Detecção de Anomalias Estruturais em Barragens Utilizando uma Arquitetura Autoencoder

Alexsander Küster Milczarek[1], **Carine Geltrudes Webber** [1]

[1] Área do Conhecimento de Ciências Exatas e Engenharias
Universidade de Caxias do Sul (UCS) – Caxias do Sul, RS – Brasil

akmilczarek@ucs.br, cgwebber@ucs.br

Abstract. *Safety in dams is an extremely important issue, since accidents involving such constructions can cause great damage to the population and the environment. To ensure the safety of these structures, it is necessary to identify possible cracks and defects, which can lead to ruptures and fractures in the concrete. These defects are known as anomalies and can occur in any environment, from nature to computer chips. In order to guarantee such safety, this study was elaborated: initially, a literature review was carried out, in which the neural networks autoencoders stood out in the solution of such problems. Autoencoders are generative networks capable of reconstructing images, based on a dataset. For training purposes, this architecture was applied, obtaining results similar to those of the works analyzed. Thus, it is clear that such a model could be used to identify cracks, serving as a tool in the prevention of environmental disasters involving dams.*

Resumo. *Segurança em barragens é um tema de extrema importância, já que acidentes envolvendo tais construções podem causar grandes danos à população e ao meio ambiente. Para garantir a segurança destas estruturas é necessário identificar possíveis rachaduras e defeitos, que podem levar a rompimentos e fraturas no concreto. Esses defeitos são conhecidos como anomalias e podem ocorrer em qualquer ambiente, desde na natureza até em chips de computadores. Visando garantir tal segurança, foi elaborado este estudo: inicialmente, foi feita uma revisão da literatura, em que as redes neurais autoencoders se destacaram na solução de tais problemas. Autoencoders são redes generativas capazes de reconstruir imagens, utilizando como base um conjunto de dados. Para fins de treinamento foi aplicada essa arquitetura, obtendo resultados semelhantes aos dos trabalhos analisados. Dessa forma, percebe-se que tal modelo poderia ser utilizado para identificação de rachaduras, servindo como uma ferramenta na prevenção de desastres ambientais envolvendo barragens.*

1. Introdução

A sociedade está sempre visando evoluir e melhorar a qualidade de vida humana a fim de proporcionar conforto, longevidade e saúde para as pessoas. Nesse ambiente, surgem as tecnologias. Responsáveis por aprimorar tarefas, elas se tornam cada vez mais especializadas a fim de alcançar tais objetivos, seja na área da saúde, da alimentação, do transporte, do entretenimento, para não dizer em todas. Com o avanço tecnológico, muito se pode melhorar na vida das pessoas, como cirurgias menos invasivas e aparelhos de

comunicação, que levam a equipamentos e programas cada vez mais complexos e especializados. Nesse contexto, emergem as ferramentas para apoiar o desenvolvimento de produtos utilizando métodos e técnicas de Inteligência Artificial (IA).

Como o nome sugere, a IA é uma área da computação responsável por programas que possuem a capacidade de aprender e raciocinar como humanos [Santana 2018]. Isso se torna muito relevante quando se trata de problemas complexos como reconhecimento de fala e rostos, predições de gastos ou detecção de fraudes. Em um mundo cada vez mais conectado e digital, algoritmos que consigam aprender e realizar tarefas complexas, antes realizadas apenas por pessoas, fazem com que elas se tornem mais rápidas e eficientes, muitas vezes evitando falhas humanas [Stahl 2022].

Na área de IA, existem sistemas que realizam previsões a partir de dados que descrevem situações passadas. Tais modelos, denominados preditivos, lidam com exemplos e amostras de ocorrências observadas e coletadas. Em alguns casos, o volume de ocorrências é muito baixo, inviabilizando o desenvolvimento de tais modelos [Tchilian 2022, Luger 2013]. Nestes casos, sistemas denominados detectores de anomalias atuam para reconhecer situações de falhas excepcionais e raras, cujos registros são normalmente em menor número [Chandola et al. 2009].

O reconhecimento de anomalias é extremamente importante, já que elas podem acontecer em diversos lugares, sejam em conjuntos de dados, em forma de ruídos de música, até mesmo em forma de fraudes bancárias. Por mais simples que aparentem, anomalias não são vistas com bons olhos, já que uma anomalia em um aparelho cirúrgico, por exemplo, pode comprometer a saúde de um paciente. Detectar essas falhas se torna relevante para prevenir problemas, sejam para empresas ou pessoas [Chandola et al. 2009].

Com isso em vista, este trabalho visa estudar e avaliar algoritmos de IA para detectar anomalias em barragens, tais como rachaduras e falhas no concreto. Barragens são construções artificiais que servem de barreira para reter e acumular grandes quantidades de líquido, podendo servir como reserva de água, controle de cheias, para geração de energia elétrica e contenção de resíduos, sejam industriais ou em decorrência da erosão [ETESCO 2021]. Identificar anomalias nestas construções tem um papel fundamental para garantir a segurança das pessoas e também da fauna e flora local [Tainara Messias dos Santos 2019].

Anomalias podem comprometer a vida útil da estrutura, como no caso das rachaduras na barragem de Mauá, ilustradas na Figura 1 [Antonelli 2011]. Um dos acidentes mais conhecidos ocorreu em 2015, com a Tragédia de Mariana, em que a barragem da mineradora Samarco rompeu e despejou em torno de 40 bilhões de litros de rejeitos de minério sobre os distritos em volta, causando não só a morte de 19 pessoas como desabrigando as famílias que ali viviam, causando uma catástrofe de poluição ambiental [Globo 2021]. De acordo com a ANA (Agência Nacional das Águas) indica que são quase 200 barragens que estão em situação de risco no Brasil, que podem provocar desde pequenos acidentes até comprometimento total de estrutura [Lin 2022].

Dessa forma, identificar anomalias, como rachaduras, é uma maneira de auxiliar na prevenção de acidentes que possam vir a ocorrer em barragens, garantindo segurança para as pessoas nas redondezas e também ao meio ambiente. Neste contexto, este trabalho visa desenvolver e avaliar um modelo de arquitetura *autoencoder* que analise imagens de

Figura 1. Rachadura em barragem de Mauá

rachaduras em concreto e classifique-as quanto a presença de anomalias. Este modelo pode estar inserido em um sistema que informe aos responsáveis pela segurança, sobre tais estados, como forma de auxiliar na identificação de problemas.

O artigo está organizado em 6 seções, sendo que a primeira descreve o problema de detecção de anomalias. A seção 2 apresenta a revisão sistemática da literatura, que permitiu o mapeamento do estado da arte na área. A seção 3 descreve a arquitetura baseada em *autoencoders*. As seções 4 e 5 detalham o método seguido e o desenvolvimento da proposta. Por fim, a seção 6 apresenta as considerações finais e trabalhos futuros.

2. Detecção de Anomalias

A detecção de anomalias consiste em encontrar elementos que são diferentes dos demais, que não se encaixam no padrão e que são considerados falhas. No mundo real, a detecção de anomalias pode ser aplicada desde na detecção de fraudes, de intrusos em redes de comunicação até na detecção de produtos defeituosos em uma linha de produção [Chandola et al. 2009]. Uma empresa que detecta previamente anomalias pode economizar muito dinheiro em produtos finais que acabariam tendo que ser devolvidos pelos clientes, gerando despesas em transporte, e prejudicando a confiabilidade da marca.

Anomalias são amostras de um determinado conjunto de dados que não obedecem o padrão dos demais elementos. Como exemplo, considera-se o seguinte conjunto de dados, indicando os números pares: 2, 4, 6, 7, 8. Sabe-se que para um número ser considerado par precisa ser múltiplo de 2. Uma anomalia nesse conjunto seria a ocorrência do número 7, que não é divisível por 2. Mas anomalias não são referentes apenas à sistemas numéricos, e sim a todos tipos de elementos em que um não se encaixam em um padrão.

A Figura 2 [Mathworks 2022] traz um exemplo de produtos com e sem anomalias. A foto da esquerda apresenta uma pílula de remédio normal, enquanto as outras duas da direita mostram anomalias de produto, sejam de textura ou estrutura. Enquanto a falha na imagem do meio pode não parecer muito significativa, a da extrema direita mostra uma situação relevante: um paciente precisa tomar a dosagem correta de um medicamento, não podendo consumir um produto em tal estado defeituoso.

Figura 2. Exemplo de Pílula Normal e com Anomalias

Dessa forma, anomalias podem ser de inofensivas até extremamente prejudiciais, variando desde pequenos ruídos em músicas e até grandes falhas em fabricação de produtos. Identificar anomalias é portanto uma tarefa muito relevante, justamente por prevenir que pequenos erros se tornem situações graves.

A dificuldade em se encontrar um elemento anômalo se deve ao fato de que as anomalias podem ocorrer em qualquer lugar. Anomalias são elementos difíceis de encontrar, muitas vezes passando desapercebidos, o que requer técnicas especiais para poder identificá-las. Uma dessas técnicas consiste no uso de modelos preditivos. Um modelo preditivo pode ser visto como uma função matemática aplicada aos dados existentes, fazendo com que sejam identificados padrões nos valores e tornando possível prever um comportamento futuro [Tchilian 2022]. Conhecendo tal comportamento, torna-se possível identificar comportamentos anormais, tais como falhas, discriminando-as do comportamento normal [André Luis da Cunha Dantas Lima 2019].

Um cenário de aplicação da detecção de falhas é na manutenção preditiva. Como o nome sugere, o método consiste em se ter monitoramento constante do objeto ou elemento que se deseja antecipar anomalias [André Luis da Cunha Dantas Lima 2019]. Com o monitoramento, fica fácil identificar algum elemento não usual no produto, podendo-se tomar as ações necessárias para eliminá-lo. Contudo, tais métodos estão sujeitos a falha, seja por uma predição não acurada ou até falha humana em identificar um produto defeituoso. Dessa forma, outra alternativa consiste no uso métodos de IA que podem fazer tanto o monitoramento constante, como a detecção em tempo real de anomalias.

