Deep Generative Models

Universidade do Minho, Departamento de Informática
Braga, Portugal

Abstract. Os *generative models*, que em tempos eram praticamente impossíveis de aplicar, evoluíram ao ponto de produzir resultados cada vez mais plausíveis e indistinguíveis da realidade. Ao longo do presente artigo científico iremos explorar os modelos generativos, com principal foco, nos modelos generativos profundos, que se baseiam em métodos de *deep learning*. No entanto, é difícil abordar modelos generativos sem falar de modelos discriminativos, pelo que neste documento ambos os conceitos serão detalhadamente explicados. Além disto, ao longo deste artigo, serão também abordados alguns exemplos de modelos generativos profundos, bem como casos de aplicação no mundo real, que mostram não só a sua utilidade e interesse, mas também os impactos revolucionários que trouxeram em áreas críticas tais como a saúde.

Keywords: Deep Learning · Machine Learning · Redes Neuronais · Algoritmos · Modelo Generativo · Modelo Discriminativo · Inteligência Artificial

1 Introdução

Nos últimos anos, os modelos generativos causaram um grande impacto nas várias áreas e subáreas de Inteligência Artificial e de *Machine Learning*. O objetivo destes modelos é adquirir a capacidade de inferir conhecimento através da aprendizagem, tendo por base um conjunto de dados de treino. Para tal, é necessário encontrar uma distribuição de dados que seja o mais semelhante possível à verdadeira. Como não é possível aprender a distribuição exata dos dados, irão ocorrer sempre variações, portanto é difícil alcançar uma precisão muito alta quando lidamos com dados da vida real.

Assim, para alcançar uma distribuição que melhor se ajuste aos dados, faz-se uso das redes neuronais, que, por sua vez, tratam-se de modelos computacionais capazes de reconhecer padrões e, principalmente, aprender continuamente. A Aprendizagem Profunda ou *Deep Learning* é um ramo de *Machine Learning* que inclui algoritmos capazes de modelar distribuições complexas e de alto nível, usando grafos profundos, isto é, com múltiplas camadas. Um exemplo são as redes neuronais profundas, que só mais recentemente é que se tornou possível treinar devido ao surgimento de recursos computacionais mais rápidos (GPU's). Estas consistem numa conjunção das redes neuronais com a lógica de organização por camadas e nós. Portanto, a inclusão de redes neuronais profundas e métodos estocásticos, para otimização, nestes modelos generativos, permitiu-nos chegar ao

termo de "modelos generativos profundos". [43] Estes modelos são, então, redes neuronais profundas com várias camadas treinadas para aproximar distribuições complexas através do uso de grandes volumes de dados. Depois destes modelos serem treinados com sucesso, são usados para inferir novos dados.

Recentemente, os modelos generativos profundos têm captado bastante a atenção da comunidade científica devido às melhorias significativas nesta mesma área, estimulando, consequentemente, novas descobertas que contribuem para o crescimento deste ramo. Estes modelos, através de uma grande variedade de dados, algoritmos complexos, e técnicas de aprendizagem, são capazes de produzir conteúdo altamente realístico, tal como imagens, texto e até fala. Ao longo deste artigo iremos então explorar os algoritmos de aprendizagem relacionados com os principais modelos generativos profundos, tal como por exemplo, *Variational Autoencoders*, *Generative Adversarial Networks*, *Autoregressive Models*, etc..

2 Generative Model vs Discriminative Model

Os modelos de aprendizagem automática podem ser categorizados de acordo com diversos critérios contudo, focaremos primeiramente a nossa atenção entre modelos generativos e modelos discriminativos. Deste modo, antes de avançarmos para os modelos generativos profundos, iremos explorar o que são modelos generativos e modelos discriminativos, uma vez que a distinção entre estes conceitos permitirá um melhor entendimento sobre o foco do nosso artigo (os modelos generativos profundos - DGMs).

De uma forma mais geral, um **Modelo Generativo** procura encontrar padrões que manifestem os diferentes atributos ou categorias dentro de um determinado input. Através destes modelos, somos ainda capazes de criar novos dados. Mas, em contrapartida, revelam-se bastante frágeis perante a presença de *outliers*, que podem afetar significativamente o modelo. Exemplos de modelos generativos : *Naive Bayes*, *Linear Discriminant Analysis* (LDA) e *Hidden Markov Model* (HMMs).

Por sua vez, um **Modelo Discriminativo**, procura, consoante um conjunto de dados, distinguir a que categoria os mesmos pertencem, conseguindo expor os limites entre as diferentes classes de um *dataset*. Estes modelos, apesar de serem mais robustos à presença de *outliers* manifestam-se mais suscetíveis ao problema da classificação incorreta dos dados, por exemplo, em situações de fronteira entre duas classes alguns pontos singulares de dados podem ficar incorretamente classificados. Exemplos de modelos discriminativos : Árvores de Decisão, Regressão Logística, *Random Forests* e *Support Vector Machine* (SVM).

Em suma, um modelo generativo preocupa-se mais em explicar como os dados foram criados através da classificação dos seus atributos, ao passo que um modelo discriminativo foca-se particularmente na criação de rótulos nos dados de modo a distinguir as categorias aos quais um conjunto de dados pertence ou não. [15]

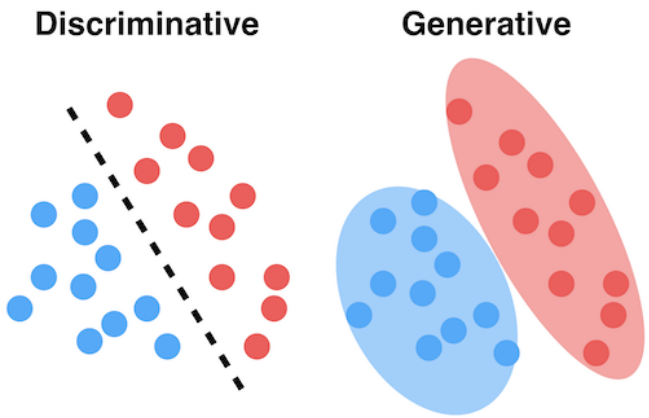

Figura 1: Comparação Visual dos Modelos. [14]

Com o intuito de consolidar as noções apresentadas, passaremos a dois breves exemplos da aplicação destes modelos no mundo real.

