Séries Temporais e Geração de Dados Sintéticos

Benjamim Oliveira (PG42815), Gonçalo Almeida (A84610),
Lucas Mello (PG40158), Nuno Pereira (PG42846)

Departamento de Informática, Universidade do Minho

Abstract. O estudo apresentado incide sobre os fundamentos de dois tópicos, sendo eles Séries Temporais e Geração de Dados Sintéticos. Ambos os tópicos são explorados maioritariamente na ótica de *Deep Learning*, um dos ramos de *Machine Learning*. Através do estudo procura-se elucidar o leitor sobre o que representa cada um dos tópicos, procurando abordar também o funcionamento base dos mesmos, bem como possíveis aplicações no mundo real.

Keywords: Deep Learning · Séries Temporais · Geração de Dados Sintéticos.

1 Séries Temporais

Séries temporais são conjuntos de pontos de dados, distribuídos equitativamente pelo espaço e indexados por ordem temporal. Durante um período de tempo são registados valores em intervalos regulares. Não existe um tempo mínimo ou máximo para o registo de valores, permitindo assim um fornecimento mais amplo de observações, para uma análise mais rigorosa.

1.1 Dados de séries temporais

Dados de séries temporais são coleções de observações obtidas através de medições repetidas ao longo do tempo. Estes dados estão em todo o lado, uma vez que o tempo é um constituinte de tudo o que é observável e à medida que o nosso mundo se torna cada vez mais instrumentalizado, mais dados são captados para numerosas aplicações em várias indústrias. Existem várias características que tornam séries temporais diferentes de outros tipos de dados. Enquanto noutros tipos de dados os registos podem ser independentes do tempo, nas séries temporais temos o tempo como variável central da nossa análise, implicando uma causalidade entre dados obtidos em diferentes períodos. Temos, como exemplo, dados meteorológicos obtidos ao longo de um ano, no hemisfério Norte. A estação do ano (variável tempo) está ligada á temperatura média ou ao índice pluviométrico. Muitos destes dados podem ser não estacionários, isto é, a média, a variância e a frequência variam ao longo do tempo. Este tipo de características tem uma grande influência na previsão, considerando que geralmente estas métricas variam ciclicamente. Esta variação cíclica permite-nos dar mais peso á variável "tempo" no momento de fazer previsão.

1.2 Análise e Previsão Usando Séries Temporais

A análise de séries temporais é largamente aplicada em vários campos como finanças, economia, ciência e engenharia com o objetivo de prever valores futuros a partir de observações recolhidas no passado. Permite-nos ainda analisar, ao longo do tempo, as alterações registadas. Em consequência podemos visualizar padrões em situações de dependência entre pontos de dados e variáveis.

Abordagens Tradicionais (Sem Redes Neuronais Artificiais) As abordagens convencionais para tratar dados sequenciais, englobam a estimativa de parâmetros de um modelo de séries temporais como modelos auto-regressivos, sistemas dinâmicos lineares e Modelos Ocultos de Markov. Os parâmetros previstos podem ser usados como atributos de um classificador para efetuar classificações. No entanto, em dados mais complexos, com várias dimensões e com muito barulho as séries temporais com dados em tempo real, não podem ser descritas com equações analíticas, devido à sua complexidade, não sendo possível calcular um modelo com precisão. Deste modo, para criar um modelo complexo e mais preciso com dados do mundo real, podemos desenvolver variáveis que capturem informações relevantes. Mas o desenvolvimento destas variáveis para cada tarefa a avaliar de um sector em especifico é dispendioso, consome muito tempo e requer o uso de mão de obra também ela dispendiosa, através de profissionais em dados.