3. Revisão Sistemática

Para iniciar um trabalho de pesquisa sobre detecção de anomalias, percebeu-se a necessidade do uso de *deep learning* e *transfer learning*. *Deep learning* é uma subárea da Inteligência Artificial que utiliza redes neurais artificiais de múltiplas camadas para fazer representações complexas de dados[Luger 2013]. Já *transfer learning* é a aplicação de

conhecimentos prévios adquiridos por um modelo treinado em uma tarefa para ajudar na resolução de uma tarefa relacionada. Não sendo foco do vigente trabalho, tais conceitos não serão aprofundados[Fuchs et al. 2021].

Com esse propósito, foi feita uma busca no portal científico *Science Direct* [1] por artigos com relevância, dado o objetivo da pesquisa. Com o intuito de se obter uma busca objetiva e atualizada, foram utilizados os seguintes filtros:

- Termos de pesquisa: *anomaly detection deep learning transfer learning*;
- Tipo de pesquisa: *Research articles*;
- Área de pesquisa: *Computer Science*;
- Anos: 2021, 2022, 2023.

A busca gerou um total de 880 artigos, publicados em diversas áreas e revistas. Após análise das publicações disponíveis, identificou-se aquelas mais relevantes ao problema a ser tratado e selecionou-se os seguintes títulos de publicação:

- *Neurocomputing*;
- *Pattern Recognition*;
- *Computers in Industry*.

Com os títulos de publicação selecionados, a busca gerou como resultado 102 artigos. A partir desses resultados, foi realizada a leitura dos títulos em busca dos trabalhos mais pertinentes à área de detecção de anomalias. Após a leitura dos títulos, apenas 11 artigos foram selecionados. Utilizou-se como critério de exclusão a não pertinência ao tema e a ausência de dados em formatos de imagens. A partir destes 11 artigos, foi feita a leitura do *abstract*, procurando-se compreender os processos e métodos aplicados em cada estudo. Identificou-se que 4 deles analisavam cenários e condições diferentes do escopo deste trabalho (ruídos em dados ou cenas de movimento), sendo portanto descartados. Por fim, foram selecionados 7 artigos para a leitura e caracterização do estado da arte, que se encaixaram no contexto deste trabalho. Neste processo, um artigo foi excluído por não abordar uma solução para detecção de anomalias. Os 6 artigos restantes são detalhados na próxima seção.

3.1. Análise dos Trabalhos Relacionados

Nesta seção são apresentados os seis trabalhos selecionados a partir do processo da revisão sistemática.

O primeiro trabalho analisado é proposto por [Zavrtanik et al. 2021] que utiliza a arquitetura *autoencoder* para a realização de detecção de anomalias em imagens. Para realizar este trabalho, os autores aplicaram uma técnica para verificar se existem anomalias em um objeto: a imagem de entrada deve ser igual a uma imagem recriada na saída da rede neural. A rede processa a imagem e depois deve reconstruí-la com as informações extraídas. Ao final, se a imagem inicial não corresponder com a imagem de saída, então detectou-se uma anomalia. A Figura 3, retirada do trabalho de [Zavrtanik et al. 2021], mostra bem esse funcionamento. No artigo, os autores argumentam que a utilização de áreas pretas na imagem faz com que a rede neural otimize o processo de reconstrução, para que ela não acabe reconstruindo a imagem com a anomalia. São feitas várias versões da

[1] https://www.sciencedirect.com

imagem inicial com diferentes regiões escondidas em preto. Ao final, a imagem reconstruída deve estar sem a anomalia, como fica evidente na figura, para que a comparação identifique a região afetada. O autor chama esse processo de *RIAD*.

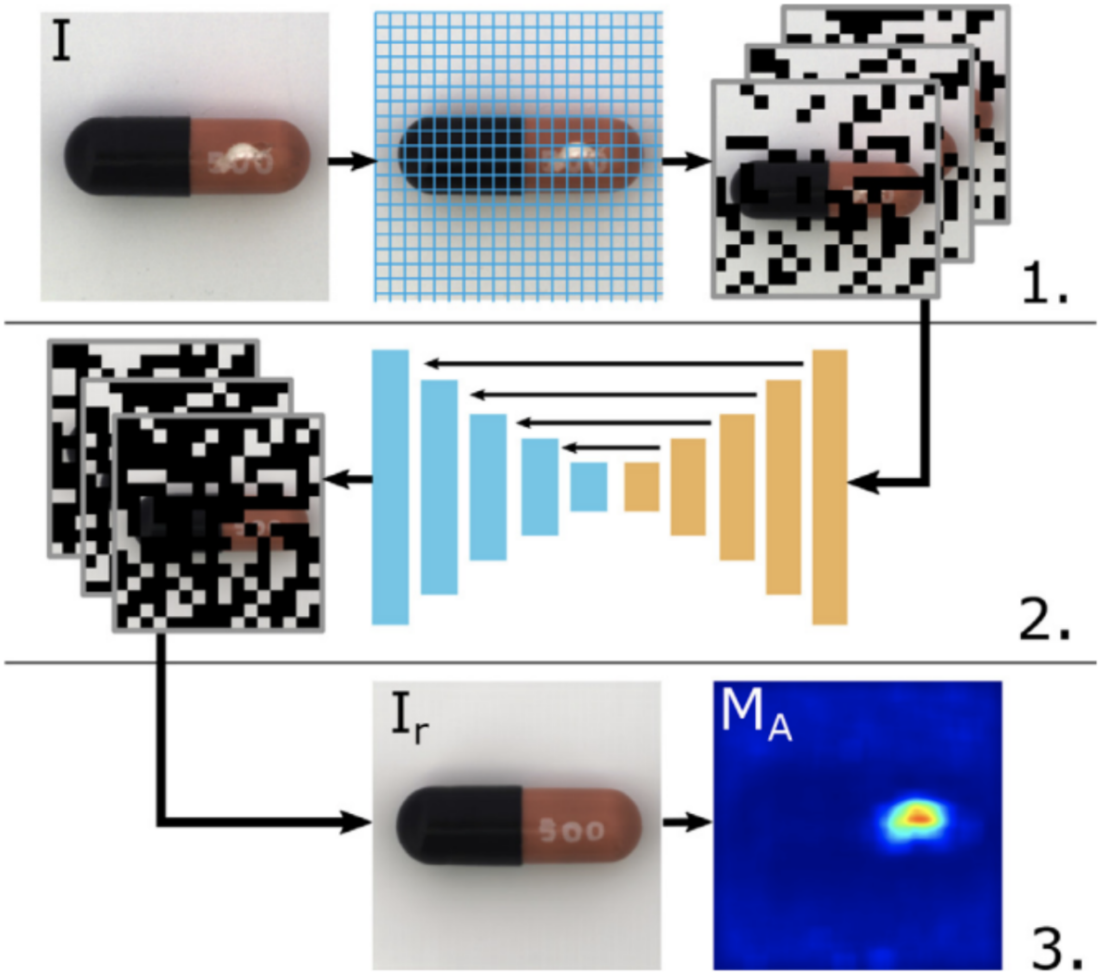

Figura 3. Representação das Etapas de uma Rede Neural Autoencoder RIAD

Para a realização do trabalho citado foi utilizada a métrica de avaliação de área sob a curva característica de operação do receptor (*AUC-ROC*). Para testar o método proposto pelo artigo, os autores utilizaram os *datasets*: MVTec [Bergmann et al. 2019], UCSD Ped2 [Mahadevan et al. 2010] e Avenue [Lu et al. 2013]. O dataset MVTec, demonstrado na imagem Figura 4 [Bergmann et al. 2019], contém 3629 imagens de treino com imagens não anômalas e 1725 imagens anômalas de diversos objetos. Já o *dataset* UCSD Ped2 [Mahadevan et al. 2010] possui 16 vídeos de treino e 12 vídeos de teste de câmeras de vigilância com fundo estático. O terceiro dataset denominado *dataset* Avenue dispõe de 30652 *frames* com fundo estático divididos entre 16 vídeos de treino e 21 vídeos de teste. Os autores definiram como objetivo obter resultados melhores do que os identificados no estado da arte naquele momento. Como resultado do método RIAD, foi verificado aumento de 4 pontos percentuais acima do melhor método descrito até o momento.

