Detecção *Seam Carving* Utilizando *Deep Learning*

Orientando: Gabriel Vieira Ferreira
Orientador: Prof. Dr. Kelton Augusto Pontara da Costa

Sumário

Introdução

- O desempenho e fácil acesso de dispositivos de imagem digital (câmeras digitais, filmadoras digitais e *smartphones*) melhorou de maneira significativa (CHOI; LEE; LEE, 2013);
- Consequentemente, imagens digitais podem ser facilmente manipuladas por usuários não profissionais, existindo um alto risco de violação de direitos autorais e aquisição ilegal de imagens (CHOI; LEE; LEE, 2013);
- O método denominado *Seam Carving* proposto por Avidan e Shamir (2007) baseia-se no acúmulo de energia para redimensionamento de imagens, criando uma descontinuidade do conteúdo da imagem e uma remoção ou adição de costuras (*seams*) (LIN et al., 2016);
- Enquanto *Local Binary Pattern (LBP)* proposto por He e Wang (1990) é um operador de textura simples e muito eficiente, rotulando os *pixels* de uma imagem limitando a vizinhança de cada *pixel* e considerando o resultado como um número binário (PIETIKINEN et al., 2011);
- O objetivo de uma *Convolutional Neural Network (CNN)* é aprender características de ordem superior nos dados por meio de convoluções. Essa rede é adequada para reconhecimento de objetos e classificação de imagens (PATTERSON; GIBSON, 2017);

Introdução

- O objetivo principal deste trabalho foi contribuir para solução de problemas de violação de direitos autorais e aquisição ilegal de imagens, utilizando uma *CNN* sobre auxilio do *LBP* para detectar adulterações em imagens causadas pelo *Seam Carving*;
- Ou seja, utilizar o método *Seam Carving* para fraudar imagens e posteriormente utilizar uma técnica inteligente a fim de detectar se a imagem foi fraudada ou não, e qual o tipo de fraude ocorreu, se *Seam Carving* (remoção de *seams*) ou *Seam Insertion* (inserção de *seams*) ou a imagem não foi adulterada;
- Há muitos trabalhos semelhantes na comunidade, em um dos trabalho os autores propuseram detectar adulterações de imagem também usando *CNN* e *LBP*, com foco em *Seam Carving* e *Seam Insertion* (CIESLAK; COSTA; PAPA, 2018). Outro trabalho propôs em um classificador multiclasse semelhante que possui quatro classes de saídas: 0% ou sem entalhe na costura, 10% entalhado na costura, 20% entalhado na costura e 40% entalhado na costura (NAZARİ; AKGÜN, 2020). Os resultados obtidos apontaram para aproximadamente 98% e 84% de acurácia em casos específicos, respectivamente;
- Os resultados obtidos no presente trabalho foram ótimos, se comparado a trabalhos relacionados, a rede alcançou 99% de acurácia.

Fundamentação Teórica - *Seam Carving*

- *Seam Carving* é um operador de imagem simples que oferece suporte ao redimensionamento de imagem com reconhecimento de conteúdo para redução e expansão (AVIDAN; SHAMIR, 2007);
- A esquerda está a imagem original, no centro está a imagem com os *seams* que foram aplicados resultando na expansão da imagem a direita (AVIDAN; SHAMIR, 2007).

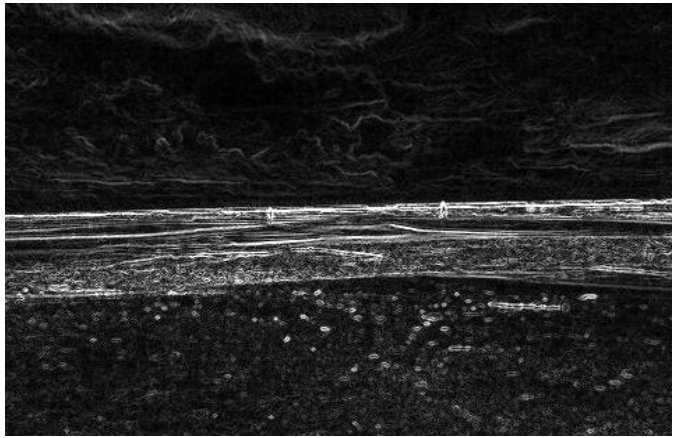
Fonte: Avidan e Shamir (2007)

Fundamentação Teórica - Algoritmo *Seam Carving*

- Neste tópico está descrito apenas a redução de *seams* do tamanho da imagem, para inserção o processo é semelhante;
- O objetivo central é remover *pixels* imperceptíveis (que contém menos informações);
- Necessário conhecer o conceito de Função de Energia, representado por e, onde I é a matriz de intensidade da imagem (AVIDAN; SHAMIR, 2007);

$$e(I) = \left|\frac{\partial I}{\partial x}\right| + \left|\frac{\partial I}{\partial y}\right|$$

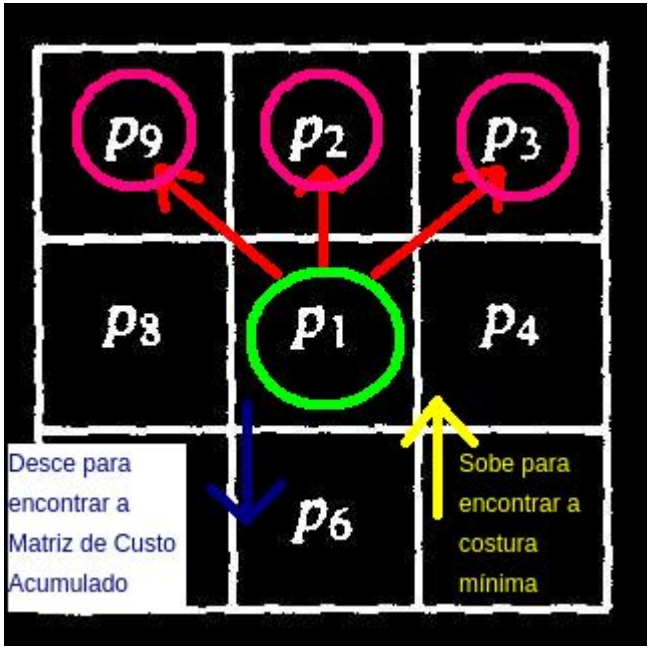
- Para cada canal de cor, a energia é calculada adicionando o valor absoluto do gradiente na direção x ao valor absoluto do gradiente na direção y;
- A energia para todos os canais de cores é somada em uma imagem 2D criando Mapa de Energia, visto na imagem inferior;



