Vision Transformers aos LLM's Multimodais

André Martins Dantas
Universidade Federal de Goiás
Instituto de Informática (INF)
Goiânia, Brasil
andre.dantas@discente.ufg.br

Maria Eduarda Silva Borba
Universidade Federal de Goiás
Instituto de Informática (INF)
Goiânia, Brasil
maria.borba@discente.ufg.br

Hugo Rodrigues Pessoni
Universidade Federal de Goiás
Instituto de Informática (INF)
Goiânia, Brasil
hugorodriguespessoni@discente.ufg.br

Pedro Ribeiro Fernandes
Universidade Federal de Goiás
Instituto de Informática (INF)
Goiânia, Brasil
pedro.fernandes@discente.ufg.br

***Resumo* — Este artigo explora a evolução dos modelos de linguagem multimodais, com foco principal na transição dos Recurrent Neural Networks (RNNs) para os atuais Large Language Models (LLMs) multimodais, destacando as limitações temporais das RNNs. Uma breve discussão sobre a revolução proporcionada pelos Transformers na manipulação eficiente de sequências extensas de dados precede uma análise mais detalhada dos Vision Transformers, ressaltando como essas arquiteturas transformaram a visão computacional. O artigo também aborda a integração de diversas modalidades em LLMs multimodais, destacando a importância dos Vision Transformers nesse contexto. Um estudo de caso prático é apresentado, delineando um roadmap essencial para compreender a evolução temporal e a crescente relevância desses modelos na inteligência artificial.**

Palavras chaves — Modelos de Linguagem Multimodais, Recurrent Neural Networks (RNNs), Large Language Models (LLMs), Vision Transformers, Visão Computacional, Inteligência Artificial.

I. INTRODUÇÃO E REVISÃO BIBLIOGRÁFICA

A visão computacional e o processamento de linguagem natural são duas áreas importantes da Inteligência Artificial (IA), as quais lidam com o entendimento e a geração de conteúdo visual e textual, respectivamente. Ambas as áreas têm avançado significativamente nos últimos anos, graças ao desenvolvimento de redes neurais profundas e, também, ao aumento da disponibilidade de dados. Uma das arquiteturas mais influentes nesse contexto é o transformer [1], que foi originalmente proposto para tarefas de *Natural Language Processing* (NLP) [2], como tradução automática e modelagem de linguagem. O transformer se tornou a base para os chamados Large Language Models (LLMs), que são modelos de linguagem pré-treinados em grandes volumes de dados textuais, como o GPT-3 e o BERT3 [3]. Esses modelos demonstraram uma capacidade impressionante de gerar e compreender textos em diversos domínios e tarefas, como resposta a perguntas, geração de resumos, escrita criativa, entre outras.

No entanto, o transformer não se limita ao domínio textual. Recentemente, foi mostrado que o transformer também pode ser aplicado ao domínio visual também, dando origem aos Vision Transformers (ViTs) [1]. Os ViTs tratam as imagens como sequências de *patches* (pedaços) e usam a técnica de *self-attention* para aprender as relações entre eles. Os ViTs superam ou combinam técnicas tradicionais como Redes Neurais Convolucionais (CNNs), que são limitadas para capturar a relação contextual entre os recursos da imagem no contexto global. Os ViTs têm diversas aplicações na visão computacional, como segmentação de imagem, classificação, detecção, predição, reconstrução, síntese e telemedicina.

Uma das aplicações mais interessantes dos ViTs é a sua integração com os LLMs, dando origem aos LLMs multimodais[1]. Esses são modelos de linguagem que incorporam informações visuais em seus espaços de representação, permitindo gerar respostas multimodais a partir de entradas de textos e/ou imagens. Por exemplo, um LLM multimodal pode ser capaz de responder a uma pergunta sobre uma imagem, gerar uma legenda para uma imagem, desenhar uma imagem a partir de uma descrição textual, ou até mesmo criar uma imagem a partir do zero. Esses modelos têm potencial para revolucionar a forma como interagimos com o conteúdo digital, abrindo novas possibilidades de comunicação, educação, entretenimento e criatividade.

Neste artigo, discutimos as aplicações e o significado dos ViTs aplicados aos LLMs multimodais, destacando os benefícios e os desafios dessa abordagem.

