Relatório Análise de Imagens - Detecção de Placas de Carro

Felipe Pereira - 263808

11 de janeiro de 2020

Resumo

Este relatório refere-se aos trabalhos da disciplina Análise de Imagens, ministrada em 2020 pelo professor Alexandre Falcão, apresentando os resultados obtidos ao implementar um pipeline de análise de imagens de carros com o objetivo de identificar as placas destes.

Objetivo

Esta primeira tarefa consiste no início do pipeline. Nela iremos trabalhar o dataset de imagens de carros. Este dataset terá cada imagem dividida em patches, aumentando assim a quantidade de imagens disponíveis.

Cada patch deverá ter W × H pixel e uma movimentação de (D x , D y) pixels. Cada patch, que originalmente é uma imagem em grayscale, deverá ser transformado e pseudo-colorizado em uma imagem RGB (Red-Green-Blue) e depois em Y-Cb-Cr (ou LAB).

Após estas transformações, deverá ser aplicado um processo de Batch Normalization que transformará cada image, Î into an image Ĵ i = (D I , J i):

$$J_{ij}(p) = \frac{I_{ij}(p) - \mu_j(p)}{\sigma_j(p)}, \quad (1)$$

$$\mu_j(p) = \frac{1}{N}\sum_{i=1}^{N} I_{ij}(p), \quad (2)$$

$$\sigma_j(p) = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N} (I_{ij}(p) - \mu_j(p))^2}, \quad (3)$$

Depois deste processo, deverá ser feito o processo de criação dos Random Kernels. Deverão ser criados b random kernels com tamanho w x h, w << W e h << H. Também deverão ser analisadas as diferenças de forçar e não forçar estes kernels a terem média igual a zero.

Estes kernels serão utilizados para a realização de convoluções nos patches. Além dos kernels criados de maneira aleatória, será testado um kernel Sobel para comparação. Por fim, executaremos uma função de ativação ReLu para chegar ao resultado final.

Dataset Composition

A primeira etapa deste estudo consistiu em dividir as imagens em patches. Estes patches foram divididos de acordo com o stride definido. Foram testados strides de 1,5,10,15 e 20 pixels. O stride de 15 pixels foi escolhido por ser o que melhor consegue separar adequadamente em um único patch as placas dos carros:

Imagem orig_0200.png
Patch 152

Imagem orig_0200.png
Patch 153

Cada patch em GRAYSCALE, foi transformado em BGR para enfim ser convertida em Y-Cb-CR:

Imagem orig_0200.png
Patch 152 em Y-Cb-Cr

Imagem orig_0200.png
Patch 153 em Y-Cb-Cr

Batch Normalization

Tirando a média dos valores, temos uma imagem parcialmente modificada:

Imagem orig_0200.png
Patch 152
Com média em zero

Cada imagem acima representa um canal da imagem. Apenas o primeiro canal possui algo que pode ser detectado através do olho humano e, mesmo assim, com auxílio computacional de Gamma adicional. Os outros dois canais normalizados não apresentam elementos visualmente perceptíveis a olho nu.

Convoluções

Foram criados 3 kernels aleatórios e 1 kernel sobel. Primeiro serão analisados os resultados dos kernels sem forçar com que a média dos mesmos fosse zero.

Resultado da convolução com os kernels sem a média igual a zero:

Imagem orig_0200.png
Patch 153
Convolução com filtro aleatório

Este kernel não ressaltou nenhum aspecto importante da imagem, causando, inclusive, perda de informação importante (o número da placa).

Resultado da convolução com os kernels com a média igual a zero:

Imagem orig_0200.png
Patch 152
Convolução com filtro aleatório e média zero

Este kernel ressaltou os relevos presentes na imagem.

Também é interessante comparar os resultados acima com os resultados da utilização de um filtro Sobel:

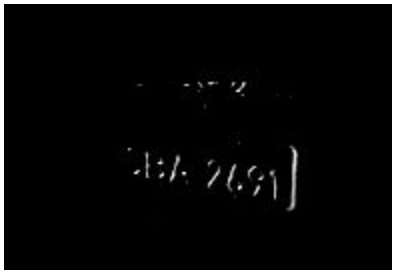

Imagem orig_0200.png
Patch 153
Convolução com filtro Sobel

Já o filtro sobel conseguiu extrair os relevos verticais, mostrando o relevo da placa e dos números e letras da placa.

