

## Relatório 27 - Prática: Modelos Generativos (III)

Lucas Augusto Nunes de Barros

### Descrição das atividades

## 1. Generative Models

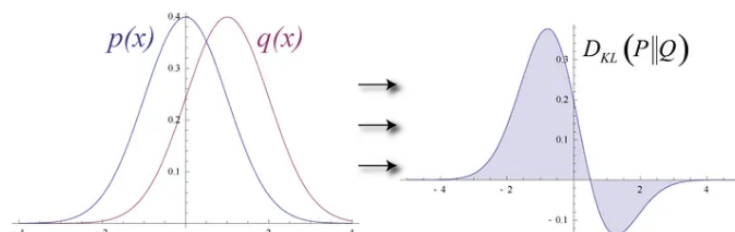
### 1.1 Variational Autoencoders

O *Autoencoder* (AE) é uma rede neural utilizada para aprender representações compactas (espaço latente) de dados de forma não supervisionada. Consiste em duas partes: o *encoder*, que comprime os dados de entrada em um vetor latente de menor dimensão, e o *decoder*, que reconstrói os dados a partir desse vetor. O objetivo é minimizar o erro de reconstrução, garantindo que os dados de saída sejam o mais similares possível aos da entrada original. *Autoencoders* são amplamente usados para tarefas como redução de dimensionalidade, remoção de ruído (*denoising*) e detecção de anomalias, mas não são projetados para geração de novos dados.

O funcionamento dessas redes se baseia em três etapas principais: compressão dos dados de entrada, representação dos dados no espaço latente e reconstrução dos dados a partir do espaço latente. O *encoder* mapeia os dados de entrada, uma imagem ou outro dado estruturado, para um espaço latente, que é um vetor de menor dimensão contendo as características mais importantes. O *decoder*, por sua vez, usa esse vetor para reconstruir os dados originais. A função de perda mede a diferença entre a entrada, dado original, e a saída reconstruída.

Já um *Variational Autoencoder* (VAE) é uma extensão dos *autoencoders* que possui uma abordagem probabilística, tornando-o um modelo de fato generativo. Diferentemente dos AEs tradicionais, os VAEs modelam o espaço latente como uma distribuição, geralmente gaussiana, permitindo a geração de novos dados a partir da amostragem. O *encoder* produz uma média ( $\mu$ ) e uma variância ( $\sigma$ ) para cada entrada, enquanto o *decoder* reconstrói os dados a partir de amostras dessa distribuição. A função de perda combina o erro de reconstrução com a divergência Kullback-Leibler (KL), que regulariza o espaço latente para se aproximar de uma distribuição padrão.

A divergência KL é uma métrica que mede quão diferente uma distribuição de probabilidade  $P$  é de outra distribuição de probabilidade  $Q$ , ou seja, ela quantifica a perda de informação ao usar a distribuição  $Q$  para aproximar os valores de  $P$ .



A imagem ilustra como a divergência KL quantifica a diferença entre duas distribuições usando uma área sombreada que reflete a perda de informação ao aproximar uma distribuição pela outra. À esquerda, as curvas gaussianas destacam as diferenças visuais, enquanto à direita, o gráfico da divergência KL traduz isso em um valor numérico, a área da figura sombreada. No contexto dos VAEs, isso ajuda a regularizar o espaço latente, garantindo que o modelo aprenda uma representação útil e gerativa.

Regularizar o espaço latente significa usar a divergência KL para forçar a distribuição aprendida pelo *encoder* a se aproximar de uma distribuição padrão, criando um espaço latente contínuo, uniforme e estruturado. Isso é especialmente importante para redes do tipo VAE, pois garante que o modelo poderá gerar novos dados de forma controlada em vez de apenas reconstruir os dados de entrada de maneira aleatória.

