

Neural style transfer

Lucas Spartcus Vieira Carvalho¹, Thiago Henrique de Castro Oliveira¹

¹Pontifícia Universidade Católica de Minas Gerais (PUC MG)

Av: Dom José Gaspar, 500 Coração Eucarístico - Belo Horizonte - MG 30535-901, Brasil

Resumo. A área de processamento de imagem com sistemas neurológicos foi selecionada para discussão nesse artigo. É apresentado um projeto focado principalmente referente a subárea transposição de estilo artístico. A pesquisa apresenta a implementação de um código na linguagem Python que realiza a análise e aprendizado de um estilo artístico de uma imagem e transpõe tal estilo para outra imagem. O artigo também apresenta a análise no funcionamento desse método, testes e comparações com outros projetos.

1. Introdução

O artigo utiliza de AI(artificial intelligence) com rede neural para desenvolver um estudo clássico de processamento de imagem interligado com inteligência artificial. O problema proposto a ser resolvido/trabalhado pela pesquisa é a transposição de um estilo artístico para outra imagem e para obter tais resultados foi utilizado na análise Neural style transfer(NST) para trabalhar tal problema.

A transferência de estilo neural refere-se a uma classe de algoritmos de software que manipulam imagens digitais, ou vídeos, para adotar a aparência ou estilo visual de outra imagem. Seu funcionamento é por meio de uma técnica de otimização usada para pegar duas imagens - uma imagem de conteúdo e uma imagem de referência de estilo e misturá-las, precisamos copiar o estilo da imagem de estilo e aplicá-lo à imagem de conteúdo. para que a imagem de saída se pareça com a imagem de conteúdo, mas "pintada" no estilo da imagem de referência de estilo. Por estilo se entende basicamente os padrões, pinceladas, entre outros fatores da pintura original.

O modelo implementado realiza a transferência de estilo por meio de duas imagens que são setadas com tamanho especificado, (1024x1024) valor selecionado pelos autores. É utilizado o adam como otimizador para atualizar os pesos da rede de forma iterativa com base nos dados de treinamento. O algoritmo de otimização Adam é uma extensão da descida de gradiente estocástica que tem uma adoção mais ampla para aplicativos de aprendizado profundo em visão computacional e processamento de linguagem natural. A otimização usa como parâmetro a learning rate(taxa de aprendizagem) a qual pode ser variada para a obtenção de resultados previsto, nesse modelo a taxa é iniciada com valor 0,01.

Para classificar imagens foi usado o modelo VGG19 (255x255), em linguagem simples, VGG é uma CNN(Convolutional Neural Network) profunda usada para classificação de imagens. Em relação a extração do estilo da imagem é utilizado uma matriz de Gram. A matriz de Gram (de tamanho CxC) é o produto escalar de todos os vetores de recursos C (achatados) de um mapa de recursos convolucionais de profundidade C. Em relação a perda é adicionado o conteúdo e a perda de estilo antes da retropropagação para obter uma imagem de saída que tenha o conteúdo de uma imagem de conteúdo e o estilo

de uma imagem de estilo. Uma soma ponderada de perda de conteúdo e estilo é realizada para obter a perda final com estados diferentes.

2. Revisão de literatura

2.1. Diferente implementação

O artigo [Jing et al. 2020] fala de alguns projetos relacionados à transferência de estilo, entre eles a estilização artística. Antes do surgimento da NST, as pesquisas relacionadas expandiram-se para um área chamada de renderização não fotorrealista (NPR) tais algoritmos não usam CNNs. Renderização baseada em traços (SBR) refere-se a um processo de colocação de traços virtuais (por exemplo, pincel traços, ladrilhos, pontilhados) em uma tela digital para renderizar uma fotografia com um estilo particular. O processo de SBR geralmente começa a partir de uma foto de origem, incrementalmente composição de traços para combinar com a foto e, finalmente, produzir uma imagem não fotorrealista, que se parece com a foto mas com um estilo artístico. Durante este processo, a função é projetada para guiar a colocação gananciosa ou iterativa de traços. O objetivo dos algoritmos SBR é representar fielmente um estilo prescrito. Portanto, eles geralmente são eficazes para simular certos tipos de estilos (por exemplo, pinturas a óleo, aquarelas, esboços). No entanto, cada algoritmo SBR é cuidadosamente projetado para apenas um estilo particular e não é capaz de simular um estilo arbitrário, que não é flexível.