Redes Neuronais Artificiais Aplicadas a Séries Temporais A alternativa ao uso de métodos tradicionais, é a utilização de Redes Neuronais Artificiais, ou até mesmo de Deep Learning. O uso de aprendizagem não supervisionada pode trazer também muitos benefícios. Desta forma o algoritmo aprende e forma camadas de representação de *features* a partir de dados não rotulados. Estes dados existem em abundância e são fáceis de obter. Aprendizagem não supervisionada tem, entre outras, as seguintes vantagens:

- A aprendizagem é feita a partir de dados crus, sem rótulos.
- As camadas formadas podem ser empilhadas para criar *deep networks*, que têm um melhor desempenho na modelação de estruturas complexas de dados.
- Podem ser utilizados dados a partir do momento que são recolhidos, sem necessidade de haver intervenção humana para processamento e rotulagem dos mesmos.

Perceptrons Multi-camada São capazes de aproximar uma função de mapeamento a partir de variáveis de *input* para variáveis de *output*. Esta capacidade é particularmente útil quando aplicada a séries temporais por alguns motivos, entre eles:

- Serem capazes de lidar bem com barulho
- Não tentam encontrar uma função linear que descreva os dados

No entanto esta técnica tem também as suas limitações, uma vez que tanto os *inputs* como os *outputs* deverão ser fixos em termos de dimensão, especialmente considerando que esta informação é desconhecida muitas vezes. [5]

Redes Neuronais Convolucionais Originalmente criadas para tratar dados provenientes de imagens, são muito usadas em tarefas de *Computer Vision*. Este tipo de rede neuronal consegue bons resultados nessa área devido à sua capacidade de trabalhar com dados crus, como por exemplo os valores dos pixeis, retirados diretamente de uma imagem. Após fornecidos os dados crus originais o modelo aprende por si a extrair *features* que considere úteis. Esta capacidade é o que dá ás Redes Neuronais Convolucionais uma vantagem extra no que toca a Séries Temporais, permitindo-nos tratar a mesma como se fosse quase uma imagem de uma só dimensão. [5]

Redes Neuronais Recorrentes Algumas redes neuronais recorrentes conseguem trabalhar, explicitamente, com a ordem entre as diferentes observações, como é o caso de LTSM (*Long Short-Term Memory*). A introdução do conceito de sequência é uma nova dimensão que o modelo treinado irá ter em conta. Para além de considerar o *input* dado para calcular o *output* considera também os *inputs/outputs* anteriores, adicionando uma noção extra de sequencialidade. [5]

1.3 Aplicações

Após um período de pesquisa sobre Séries Temporais conseguimos recolher algumas aplicações apresentadas na literatura. Deste modo iremos cobrir as seguintes aplicações:

Distribuição Energética A indústria da produção/distribuição energética depara-se constantemente com um problema, não saber quanta energia uma determinada zona irá necessitar de antemão. Se carregar os cabos com demasiada energia irá ter um desperdício de energia grande, por consequência verá os seus lucros cortados, mas por outro lado, se não for distribuída energia suficiente, irão existir quedas de energia do lado dos consumidores. As empresas do ramo elétrico sabem no entanto qual o valor de energia consumida no passado nessas mesmas zonas. A partir daqui o problema passa a envolver séries temporais. É fácil concluir que há noite será gasta mais energia do que durante o dia, ou que durante o Inverno o gasto será maior, no entanto prever os valores exatos não é tão fácil, pelo menos sem o auxilio de algoritmos como Redes Neuronais Artificiais. [1, 2]

Vendas O ramo das vendas também pode fazer bom uso de Séries Temporais para melhorar o fluxo de negócio. Questões como a sazonalidade de certos produtos podem ser melhor compreendidos com séries temporais, considerando a

inevitabilidade de certos produtos serem comprados em maior volume em certas alturas do ano.