II. FUNDAMENTOS TEÓRICOS

A. Mecanismos e Técnicas

As Redes Neurais Recorrentes (RNNs) são uma classe de arquiteturas de rede neural projetadas para lidar com dados sequenciais, como texto ou séries temporais. Elas possuem uma estrutura de *loop* que permite a propagação de informações ao longo de sequências, capturando dependências temporais. No entanto, as RNNs podem ter dificuldades em lidar com sequências muito longas devido a problemas de dissipação e explosão do gradiente.

Os Transformers inicialmente foram estudados para o NLP e sua abordagem de *self-attention* [4] permite que cada palavra em uma sequência influencie todas as outras, destacando relações contextuais. Com múltiplas cabeças de atenção para perspectivas variadas, camadas de normalização e *feedforward*, esses modelos convertem sequências de entrada em representações contextuais ricas. São treinados via retropropagação.

Os ViTs representam uma adaptação dos Transformers para tarefas de visão computacional. Ao contrário das arquiteturas convolucionais tradicionais, esses modelos dividem uma imagem em *patches*, tratando cada *patch* como uma "palavra" e aplicando mecanismos de *self-attention* para capturar relacionamentos entre eles. Essa abordagem mostrou-se eficaz, superando modelos convolucionais em algumas tarefas de visão.

Assim como foi surgindo mudanças na arquitetura

transformer para adaptar a visão computacional, tentativas foram feitas para adaptar, também, para lidar com dados de múltiplas modalidades, como texto e imagem, em uma única arquitetura. Utilizando os mecanismos de *self-attention* para processar e entender simultaneamente informações de diferentes tipos, permitindo a criação de sistemas mais abrangentes que podem, por exemplo, compreender uma descrição textual de uma imagem. Essa abordagem multimodal tem implicações significativas em áreas como a compreensão de linguagem visual e a geração de descrições de imagens [5].

B. *Possível solução para o Problema*

Além disso, examinamos a integração de modalidades diversas em LLMs multimodais, identificando os Vision Transformers como peças-chave nesse processo. Este artigo apresenta um caso prático para ilustrar a aplicação desses modelos. Assim, os Vision Transformers emergem como uma possível solução para superar as barreiras entre linguagem e visão, impulsionando uma fase significativa na revolução multimodal e proporcionando uma compreensão mais holística e contextual das informações.

III. METODOLOGIA

A. *O que é Vision Transformer (ViT)?*

As redes de arquitetura Transformer são tipos de redes neurais capazes de processar dados sequenciais, como texto, áudio ou imagens em série. A arquitetura Transformer foi introduzida pela primeira vez em 2017 e tem sido amplamente utilizada em tarefas de processamento de linguagem natural, como tradução automática, análise de sentimentos, geração de texto e classificação de tópicos [4].

A principal vantagem da arquitetura Transformer é sua capacidade de processar dados sequenciais de forma paralela, sem a necessidade de processar uma entrada sequencial passo a passo. Isso torna a arquitetura muito eficiente para treinar modelos de redes neurais em grandes conjuntos de dados.

Além disso, esta arquitetura é escalável e permite a adição de camadas adicionais para aumentar a profundidade do modelo e aprimorar sua capacidade de aprendizado. Isso tem levado ao desenvolvimento de diversas derivações da arquitetura Transformer, incluindo o BERT (*Bidirectional Encoder Representations from Transformers*) Devlin et al. (2018) e o GPT-3 (*Generative Pretrained Transformer* 3) [7].

B. *Funcionalidade e Mecanismo*

A arquitetura Transformer original é construída por um bloco de codificadores (6 camadas de encoder) e outro bloco de decodificadores (este também com 6 camadas de *decorder*), como detalhado na figura 1. O modelo utiliza mecanismos de *self-attention* para capturar as dependências de longo alcance em uma sequência de entrada [4].

A entrada do modelo é uma sequência de vetores de palavras (*embedding*) que são passados para uma série de camadas de codificação. Cada camada de codificação contém dois subcampos: um módulo de *self-attention* e um módulo de *feed-forward*. O módulo de *self-attention* calcula pesos de atenção para cada palavra em relação a todas as outras palavras da sequência, permitindo que o modelo dê mais importância às palavras relevantes para a tarefa em questão. O módulo de *feed-forward*, por sua vez, processa cada palavra individualmente [4].

O modelo também usa um mecanismo de normalização residual e uma camada de saída linear após cada camada de codificação. O processo de codificação é realizado várias vezes para permitir que o modelo capture informações de diferentes granularidades [4].

