

**FUNDAÇÃO GETULIO VARGAS
ESCOLA DE MATEMÁTICA APLICADA**

DANILO LEMOS CARDOSO

SÍNTESE DE TEXTURAS BASEADO EM AMOSTRA

Rio de Janeiro
2022

DANILO LEMOS CARDOSO

SÍNTESE DE TEXTURAS BASEADO EM AMOSTRA

Trabalho de conclusão de curso apresentada para a Escola de Matemática Aplicada (FGV/EMAp) como requisito para o grau de bacharel em Matemática Aplicada.

Área de estudo: processamento de imagens.

Orientador: Asla Medeiros e Sá

Rio de Janeiro

2022

Ficha catalográfica elaborada pela BMHS/FGV

Sobrenome, Nome

Síntese de Texturas Baseado em Amostra: subtítulo/ Danilo Lemos Cardoso. –
2022.

28f.

Trabalho de Conclusão de Curso – Escola de Matemática Aplicada.

Advisor: Asla Medeiros e Sá.

Includes bibliography.

1. Matemática 2. Aplicada 2. na matemática I. Medeiros e Sá, Asla II. Escola de
Matemática Aplicada III. Síntese de Texturas Baseado em Amostra

DANILO LEMOS CARDOSO

SÍNTESE DE TEXTURAS BASEADO EM AMOSTRA

Trabalho de conclusão de curso apresentada para a Escola de Matemática Aplicada (FGV/EMAp) como requisito para o grau de bacharel em Matemática Aplicada.

Área de estudo: processamento de imagens.

E aprovado em 07/12/2022
Pela comissão organizadora

Asla Medeiros e Sá
Escola de Matemática Aplicada

Jorge Luis Poco Medina
Escola de Matemática Aplicada

Emilio Vital Brazil
IBM

Resumo

Temas relacionados à análise e síntese de imagens vêm tendo um crescente aumento de interesse nos últimos anos, tanto por parte de pesquisadores, quanto pelo público geral. Isso se deve ao aumento do poder computacional e da geração de dados, que possibilitam o estudo e desenvolvimento de modelos mais ricos, permitindo assim aplicações como *inpainting* e transferência de estilo.

Neste trabalho será tratada a geração de texturas de tamanho arbitrário usando uma amostra limitada, tentando assim modelar seu processo de geração para produzir um resultado perceptualmente semelhante ao original. Será feita uma revisão da literatura sobre o tema, mostrando os principais resultados e abordagens, como a área foi se desenvolvendo até chegar no conhecimento de hoje, e como essa área influenciou em outros temas relacionados a imagem.

No final será mostrada uma implementação do método computacional, aplicando-o em diferentes tipos texturas para observar como características da imagem original podem mudar a qualidade do resultado. Em seguida serão exploradas variações do método que permitem um melhor controle da forma do resultado final.

Palavras-chave: síntese de textura, aprendizado profundo, transferência de estilo.

Abstract

Recent years have seen an increasing interest in topics related to the analysis and synthesis of images, both from researchers and the public. This is due to the constant increasing in computational power and data generation, which make the study and development of richer models possible, thus allowing applications such as inpainting and style transfer.

This dissertation will deal with the generation of textures of arbitrary size using a limited sample, thus trying to model its generation process and produce a result perceptually similar to the original. A review of the literature on the subject will be carried out, showing the main results and approaches, how the area has developed until it reaches today's knowledge, and how this area has influenced other topics related to image.

At the end, an implementation of the computational method will be shown, applying it to different types of textures to observe how characteristics of the original image can affect the result's quality. Next, variations of the method will be explored that allow more control over the shape of the final result.

Keywords: texture synthesis, deep learning, style transfer.

Sumário

1	INTRODUÇÃO	7
1.1	O que é textura	7
1.2	O que é síntese de textura	8
2	MODELOS	9
2.1	Modelos paramétricos	9
2.2	Modelos não paramétricos	11
2.3	Modelos de aprendizado profundo	12
3	APRENDIZADO PROFUNDO	14
3.1	Aprendizado de Representação	14
3.2	Redes neurais	14
3.3	Redes convolucionais	16
3.4	Síntese de textura com Redes Convolucionais	17
4	OUTRAS APLICAÇÕES	19
4.1	Analogias de Texturas	19
4.2	Transferência de estilo	20
5	RESULTADOS	22
6	CONCLUSÃO	26
	Referências	27

1 Introdução

Síntese de Textura é um tema clássico na área de processamento de imagens, tendo aplicações como “*inpainting*” (completar ou substituir elementos da imagem de forma realística), adaptação de textura para serem aplicadas em sólidos, e transferência de estilo. O foco do trabalho será nessa última aplicação, mostrando como a área foi se desenvolvendo nos últimos anos para termos os resultados de hoje.

Este trabalho foi baseado em dois *surveys* da área de Síntese de Textura. O primeiro, de Wei, Lefebvre, Kwatra e Turk ([WEI et al., 2009](#)), dá ênfase em abordagens não paramétricas e texturas dinâmicas. Já o segundo, de Raad, Davy, Desolneux e Morel ([RAAD et al., 2018](#)), mostra a síntese com métodos mais recentes usando *Deep Learning* e Redes Convolucionais. Analisando os dois *surveys*, fica claro o quanto a área de Síntese de Textura se modificou depois da revolução do *Deep Learning*, que se iniciou em 2015 com o trabalho de Yann LeCun, Yoshua Bengio e Geoffrey Hinton ([LECUN; BENGIO; HINTON, 2015](#)), e afetou diferentes áreas relacionadas a *Machine Learning*.

No final será mostrada uma implementação do método de síntese, comparando os resultados obtidos com o dos artigos citados e analisando os tempos gastos para gerar a imagem.

1.1 O que é textura

A palavra “textura” pode ter diferentes significados dependendo do contexto. Um deles se refere às diferentes características da superfície de um objeto, sejam elas visuais (cor, desenhos), geométricas (relevo, forma) ou táteis (maciez, dureza). Em computação gráfica, textura geralmente é o nome que se dá a uma imagem (matriz de pixels) que descreve alguma característica da superfície de um objeto, como a cor ou a direção do vetor normal (utilizada para iluminação).

Para o processo de síntese, é preciso restringir o conjunto de imagens consideradas texturas àquelas que apresentam algum tipo de padrão perceptual em seu domínio. Com isso é possível fazer a síntese estudando e imitando o processo que gerou esse padrão.

Cross, G.R. e Jain, A.K. ([CROSS; JAIN, 1983](#)) descrevem textura como sendo um Campo Markoviano Aleatório (*Markov Random Field*, MRF) bidimensional, tal que cada imagem é uma amostra deste campo (Figura 1). Esse modelo é o mais usado no processo de síntese, pois satisfaz a propriedade de Markov (o valor de cada pixel dado sua vizinhança não depende do resto da textura) e a homogeneidade (a distribuição é invariante por translação), logo se encaixa com as necessidades descritas anteriormente.

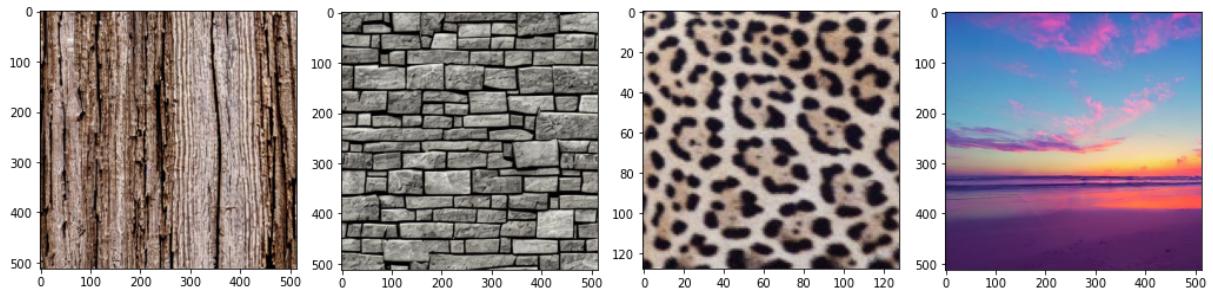


Figura 1 – As três primeiras imagens são texturas por apresentarem um padrão visual estacionário. A quarta imagem apresenta uma estrutura não estacionária, não permitindo uma extração infinita.