Fonte: Lykov (2013)

Fundamentação Teórica - Algoritmo *Seam Carving*

- Uma matriz de custo acumulado deve ser construída a partir da borda superior iterando através das linhas do mapa de energia;
- O valor de um *pixel* na matriz de custo acumulado é igual ao seu valor de *pixel* correspondente no mapa de energia adicionado ao mínimo de seus três vizinhos superiores (superior esquerdo, superior central e superior direito);
- Se um *pixel* vizinho não estiver disponível devido à borda esquerda ou direita, não é usado no cálculo;
- A costura mínima é então calculada retrocedendo da borda inferior para a superior, implementando com programação dinâmica;
- Costuras internas ou *seams* está definida pela letra s, onde i corresponde a coordenada da linha e $x(i)$ corresponde coordenada da coluna em i. A matriz I é uma imagem $n \times m$, onde n é o número de linhas e m o número de colunas. A variável x representa um mapeamento x: [1, ..., n]→[1, ..., m]. (AVIDAN; SHAMIR, 2007).



Fonte: O Autor

$$s^x = \sum_{i=1}^{n} s_i^x = \{x(i), i\}_{i=1}^{n}, \quad \forall i, |x(i) - x(i-1)| \leq 1$$

Fundamentação Teórica - Algoritmo *Seam Carving*

- As coordenadas mínimas de costura do são então usadas para remover a costura mínima.
- Todos os *pixels* em cada linha após o *pixel* a ser removido são deslocados em uma coluna.
- O mesmo código pode ser facilmente usado para reduzir a altura da imagem simplesmente tomando a transposição da imagem de entrada;
- A esquerda a imagem original de tamanho *332 x 480*, ao centro, exemplos de *seams* aplicados, a direita a imagem de tamanho *272 x 400*, após a aplicação da remoção de costuras.

Fonte: Lykov (2013)

Fundamentação Teórica - *Local Binary Pattern (LBP)*

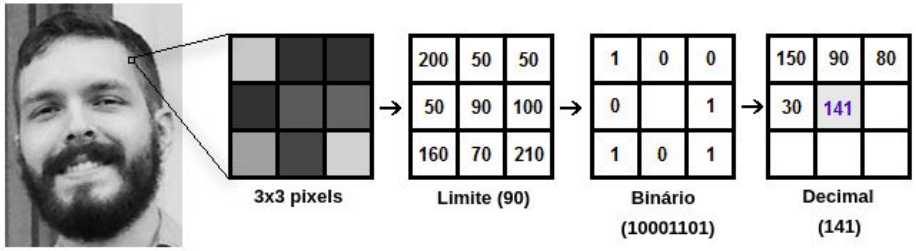
- O operador *Local Binary Patterns (LBP)* rotula os *pixels* de uma imagem com números decimais, chamados de padrões binários locais ou códigos *LBP*, que codificam a estrutura local em torno de cada *pixel* (HUANG et al., 2011);
- O processo *LBP* também pode ser descrito através da equação apresentada abaixo, onde g_c é o valor do *pixel* central, g_p é o valor de seus vizinhos, P é o número total de vizinhos envolvidos e R é o raio da vizinhança (HUANG et al., 2011);
- Geralmente R é igual a 1 e P igual 8, o que significa que os *pixels* vizinhos serão conectados em oito (vizinhos nas horizontais, verticais e diagonais). Esses valores são escolhidos como padrão devido a compensação sobre esforço computacional (HUANG et al., 2011).

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)2^P, \quad s(x) = \begin{cases} 1, x \geq 0 \\ 0, x \leq 0 \end{cases}$$

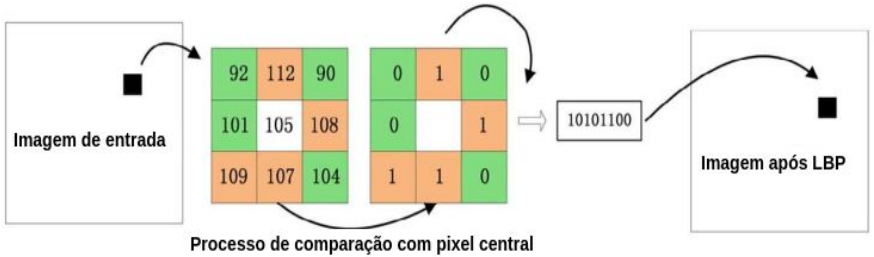
Fonte: Prado (2017)

Fundamentação Teórica - Algoritmo *LBP*

- Suponha uma parte de uma imagem de dimensão *3x3 pixels* em tons de cinza. Essa mesma parte pode ser representada como uma matriz *3x3* contendo a intensidade de cada *pixel* (*0 ~ 255*);
- O *LBP* obtém o valor central da matriz de intensidade de cada *pixel* para ser usado como limite, então define os novos valores dos 8 vizinhos;
- Para cada vizinho do limite, o *LBP* define um novo valor binário, 1 para valores iguais ou superiores ao limite e 0 para valores inferiores ao limite;
- ignorando o valor central, a matriz conterá apenas valores binários. Esses valores são concatenados (10001101);
- Em seguida, esse valor binário é convertido em um valor decimal, tornando-se o código *LBP* naquele local.