Depois de serem codificadas, as sequências de entrada são passadas para o decodificador, que é semelhante ao codificador, mas também incorpora informações da saída do modelo. O decodificador também utiliza mecanismos de *self-attention* para gerar uma saída em cada etapa do processo.

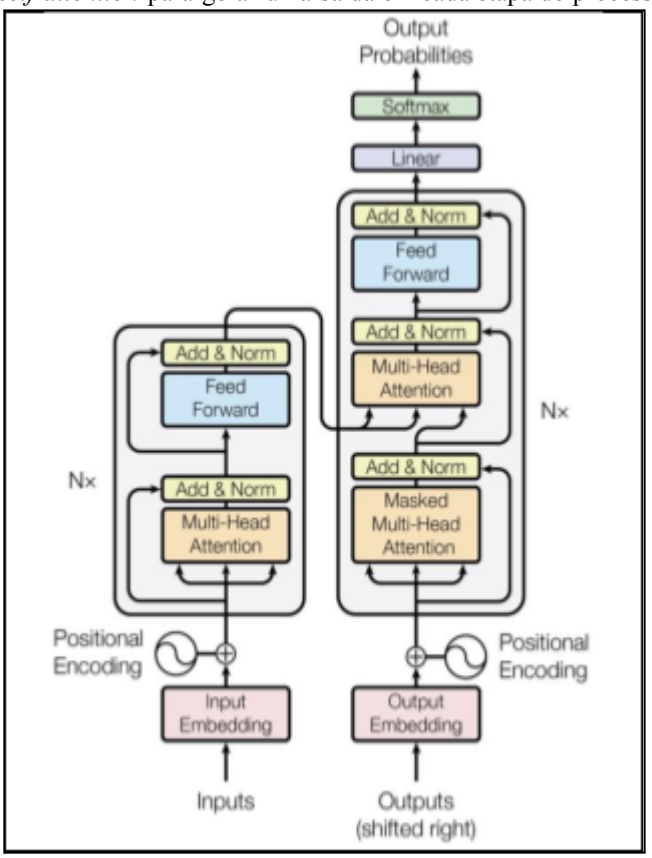

Fig. 1. Arquitetura geral de uma rede Transformer Fonte: Vaswani et al. (2017)[4]

C. *Aplicações e Significado*

A ideia principal é tratar as imagens como uma sequência de *patches* (recortes retangulares da imagem original) e, em seguida, aplicar a arquitetura para gerar uma representação compacta da imagem. Isso é feito em duas etapas: a primeira etapa é a extração de características dos recortes de imagem e a segunda é a agregação das características em uma representação global da imagem [6].

Na primeira etapa, a arquitetura Transformer é usada para extrair características dos recortes de imagem em uma sequência de vetores. Cada recorte é tratado como uma palavra na sequência e a arquitetura é usada para gerar uma representação contextualizada de cada recorte, considerando os demais. Isso significa que a arquitetura dotada de seu mecanismo de atenção é capaz de capturar informações contextuais de diferentes partes e, assim, gerar uma representação mais rica e informativa de cada pequeno grupo [6].

Na segunda etapa, as características geradas a partir dos recortes são agregadas em uma única representação global da imagem.O resultado desse agrupamento de recortes é uma camada linear, onde cada subgrupo representa uma matriz de 1x1. Essa representação global é então usada como entrada para uma camada classificadora, que atribui um rótulo à imagem, como representado pela figura 2 [6].

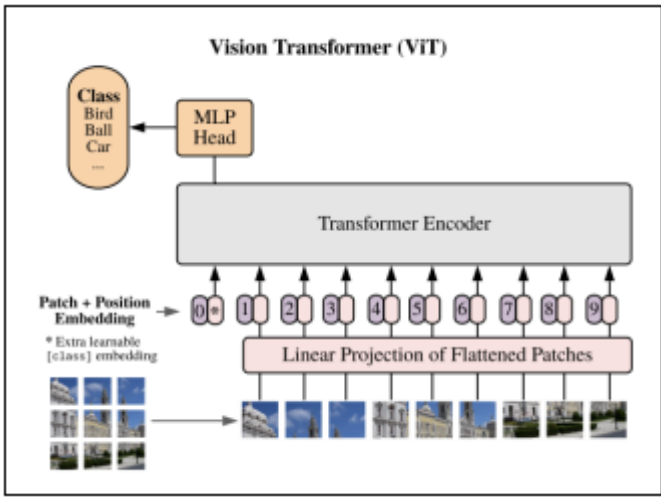

Fig. 2. Apresentação do modelo ViT.