Embora certamente seja válido usar Séries Temporais para este tipo de situação, *Lukovic* [3] considera que pode também ser tratado como um problema de regressão para obter melhores resultados. [3]

Previsão Financeira Conseguir prever com alguma certeza variáveis económicas pode também ter um impacto muito positivo no momento de recolher os lucros. Considerando, por exemplo, movimentos cíclicos no mercado, como por exemplo alturas do ano em que o investimento seja maior, podem ser treinados modelos capazes de prever com alguma fiabilidade o estado financeiro do mercado em determinado ponto temporal. [4]

2 Geração de Dados Sintéticos

Existem diversas maneiras de adquirir conjuntos de dados para realizar pesquisas e desenvolvimentos mas, em diversas situações, somente o uso de dados reais não é suficiente ou mesmo possível. No processo de desenvolvimento de um modelo de *Machine Learning* (ML) vários obstáculos podem ser encontrados [12], seja a pouca quantidade de dados, a falta de dados, a presença de dados sensíveis sobre utilizadores que não são disponibilizados aos desenvolvedores por questões de privacidade, entre outros diversos exemplos. Com isto surge a necessidade de gerar dados "falsos" que consigam capturar a essência dos dados reais e contornar estes obstáculos, este tipo de dados são conhecidos por "dados sintéticos". Conjuntos de dados sintéticos desenvolvidos da maneira correta fornecem detalhes confiáveis e são alternativas baratas e escalonáveis aos dados reais [13].

2.1 Definição e História

No início da década de 1990 foram apresentados os primeiros modelos de dados sintéticos. O modelo de dados totalmente sintéticos foi proposto por Ruben [14], e um modelo de dados parcialmente sintéticos foi proposto por Little [15]. Conceitualmente, dados sintéticos não são dados reais, mas dados falsos gerados a partir de dados reais de forma que consigam capturar a sua distribuição estatística. É imprescindível que os dados sintéticos capturem bem a essência dos dados reais, de forma a que se obtenha uma maior quantidade de dados com a sua combinação, e que isso não comprometa os resultados finais de modelos de classificação ou analíticos desenvolvidos. O processo para gerar dados sintéticos é conhecido como ”synthesis” [16].

Atualmente existem diversos métodos para geração de dados sintéticos com ML utilizando árvores de decisão, modelos de regressão, *support vector machines*, entre outros. Este artigo tem o foco de apresentar modelos utilizados para a geração de dados sintéticos com *Deep Learning* (DL). DL constitui uma técnica recente e moderna para realizar o processamento de imagens e análise de dados

com grande complexidade [17]. Pertence ao campo de ML e é semelhante estruturamente e funcionalmente ao cérebro humano, sendo conhecida pelo nome redes neuronais artificiais (RNA). DL funciona como um avanço das técnicas utilizadas por ML, trazendo mais complexidade aos métodos utilizados e, consequentemente, mais profundidade na análise e na aprendizagem dos dados.

2.2 Tipos de Geração de Dados Sintéticos

Antes de escolher um método para gerar dados sintéticos é importante considerar qual o tipo de abordagem que se encaixa melhor no modelo que está a ser desenvolvido. Em geral existem duas categorias de geração de dados. Na primeira categoria são gerados dados sintéticos a partir de dados reais, e são ideais em casos que a quantidade de dados é insuficiente ou existe uma grande quantidade de dados em falta. Na segunda categoria é gerado um conjunto inteiro composto por dados sintéticos, este modelo é ideal em casos onde conseguir dados reais requer muito custo e esforço.

Dados Sintéticos a partir de Dados Reais Como o próprio nome diz esta abordagem consiste em adquirir um conjunto de dados sintéticos a partir de dados reais. Para realizar esta tarefa é necessário construir um modelo que capture a distribuição e estrutura do conjunto de dados reais (estrutura significa os relacionamentos multivariados e interações nos dados). Se o modelo construído for uma boa representação dos dados, consequentemente os dados sintéticos a partir deste modelo terão propriedades similares aos dados reais [16].

Dados Sintéticos sem Dados Reais Nesta abordagem a geração de dados sintéticos não é realizada a partir de um conjunto de dados reais, em vez disso os dados podem ser gerados a partir de modelos previamente existentes como modelos desenvolvidos através de pesquisas, recolha de dados, ou até mesmo de simulações criadas por *game engines* que são capazes de replicar e coletar praticamente qualquer tipo de espaço 3D. Existem diversos exemplos de abordagens de utilização de dados gerados em espaços tridimensionais criados por *game engines* que mostram algumas das possibilidades de atingir com esta metodologia. Alguns destes exemplos são UnrealCV [18] e SIM4CV [19].