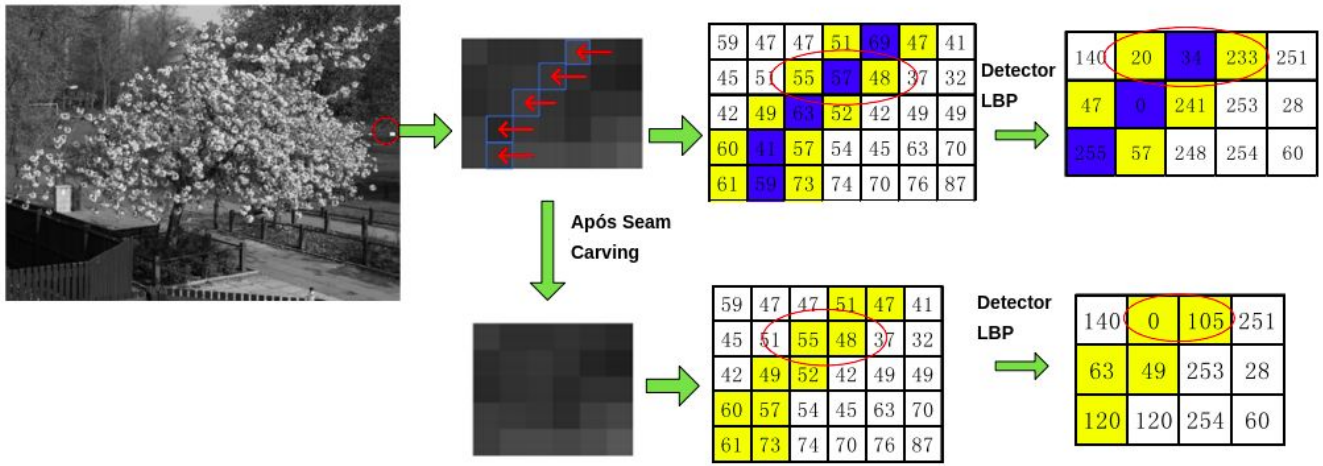
Fonte: Prado (2017)



Fonte: Yin et al. (2015)

Fundamentação Teórica - *LBP* e *Seam Carving*

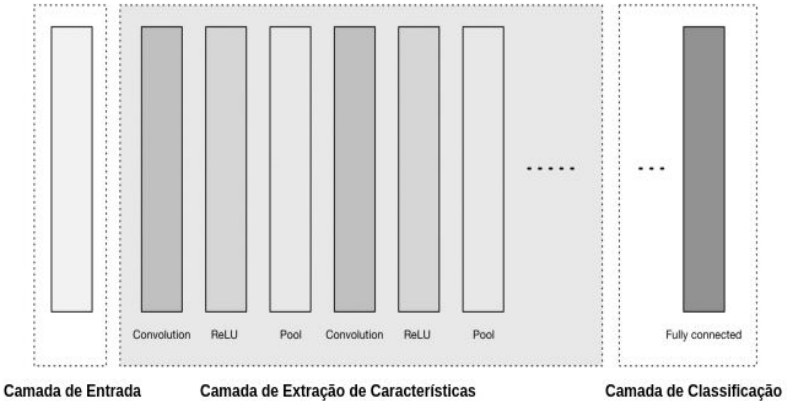
- Quando uma costura vertical é removida, todos os *pixel*s em seu lado direito são deslocados para a esquerda para preencher a lacuna do caminho removido (YIN et al., 2015);
- Nos *pixels* exatamente adjacentes a essa junção, os códigos *LBP* são significativamente alterados. Portanto, seus códigos *LBP* serão diferentes dos originais (YIN et al., 2015);
- Na imagem abaixo, em azul os *pixels* a serem removidos (*seams*), em amarelo os *pixels* adjacentes (YIN et al., 2015).



Fonte: Yin et al. (2015)

Fundamentação Teórica - *Convolutional Neural Network (CNN)*

- *Convolutional Neural Network (CNN)* é um dos algoritmos de aprendizado profundo mais comumente usados, são amplamente usados para tarefas como reconhecimento de imagens e detecção de objetos (rostos, placas de rua, animais, entre outros), (RAVICHANDIRAN, 2019).



Fonte: Patterson e Gibson (2017)

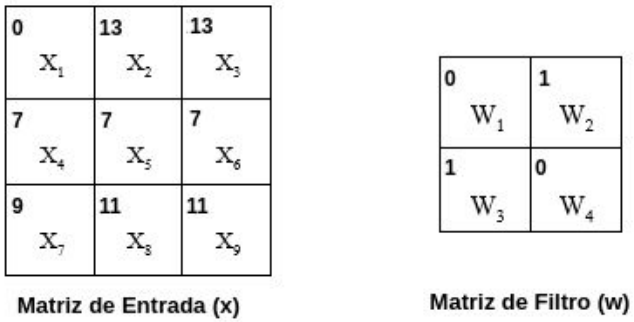
Fundamentação Teórica - Funcionamento

- Quando alimentamos a imagem para um computador, ele basicamente a converte em uma matriz de valores de *pixel* (valores entre 0 e 255);
- As dimensões desta matriz serão de [largura da imagem x altura da imagem x número de canais] (RAVICHANDIRAN, 2019);
- Uma imagem em tons de cinza tem um canal e as imagens coloridas têm três canais: vermelho, verde e azul (*red*, *green and blue* - *RGB*) (RAVICHANDIRAN, 2019).

Fonte: Ravichandiran (2019)

Fundamentação Teórica - Camada Convolucional

- A Camada Convolucional é a primeira camada da *CNN*;
- As características que nos ajudarão a entender que esta é a imagem de um cavalo (estrutura corporal, rosto, pernas, cauda) são encontradas por meio de convoluções;
- Cada imagem de entrada é representada por uma matriz de valores de *pixel* onde x_i refere-se ao *pixel* da posição i;
- Além da matriz de entrada, no processo de convolução é utilizada uma matriz de filtro ou *kernel*, onde w_i refere-se ao *pixel* da posição i.



