

# Neural Style Transfer em Produção

1<sup>st</sup> Bruno Uhlmann Marcato  
*Engenharia de Computação*  
*UTFPR - Apucarana*  
marcato.2001@alunos.utfpr.edu.br

2<sup>nd</sup> Cristian Andre Sanches  
*Engenharia de Computação*  
*UTFPR - Apucarana*  
cristiansanches@alunos.utfpr.edu.br

3<sup>rd</sup> Guido Margonar Moreira  
*Engenharia de Computação*  
*UTFPR - Apucarana*  
guidomoreira@alunos.utfpr.edu.br

**Abstract**—Este projeto aborda a transferência de estilo em imagens digitais, explorando métodos avançados de Machine Learning, como o Neural Style Transfer (NST) e o Fast Neural Style Transfer (FNST). O projeto propõe uma aplicação prática que combina o conteúdo de uma imagem com o estilo de outra. O FNST é adotado devido à sua eficiência em termos de velocidade e aplicabilidade em ambientes de produção. Seis estilos artísticos são oferecidos na aplicação, treinados em um conjunto de dados diversificado.

**Index Terms**—Web Application, Style Transfer, Convolutional Neural Network

## I. INTRODUÇÃO

O crescente desenvolvimento tecnológico tem contribuído para a universalização do acesso a diferentes formas de arte por uma grande parcela da população, podendo acessar pela internet músicas, filmes, vídeos, ilustrações entre outras além de ter gerado novas ferramentas possibilitando a criação de campos artísticos completamente inovadores, sejam nas áreas de imagens geradas por computador (do inglês *Computer Generated Images*, CGI), Ilustração digital, animação 3D e 2D, entre outras, porém assim como a câmera fotográfica substituiu a pintura para a maioria das pessoas quando se trata de registrar momentos, os modelos de inteligência artificial geradoras de imagem tendem a gerar um impacto de proporção semelhantes no mundo da arte digital, no caso a aplicação de efeitos visuais instantâneos a fotos é algo que já está presente em inúmeras redes sociais tais como: *Instagram*, *Snapchat* e *TikTok*, tendo em vista esse tipo de ferramenta este projeto tem por objetivo a criação de uma aplicação capaz de realizar transferência de estilo, que consiste em pegar o estilo de uma imagem e aplicar em outra.

Para tal tarefa serão usadas ferramentas de *Machine Learning* (ML), mais especificamente técnicas de *Deep Learning* (DL) voltados a área de processamento de imagens, que precisaram ser treinadas para lidar com o conjunto de dados preparados e gerar imagens com o resultado desejado, recebendo duas imagens de entrada: a de estilo e a de conteúdo. É esperado que a imagem na saída herde várias características da imagem de estilo como: traço e contorno do pincel, estilo das cores, estilo de contorno, texturas, entre outros.

Este trabalho está organizado da seguinte forma: A seção II descreve a revisão bibliográfica e trabalhos relacionados; Já na seção III são apresentados os conceitos teóricos necessários para a compreensão do texto; A seção IV descreve a metodologia de implementação dos algoritmos propostos; na seção V

se encontram os resultados experimentais; e finalmente, a Seção VI apresenta as conclusões e sugestões para trabalhos futuros.

## II. TRABALHOS RELACIONADOS

Grande parte dos trabalhos relacionados à técnica de transferência de estilo utilizando redes neurais, ou *neural style transfer* (NST) concentra-se em aprimorar e aperfeiçoar o algoritmo original. Essas melhorias podem envolver ajustes em aspectos específicos do processo, como a introdução de novas funções de perda para capturar melhor os detalhes ou a otimização da eficiência computacional. Em essência, esses estudos visam encontrar maneiras de tornar a transferência de estilo mais eficaz, realista e flexível para diversas aplicações.

