Identificação de Assinaturas em Quadros por Meio de Características SIFT e Classificação com KNN.

Gabriel Gameiro Perez

Lucas Ferreira  
Mateus Castro Fortes

**Resumo**: Com o surgimento, a popularização e a facilidade de consumo de IA’s (Inteligências artificiais) generativas, aquelas que são capazes de “gerar” conteúdo - seja de maneira escrita, visual, áudio ou uma mescla dessas - se tornou imprescindível a criação de ferramentas que possam combater plágio e uso indevido de propriedade intelectual; Para IA’s generativas de texto, já existem maneiras de se trabalhar, utilizando ferramentas anti-plágio comuns, que precisam sofrer alterações, no entanto a maior parte do trabalho já está feito. Este artigo se propõe a debruçar sobre outro tipo de análise, nosso objeto de estudo são imagens de assinaturas, a fim de promover o debate sobre se é possível distinguir a caligrafia e um humano da de uma IA, para tal, utilizamos a técnica SIFT (Scale-Invariant Feature Transform) para extrair características que distinguem os dois tipos de assinaturas e o algoritmo K-Neareast Neighbors (KNN) para realizar a classificação dos dados obtidos. A discussão fomentada aqui, através da análise de nossos resultados, será comparada com a de pesquisas anteriores, onde apontaremos limitações dos métodos e possíveis melhorias.

Palavras-chave: SIFT, KNN, Keypoints, modelo de machine learning, clustering.

**Abstract:** With the risen, popularization and the easy consumption of generative AI’s (artificial intelligences), the type that can “generate” content, written, visual, audio or a mix of the three, has become clear the need of creating anti-plagiarism tools and inadequate use of intellectual properties; for text generative AI’s, there are already ways of working on it, utilizing common text tools, used on anti-plagiarism that may need to undergo alterations, thus, the majority of the work is already done. This article proposes to deflect on another kind of analysis, our object of study are signatures, with the will to promote the debate if it is possible to differentiate between human or IA made calligraphy, for that, we are utilizing the SIFT (Scale-Inverted Feature Transform) technique for extracting characateristics that can distinguish between signatures and then, utilizing the K-Nearest Neighbors(KNN) algorithm to realize the classification of the data gathered. The discussion fomented here, through the analysis of our results will be further compared to previous works, where we will appoint limitations of methods and adjustments.

Keywords: SIFT, KNN, Keypoints, machine learning models, clustering.

**2. Revisão da Literatura:**

* Apresente uma revisão da literatura relevante sobre técnicas de visão computacional, características SIFT e algoritmo K-Nearest Neighbors (KNN).
* Destaque estudos anteriores relacionados à identificação de assinaturas ou problemas semelhantes.

3.Metodologia.

3.1. Obtenção de dados  
  
A obtenção dos dados para pesquisa foi feita de duas maneiras, a primeira, onde obtivemos as pinturas originais de pintores variados, extraindo-as do site: [https://www.claudemonetgallery.org](https://www.claudemonetgallery.org/).

No que tange os dados obtidos através de inteligência artificial generativa, os membros do grupo utilizaram a IA generativa Stable Diffusion (hospedada em [https://stablediffusionweb.com/#demo](https://stablediffusionweb.com/%23demo)) no dia 30/07/2023, onde o prompt dado para a IA era “monet pictures with his signature”.

Em seguida, para ambos os tipos de imagem, através de um método padronizado de redimensionamento, todas as imagens coletadas permaneceram com uma proporção de **1000x1000 pixels**. Proporção essa escolhida para manter o tamanho do recorte para todas das imagens a serem analisadas sem perder definição e características importantes do objeto de estudo.

Após isso seguiu-se a etapa de recorte das imagens em um tamanho de **430x140 pixels**, tamanho este escolhido pela capacidade de comportar todas as assinaturas de forma competente, extraindo assim nosso elemento final a ser processado.