Fonte: Ravichandiran (2019)

Fundamentação Teórica - Camada Convolucional

- A matriz de filtro é deslizada sobre a matriz de entrada por um *pixel;*
- Executa-se a multiplicação por elemento, soma-se os resultados e é produzido um único número, vários filtros podem ser usados;
- O resultado dessa operação é uma matriz denominada mapa de características;
- A operação de convolução pode ser definida pela fórmula abaixo, onde o_{ij} representa a saída que formará a matriz convolvida na linha i e coluna j. A variável X representa a matriz de entrada e W o filtro de tamanho $P \times Q$;

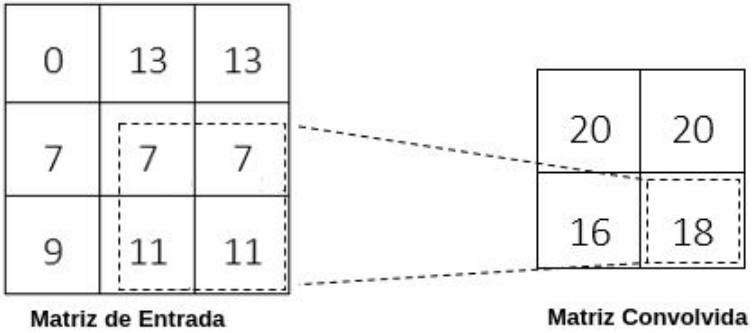
$$o_{ij} = \sum_{m=0}^{p-1} \sum_{n=0}^{q-1} W_{m,n} \cdot X_{i+m,j+n}$$

0·0 + 13·1 + 7·1 + 7·0 =20;
13·0 + 13·1 +7·1 + 7·0 = 20;
7·0 + 7·1 + 9·1 +11·0 = 16;
7·0 + 7·1 + 11·1 + 11·0 = 18

$$\hat{y}_i = f(o_{ij})$$

- Assim que a operação de convolução é realizada, alimentamos o resultado o_{ij}, para uma rede sem realimentação (feedforward) para prever a saída y_i.

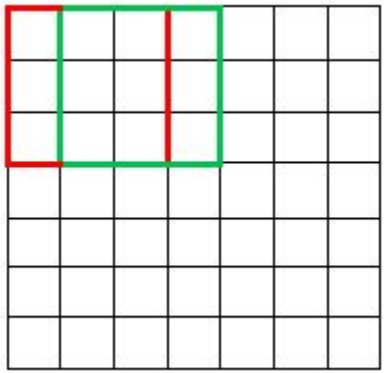
Imagem Convolvida

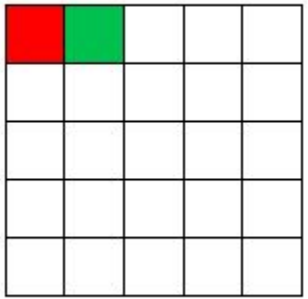


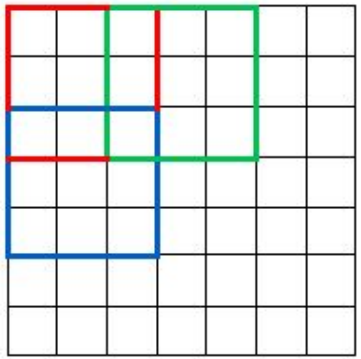
Fonte: Ravichandiran (2019)

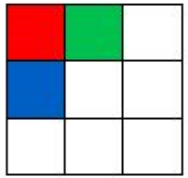
Fundamentação Teórica - Camada Convolucional - *Stride*

- É possível deslizar a matriz de filtro sobre a matriz de entrada com qualquer número de *pixels*;
- O número de *pixels* que é deslizado sobre a matriz de entrada é denominado de passo (*stride*);
- Se *stride* é definido como um número pequeno, pode-se codificar uma representação mais detalhada da imagem;
- Nas imagens abaixo: a direita temos *stride* igual a 1, a esquerda igual 2.





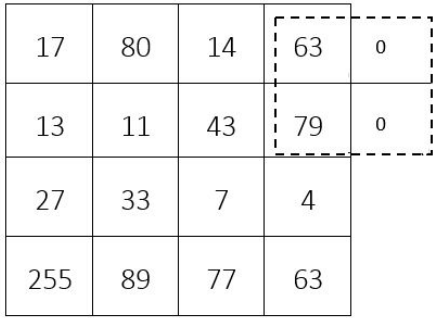




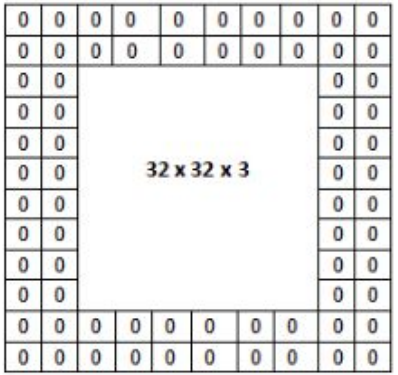
Fonte: Deshpande (2016)

Fundamentação Teórica - Camada Convolucional - *Padding*

- Quando move-se a matriz de filtro sobre a matriz de entrada, há um momento em que ela atinge a borda e a matriz de filtro não se ajusta completamente a matriz de entrada;
- Nesse caso é realizado a ação de preenchimento (*padding*), pode-se preencher a matriz de entrada além de sua borda;
- Essa ação é denominada de Preenchimento de Zero ou *Zero Padding*. Existem outros modelos de *padding.*



Fonte: Ravichandiran (2019)



Fonte: Deshpande (2016)