No segundo trabalho analisado, proposto por [Zhou et al. 2021], foi utilizado um

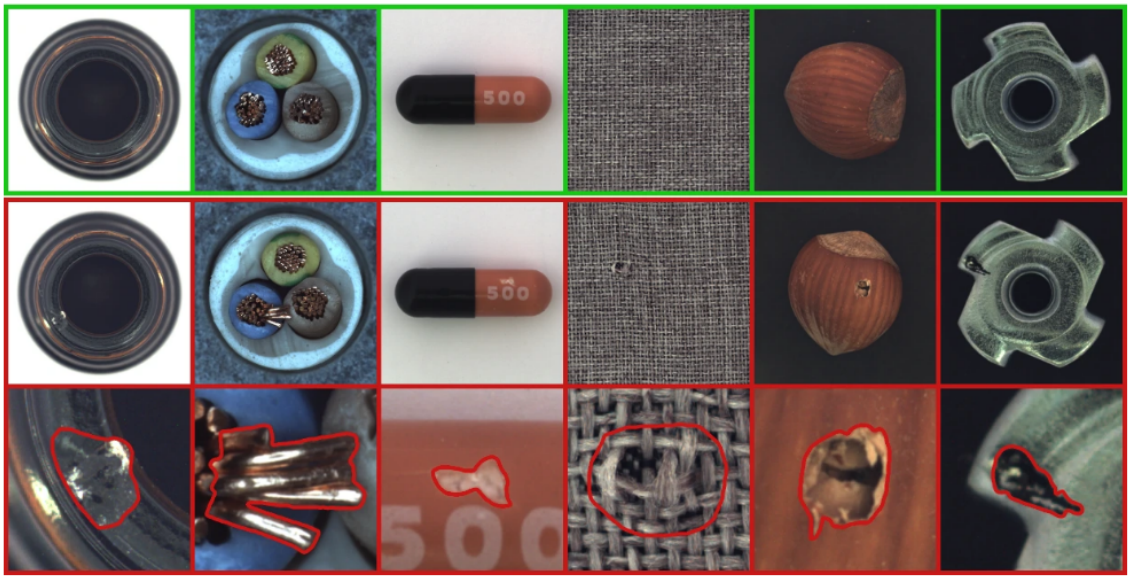

Figura 4. Exemplos do *Dataset* MVTec

modelo hibrido de técnicas de detecção de anomalia. O modelo proposto utiliza o conceito de *VAE* como um extrator de características. Em seguida, é utilizado um método de *SVDD* como classificador. Esta técnica é nomeada pelo autor como *Deep SVDD-VAE*. Nessa arquitetura conjunta, o VAE e o SVDD são usados no mesmo estágio, de forma que os parâmetros de ambos os modelos são aprendidos em conjunto. A aplicação desta técnica híbrida tem como objetivo prevenir a ocorrência de *hypersphere collapses*, que são um problema onde as características de classificação são muito semelhantes em classes diferentes, causando problemas de classificações erradas na rede. Para a avaliação deste método foi utilizada a métrica de avaliação da área sob a curva (*Area Under Curve* - AUC). O método que possui a maior área sobre a curva tem um resultado melhor na detecção de anomalias. De forma a se ter uma comparação da eficácia do método, o Deep SVDD-VAE foi comparado com outras técnicas de detecção de anomalias como *OC-SVM* e *KDE*. Para a realização dos testes foram utilizados os *datasets* MNIST [Y. LeCun 2022], CIFAR-10 [Krizhevsky 2009] e GTSRB [Stallkamp et al. 2011]. O *dataset* MNIST possui mais de 70000 dígitos de 0 a 9 escritos a mão em tons de cinza. Já o CIFAR-10 é composto de 60000 imagens de objetos coloridos do mundo real. Finalmente o *dataset* GTSRB abrange imagens de sinais de transito de 43 tipos, destes o autor utilizou a apenas os de sinais de "pare", totalizando mais de 1000 imagens. Por fim, o método não obteve um acurácia superior a maioria dos métodos comparados, porém conseguiu resolver o problema proposto de prevenir a ocorrência de *hypersphere collapses*.

No modelo proposto por [Božič et al. 2021], os autores desenvolveram um modelo híbrido utilizando uma rede classificadora apenas com rótulos, identificando se existem anomalias na imagem, e uma rede de aprendizado com imagens em que os pixels com anomalia estão destacados. Os autores chamam o primeiro método de treinamento fracamente supervisionado e o subsequente de completamente supervisionado. Realizar o treinamento completamente supervisionado pode ser uma tarefa custosa, já que a rotulação de um *dataset* a nível de pixel é algo demorado e complexo. Em vista disso, foram desenvolvidas redes treinadas de forma fracamente supervisionada ou não supervi-

sionada, onde a rede aprende sozinha ou com pouca ajuda humana. Contudo, tais redes possuem uma acurácia menor que redes treinadas de forma completamente supervisionada. Motivado por este problema, [Božič et al. 2021] propõe uma rede hibrida constituída de duas sub-redes, como demonstrado na Figura 5 [Božič et al. 2021], sendo a primeira treinada com imagens rotuladas a nível de pixel e a segunda a nível de imagem. Esta é capaz de ter a acurácia próxima a de uma rede supervisionada, mas utilizando *datasets* com pouca rotulação a nível de pixel. A métrica utilizada foi a precisão média (*AP*), que trata-se do cálculo da área sob a curva de *precision-recall*. Para a realização da pesquisa foram utilizados os seguintes *datasets*: DAGM [Weimer et al. 2016], KolektorSDD [Tabernik et al. 2019], KolektorSDD2 [Božič et al. 2021] e Severstal steel [Severstal 2023]. O *dataset* DAGM contém imagens geradas por computador de 10 tipos de superfícies diferentes e vários tipos de defeitos. Já o *dataset* KolektorSDD possui imagens em preto e branco de produtos reais. Assim como o *dataset* KolektorSDD, o *dataset* KolektorSDD2 possui possui imagens de produtos reais, entretanto neste *dataset* elas são coloridas. Por fim temos o *dataset* Severstal steel que compõe-se de 12568 imagens em preto e branco dividido em 4 classes com uma série de diferentes defeitos. O modelo proposto teve uma acurácia superior ao modelo comparado no trabalho de [Božič et al. 2021], tendo atingido o objetivo proposto.

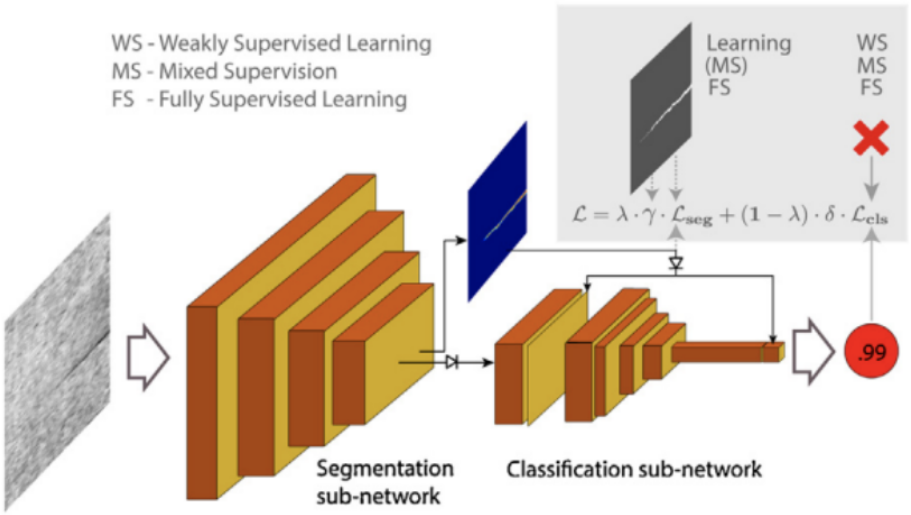

Figura 5. Modelo Híbrido Proposto por [Božič et al. 2021]

Em seguida foi analisado do artigo de [Yang et al. 2022], onde é proposto uma rede neural para identificar defeitos em produtos a nível de pixel utilizando visão computacional. Este trabalho foi realizado com o intuito de se obter uma acurácia superior ao modelos analisados pelo autor em seu estado da arte. Para realização desta tarefa, utilizou-se o método *deep feature correspondence* para a detecção e segmentação de anomalias de maneira não supervisionada. Este método consiste em pré-treinar o modelo com um conjunto de imagens e então comparar as características deste conjunto com a imagem avaliada. Caso a imagem não possua certas características semelhantes, ela é considerada anômala. Para realizar os experimentos foi utilizada pelo autor uma rede VGG19 [Simonyan and Zisserman 2014]. Para realizar a experimentação do modelo, foram utili-

zados os *datasets* MVTec AD [Bergmann et al. 2019] e BottleCap dataset desenvolvido por [Yang et al. 2022] contendo 1100 imagens reais coloridas de tampas de garrafas. Para a avaliação dos resultados foram utilizadas as métricas de AUC-ROC e área sob a curva de sobreposição por região (*AUC-PRO*). Além destes, também se foi avaliada a capacidade de segmentação das anomalias utilizando a métrica de interseção sobre união, conhecida por *IOU*. Após analisar o estudo feito por [Yang et al. 2022] pode-se perceber que foi obtido uma acurácia superior aos métodos comparados utilizando o *dataset* MVTec AD, atingindo assim o objetivo proposto pelo autor. Além disso a técnica demonstra possuir uma boa eficacia em *datasets* de imagens industriais em geral, como pode-se ver pelos testes feitos com o *dataset* BottleCap.

No trabalho de [Shi et al. 2021] é demonstrado que técnicas de detecção de anomalias baseadas em características (*feature-based*), como VAE e *GAN*, podem ter uma grande quantidade de falsos positivos quando aplicados em imagens com texturas complexas e bordas afiadas. Para resolver este problema, [Shi et al. 2021] propõem uma técnica de representação de características multi escala (*multi-scale feature representation*). Assim, é utilizada uma rede neural convolucional (CNN) pré-treinada para se obter uma representação hierárquica das características da imagem. Depois, é feito um pré-processamento destas para então serem usadas em uma rede baseada em características. Para realização dos testes, os autores utilizaram o *dataset* MVTec [Bergmann et al. 2019]. Para avaliar a eficacia dos método proposto, o mesmo foi comparados com as técnicas citadas no estado da arte pesquisado pelos autores. A quantificação dos resultados foram obtidos utilizando as métricas de AUC-ROC e curva de sobreposição por região (*PRO-AUC*). Neste trabalho os autores obtiveram resultados muitas vezes superiores ao estado da arte com o qual foram comparados, tendo assim obtido sucesso no objetivo proposto.