Em [1], os autores propõem um método que une a versatilidade do NST com a eficiência das redes de transferência rápida de estilo (do inglês *Fast Neural Style Transfer*, FNST). A abordagem permite estilização em tempo real com qualquer combinação de imagens de conteúdo e estilo. O modelo aprende a prever os parâmetros diretamente a partir da imagem de estilo e é treinado em um amplo conjunto de pinturas. Além disso, o estudo destaca a capacidade do modelo de generalizar para estilos de pintura não observados anteriormente, oferecendo informações semânticas das pinturas.

No trabalho feito em [2], os autores propuseram uma aplicação prática de NST para *Data Augmentation* destacando seu impacto positivo nas tarefas de classificação de imagens. Os resultados mostram melhorias significativas na precisão da classificação para modelos como *VGG16* e *VGG19*. A pesquisa também menciona a possibilidade de estender essa técnica para outras tarefas de visão computacional, como segmentação e detecção de objetos, e enfatiza a importância do desenvolvimento de modelos de transferência de estilo neural mais rápidos e eficientes.

Em trabalhos na qual são criadas aplicações específicas para a técnica de NST, é possível citar o trabalho feito em [3], onde é demonstrado como a ferramenta *ImagineNet* pode ser aplicada na prática para reestilizar aplicativos móveis de várias maneiras. Isso inclui reestilizar os ativos gráficos de um aplicativo, reestilizar um aplicativo com conteúdo fornecido pelo usuário e até mesmo reestilizar um aplicativo com interfaces gráficas de usuário (*Graphic User Interfaces - GUIs*) geradas dinamicamente. O artigo destaca a capacidade do modelo de transferência de estilo em reter detalhes de conteúdo enquanto adiciona características estilísticas, como

textura, sombras e contraste, melhorando a coesão do design. Essa abordagem permite resultados visualmente atraentes e distintos em aplicativos móveis.

### III. FUNDAMENTAÇÃO TEÓRICA

A transferência de estilo (do inglês *Style Transfer*, ST) é uma técnica de visão computacional que combina o conteúdo de uma imagem com o estilo de outra, criando uma nova imagem que preserva as informações do conteúdo, mas aplica o estilo da segunda imagem. Esse processo é realizado através da otimização iterativa de uma terceira imagem, de forma a minimizar a diferença entre o conteúdo da imagem original e o conteúdo da imagem gerada, ao mesmo tempo em que minimiza a diferença entre o estilo da imagem gerada e o estilo da imagem de referência.

Exemplos de algoritmos de ST são as técnicas *image analogy* [4] e *image quilting* [5]. A primeira técnica se baseava em um par de imagens de treinamento, uma foto A e uma obra de arte A' representando essa foto, uma transformação poderia ser aprendida de forma supervisada e então aplicada para criar uma nova obra de arte a partir de uma nova foto B, por analogia (Figura 1). Por questões práticas, esse algoritmo é inviável na maioria das situações, dado que o par de imagens de entrada raramente existem.



Fig. 1: Exemplo da técnica de *image analogy* do artigo original [4]

#### A. Neural Style Transfer

As redes neurais profundas (DNN) ultrapassaram o desempenho do nível humano em tarefas como reconhecimento e detecção de objetos [6]. Em 2015 surge a técnica de NST [7], afim de utilizar as vantagens das DNN's para a tarefa de transferência de estilo. Especificamente, os autores utilizaram redes neurais convolucionais (CNN) pela sua performance conhecida em tarefas de processamento de imagem. O NST trata do processo de otimização de uma imagem sendo gerada com o conteúdo e estilo das imagens de entrada. A otimização ocorre da mesma forma que outras técnicas de ML, um algoritmo otimizador iterativamente reduz o valor de uma

função de perda, com a diferença de que o modelo é mantido estático, isto é, sem mudança nos seus pesos, onde a entrada da rede (imagem gerada) é otimizada. O algoritmo é descrito nas seguintes etapas:

1) *Entradas*: As entradas do algoritmo são (i) a imagem de conteúdo ( $I_c$ ); (ii) A imagem de estilo ( $I_s$ ) e; (iii) A imagem gerada ( $I_g$ ), inicializada com ruído ou como uma cópia da imagem de conteúdo. Exemplos de entrada podem ser vistos na figura 2.