Fundamentação Teórica - Camada Convolucional - Retropropagação

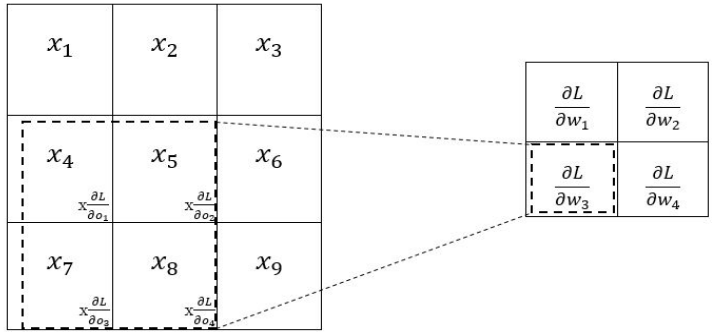
- Os valores ótimos da matriz do filtro, com os quais pode-se extrair os recursos importantes das imagens são obtidos por retropropagação;
- A operação de retropropagação se inicia após prever a saída, então é calculado a perda L;

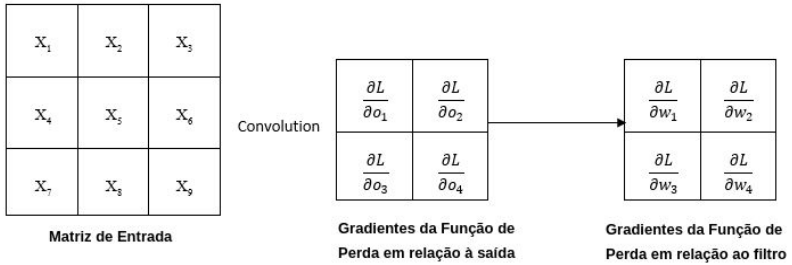
$$L = \frac{1}{2}\sum_i (y_i - \hat{y}_i)^2$$

- Para encontrar a matriz de filtro ideal, calcula-se os gradientes da função de perda L em relação a todos valores utilizando os valores da matriz de filtro original (w_1, w_2, w_3, w_4 - supondo a matriz de filtro vista na pág. 14 desta apresentação);
- As equações mostram somente o processo em w_1.

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial o_1}\cdot\frac{\partial o_1}{\partial w_1} + \frac{\partial L}{\partial o_2}\cdot\frac{\partial o_2}{\partial w_1} + \frac{\partial L}{\partial o_3}\cdot\frac{\partial o_3}{\partial w_1} + \frac{\partial L}{\partial o_4}\cdot\frac{\partial o_4}{\partial w_1}$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial o_1}\cdot\partial x_1 + \frac{\partial L}{\partial o_2}\cdot\partial x_2 + \frac{\partial L}{\partial o_3}\cdot\partial x_4 + \frac{\partial L}{\partial o_4}\cdot\partial x_5$$





Fonte: Ravichandiran (2019)

Fundamentação Teórica - Camada Convolucional - Retropropagação

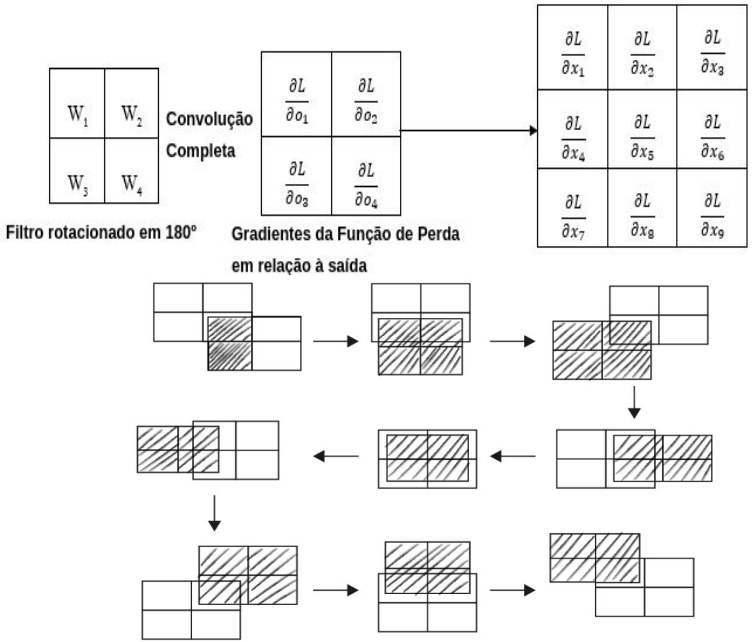
- É necessário também calcular o gradiente de perda em relação às entradas, pois é usado para calcular os gradientes dos filtros presentes na camada anterior;
- Será utilizada a matriz de entrada vista na pág. 14 dessa apresentação, ou seja, de x_1 à x_9.
- Analisando os valores de saída abaixo, x_1 está presente apenas em o_1, logo calcula-se os gradientes de perda em relação a o_1 sozinho, anulando os outros termos;

$$o_1 = x_1w_1 + x_2w_2 + x_4w_3 + x_5w_4$$
$$o_2 = x_2w_1 + x_3w_2 + x_5w_3 + x_6w_4$$
$$o_3 = x_4w_1 + x_5w_2 + x_7w_3 + x_8w_4$$
$$o_4 = x_5w_1 + x_6w_2 + x_8w_3 + x_9w_4$$

$$\frac{\partial L}{\partial x_1} = \frac{\partial L}{\partial o_1} \cdot \frac{\partial o_1}{\partial x_1}$$

