA coletadas do banco de dados do site Kaggle [1].

2. Proposta

O problema principal foi dividido em dois subproblemas: um de segmentação de imagem utilizando uma algoritmo de aprendizado não supervisionado (clusterização) e outro de classificação de imagens. Estes foram baseados em [4].

Antes da segmentação da imagem, houve um trabalho de pré-processamento dos dados baseado em [2]. Foram aplicados filtros para diminuir o nível de ruído da imagem, facilitando o trabalho de segmentação.

A parte de clusterização envolveu determinar em que regiões da imagem estavam os caracteres. Em imagens geradas por CAPTCHA, frequentemente, há sobreposição de letras e números, além de ruídos e manchas que não representam caracter nenhum. Estas artimanhas são utilizadas justamente para dificultar a ação de algoritmos sobre as imagens. Com o uso de clusterização, os caracteres foram identificados e a imagem foi cortadas para o uso posterior na etapa de classificação.

Na etapa de classificação, os fragmentos selecionados na etapa anterior, são separados em treino e teste, e inseridos na entrada de uma rede neural. Essa rede neural foi treinada para identificar os dígitos individualmente. Após ser aplicada nas amostras de teste, os caracteres foram identificados e a previsão da imagem juntou as saídas individuais de cada amostra e identificou os caracteres gerados pelo CAPTCHA originalmente.

2.1. Pré Processamento

Antes de tudo foram aplicadas técnicas de processamento de imagem para remover possíveis distorções e ruídos e facilitar a segmentação não supervisionada da imagem.

Nossa abordagem consistiu em aplicar um filtro que melhor convertesse a imagem em preto e branco a fim de remover as influências do degradê em cinza no fundo da imagem e posterior aplicação de filtros de *erosion* e *dilatation* visando remover

linhas artificialmente adicionadas por cima dos caracteres (Figura 1).

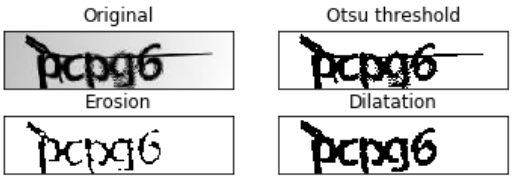

Figura 1. Etapas usadas para preprocessamento.

Foram testados 3 filtros diferentes (Figura 2) para a primeira etapa (*adaptative threshold*, filtro Otsu e filtro Otsu com ruído aleatório adicionado) e diferentes valores de iterações para os filtros de *erosion* e *dilatation*. Os filtros escolhidos, a saber filtro Otsu com 1 iteração de cada uma das distorções apresentadas, foram os que melhor removeram o ruído, selecionados por inspeção visual em uma amostra aleatória das imagens.

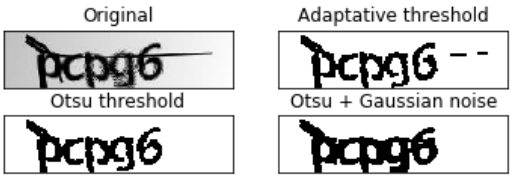

Figura 2. Resultado do preprocessamento para cada filtro com erosion e dilatation.

2.2. Segmentação da Imagem em Caracteres

Para separar os caracteres presentes na imagem processada, foi utilizado um algoritimo de clusterização e posteriormente expandiu-se um retângulo ao redor dos centroides dos clusters para determinar a região da imagem a ser salva como um caractere.

A abordagem escolhida para a etapa de clusterização foi o algoritmo k-means. O dataset utilizado, tinha a especificação de apenas de 5 caracteres por imagem. Assim, o número de 5 clusters foi determinado previamente. Por existir caracteres sobrepostos, a clusterização não identificava as fronteiras entre os caracteres de maneira tão eficiente. Além disso, o algoritmo, recorrentemente, identificava clusters na vertical, o que não é adequado já que os caracteres são dispostos na horizontal (Figura 3).