A imagem é dividida em conjuntos de tamanho fixo, acrescidas do seu posicionador e linearizadas. O resultado é aplicado às camadas de codificação e depois aplicadas a uma camada de classificação [6].

O artigo mostrou que essa abordagem ViT, supera as arquiteturas tidas como referência em várias tarefas de classificação de imagem em grande escala, como o reconhecimento de objetos em imagens da base de dados ImageNet, sugerindo que a arquitetura Transformer pode ser aplicada com sucesso em outras áreas além do processamento de linguagem natural.

A arquitetura Transformer aplicada a visão computacional possui apenas camadas codificadoras, estas suficientes para a tarefa de extração de características e classificação de uma imagem de entrada. Como o intuito da rede é apenas apresentar uma distribuição de probabilidades das N possibilidades para uma imagem, a camada de decodificação se torna desnecessária. Em problemas de visão computacional destinados a geração de novas imagens, a inserção de camadas de decodificadores serviria para este propósito, como é feito no processamento de linguagem natural, onde novas representações são geradas no processo de decodificação.

D. Desenvolvimentos Recentes e Adaptações

Há um esforço para melhorar a eficiência dos ViTs, abordando seus altos custos computacionais e de memória. Técnicas como a introdução de módulos leves para adaptações específicas de tarefas têm sido desenvolvidas. Estas adaptações visam melhorar a transferibilidade de ViTs pré-treinados para vários domínios visuais com treinamento adicional mínimo.

Em resumo, os Vision Transformers representam uma mudança significativa na abordagem de tarefas relacionadas a imagens em aprendizado de máquina, oferecendo uma alternativa promissora às CNNs tradicionais. Sua capacidade de processar imagens como sequências e capturar dependências globais oferece uma ferramenta poderosa para tarefas complexas de reconhecimento visual. No entanto, sua eficiência e aplicabilidade em cenários com recursos limitados continuam sendo áreas de pesquisa ativa. O que buscamos com esse artigo é trazer mais clareza sobre esse assunto, uma vez que, seu potencial é tão alto quanto a evolução dos modelos de LLMs utilizados no mercado hoje.

IV. EMBEDDING SIMPLIFICADO

Como dito anteriormente, os ViTs e as CNNs apresentam diferenças fundamentais na visão computacional. Os ViTs, adaptados do processamento de linguagem natural, destacam-se no processamento global de informações e na captura de dependências de longo alcance. Contudo, eles possuem um viés indutivo mais fraco em comparação com as CNNs, o que os torna mais sensíveis à qualidade e quantidade dos dados de treinamento.

No campo emergente da patologia digital (como nosso exemplo simples), o estudo de Deininger *et al.* oferece uma análise comparativa significativa entre estes dois modelos, especificamente na detecção de tumores em imagens patológicas digitais [8] no ano de 2021, um pouco antes do "*boom*" dos LLMs. Os autores realizaram a comparação empregando abordagens auto supervisionadas inovadoras, como técnicas de pré-treinamento em grandes quantidades de imagens não anotadas, para contornar a escassez de dados anotados, que é um desafio comum em patologia digital.

Os resultados deste estudo mostram que, em termos de precisão na detecção de tumores, o ViT superou ligeiramente a CNN alcançando uma precisão marginalmente superior em três dos quatro tipos de tecidos analisados. Essa eficácia pode ser atribuída à habilidade dos ViTs de capturar relações globais nos dados, uma característica crucial para a análise detalhada de imagens patológicas. No entanto, mais esforço de treinamento devido à sua arquitetura mais complexa e à necessidade de grandes quantidades de dados para aproveitar efetivamente suas capacidades de atenção global [8].

Olhando para o futuro, modelos ViTs representam um caminho promissor, oferecendo novas perspectivas para a análise de imagens complexas, com o potencial de transformar a análise atualmente feita de imagens patológicas, oferecendo precisão e detalhamento sem precedentes no diagnóstico e pesquisa médica como ressaltam os autores [8].

Em contrapartida, as CNNs se destacam em eficiência computacional e velocidade para entradas menores, com estruturas hierárquicas que auxiliam em diversas tarefas. Deste modo, a escolha entre eles depende do equilíbrio desejado entre precisão, robustez e eficiência, mas ambas as arquiteturas continuam sendo fundamentais no avanço da visão computacional com aplicações em diversos setores e mercados econômicos.