$$\frac{\partial L}{\partial x_1} = \frac{\partial L}{\partial o_1} \cdot w_1$$

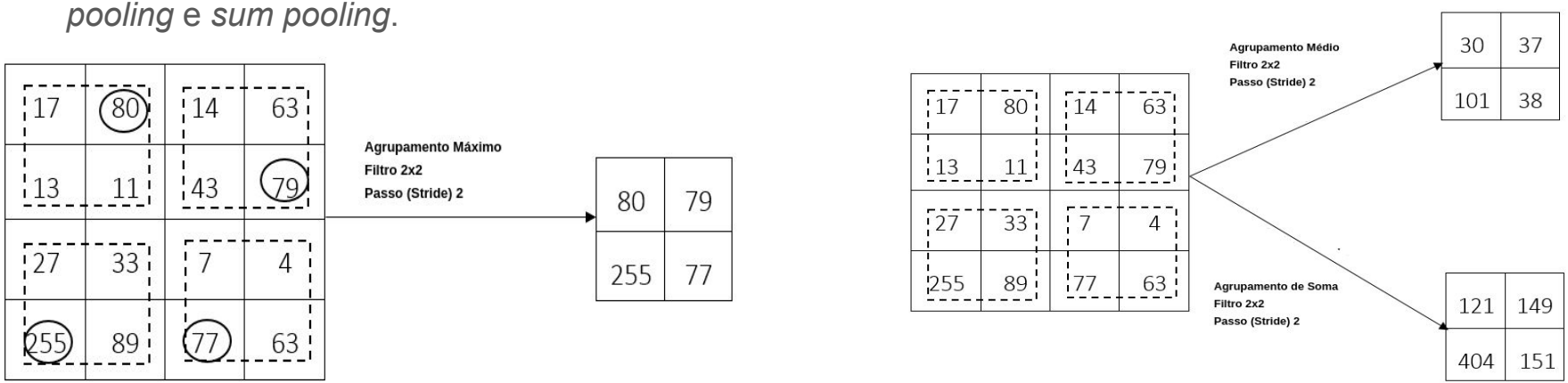
- Em vez de usar a matriz do filtro diretamente, é girada em 180 graus;
- Em vez de realizar convolução simples, realiza-se convolução completa, vista nas imagens abaixos.



Fonte: Ravichandiran (2019)

Fundamentação Teórica - Camada *Pooling*

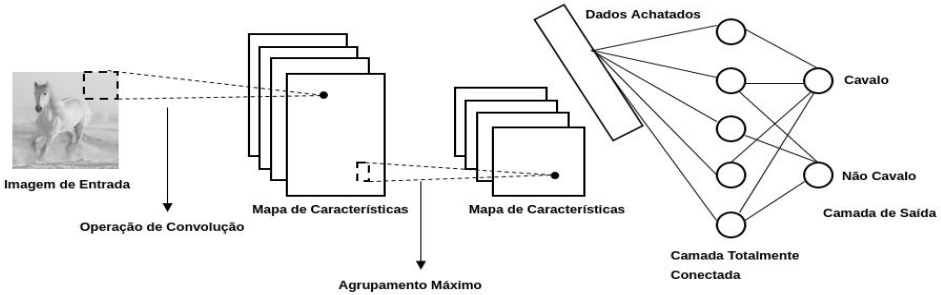
- Os mapas de características resultantes são muito grandes em dimensão. Para reduzi-los realiza-se uma operação de agrupamento (*pooling*);
- Isso mantém apenas os detalhes necessários para que a quantidade de processamento seja reduzida;
- Existem diferentes tipos de operações de agrupamento, incluindo *max pooling*, *average pooling* e *sum pooling*;
- O *max pooling* é o mais utilizado, um filtro é deslizado na matriz convolvida obtendo o valor máximo da janela de filtro, como mostra a imagem a esquerda. À direita temos as operações *average pooling* e *sum pooling*.



Fonte: Ravichandiran (2019)

Fundamentação Teórica - Camada Totalmente Conectada

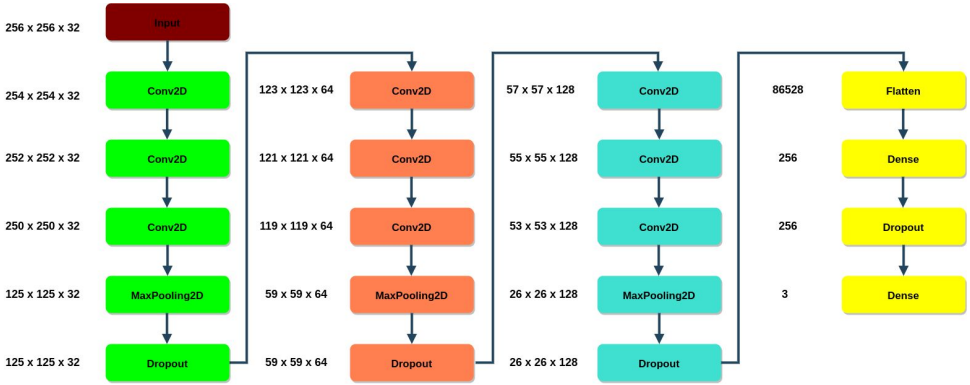
- Camadas Convolucionais extraem recursos da imagem produzindo um mapa de características;
- É preciso classificar esses características extraídos;
- Utiliza-se um algoritmo que classifique esses características e prediz se os são características de um cavalo ou de outra coisa;
- Na Camada Totalmente Conectada, o mapa de características é nivelado e o convertido em um vetor;
- Logo após, usado como entrada para a rede *feedforward* que aplica uma função de ativação e retorna a saída, informando se a imagem contém um cavalo ou não.



Fonte: Ravichandiran (2019)

Metodologia - Arquitetura *CNN* utilizada

- Está pesquisa se caracteriza como aplicada, quantitativa, metodológica;
- Foi utilizada uma arquitetura sequencial de três camadas idênticas com diferentes tamanhos de filtro;
- Possuem três módulos convolucionais (*Conv2D*), seguido de um módulo de pooling (*MaxPooling2D*) e por fim um módulo *Dropout;*
- A primeira camada em verde possui um filtro de tamanho 32, a segunda em laranja, 64, e a terceira em azul, 128;
- Após a terceira camada há uma sequência de quatro módulos que formam a Camada de Classificação: um módulo *Flatten*, um módulo *Dense*, um módulo *Dropout* seguido de outro módulo *Dense*.