Como último trabalho analisado, tem-se o proposto por [Fuchs et al. 2021]. Muitas vezes para identificar anomalias em peças de alumínio, é utilizado o método de tomografia computadorizada, já que as anomalias podem estar na parte interior de um objeto. Contudo, este método pode exibir informações sigilosas de empresas, como por exemplo a quantidade de material utilizado na fabricação do produto e as em algumas vezes até o método de fabricação. Tendo isso em vista, [Fuchs et al. 2021] desenvolveram um *dataset* composto de imagens 3D de diversas peças de alumínio. As imagens foram desenvolvidas pelos próprios autores, a fim de disponibilizar uma forma de treinamento de redes neurais que não exponham segredos industriais das empresas. Os autores utilizam a combinação de uma arquitetura *autoencoder* do tipo *U-Net*. Para realizar a avaliação do trabalho foi usada a técnica de probabilidade de detecção *POD* e IOU. Este trabalho não tem como objetivo principal a detecção de anomalias, mas sim a criação de um *dataset* realista de peças de alumínio. Porém ele demonstra a dificuldade de se obter um *dataset* para certos tipos de detecção de defeitos. Por fim, os autores atingem o objetivo proposto de criar um *dataset* que seja realista para treinamentos futuros.

3.2. Pontos Relevantes da Análise Realizada

Para sumarizar os trabalhos analisados, elaborou-se a Tabela 1 contendo os principais resultados de cada pesquisa. Após a realização da revisão dos trabalhos selecionados foi possível observar o uso constante de alguns métodos para detecção de anomalias em imagens. Como observado na Figura 6, os *autoencoders* e as redes neurais convolucionais são os métodos mais utilizados atualmente. Em alguns casos eles foram empregados em

conjunto, como no trabalho de [Shi et al. 2021].

Referência	Modelo	Dataset	Desempenho	Métrica
[Zhou et al. 2021]	Deep VAE-SVDD	MNIST	98.8	AUC
[Yang et al. 2022]	DFC	MVTec	97.8	ROC AUC
[Shi et al. 2021]	CNN VAE	MVTec	95.0	ROC AUC
[Zavrtanik et al. 2021]	RIAD	MVTec	94.2	ROC AUC
[Božič et al. 2021]	RN de dois estágios	KolektorSDD	93.43	AP
[Fuchs et al. 2021]	Autoencoder	Criado pelo autor	85.9	IoU

Tabela 1. Principais Trabalhos Relacionados

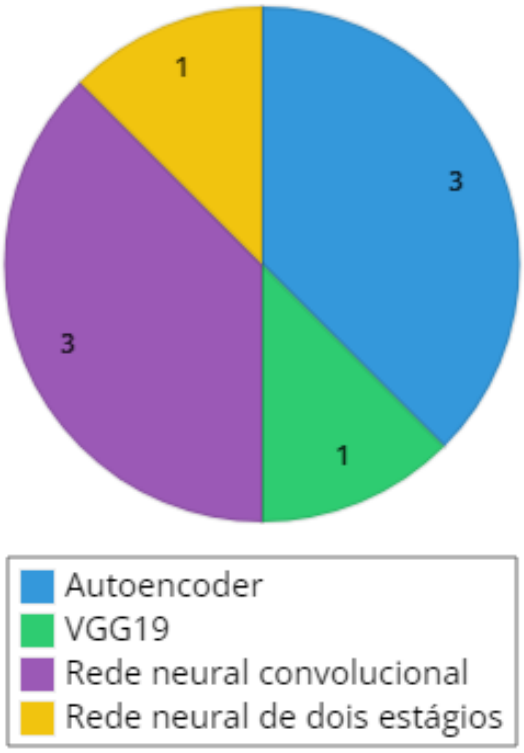

Figura 6. Relação de Arquiteturas Utilizadas nos Trabalhos Analisados

Pode-se observar também o uso de modelos pré-treinados, como no artigo de [Yang et al. 2022]. Nestes trabalhos pode-se observar o uso constante do *dataset* MVTec [Bergmann et al. 2019], aparecendo em três dos seis trabalhos selecionados, sendo amplamente citado pelos autores como um *dataset* referência no treinamento de modelos de detecção de anomalias. Além disso, diversos autores relevam que a área é muito desafiadora, já que ruídos em imagens e imagens com texturas complexas podem gerar um grande número de falsos-positivos [Shi et al. 2021].

Também ficou evidente a dificuldade em se obter *datasets* de imagens contendo anomalias nesta área. Atribui-se a este fato alguns motivos, tais como: a baixa ocorrência das anomalias, ausência de registros quando elas ocorrem e ainda a proteção a situações que não devem ser divulgadas ou que revelariam segredos industriais por meio dos *datasets* [Fuchs et al. 2021] [Božič et al. 2021]. Apesar destes desafios, nota-se na Tabela 1

que os autores obtiveram resultados promissores nas arquiteturas analisadas. Os resultados descritos na literatura registram valores de acurácia próximos a 100%.

4. AutoEncoders

As redes neurais *autoencoders* foram propostas como uma solução não-linear para a análise de componentes principais (PCA) [Kramer 1991]. Tal método vem sendo amplamente utilizado para várias tarefas, como redução de dimensionalidade, remoção de ruídos, detecção de anomalias e até mesmo para a geração de novos dados semelhantes aos dados de treinamento. Durante o treinamento, o *autoencoder* é alimentado com um conjunto de dados de entrada e a função objetivo é minimizar a diferença entre a imagem reconstruída pelo *decoder* e a imagem original. Isso é feito ajustando os pesos e os bias das camadas do *encoder* e do *decoder* através de um processo chamado de otimização.

O *autoencoder* é composto por três partes: a primeira é chamada de *encoder*, capaz de extrair características de uma imagem. Depois, estas características são mapeadas na segunda parte, em um espaço vetorial chamado espaço latente. Por fim, a última parte, chamada de *decoder*, é capaz de utilizar as características de um ponto do espaço latente para gerar imagens. O *decoder* recebe essa representação do espaço latente e tenta reconstruir a imagem original. Ele expande a representação de volta à dimensão original, tentando gerar uma imagem o mais próxima possível da entrada original. A ideia é que o *autoencoder* aprenda a reconstruir a entrada com o mínimo de perdas possível, forçando-o a capturar as características mais importantes dos dados.

Para a identificação de anomalias, primeiramente uma rede neural *autoencoder* deve ser treinada com um conjunto de imagens que serão os elementos que a rede ficará especializada em reconstruir. O *encoder* recebe cada imagem e a comprime em uma representação de menor dimensão no espaço latente. Essa representação é uma versão compacta da imagem original, onde as características mais importantes são mantidas e as informações menos relevantes são descartadas. Para exemplificar, suponha um modelo gerado a partir de imagens de peças amarelas. Após o aprendizado, ele reconhecerá imagens de peças amarelas com baixa taxa de erro. Caso a rede receba uma imagem vermelha, ao invés de amarela, sua taxa de erro será alta, já que o modelo terá dificuldade em reconstruir o elemento com eficácia. Isso posto, comparando-se a taxa de erro média das imagens amarelas com a taxa de erro da imagem vermelha, pode-se afirmar que as imagens que diferem dos padrões utilizados no treinamento (peças vermelhas) produzirão uma taxa de erro alta. Em outras palavras, o modelo mapeia o padrão das imagens do conjunto de treino para um espaço latente. Imagens que não correspondem aos padrões reconhecidos são sinalizadas pelas altas taxas de erro.

Por meio dos estudos e experimentos realizados e documentados na literatura, fica evidente que os *autoencoders* são eficazes na geração de imagens e detecção de falhas. Em estudos posteriores, tal como o trabalho de [Kingma and Welling 2022], foi introduzida uma variação mais eficiente do modelo. Modelos baseados nestas variações, conhecidos como *autoencoders* variacionais, utilizam métodos probabilísticos para o mapeamento no espaço latente. O detalhamento destes modelos é descrito na seção seguinte.

4.1. *Autoencoders* Variacionais

O modelo *autoencoder* variacional (VAE) corresponde a uma arquitetura de rede neural generativa profunda capaz de aprender padrões em grupos de imagens e gerar novas ima-

gens baseadas nesses padrões. Como pode ser observado na Figura 7, essa rede segue uma arquitetura que utiliza os três componentes principais: o *encoder*, o espaço latente e o *decoder* [Foster 2019a].

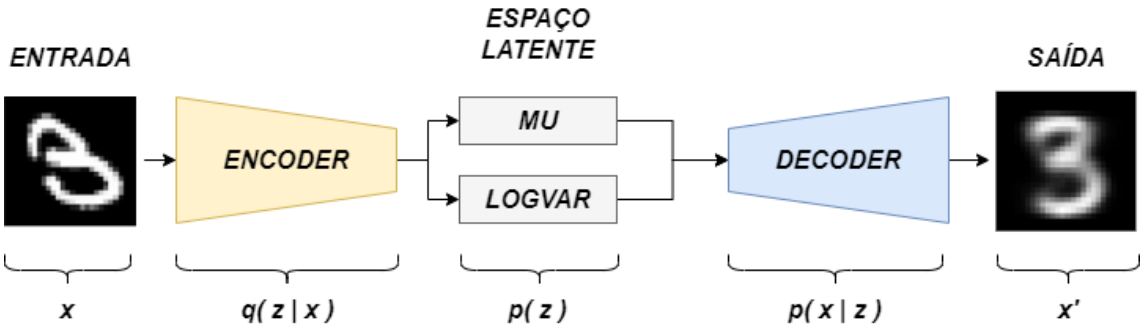

Figura 7. Arquitetura de um autoencoder variacional

O componente *encoder* transforma uma imagem de entrada em uma representação latente, ou seja, em uma descrição compacta e significativa de um conjunto de dados. O espaço latente é uma região multidimensional onde os vetores latentes são mapeados. Finalmente, o componente *decoder* utiliza a representação latente para gerar uma imagem de saída. O VAE foi criado como uma variação do *autoencoder*. Enquanto um *autoencoder* simplesmente mapeia a representação latente, um VAE utiliza uma distribuição normal para mapear esta representação [Foster 2019a, Fuchs et al. 2021].