Exemplo 1 - ZOO :
Imaginemos que um pai leva os seus dois filhos, Gabriel e Diogo, a um zoo muito pequeno com apenas tigres e ursos. Contudo, há uma particularidade, o Gabriel apenas consegue aprender as coisas em profundidade, enquanto que o Digo consegue apenas aprender as diferenças entre o que observou.

No final da visita, o pai mostra uma imagem a ambos e questiona se o animal é um tigre ou um urso. O Diogo **conhece as diferenças** dos animais com base no que aprendeu e então afirma que é um tigre. O Gabriel desenha um tigre e um urso com base no que aprendeu, de seguida compara os seus desenhos com a imagem apresentada e chega a conclusão que a **representação mais aproximada** é um tigre, então afirma que se trata de um tigre.

Com este exemplo podemos ver que ambos chegam a resposta certa mas o processo de aprendizagem e raciocínio é diferente. Em aprendizagem automática, chamamos ao Gabriel um modelo Generativo, onde estão modelados os limites de decisão entre classes, e ao Diogo de um modelo Discriminativo, onde se encontra explicitamente modelado a distribuição real de cada classe. [13]

Exemplo 2 - SPAM :
Tendo em consideração o conteúdo de um email, um **algoritmo discriminativo** poderia classifica-lo como sendo Spam ou Não_Spam. Sendo Spam um dos rótulos, Y, e o conteúdo do email as características associadas aos dados de entrada, X. Poderíamos classificar matematicamente este problema como P(Y|X), ou por outras palavras, a probabilidade de o email ser spam dado o conteúdo do email. Deste modo, uma maneira de pensar nestes algoritmos é que mapeiam as características, X, aos rótulos, Y, sendo a sua correlação a sua principal preocupação.

No caso dos **algoritmos generativos**, acontece sensivelmente o oposto. O objetivo passa por, assumindo que o email é do tipo Spam, Y, perceber quais são

os atributos e características, X, que fazem o email ser associado ao tipo Spam. Ou seja, o foco destes algoritmos prende-se a como chegar a X, as características, a partir de Y, o rótulo ou categoria. [12]

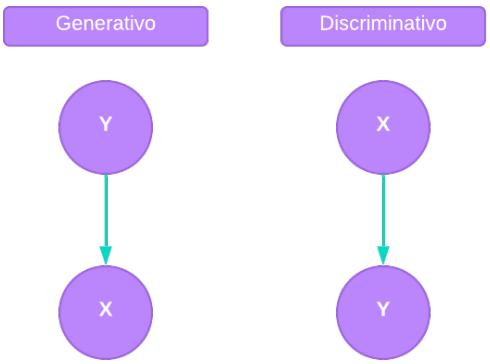

Figura 2: Modelos : Generativo Vs Discriminativo. [10]

A imagem acima ilustra a principal diferença dos modelos generativo e discriminativo. Representando os círculos as variáveis e as setas apontam para as probabilidades a inferir. No caso da classificação de spam, podemos ter X como sendo as palavras nos e-mails e Y sendo desconhecido. À direita (modelo discriminativo), a orientação segue de X para Y, indicando que podemos inferir P(Y|X) diretamente a partir do X fornecido. À esquerda (modelo gerador), ocorre a direção oposta, o que significa que precisamos inferir os valores de P(Y) e P(X|Y) dos dados primeiro e, em seguida, usá-los para calcular o nosso objetivo, P(Y|X). [10]

Com este último exemplo, exploramos um pouco da vertente matemática associada a cada um dos modelos. De seguida, iremos finalizar este assunto culminando assim no entendimento do seu lado mais científico. Relativamente aos modelos generativos, usam o Teorema de Bayes para encontrar a probabilidade condicional P(Y|X), recorrem ainda a conceitos de probabilidade conjunta e dependem dos dados de treino para estimar a probabilidade P(Y) e a probabilidade de P(X|Y). Já os modelos discriminativos assumem diretamente a probabilidade condicional P(Y|X) com base nos dados de treino fornecidos.

3 Deep Generative Models

3.1 Definição e Caracterização

Tendo um melhor conhecimento acerca das diferenças entre modelo generativo e discriminativo, nesta secção será dado foco apenas no primeiro, especialmente nos modelos generativos profundos.

Os **Modelos Generativos Profundos (DGM)** são redes neuronais, com varias camadas escondidas, treinadas para aproximar distribuições probabilísticas complicadas de grandes dimensões, utilizando um número significativo de amostras. [3]

De um modo geral, qualquer modelo generativo tem como principal objetivo aprender a distribuição dos dados de treino, e a partir dela gerar novos dados com algumas variações. Desta forma, visto que inicialmente nem sempre era possível aprender a distribuição exata dos dados, era recorrente modelar uma distribuição que fosse o mais aproximado à realidade.

Consequentemente, com o aparecimento e evolução das redes neuronais nasceram assim os **DGM** que, aproveitando todo o poder deste tipo de aprendizagem, permitiu assim obter uma distribuição praticamente exata dos dados de treino. Desta forma, tornou-se possível obter resultados muito mais realistas baseados nos dados fornecidos, como por exemplo os *deep fakes*.

3.2 Exemplos de Modelos

Os modelos generativos profundos podem ser classificados como pertencendo a um dos cinco grupos principais, sendo eles os seguintes:

- Autoregressive generative models
- Flow-based models
- Latent variable models
- Energy-based models
- Score-based models

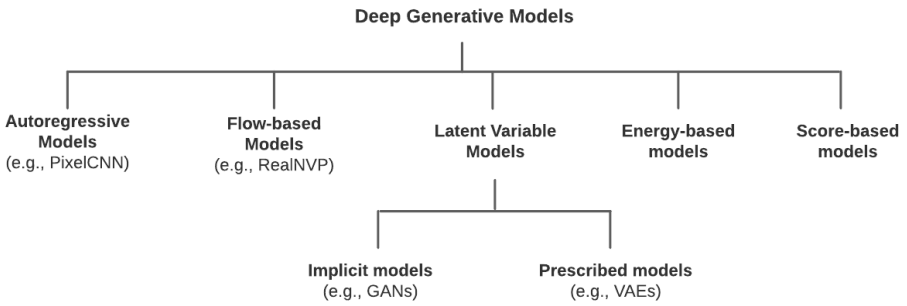

Figura 3 : Deep Generative Models [33]

Neste tópico iremos abordar vários exemplos de modelos, explicando como é que eles funcionam e como é que poderão ser aplicados.

