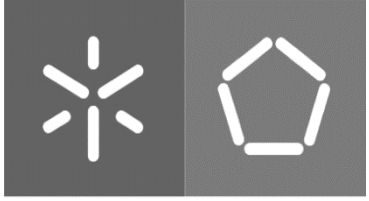
Universidade do Minho
Escola de Engenharia

João Pedro Martins Barros Costa Leite

Exploração das GANs na Geração de Imagens do Interior de Veículos

Guimarães, Março 2021



Universidade do Minho
Escola de Engenharia

João Pedro Martins Barros Costa Leite

Exploração das GANs na Geração de Imagens do Interior de Veículos

Dissertação de Mestrado em Engenharia Eletrónica Industrial e Computadores

Trabalho efetuado sob a orientação do
Professor Doutor Jaime Fonseca
Doutor Sandro Queirós

Guimarães, Março 2021

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada.

Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

Licença concedida aos utilizadores deste trabalho

Atribuição
CC BY

https://creativecommons.org/licenses/by/4.0/

AGRADECIMENTOS

Primeiro, quero agradecer ao meu orientador, professor doutor Jaime Fonseca, por me ter dado a oportunidade de realizar esta dissertação e por me ter orientado, juntamente com o meu coorientador doutor Sandro Queirós, a quem também agradeço o apoio e partilha de conhecimentos ao longo do desenvolvimento.

Também quero agradecer a toda equipa que trabalhou comigo em laboratório, especialmente ao doutor João Borges, pelo acompanhamento e ajuda durante todo o trabalho.

Agradeço, por fim, a toda minha família, à minha companheira, e amigos, por me apoiarem, motivarem e acreditarem em mim durante estes longos anos que passaram.

This work is supported by: European Structural and Investment Funds in the FEDER component, through the Operational Competitiveness and Internationalization Programme (COMPETE 2020) [Project nº 039334; Funding Reference: POCI-01-0247-FEDER-039334]

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

RESUMO

Tendo em conta a evolução futura dos veículos autónomos partilhados (SAV, do inglês *Shared Autonomous Vehicles*), novos sistemas de monitorização no interior do veículo serão implementados. Com o foco na segurança do veículo e dos passageiros, nomeadamente na deteção de objectos perigosos na posse dos passageiros ou perdidos no interior do veículo, será necessário o desenvolvimento de algoritmos capazes de detetar estes mesmos objetos de forma eficiente e robusta. Atualmente, estes algoritmos são desenvolvidos com recurso a técnicas de visão por computador, mais propriamente métodos baseados em *machine learning* e *deep learning*. Estes últimos, apesar de demonstrarem excelentes resultados, requerem grandes quantidades de dados para o seu treino, devendo estes conter a maior variedade possível. A obtenção deste *dataset* é assim um processo demorado e dispendioso, sendo vantajoso o desenvolvimento de algoritmos capazes de gerar imagens artificiais realistas. Nos últimos anos, novas técnicas, também baseadas em *deep learning*, têm vindo a ser propostas para este fim, nomeadamente as baseadas em *Generative Adversarial Networks* (GANs). Neste projeto, explorou-se o uso destes algoritmos para a geração automática de imagens artificiais do interior de veículos, para uso futuro no treino de algoritmos de deteção de objetos e monitorização no contexto dos SAVs.

Após um estudo inicial do estado da arte e avaliação de diversas arquiteturas de GANs, focou-se o trabalho na BigGAN, uma rede condicional que apresentou bons resultados para imagens de alta resolução, de *datasets* complexos. De forma a mitigar a existência de um número reduzido de imagens para o cenário em causa, optou-se por utilizar *datasets* públicos de grandes dimensões no treino da rede, sendo as imagens do interior de veículos apenas uma (ou algumas) das classes geradas pela rede. Partindo deste pressuposto, testaram-se duas resoluções de imagem, parametrizações da rede e abordagens de pré-processamento. Para além disso, testou-se a combinação desta rede com 2 técnicas recentes que visam melhorar a estabilidade do treino e a capacidade de generalização da GAN, nomeadamente a *consistency regularization* (CR) e a *differentiable augmentation*. No geral, obtiveram-se resultados satisfatórios com a abordagem proposta, sendo que o melhor resultado (qualitativo e quantitativo) foi obtido com a aplicação de CR-BigGAN, utilizando *flips* e translações para aumentar artificialmente o *dataset*, com um *Fréchet Inception Distance* (FID) de 28,23 e um *Inception Score* (IS) de 17,19.