O componente *encoder* é a etapa representada na Figura 7 pela parte em amarelo, como sendo $q(z \mid x)$. Ele é constituído por diversas camadas convolucionais, sendo que a primeira possui dimensões correspondentes à imagem de entrada com os três canais de cor RGB. As dimensões da imagem são reduzidas e compactadas à medida que o processamente avança entre as camadas até o espaço latente.

O *encoder* é o responsável por construir a representação latente dos dados, que é uma representação matemática das principais características da imagem. Tal representação produz uma compressão dos dados, sendo realizada por meio de uma sequência de camadas convolucionais que extraem características relevantes da imagem [Foster 2019a, Doersch 2021].

Durante a fase de codificação, o *encoder* do VAE mapeia os dados de entrada em dois vetores: o vetor *mu* (média) e o vetor *logvar* (logaritmo da variância). Esses vetores são usados para definir uma distribuição de probabilidade multivariada no espaço latente. O espaço latente em um VAE é uma representação de baixa dimensão dos dados de entrada, onde cada ponto no espaço latente corresponde a uma amostra gerada. Os vetores *mu* e *logvar* são calculados pelo *encoder* e são usados para definir uma distribuição latente no espaço latente, permitindo a geração de novas amostras e a aprendizagem probabilística de representações latentes.

O vetor *mu* representa a posição média da distribuição no espaço latente e é calculado diretamente pelas camadas do *encoder*. O vetor *logvar*, por outro lado, representa a logaritmo da variância da distribuição e é calculado de forma análoga.

Durante a fase de treinamento, o VAE tenta minimizar duas perdas principais: a perda de reconstrução e a perda de regularização KL (Kullback-Leibler). A perda

de reconstrução mede o quão bem o VAE reconstrói a entrada original a partir da representação latente, enquanto a perda KL penaliza a diferença entre a distribuição latente aprendida e uma distribuição latente padrão (geralmente uma distribuição normal padrão).

A perda KL é calculada a partir dos vetores *mu* e *logvar*, usando a fórmula [Doersch 2021]:

$$\text{kl\_loss} = -0.5 \cdot \sum(1 + \log(\text{logvar}) - \mu^2 - \exp(\text{logvar}))$$

Essa fórmula é derivada a partir da divergência de Kullback-Leibler entre a distribuição latente aprendida e a distribuição normal padrão.

O espaço latente recebe dados da saída do *encoder* e possui um tamanho pré-definido em termos de dimensões. O número de dimensões deve ser proporcional à complexidade das imagens do *dataset*. Por exemplo, suponha um espaço latente de 20 dimensões, ele terá um vetor *logvar* e um vetor *mu* cada um com 20 dimensões, obtidos a partir da saída do *encoder*. Se a camada *flatten* do *enconder* tiver 12.544 parâmetros, então o *logvar* e o *mu* terão 250.900 parâmetros, calculados como (12.544 + 1) * 20 [Foster 2019a].

O valor *mu* é atualizado para representar a média da distribuição latente aprendida pelo *autoencoder*, enquanto o *logvar* é atualizado para representar o logaritmo da variância dessa distribuição, ambos necessários para realizar a equação da distribuição normal.

Posteriormente, essas camadas são conectadas a uma camada que utiliza uma função *lambda* que pode ser personalizada. No caso do *autoencoder*, ela contém um cálculo onde utiliza uma distribuição normal[2] para realizar uma amostragem do espaço latente, ou seja, para produzir uma saída que será utilizada no *decoder*.

O *decoder*, ilustrado na Figura 7 por *p(x | z)*, é a parte final do *autoencoder*. Ele é constituído de múltiplas camadas convolucionais que tem como objetivo gerar a reconstrução de uma imagem a partir de uma amostra do espaço latente. O *decoder* é comumente uma versão espelhada do *encoder*, mas pode ter mais ou menos camadas. A diferença do *decoder* em comparação ao *encoder* é o uso de camadas convolucionais transpostas. Enquanto as camadas convolucionais do *encoder* extraem apenas os dados mais relevantes de informação, as camadas transpostas do *decoder* realizam o processo inverso, expandindo o espaço latente em uma imagem reconstruída [Foster 2019a].

Após o treinamento do VAE, o espaço latente aprendido pode ser utilizado para gerar novas amostras de dados. Isso é possível por meio da amostragem a partir da distribuição latente aprendida durante o treinamento. Ao realizar essa amostragem, o VAE é capaz de gerar novas amostras de dados que possuem características semelhan-

[2]A distribuição normal, também conhecida como distribuição gaussiana, é uma das distribuições de probabilidade mais importantes e amplamente utilizadas na estatística e na ciência de dados. Ela descreve uma curva simétrica em forma de sino, definida por sua média e desvio padrão. A forma da curva normal é determinada pela média e pelo desvio padrão. A média representa o centro da curva, enquanto o desvio padrão controla o quão espalhados os valores estão em relação à média. Quanto maior o desvio padrão, mais achatada e dispersa será a curva [Foster 2019b].

tes à distribuição dos dados de treinamento. Isso significa que as amostras geradas serão parecidas com as amostras originais, mantendo características e padrões semelhantes às observadas no conjunto de treinamento.

4.2. Concatenação de Camadas

Além do uso do VAE como uma rede capaz de gerar novas imagens a partir de um conjunto de outras imagens, ele também pode ser utilizado como uma rede capaz de reconstruir imagens. A reconstrução pode ser utilizada como uma rede *denoising*, capaz de remover defeitos em imagens, como sinais de estática ou de envelhecimento ou em outros problemas. Um uso comum é na detecção de anomalias, onde se pretende reconstruir a imagem e verificar se a taxa de erro nessa reconstrução é maior que o esperado. No entanto, a utilização do VAE como uma rede de reconstrução pode resultar em imagens borradas, como demonstrado no item da esquerda Figura 8 [Sikka 2023], principalmente em conjuntos complexos, como rostos ou paisagens. Para melhorar a qualidade da reconstrução, a concatenação de camadas pode ser utilizada para combinar as informações de uma determinada camada convolucional do *encoder* com uma do mesmo tamanho do *decoder*. Isso permite que a reconstrução seja mais precisa e detalhada como demonstrado no item da direita da Figura 8. Em detecção de anomalias isso permite que a rede seja focada na reconstrução e tenha taxas de erro mais estáveis [Zavrtanik et al. 2021].

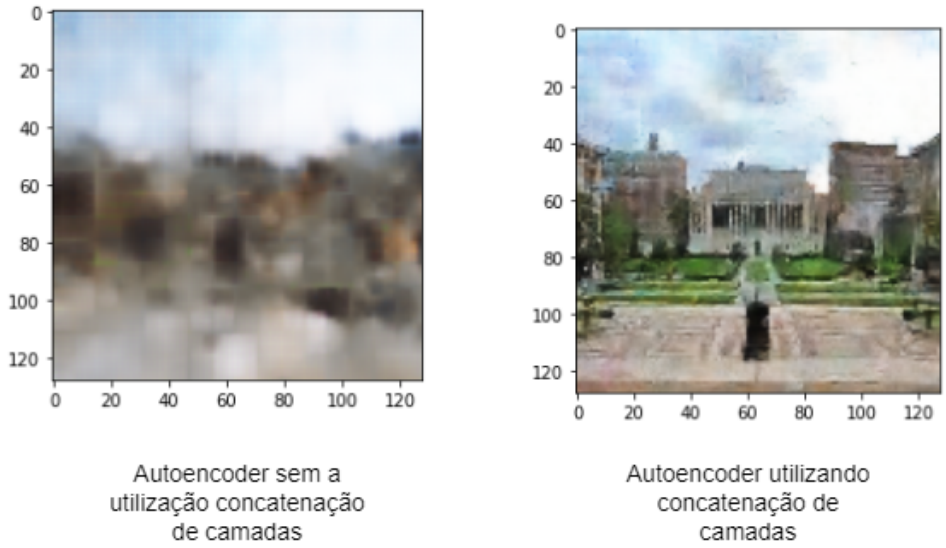

Figura 8. Comparação de uma rede sem concatenação de camadas como uma que utiliza concatenação

4.3. Treinamento

O treinamento de um VAE segue uma abordagem similar ao treinamento de uma rede neural com o objetivo de manter a imagem reconstruída o mais próxima possível da original. Para alcançar esse objetivo, são utilizadas duas funções de perda em conjunto. A primeira função de perda é responsável por calcular a taxa de erro na reconstrução, normalmente realizada por uma função MSE (Mean Squared Error) ou a sua versão quadrática, a RMSE (Root Mean Squared Error). Ambas são funções de perda que medem a média dos erros

entre o valor previsto e o valor real. A segunda função de perda é a medida KL (Kullback-Leibler), calculada como a divergência de Kullback-Leibler e utilizada para penalizar a distância entre a distribuição das amostras latentes amostrada do espaço latente e uma distribuição normal de probabilidade. O objetivo do uso da perda KL é incentivar que a distribuição das amostras latentes se aproxime da distribuição normal padrão, o que permite que o espaço latente seja bem estruturado e mais fácil de explorar. Além disso, a perda KL atua como uma regularização, evitando que o modelo fique preso em um espaço latente muito restrito e limitado, como demonstrado Figura 9 [Foster 2019a] na qual observa-se na imagem da esquerda uma imagem gerada a partir de um espaço latente de um autoencoder e à direita uma imagem gerada por um autoencoder variacional.