## 1.2 Generative Adversarial Networks

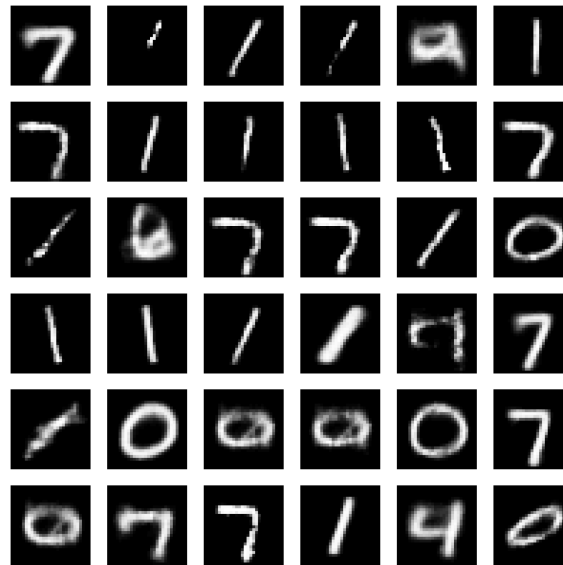
As *Generative Adversarial Network* (GAN) representam um modelo de aprendizado profundo que opera de forma similar às *Variational Auto Encodes*, sendo composta por dois componentes principais: o gerador (*generator*) e o discriminador (*discriminator*). O gerador mapeia o ruído aleatório a partir de uma distribuição normal e tenta reproduzir essa distribuição de probabilidade que imita os dados reais, enquanto o discriminador aprende a distinguir entre imagens reais e as geradas por esse gerador.

A rede leva esse nome pois a forma com que esse processo ocorre é adversarial, ou seja, o gerador entende o discriminador como um “adversário” e tenta enganá-lo, produzindo imagens que parecem reais. Diferentemente das redes VAE, que assumem distribuições gaussianas no espaço latente, as redes GAN não modelam explicitamente uma distribuição latente fixa, permitindo maior flexibilidade, mas também maior complexidade no treinamento, que visa alcançar um ponto onde o discriminador não consiga mais diferenciar as imagens geradas e reais.

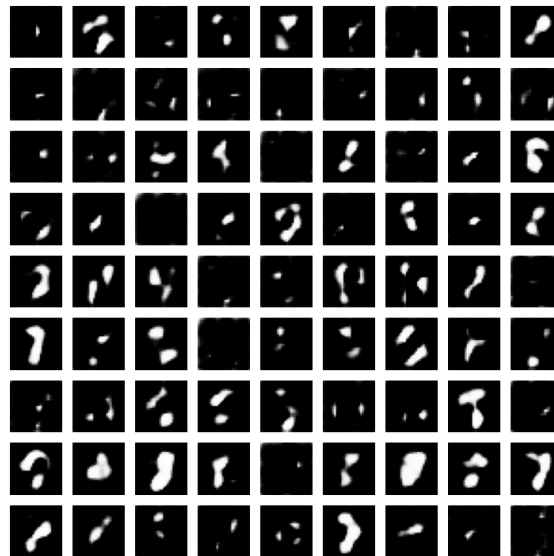
## 1.2 Prática

As práticas foram desenvolvidas sem maiores intercorrências. Para o desenvolvimento das atividades foi utilizado o dataset MNIST que contém números de 0-9 escritos a mão. Esse dataset foi escolhido devido às suas características similares ao *Fashion-MNIST* utilizado durante as aulas do curso. Além de possuir imagens de mesmas dimensões, também há quantidade suficiente de imagens para treinar a GAN. Alguns testes foram realizados com o dataset CIFAR10 porém os resultados não foram satisfatórios, de forma que foram deixados de lado para esse relatório.

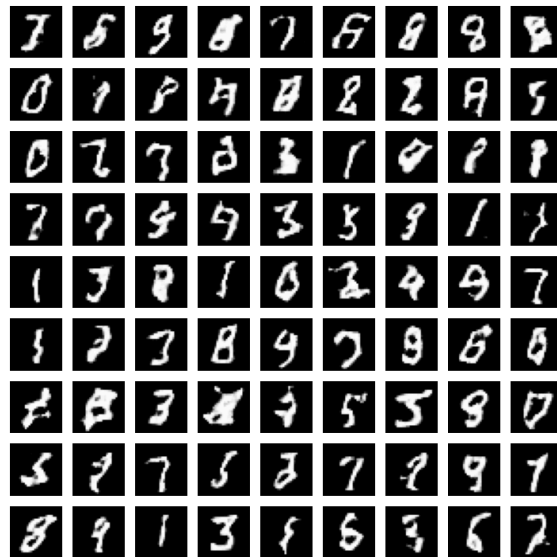
Abaixo são apresentadas algumas saídas dos modelos treinados.



Saída da rede VAE



Primeira imagem gerada pela GAN durante o treinamento



Última imagem gerada pela GAN durante o treinamento

## 2. Let's build GPT (Generative Pre-trained Transformer)

A proposta do vídeo proposto é implementar um modelo GPT desenvolvendo seu passo a passo e entendendo os componentes existentes em um modelo de geração de texto.

O *dataset Tiny Shakespear*, é um conjunto de dados que contém textos de William Shakespeare. Ele é composto por cerca de 1,1 milhão de caracteres, organizados em sequências de texto que preservam a estrutura linguística e estilística do autor. Esse foi o *dataset* utilizado para treinar os modelos de linguagem implementados durante o vídeo proposto.