Autoregressive Models

Para começar o nosso estudo acerca dos vários modelos generativos profundos, iremos abordar, em primeiro lugar, os **Modelos Autorregressivos**.

Na generalidade, **modelos autorregressivos** são redes que utilizam dados de uma sequência para prever um valor futuro dessa mesma sequência. Neste caso estamos perante uma abordagem de aprendizagem supervisionada, ou seja, as redes são treinadas a partir de resultados pré-definidos, o que permite avaliar o sucesso das mesmas.

Desta forma, num contexto mais específico na área de *deep learning*, este modelo assenta na utilização de redes neuronais com a particularidade de os resultados obtidos não dependerem apenas dos dados de *input* fornecidos, mas também de *outputs* obtidos em iterações anteriores. É de notar que estamos perante redes *feedforward*, isto é, cada camada está conectada à próxima, no entanto não existe conexão no sentido contrário.

Como se pode observar na figura 4, na iteração n, dados os valores de input **I1, I2, ... , I5**, obtém-se como resultado o valor **O1**. Progredindo para a próxima iteração, é possível observar que o valor obtido anteriormente (**O1**), passou a estar incluído nos dados de input atuais. Do mesmo modo, visto que a janela de *inputs* deve permanecer constante, o valor **I1** deixou de estar presente na mesma. Assim sendo, será então calculado um novo *output* baseado nos novos dados fornecidos.

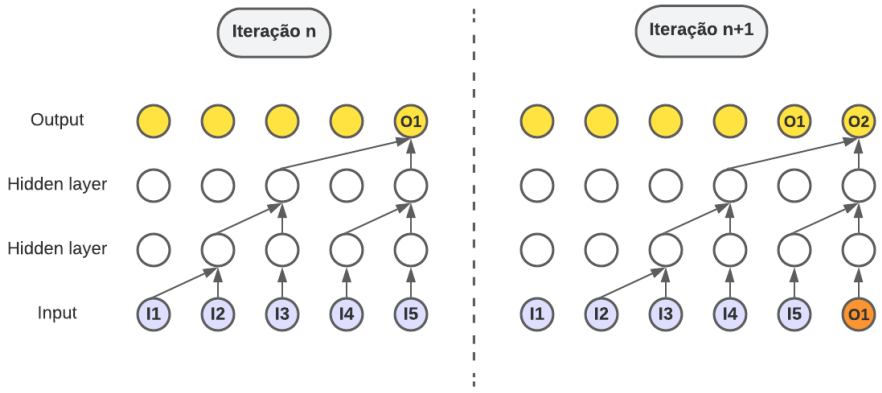

Figura 4: Exemplo de uma rede autorregressiva. [1]

A partir da análise deste exemplo podemos concluir que, a partir de um certo momento, todos os dados de input da rede serão valores de resultados obtidos pela mesma rede em iterações anteriores. Esta caraterística reforça o facto destes modelos serem generativos, uma vez que, desta forma serão capazes de gerar novos dados com variações, relativamente aos que foram fornecidos inicialmente à rede.

Generative Adversarial Networks (GANs)

Em 2014, *Ian Goodfellow* e outros investigadores da universidade de Montreal, apresentaram uma nova abordagem de modelos generativos baseada em *deep learning*, tendo-a denominado *Generative Adversarial Networks* (*GANs*). A arquitetura desta abordagem consiste em duas redes neuronais que atuam uma contra outra, derivando dai o nome *adversarial*, ou adversárias em tradução livre. Num modelo *GAN*, uma das redes neuronais usadas é, por convenção, denominada de *Generator* enquanto que a outra é chamada de *Discriminator*. Tendo isto em conta, iremos agora explicar como uma G*enerative Adversarial Network* funciona.

Como já se referiu, num modelo generativo, o objetivo é criar dados sintéticos que tenham significado num dado domínio, ou seja, pretende-se, por exemplo, que o modelo gere imagens de rostos humanos, áudio, excertos de texto, entre outros. Ora, numa *GAN*, o *Generator* é a parte da arquitetura responsável pela criação dos dados sintéticos. Esta rede neuronal, recebe como *input* um *noise vector* que nada mais é do que um vetor cujos valores são obtidos aleatoriamente a partir de uma dada distribuição como por exemplo a gaussiana. Desta forma, a finalidade do *Generator* é criar uma amostra o mais plausível possível ou até mesmo indistinguível da realidade. Por outro lado, o objetivo do *Discriminator* consiste em avaliar a autenticidade de uma amostra que recebe como *input*, ou seja, determinar se uma dada imagem , texto ou áudio é real ou se foi criada pelo *Generator*. Na figura X é possível observar a exemplificação de uma *Generative Adversarial Network*.

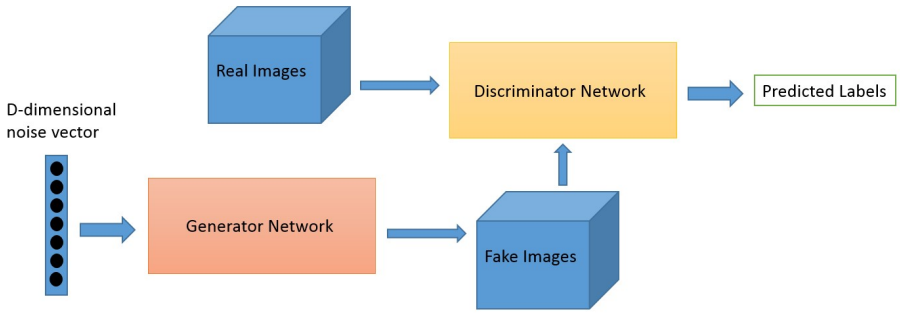

Figura 5: Arquitetura das *GANs*. [2]

Após se ter definido a arquitetura da *GAN*, passemos ao processo de treino de cada uma das redes neuronais referidas. Por norma, começa-se por treinar o *Discriminator*. Este modelo recebe dois tipos de *input* sendo eles, uma amostra produzida pelo *Generator* e uma amostra do mundo real. Ora, o processo de treino pretende que o *Discriminator* seja capaz de classificar corretamente a amostra do *Generator* como falsa e a amostra do mundo real como autêntica. Cada vez que este modelo erra na classificação, recorre-se ao método de *back-propagation* para ajustar os pesos da rede. Este processo é repetido por n épocas (*epochs*).