Esta abordagem é perfeita em casos onde a recolha de dados não é algo trivial, algo comum quando se trabalha no campo da visão computacional.

2.3 Vantagens e Desvantagens dos Dados Sintéticos

Algumas das principais vantagens dos dados sintéticos são:

- `Confiabilidade`: São suficientemente semelhantes aos dados reais. Isto é conseguido através da aprendizagem das propriedades estatísticas de um conjunto de dados real;

- `Privacidade`: Os dados sintéticos não contêm quaisquer informações pessoais, tornando mais fácil a partilha de dados para melhorias dos modelos;
- `Custos`: A geração de dados sintéticos é significamente mais económica e mais eficiente do que a recolha de dados reais em vários casos como, por exemplo, na área dos veículos autónomos em que a obtenção de dados reais é um processo demoroso e custoso;
- `Quantidade`: Podem ser gerados em grandes quantidades enquanto que, com dados reais, pode até ser impossível obter os mesmos números de dados;
- `Controlo`: A recolha de dados reais, na maioria dos casos, não representa todos os comportamentos possíveis, até porque existem eventos raros, difíceis ou até perigosos de serem obtidos. A geração de dados sintéticos é completamente controlada de forma a ser possível simular facilmente estas situações.

Por outro lado, algumas das suas principais desvantagens são:

- `Dependência`: Para serem gerados são necessários, em antemão, dados reais e, dependendo do problema, em grandes quantidades. Estão também altamente dependentes do modelo que os gerou pois, se não forem quase idênticos aos dados reais, a qualidade dos modelos criados posteriormente com estes mesmos dados é comprometida;
- `Perda de Informação`: Independentemente da sua qualidade, não deixam de ser uma réplica de propriedades específicas de um conjunto de dados real. O modelo gerador procura tendências para replicar, o que implica que alguns comportamentos aleatórios possam ser perdidos;
- `Vulnerabilidade`: Caso um modelo gerador seja submetido a um ataque de inversão de modelo, o invasor pode ganhar acesso a grande parte dos dados originais e, consequentemente, obter acesso a dados privados.

2.4 Geração de Dados Sintéticos com Deep Learning

Em geral, existem dois tipos de arquiteturas RNA que são utilizadas para a geração de dados sintéticos, a primeira é conhecida por *Variational Autoencoder* (VAE) e a segunda é conhecida por *Generative Adversarial Network* (GAN).

Variational Autoencoder Um *autoencoder* possui o objetivo de realizar uma redução de dimensionalidade nos dados (semelhante ao *Principal Component Analisys*). O processo é feito através de um codificador e um decodificador. O codificador recebe uma entrada de dados e realiza um processo de compressão no espaço dos dados, retornando uma representação de espaço codificada ou "espaço latente" onde os dados semelhantes encontram-se mais próximos e dados menos semelhantes encontram-se mais afastados. Por outro lado, o decodificador tem a tarefa de descomprimir os dados. Como ocorre em qualquer método de redução de dimensionalidade, esta compressão dos dados pode não ser bem definida e, consequentemente, resultar numa grande perda de informação. Logo o principal objetivo com *autoencoders* é encontrar a combinação perfeita que mantenha o máximo de informação quando uma entrada é codificada, de maneira que no

processo de decodificação tenha um risco mínimo de erro na reconstrução dos dados. Tanto o codificador quanto o decodificador são duas redes neuronais, e para o processo para encontrar a melhor combinação é utilizando um processo iterativo de otimização. Em cada iteração são fornecidos dados à arquitetura (codificador e decodificador) e é comparada a saída de dados decodificada com os dados originais. Então, é propagado o erro pela arquitetura para atualizar os pesos das redes. Um problema na utilização de autoencoders é que ele é treinado exclusivamente para codificar e decodificar os dados com o mínimo de perda possível, não importa como o espaço latente fique organizado, portanto durante o treinamento é natural que a *neural network* tome vantagem de qualquer possibilidade de *overfitting* para completar a sua tarefa de maneira satisfatória. É necessário um modelo que realize este processo de uma maneira mais generalizada se o objetivo é utilizá-lo com propósito generativo.