V. RESULTADOS E CONCLUSÕES

Este estudo prático aborda uma metodologia multimodal integrando ViTs e LLMs para a tarefa específica de geração de descrições visuais. Com base nisso, buscamos desenvolver um exemplo prático de como essas tecnologias podem unir forças para produzir resultados relevantes, nesse caso, para a descrição de patologias encontradas em folhas de plantas. Em outras palavras, é enviado uma imagem de uma folha de uma árvore qualquer, o modelo de classificação com ViT's retorna se a mesma está saudável ou se possui uma das anomalias pré definidas.

Inicialmente, realiza-se o pré-processamento das imagens por meio do ViTFeatureExtractor, uma ferramenta eficiente na extração de features visuais representativas, como, por exemplo, a existência de certas patologias pré-definidas nas folhas das plantas.

O ViTFeatureExtractor utiliza a arquitetura Transformer para realizar a extração de *features* visuais em uma imagem. Inicialmente, a imagem é dividida em *patches*, e cada *patch* é linearizado para formar uma sequência de *embeddings*. Essa sequência é então processada por camadas de *self-attention*, permitindo que cada patch interaja contextualmente com outros. As representações resultantes são utilizadas como features visuais, capturando eficientemente informações significativas da imagem. Essa abordagem inovadora oferece uma representação rica e contextualizada, contribuindo para o desempenho excepcional dos modelos Vision Transformers em tarefas de visão computacional, além do output perfeito para a geração

de descrições multimodais.

O conjunto de dados é então transformado para incorporar essas *features*, garantindo uma representação mais robusta no contexto do treinamento do modelo. Durante esta fase, destaca-se a relevância da colaboração entre o ViT e o LLM GPT-2, onde o último é empregado para gerar descrições textuais a partir das features visuais extraídas pelo primeiro. Esse procedimento, conhecido como geração de *features* textuais por LLM, proporciona uma abordagem única para a compreensão contextual das imagens.

O processo de geração de *features* textuais utiliza do ViTFeatureExtractor para obter as representações visuais da imagem. Essas representações são então fornecidas como entrada para um LLM, como o GPT-2, que é treinado para gerar descrições textuais contextualmente relevantes. O LLM utiliza as features visuais do ViT para informar a geração do texto, criando assim uma abordagem multimodal que integra informações visuais e linguísticas de maneira coesa e eficaz.

O resultado final é um modelo treinado capaz de gerar descrições multimodais, recebendo imagens, classificando-as, gerando descrições visuais e, por fim, mesmo que ainda com baixa coerência entre a imagem e o output, consolidando as informações visuais e textuais em um output multimodal. Esta abordagem inovadora não apenas evidencia a sinergia entre ViTs e LLMs, mas também ressalta a potencialidade de modelos multimodais para uma representação mais rica e holística de dados complexos, contribuindo para avanços significativos em áreas como visão computacional e processamento de linguagem natural.

VI. REFERÊNCIAS

[1] LI, Yunxin. HU, Baotian.WANG, Wei. CAO, Xiaochun. ZHANG, Min. Towards Vision Enhancing LLMs: Empowering Multimodal Knowledge Storage and Sharing in LLMs. arXiv preprint arXiv:2311.15759, 2023.

[2] MARKOWITZ, Dale. Explicação Sobre Transformadores: Entenda O Modelo Por Trás De GPT, BERT, E T5. YOUTUBE https://www.youtube.com/watch?v=SZorAJ4I-sA&list=PLD1f4T8AE2_sOU98jTWpVdqv1ADcTs7kc&index=5.

[3] XU, Mengwei et al. A Survey of Resource-efficient LLM and Multimodal Foundation Models. arXiv preprint arXiv:2401.08092v1, 2024.

[4] VASWANI, Ashish et al. Attention is all you need. In NIPS, 2017. arXiv preprint arXiv:1706.03762, 2017.

[5] SCHAMBACH, Maximilian. A Brief History of Vision Transformers: Revisiting Two Years of Vision Research. Merantix Momentum Insights, 2022.

[6] DOSOVITSKIY, A. et al. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929, 2020.

[7] BROWN, T. et al. Language models are few-shot learners. In: LAROCHELLE, H. et al.(Ed.). Advances in Neural Information Processing Systems. Disponível em: <https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf>

[8] STEINER, Andreas et al. How to train your vit? data, augmentation, and regularization in vision transformers. arXiv preprint arXiv:2106.10270, 2021.