Em geral, as analogias de imagem são eficazes para uma variedade de estilos artísticos. No entanto, os pares de dados de treinamento são geralmente indisponíveis na prática. Outra limitação é que a imagem analogias exploram apenas recursos de imagem de baixo nível. Portanto, analogias de imagem normalmente não conseguem capturar conteúdo de forma eficaz e estilo, o que limita o desempenho. Desta forma tais tipos de algoritmo têm a limitações em flexibilidade, diversidade de estilo e extrações de estrutura de imagem eficazes.

2.2. Semelhante

Nesta pesquisa [Yanghao Li† 2017] é apresentado semelhanças em relação a nossa, tendo em vista que é demonstrado que a captura o estilo das imagens artísticas e trasferencia das mesmas para outras imagens usando Redes Neurais Convolucionais (CNN). Este trabalho formulou o problema como encontrar uma imagem que combinasse tanto as estatísticas de conteúdo quanto de estilo com base nas ativações neurais de cada camada na CNN. Os autores provaram teoricamente que combinando as matrizes Gram(fator que usamos para extração do estilo) do sistema neural ativações podem ser vistas como minimizando um valor máximo específico discrepância média (MMD). Isso revela que a transferência de estilo neural é intrinsecamente um processo de alinhamento da distribuição das ativações neurais entre as imagens.

Os autores afirmam que a informação de estilo na transferência de estilo neural é intrinsecamente representada pelas distribuições de ativações em uma CNN, e a transferência de estilo pode ser alcançada pelo alinhamento da distribuição. Além disso, é concluído pelos autores que a transferência de estilo neural é essencialmente um problema de adaptação de domínio especial, tanto teórica quanto empiricamente.

3. Metodologia

A metodologia aplicada no trabalho é implementada na linguagem Python. Realizamos no início a importação das bibliotecas que usaremos, sendo a principal delas o Keras, é uma biblioteca de rede neural de código aberto escrita em Python. Ele é capaz de rodar em cima de TensorFlow, projetado para permitir experimentação rápida com redes neurais profundas, ele se concentra em ser fácil de usar, modular e extensível. Importamos principalmente o VGG19 tal parâmetro é usado para classificar imagens, em linguagem simples, VGG é uma CNN(Convolutional Neural Network) profunda usada para classificação de imagens.

Em seguida realizamos a matriz GRAM, a matriz de Gram (de tamanho CxC) é o produto escalar de todos os vetores de recursos C (achatados) de um mapa de recursos convolucionais de profundidade C. Em seguida realizamos a importação de duas imagens sendo uma content e outra style ambas as imagens com 1024x1024 para que não haja desconexão entre elas.

Realizamos então o cálculo da perda que será usado no treinamento. Tal cálculo consiste de $\text{perda} = \text{contentWeight} * \text{contentLoss} + \text{styleWeight} * \text{styleLoss}$. Sendo styleLoss um valor resultante da subtração da saída com o estilo esperado elevado a 2, contentLoss um valor resultante da subtração da saída do conteúdo com o conteúdo esperado elevado a 2. Os valores contentWeight e styleWeight valem a 1e-2 ou 0.01.

No treinamento é usado uma fita de gradiente, passando como parâmetro para o gradiente a perda previamente calculada e a imagem. Então o gradiente é aplicado e recorta os valores do tensor para um mínimo(0.0) e um máximo(1.0) especificados. Caso épocas Mod 100 seja igual a 0 a gente printa a época e a perda.

4. Experimentos

Durante a implementação do projeto, foram realizados uma série de testes com o propósito de entender, ajustar e otimizar os parâmetros do modelo e seus impactos. Com isso, nosso objeto de estudo foi considerar diversas taxas de aprendizado(learning rate) em diversos otimizadores, bem como provocar diversas situações de underfitting e overfitting com as épocas.

Abaixo, serão mostrados alguns desses casos de teste, que iniciarão a discussão acerca dos resultados. Todos os testes foram realizados com a mesma imagem, em mesmo tamanho, e para o mesmo estilo.

4.0.1. Épocas: 4000/ Otimizador: Adam/ Learning rate: 0.01

Esse é o teste base, de onde se derivaram os outros, e o objeto base de comparação. Nessa configuração foi obtido o melhor resultado geral em várias situações.