1.2 O que é síntese de textura

O processo de síntese de textura baseado em amostra não tem uma definição clara matematicamente, é algo mais intrínseco à percepção humana. O objetivo é, a partir de uma amostra de textura, gerar outras texturas de tamanho arbitrário que imitam o processo visual da amostra (Figura 2). Esse processo é baseado em métricas perceptuais, que não podem ser definidas de forma fechada, pois podem depender de aspectos finos da imagem, como forma e iluminação. Assim, os trabalhos na área nos últimos anos consistem em tentar descobrir melhores aproximações para essa métrica perceptual.

Ao restringir o conjunto de imagens aos MRF, o processo de síntese pode ser descrito como uma re-amostragem da distribuição condicional da amostra. Com isso, o desafio do método passa a ser descobrir a distribuição a partir da amostra.



Figura 2 – Exemplo de síntese de textura. A imagem maior é visualmente semelhante à imagem menor.

2 Modelos

Com o crescimento da área de processamento de imagens e do poder computacional, os modelos de Síntese de Textura foram se aprimorando ao longo do tempo. As melhorias são tanto de qualidade do resultado quanto no tempo de execução do método. Neste capítulo será apresentado um breve resumo de alguns dos principais modelos da área.

2.1 Modelos paramétricos

Os modelos que fazem a síntese a partir de um conjunto de estatísticas da amostra original são chamados modelos paramétricos. Esses modelos partem de um ruído e fazem a síntese reduzindo a diferença entre as estatísticas desse ruído e da amostra utilizando uma função de otimização. A qualidade do modelo vai depender do conjunto de estatísticas escolhido e do tipo de otimização utilizado.

Heegen e Bergen ([HEEGER; BERGEN, 1995](#)) foram um dos pioneiros no ramo. O método parte de um ruído branco e faz a síntese aproximando os histogramas das funções de distribuição acumulada. O método usa transformações por pirâmides, que decompõe a imagem em representações em diferentes escalas, e faz a aproximação do histograma em cada uma dessas camadas.



Figura 3 – Síntese de Textura de Heegen e Bergen ([HEEGER; BERGEN, 1995](#)). O modelo conseguia sintetizar elementos estocásticos da textura.

De Bonet ([DE BONET, 1997](#)) propôs um método que calcula a distribuição conjunta da amostra em diferentes escalas. Em seguida a amostragem é feita partindo de escalas mais grosseiras até escalas mais finas.

Zhu, Wu e Mumford ([ZHU; WU; MUMFORD, 1998](#)) utilizam um banco de filtros para obter uma representação da imagem e extrair o histograma. Em seguida deriva-se a distribuição desses filtros para serem re-amostrados usando a técnica de Gibbs Sampling.



Figura 4 – Modelo de De Bonet ([DE BONET, 1997](#)). O trabalho era feito escala por escala. Isso permitia sintetizar elementos mais regulares da textura.

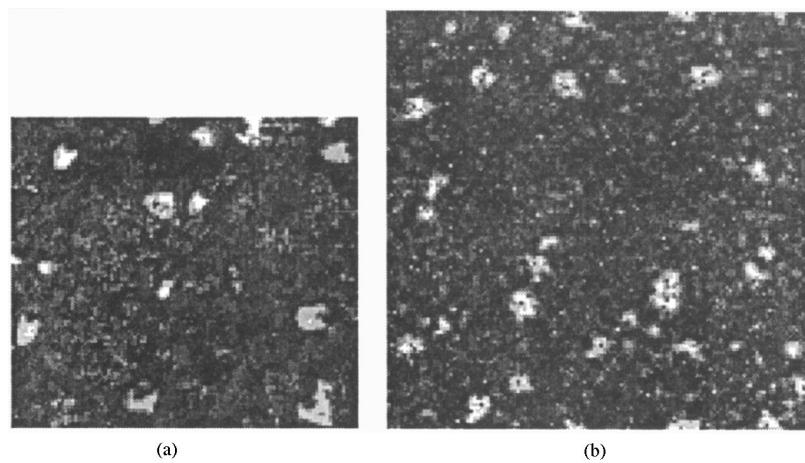


Figura 5 – Modelo de Zhu, Wu e Mumford ([ZHU; WU; MUMFORD, 1998](#)) usando 5 filtros.

Portilla e Simoncelli ([PORTILLA; SIMONCELLI, 1999](#)) propuseram fazer a amostragem usando as autocorrelações da Transformada de Wavelet da amostra.

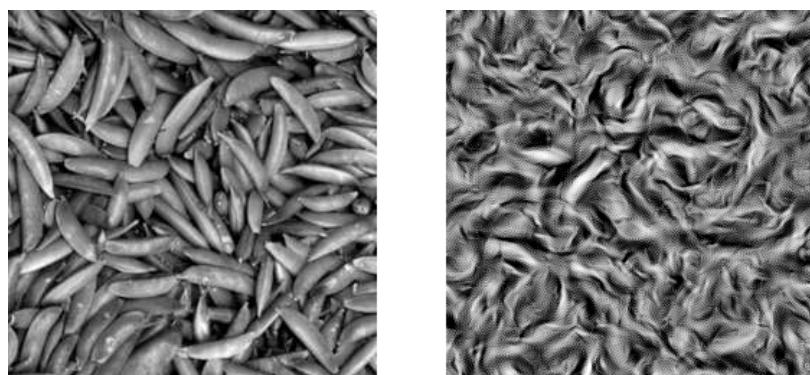


Figura 6 – Teste do modelo de Portilla e Simoncelli com uma textura mais estruturado ([PORTILLA; SIMONCELLI, 1999](#)). É possível notar que há uma dificuldade em representar tais estruturas.

Béla Julesz foi um neurocientista que trabalhava na percepção visual humana. Em uma de suas publicações ([JULESZ, 1981](#)), Julesz fez experimentos que mostravam que um conjunto de imagens com mesma informação visual podem ser representadas pela densidade do que ele chama de “Textons”, que são estruturas como cor e arestas presentes na imagem. Esse tipo de representação, embora tenha guiado o trabalho de modelos paramétricos, não deixa claro como fazer essa representação em Textons. Assim, na tentativa de encontrar tal representação, os métodos acabam sendo difíceis de implementar, e nem sempre conseguem ter sucesso em diferentes texturas.

2.2 Modelos não paramétricos

Diferentemente dos modelos paramétricos, os modelos não paramétricos não dependem do cálculo de estatísticas da amostra original para o processo de síntese. Ele gera a imagem pegando informação diretamente da amostra de modo a simular a amostragem da textura.

Essa forma de amostragem foi proposta inicialmente por Efros e Leung ([EFROS; LEUNG, 1999](#)). Ela consistia em re-amostrar a textura pixel por pixel, pegando diretamente da imagem original o pixel que tem a vizinhança mais parecida com a vizinhança de seu destino (Figura 7). Os resultados na época foram bem superiores aos que se podiam obter com os métodos paramétricos, mas a procura por todas as vizinhanças na amostra tornava o método lento.

Mais tarde, Efros e Freeman ([EFROS; FREEMAN, 2001](#)) propuseram o método de *Quilting* (costura), que fazia a amostragem a partir de pedaços da imagem original. O método consistia em selecionar janelas de tamanho fixo da amostra e distribuí-las com uma sobreposição sobre elas. Em seguida é usado um algoritmo de min-cut para dividir essas janelas em pedaços que se encaixam para formar a nova textura (Figura 8). Essa abordagem era mais rápida computacionalmente do que o método anterior, e produzia resultados tão bons quanto.

Com os avanços na área, Vivek Kwatra ([KWATRA et al., 2005](#)) propôs uma amostragem fazendo a minimização do que ele define como função de energia. Essa função é a diferença quadrática entre as vizinhanças da textura gerada e as vizinhanças mais próximas de cada uma (Figura 9). O método usa uma variação do algoritmo EM, onde na faze “E” a energia é diminuída por mínimos quadrados, e na faze “M” a energia é diminuída escolhendo as vizinhanças mais próximas na amostra.