Imagem em preto e branco de ponte sobre a água

Descrição gerada automaticamente com confiança baixa

Figura A: Obra extraída do acervo online consultado

Barco a vela na água

Descrição gerada automaticamente com confiança baixa

Figura B: Obra com dimensionamento normalizado

Animal na grama

Descrição gerada automaticamente com confiança média

Figura C: Assinatura recortada

3.2. Arquitetura de Plataforma

Para hospedar toda a estrutura da aplicação, utilizamos a plataforma de serviços de cloud AWS, meramente pela disponibilidade de recurso. Uma vez que nos foram dadas uma sorte de contas na mesma somando um total de 600$ em serviços para utilizarmos.

Optamos de forma final por utilizar duas instâncias EC2 para a disponibilização da aplicação, onde uma delas fica responsável pelo acesso público à arquitetura, por hospedar nossa página web e enviar as requisições feitas pelos usuários para a segunda instância. O segundo servidor é nossa API e onde executamos nossos modelos em cima dos dados recebidos pelo front end.

Um ponto também importante de se salientar é a razão da composição simples da arquitetura.

Uma vez com recursos limitados para o desenvolvimento de uma aplicação que normalmente exigiria bastante poder computacional, foi preciso reduzir ao mínimo a escalabilidade e resiliência da composição do projeto para evitarmos a perca desnecessária de recursos valiosos. Desta forma as camadas de segurança da arquitetura se atêm às segmentações de redes nela feita e no acesso SSH com múltiplas chaves .pem.

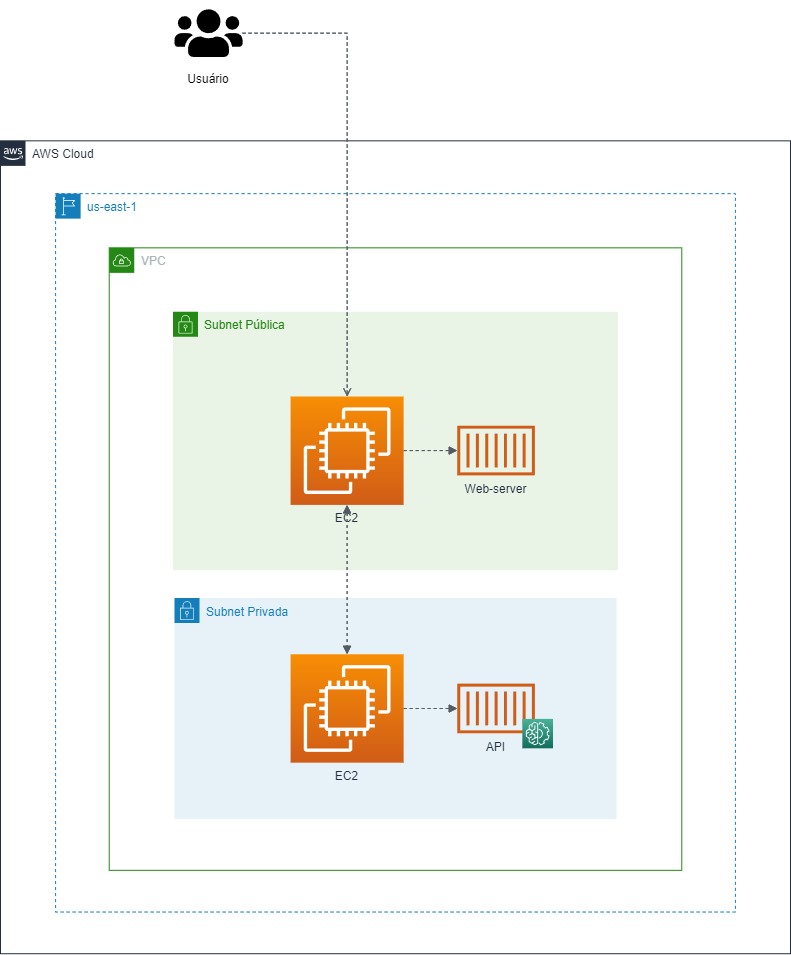
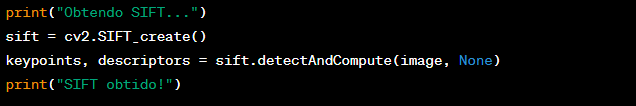


Figura 4: Arquitetura em nuvem

3.3. Extração de características

Para extrair características das assinaturas através do SIFT, utilizamos um script em python, além do auxílio da biblioteca OpenCV (retratada como cv2), popular escolha para trabalhar com imagens e visão computacional.