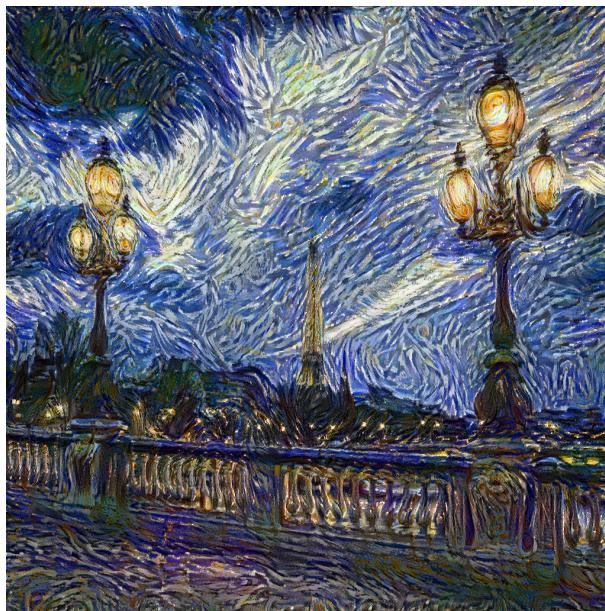


Figure 1. Epocas: 4000/ Otimizador: Adam/ Learning rate: 0.01

4.0.2. Epocas: 10000/ Otimizador: Adam/ Learning rate: 0.01

Esse é o teste para verificar um possível overfitting, Nele, o número de épocas é extrapolado, buscando que o output seja bem mais enviesado para o estilo artístico.



Figure 2. Epocas: 10000/ Otimizador: Adam/ Learning rate: 0.01

4.0.3. Epocas: 500/ Otimizador: Adam/ Learning rate: 0.01

Esse é o teste para verificar um possível underfitting, Nele, o número de épocas é bem baixo, buscando que o output seja bem menos adaptado para o estilo artístico.

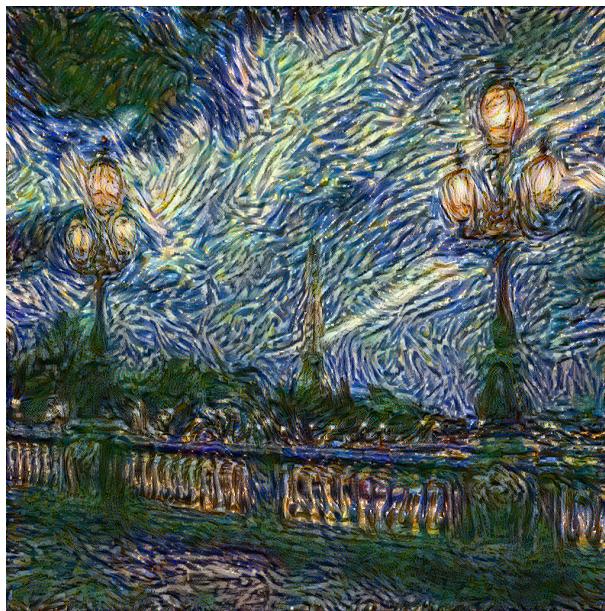


Figure 3. Epocas: 500/ Otimizador: Adam/ Learning rate: 0.01

4.0.4. Epocas: 4000/ Otimizador: Adamax/ Learning rate: 0.01

Esse teste busca verificar o impacto do uso de outros otimizadores na descida do gradiente e no resultado final da imagem.



Figure 4. Epocas: 4000/ Otimizador: Adamax/ Learning rate: 0.01

4.0.5. Epocas: 4000/ Otimizador: Adadelta/ Learning rate: 0.01

Esse teste busca verificar o impacto do uso de outros otimizadores na descida do gradiente e no resultado final da imagem.

4.0.6. Epocas: 500/ Otimizador: Adadelta/ Learning rate: 0.01

Esse teste busca verificar o impacto do uso de outros otimizadores na descida do gradiente e no resultado final da imagem.