2) *Extração de características*: Como extrator de características é utilizado o modelo VGG19 [8], uma arquitetura de CNN amplamente utilizada que se destaca por sua simplicidade e profundidade. Ela possui 19 camadas, o que a torna suficientemente profunda para capturar representações de alto nível, como texturas, padrões e objetos. Ao mesmo tempo, sua simplicidade a torna uma escolha computacionalmente eficiente. A rede é pré-treinada no conjunto de dados ImageNet [9], afim de economizar tempo e recursos, uma vez que a rede já aprendeu representações de uma gama diversificada de imagens.

3) *Representação de conteúdo*: A partir das características extraídas da rede, podemos fazer a representação do conteúdo de  $I_c$ , afim de transferi-lo para  $I_g$ . Para isso, usamos os valores de uma camada da CNN após passar a imagem de conteúdo por ela. Normalmente são utilizadas camadas mais profundas na rede, pois nelas a saída tende a preservar o conteúdo da imagem e a estrutura espacial, mas a cor, a textura e a forma exata não [10].

Com a representação do conteúdo, definimos a função de perda  $\mathcal{L}_{content}$  como o erro quadrático médio (MSE) entre as representações de conteúdo de  $I_c$  e  $I_g$ . Isto significa que ambas as imagens são passadas pela rede, onde são extraídos os mapas de características na camada selecionada. Os filtros gerados são achatados e concatenados afim de termos uma matriz com todas as representações geradas. Esse processo resulta em duas matrizes, uma para  $I_c$  e outra para  $I_g$ , com dimensão  $m \times n$ , sendo  $m$  o número de filtros gerados e  $n$  o número de pixels das imagens. Matematicamente, a função de custo é definida como

$$\mathcal{L}_{content}(\vec{I}_c, \vec{I}_g, l) = \frac{1}{2} \sum_{i,j} (G_{ij}^l - C_{ij}^l)^2 \quad (1)$$

onde  $G$  e  $C$  são as matrizes de filtros de  $I_g$  e  $I_c$ , respectivamente, e  $l$  é a camada convolucional selecionada para representação do conteúdo.

4) *Representação de estilo*: Da mesma forma, a representação do estilo de uma imagem pode ser obtida a partir dos filtros obtidos por camadas convolucionais da rede. Como características geradas por essas camadas contém informações espaciais da imagem de entrada, uma etapa a mais é necessária para extrair o estilo dessa imagem. O método proposto para diferenciar o conteúdo do estilo é utilizar matrizes de Gram ( $G$ ) para calcular a correlação entre as características geradas para representar o conteúdo.



(a) Imagem de conteúdo.

(b) Imagem de estilo.

(c) Imagem com ruído.

Fig. 2: Imagens de entrada do NST

Matematicamente temos

$$G_{ij}^l = \sum_k V_{ik}^l V_{jk}^l \quad (2)$$

onde  $V$  é a matriz de filtros gerada pela imagem de entrada. A equação nos diz que o elemento  $(i,j)$  da matriz de Gram é igual à soma do produto dos elementos de duas linhas distintas na matriz de conteúdo. Esta operação é semelhante ao produto escalar (ou produto interno) entre dois vetores. Por este motivo a matriz de Gram é considerada como um espaço de produto escalar. Podemos simplificar a equação 2 usando a representação matricial de  $V$ .

$$G = VV^T \quad (3)$$

ou seja, obtemos a matriz de Gram a partir da multiplicação entre a matriz de representação de conteúdo e sua transposta. Essa matriz nos fornece, então, uma forma de representar o estilo da imagem de entrada.