ReLU

Nesta fase do pipeline aplicamos uma função de ReLu no resultado da convolução:

$$A_{ij}(p) = \max\{0, C_{ij}(p)\}.$$

Vamos aplicar o ReLu ao resultado da convolução do filtro aleatório com média zero:

Imagem orig_0200.png
Patch 152
Relu com filtro filtro aleatório e média em zero

Vamos aplicar o ReLu ao resultado da convolução do filtro Sobel:

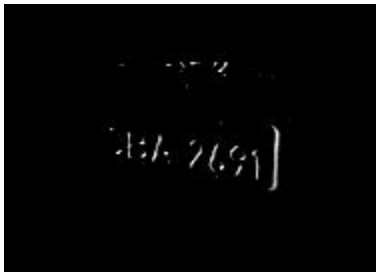

Imagem orig_0200.png
Patch 153
Convolução com filtro Sobel

Pooling

As operações de Pooling são responsáveis pelo processamento de imagem que, antes do advento das redes neurais, eram normalmente conhecidas pelos nomes de dilatação seguida de sub-amostragem (no caso do Max Pooling).

A operação de Max Pooling é responsável por verificar o valor de pixel mais alto em uma dada relação de adjacência, normalmente processada através de um kernel deslizante sobre a imagem. A definição teórica de Max Pooling é dada por:

$$P_i(p) = \max_{q \in \mathcal{A}(p)} \{R_i(q)\},$$

Fórmula de Max Pooling

Ao aplicar a operação de Max Pooling após a operação de ReLu, temos o seguinte resultado:

Imagem orig_0200.png
Patch 152
Max Pooling depois de ReLu

Outra possibilidade, é executar uma função de Average Pooling, definida por:

$$P_i(p) = \frac{1}{|\mathcal{A}(p)|} \sum_{q \in \mathcal{A}(p)} \{R_i(q)\},$$

Fórmula de Average Pooling

A operação de Average Pooling é relativamente parecida com a de Max Pooling em sua forma de atuação, tendo como diferença principal, a execução da média dos pixels da relação de adjacência. Temos como resultado desta operação:

Imagem orig_0200.png
Patch 152
Average Pooling depois de ReLu

Super Pixels

Os chamados “Super Pixels” são imagens definidas em áreas e, dentro destas áreas, temos que apenas um pixel se sobressai aos demais, daí o nome de “Super Pixel”. Esta técnica simplifica a imagem e tenta extrair semânticas mais simples (com uma diversidade menor de valores de pixels) em comparação com a imagem original.

Nos testes realizados, obtivemos uma melhor divisão da imagem quando aplicamos 30 Super Pixels:

Imagem orig_0200.png
Imagem dividida em 30 Super Pixels

haar-like

Existem diversas formas de extrair características das imagens e a haar-like é mais uma delas. Ela é muito utilizada para a extração de características importantes em imagens de rostos, dado que sua natureza permite a extração de características de identificação importantes como: sobrancelhas, nariz e boca.

Para cada patch da figura, foram extraídas as haar features que serão, posteriormente, passadas para um classificador (no caso, SVC), para que tenhamos um pipeline completo de detecção de placas.

Conclusão

Existem diversos tipos de parâmetros que podem ser ajustados no pipeline. Indo desde a escolha do tipo de imagem a usar, tamanho dos patches, stride, formas de realizar a normalização, os valores a serem usados nos filtros, dentre outros.

Foi constatado que filtros com a média centrada em zero possuem um comportamento melhor do que filtros com valores completamente aleatórios. Além disto, foi constatado também que filtros já conhecidos e testados, como o Sobel, possuem efeitos conhecidos e com ótimos resultados para os resultados.

Nosso pipeline está sendo incrementado com diversos tipos de técnicas para extração de características. Na próxima entrega, iremos testar todas estas características em um modelo de classificação, afim de obtermos o melhor desempenho possível.