A solução encontrada foi projetar os dados

Figura 3. Clusters encontrados pelo kmeans nas 2 dimensões das imagens.

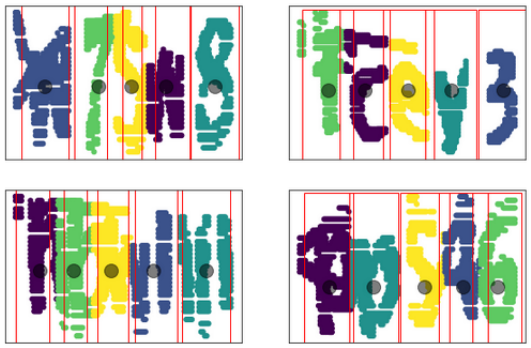

Figura 4. Clusters encontrados pelo kmeans no eixo horizontal. Retangulo ao redor de cada caractere encontrado pelo kmeans.

no plano horizontal e em seguida aplicar o k-means apenas em 1 dimensão. Com isso, a informação utilizada de fato foi a posição horizontal dos centroides. Para cortar as imagens os centros foram posicionados no meio do eixo vertical (Figura 4).

Com as posições do centro, foi necessário determinar os tamanhos para realizar o corte na imagem. Através de tamanhos pré-estabelecidos, foi determinado o tamanho de 21 pixels a direita e a esquerda do centro gerado pelo k-means. Essas imagens foram utilizadas na entrada da rede neural.

2.3. Reconhecimento de Caracteres

A etapa de reconhecimento de caracteres consistiu em uma rede neural convolucional inspirada em [4]. A rede possui a seguinte arquitetura:

1. Duas camadas convolucionais com kernel de lado 5, função de ativação linear retificada e max-polling de 2. A primeira com 6 kernels e a segunda com 16 kernels.
2. Duas camadas intermediárias *fully connected* com respectivamente 400 e 340 neurônios ReLU e dropout de 30%

3. A camada de saída com 19 neurônios e função de ativação softmax

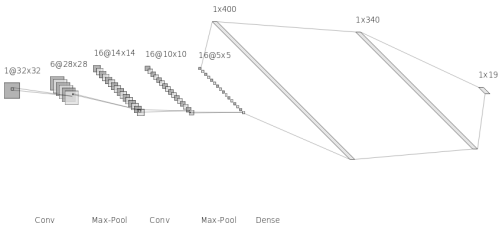

Figura 5. Arquitetura da rede neural.

O modelo foi escolhido usando validação cruzada (k-fold) e foi o que obteve a maior acurácia média entre os valores testados tanto para número de neurônios da última camada intermediária quanto para o valor de dropout (Tabela **??**).

H	Dropout	Avg Accuracy
50	0	0.8719
50	0.3	0.9197
50	0.5	0.9008
120	0	0.8996
120	0.3	0.9444
120	0.5	0.9499
340	0	0.9048
340	0.3	0.9541
340	0.5	0.9382

Tabela 1. Comparação entre modelos testados. H é o número de neurônios na última camada intermediária da rede neural.

3. Resultados

Como mostrado na Tabela 1, a acurácia do modelo de reconhecimento de caracteres foi de 0.9541 - a curva de aprendizado do modelo com o valor da função de custo pelo número de épocas está mostrado na Figura 6 e algumas predições na Figura 7.

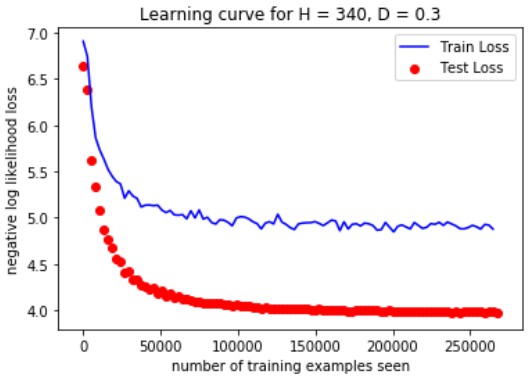

Figura 6. Curva de aprendizado da rede neural.