Para entender o motivo da matriz de Gram conseguir separar o conteúdo do estilo, lembre-se de que cada linha da matriz é um mapa de características achatado, contendo informações espaciais. Ao multiplicar os elementos, verificamos se eles se sobreponem em sua localização na imagem. Isso é feito para todos os locais da imagem e com todos os pares possíveis de filtros. A soma é então realizada, descartando toda a relevância espacial da informação. Essa sobreposição é também chamada de correlação, isto é, cada elemento  $(i,j)$  de  $G$  mede quão relacionado a característica  $i$  está de  $j$ . Uma alta correlação entre duas características diz que as duas tendem a aparecer juntas frequentemente. Logo, todas as correlações entre as características diz como a imagem de entrada está estilizada.

Com a representação do estilo, definimos a função de perda  $\mathcal{L}_{style}$  como o erro quadrático médio (MSE) entre as representações de estilo de  $I_s$  e  $I_g$ .

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^l - S_{ij}^l)^2 \quad (4)$$

onde  $G$  e  $S$  são as matrizes de Gram da imagem gerada e da imagem de estilo, respectivamente;  $N$  representa o número de filtros na camada  $l$  e  $M$  representa o número de elementos nesses  $N$  filtros.

No trabalho original [7], os autores afirmam que os melhores resultados gerados surgem ao comparar o estilo em múltiplas camadas. Logo, a função de perda para o estilo é definida como a soma ponderada do erro em cada uma das camadas.  $E_l$  e  $w_l$  são o erro e o peso da camada  $l$ , respectivamente.

$$\mathcal{L}_{style}(\vec{I}_s, \vec{I}_g) = \sum_{l=0}^L w_l E_l \quad (5)$$

5) *Otimização:* A otimização do NST é realizada por meio de um otimizador, comumente o gradiente descendente ou suas variações como o *Adam* [11]. A tarefa consiste em ajustar os valores dos pixels na imagem de saída  $I_g$  para minimizar a função de custo total  $\mathcal{L}_{total}$ . Isso envolve calcular os gradientes da função de custo em relação aos pixels da imagem de saída e, então, ajustar os valores dos pixels de forma iterativa. A função de custo é definida como a soma entre as funções de custo de conteúdo e estilo.

$$\mathcal{L}_{total}(\vec{I}_c, \vec{I}_s, \vec{I}_g) = \alpha \mathcal{L}_{content}(\vec{I}_c, \vec{I}_g) + \beta \mathcal{L}_{style}(\vec{I}_s, \vec{I}_g) \quad (6)$$

onde  $\alpha$  e  $\beta$  são os pesos para reconstrução de conteúdo e estilo, respectivamente. Após o processo de otimização, a imagem gerada  $I_g$  possui o conteúdo de  $I_c$  mesclado ao estilo de  $I_s$  (Figura 3).

### B. Fast Neural Style Transfer

O algoritmo de NST descrito anteriormente é classificado como um método baseado em otimização, isto é, a rede permanece a mesma, apenas a entrada é otimizada. Por isso, a técnica é inviável para a utilização em ambientes de produção, tendo em vista a velocidade da geração da imagem estilizada, além da necessidade de repetir todo o processo de otimização quando a imagem de conteúdo é mudada. Com isso em mente [12] e [13] trabalharam estes problemas de forma independente, ambos propondo métodos baseados em redes *feed-forward*.

Para exemplificação [12] será usado. A técnica se baseia em treinar uma rede de transformação de imagens (*Transform Network*)  $f_W$  para realizar a transferência de estilo, ao contrário de otimizar uma imagem, conforme proposto



Fig. 3: Imagem final gerada após o processo de otimização

originalmente por [7]. A rede é uma DR-CNN (*Deep residual convolutional neural network*), com um conjunto de pesos  $W$ , que mapeia uma imagem  $x$  para uma imagem de saída  $\hat{y}$  por  $\hat{y} = f_W(x)$ . Além disso, é definida uma rede de perda (*Loss Network*)  $\phi$  que tem a mesma tarefa da rede utilizada no método anterior: Extrair características da imagem de entrada para o cálculo das funções de perda do conteúdo ( $l_{feat}^\phi$ ) e do estilo ( $l_{style}^\phi$ ). A rede  $\phi$  segue as mesmas regras do método anterior, normalmente é utilizada a rede VGG-16 ou VGG-19, pré-treinada no conjunto de dados *ImageNet*.