Transformadores são uma arquitetura de rede neural introduzida em 2017 no artigo "*Attention is All You Need*" (Vaswani et al.), projetada para tarefas de processamento de linguagem natural. Diferente de arquiteturas anteriores, como redes recorrentes (RNNs), os Transformadores utilizam o mecanismo de atenção para modelar a dependência entre os *tokens* em uma sequência.

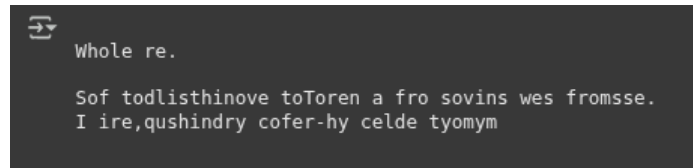
Esse tipo de rede consiste em camadas de codificadores e decodificadores, com o mecanismo de *self-attention*, permitindo os *tokens* perceberem uns aos outros, dessa forma capturando relações contextuais entre eles. Esse mecanismo opera projetando os *embeddings* e calculando uma matriz de pesos de atenção com base na similaridades encontradas.

O GPT - *Generative Pre-trained Transformer*, é uma arquitetura baseada no conceito de Transformadores e focada na geração de texto. Esses modelos são pré-treinados em grandes quantidades de texto, capturando as relações entre os *tokens*. Após o pré-treinamento, os modelos são ajustados (*fine-tuning*) para tarefas específicas. O modelo GPT utiliza apenas o componente decodificador do Transformer, tornando-o ideal para gerar texto coerente e contextualmente relevante.

A base fundamental para a geração de texto é o processo de *tokenização* do texto, convertendo palavras em *tokens*. Cada *token* é então mapeado para um vetor

de *embedding*, que representa as características semânticas dos tokens em um espaço multidimensional. Esses *embeddings* são então processados pelas camadas de *self-attention* para aprenderem informações contextuais de cada *token*. Durante a geração, o modelo prevê o próximo *token* com base no contexto anterior. Esse processo é repetido iterativamente, produzindo texto coerente que reflete padrões aprendidos durante o treinamento.

Inicialmente é desenvolvido um modelo que não utiliza a relação entre os *tokens*, ou seja, a rede aprende a relação de cada *token* porém sem levar em consideração os *tokens* anteriores da sequência (contexto). Uma forma de contornar essa necessidade de contextualizar a geração dos *tokens* é através de um truque matemático, utilizado para calcular a relação com os *tokens* anteriores da sequência.



```
Whole re.  
  
Sof todlisthinove toToren a fro sovins wes fromsse.  
I ire,qushindry cofer-hy celde tyomym
```

Saída do modelo após o treinamento sem o truque matemático.

O truque matemático referido envolve o uso de operações matriciais para calcular relações contextuais. Na primeira versão do modelo, a ausência desse truque resulta em um modelo que não escreve corretamente, mesmo com treinamento exaustivo o modelo é incapaz de escrever de forma coerente, além de ser computacionalmente ineficiente, pois faz uso de *loops* de repetição aninhados.

O uso desse truque permite ao modelo generalizar melhor para pesos dinâmicos melhorando a capacidade do modelo de capturar relações contextuais e linguísticas, essencial para a geração de contexto.

Outras versões do truque matemático são expostas, sendo na segunda versão os *loops* substituídos por uma multiplicação matricial, uma matriz de pesos uniformes é utilizada, aumentando a eficiência, pois agora para prever *tokens* era tirada uma média dos *tokens* anteriores, contextualizando melhor.

Já na terceira versão a normalização é feita com uso da função *softmax* e ainda mantendo os pesos uniformes. Enquanto que na quarta versão é implementado o próprio método de *self-attention*, utilizando uma matriz de pesos dinâmicos, permitindo aprendizado adaptativo.

Essa última versão é superior às demais por sua capacidade de calcular pesos dinâmicos baseados na similaridade entre *tokens*, o que permite ao modelo aprender partes do contexto mais relevantes e quais as relações intrínsecas dos elementos linguísticos envolvidos em cada contexto.

Alguns conceitos novos são apresentados para explicar como o truque matemático faz com que o modelo aprenda as relações entre *tokens*, esses são os conceitos de *queries*, *keys* e *values*, fundamentais para o funcionamento do mecanismo de atenção, incluindo o *self-attention* utilizado no modelos de exemplo implementado durante o vídeo.