Estando o modelo discriminativo treinado, passamos para o modelo generativo. O processo de treino do *Generator* basea-se no *feedback* obtido pelo *Discriminator*. Isto significa que o treino do *Generator* consiste em produzir uma imagem ou texto (ou outro domínio qualquer) falso e passá-lo como *input* ao *Discriminator*, sendo que este irá classificá-los como autêntico ou não. Se o modelo generativo não for capaz de "enganar" o discriminativo então, os pesos da sua rede (do *Generator*) terão de ser ajustados usando o método de *back-propagation* e o processo de treino é repetido gerando-se um novo *noise vector*. Por outro lado, se o *Generator* conseguir produzir amostras que o *Discriminator* interpreta com verdadeiras, então é o modelo discriminativo que terá que ser reajustado, repetindo-se assim o seu processo de treino referido anteriormente.

Assim sendo, e como é possível entender, durante estes treinos, o *Generator* torna-se cada vez melhor a produzir amostras realistas e o *Discriminator* torna-se cada vez melhor a diferenciar o que é real do que é sintético. Desta forma, ambos os modelos vão evoluindo um contra o outro, até ao ponto em que o *Discriminator* se torna incapaz de distinguir as amostras geradas,das amostras do mundo real.

É interessante salientar que as *GANs* são muito utilizadas no processo de melhoramento ou criação de imagens, sendo que as redes neuronais usadas na arquitetura costumam ser as redes neurais convolucionais.

Variational Autoencoders (VAEs)

Os VAEs foram primeiramente introduzidos em 2013 por dois investigadores da Universidade de Amesterdão, Diederik P. Kingma e Max Welling. Basicamente, trata-se de uma junção de duas áreas de conhecimento, uma que envolve redes neuronais profundas e outra de modelos probabilísticos [40]

Tal como visto anteriormente, os modelos generativos profundos permitem a obtenção de conteúdos cada vez mais realísticos. A figura seguinte (Fig.6) é um exemplo de como é possível gerar imagens de pessoas fictícias através do modelo de *Variational Autoencoder*. [41,45]

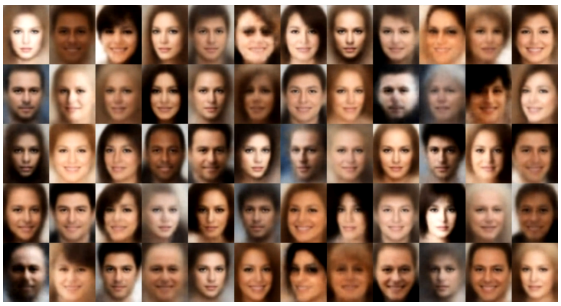

Figura 6: Imagens geradas com um Variational Autoencoder. [45]

Em primeiro lugar, antes de abordarmos o conceito de *Variational Autoencoders*, é necessário entender o funcionamento da versão mais simplificada deste modelo. Neste caso, trata-se do modelo de *Autoencoders* (AEs), cujo objetivo

é criar uma reconstrução dos dados de input, através do uso de uma rede neuronal com várias camadas (Fig.7), tentando sempre minimizar a diferença entre o input e o output.

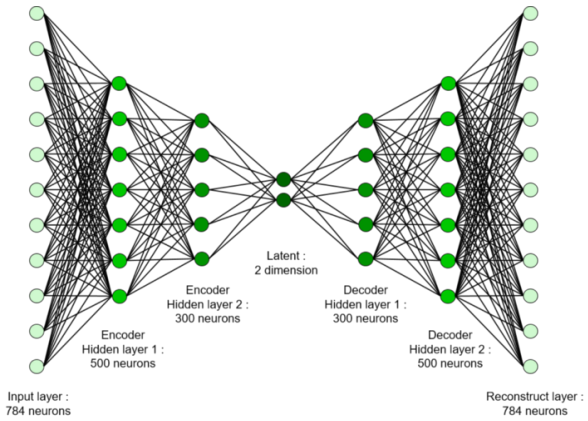

Figura 7: Representação gráfica da rede de um *Autoencoder* [37]

Um *Autoencoder* (Fig.8) é composto por duas partes principais: *Encoder* e *Decorder*. O Encoder é responsável por comprimir os dados de input de modo a obter uma representação mais pequena destes dados, a qual se dá o nome de *bottleneck* ou *latent space representation*. De seguida, o Decoder irá reconstruir os dados comprimidos do bottleneck, até obter a dimesão original dos mesmos.

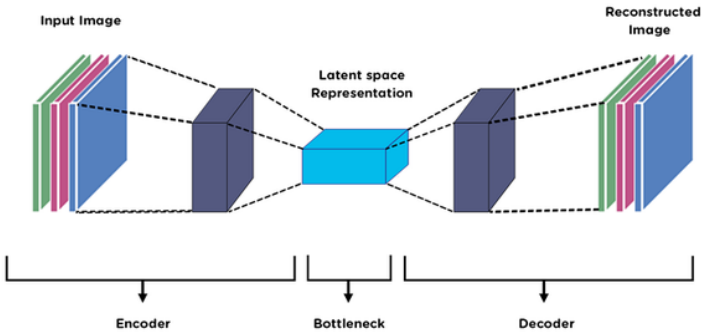

Figura 8: Arquitetura de um *Autoencoder* [42]

Por último, quando é obtida a versão reconstruída dos dados originais, esta é computada novamente, de forma a que no final seja possível comparar as diferenças, pixel a pixel, entre o output e o input. O erro obtido a partir do cálculo desta diferença, também denominado por *reconstruction loss*, é retropropagado pela rede (método de *back-propagation*), atualizando assim os pesos da mesma. [42] Portanto, a cada iteração o autoencoder é cada vez mais otimizado, uma vez que o bottleneck seleciona a estrutura principal dos dados e

garante a propagação da informação mais relevante dos mesmos, de modo a que possam ser reconstruídos com um grau de maior fidelidade. Outro aspeto crucial dos autoencoders trata-se da dimensão do espaço latente/bottleneck, quando a dimensão deste bottleneck é pequena, mais relevantes serão as características que o autoencoder irá aprender sobre os dados. Caso contrário, o autoencoder irá apenas aprender a copiar os dados de input sem reter nenhuma informação essencial sobre os mesmos.