Variational Autoencoders (VAE) é uma arquitetura de deep learning capaz de visualizar dados de alta dimensão num espaço de dimensão significativamente inferior e o modelo foi proposto por Diederik P. Kingma e Max Welling num artigo publicado em 2013 [20]. VAE é um autoencoder com regularizações e regras para não possuir *overfitting* e assegurar que o espaço latente possua boas propriedadade que possibilitam um processo generativo de novos dados. VAE possui a mesma arquitetura de um *autoencoder* padrão (codificador, decodificador e uma *loss function*), porém ao contrário de codificar os dados em um único ponto, é codificado a distribuição do espaço latente. Para exemplificar o processo de treino realizado por um modelo VAE suponhamos que queremos codificar uma imagem 28x28, e que esta imagem possui 784 dimensões:

- `Codificador`: o codificador irá codificar os dados da imagem numa nova representação latente x com uma dimensão muito inferior a 784. Este processo tipicamente é chamado de *bottleneck*. Esta nova representação latente dos dados é uma densidade de probabilidade gaussiana, o que torna nossos dados codificados estocásticos.

- `Decodificador`: O decodificador recebe x e produz os parâmetros para a distribuição de probabilidade dos dados, como pesos e bias. No contexto do nosso exemplo tomamos o cenário em que a imagem é preta e branca com a sua representação de pixels como 1 e 0 respectivamente. A probabilidade de cada pixel ser 1 ou 0 é feita utilizando a distribuição de Bernoulli, ou seja, o decodificador recebe x e retorna, para cada um dos 784 pixeis, uma distribuição de Bernoulli que consiste na probabilidade deste ter o valor 0 (P(pixel = 0)) e na probabilidade de ter o valor 1 (P(pixel = 1)). Neste processo existe uma perda de informação pois o decodificador recebeu uma entrada x com uma dimensão inferior à entrada original, o que implica não conseguir toda a informação dos dados originais. Para calcular a quantidade de informação perdida é utilizada uma *loss function* que mostra o grau de efetividade que o decodificador alcançou ao reconstruir a imagem com os dados comprimidos pelo codificador.

Generative Adversarial Network *Generative Adversarial Network* (GAN) é uma arquitetura de DP proposta por Ian J. Goodfellow em 2014 [21]. GAN não é o primeiro modelo proposto para a geração de dados sintéticos, mas é o modelo que elevou o patamar deste processo pela sua versatilidade e resultados em comparação aos demais. Com a utilização de GANs foi possível realizar tarefas que até então não eram possíveis, como a geração de imagens falsas com qualidade e semelhança de imagens reais. GAN é formado por dois modelos de treino simultâneos, um denominado "gerador" e outro "discriminador" com as funções de gerar dados falsos e discernir os dados falsos a partir de um conjunto de dados reais respetivamente. Para explicar o conceito por detrás do modelo *Generative Adversarial Network* podemos analisar o seu nome em partes e explicá-los. A palavra *Generative* implica gerar novos dados. Os dados que o modelo GAN irá aprender a gerar dependem do conjunto de dados reais que é disponibilizado para treino. Supondo que queremos treinar um modelo GAN para gerar imagens que se pareçam com desenhos animados devemos utilizar um conjunto de dados de treino de imagens de desenhos animados. O termo *Adversarial* representa a ideia principal deste modelo, onde o gerador é um adversário do discriminador. Enquanto o gerador procura criar dados falsos que sejam indistinguíveis do conjunto de dados treino, o discriminador tem o objetivo de fazer a distinção entre os dados falsos gerados pelo discriminador e os dados de treino. Neste processo um modelo aperfeiçoa o outro pois quanto mais precisos são os dados gerados pelo gerador mais minuciosa é a distinção feita pelo discriminador. Uma metáfora frequentemente utilizada para exemplificar este processo e que o próprio criador Ian Goodfellow utiliza muito é a seguinte: "Existe um criminoso (gerador) que falsifica dinheiro, e um detetive (discriminador) que tenta pega-lo. Quanto mais autênticas as notas falsas se tornam, mais preparado o detetive deve estar em detectá-las e vice versa" [22]. Finalmente a palavra *Network* representa a classe de modelo de *machine learning* mais utilizada para representar o discriminador e o gerador: as redes neuronais.