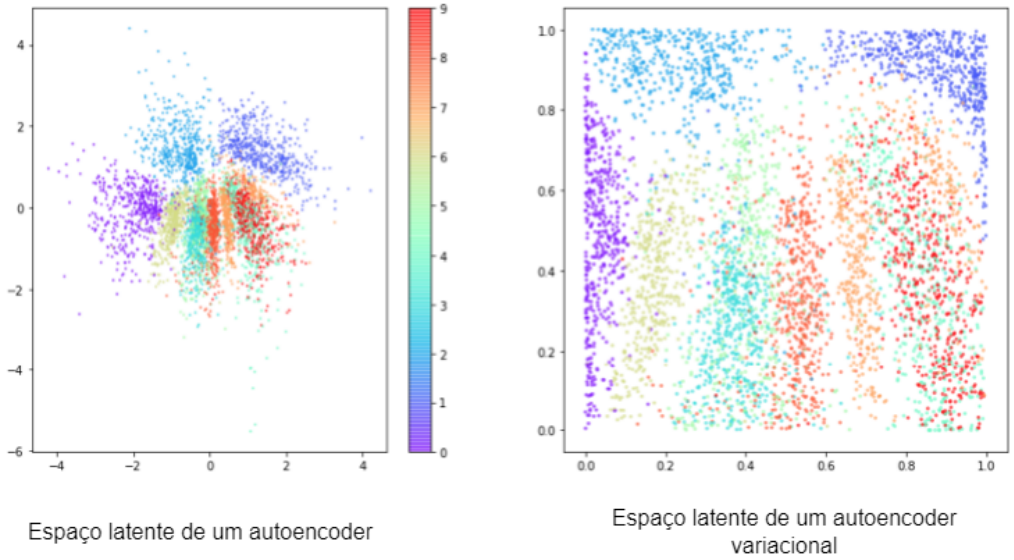

Figura 9. Comparação do espaço latente de um autoencoder com a de um autoencoder variacional

4.4. Métricas de Avaliação

Após a etapa de treinamento, é importante realizar uma avaliação do modelo para obter uma noção do seu desempenho. Para isso, são utilizadas métricas de avaliação, como acurácia, precisão e *recall*. Para calculá-las, é utilizada uma matriz de confusão. A matriz de confusão é uma tabela onde são colocados os resultados da classificação da rede neural, como mostra a Figura 10 [Rodrigues 2019]. Quando a rede classifica corretamente um elemento existente ele é um verdadeiro positivo (VP), indicando que a rede acertou a classificação. O mesmo se aplica aos casos negativos: quando a rede neural indica que não existe o elemento na imagem e ele realmente não estava presente são chamados de verdadeiros negativos (VN). Ainda, a rede pode indicar a presença de um elemento quando ele não existe, conhecido como falso positivo (FP), e o contrário, indicar como o elemento não existente mas ele estar presente na imagem, falso negativo (FN) [Rodrigues 2019].

Tendo em vista os valores obtidos na matriz de confusão, fica muito mais simples calcular as métricas de avaliação. Em um problema de classificação, as métricas mais comumente utilizadas são:

		Detectada	
		Sim	**Não**
Real	**Sim**	Verdadeiro Positivo (VP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (VN)

Figura 10. Matriz de Confusão

- Acurácia: A acurácia se trata do quociente do número de predições corretas sobre o número total de predições feitas. Esta métrica é normalmente utilizada quando se tem o mesmo número de amostras em todas as classes [Developers 2022];
- Precisão: A precisão é a razão de positivos corretamente classificados pelo número de resultados classificados como positivos. A precisão reflete o quão confiável o modelo é em classificar dados como positivo [Developers 2022];
- *Recall*: *Recall* é o número de resultados classificados corretamente como positivos divididos pelo pelo número de resultados que deveriam ser classificados como positivo. Assim como na precisão, o *recall* mede o quão confiável o modelo é em fazer precisões positivos [Developers 2022];
- F1-score: F1-score realiza a média harmônica entre a precisão e o *recall*. Ele demonstra o quão corretamente o modelo está classificando os dados [Developers 2022];
- Curva ROC: A curva ROC (curva de característica de operação do receptor) é um gráfico que mostra o desempenho de um modelo em todos os pontos da classificação. Este gráfico é calculado utilizando o *recall* sobre a taxa de falsos positivos [Developers 2022];
- AUC: AUC (*Area under the curve*) é a área sobre a curva ROC. Ela calcula a probabilidade de o modelo classificar um exemplo positivo aleatório mais alto do que um exemplo negativo aleatório [Developers 2022].

5. Materiais e Métodos

Nesta seção são descritos os materiais e métodos utilizados na realização deste trabalho.

5.1. Métodos de pesquisa

Este trabalho constitui-se em uma uma pesquisa de natureza exploratória que visa pesquisar, compreender e aplicar um modelo de IA em forma de uma rede neural autoencoders no cenário de detecção de anomalias em barragens. Dessa forma, o trabalho visa detectar rachaduras e defeitos no concreto que podem acabar resultando em desastres. Visando aplicar um modelo de IA na detecção de anomalias que identifique defeitos e rachaduras em barragens, o trabalho foi dividido nas seguintes etapas:

- Etapa 1: Identificação de bases de dados (construção do *dataset*)
- Etapa 2: Pré-processamento dos dados

- Etapa 3: Aplicação de algoritmos
- Etapa 4: Avaliação do modelo

5.2. Materiais

Utilizou-se a plataforma Anaconda[3] para gerenciamento de pacotes e a plataforma Jupyter Notebook[4] para a escrita do código na linguagem de programação Python. Também, fez-se o uso da biblioteca TensorFlow para auxiliar na tarefa de implementação do autoencoder variacional. O treinamento e a validação da rede foi feito utilizando as imagens disponíveis no *dataset* Concrete Crack Images for Classification[5], contendo imagens de concreto rachado. Neste experimento foi utilizado um computador com as seguintes especificações para o treinamento e avaliação da rede:

- Processador Intel core i5 10400
- Placa mãe MSI MAG B560 Tomahawk
- Memória RAM de 16 Gigabytes DDR4
- Placa de video Nvidia Geforce RTX 3060

6. Desenvolvimento

O desenvolvimento deste trabalho foi organizado em seções para melhor abordagem das etapas.

6.1. Etapa 1: Identificação de bases de dados (construção do *dataset*)

O trabalho foi realizado utilizando o *dataset Concrete Crack Images for Classification* da Kaggle. Ele contém ao todo 40.000 imagens de paredes de concreto, 20.000 de concreto sem rachaduras e 20.000 com rachaduras. Cada imagem possui uma resolução de 227x227 *pixels* com canais de cor RGB.

Das 20.000 imagens sem anomalias:

- 12.500 foram utilizadas para treinar a rede neural;
- 2.500 para a taxa média de erro;
- 5.000 para validação da rede.

E das 20.000 imagens com anomalias:

- 5.000 foram utilizadas para validação da rede;
- 2.500 para a taxa média de erro;
- 12.500 não foram utilizadas.

Como o objetivo era ter a mesma quantidade de imagens com e sem rachaduras para validação para a rede, não foi-se necessário utilizar as 15.000 imagens restantes de anomalias.

[3]http://www.anaconda.org
[4]https://jupyter.org/
[5]https://www.kaggle.com/datasets/arnavr10880/concrete-crack-images-for-classification

6.2. Etapa 2: Pré-processamento dos dados

Para a adequação das imagens foi necessário realizar um redimensionamento de seu tamanho, aplicando-se a função *flow_from_directory* da classe *ImageDataGenerator* da biblioteca Keras, assim reduzindo seu tamanho de 227x277 *pixels* para 224x224 *pixels*. A razão dessa mudança é devido ao fato de que quando a imagem passar pela rede *autoencoder*, seu tamanho será divido por dois devido ao uso de *stride* dois nas camadas convolucionais. Quando a imagem possui um tamanho ímpar de *pixels*, o número com vírgula é descartado. Ou seja, se o tamanho da imagem é 227 *pixels*, na próxima etapa ele passa a ser 113, já que não utilizaria a medida correta de 113,5. Na reconstrução da imagem, ele faz o processo inverso, dobrando a quantidade de *pixels*. Então, seus valores poderiam passar de 28 *pixels* para 56, para 112 e assim por diante. Quando o algoritmo fizer a comparação das duas imagens no final, ocorrerá um erro, já que seus tamanhos ficarão diferentes justamente por essa perda de *pixels* no processo.

6.3. Etapa 3: Aplicação de algoritmos

Para realização deste trabalho foi utilizada uma rede neural *autoencoder* variacional. Na fase de treinamento da rede, foram utilizadas 12.500 imagens sem anomalias oriundas do *dataset Concrete Crack Images for Classification*. Devido a fase de treinamento ser feita de forma não supervisionada, não foi aplicado nenhum algoritmo de *cross-validation*.