[9] HAMADI, Raby. Large Language Models Meet Computer Vision: A Brief Survey. arXiv preprint arXiv:2311.16673, 2023.

[10] YUE, Xiaoyu et al. Vision transformer with progressive sampling. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021. p. 387-396.

[11] PARK, Namuk; KIM, Songkuk. How do vision transformers work?. arXiv preprint arXiv:2202.06709, 2022.

[12] THISANKE, Hans et al. Semantic segmentation using Vision Transformers: A survey. Engineering Applications of Artificial Intelligence, v. 126, p. 106669, 2023.

[13] CHEN, Shoufa et al. Adaptformer: Adapting vision transformers for scalable visual recognition. Advances in Neural Information Processing Systems, v. 35, p. 16664-16678, 2022.

[14] PARVAIZ, Arshi et al. Vision Transformers in medical computer vision—A contemplative retrospection. Engineering Applications of Artificial Intelligence, v. 122, p. 106126, 2023.

[15] WANG, Yikai et al. Multimodal token fusion for vision transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022. p. 12186-12195.

VII. APÊNCIDES

A - Link com o código completo Google Colab

https://colab.research.google.com/drive/1KdXPGAbMEdt-MynQFODEac6CI9jlMmAF?usp=sharing

B - Imagem do Dataset utilizado

C - Código por células

Célula 1 - Carregamento de Dados e Visualização

```
# Importação de bibliotecas
from datasets import load_dataset
from transformers.utils.dummy_vision_objects import
ImageGPTFeatureExtractor
import random
from PIL import ImageDraw, ImageFont, Image

# Carrega o conjunto de dados 'beans'
ds = load_dataset('beans')

# Obtém rótulos do conjunto de treinamento
labels = ds['train'].features['labels']

# Função para exibir exemplos de fotos e suas labels
def show_examples(ds, seed: int = 1234, examples_per_class: int = 3,
size=(350, 350)):

  w, h = size
  labels = ds['train'].features['labels'].names
  grid = Image.new('RGB', size=(examples_per_class * w, len(labels) * h))
  draw = ImageDraw.Draw(grid)
  font =
ImageFont.truetype("/usr/share/fonts/truetype/liberation/LiberationMono-Bold.ttf", 24)

  for label_id, label in enumerate(labels):

    # Filter the dataset by a single label, shuffle it, and grab a few samples
    ds_slice = ds['train'].filter(lambda ex: ex['labels'] ==
label_id).shuffle(seed).select(range(examples_per_class))

    # Plot this label's examples along a row
    for i, example in enumerate(ds_slice):
      image = example['image']
```

```
            idx = examples_per_class * label_id + i
            box = (idx % examples_per_class * w, idx // examples_per_class *
h)
            grid.paste(image.resize(size), box=box)
            draw.text(box, label, (255, 255, 255), font=font)

    return grid

show_examples(ds, seed=random.randint(0, 1337), examples_per_class=3)
```

Célula 2 - Pré-processamento com ViTFeatureExtractor

```
# Importação de bibliotecas
from transformers import ViTFeatureExtractor

image = ds['train'][400]['image']

# Modelo ViT pré-treinado
model_name_or_path = 'google/vit-base-patch16-224-in21k'
feature_extractor =
ViTFeatureExtractor.from_pretrained(model_name_or_path)

# Processa um exemplo
feature_extractor(image, return_tensors='pt')

def process_example(example):
    inputs = feature_extractor(example['image'], return_tensors='pt')
    inputs['labels'] = example['labels']
    return inputs
```

Célula 3 - Transformação do Conjunto de Dados

```
from datasets import load_dataset

ds = load_dataset('beans')

def transform(example_batch):
    # Torna uma lista de imagens PIL em valores de pixeis
    inputs = feature_extractor([x for x in example_batch['image']],
return_tensors='pt')
    inputs['labels'] = example_batch['labels']
    return inputs

prepared_ds = ds.with_transform(transform)
```