Assim sendo, o objetivo é descobrir qual o melhor par de encoder/decoder (e qual a melhor dimensão de bottleneck) que consegue manter o máximo de informação possível no processo de compressão (encoding) mas que tenha o menor erro possível de reconstrução (decoding).

Posto isto, iremos agora abordar os modelos de ***Variational Autoencoders***. Tal como foi visto, para obter os melhores resultados possíveis é necessário encontrar uma dimensão adequada para o bottleneck e que seja regular ao longo das iterações. Uma solução para esse problema seria introduzir a variação dessa dimensão à medida que o autoencoder é treinado. Deste raciocínio surgem os *Variational Autoencoders*, que podem ser definidos como sendo autoencoders cujo treino é regularizado de modo a evitar que o modelo apenas aprenda a copiar os dados sem reter nenhuma informação útil (*overfitting*). [36]

A principal diferença entre os VAEs e os AEs, é que em vez dos dados de entrada serem mapeados por um único vetor fixo, tal como acontece nos Autoencoders (Fig.9 - b), estes são mapeados como uma distribuição ao longo do espaço latente. [38]

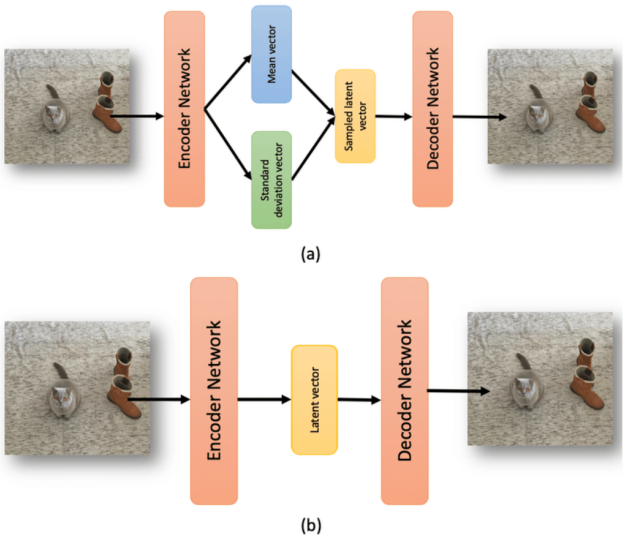

Figura 9: (a)- Arquitetura de um VAE com a representação de dois vetores. distintos. (b)- Arquitetura de um AE com a representação de um vetor fixo do espaço latente. [39]

Portanto, em vez de termos um só vetor a representar o bottleneck, temos dois vetores separados: um que representa a média da distribuição e outro que representa o desvio padrão da mesma, tal como é possível observar na Fig.9 - a). Depois dos dados atravessarem o bottleneck, é selecionado um ponto do espaço latente que é posteriormente definido como ponto de amostragem. Este irá percorrer o decoder, de modo que o erro de reconstrução possa ser calculado e, depois, retropropagado pela rede.

Tal como foi explicado, os VAEs codificam os dados de input como distribuições, em vez de pontos simples como os Autoencoders fazem. Este facto sozinho não basta para reduzir o erro de reconstrução, daí estes modelos serem treinados através da função de perda, *loss function* (2). Esta é representada por dois termos (1):

$$Loss = Reconstruction\ Loss + Regularizer \tag{1}$$

$$Loss = E_{q_\theta(z|x)}\left[\, log p_\phi(x|z)\,\right] - D_{KL}(\, q_\theta(z|x) \parallel p(z)) \tag{2}$$

As distribuições devolvidas pelo encoder devem respeitar o princípio de continuidade, isto é, que dois pontos no espaço latente não podem gerar dois conteúdos completamente diferentes quando reconstruídos, e, ainda, o princípio de completude, que diz que um ponto de amostragem escolhido deve ter conteúdo útil quando reconstruído (decoded). O segundo termo da função de perda (2), é o que permite regularizar a distribuição quando estes princípios não são cumpridos, uma vez que, obriga que as distribuições sejam próximas de uma distribuição normal padrão. Deste modo, é possível cumprir o objetivo do modelo: Minimizar o erro de reconstrução, obtendo o máximo de informação possível.

Normalizing Flow Models (NF)

Quando falamos em ***Flow Based Models*** ou ***Normalizing Flow Models***, referimo-nos a modelos onde ocorre a modelação de uma distribuição de probabilidade considerando o melhor fluxo de normalização. Por **normalização**, entendamos que se refere à mudança de variáveis conforme uma densidade de probabilidade normalizada após aplicar uma transformação invertível. Por conseguinte, **fluxo** refere-se ao processo pelo qual transformações revertíveis (bijetivas) possam ser compostas de modo a formar uma nova transformação, também esta reversível, mais complexa. [17] Deste modo, podemos usar a normalização de fluxos para transformar distribuições simples (como gaussianas) em distribuições mais ricas e complexas que poderam ser usadas para modelos generativos.