A arquitetura, ilustrada na Figura 11, foi criada neste trabalho, utilizando os seguintes componentes:

- Camada de entrada do *encoder*: Essa camada recebe uma imagem de 224x224 *pixels* com 3 filtros referentes ao RGB da imagem;
- Primeiro bloco do *encoder*: Esse bloco possui uma camada convolucional de 112x112 com 16 filtros, seguido de uma camada de *batch normalization* e função de ativação *Leaky ReLu*. Também possui uma camada de *dropout* com uma taxa de 25%;
- Segundo bloco do *encoder*: Esse bloco é idêntico ao primeiro bloco do *encoder*, porém com um tamanho de 56x56 com 32 filtros;
- Terceiro bloco do *encoder*: Esse bloco possui uma camada convolucional de 28x28 com 32 filtros, seguido de uma função de ativação *Leaky ReLu*. Não foi utilizado *batch normalization* e *dropout* neste bloco devido aos conflitos causados na concatenação feita com o *decoder*;
- Terceiro bloco do *encoder*: Esse bloco possui uma camada convolucional de 14x14 com 64 filtros, seguido de uma ativação *Leaky ReLu*;
- Espaço latente: Um espaço latente com 20 dimensões;
- Camada de entrada do *decoder*: Essa camada recebe os dados vindos do espaço latente e possui um tamanho de 14x14 com 64 filtros;
- Primeiro bloco do *decoder*: Esse bloco possui uma camada convolucional transposta de 28x28 com 32 filtros, seguido de uma camada de *batch normalization* e uma ativação *Leaky ReLu*. Também conta com uma camada de *dropout* com uma taxa de 25%. Por fim, neste bloco é realizada a concatenação com o terceiro bloco do *encoder*;
- Segundo bloco do *decoder*: Esse bloco é idêntico ao primeiro bloco do *decoder*, mas com um tamanho de 56x56 com 32 filtros e sem a utilização da concatenação;

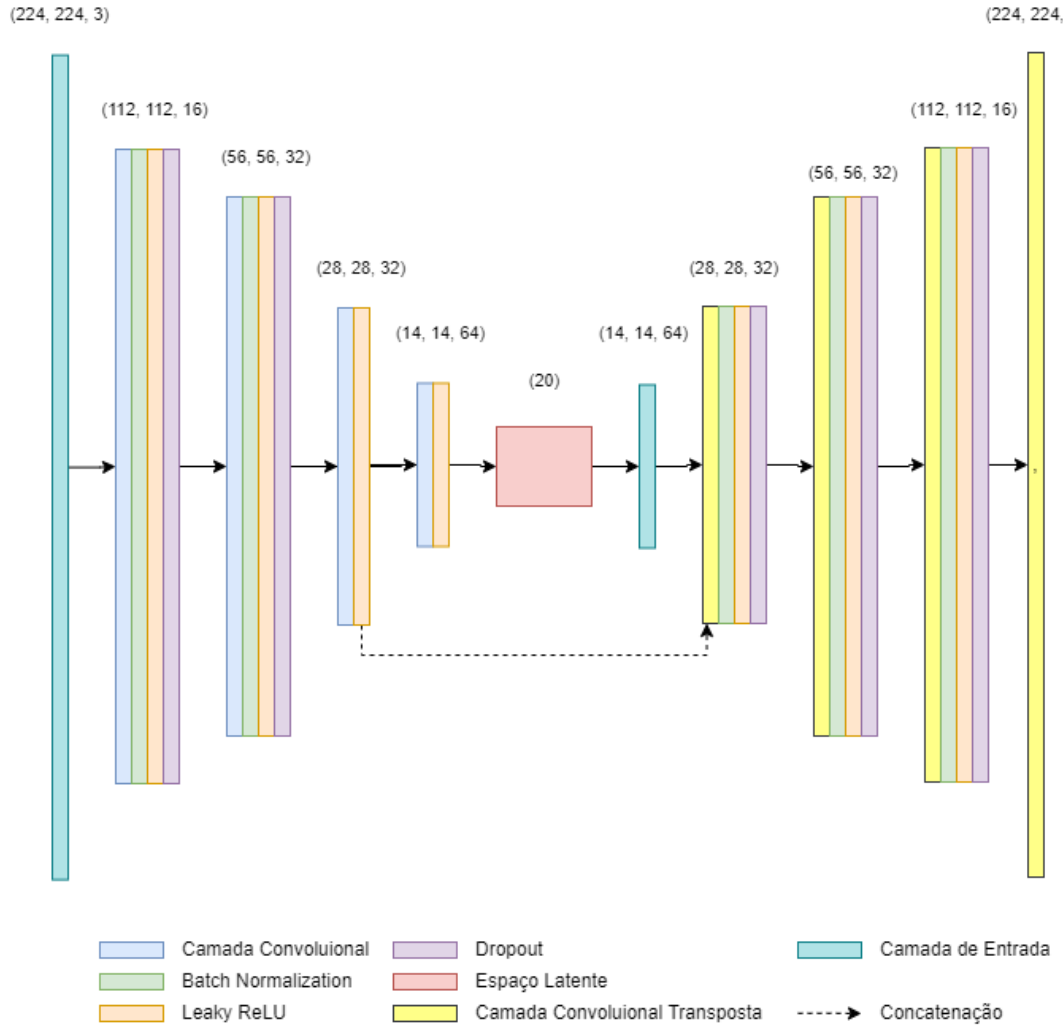

Figura 11. Arquitetura do autoencoder variacional construído

- Terceiro bloco do *decoder*: Esse bloco é idêntico ao segundo bloco do *decoder*, porém com um tamanho de 112x112 com 16 filtros;
- Quarto bloco do *decoder*: Esse bloco consiste em uma camada convolucional transposta com uma ativação sigmoidal e é responsável pela saída da rede.

Também, foram utilizados os hiper-parâmetros listados na Tabela 2.

Após o treino do *autoencoder* variacional, foi necessário descobrir a taxa de erro médio do modelo. Para tal, foi calculada a média harmônica da taxa de erro de 2.500 imagens sem anomalias. Posteriormente, essa taxa de erro foi utilizada para identificar a existência de anomalias em outras imagens.

6.4. Etapa 4: Avaliação do Modelo

A avaliação do modelo foi feita utilizando um conjunto de 10.000 imagens, sendo 5.000 com anomalias e 5.000 sem anomalias. Pra avaliar tal modelo, foram utilizadas as métricas de avaliação como precisão, *recall* e F1-score, que colocam em números os acertos da rede e fazem com que seja simples de visualizar seu funcionamento e resultados [Xu et al. 2021, An and Cho 2015].

Nome	Descrição	Valor/Dimensão
dropout	Taxa de dropout	0.25
batch_size	Tamanho do lote	32
learning_rate	Taxa de aprendizado	0.0001
r_loss_factor	Multiplicador do fator de perda RMSE	100
epochs	Número de épocas	50
encoder_input	Dimensões de entrada do encoder	224x224x3
encoder_conv_0	Dimensões da saída do encoder conv_0	112x112x16
encoder_conv_1	Dimensões da saída do encoder conv_1	56x56x32
encoder_conv_2	Dimensões da saída do encoder conv_2	28x28x32
encoder_conv_3	Dimensões da saída do encoder conv_3	14x14x64
z_dim	Dimensão do espaço latente	20
decoder_conv_3	Dimensões da saída do decoder conv_3	14x14x64
decoder_conv_2	Dimensões da saída do decoder conv_2	28x28x32
decoder_conv_1	Dimensões da saída do decoder conv_1	56x56x32
decoder_conv_0	Dimensões da saída do decoder conv_0	112x112x16
decoder_output	Dimensões de saída do decoder	224x224x3

Tabela 2. Hyperparâmetros e Dimensões

Foram realizadas 22 execuções do modelo, com um tempo médio de duas horas por execução. Cada execução obteve um resultado diferente devido ao uso da camada *dropout* que zera o valor de alguns neurônios selecionados aleatoriamente [Academy 2022]. Estas 22 execuções obtiveram resultados que variaram entre 83.1% e 87%. Para fins dessa análise foi considerado o melhor modelo e seus resultados.

As imagens foram classificadas de acordo com a Figura 12, que é uma matriz de confusão feita com os valores obtidos da execução do melhor caso da rede neural. Ela apresenta 4606 verdadeiros positivos, 4099 verdadeiros negativos, 901 falsos positivos e 394 falsos negativos. Isso significa que, das 10.000 imagens, 8.705 foram classificadas corretamente, identificando rachaduras em imagens que realmente tinham defeitos e não identificando anomalias em imagens que realmente não as possuíam.

A precisão, que indica a proporção de imagens classificados como positivas que realmente são positivas, teve resultado de 83,6%. Isso significa que o modelo teve um bom desempenho na detecção de verdadeiros positivos, minimizando assim a ocorrência de falsos positivos. Em suma, significa que o modelo identificou as anomalias em imagens que realmente possuíam anomalias em 83,6% dos casos.

O *recall* foi de 92,1%, que mede a proporção de instâncias positivas corretamente identificadas pelo modelo em relação ao total de instâncias positivas total. Nesse caso, seria a quantidade de imagens classificadas com anomalias de todas aquelas que possuem anomalias realmente. F1-Score é uma medida que realiza média harmônica da precisão e do *recall*, juntado-os em uma única métrica e permitindo avaliar a o equilíbrio entre eles. Neste modelo o F1-Score foi de 87,6%.

A Figura 13 demonstra um exemplo de funcionamento da rede neural, apresentando algumas imagens originais, que foram utilizadas como entrada para o modelo, e

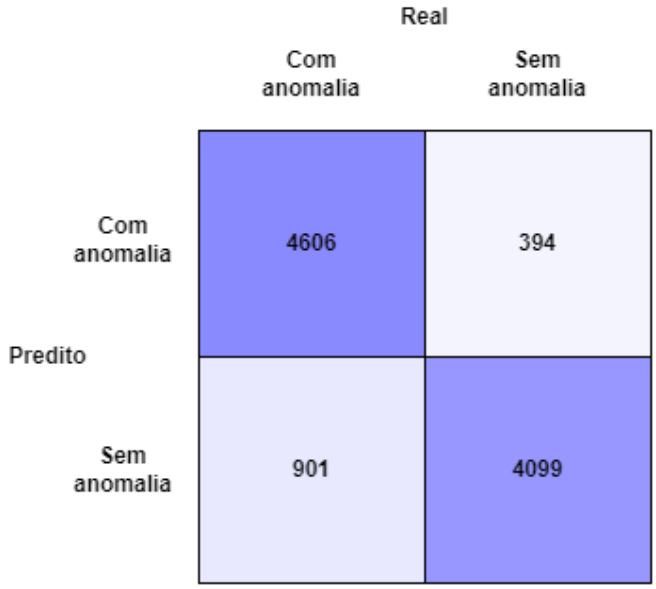

Figura 12. Matriz de confusão do melhor resultado

as imagens reconstruídas, que são as que a rede refez com base nos pesos obtidos no treinamento. Percebe-se a eficácia do *autoencoder*, sendo difícil identificar falhas na reconstrução, demonstrando o bom funcionamento do modelo.