A tabela 1 resume a função do gerador e do discriminador.

Table 1. Função Gerador e Discriminador.

	Gerador	Discriminador
Input	Vetor de números aleatórios	O discriminador recebe o input das duas fontes: .Conjunto de dados reais para treino .Conjunto de dados falsos gerados pelo gerador
Output	Dados falsos que se esforçam para ser o mais convincentes possível	Avaliação do modelo

Exemplo de aplicação de VAE O mecanismo VAE pode ser utilizado para gerar imagens de alta qualidade. Dadas imagens de input de pessoas, objetos ou

até caracteres animados, o modelo gera novas imagens semelhantes às fornecidas. O *autoencoder* procura mapear cada imagem de input para uma distribuição normal multivariada no espaço latente e, posteriormente, transformá-la numa imagem de saída com perdas de reconstrução baixas. Uma das vantagens de utilizar este modelo é que este segue uma distribuição de probabilidades Gaussiana, o que permite alcançar a alta qualidade nas imagens resultantes.

Exemplo de aplicação de GAN O mecanismo GAN pode ser utilizado para aplicar a técnica de *Super-Resolution* que permite gerar uma imagem com alta resolução a partir de uma imagem com uma resolução inferior.

Isto é conseguido com o modelo treinado a deduzir detalhes foto-realistas durante o processo de *upsampling*. No estudo [30] os autores propuseram uma variante deste mecanismo em que, durante o treino do modelo, são aumentados progressivamente o gerador e o discriminador de forma irem sendo adicionandas novas camadas que modelam detalhes cada vez mais pormenorizados, o que acelera e estabiliza o próprio processo de treino.

References

1. Busseti, E., Osband, I. and Wong, S., 2012. Deep Learning for Time Series Modeling.
2. Gasparin, A., Lukovic, S. and Alippi, C., 2019. Deep Learning for Time Series Forecasting: The Electric Load Case. [online] Available at `https://www.researchgate.net/publication/334624519_Deep_Learning_for_Time_Series_Forecasting_The_Electric_Load_Case` [Accessed 17 March 2021].
3. Gasparin, A., Lukovic, S. and Alippi, C., 2019. Deep Learning for Time Series Forecasting: The Electric Load Case. [online] Available at: `https://www.researchgate.net/publication/334624519_Deep_Learning_for_Time_Series_Forecasting_The_Electric_Load_Case` [Accessed 17 March 2021].
4. Koenecke, A., n.d. Applying Deep Neural Networks to Financial Time Series Forecasting.
5. Jason, B., 2018. Deep Learning for Time Series Forecasting.
6. Investopedia. 2021. Understanding Time Series. [online] Available at: `https://www.investopedia.com/terms/t/timeseries.asp` [Accessed 18 March 2021].
7. Längkvist, M., Karlsson, L. and Loutf, A., 2014. A review of unsupervised feature learning and deep learning for time-series modeling. Elsevier,.
8. InfluxData. n.d. What is time series data? | Definition and Discussion | InfluxData. [online] Available at: `https://www.influxdata.com/what-is-time-series-data` [Accessed 18 March 2021].
9. Medium. n.d. Time Series Smoothing for better Forecasting. [online] Available at: `https://towardsdatascience.com/time-series-smoothing-for-better-forecasting-7fbf10428b2` [Accessed 18 March 2021].
10. GitHub. n.d. cerlymarco/tsmoothie. [online] Available at: `https://github.com/cerlymarco/tsmoothie` [Accessed 18 March 2021].
11. Cerliani, M., n.d. Time Series Clustering and Dimensionality Reduction. [online] Medium. Available at: `https://towardsdatascience.com/time-series-clustering-and-dimensionality-reduction-5b3b4e84f6a3` [Accessed 18 March 2021].