A função de perda total é definida da mesma forma que na equação 6 com a adição de uma nova função de perda chamada de *total variation loss* ( $l_{TV}^\phi$ , que é a soma das diferenças absolutas dos valores de pixel vizinhos nas imagens de entrada. Isso mede a quantidade de ruído nas imagens. Ao minimizar essa nova função, a imagem final é forçada a manter uma suavização, melhorando a sua qualidade visual. A função de perda total é definida matematicamente da seguinte forma:

$$l_{total}^\phi = \lambda_c l_{feat}^\phi(y, yc) + \lambda_s l_{style}^\phi(y, ys) + \lambda_{TV} l_{TV}^\phi(y) \quad (7)$$

onde  $\lambda_c$ ,  $\lambda_s$  e  $\lambda_{TV}$  são os pesos para cada função;  $y$  é a imagem gerada por  $f_W$ , inicializada como ruído;  $yc$  e  $ys$  são as imagens de conteúdo e de estilo, respectivamente. A visão geral do método pode ser visto na figura 4.

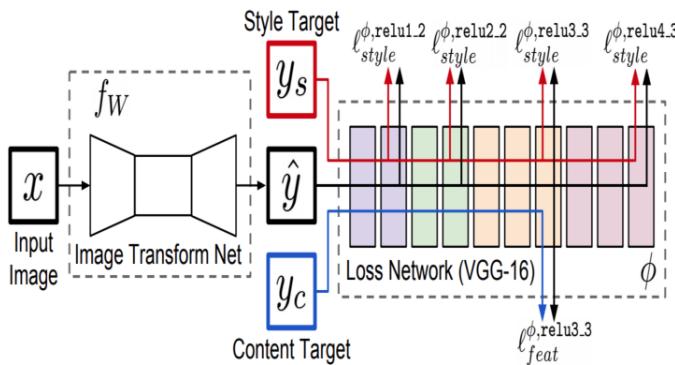


Fig. 4: Visão geral do método proposto por [12]

#### IV. METODOLOGIA

Nosso sistema é composto por um *layout* simples, onde o usuário preencherá algumas opções disponíveis, e o sistema exibirá a imagem estilizada de acordo com as especificações. A figura 5 mostra um fluxograma das opções disponíveis ao usuário.

Métodos de NST baseados em otimização se mostram eficientes em uma grande variedade de problemas, porém falha no quesito velocidade, fator importante para aplicações reais do problema, como transferência de estilo em vídeos, ou simples requisições de usuários. Nesse sentido, o método utilizado foi o FNST. Portanto, 1 modelo diferente é treinado para cada um dos 6 estilos disponibilizados.

##### A. Estilos

Ao total são 6 modelos disponíveis na aplicação. A figura 6 mostra cada uma das obras de arte utilizadas, assim como seus nomes.

##### B. Treino

Para o criação dos modelos o conjunto de dados utilizado neste trabalho foi o MS-COCO 2014 [14]. Que é amplamente reconhecido na comunidade de visão computacional e fornece mais de 83.000 imagens que abrangem várias categorias como: pessoas, animais, meios de transporte, comidas, objetos do cotidiano entre outros. Alguns exemplos podem ser observados na Figura 7.



Fig. 7: Imagens exemplo do dataset [14].

Antes do treinamento, as imagens do conjunto de dados são submetidas a um processo de pré-processamento. Onde tem seu tamanho alterado para um formato adequado ao modelo e por um processo de normalização para garantir consistência e estabilidade durante o treinamento por meio da biblioteca OpenCV [15]<sup>1</sup>.

O treinamento foi conduzido por uma única época, onde cada imagem no conjunto de dados foi utilizada uma vez. Dado o tamanho significativo do conjunto de dados MS-COCO, uma única época foi suficiente para o modelo extrair as características relevantes.