Em suma, este trabalho demonstrou o potencial que as GANs poderão ter na geração automática de imagens de interior de veículos, de modo a aumentar os *datasets* disponíveis para o treino de algoritmos de inteligência artifical no contexto dos SAVs. Com maior poder computacional e mais recursos, seria possível obter resultados melhores, aumentando a capacidade das redes (isto é, tornando-as maiores e mais complexas) e processando uma maior quantidade de dados ao mesmo tempo.

Palavras Chave: Shared Autonomous Vehicles, Geração de imagens, Deep learning, Generative Adversarial Networks

ABSTRACT

In view of the future evolution of Shared Autonomous Vehicles (SAVs), new in-car monitoring systems will have to be implemented. With the focus on vehicle and passenger safety, namely the detection of dangerous objects held by passengers or lost inside the vehicle, it will be necessary to develop algorithms capable of detecting these same objects in an efficient and robust way. Currently, these algorithms are developed using computer vision techniques, more specifically methods based on machine learning and deep learning. These, although demonstrating excellent results, require large amounts of data for their training and should contain as much variety as possible. Obtaining this dataset is therefore a time-consuming and expensive process, and the development of algorithms capable of generating realistic artificial images is advantageous. In recent years, new techniques, also based on deep learning, have been proposed for this purpose, namely those based on Generative Adversarial Networks (GANs). In this project, the use of these algorithms for the automatic generation of artificial images of the vehicle interior, for future use in the training of object detection and monitoring algorithms in the context of SAVs, was explored.

After an initial study of the state-of-the-art and evaluation of various GAN architectures, work was focused on BigGAN, a conditional neural network that showed good results for high-resolution images of complex datasets. In order to mitigate the existence of a small number of images for the scenario in question, it was decided to use large public datasets in the training of the network, with images of the interior of vehicles being only one (or some) of the classes generated by the network. Based on this assumption, different image resolutions, network parameterizations and pre-processing approaches were tested. In addition, the combination of this network with 2 recent techniques, namely the consistency regularization (CR) and the differentiable augmentation, aimed at improving the stability of the training and the generalization capacity of the GAN. Overall, satisfactory results have been achieved with the proposed approach, with the best result (qualitative and quantitative) being obtained with the application of CR-BigGAN, using flips and translations as data augmentation during training, with a Fréchet Inception Distance of 28.23 and an Inception Score of 17.19.

In short, this work has demonstrated the potential that GANs may have in the automatic generation of vehicle interior images, in order to increase the size of the datasets available for the training of artificial intelligence algorithms in the context of SAVs. With more computational power and more resources, better results could be achieved by increasing the capacity of the networks (i.e. making them bigger and more complex) and processing a larger amount of data at the same time.

Keywords: Shared Autonomous Vehicles, Image Generation, Deep Learning, Generative Adversarial Networks

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

1

INTRODUÇÃO

Este documento trata uma dissertação realizada no âmbito do Mestrado Integrado em Engenharia Eletrónica Industrial e Computadores (MIEEIC) na Universidade do Minho. Neste capítulo é introduzido o contexto em que se enquadra o projeto, a sua necessidade no avanço tecnológico associado ao contexto e os seus contributos na área.

1.1 Contexto

Com a evolução tecnológica associada à mobilidade e autonomia, com maior responsabilidade associada ao crescimento da Inteligência Artificial, deparamo-nos numa fase que poderá ter um grande impacto na forma como vemos os meios de transporte nos dias de hoje. Numa era em que existe uma constante sensibilização quanto ao impacto ambiental das tecnologias atuais e uma constante preocupação quanto à segurança rodoviária, ou falta dela, deparamo-nos numa possibilidade de evolução tecnológica, os veículos autónomos partilhados (*SAV*, do inglês *Shared Autonomous Vehicles*). Este conceito permitiria combater os dois problemas acima mencionados: utilizar um só veículo para realizar viagens com diversas pessoas, muitas vezes totalmente desconhecidas, reduzindo a quantidade de veículos a circular nas estradas e, consequentemente, o impacto ambiental produzido pela quantidade de carros que circulam nas estradas; e condução destes veículos de maneira autónoma, com recurso a Inteligência Artificial, permitindo assim reduzir o risco de acidente associado ao erro humano.