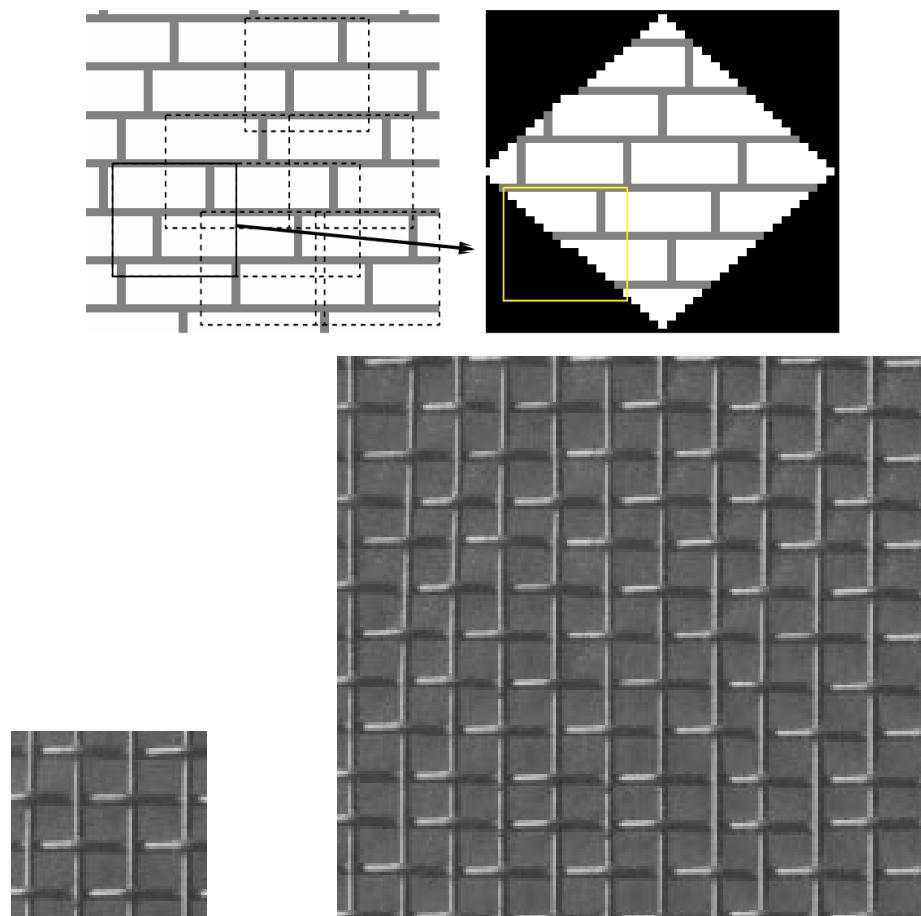


Figura 7 – Exemplo do modelo de Efros e Leung ([EFROS; LEUNG, 1999](#)). Ele vai escolhendo o pixel com vizinhança mais parecida na amostra.

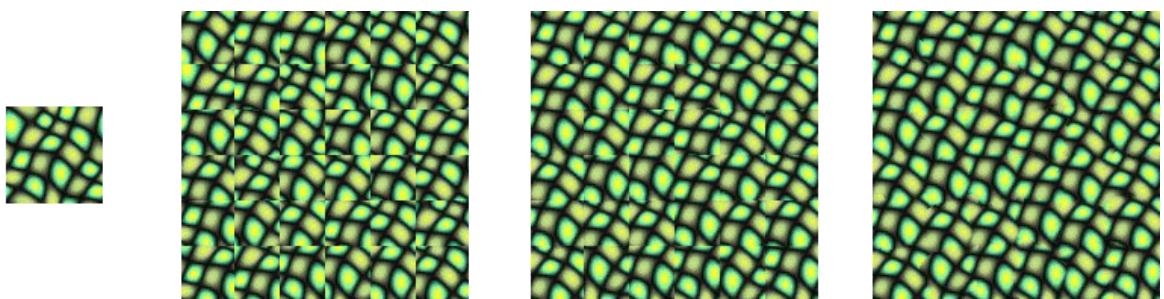


Figura 8 – Passos do modelo *Quilting* de Efros e Freeman ([EFROS; FREEMAN, 2001](#)). Os pedaços da amostra são escolhidos de forma aleatória e o corte é feito para encaixá-los.

2.3 Modelos de aprendizado profundo

Os modelos de aprendizado profundo se aproveitam dos avanços recentes na área de Aprendizagem de Máquina para melhorar o processo de síntese. Os métodos a seguir utilizam as representações geradas por redes treinadas para a detecção de objetos para melhorar métodos paramétricos de síntese já existentes. As representações geradas por essas redes conseguem sintetizar bem a informação visual da imagem, oferecendo uma boa métrica perceptual ([ZHANG; ISOLA et al., 2018](#)).

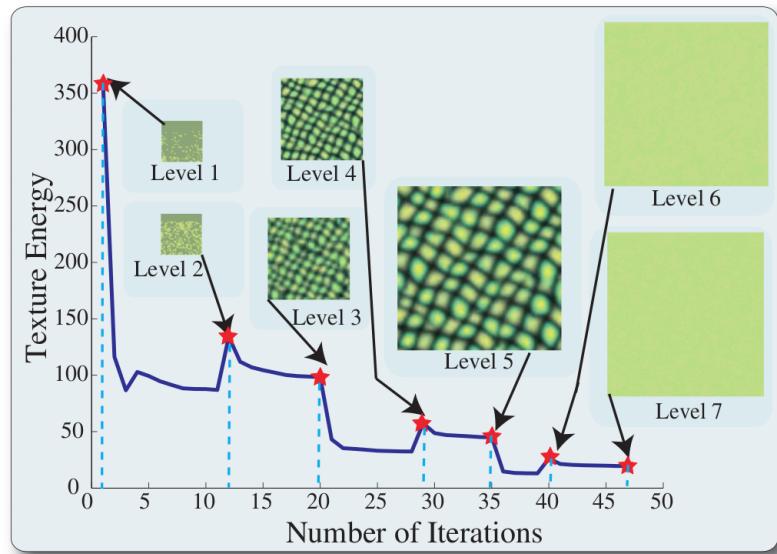


Figura 9 – Método utilizado no modelo de Vivek Kwatra ([KWATRA et al., 2005](#)). A função de energia cai a cada nível de resolução.

O primeiro a propor Síntese de Textura baseado nesses modelos foram Gatys, Ecker e Bethge ([GATYS, L.; ECKER; BETHGE, 2015](#)). O trabalho utiliza as autocorrelações da textura semelhante ao que foi proposto por Portilla e Simoncelli ([PORTILLA; SIMONCELLI, 1999](#)), porém, em vez de utilizar transformações da amostra escolhidas a mão, utilizam a saída da rede convolucional pré-treinada. Esse método será tratado com detalhes nesse trabalho, seguindo de uma implementação.



Figura 10 – Exemplo do modelo de Gatys, Ecker e Bethge ([GATYS, L.; ECKER; BETHGE, 2015](#)), que utiliza as matrizes de Gram das convoluções. É possível ver que o método consegue extrapolar bem a estrutura da amostra, produzindo assim resultados mais “criativos”.

Lu, Zhu e Wu ([LU; ZHU; WU, 2016](#)) utilizam as representações das redes convolucionais no lugar do banco de filtros no método de Zhu, Wu e Mumford ([ZHU; WU; MUMFORD, 1998](#)), produzindo melhores resultados sem se preocupar com o tipo de filtros escolhidos.

3 Aprendizado Profundo

Um dos clássicos problemas na área de classificação de imagens era o reconhecimento de dígitos escritos à mão. A dificuldade nesse tipo de problema vinha pela grande variação de forma e posição dos números, tornando os modelos muito complexos e pouco precisos. O desenvolvimento recente na área de *Machine Learning* permitiu a criação de modelos simples capazes de aprender com os dados e fazer generalizações precisas, em alguns casos superando a percepção humana.

Neste capítulo serão apresentadas técnicas que usam Redes Neurais, como elas foram desenvolvendo ao longo dos anos, e como podemos aplicá-las na Síntese de Textura.

3.1 Aprendizado de Representação

No processo de análise de imagens, precisamos criar representações que melhor explicam seu conteúdo. Essas representações podem ser organizadas hierarquicamente, onde a primeira camada pode identificar estruturas como arestas, a segunda indica cantos, etc. Ao descer no nível das representações queremos ter uma melhor informação semântica da imagem.