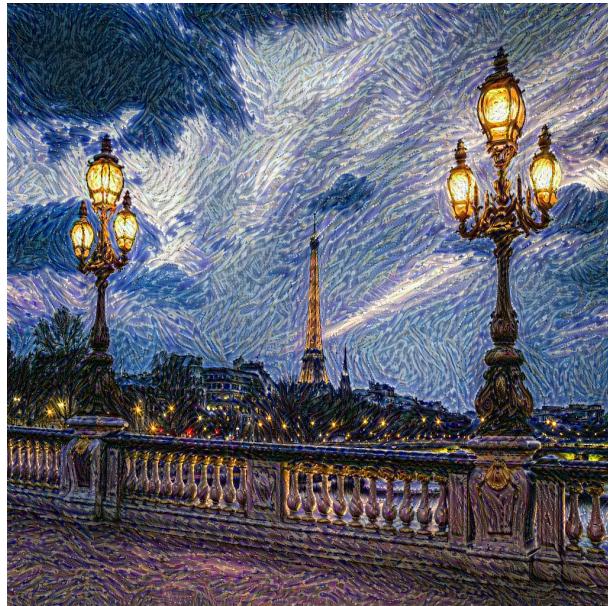


Figure 5. Epocas: 500/ Otimizador: Adadelta/ Learning rate: 0.01

4.0.7. Epocas: 4000/ Otimizador: SGD/ Learning rate: 0.01

Esse teste busca verificar o impacto do uso de outros otimizadores na descida do gradiente e no resultado final da imagem.

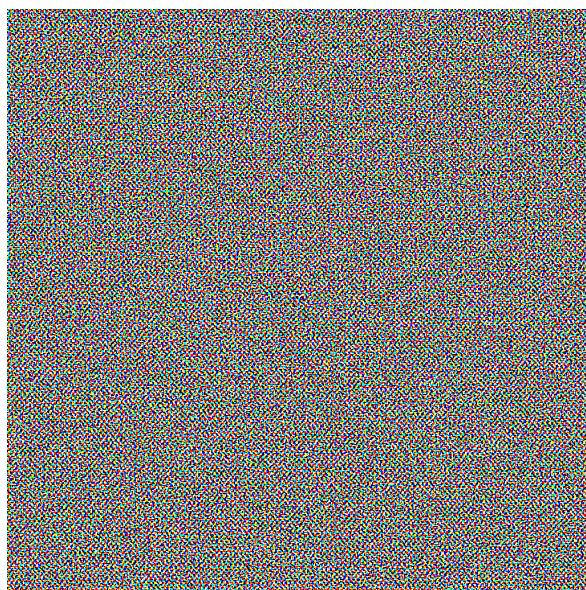


Figure 6. Epocas: 4000/ Otimizador: SGD/ Learning rate: 0.01

4.0.8. Epocas: 500/ Otimizador: SGD/ Learning rate: 0.01

Esse teste busca verificar o impacto do uso de outros otimizadores na descida do gradiente e no resultado final da imagem.

4.0.9. Epocas: 4000/ Otimizador: Adam/ Learning rate: 0.1

Teste de learning rate. Forçando uma descida mais brusca e verificando o que acontece com baixo e alto número de épocas.

4.0.10. Epocas: 500/ Otimizador: Adam/ Learning rate: 0.1

Teste de learning rate. Forçando uma descida mais brusca e verificando o que acontece com baixo e alto número de épocas.

4.0.11. Epocas: 500/ Otimizador: Adam/ Learning rate: 1

Teste de learning rate. Forçando uma descida mais brusca e verificando o que acontece com baixo e alto número de épocas.

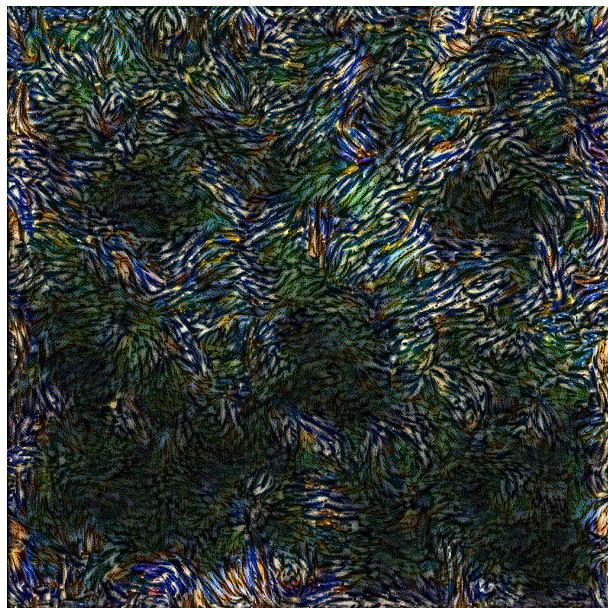


Figure 7. Epocas: 500/ Otimizador: Adam/ Learning rate: 1

5. Discussão

Com os testes feitos, conseguimos elaborar algumas análises e algumas ideias.