<sup>1</sup><https://opencv.org/>

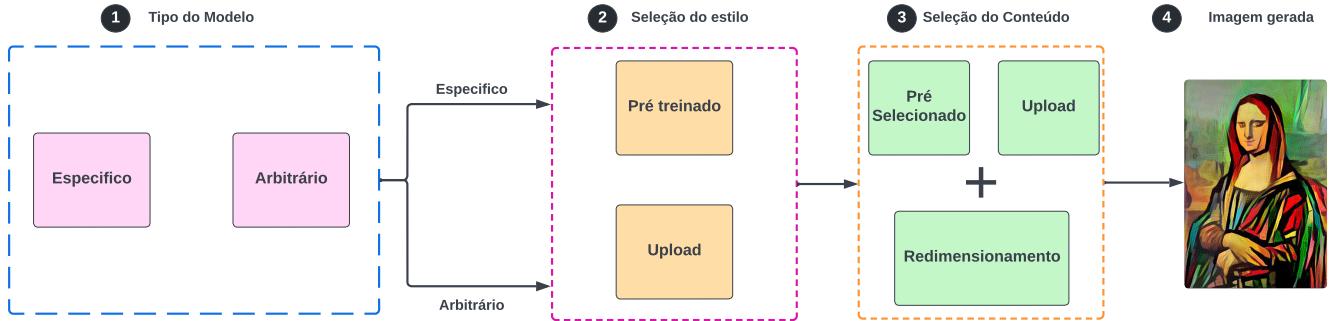


Fig. 5: Fluxograma da aplicação



Fig. 6: Estilos disponibilizados na aplicação

Durante o treinamento do modelo foi utilizado o dataset inteiro, enquanto para uma validação parcial, durante o treinamento, foram utilizados imagens do próprio treino. Assim a cada 500 iterações o modelo é salvo para realizar testes com as imagens de treinamento ainda não usadas. O Batch Size foi definido como 4 e a seed aleatória para todas as operações no treinamento do modelo foi 42 (o que garante a reprodução dos resultados que serão obtidos).

Dos modelos neste projeto três dos utilizados foram adaptados do trabalho do FNST [12] com o fim de comparação para confirmar se o desempenho dos modelos criados manualmente é capaz de se igualar ou até superar o desempenho dos modelos originais.

### C. Aplicativo

Com os modelos já treinados e pronto para uso, fica faltando apenas criar uma plataforma onde seja possível acessar o Style Transfer, para esse fim foi escolhido o Streamlit, uma framework open-source criada com o fim de facilitar a criação de aplicativos de machine learning e data science, ele fornece uma biblioteca para Python com o mesmo nome<sup>2</sup> onde é possível configurar os componentes do site que chamaram as funções do modelo, como se pode observar no exemplo na Figura 8, e colocar o site no ar sem custo.

<sup>2</sup><https://docs.streamlit.io/library/get-started>

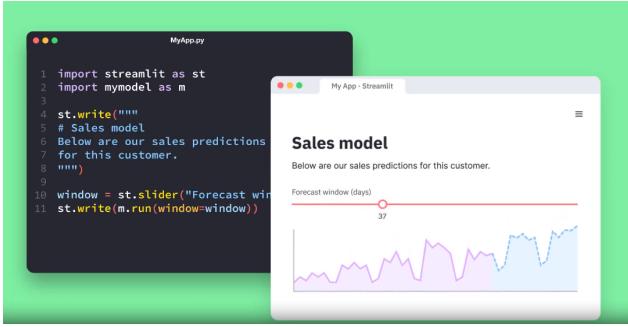


Fig. 8: Exemplo aplicação criada no Streamlit.

## V. RESULTADOS E DISCUSSÃO

Com os modelos treinados e as configurações feitas utilizando a biblioteca do streamlit o site foi colocado no ar<sup>3</sup> contando com as opções de selecionar os estilos, a imagem padrão, o tamanho da imagem gerada como se pode observar na Figura 9 e a possibilidade de inserir uma imagem de estilo de entrada pelo *upload* de arquivo 10.