Para um problema de classificação de imagens, é preciso saber quais informações são úteis para obter uma melhor representação de seu conteúdo. Uma representação boa esconde informações redundantes e realça fatores que melhor explicam a imagem. Bengio, Courville e Vincent ([BENGIO; COURVILLE; VINCENT, 2013](#)) falam do importante papel que a aprendizagem de representações tem em Machine Learning. Um algoritmo que aprendesse tais representações eliminaria o trabalho de desenvolvê-las, o que tornaria mais rápido a criação de aplicações.

3.2 Redes neurais

Redes Neurais são um tipo de modelo de *Machine Learning* inspirado em redes de neurônios naturais e como eles se comunicam. Elas podem ser representadas por um grafo, onde cada nó representa um valor e cada aresta representa uma dependência. Essas redes podem ter vários formatos dependendo do tipo de aplicação.

Um exemplo comum de Rede Neural é o *Multi Layer Perceptron*, ou MLP ([ZHANG; WOODLAND, 2015](#)). Nesse modelo, a rede é dividida em camadas de diferentes tamanhos, e o dado flui da camada de entrada até a saída, que pode representar uma classificação da entrada ou uma função geral (Figura 11).

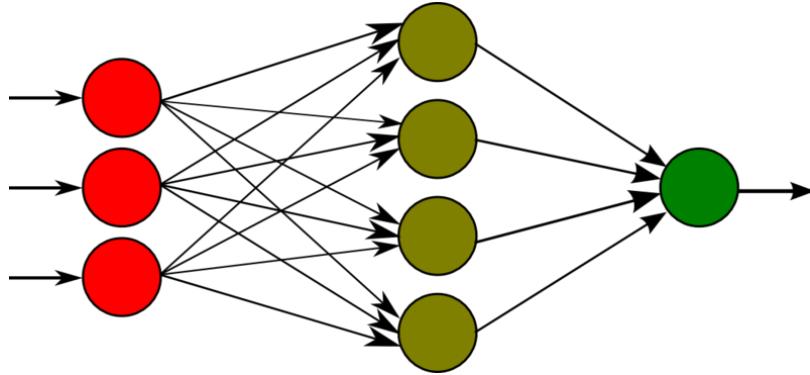


Figura 11 – Rede MLP. Cada camada da rede atua como uma representação do dado de entrada, que é usada para calcular a próxima camada. (<https://commons.wikimedia.org/wiki/File:MultiLayerPerceptron.svg> . Acesso: 15/12/2022.)

Cada camada da rede tem como parâmetros os offsets \mathbf{b}_m e uma matriz de pesos $\mathbf{W}_{m \times n}$. Assim, a transformação da entrada \mathbf{x}_n na saída \mathbf{y}_m pode ser representada com a seguinte operação:

$$\mathbf{y} = \varphi(\mathbf{W}\mathbf{x} + \mathbf{b}), \quad (3.1)$$

onde φ é uma função não linear, comumente chamada de “função de ativação”. A função não linear mais comumente usada é a ReLU, que trunca valores negativos no 0.

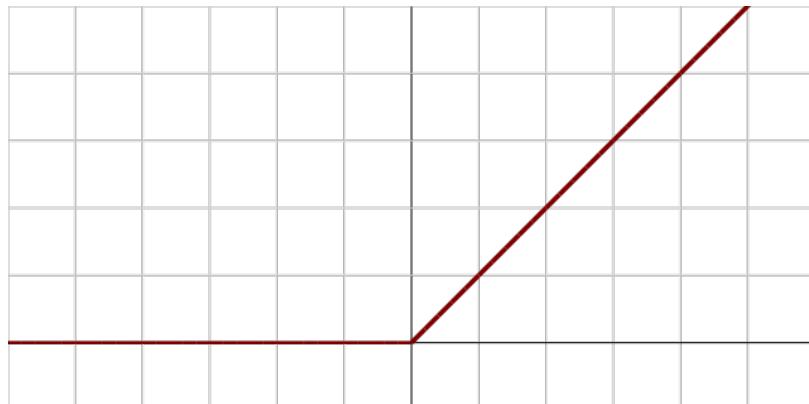


Figura 12 – Função ReLU. O seu bico não diferenciável facilita a distorcer o espaço de entrada.

Os parâmetros são calculados minimizando uma função de erro na saída. Essa minimização é feita usando o gradiente dos parâmetros em relação à função de perda. Como a derivada de todas as funções aplicadas é conhecida, o gradiente pode ser calculado computacionalmente utilizando *backpropagation*.

Esse modelo é útil por envolver operações bem simples e rápidas de serem calculadas computacionalmente. Cada camada distorce o espaço de entrada para torná-lo linearmente separável no final. Pela sua flexibilidade, o modelo é geralmente chamado de approximator universal de qualquer função, pois é capaz de aprender funções complexas a partir de amostras.

3.3 Redes convolucionais

A rede MLP, quando aplicada diretamente aos pixels de uma imagem concatenando-os linha por linha, aparecem dois problemas consideráveis: a quantidade de parâmetros cresce muito, e informações espaciais da imagem são perdidas. Geralmente, um objeto que queremos identificar pode aparecer em diferentes posições da imagem, e ao transformá-la em um vetor, perdemos essa invariância por translação. Com base nisso se tornou necessário desenvolver uma rede que aproveite a informação espacial da imagem e que tenha poucos parâmetros.

A convolução é um filtro linear muito utilizado em processamento de imagens. Para transformar a imagem, ela recebe uma matriz chamada *kernel*, e faz a convolução somando a vizinhança de cada ponto com os pesos definidos pelo *kernel*. Dependendo da matriz escolhida, vários filtros podem ser gerados, como detector de arestas, suavizador, etc.

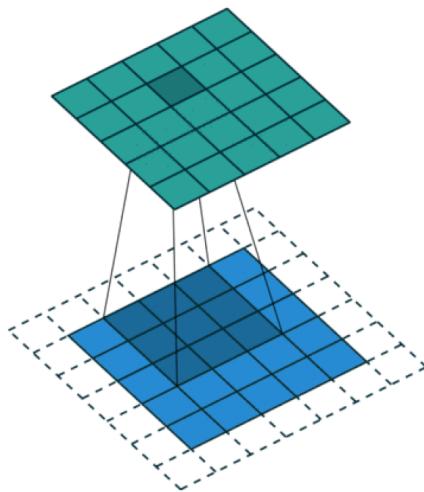


Figura 13 – Exemplo de como a convolução opera em uma imagem. Em azul a origem e em verde o destino. O *kernel* nesse caso é uma matriz 3×3 . Para manter a resolução da imagem, ela precisa ser extrapolada nas vizinhanças das bordas; dependendo da aplicação ela pode ser preenchida com 0s ou com o valor na borda. (Fonte: https://commons.wikimedia.org/wiki/File:Convolution_arithmetic_-_Padding_strides.gif. Acesso: 15/12/2022, adaptado.)

Uma rede neural convolucional é formada por um conjunto de convoluções que geram representações da entrada, tendo o *kernel* como parâmetro a ser aprendido. A rede empilha essas convoluções seguidas de aplicações de funções ReLUs para tirar a linearidade, e camadas de *pooling*, que diminuem o tamanho da imagem juntando pixels vizinhos através de operações como a média ou o máximo.

Uma imagem na entrada da rede tem dimensão $W \times H \times 3$, onde a última representa os três canais de cores. As convoluções operam nos canais de cores e podem gerar novos canais, geralmente chamados de *features*, cada uma representando alguma informação da imagem.

Esse tipo de rede gerou uma revolução no ramo de classificação de imagens. A classificação era feita passando por algumas camadas de *pooling* e ligando a saída em uma rede MLP.

Um exemplo de arquitetura que obteve sucesso em classificação de imagens foi a rede VGG-19 ([SIMONYAN; ZISSERMAN, 2014](#)). Essa rede tem camadas de convolução e ReLU, e a cada *pooling* dobra o número de *features*, indicando que quanto mais abaixo a informação está na rede, mais representações ela pode ter.