* Primeiro, criamos um objeto SIFT usando “cv2.SIFT\_create()”.
* Logo em seguida, usamos esse objeto SIFT para dectectar keypoints (pontos de interesse) e calcular descritores SIFT para a imagem.
* **keypoints** contém informações sobre os pontos de interesse encontrados na imagem.
* **descriptors** contém os vetores de características SIFT correspondentes a esses pontos de interesse

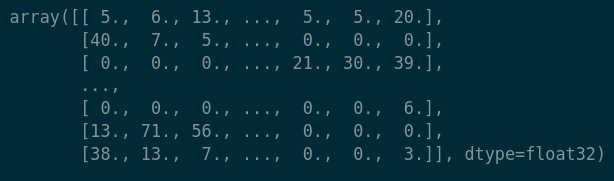
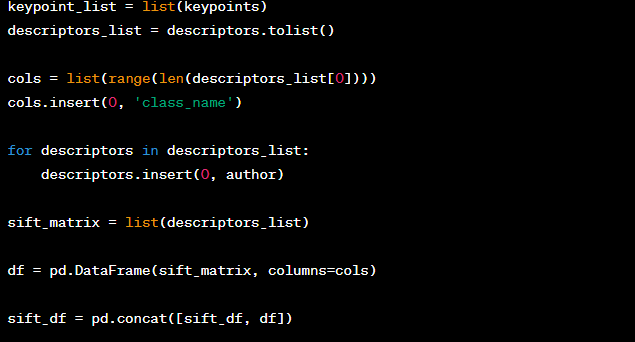


Figura 5: Vetor de descriptors

Figura F: Vetor de descriptors

Isso é essencial para representar as assinaturas como vetores (arrays) de características SIFT, que serão usadas para treinar e testar o modelo KNN.

Após essa obtenção dos **keypoints** e **descriptors**, nós os tranformamos em coordenadas:



Nesta parte do código, estamos convertendo os **keypoints** em coordenadas e adicionando essas coordenadas aos descritores SIFT para criar uma matriz de características que será usada para treinar o modelo KNN.

* **keypoints** contém informações sobre os keypoints detectados na imagem, incluindo suas coordenadas x e y.
* **descriptors** contém os vetores de características SIFT correspondentes a esses keypoints.

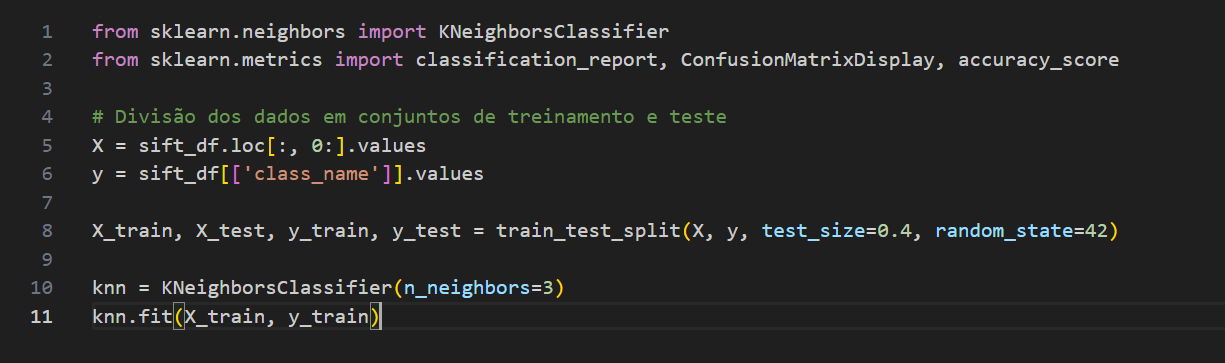
Para converter os keypoints em coordenadas:

* Inicialmente, criamos uma lista chamada **keypoint\_list** que armazena os keypoints.
* Também transformamos os descritores em uma lista chamada **descriptors\_list**.
* Em seguida, criamos uma lista de colunas (**cols**) que representam as características SIFT e adicionamos a coluna 'class\_name' como o primeiro elemento.