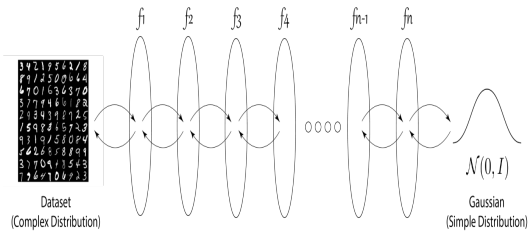

Figura 10: Transformação Fluxos [19]

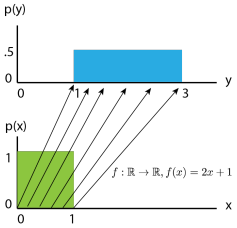

Figura 11: Mudança Variáveis [20]

Em adição, este modelo necessita de seguir requisitos estruturais específicos, entre eles : as dimensões de *input* e *output* devem ser iguais; todas as transformações tem de ser invertíveis; e o calculo do determinante do Jacobiano precisa de ser eficiente e diferenciável (por norma assumindo a forma triangular inferior). Alguns exemplos populares, que satisfazem estes requisitos, são : o *Planar Flow*, o modelo NICE (*Nonlinear Independent Components Estimation*), o modelo RealNVP (*Real Non-Volume Preserving*) e o modelo *GLOW*. [17] Devido a abordagem ligeira deste artigo, não iremos explorar em profundidade estes exemplos mencionados, contudo o leitor poderá explorar mais detalhadamente estes tópicos através dos artigos presentes na bibliografia, caso assim o deseje.

Comparativamente aos GANs e VAEs, estes modelos apresentam as seguintes vantagens : não necessitam de introduzir ruído no *output* conseguindo assim modelos mais poderosos de variância local; o seu processo de treino é mais estável comparativamente as GANs; são mais simples de convergirem e aprendem explicitamente a distribuição de dados e, portanto, a função de perda é simplesmente a probabilidade de log negativo. [18] Em contrapartida, apresentam as seguintes desvantagens : comparativamente aos outros modelos as amostras geradas não são tão boas; e de modo a preservar a bijetividade do modelo ao longo das transformções é necessário criar um espaço de maior grau, muitas vezes mais difícil de interpretar. [19]

Energy Based Models (EBMs)

Este modelo traz uma nova perspetiva aos modelos generativos, baseando-se em conceitos oriundos da área da Física. O principal objetivo dos *Energy Based Models* é encontrar uma distribuição que se adeque aos dados através da definição de uma função de energia (*energy function, E(x)*).

Primeiramente, é crucial revisitar o conceito de *Energy Function*. Dado um input x, e $E(x) = 0$, podemos então concluir que o modelo respondeu positivamente aos dados do input. Quando $E(x) > 0$, e apresenta valores muito altos, então dizemos que o modelo respondeu de forma negativa ao input dado.

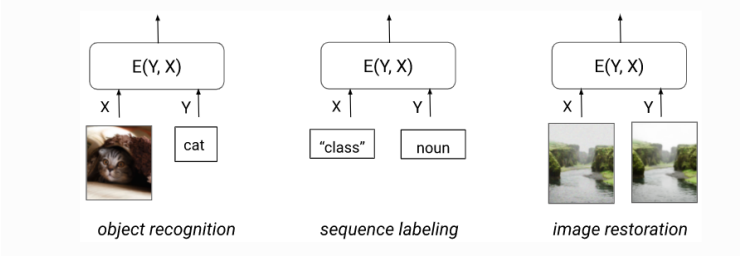

Figura 12: Aplicações de Energy Based Models [32]

Tal como está representado na Fig.10, existem vários tipos de aplicações que é possível concretizar com este modelo. Para melhor compreensão podemos explorar o seguinte caso: Considerando que o objetivo do modelo em causa é comparar uma imagem X com uma dada label Y, que na prática é uma distribuição de dados probabilística que é capaz de classificar as imagens que são dadas como input. Assim sendo, se a imagem corresponde à label, por exemplo, a imagem de um gato corresponde à categoria de *cat*, então a função *E(x)* irá retornar 0, concluindo, assim, que o modelo respondeu de forma positiva, e que a correspondência entre a imagem e a label está correta. Mas se a imagem introduzida fosse a de um cão, então a função iria devolver um valor bastante alto, demonstrando assim que a correspondência está errada.

Portanto, a ideia principal dos EBMs é formular uma função de energia adequada,e ,eventualmente, a distribuição de Boltzmann, que é definida da seguinte forma [33]:

$$p(x) = \frac{exp(-E(x))}{Z} \tag{3}$$

onde

$$Z = \sum_{x} exp(-E(x)) \tag{4}$$

Sendo Z a função de partição, esta é responsável por normalizar a função exponecial da energia, de forma a obter valores que estejam entre os valores de 0 e 1, ou seja valores probabilísticos. [33] É ainda importante referir que para minimizar a energia dos dados de input, são usadas técnicas de amostragem, isto é, compara-se os pontos de input com os pontos de amostragem, para que no final do treino do modelo se obtenha uma probabilidade alta para os dados de entrada. [32] Assim sendo, além da função de energia, é crucial calcular ou pelo menos aproximar a função de partição, de modo a conseguir obter uma distribuição *p(x)* que seja a mais próxima possível da realidade.

Score-Based Models (SGMs)

Para concluir o estudo dos modelos generativos profundos iremos abordar os ***Score-Based Models***.

Os modelos ***Score-Based*** são algoritmos onde o objetivo principal é encontrar uma função ***Score*** que será responsável por conseguir inverter o ruído inserido previamente nos dados.

Desta forma, podemos assumir que este modelo é constituído por duas etapas. A primeira consiste na introdução gradual de ruído nos dados de treino. Assim sendo, é de notar que esta distorção não será feita aleatoriamente, mas basear-se-á num determinado padrão, como por exemplo a distribuição de Gauss.

De seguida, tendo os dados de treino completamente distorcidos pelo ruído será necessário revertê-lo. Esta segunda etapa, que integra o treino da rede neuronal, consiste no cálculo da função *Score* e sua utilização para gerar novamente os dados fornecidos inicialmente. A função *Score* representa um vetor gradiente num ponto, que varia ao longo do tempo, sendo que a sua direção aponta para áreas de maior densidade de probabilidade. Assim sendo, para reverter a distorção, será estimada a respetiva função *Score* para cada distribuição de ruído, de forma gradual tal como no processo contrário.

Por fim, sabendo cada função *Score*, será possível reverter o ruído e obter a distribuição dos dados. Consequentemente, dada a distribuição será possível gerar novas amostras de dados.