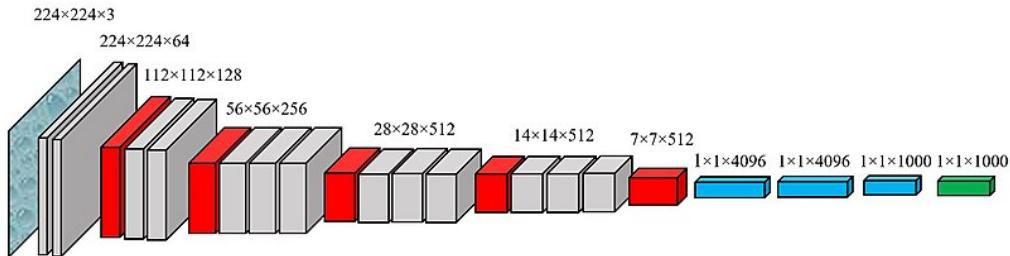


Figura 14 – Arquitetura da rede VGG-19. Cada bloco cinza representa uma convolução e um ReLU, os blocos vermelhos representam *pooling*, os blocos azuis são camadas da rede MLP e em verde é a saída da classificação (ela é treinada para diferenciar 1000 classes). (https://en.wikipedia.org/wiki/File:VGG_neural_network.png. Acesso: 15/12/2022.)

3.4 Síntese de textura com Redes Convolucionais

As redes convolucionais treinadas para classificação de imagem (em particular a rede VGG-19) fazem uma série de operações em pirâmide para extrair informação útil da imagem que se quer classificar. A riqueza semântica de cada uma dessas informações vai depender do nível da camada, com as primeiras camadas salientando detalhes mais primitivos, como arestas e cantos, e as últimas camadas salientando objetos complexos como rostos e animais.

Para a criação do método de síntese de textura utilizando essas redes, a equipe de Gatys percebeu que mantendo as relações entre as camadas, a textura pode ser randomizada sem que a informação perceptual da imagem se perca. Para isso, eles utilizaram a matriz de Gram gerada por cada uma dessas camadas. A matriz tem tamanho $N_l \times N_l$, onde N_l é o número de *features* da camada l da rede. A saída de cada camada convolucional tem dimensão $W_l \times H_l \times N_l$; para calcular a matriz de Gram definimos F_l como sendo uma planificação linha a linha da saída da camada, ou seja, F_l tem dimensão $N_l \times (W_l H_l)$ (Figura 15). Assim, $G_l = F_l F_l^T$. Essa matriz é útil pois não depende da resolução da imagem que entrou na rede, assim pode ser usada para gerar imagens em qualquer resolução.

A geração da textura é feita através de algoritmo de otimização baseado em gradiente. Para isso comparamos as matrizes de Gram da amostra e da imagem sendo gerada para calcular o gradiente com respeito a perda quadrática. Em seguida, a imagem

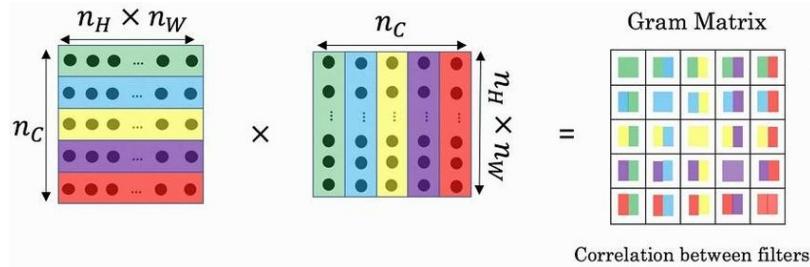
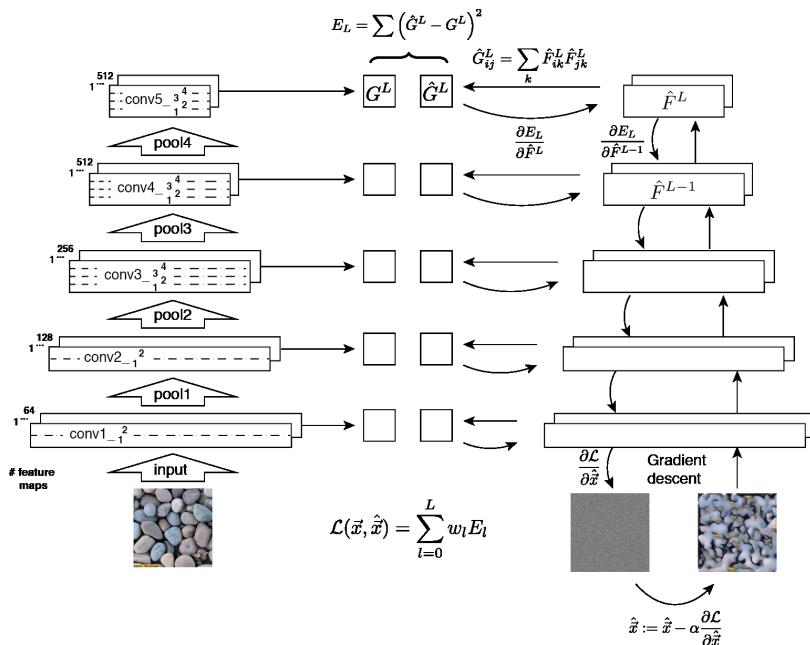


Figura 15 – Esquema da matriz de Gram. Ela representa a correlação entre as *features* das camadas convolucionais. (Fonte: https://www.researchgate.net/figure/The-Gram-Matrix-is-created-from-a-target-image-and-a-reference-image_fig4_356667127. Acessado: 15/12/2022.)

é atualizada para diminuir essa perda, forçando-a a ter as mesmas autocorrelações entre as *features*.



4 Outras aplicações

O trabalho de síntese de textura não fica limitado apenas a re-amostrar a imagem de entrada. Ao longo dos anos foram aparecendo trabalhos que aproveitavam os algoritmos desenvolvidos até então para criar aplicações como *inpainting* e renderizações não foto-realísticas.

Neste capítulo será falado de duas variações bem interessantes de síntese de textura, que nos permitem ter mais controle do conteúdo no resultado.

4.1 Analogias de Texturas

O método de analogias de texturas foi proposto no trabalho de Hertzmann, Jacobs, Oliver, Curless e Salesin ([HERTZMANN et al., 2001](#)). O algoritmo consiste em fazer analogias de imagens, ou seja, ele recebia três imagens, A , A' e B , e tentava gerar B' de modo que A' esteja para A assim como B' estará para B .



Figura 17 – Exemplo de analogia de imagens. As relações entre A' e A devem ser aplicadas em B para gerar B' . Fonte: ([HERTZMANN et al., 2001](#)).

O algorítimo faz a síntese utilizando a representação multi-escala das imagens, gerando da camada mais grosseira até a camada mais fina. A geração é feita pixel por pixel, e leva em conta a sua vizinhança em B e vizinhança de seu correspondente em A , como mostra a Figura 18.

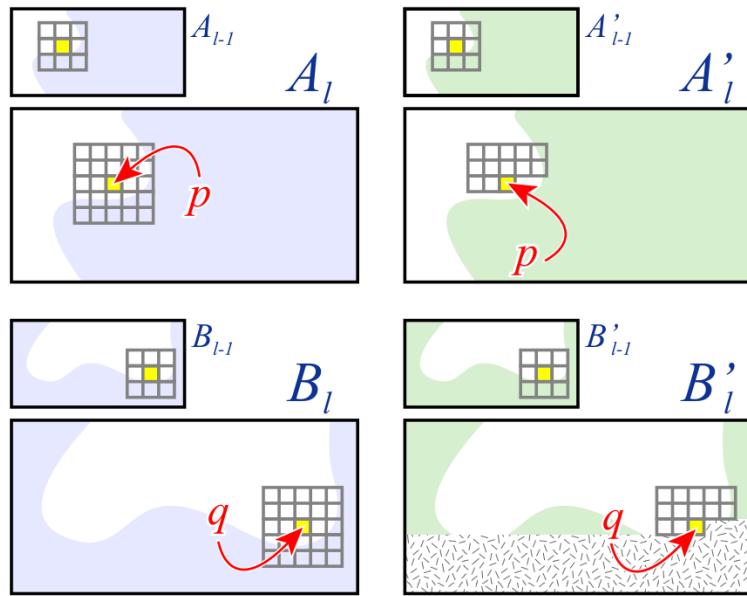


Figura 18 – Para gerar o pixel q da camada l , o algoritmo usa a vizinhança correspondente em B_l , B'_{l-1} e B_{l-1} . Ele busca o melhor correspondente nas imagens A . Fonte: (HERTZMANN et al., 2001).