Agora, para cada descritor em descriptors\_list (ou seja, para cada keypoint), inserimos a classe (no caso, author) como o primeiro elemento do descritor. Isso é feito para associar cada descritor a uma classe específica (por exemplo, a assinatura correspondente).

Depois disso, criamos um DataFrame (df) com as informações das coordenadas dos keypoints e das características SIFT, onde as colunas representam as características SIFT e a classe a que pertence. Finalmente, concatenamos esse DataFrame df com o DataFrame sift\_df, que acumula todas as informações das imagens processadas. Essa matriz de características (sift\_matrix) é essencial para o treinamento do modelo KNN, onde cada linha representa uma imagem e suas características SIFT.

O treinamento do modelo KNN e divisão dos dados se dá da seguinte maneira:



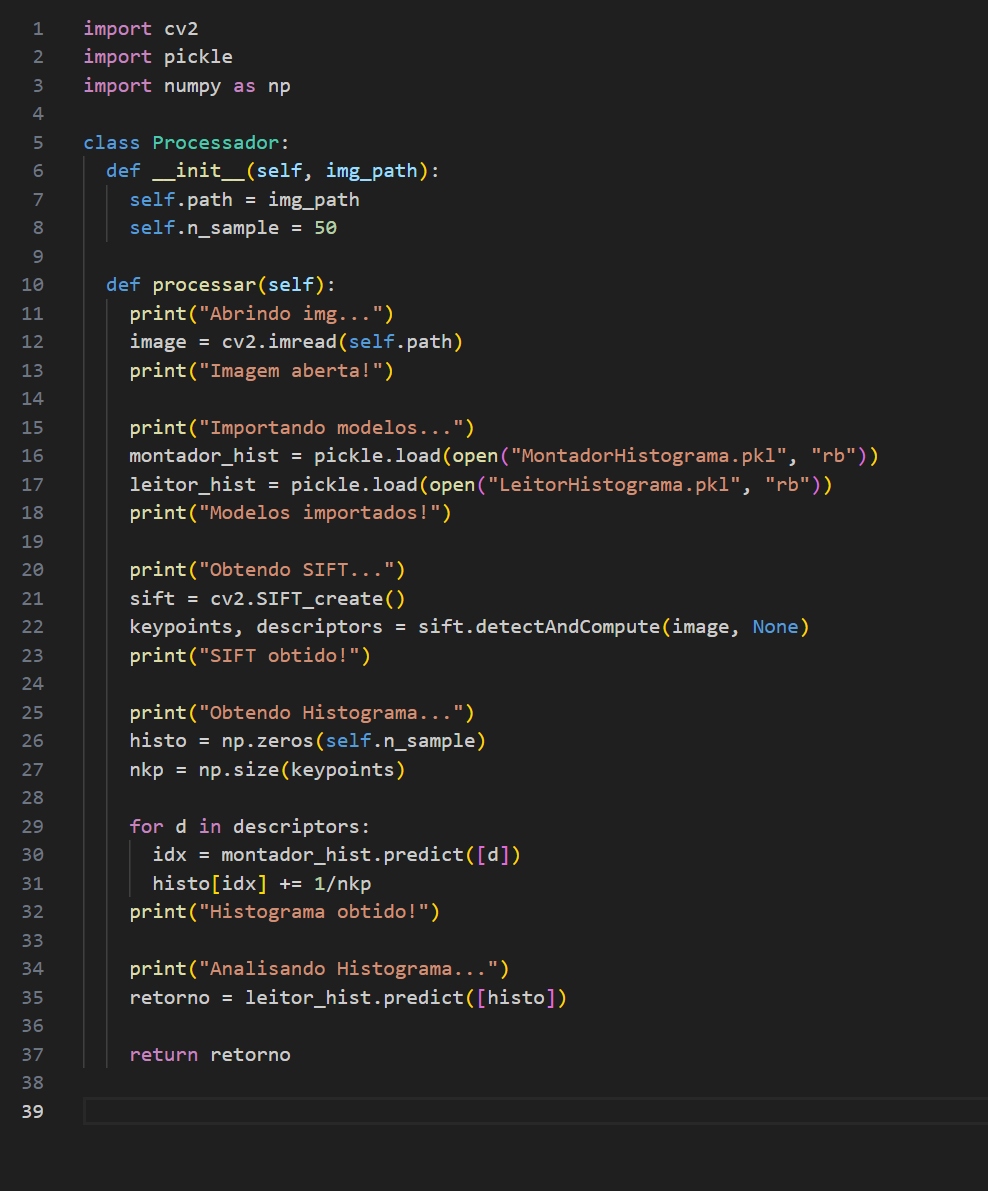
Onde o seguinte ocorre:

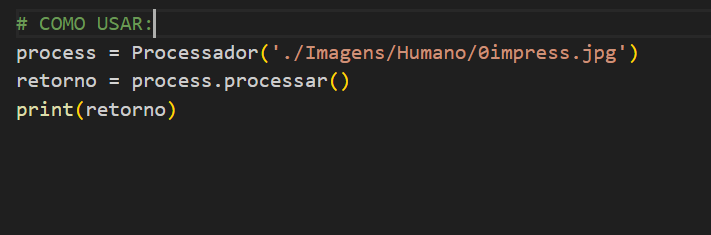
* Importamos a classe **KNeighborsClassifier** da biblioteca **sklearn.neighbors** para criar um modelo KNN.
* Também importamos algumas métricas úteis para avaliar o desempenho do modelo.
* Em seguida, dividimos nossos dados em conjuntos de treinamento (**X\_train**, **y\_train**) e teste (**X\_test**, **y\_test**) usando a função **train\_test\_split**. O parâmetro **test\_size** define a proporção de dados que será usada para teste (40% neste caso).
* Criamos uma instância do modelo KNN com **n\_neighbors=3**. Isso significa que ele considerará os 3 vizinhos mais próximos ao fazer previsões.
* Treinamos o modelo KNN com os dados de treinamento usando **knn.fit(X\_train, y\_train)**.

Essas etapas são fundamentais para treinar o modelo KNN com as características SIFT extraídas e, em seguida, testá-lo para fazer previsões com os dados de teste. A escolha do número de vizinhos (3 neste caso) é um hiperparâmetro que pode ser ajustado com base na natureza dos dados e nos resultados desejados.

3.4 Utilização do modelo

Para consumo do modelo treinado anteriormente e sua utilização, nós optamos por exportá-lo e o utilizar em outro arquivo.





* A primeira parte do código incluem-se as importações necessárias, como a biblioteca OpenCV (**cv2**), a biblioteca **pickle** para carregar modelos previamente treinados e a biblioteca NumPy (**numpy**) para operações numéricas.
* A classe **Processador** é definida para realizar o processamento da imagem. Ela possui os seguintes métodos e atributos:
  + **\_\_init\_\_(self, img\_path)**: O construtor da classe recebe o caminho (**img\_path**) para a imagem que será processada e define **self.n\_sample** como 50, que é o número de características SIFT que serão usadas na análise.
  + **processar(self)**: Este método realiza o processamento da imagem, incluindo a abertura da imagem, a importação de modelos previamente treinados, a extração de características SIFT e a análise do histograma resultante.
* O código inclui um exemplo de uso:
  + Cria uma instância da classe **Processador** com o caminho para uma imagem específica (**'./Imagens/Humano/0impress.jpg'**).
  + Chama o método **processar** para realizar a análise da imagem.
  + Imprime o resultado da análise (**retorno**).

Para analisar uma imagem, basta seguir os seguintes passos:

* Garanta que você tenha as bibliotecas OpenCV (**cv2**), NumPy (**numpy**) e **pickle** instaladas em seu ambiente Python.
* Crie uma instância da classe **Processador** e forneça o caminho para a imagem que deseja analisar. Por exemplo:



* Utilize o método **processar()** na instância criada para processar a imagem:
* O método **processar()** realizará a análise da imagem, incluindo a extração de características SIFT e a análise do histograma.
* O resultado da análise será armazenado na variável **retorno**. Dependendo do modelo treinado e da análise realizada, o resultado pode variar, mas geralmente será uma classe ou uma previsão relacionada à imagem.
* Você pode imprimir ou usar o resultado (**retorno**) conforme necessário para

4. Resultados

A precisão do **método processar** está ligada diretamente com o modelo gerado na etapa anterior, através da nossa pesquisa é possível afirmar com 88% de certeza que uma assinatura em uma imagem foi feita por um ser humano ou IA.

