Relatório Análise de Imagens - Detecção de Placas de Carro

Felipe Pereira - 263808

10 de novembro 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 conseguia separar adequadamente em um único patch as placas dos carros:

Imagem 001
Patch 1

Imagem 002
Patch 2

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

Imagem 003
Patch 1 em Y-Cb-Cr

Imagem 004
Patch 2 em Y-Cb-Cr

Batch Normalization

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

Imagem 005
Patch 1 com média em zero

Já ao realizar a normalização completa da imagem, obtemos (é preciso aumentar o Gamma do monitor para poder enxergar):

Imagem 006
Patch 1 normalizado (canal 1)

Imagem 007
Patch 1 normalizado (canal 2)

Imagem 008
Patch 1 normalizado (canal 3)

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 009
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 010
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:

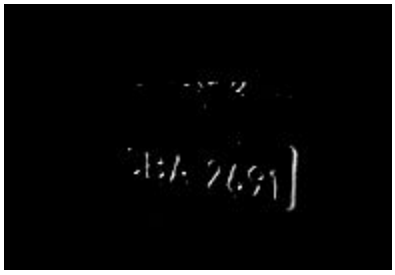

Imagem 011
Convolução com filtro Sobel

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 012
Relu com filtro filtro aleatório e média em zero

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

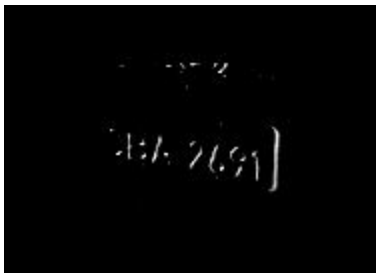

Imagem 013
Convolução com filtro Sobel

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.