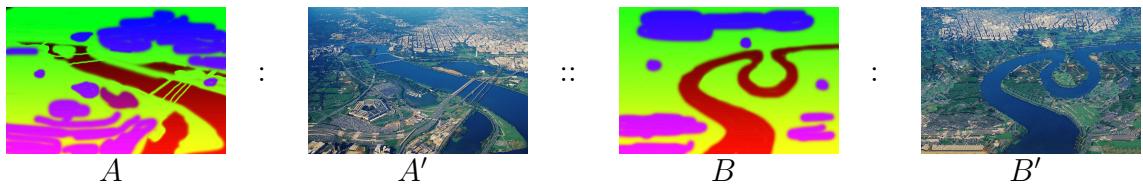


Figura 19 – Outro exemplo de analogia de textura. Este exemplo mostra que o método também pode receber um descritor de uma imagem para gerar outra. Fonte: (HERTZMANN et al., 2001).

4.2 Transferência de estilo

O método de *Style Transfer* proposto por Gatys, Ecker e Bethge (GATYS, L. A.; ECKER; BETHGE, 2016) permite, além de re-amostrar a textura, controlar a geometria do resultado. Para isso o algoritmo recebe duas imagens, uma definirá o estilo e outra definirá o conteúdo do resultado.

Para implementar o método, é preciso gerar as representações da imagem de conteúdo na rede convolucional. Então, basta adicionar à função de perda a diferença entre a representação da imagem gerada e do conteúdo. Assim vão existir duas perdas, $\mathcal{L}_{content}$ que representa a perda do conteúdo com as representações da rede, e \mathcal{L}_{style} , que representa a perda com as matrizes de Gram.

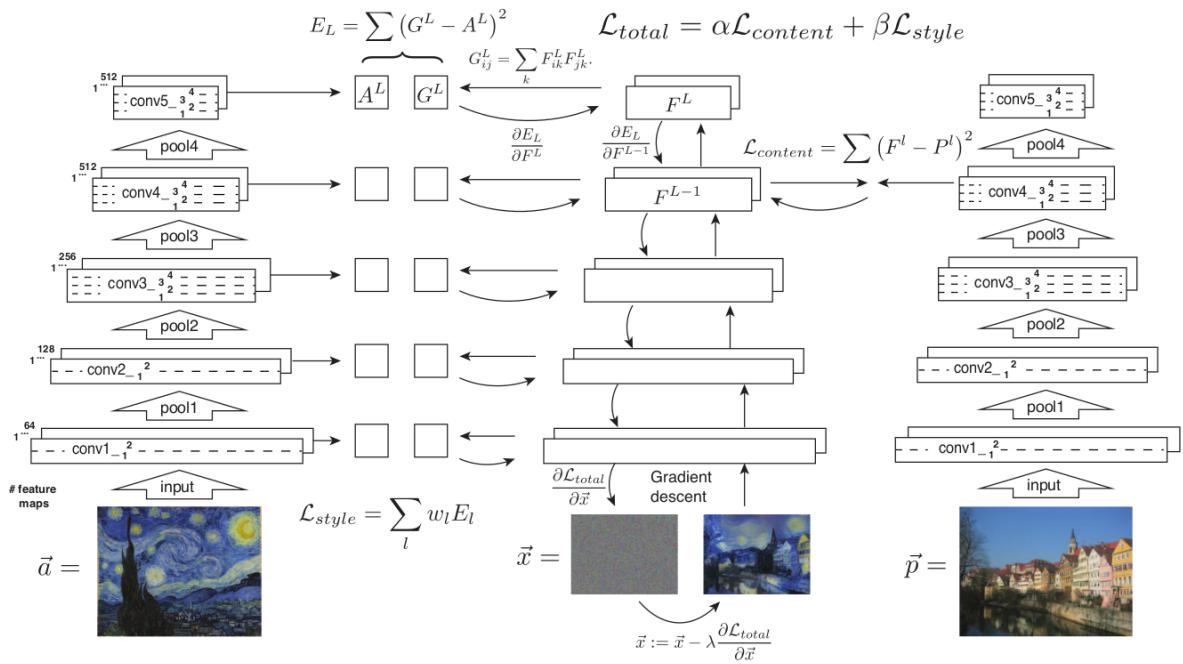


Figura 20 – Método da transferência de estilo. Faz a otimização das matrizes de Gram e das *features* ao mesmo tempo. Fonte: ([GATYS, L. A.; ECKER; BETHGE, 2016](#))

5 Resultados

Foi feita a implementação do método de Síntese de Textura usando a abordagem das matrizes de Gram. Para isso foi utilizada a linguagem Python com a biblioteca PyTorch. O PyTorch reúne uma série de funções que facilitaram a implementação do método. As principais foram: implementação nativa da rede VGG-19, operações aritméticas aceleradas na GPU, cálculo automático de gradiente, e implementação do método de otimização L-BFGS. O código utilizado, bem como as texturas, estão disponíveis em <https://github.com/danilolc/texture-synthesis>. O arquivo principal é o *texture_synth.py*, que pode ser modificado para executar as diferentes implementações do trabalho.

A primeira ideia era escolher um conjunto de textura de resolução 128×128 e tentar gerar imagens de 256×256 pixels. Foram escolhidos um conjunto de diferentes tipos de texturas para explorar o comportamento do algoritmo.

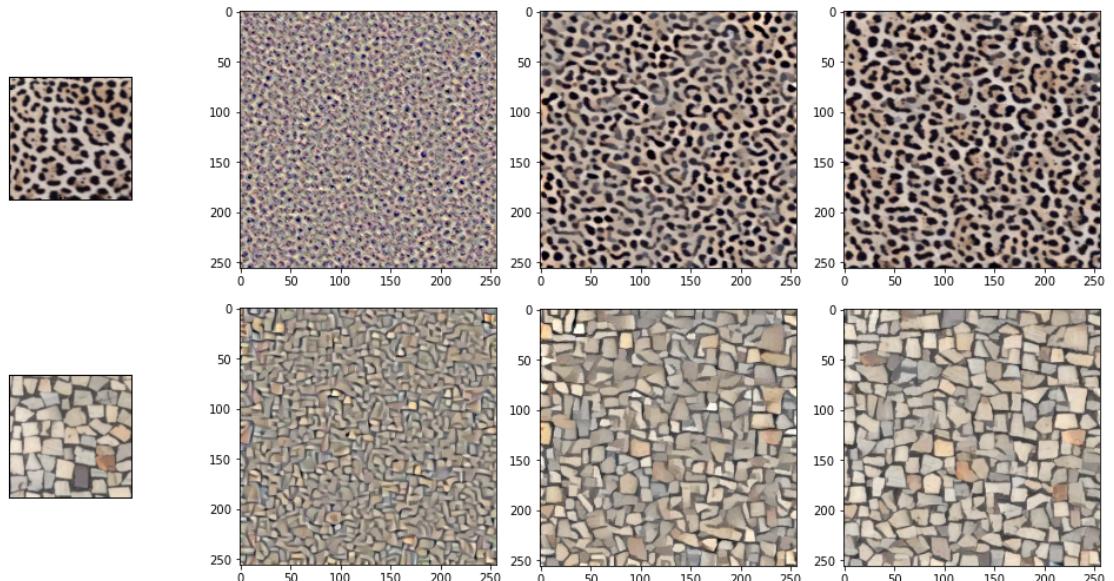


Figura 21 – Teste com texturas mais caóticas. A primeira imagem é a amostra, e as seguintes são sínteses a medida que o número de iterações cresce. No processo, a escala dos objetos se mantém, portanto, o resultado terá quatro vezes mais informação.