Célula 4 - Treinamento do Modelo

```
# Importação de bibliotecas
import numpy as np
from datasets import load_metric
from transformers import ViTForImageClassification, TrainingArguments,
Trainer
import torch

# Definindo métricas para a avaliação do modelo
metric = load_metric("accuracy")
def compute_metrics(p):
    return metric.compute(predictions=np.argmax(p.predictions, axis=1),
references=p.label_ids)
# Configuração do DataLoader
def collate_fn(batch):
    return {
        'pixel_values': torch.stack([x['pixel_values'] for x in batch]),
        'labels': torch.tensor([x['labels'] for x in batch])
    }

# Carrega os rótulos
labels = ds['train'].features['labels'].names

# Inicializa um modelo ViT para classificação de imagens
model = ViTForImageClassification.from_pretrained(
    model_name_or_path,
    num_labels=len(labels),
    id2label={str(i): c for i, c in enumerate(labels)},
    label2id={c: str(i) for i, c in enumerate(labels)}
)

# Configuração de argumentos de treinamento
training_args = TrainingArguments(
  output_dir="./vit-base-beans-demo-v5",
  per_device_train_batch_size=16,
  evaluation_strategy="steps",
  num_train_epochs=4,
  fp16=True,
  save_steps=100,
  eval_steps=100,
  logging_steps=10,
  learning_rate=2e-4,
  save_total_limit=2,
  remove_unused_columns=False,
  push_to_hub=False,
  report_to='tensorboard',
  load_best_model_at_end=True,
)

# Configuração do Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    data_collator=collate_fn,
    compute_metrics=compute_metrics,
    train_dataset=prepared_ds["train"],
    eval_dataset=prepared_ds["validation"],
    tokenizer=feature_extractor,
)

# Treinamento do modelo
train_results = trainer.train()
trainer.save_model()
```

Célula 5 - Avaliação do Modelo

```
# Avaliação no conjunto de validação
metrics = trainer.evaluate(prepared_ds['validation'])
trainer.log_metrics("eval", metrics)
trainer.save_metrics("eval", metrics)
```

Célula 6 - Extração das features visuais

```
from transformers import ViTFeatureExtractor
from PIL import Image

# Carregar modelo ViT pré-treinado para extrair features visuais
vit_model_name = "google/vit-base-patch16-224-in21k"
feature_extractor = ViTFeatureExtractor.from_pretrained(vit_model_name)

def extract_visual_features(image_path):
    # Carregar e extrair features visuais da imagem usando o modelo ViT
    image = image_path
    inputs = feature_extractor(image, return_tensors="pt")
    return inputs
```

Célula 7 - Geração de features textuais por LLM

```
from transformers import GPT2LMHeadModel, GPT2Tokenizer

# Carregar modelo pré-treinado de linguagem (GPT-2)
tokenizer = GPT2Tokenizer.from_pretrained("gpt2", padding_side='left')
lm_model = GPT2LMHeadModel.from_pretrained("gpt2")

def generate_textual_features(visual_features, attention_mask):
    # Gerar descrição usando GPT-2 com as features visuais, a máscara de
atenção e outros parâmetros
```

```
    description_ids = lm_model.generate(inputs=visual_features,
attention_mask=attention_mask, max_new_tokens=10)
    return description_ids
```

Célula 8 - Geração de Descrições Multimodais

```
def generate_description(image_path):
    # Carregar e extrair features visuais da imagem usando o modelo ViT
    image = image_path
    inputs = feature_extractor(image, return_tensors="pt")

    # Remover a chave 'pixel_values' dos inputs
    inputs = {k: v for k, v in inputs.items() if k != 'pixel_values'}

    # Extrair a máscara de atenção da imagem
    attention_mask = feature_extractor(image,
return_tensors="pt").get('attention_mask')

    # Definir o ID do token de preenchimento como o ID do token EOS
    pad_token_id = tokenizer.eos_token_id

    # Definir o lado do preenchimento como 'esquerda'
    padding_side = 'left'

    # Gerar descrição usando GPT-2 com as features visuais, a máscara de
atenção, o ID do token de preenchimento e o lado do preenchimento
    description_ids = lm_model.generate(**inputs,
attention_mask=attention_mask, pad_token_id=pad_token_id,
max_new_tokens=10)

    # Converter IDs de descrição de volta para texto
    description = tokenizer.decode(description_ids[0],
skip_special_tokens=True)
    return description

# Escolher aleatoriamente um exemplo do conjunto de treinamento
random_index = random.randint(0, len(ds['train']) - 1)
random_image_path = ds['train'][random_index]['image']

# Exemplo de uso
description = generate_description(random_image_path)
print("Generated Description:", description)
```