Começando pelos testes variando apenas o número de épocas do modelo base(4000 épocas, Adam, 0.01). Ao variar as épocas para 10.000, percebe-se que não houve um grande ganho em qualidade na imagem. Observando a descida do gradiente e

as losses durante essas épocas percebem-se duas coisas. Primeiramente, a loss começa a caminhar a passos cada vez mais curtos, já que cada vez mais nos aproximamos do mínimo local. Em contrapartida, observando o output final da imagem, percebe-se que a imagem final apresenta artefatos muito evidentes da imagem original, o que nos leva a conclusão de que apesar de conseguir alcançar uma loss menor, esse modelo se ajustou demais a imagem de estilo e, consequentemente, passa informações indesejadas a imagem final.

No segundo teste variando as épocas, foi feito justamente o oposto. Com 500 épocas se obtém um modelo muito pouco ajustado, e com uma loss ainda muito alta. Isso resulta em uma imagem pouco nítida, com ruído moderado e cores não condizentes com o estilo final. Esse modelo é simples demais e se mostrou insuficiente para replicar o estilo.

O segundo objeto de estudo dos experimentos foi ajustar os otimizadores e seus parâmetros, juntamente com o número de épocas. O primeiro escolhido foi o Adamax. Esse é um otimizador é uma variante do Adam e, de acordo com a literatura, lida melhor com volumes grandes e esparsos de dados. Ao utilizá-lo na mesma configuração do teste base, se obtém uma imagem nítida, mas que falhou em replicar com o máximo de fidelidade o estilo artístico. A imagem resultante ainda mantém mais características originais do que é esperado, o que nos mostra que o modelo convergiu, mas não desceu fundo o suficiente.

Foi testado também o otimizador Adadelta. Para esse otimizador foram feitos testes com 4000 e 500 épocas, para verificar como a descida do gradiente ocorre. Percebe-se que com 500 épocas, o optimizador apresenta um ótimo resultado, similar ao Adamax com 4000 épocas e superior em qualidade ao Adam com 500 épocas. Em contrapartida, com 4000 épocas a imagem resultante, apesar de replicar bem o estilo, apresenta muitos artefatos, com uma imagem final ruim. O Adadelta é uma extensão do Adagrad, e que suaviza a learning rate brusca na descida. Ainda assim, o Adam utiliza o momento e a taxa de aprendizado adaptativa, o que resulta numa descida bem mais consistente. Isso explica a eficiência com baixas iterações, mas que com o passar do tempo perde eficiência devido a descida um pouco mais agressiva.

Foi testado também o otimizador SGD, com 4000 e 500 épocas. Esse modelo nunca alcançou a convergência, e se mostrou ineficiente para resolver o problema.

Como último objeto de estudo, varia-se então a taxa de aprendizado. Com 4000 épocas e uma taxa de aprendizado de 0.1, a descida é feita a passos largos demais, fazendo com que se "pule" o mínimo. Esse resultado pode ser observado para qualquer número de épocas e para qualquer learning rate alta. Nos testes, esses números foram extrapolados para fim de visualização.

6. Conclusão

Após a conclusão dos testes e finalização da implementação, concluiu-se que o trabalho nos auxiliou na compreensão do funcionamento de aprendizado de máquinas. Demonstrando diversas formas para resolver os mesmos problemas, utilizando de diferentes técnicas, número de épocas ou taxa de aprendizado distintas.

Por fim, em nossa pesquisa conseguimos implementar o que havíamos proposto, sendo uma aplicação de transposição de um estilo artístico funcional. Conseguimos então

usar das redes neurais para transpor um estilo artístico de uma imagens para outra, de forma eficiente e com alta acurácia.

References

- Jing, Y., Yang, Y., Feng, Z., Ye, J., Yu, Y., and Song, M. (2020). Neural style transfer: A review. *IEEE Transactions on Visualization and Computer Graphics*, 26(11):3365–3385.
- Yanghao Li†, Naiyan Wang, J. L. X. H. (2017). Demystifying neural style transfer. *Institute of Computer Science and Technology, Peking University*.