Na figura abaixo pode-se observar um exemplo das fórmulas matemáticas (Equações diferenciais estocásticas), tanto para introduzir ruído, como também para revertê-lo.

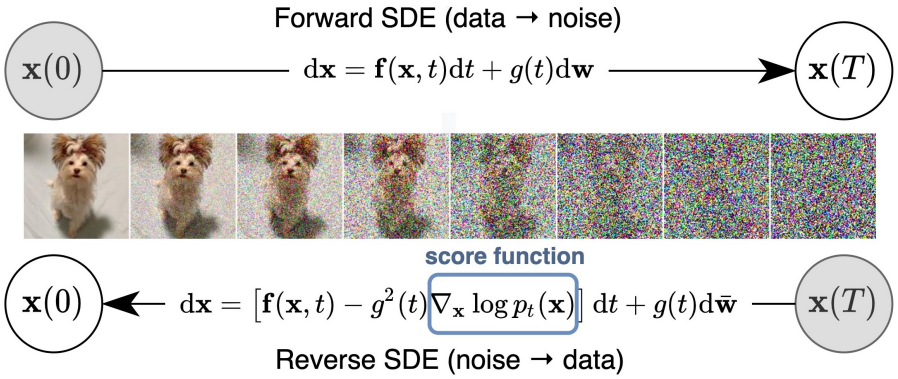

Figura 13: Modelo *Score-Based* . [7]

3.3 Casos Práticos & Aplicações no Mundo Real

A área que engloba os modelos generativos é das mais promissoras no âmbito da Inteligência Artificial. Estes modelos permitem-nos não só, analisar e melhor entender a imensa informação disponível no mundo ao nosso redor, mas também criar novos dados relativos a um determinado domínio. Ora, é inegável que a possibilidade de gerar dados sintéticos a partir de elementos pré-existente é, além de interessante, extremamente útil. Assim sendo, neste tópico irão ser apresentados alguns exemplos do uso destes modelos no mundo real.

Em 2016, no âmbito de um projeto de *Deep Learning*, *Martín Abadi* e *David Anderse*, investigadores da *Google*, implementaram um modelo generativo profundo constituído por três redes neuronais, sendo elas a *Alice*, o *Bob* e a *Eve*.

O objetivo era simples: a *Alice* teria que enviar mensagens encriptadas ao *Bob*, enquanto a *Eve* tentaria intercetar esta informação e descodificá-la. Verificou-se que após quinze mil iterações do processo de treino, a *Alice* criou o seu próprio sistema de encriptação que permitia ao *Bob* descodificar por completo as mensagens que recebia, enquanto a *Eve* não era capaz de o fazer.

Um outro exemplo prático de modelos generativos profundos,foi o *GPT 3*. Este é um modelo de linguagem autoregressivo apresentado em 2020 pela *OpenAI*,uma empresa de investigação em Inteligência Artificial. O *GPT 3* possui vários casos de uso, mas, sem dúvida, um dos mais interessantes, é a capacidade de converter um dado texto passado como *input*, em código de uma dada linguagem que executa o que foi descrito nesse texto.

Já no âmbito das artes, em 2018, leiloou-se em *Nova York* o primeiro quadro produzido usando Inteligência artificial, mais precisamente, uma *Generative Adversarial Network*. A obra foi vendida por 432.000 dólares.

Por outro lado, na área da saúde, *Insilico*, empresa de biotecnologia sediada em *Hong Kong*, tem recorrido desde 2016, a modelos generativos profundos, de forma produzir novas moléculas com determinadas propriedades capazes de combater doenças. De acordo com [29], o uso da Inteligência Artificial neste ramo, levou a uma diminuição muito significativa na duração do processo de descoberta de medicamentos, tendo reduzido de um período de anos para um período dias.

Desta forma, podemos concluir que devido às suas características, os modelos generativos profundos além de extremamente úteis e eficazes, são também fatores inovadores e até mesmo revolucionários nas mais diversas áreas do quotidiano.

4 Conclusões

Modelos generativos é dos tópicos da inteligência artificial mais estudados e que mais evoluiu nos últimos anos, principalmente devido à introdução de métodos de *deep learning*. Ao longo deste documento foram explorados alguns exemplos mais comuns de modelos generativos profundos bem como a sua aplicação e contributo em contexto real.

Apesar dos grandes benefícios que os modelos generativos oferecem, estes também levantam algumas questões éticas que se podem tornar preocupantes. Um dos exemplos mais conhecidos são os *Deepfakes*. Estes consistem na criação de imagens, vídeos ou sons recorrendo a um modelo generativo profundo, sendo que os resultados são tão plausíveis ao ponto de se tornarem indistinguíveis da realidade. Embora este seja um conceito interessante, tem sido usado de forma inapropriada, nomeadamente com a produção de notícias falsas que instigam a disseminação da desinformação, com o roubo de identidade, e com a criação de conteúdo falso que poderá ser usado para exercer chantagem sobre terceiros.

Assim sendo, podemos concluir que os modelos generativos profundos revelam-se muito eficazes e úteis em diversas áreas do quotidiano, no entanto, conduziram-nos a uma realidade na qual é cada vez mais difícil distinguir o autêntico do fictício.