12. Popic, S., Pavkovic, B., Velikic, I., & Teslic, N. (2019). Data generators: a short survey of techniques and use cases with focus on testing.2019 IEEE 9th International Conference on Consumer Electronics (ICCE-Berlin).
13. Gupta, A., Vedaldi, A., & Zisserman, A. (2016). Synthetic Data for Text Localisation in Natural Images. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
14. D. B. Rubin, T. E. Raghunathan, J. P. Reiter. (1993). Discussion: Statistical disclosure limitation. Journal of Official Statistics 9, 462–468.
15. R. J. A. Little (1993). Statistical analysis of masked data. Journal of Official Statistics 9, 407–426.
16. Emam, K. E., Mosquera, L., & Hoptroff, R. (2020). Practical Synthetic Data Generation: Balancing Privacy and the Broad Availability of Data (1st ed.). O'Reilly Media.
17. Schmidhuber, Juergen. (2014). Deep Learning in Neural Networks: An Overview. Neural Networks.
18. Qiu, Weichao & Yuille, Alan. (2016). UnrealCV: Connecting Computer Vision to Unreal Engine.
19. Mueller, Matthias & Casser, Vincent & Lahoud, Jean & Smith, Neil & Ghanem, Bernard. (2018). Sim4CV: A Photo-Realistic Simulator for Computer Vision Applications. International Journal of Computer Vision.
20. Kingma, Diederik & Welling, Max. (2014). Auto-Encoding Variational Bayes. 2nd International Conference on Learning Representations (ICLR2014)
21. Goodfellow, Ian & Pouget-Abadie, Jean & Mirza, Mehdi & Xu, Bing & Warde-Farley, David & Ozair, Sherjil & Courville, Aaron & Bengio, Y.(2014). Generative Adversarial Nets.
22. Langr, J., & Bok, V. (2019). GANs in Action: Deep learning with Generative Adversarial Networks (1st ed.). Manning Publications.
23. Foster, D. (2019). Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play (1st ed.). O'Reilly Media.
24. towardsdatascience. n.d. Understanding Variational Autoencoders (VAEs). [online] Available at: https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73 [Accessed 17 March 2021].
25. jaan. n.d. Tutorial - What is a variational autoencoder ?. [online] Available at: https://jaan.io/what-is-variational-autoencoder-vae-tutorial/ [Accessed 18 March 2021].
26. Bernard Marr. (2018). Does Synthetic Data Hold The Secret To Artificial Intelligence? [online] Available at: https://www.forbes.com/sites/bernardmarr/2018/11/05/does-synthetic-data-hold-the-secret-to-artificial-intelligence/?sh=50d7c74c42f8 [Accessed 17 March 2021]
27. Laboratory for Information and Decision Systems, Massachusetts Institute of Technology. (2020). The real promise of synthetic data [online] Available at: https://news.mit.edu/2020/real-promise-synthetic-data-1016 [Accessed 17 March 2021]
28. Roger Grosse, Jimmy Ba. (2019). CSC421/2516 Lecture 17: Variational Autoencoders. [online] Available at: http://www.cs.toronto.edu/~rgrosse/courses/csc421_2019/slides/lec17.pdf [Accessed 18 March 2021]
29. Hamed Alqahtani, Manolya Kavakli-Thorne, Gulshan Kumar Ahuja. (2019). Applications of Generative Adversarial Networks (GANs): An Updated Review.
30. Tero Karras, Timo Aila, Samuli Laine, Jaakko Lehtinen. (2018). Progressive Growing of GANs for Improved Quality, Stability, and Variation.