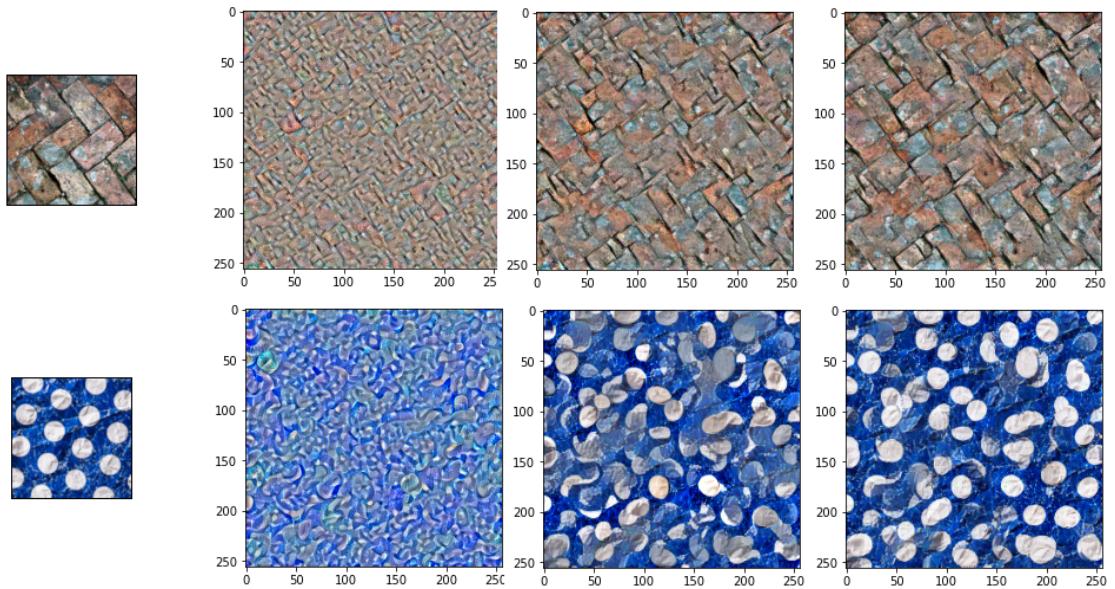


Figura 22 – O método não funciona bem em textura com estrutura regular global, ele dá preferência por estruturas locais.

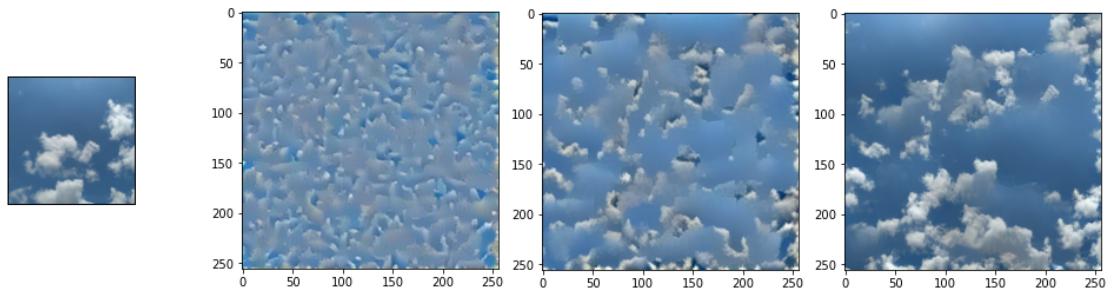


Figura 23 – Nuvens apresentam uma estrutura caótica ideal para esse método.

As iterações foram feitas até que não fosse possível perceber diferenças na imagem entre os passos de iteração. Todos os testes feitos convergiram para algum valor, que nem sempre mostrava a textura de forma ideal (como pode ser visto na Figura 22). Com isso foi pensado em adicionar ruído na imagem em intervalos fixos de iterações para forçar a mudança do ponto de convergência. Isso gerou um resultado interessante ao movimentar suavemente a textura a cada aplicação do ruído, como mostra a Figura 24.

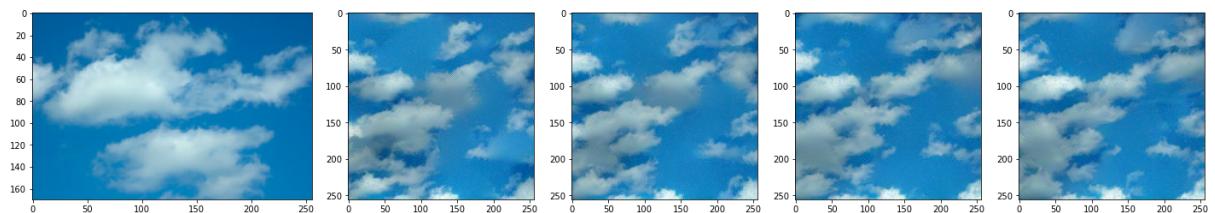


Figura 24 – A terceira imagem foi gerada a partir da adição de ruído na segunda, assim como da terceira para quarta e da quarta para quinta. Com isso, a sequência de imagens se comporta de um modo semelhante ao movimento de nuvens.

Não é preciso se limitar a texturas com esse tipo de implementação. É possível verificar o que acontece no resultado com imagens não textura, como feito na Figura 25. O resultado do método nessas imagens preserva cores e pequenos objetos, mas não reproduzem a informação espacial da imagem.

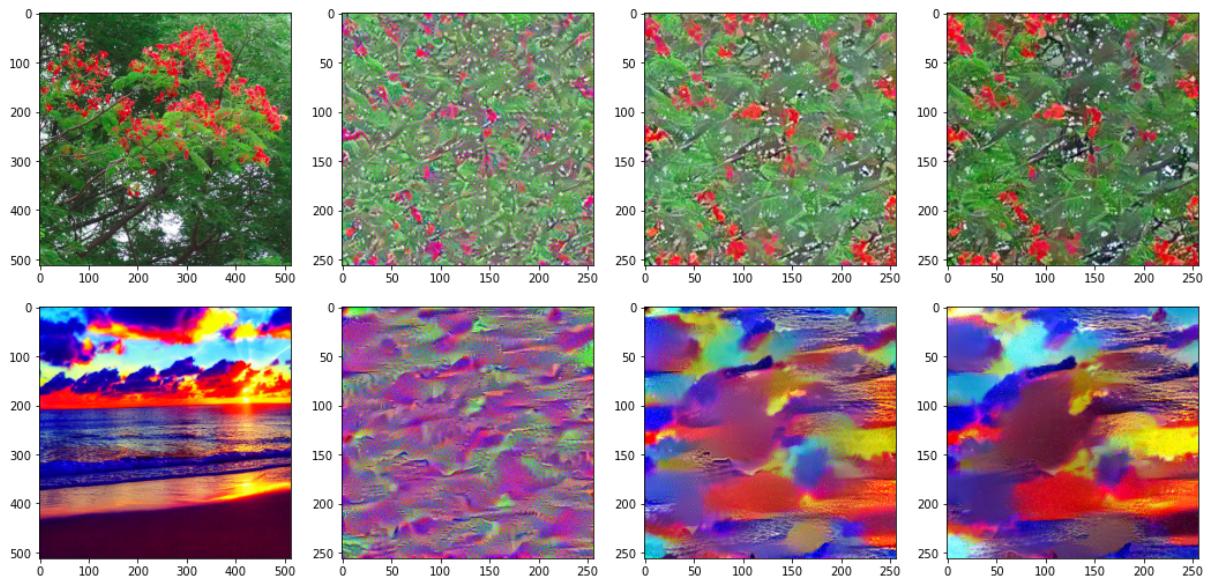


Figura 25 – Teste de Síntese de Textura com imagem não textura.

Todos os testes foram executados em um notebook Dell G3 3590, com um processador Intel Core i5-9300H em 2,4GHz e uma placa de vídeo (GPU) NVIDIA GeForce GTX 1050 de 3 GB de memória. Importar o modelo VGG-19 (depois de baixado) e calcular as matrizes de Gram da amostra é feito quase instantaneamente. É possível escolher entre rodar o modelo no processador (CPU) ou na placa de vídeo (CUDA). No segundo o método converge em poucos segundos, como mostra a Tabela 1.

Tabela 1 – Tempo de execução para gerar a imagem da Figura 23 (uma imagem de resolução 256x256). O resultado convergiu a partir de 18 iterações do L-BFGS.

Dispositivo	Tempo total	Tempo por iteração
CPU	267s	14,8s/it
CUDA	40s	2,2s/it

Não existe uma forma fechada de comparar a qualidade do resultado, assim a melhor maneira de verificar se o método funciona bem para a textura dada é comparando os resultados visualmente e julgando-os com base na percepção.

Uma vez implementada a síntese de textura, implementar a transferência de estilo passa a ser trivial, apenas tendo que adicionar a diferença das *features* na função de perda. Os resultados deste método são mostrados nas figuras a seguir.



Figura 26 – Resultado de transferência de estilo, a primeira imagem é usada como estilo e a segunda como conteúdo. O resultado depende da escala das duas imagens.

