

## Relatório de Entrega - Trabalho Prático 2

Gustavo Chaves Ferreira - 2022043329

Júlio Guerra Domingues - 2022431280

### Introdução

O seguinte relatório descreve a implementação de duas abordagens para o reconhecimento de caracteres presentes em CAPTCHAs, sistemas amplamente adotados para se diferenciar interações humanas de acessos automatizados em aplicações Web. A primeira delas combina descritores obtidos pelo método HOG (Histogram of Oriented Gradients) com o classificador SVC (Support Vector Classification). Já a segunda é elaborada em torno de uma CNN (Convolutional Neural Network), cuja arquitetura tem sido utilizada amplamente ao longo da última década para o reconhecimento bem sucedido de imagens. Serão discutidos os principais métodos implementados no código fonte, assim como os resultados de testes para o reconhecimento de CAPTCHAs compostos por 6 caracteres cada.

O código fonte da implementação de ambos os classificadores pode ser encontrado no repositório GitHub: [https://github.com/juliogdomingues/icv\\_ufmq.git](https://github.com/juliogdomingues/icv_ufmq.git). Os modelos treinados serão disponibilizados em: <https://drive.google.com/drive/folders/1k3eFOhoYBFb0C2QaJ9kBSz9RAX-KGDrQ>.

### Pré-processamento

Após uma análise inicial dos dados disponibilizados, foram encontrados alguns labels de tamanho diferente de 6 ou contendo o símbolo '?'. Como todos os CAPTCHAs são de comprimento 6 e possuem somente caracteres alfanuméricos, tais instâncias foram filtradas e não aparecem no desenvolvimento dos modelos. Ademais, todas as imagens foram convertidas para possuírem um único canal (ou seja, todos os pixels são descritos apenas por um tom de cinza).

### Implementação do classificador baseado em HOG + SVC

Dentro da pasta 'hog+svc', no arquivo intitulado 'hog\_svc\_main.py', pode-se encontrar as funções responsáveis pelo extrator HOG. Aqui, cada uma das imagens fornecidas contendo um CAPTCHA de 6 caracteres é dividida em 6 partes arbitrárias no eixo horizontal pelo método 'split\_images', que as armazena em diretórios separados dos originais. Essas imagens serão usadas para treino, validação e teste, visto que representam os caracteres de forma (quase) individual. Pode-se empregar tal divisão porque a disposição dos caracteres em toda a base de dados é praticamente uniforme. Ademais, 'read\_and\_strip\_chars\_from\_files' lê cada uma das labels ground-truth e as divide em 6 caracteres, concatenando-os em um vetor final (para que as labels estejam coerentes com as novas imagens cortadas).

A função 'compute\_gradients\_magnitude\_and\_orientation' itera por todos os pixel de uma imagem recebida como parâmetro, calculando os gradientes nas direções x e y de cada um deles, e por fim obtendo a magnitude e o ângulo de cada um deles através das fórmulas vistas no slides de aula. O método 'compute\_gradients\_histograms' recebe duas matrizes contendo as magnitudes e os ângulos calculados anteriormente, e fica responsável por gerar todos os 7\*15\*4 histogramas de 9 bins de uma imagem 64x128 (representada pelas duas matrizes citadas). Somamos nos bins correspondentes aos ângulos do gradiente de cada pixel valores proporcionais às magnitudes correspondentes. Já a função 'compute\_image\_feature\_vector' é a encarregada de gerar o vetor de características final da imagem. Para cada um dos 7\*15 blocos de 2x2 células (cada célula contendo dimensões de 8x8 pixels), ela concatena os 4 histogramas desse bloco na resposta final. Como deslizamos os blocos com sobreposição de 2 células 2x2 tanto na horizontal quanto na vertical, ao fim temos um vetor de dimensão 7\*15\*4\*9 = 3780. A chamada 'compute\_hog\_features' apenas agrega em sequência as três funções anteriores para gerar o feature vector de cada uma das imagens.

Gerados os vetores de características para cada uma das imagens 6\*10000 imagens fornecidas, utilizamos o conjunto imagens + labels de treino para criar um classificador do tipo Support Vector Classification, utilizando os dados de validação para se determinar o melhor conjunto de hiperparâmetros 'C' e 'kernel' para o modelo. Por fim, salvamos o melhor modelo encontrado em um arquivo '.pkl' para reuso posterior (especialmente no arquivo que gera estatísticas 'hog\_svc\_main.py').