Fig. 9: Aplicação rodando no Streamlit com imagens padrão.

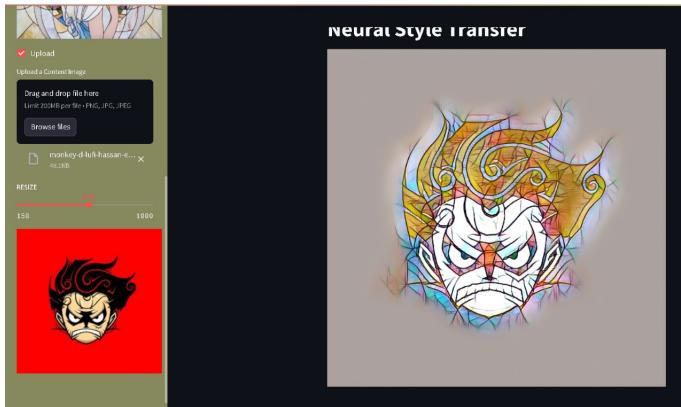
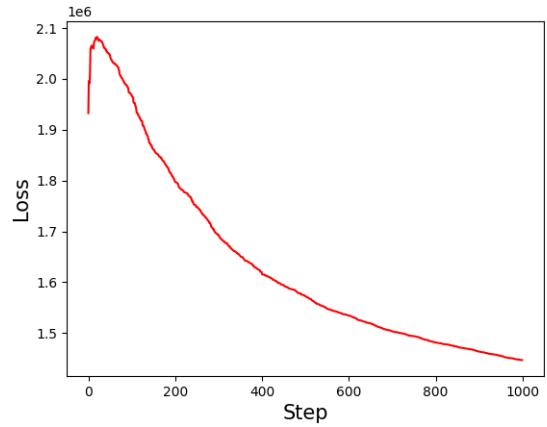


Fig. 10: Aplicação rodando no Streamlit com imagens por *upload*.

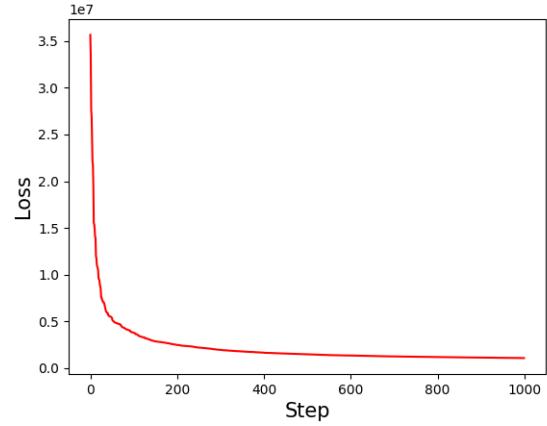
<sup>3</sup><https://brunomarcato-neuralstyletransferapp-app-cbp27l.streamlit.app/>

Rodando a aplicação utilizando os estilos padrões(6 estilos: mosaico, Picasso, Noite Estrelada, O Grito, Udnie e Ondas.) nas imagens padrões disponibilizadas (no total 9 imagens) os resultados obtidos podem ser observados em 11, as imagens de a-c foram geradas pelos modelos criados manualmente enquanto as de d-f utilizaram os modelos já prontos, a análise dos resultados mostra a eficiência no modelo visto que mesmo com a aplicação dessa mudança drástica de estilos a silhueta das imagens originais se mantiveram e alguns dos resultados seriam capaz de se passar pela arte feita por um artista humano.

Além da saída do modelo foram gerados os gráficos 12, onde estão presentes a perda de conteúdo 12a e a perda do estilo 12b, a análise de ambos os gráficos deixa claro como o modelo consegue extrair as características referentes ao estilo da imagem com bem mais eficiência que o conteúdo visto que a perda do estilo levou pouco tempo para convergir enquanto a perda de conteúdo provavelmente não aparenta ter chegado no limite.