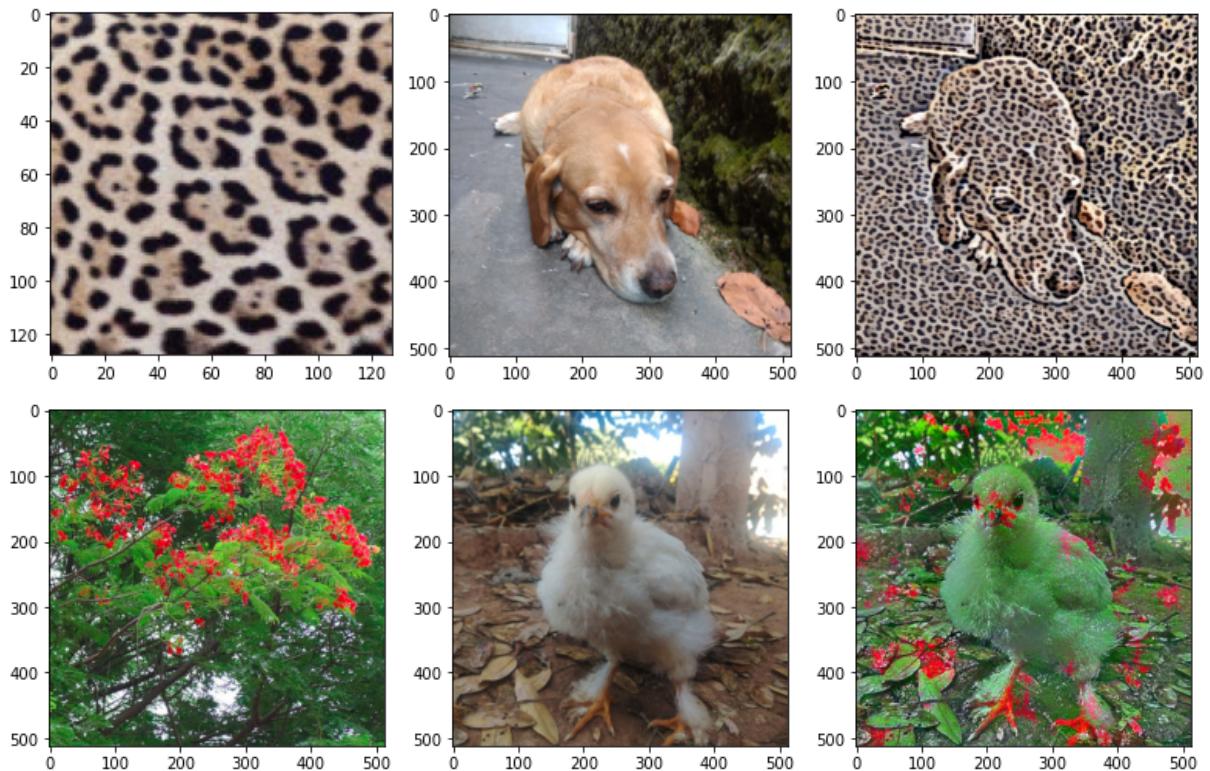


Figura 27 – Mais resultados de transferência de estilo. Na última é possível ver que o método não fica restrito a texturas.

6 Conclusão

Esse trabalho mostrou o quanto pode ser difícil a tarefa de processamento e síntese de texturas e imagens no geral, e o quanto foi preciso andar para chegar nos avanços que existem hoje. Uma grande quantidade de trabalhos são publicados todos os anos na área, cada um tentando encontrar uma maneira nova de melhorar a solução do problema.

Redes Neurais e *Deep Learning* vêm se mostrando ferramentas bem poderosas para o trabalho com imagem. A diferença entre as camadas de Redes Convolucionais pré-treinadas para a detecção de objetos podem oferecer uma excelente métrica perceptual, abrindo caminho para diversos novos trabalhos. O aprendizado automático de representações facilita o trabalho de criar aplicações de processamento e síntese de imagens, mas a disponibilidade de dados e de processamento ainda pode ser um problema.

Ferramentas para trabalhar com *Machine Learning* como o PyTorch vêm se mostrando mais poderosas a cada dia, oferecendo facilidades para trabalhar com grandes quantidades de dados, além de fácil acesso à GPU, que melhora bastante a velocidade de operações aritméticas. Essas ferramentas também contam com sistemas de cálculo automático de gradiente, tornando o trabalho de otimização mais fácil e menos suscetível a erros.

Referências

- BENGIO, Yoshua; COURVILLE, Aaron; VINCENT, Pascal. Representation learning: A review and new perspectives. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 35, n. 8, p. 1798–1828, 2013.
- CROSS, George R; JAIN, Anil K. Markov random field texture models. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE, n. 1, p. 25–39, 1983.
- DE BONET, Jeremy S. Multiresolution sampling procedure for analysis and synthesis of texture images. In: PROCEEDINGS of the 24th annual conference on Computer graphics and interactive techniques. [S.l.: s.n.], 1997. P. 361–368.
- EFROS, Alexei A; FREEMAN, William T. Image quilting for texture synthesis and transfer. In: PROCEEDINGS of the 28th annual conference on Computer graphics and interactive techniques. [S.l.: s.n.], 2001. P. 341–346.
- EFROS, Alexei A; LEUNG, Thomas K. Texture synthesis by non-parametric sampling. In: IEEE. PROCEEDINGS of the seventh IEEE international conference on computer vision. [S.l.: s.n.], 1999. v. 2, p. 1033–1038.
- GATYS, Leon; ECKER, Alexander S; BETHGE, Matthias. Texture synthesis using convolutional neural networks. **Advances in neural information processing systems**, v. 28, 2015.
- GATYS, Leon A; ECKER, Alexander S; BETHGE, Matthias. Image style transfer using convolutional neural networks. In: PROCEEDINGS of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2016. P. 2414–2423.
- HEEGER, David J; BERGEN, James R. Pyramid-based texture analysis/synthesis. In: PROCEEDINGS of the 22nd annual conference on Computer graphics and interactive techniques. [S.l.: s.n.], 1995. P. 229–238.
- HERTZMANN, Aaron et al. Image analogies. In: PROCEEDINGS of the 28th annual conference on Computer graphics and interactive techniques. [S.l.: s.n.], 2001. P. 327–340.
- JULESZ, Bela. A theory of preattentive texture discrimination based on first-order statistics of textons. **Biological Cybernetics**, Springer, v. 41, n. 2, p. 131–138, 1981.
- KWATRA, Vivek et al. Texture optimization for example-based synthesis. In: ACM SIGGRAPH 2005 Papers. [S.l.: s.n.], 2005. P. 795–802.
- LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep learning. **nature**, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015.

- LU, Yang; ZHU, Song-Chun; WU, Ying. Learning FRAME models using CNN filters. In: 1. PROCEEDINGS of the AAAI Conference on Artificial Intelligence. [S.l.: s.n.], 2016. v. 30.
- PORTILLA, Javier; SIMONCELLI, Eero P. **Texture representation and synthesis using correlation of complex wavelet coefficient magnitudes**. [S.l.]: Citeseer, 1999.
- RAAD, Lara et al. A survey of exemplar-based texture synthesis. **Annals of Mathematical Sciences and Applications**, International Press of Boston, v. 3, n. 1, p. 89–148, 2018.
- SIMONYAN, Karen; ZISSERMAN, Andrew. Very deep convolutional networks for large-scale image recognition. **arXiv preprint arXiv:1409.1556**, 2014.
- WEI, Li-Yi et al. State of the art in example-based texture synthesis. **Eurographics 2009, State of the Art Report, EG-STAR**, Eurographics Association, p. 93–117, 2009.
- ZHANG, Chao; WOODLAND, Philip C. Parameterised sigmoid and ReLU hidden activation functions for DNN acoustic modelling. In: SIXTEENTH Annual Conference of the International Speech Communication Association. [S.l.: s.n.], 2015.
- ZHANG, Richard; ISOLA, Phillip et al. The unreasonable effectiveness of deep features as a perceptual metric. In: PROCEEDINGS of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2018. P. 586–595.
- ZHU, Song Chun; WU, Yingnian; MUMFORD, David. Filters, random fields and maximum entropy (FRAME): Towards a unified theory for texture modeling. **International Journal of Computer Vision**, Springer, v. 27, n. 2, p. 107–126, 1998.

"