References

1. Autoregressive Models in Deep Learning https://learn-neural-networks.com/wp-content/uploads/2020/04/gan.jpg Consultado em: 17.03.2022.
2. Generative adversarial networks — A Brief Survey https://www.researchgate.net/figure/Architecture-of-the-GAN-22_fig1_335199153 Consultado em: 17.03.2022.
3. Ruthotto, L and Haber, E. An Introduction to Deep Generative Modeling (2021)
4. An Intuitive Introduction to Deep Autoregressive Networks https://ml.berkeley.edu/blog/posts/AR_intro/ Consultado em: 17.03.2022.
5. Autoregressive Models in Deep Learning — A Brief Survey https://www.georgeho.org/deep-autoregressive-models/#fn:1 Consultado em: 17.03.2022.
6. Autoregressive Models — PixelCNN https://towardsdatascience.com/autoregressive-models-pixelcnn-e30734ede0c1 Consultado em: 17.03.2022.
7. Generative Modeling by Estimating Gradients of the Data Distribution https://yang-song.github.io/blog/2021/score/ Consultado em: 17.03.2022.
8. Song, Y. and Ermon S. Improved Techniques for Training Score-Based Generative Models (2020)
9. Inject Noise to Remove Noise: A Deep Dive into Score-Based Generative Modeling Techniques https://wandb.ai/ucalyptus/ScoreGM/reports/Inject-Noise-to-Remove-Noise-A-Deep-Dive-into-Score-Based-Generative-Modeling-Techniques--Vmlldzo1OTE2NDg Consultado em: 18.03.2022.
10. Modelos Gráficos Probabilísticos Generativos vs Discriminativos https://ichi.pro/pt/modelos-graficos-probabilisticos-generativos-vs-discriminativos-40857457895478 Consultado em: 18.03.2022
11. Modelo gerativo https://stringfixer.com/pt/Generative_model Consultado em: 19.03.2022
12. A Beginner's Guide to Generative Adversarial Networks (GANs) https://wiki.pathmind.com/generative-adversarial-network-gan Consultado em: 19.03.2022
13. Generative VS Discriminative Models https://medium.com/@mlengineer/generative-and-discriminative-models-af5637a66a3 Consultado em: 19.03.2022
14. Deep Understanding of Discriminative and Generative Models in Machine Learning https://www.analyticsvidhya.com/blog/2021/07/deep-understanding-of-discriminative-and-generative-models-in-machine-learning/ Consultado em: 19.03.2022
15. Generative vs. Discriminative Models in Machine Learning https://betterprogramming.pub/generative-vs-discriminative-models-d26def8fd64a Consultado em: 19.03.2022
16. Flow-based generative model https://en.wikipedia.org/wiki/Flow-based_generative_model Consultado em: 15.03.2022
17. Normalizing flow models https://deepgenerativemodels.github.io/notes/flow/ Consultado em: 17.03.2022
18. Flow-based Deep Generative Models https://lilianweng.github.io/posts/2018-10-13-flow-models/ Consultado em: 18.03.2022
19. Introduction to Normalizing Flows https://towardsdatascience.com/introduction-to-normalizing-flows-d002af262a4b Consultado em: 18.03.2022

20. Normalizing Flows Tutorial, Part 1: Distributions and Determinants https://blog.evjang.com/2018/01/nf1.html Consultado em: 20.03.2022
21. Cornish R., Caterini A., Deligiannidis G., Doucet A., Relaxing Bijectivity Constraints with Continuously Indexed Normalising Flows, (2021)
22. Bond-Taylor S., Leach A., Long Y., Willcocks C., Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models, (2022)
23. Ruthotto L., Haber E., An Introduction to Deep Generative Modeling, (2020)
24. Dinh L., Krueger D. and Bengio Y. ,NICE: NON-LINEAR INDEPENDENT COMPONENTS ESTIMATION, (2015)
25. Jørgensen P., Schmidt M. and Winther O., Deep generative models for molecular science, (2017)
26. Kingma D. and Dhariwal P., Glow: Generative Flow with Invertible 1×1 Convolutions, (2018)
27. Dinh L., Sohl-Dickstein J., Bengio S., DENSITY ESTIMATION USING REAL NVP, (2017)
28. Kingma D., Dhariwal P., Glow: Generative Flow with Invertible 1x1 Convolutions, (2018)
29. Insilico Medicine Develops and Validates Powerful AI System To Transform Drug Discovery https://www.biospace.com/article/releases/insilico-medicine-develops-and-validates-powerful-ai-system-to-transform-drug-discovery-/,
30. AI Art at Christie's Sells for $432,500 https://www.nytimes.com/2018/10/25/arts/design/ai-art-sold-christies.html, The New York Times , (2018). Consultado em: 19.03.2022
31. AI Art at Christie's Sells for $432,500 https://www.techtarget.com/searchenterpriseai/definition/GPT-3, Ronald Schmelzer, (2021). Consultado em: 14.03.2022
32. Deep Energy-Based Generative Models https://uvadlc-notebooks.readthedocs.io/en/latest/tutorial_notebooks/tutorial8/Deep_Energy_Models.html. Consultado em: 19.03.2022
33. Jakub M. Tomczak, E. Deep Generative Modeling, Springer, (2022)
34. Igor Mordatch, Concept Learning with Energy-Based Models, (2018)
35. Yann LeCun, Energy-Based Models https://atcold.github.io/pytorch-Deep-Learning/en/week07/07-1/. Consultado em: 15.03.2022
36. Understanding Variational Autoencoders (VAEs), https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73, Joseph Rocca, Towards Data Science, 2019. Consultado em: 16.03.2022
37. Extreme Rare Event Classification using Autoencoders in Keras https://processminer.com/autoencoders-in-keras/ Chitta Ranjan, ProcessMiner. Consultado em: 14.03.2022
38. Variational Autoencoders Explained, https://kvfrans.com/variational-autoencoders-explained/, Kevin Frans, (2016). Consultado em: 15.03.2022
39. García-Ordás, María Benítez-Andrades, José García, Isaías Benavides, Carmen Alaiz Moreton, Hecto, Detecting Respiratory Pathologies Using Convolutional Neural Networks and Variational Autoencoders for Unbalancing Data. Sensors. (2020)
40. Diederik P Kingma, Max Welling, Auto-Encoding Variational Bayes, (2013)
41. What is a variational autoencoder?, https://jaan.io/what-is-variational-autoencoder-vae-tutorial/, JAAN ALTOSAAR.

42. Basics of Autoencoders, https://medium.com/@birla.deepak26/autoencoders-76bb49ae6a8f, Deepak Birla, 2019.
43. Lars Ruthotto, Eldad Haber, An Introduction to Deep Generative Modeling, (2021)
44. Deep Generative Models: Algorithms and Applications, https://www.frontiersin.org/research-topics/26343/deep-generative-models-algorithms-and-applications#overview. Consultado em: 16.03.2022
45. Variational Autoencoder implementation with Tensorflow. https://github.com/wojciechmo/vae, wojciechmo