(a) Perda de Conteúdo



(b) Perda de Estilo

Fig. 12: Perda de Conteúdo e de Estilo



Fig. 11: Aplicação dos estilos padrão nas imagens *Orange* e *Vintage*.

## VI. CONCLUSÃO

Por meio da análise dos resultados obtidos, pelos modelos de geração de imagem Fast Style Transfer utilizando o VGG19, foi possível constatar um desempenho satisfatório ao objetivo do projeto visto que as imagens geradas mantiveram seu conteúdo enquanto aplicavam os estilos desejados, tendo os resultados em par com os outros modelos analisados no projeto. Além disso o desempenho referente ao tempo de resposta da aplicação no streamlit seria aceitável em um contexto de mercado com delay para processamento não passando de alguns segundos, o que possibilitaria inclusão de mais estilos e exemplos sem maiores complicações. Ainda assim o modelo não é perfeito para certos tipos de imagem pode ser necessário ajustes no parâmetro resize enquanto para certas imagens em específico alguns dos estilos podem simplesmente não apresentar um resultado satisfatório.

Assim dentre algumas possíveis melhorias futuras no projeto estariam inclusas: a possibilidade de alterar a maneira que a paleta de cores da imagem de entrada interage com a paleta de cores do estilo permitindo ao usuário mais customização, tratamento de pixels transparente para permitir o uso de pngs e criação de máscaras para diferentes que permitam a aplicação de diferentes estilos a seções de uma mesma imagem. Tendo todas essas considerações em mente a ferramenta se mostrou bem capaz e promissora para futuros projetos que busquem aperfeiçoar o uso desses métodos.

## REFERENCES

- [1] G. Ghiasi, H. Lee, M. Kudlur, V. Dumoulin, and J. Shlens, “Exploring the structure of a real-time, arbitrary neural artistic stylization network,” 2017.
- [2] X. Zheng, T. Chalasani, K. Ghosal, S. Lutz, and A. Smolic, “Stada: Style transfer as data augmentation,” *arXiv preprint arXiv:1909.01056*, 2019.
- [3] M. H. Fischer, R. R. Yang, and M. S. Lam, “Imaginenet: Restyling apps using neural style transfer,” 2020.
- [4] A. Hertzmann, C. Jacobs, N. Oliver, B. Curless, and D. Salesin, “Image analogies,” *Proceedings of ACM SIGGRAPH*, vol. 2001, 06 2001.
- [5] A. A. Efros and W. T. Freeman, “Image quilting for texture synthesis and transfer,” in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH ’01. New York, NY, USA: Association for Computing Machinery, 2001, p. 341–346. [Online]. Available: <https://doi.org/10.1145/383259.383296>
- [6] R. Geirhos, D. H. J. Janssen, H. H. Schütt, J. Rauber, M. Bethge, and F. A. Wichmann, “Comparing deep neural networks against humans: object recognition when the signal gets weaker,” 2018.
- [7] L. A. Gatys, A. S. Ecker, and M. Bethge, “A neural algorithm of artistic style,” 2015.
- [8] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2015.
- [9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.
- [10] A. Mahendran and A. Vedaldi, “Visualizing deep convolutional neural networks using natural pre-images,” *International Journal of Computer Vision*, 2016.
- [11] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2017.
- [12] J. Johnson, A. Alahi, and L. Fei-Fei, “Perceptual losses for real-time style transfer and super-resolution,” 2016.
- [13] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky, “Texture networks: Feed-forward synthesis of textures and stylized images,” 2016.
- [14] S. B. J. H. P. P. D. R. P. D. . C. L. Z. Tsung-Yi Lin, Michael Maire, “Coco -common objects in context,” may 2014. [Online]. Available: <https://cocodataset.org/#home>
- [15] Itseez, “Open source computer vision library,” <https://github.com/Itseez/opencv>, 2015.