Fonte: O Autor

Metodologia - Base de Dados

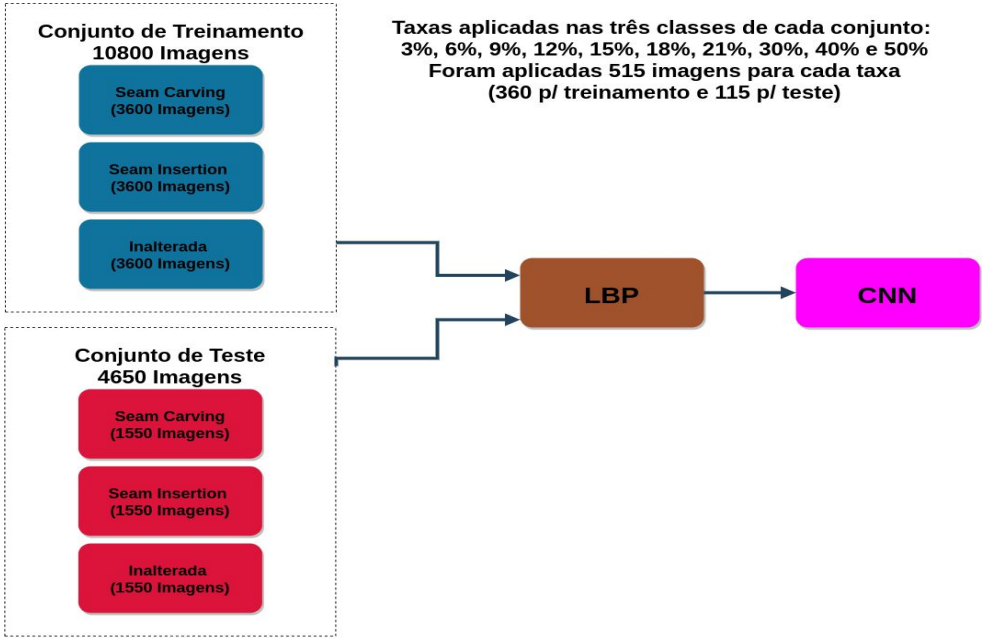
- Para realização do treinamento e teste da *CNN* foi utilizado uma base de dados que possui 5150 imagens em cores brutas não compactadas, de dimensão 256×256 pixels, no formato *Windows Bitmap (BMP)* (LIU; COOPER; ZHOU, 2013) <https://www.shsu.edu/qxl005/New/Downloads/index.html>

Metodologia - Execução do Projeto

- O processo iniciou adulterando a base de dados original;
- As 5150 imagens foram divididas em 10 taxas (3%, 6%, 9%, 12%,15%, 18%, 21%, 30%, 40% e 50%) distintas de adulteração para *Seam Carving e Seam Insertion*;
- O correto seria formar 10300 imagens, porém sem intenção 500 imagens foram passadas a mais para a taxa 3% na execução manual dos algoritmos, formando 10800 imagens para teste (70%, 3600 imagens para cada uma das três classes) e 4650 para treinamento (30%, 1550 imagens para cada uma das três classes);
- Para a classe de Imagens Intocadas apenas foi executado o *LBP* em 5150 (3600 para treinamento e 1550 para teste).

Metodologia - Execução do Projeto

- Na aplicação do *Seam Carving,* utilizando como exemplo a taxa de 50% do conjunto de treinamento, as 360 imagens foram alteradas em 25% de linhas de *pixels* e 25% de colunas de *pixels* em relação a imagem original;
- Após todas as alterações de todas as imagens nas diferentes taxas de cada classe, foi realizado *LBP* em todas as imagens, conjunto de teste e treinamento;
- Todo o processo de aplicação dos algoritmos *Seam Carving*, *Seam Insertion* e *LBP* para montagem da base de dados e os algoritmos referentes a *CNN* foram escritos na linguagem de programação *Python;*
- Foram utilizadas as bibliotecas *tensorflow*, *sklearn*, *matplotlib*, *numpy* e *pandas*. Todo o código de programação foi escrito manualmente pelo autor e estão disponíveis no GitHub do mesmo. <https://github.com/GabesSeven/TCC_-_Trabalho_de_Conclusao_de_Curso_-_Course_Conclusion_Work>



Fonte: O Autor

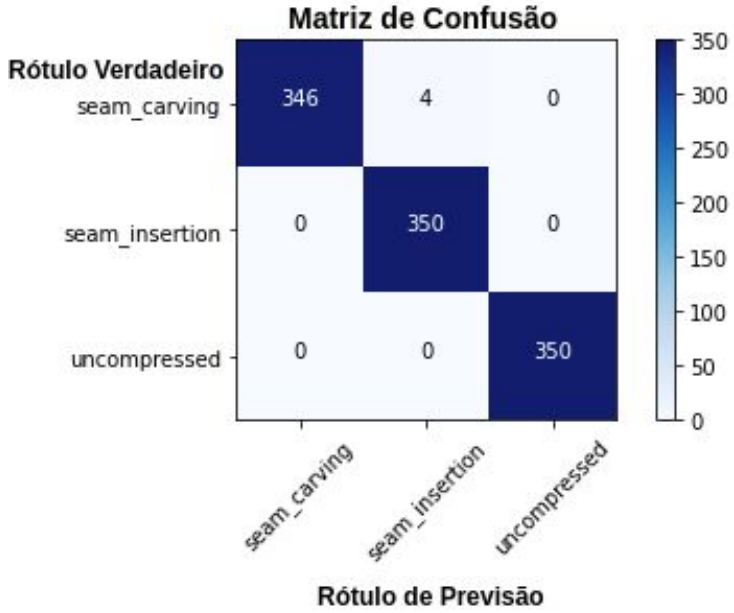
Resultados e Discussão

- Época se refere a um ciclo em todo o conjunto de dados de treinamento, se alimentarmos uma rede neural com os dados de treinamento por mais de uma Época em padrões diferentes;
- A Precisão é uma forma de medir a frequência com que o algoritmo classifica um ponto de dados corretamente;
- Função de Perda mede o índice de erro do modelo da rede;
- O Treinamento foi realizado utilizando vinte Épocas, para cada uma a rede divide os dados em duas partes, dados de treinamento (*loss* e *accuracy*) e dados de validação (*val_loss* e *val_acc*);
- Neste projeto, na maioria dos casos, com cada Época aumentando, a Perda (*loss*) diminuiu e a Precisão (*accuracy*) aumentou, também *val_loss* diminuiu e *val_acc* aumentou;
- O algoritmo funcionou corretamente, significa que a construção do modelo está aprendendo e funcionando bem. A acurácia final foi de 99%.