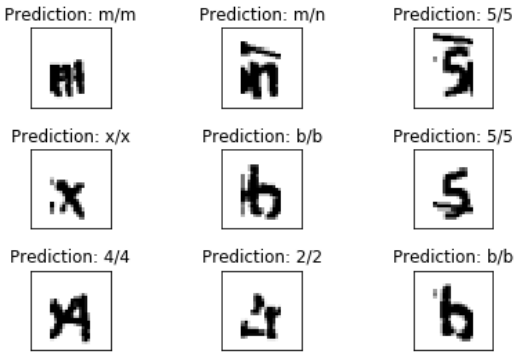

Figura 7. Classes encontradas pela rede neural para alguns caracteres individuais de entrada.

A acurácia do modelo completo (classificação do CAPTCHA todo) foi de 0.3246. Algumas das predições podem ser vistas na Tabela 2

Output	Target
g7w6y	g7wxw
6cn48	62nb4
6byng	6bxwg
gpbwn	dpbyd
wm47f	wm47f
ybncw	wbncw
532fm	5325m

Tabela 2. Comparação entre resultados previstos e rótulos para o modelo final.

A matriz de confusão (Figura 8) aponta que a maior parte das predições incorretas tem a ver com os caracteres 'm' e 'n'. Analisando-se os resultados foi possível perceber que 37% das predições incorretas (25% das predições totais) foram causadas por confusões entre esses dois caracteres.

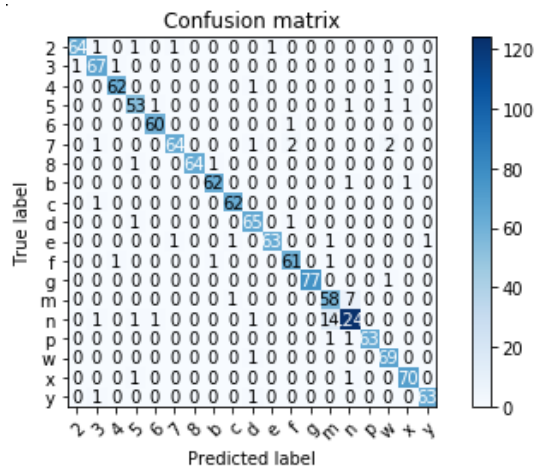

Figura 8. Matriz de confusão para a saída da rede neural.

4. Conclusões

Foi possível desenvolver um sistema que identifica imagens de CAPTCHA com uma acurácia total de 0.3246 para os dados contidos no *dataset* [1]. O valor é baixo comparado com os valores atingidos por outros modelos de classificação de textos, mas mostra uma vulnerabilidade bastante relevante presente no tipo de CAPTCHA analisado. Para trabalhos posteriores, podem ser testados outros modelos de clusterização para que este englobe CAPTCHA's com diferentes números de caracteres. Na parte de reconhecimento de caracteres, podem ser testadas redes neurais com diferentes arquiteturas, bem como outros modelos de classificação. É possível também que os métodos aqui desenvolvidos sejam usados em conjunto com bancos de dados com mais imagens para melhorar seu desempenho.

Referências

[1] Fournierp. Captcha version 2 images. https://www.kaggle.com/fournierp/captcha-version-2-images,. (acessado em 01/06/2019).

[2] Fournierp. Opencv word segmenting on captcha images. https://www.kaggle.com/fournierp/opencv-word-segmenting-on-captcha-images. (acessado em 01/06/2019).

[3] Adam Geitgey. How to break a captch system in 15 minutes with machine learning. https://medium.com/@ageitgey/how-to-break-a-captcha-system-in-15-minutes-with-machine-learning-dbebb035a710. (acessado em 01/06/2019).

[4] Martin Kopp, Matej Nikl, and Martin Holena. Breaking captchas with convolutional neural networks. *ITAT 2017 Proceedings*, pages 93–99, 2017.