Mas para chegar à um resultado competente de acuracidade utilizando a base de dados e os métodos de processamento previamente apresentados, foi feita uma exploração para encontrar a melhor combinação de valores de parâmetros do modelo programado.

E pra isso foi codificado um loop de treinamento do modelo com dados aleatórios e pré-limitados, e após aproximadamente 5 horas de execução dessa exploração fomos capazes de denotar que os valores mais assertivos para a proposta feita foram:

* Tamanho(%) do nosso Vocabulário SIFT pré-definido: **14%**
* Volume(%) de dados pra teste do modelo: **18,1%**
* Quantidade de **neighbors** para a criação do modelo KNN: **3**

Além dos números, a assertividade do modelo pode, também, ser endossada por dados de:  
  
Gráfico

Descrição gerada automaticamente

Figura 7: Matriz de confusão

Calendário

Descrição gerada automaticamente

Figura 8: Tabela de validações do algoritmo

5. Discussão

Com os dados apresentados, é possível levantar a discussão que no estado atual das IA’s generativas de pintura (mais especificamente A Stable Diffusion), é possível para um algoritmo treinado com SIFT e KNN com os parâmetros por nós utilizados reconhecer quando uma assinatura é feita por uma IA ou por um ser humano, com uma acurácia de 80%.

No entanto, o campo da inteligência artificial é um dos que mais recebe investimentos e pesquisa ao redor do mundo, salientamos aqui a necessidade não somente dessa pesquisa generativa mas também de ferramentas de regulação amparadas em dados para que os autores de obras originais possam ter seus direitos resguardados, além de um trato de segurança de informação, disponibilizando exatamente as bases utilizadas para treinar IA’s generatias e ferramentas anti-plágio. Porém, nosso estudo, conta com uma grande limitação: capacidade de computação, utilizando a infraestrutura da AWS, com servidores no brasil, foram extinguidos em poucos meses de testes a quantia de 100 doláres, sendo possível também salientar que o acesso a ambientes de testes para fins acadêmicos no que tangem computações complexas é altamente restritiva, sendo cara para estudantes e instituições de ensino.

6. Conclusão

Como dito anteriormente, através da leitura deste artigo, é possível levantar uma série de discussões relevantes sobre:

* Inteligências artificiais generativas e sua “geração” de conteúdo
* Ferramentas anti-plágio e sua adaptação para um mundo “generativo”
* Acesso à computação na nuvem para estudantes de nível superior.

Tendo nossas demonstrações sendo baseadas em dados, comprove-se o paradigma que inteligências artificiais são específicas para determinadas funções, onde evidenciamos que IA’s que são capazes de gerar pinturas decorrentes da amálgama de dados (muitas vezes obtidos sem o consentimento do autor original), não “sabem” assinar; Mimetizar a complexa escrita humana e principalmente a assinatura de um indivíduo, elemento que de maneira similar à pintura, carrega estilo e características de personalidade únicas daquele que a confeccionou se mostra impossível até a data presente da publicação deste artigo.

7. Referências:

Acervo de obras utilizado para acúmulo de base de estudo:

[https://www.claudemonetgallery.org](https://www.claudemonetgallery.org/)

Stable Diffusion:

[https://stablediffusionweb.com/#demo](https://stablediffusionweb.com/%23demo)

Alexander Blessing e Kai Wen. Using Machine Learning for Identification of Art Paintings: <https://cs229.stanford.edu/proj2010/BlessingWen-UsingMachineLearningForIdentificationOfArtPaintings.pdf>

Lucas Gianoglio Vinhas. CLASSIFICAÇÃO DE PINTURAS COM APRENDIZADO DEMÁQUINA: <https://repositorio.unesp.br/server/api/core/bitstreams/0fad7bbb-5872-4aee-8fa0-9023543c9245/content>