### Implementação do classificador baseado em CNN

A primeira abordagem com uma CNN foi projetada para prever diretamente o texto completo de cada CAPTCHA a partir da imagem bruta. O modelo recebe como entrada imagens em escala de cinza de tamanho 128x64 pixels e as processa por meio de uma série de camadas convolucionais

compartilhadas. Essas camadas extraem características da imagem usando filtros 3x3, ativação ReLU e max-pooling para redução progressiva de dimensionalidade, culminando em uma camada *flatten*. Cada caractere do CAPTCHA é então predito por um dos seis branches independentes conectados à camada compartilhada. Cada branch consiste em uma camada totalmente conectada (Dense) com 128 neurônios e ativação ReLU, seguida de uma camada de saída com 36 neurônios e ativação softmax, correspondente às classes alfanuméricas (A-Z, 0-9). Essa abordagem, embora conceitualmente elegante, enfrenta o desafio de lidar com um espaço de saída altamente complexo, resultando em uma taxa de acurácia mais limitada para predições completas.

A segunda abordagem foi desenvolvida para superar as limitações do modelo inicial, propondo uma arquitetura de CNN focada na predição de caracteres individualmente. Nesse caso, cada CAPTCHA é segmentado em 6 partes, correspondendo a caracteres isolados, e o modelo recebe imagens de tamanho 32x32 pixels como entrada. A rede é composta por três camadas convolucionais sequenciais com 32, 64 e 128 filtros, respectivamente, todas utilizando filtros 3x3, ativação ReLU e max-pooling para extração e compressão das características. A saída da última camada convolucional é achatada e conectada a uma camada totalmente conectada de 128 neurônios com ativação ReLU, seguida de uma camada de saída com 36 neurônios e ativação softmax para a classificação do caractere. Essa abordagem simplifica o problema, reduzindo significativamente o espaço de saída e possibilitando uma maior acurácia, tanto para caracteres individuais quanto para CAPTCHAs completos.

## Resultados e discussão

No código disponibilizado, é possível gerar, para ambos os modelos, outputs contendo os dados de teste na seguinte forma: a imagem do CAPTCHA de teste original seguido do texto correto em comparação com o texto predito pela abordagem em questão (HOG ou CNN), como visto a seguir:

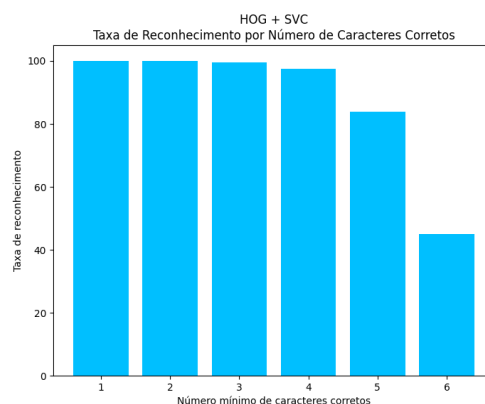


Previsto: JW4CZF | Verdadeiro: JW4CZF

Previsto: H27LCM | Verdadeiro: H27LCM

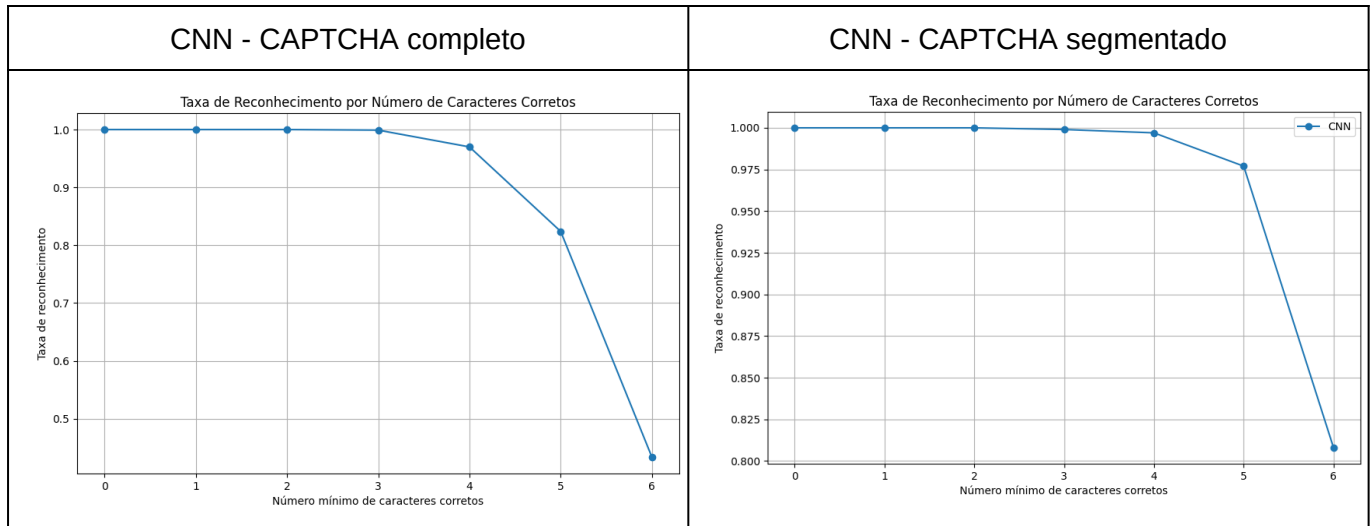
### HOG + SVC:

Entre as vantagens do uso dessa abordagem, pode-se destacar o fato de que o vetor de características para uma dada imagem é sempre fixo, ao contrário das CNN que estão atreladas à aleatoriedade na etapa de treinamento e otimização. Ademais, tais vetores são mais explicáveis e interpretáveis do que os resultados de redes neurais, visto que capturam padrões nas orientações das bordas da imagem de entrada. Porém, o HOG não possui tanto potencial por si só de extrair padrões muito complexos em datasets, e fica limitado à utilização em aplicações onde existe baixa variância intraclasse ou onde existe uma fase de pré-processamentos dos dados para padronização dos mesmos. Os resultados de desempenho no conjunto de testes disponibilizado é o abaixo. Nota-se que em quase todos os casos, o classificador acerta ao menos quatro dos caracteres. A performance apresenta uma queda para o desempenho em cinco e principalmente seis caracteres.



Rede neural convolucional:

Com o emprego das CNN's, é possível obter vantagens como a melhor detecção de padrões complexos (resultando em uma melhor acurácia geral). Entretanto, o modelo resultante é menos interpretável e a chance de se obter overfitting cresce em comparação com o método anterior. Os resultados da avaliação das duas redes descritas anteriormente podem ser visualizados nos gráficos abaixo. Percebe-se a superioridade de tal abordagem na maioria dos casos, com o desempenho sendo próximo dos 80% de acerto para os 6 caracteres em um dos modelos. A rede treinada apenas com as imagens brutas e sem nenhum tratamento apresenta um desempenho um pouco inferior, ainda que apresente uma estrutura interna mais complexa (o que é esperado, visto que o número de combinações possíveis de 6 caracteres é muito maior do que de dígitos individuais, logo, o aprendizado é dificultado consideravelmente).

**Conclusão**

Os resultados obtidos demonstram as vantagens e limitações de cada abordagem. O método baseado em HOG + SVC se destacou pela explicabilidade e estabilidade, uma vez que utiliza vetores de características fixos e interpretáveis. Contudo, essa abordagem é limitada pela simplicidade dos padrões que o HOG consegue capturar, resultando em uma performance inferior para problemas mais complexos. Por outro lado, as redes neurais convolucionais mostraram-se mais eficazes no reconhecimento de CAPTCHAs, aproveitando-se de sua capacidade de aprender padrões complexos diretamente das imagens. A abordagem segmentada, que utiliza uma CNN para prever cada caractere individualmente, apresentou o melhor desempenho geral, atingindo uma acurácia superior tanto para caracteres isolados quanto para CAPTCHAs completos.

Um aspecto essencial para o sucesso das abordagens foi o uso do conhecimento de domínio. No caso dos CAPTCHAs, saber que as letras são dispostas uniformemente e que a segmentação prévia era possível permitiu simplificar o problema e obter ganhos significativos de desempenho. Essa simplificação viabilizou um aprendizado mais eficiente pelas redes neurais, reduzindo a complexidade do espaço de saída e explorando características específicas de cada caractere. Assim, concluímos que a integração entre técnicas avançadas de aprendizado de máquina e uma análise criteriosa do problema é fundamental para alcançar soluções mais robustas e eficientes em tarefas de reconhecimento visual.