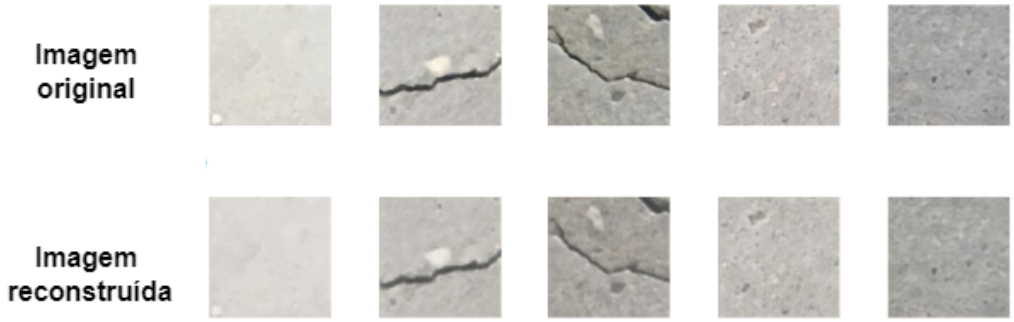

Figura 13. Comparação da imagem original com a imagem reconstruída

7. Considerações Finais e Trabalhos Futuros

A detecção de anomalias tem se tornado um tema relevante no campo da tecnologia, pois as anomalias podem variar desde pequenos desgastes em peças mecânicas até a falta de pedaços em cápsulas de remédios. Nesse cenário, a tecnologia surge como uma aliada, auxiliando na detecção dessas falhas para prevenir erros de fabricação de equipamentos, fraudes e outros problemas relacionados [Božič et al. 2021].

Pode-se considerar como uma anomalia todas as rachaduras e defeitos em barragens de concretos. Uma barragem é uma construção artificial feita com objetivo de represar água para diversos fins, desde geração de energia até abastecimento de água. Uma rachadura pode se tornar muito grave em tal cenário, já que pode ocasionar o rompimento dessas estruturas e gerar graves acidentes tanto para a população local

como para a fauna e flora. Identificar tais problemas é uma forma de prevenir maiores acidentes, garantindo que as construções se mantenham o mais seguras possível [Globo 2021, Tainara Messias dos Santos 2019].

Com isso em mente, foi realizada uma revisão da literatura, encontrando trabalhos similares de detecção de anomalias. Estes estavam relacionados com a detecção em produtos confeccionados em fábricas, tais como peças de plástico, metal e remédios, não tendo sido encontrados trabalhos sobre barragens. Apesar disso, foi possível generalizar os métodos observados nas obras para a aplicação específica de detecção de anomalias em paredes de concreto, assim, utilizando o modelo para o caso das barragens. Dentre as diversas formas de detectar anomalias, as redes neurais *autoencoders* foram as mais recorrentes encontradas na revisão, sendo um modelo de Inteligência Artificial capaz de desconstruir e reconstruir imagens [Zavrtanik et al. 2021, Fuchs et al. 2021].

Uma rede *autoencoder*, nesse cenário, utiliza imagens de estruturas de concreto em bom estado para o treinamento. A rede analisa tais informações, para ao final da execução, reconstruir a imagem com base nos dados adquiridos, comparando ambas as figuras. Imagens que não possuem anomalias tem uma taxa de erro menor na reconstrução, e aquelas com anomalias possuem uma taxa maior. Dessa forma é possível identificar se a imagem possui uma anomalia comparando a taxa de erro da imagem com a taxa de erro média da rede [Tensorflow 2023, Foster 2019a].

Para o desenvolvimento do modelo, foi-se selecionado o *dataset Concrete Crack Images for Classification*, que contém 40.000 imagens de barragens, metade com rachaduras e a outra metade sem. A rede foi treinada utilizando 12.500 imagens de estruturas de concreto sem defeitos e foram utilizadas 2.500 sem anomalia para definir a taxa de erro média da rede. Por fim, o modelo foi avaliado utilizando 10.000 imagens, sendo dessas, 5.000 com anomalias e 5.000 sem anomalias.

Com isso, obteve-se uma acurácia de 87%, uma precisão de 83,6% e um *recall* de 92,1%, por fim gerando um F1-Score de 87,6% no melhor resultado. Utilizando essas métricas de avaliação, pode-se concluir que é possível o uso das redes neurais *autoencoder* na detecção de anomalias. Entretanto, é necessária a utilização de imagens de barragens reais antes da aplicação em casos do mundo real, para verificar sua eficácia prática em tais problemas. Existe uma grande dificuldade na obtenção de tais imagens, já que elas não estão disponíveis para o público e o acesso a barragens são restritos, necessitando de permissão de órgãos públicos ou privados para obtenção das fotos. Assim, mesmo o trabalho apresentando uma boa acurácia em casos simulados, para aplicações reais ainda deve-se fazer testes e experimentos que comprovem sua eficiência.

Referências

Academy, D. S. (2022). Deep learning book.

An, J. and Cho, S. (2015). Variational autoencoder based anomaly detection using reconstruction probability. *Special lecture on IE*, 2(1):1–18.

André Luis da Cunha Dantas Lima, V. M. A. E. G. S. N. (2019). Manutenção preditiva aplicada a ambientes de missão crítica de supercomputação utilizando inteligência artificial: Uma revisão sistemática de literatura.

Antonelli, D. (2011). Rachaduras põem em dúvida obra de mauá.

Bergmann, P., Fauser, M., Sattlegger, D., and Steger, C. (2019). Mvtec ad – a comprehensive real-world dataset for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Božič, J., Tabernik, D., and Skočaj, D. (2021). Mixed supervision for surface-defect detection: From weakly to fully supervised learning. *Computers in Industry*, 129:103459.

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3).

Developers, G. (2022). Classificação: curva roc e auc.

Doersch, C. (2021). Tutorial on variational autoencoders.

ETESCO (2021). O que são barragens? conheça as principais características.

Foster, D. (2019a). *Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play*. O'Reilly Media, first edition.

Foster, D. (2019b). Generative deep learning teaching machines to paint, write, compose and play.

Fuchs, P., Kröger, T., and Garbe, C. S. (2021). Defect detection in ct scans of cast aluminum parts: A machine vision perspective. *Neurocomputing*, 453:85–96.

Globo, M. (2021). Tragédia em mariana (mg).

Kingma, D. P. and Welling, M. (2022). Auto-encoding variational bayes.

Kramer, M. A. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report.

Lin, N. (2022). Quase 200 barragens estão em situação crítica no país, segundo a ana.

Lu, C., Shi, J., and Jia, J. (2013). Abnormal event detection at 150 fps in matlab. In *2013 IEEE International Conference on Computer Vision*, pages 2720–2727.

Luger, G. (2013). *Inteligência Artificial*. PEARSON BRASIL.

Mahadevan, V., Li, W., Bhalodia, V., and Vasconcelos, N. (2010). Anomaly detection in crowded scenes. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1975–1981.

Mathworks (2022). Detect image anomalies using explainable one-class classification neural network.

Rodrigues, V. (2019). Métricas de avaliação: acurácia, precisão, recall... quais as diferenças?

Santana, M. (2018). Deep learning: do conceito às aplicações.

Severstal (2023). Severstal: Steel defect detection.

Shi, Y., Yang, J., and Qi, Z. (2021). Unsupervised anomaly segmentation via deep feature reconstruction. *Neurocomputing*, 424:9–22.

Sikka, M. (2023). Using skip connections to enhance denoising autoencoder algorithms.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition.

Stahl, A. (2022). The rise of artificial intelligence: Will robots actually replace people?

Stallkamp, J., Schlipsing, M., Salmen, J., and Igel, C. (2011). The german traffic sign recognition benchmark: A multi-class classification competition. In *The 2011 International Joint Conference on Neural Networks*, pages 1453–1460.

Tabernik, D., Šela, S., Skvarč, J., and Skočaj, D. (2019). Segmentation-Based Deep-Learning Approach for Surface-Defect Detection. *Journal of Intelligent Manufacturing*.

Tainara Messias dos Santos, I. C. d. P. (2019). Segurança de barragens: Uma revisão sistemática acerca das possíveis causas de rupturas em barragens e suas consequências.

Tchilian, F. (2022). Modelo preditivo: o que é, para que serve e como aplicá-lo?

Tensorflow (2023). Introdução aos codificadores automáticos.

Weimer, D., Scholz-Reiter, B., and Shpitalni, M. (2016). Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection. *CIRP Annals*, 65(1):417–420.

Xu, W., Jang-Jaccard, J., Singh, A., Wei, Y., and Sabrina, F. (2021). Improving performance of autoencoder-based network anomaly detection on nsl-kdd dataset. *IEEE Access*, 9:140136–140146.

Y. LeCun, C. Cortes, C. B. (2022). Mnist handwritten digit database.

Yang, J., Shi, Y., and Qi, Z. (2022). Learning deep feature correspondence for unsupervised anomaly detection and segmentation. *Pattern Recognition*, 132:108874.

Zavrtanik, V., Kristan, M., and Skočaj, D. (2021). Reconstruction by inpainting for visual anomaly detection. *Pattern Recognition*, 112:107706.

Zhou, Y., Liang, X., Zhang, W., Zhang, L., and Song, X. (2021). Vae-based deep svdd for anomaly detection. volume 453, pages 131–140.