epoch	*loss*	*accuracy*	*val_loss*	*val_acc*
1	0.8617	0.4862	0.4369	0.7524
2	0.3241	0.8457	0.1644	0.9371
3	0.1542	0.9376	0.1644	0.9371
4	0.1111	0.9559	0.1018	0.9590
5	0.0887	0.9665	0.1004	0.9581
6	0.0762	0.9713	0.0441	0.9848
7	0.0668	0.9727	0.0504	0.9838
8	0.0528	0.9804	0.0281	0.9924
9	0.0515	0.9800	0.0328	0.9848
10	0.0466	0.9838	0.0126	0.9981
11	0.0366	0.9858	0.0143	0.9933
12	0.0401	0.9846	0.0352	0.9857
13	0.0319	0.9883	0.0134	0.9971
14	0.0303	0.9890	0.0119	0.9962
15	0.0215	0.9927	0.0132	0.9971
16	0.0236	0.9912	0.0085	0.9981
17	0.0222	0.9926	0.0112	0.9981
18	0.0196	0.9924	0.0045	0.9990
19	0.0168	0.9934	0.0040	0.9990
20	0.0183	0.9933	0.0106	0.9962

Fonte: O Autor

Resultados e Discussão



Fonte: O Autor

- A rede classificou apenas quatro dos exemplos incorretamente, como mostra a Matriz de Confusão na imagem acima.

Conclusão

- Este trabalho propôs uma *Convolutional Neural Network (CNN)* para detectar fraudes em imagens causadas por uma técnica denominada *Seam Carving*;
- Muitos trabalhos estão sendo produzidos envolvendo técnicas de Deep Learning e segurança da informação, este é um tema importante para a área da tecnologia;
- Este gênero de trabalho contribui para criação de técnicas que impeçam fraudes e aquisição ilegal de imagens;
- Os resultados obtidos foram ótimos, pois a Rede Neural obteve 99% de acurácia, comparando com os autores (CIESLAK;COSTA; PAPA, 2018) (NAZARİ; AKGÜN, 2020) citados na introdução, a rede obteve resultados muito próximos. Ambos autores utilizaram metodologia muito semelhante em relação a Rede Neural utilizada e são trabalhos recentes;
- Uma abordagem futura poderia ser utilizar outros modelos de redes mais complexos para analisar de uma forma mais ampla o método *Seam Carving*.

Referências

- AVIDAN, S.; SHAMIR, A. **Seam carving for content-aware image resizing**. In: . New York,NY, USA: Association for Computing Machinery, 2007. (SIGGRAPH '07), p. 10–es. ISBN 9781450378369. Disponível em: <https://doi.org/10.1145/1275808.1276390>.
- CHOI, C.-H.; LEE, H.-Y.; LEE, H.-K. **Estimation of color modification in digital images by cfa pattern change**. Forensic Science International, v. 226, n. 1, p. 94 – 105, 2013. ISSN 0379-0738.Disponível em: <http://www.sciencedirect.com/science/article/pii/S0379073812005518>.
- CIESLAK, L. F. d. S.; COSTA, K. A. P. d. C.; PAPA, J. P. **Seam carving detection using convolutional neural networks.** In: 2018 IEEE 12th International Symposium on Applied Computational Intelligence and Informatics (SACI). [S.l.: s.n.], 2018. p. 000195–000200.
- DESHPANDE, A. **A BEGINNER'S Guide To Understanding Convolutional Neural Networks Part 2**. [S. l.], 2016. Disponível em: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/. Acesso em: 10 dez. 2020.
- HE, D.-C.; WANG, L. **Texture unit, texture spectrum, and texture analysis**. IEEE Transactionson Geoscience and Remote Sensing, v. 28, p. 509–512, 1990.
- HUANG, D.; SHAN, C.; ARDABILIAN, M.; WANG, Y.; CHEN, L. **Local binary patterns and its application to facial image analysis: A survey**. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), v. 41, n. 6, p. 765–781, Nov 2011. ISSN 1558-2442.
- LYKOV, K. **SEAM Carving Algorithm**. [S. l.], 2013. Disponível em: https://kirilllykov.github.io/blog/2013/06/06/seam-carving-algorithm/. Acesso em: 10 dez. 2020.
- NAZARİ, H.; AKGÜN, D. **A deep learning model for image retargetting level detection**. In:2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies(ISMSIT). [S.l.: s.n.], 2020. p. 1–4.
- PATTERSON, J.; GIBSON, A. **Deep Learning: A Practitioner's Approach**. 1st. ed. [S.l.]:O'Reilly Media, Inc., 2017. ISBN 1491914254.
- PIETIKINEN, M.; HADID, A.; ZHAO, G.; AHONEN, T. **Computer Vision Using Local Binary Patterns**. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2011. ISBN 0857297473.
- PRADO, K. **Face Recognition: Understanding LBPH Algorithm**. [S. l.], 2017. Disponível em: https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b. Acesso em: 10 dez. 2020.
- RAVICHANDIRAN, S. **Hands-On Deep Learning Algorithms with Python. Packt Publishing**, 2019. ISBN 9781789344158. Disponível em: <https://books.google.com.br/books?id=MYfDwQEACAAJ>.
- YIN, T.; YANG, G.; LI, L.; ZHANG, D.; SUN, X. **Detecting seam carving based image resizing using local binary patterns**. Comput. Secur., Elsevier Advanced Technology Publications, GBR, v. 55, n. C, p. 130–141, nov. 2015. ISSN 0167-4048. Disponível em:<https://doi.org/10.1016/j.cose.2015.09.003>

.