De forma simplista as *queries* (consultas) representam os vetores que cada *token* utiliza para buscar informações relevantes no contexto da sequência. As *keys* (chaves), por sua vez, são vetores que descrevem as características de cada *token*, funcionando como uma representação que outros tokens podem consultar. Assim, as *queries* e as *keys* interagem sendo uma o objeto de análise da outra. Já os *values*

(valores) representam as contribuições de cada *token* para a representação final de outros *tokens*, ou seja, como ele interage com os demais *tokens*.

Em notas o autor do curso repassa algumas informações e uma delas é a diferença entre os termos *attention*, *self-attention* e *cross-attention*. Essa diferença se baseia na origem das *queries*, *keys* e *values*. O método *attention* é o conceito geral, enquanto o *self-attention* aplica o método em uma única sequência de *tokens*, o *cross-attention*, conecta duas sequências distintas, onde geralmente uma sequência depende da outra. Em modelos GPT, apenas o *self-attention* é utilizado, com uma máscara para garantir a causalidade (*tokens* futuros não tem influência).

Após implementar o mecanismo de *self-attention* a rede ainda não consegue escrever de forma satisfatória, pois a relação compreendida entre os elementos linguísticos ainda é fixa, ou seja, o modelo apenas compreende uma das formas como os elementos podem se combinar. Para solucionar essa limitação é introduzido o conceito de *multi-head attention*. Até então, o método de *self-attention* havia sido implementado em apenas uma “cabeça”, fazendo com que a compreensão do modelo fosse limitada a capacidade interpretativa desta cabeça, sendo assim o *multi-head attention* traz possibilidade de novas interpretações, permitindo ao modelo compreender múltiplas relações entre *tokens*.

O *single-head attention*, como foi implementado, utiliza um único conjunto de projeções, limitando o modelo a uma única visão das dependências contextuais. Já o *multi-head attention* divide o espaço de *embedding* em várias cabeças, permitindo que cada uma se especialize em diferentes aspectos do contexto, como gramática, semântica ou co-referências, resultando em uma compreensão mais completa do contexto. Nos modelos GPT, o *multi-head attention* é essencial para a geração de textos coerentes, pois combina informações de várias “cabeças” para criar uma resposta mais completa.

```
I am sound to do for a king sleep:  
I came to convert thy grief; and then be thief  
My indictment state and heart my soldier;  
Some thy fable of life is flat, to woo.  
  
ESCALUS:  
Learn's is that, and but that thy, by edict.  
  
POLIXENES:  
Your tongue, my lord.  
If you did mean they will this bud most know two:  
if you wish met; but that they smoth were noted trainful  
doing the one, and they stand goods for  
minemen, know not at such receivity to me  
welcome tof what's seen men.  
  
Shepherd:  
Out of this, night, if thou!  
  
ESCALUS:  
What are the prince, happy neck of his passes.  
  
POMPHEY:  
Then what make, fit shore sound for some requish,  
short this he hath done; would afflict him the mock.
```

Saída do modelo após a implementação do truque matemático e do *Multi-Head Attention*

## **Conclusão**

Baseado no que foi exposto sobre as tecnologias, é notável que sistemas de inteligência artificial fundamentados transformers, representam um avanço significativo no processamento de linguagem natural. Sua capacidade de geração e contextualização demonstram um enorme potencial para diversas aplicações em áreas diversas.

## Referências

[1] KERAS.IO. *Layer activation functions*

Disponível em: <<https://keras.io/api/layers/activations/#leakyrelu-function>> , Acesso em 12 de junho de 2025

[2] TENSORFLOW.ORG. *Tiny Shakespeare*

<[https://www.tensorflow.org/datasets/catalog/tiny\\_shakespeare?hl=pt-br](https://www.tensorflow.org/datasets/catalog/tiny_shakespeare?hl=pt-br)> , Acesso em 12 de junho de 2025

[3] PROCEEDINGS.NEURIPS.CC. *Attention is All you Need*

Disponível em:

<[https://proceedings.neurips.cc/paper\\_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf)> , Acesso em 14 de junho de 2025

[4] SOL.SBC.ORG.BR.

Disponível em: <<https://sol.sbc.org.br/index.php/dsw/article/view/21907/21730>> , Acesso em 14 de junho de 2025

[5] KAGGLE.COM. Procurando Nemo Script PT-BR

Disponível em: <<https://www.kaggle.com/datasets/ashtrindade/procurando-nemo-script-ptbr>> , Acesso em 15 de junho de 2025

[6] YOUTUBE.COM. *Let's build GPT: from scratch, in code, spelled out.*

Disponível em: <<https://youtu.be/kCc8FmEb1nY>> , Acesso em 12 de junho de 2025