Este novo paradigma cria necessidades de monitorização do interior dos veículos, estando esta focada em garantir a segurança do veículo e dos passageiros. No que diz respeito à segurança do veículo e dos passageiros no contexto de partilha nos SAVs, é relevante a deteção de objetos potencialmente perigosos, capazes de ferir passageiros. Por outro lado, é relevante identificar também possíveis objetos perdidos no interior do veículo, podendo associar estes ao proprietário que realizou uma viagem anteriormente.

Posto isto, surge a necessidade de desenvolver algoritmos capazes de detetar objetos dentro dos veículos. Estes algoritmos que procuram detetar os diversos objetos são implementados com recurso a visão por computador, sendo que atualmente se tem dado mais foco aos baseados em técnicas de *deep learning* (*DL*). Estas técnicas apresentam melhores resultados quando comparados com técnicas associadas a *machine learning* (*ML*) ou outras ditas tradicionais. No entanto, estas técnicas requerem grandes quantidades de dados para o treino, isto é, precisam de um *dataset* com uma grande variedade e quantidade de casos.

1.2 Motivação

Atualmente, um dos grandes problemas na aplicação de métodos baseados em Inteligência Artificial está na falta de dados que permitam aplicar os métodos com sucesso. Esta limitação é particularmente crítica quando se trata de métodos orientados à visão por computador, pois existe um grande número de possibilidades e variáveis, desde cores, brilho, iluminação, entre outros. Isto torna muito complicado generalizar um método para que seja capaz de abordar todas as situações que podem ocorrer ao longo de um dia.

Analisando o interior de veículos, consideram-se algumas variáveis, facilmente identificáveis, como por exemplo a diferença do interior de modelo para modelo, a diferença de cores no interior, a diferença de luminosidade na situação do dia para a noite. Deste modo, existe uma dificuldade real em criar sistemas de monitorização robustos para suportar todas estas variantes.

Surge então a necessidade de obter um conjunto vasto e complexo de dados, que permita abordar todas as possibilidades supra referidas. No entanto, a obtenção de um *dataset* é sempre um processo demorado e dispendioso. Este processo envolve preparação, recolha e tratamento de dados, tirando muito tempo que poderia ser utilizado no desenvolvimento tecnológico e não na sua preparação.

Recentemente, têm surgido novas técnicas/algoritmos capazes de gerar dados orientados a visão (imagens), baseados em métodos de Inteligência Artificial, nomeadamente as *Generative Adversarial Networks* (*GANs*).

1.3 Objetivos e contribuições da dissertação

Foi objetivo desta dissertação estudar e avaliar diferentes GANs para a geração de imagens do interior de veículos. Para isto, numa fase inicial, foi necessário recolher uma quantidade significativa de dados que permitissem servir de base para o nosso estudo e analisar até que ponto a partir de um certo conjunto de imagens seria possível gerar novas imagens. Numa segunda fase, estudar as diferentes *GANs* existentes e selecionar aquela que, tendo em conta os recursos disponíveis, poderia ser aplicada ou testada na geração de imagens. Numa fase final, realizou-se a geração de imagens recorrendo à GAN selecionada e ao *dataset* criado (figura 1).

Esta dissertação contribui com múltiplos *datasets* para o desenvolvimento de GANs no contexto de monitorização do interior dos SAV. Mais ainda, é proposta uma abordagem para a geração destes *datasets*, com o recurso a diferentes técnicas do estado de arte. Este método pode ser aplicado para diferentes *datasets* permitindo ao utilizador explorar outro tipo de geração de imagens que não o interior de veículos. Graças aos inúmeros testes realizados, este trabalho apresenta um estudo dos cuidados e meios a considerar na utilização destas técnicas, nomeadamente em aplicações nas quais a quantidade de dados disponível é limitada.

1.4 Estrutura da dissertação

Esta dissertação está dividida em cinco capítulos. Estes organizam-se sequencialmente de modo a que o leitor consiga compreender o trabalho desenvolvido, acompanhando o processo realizado.

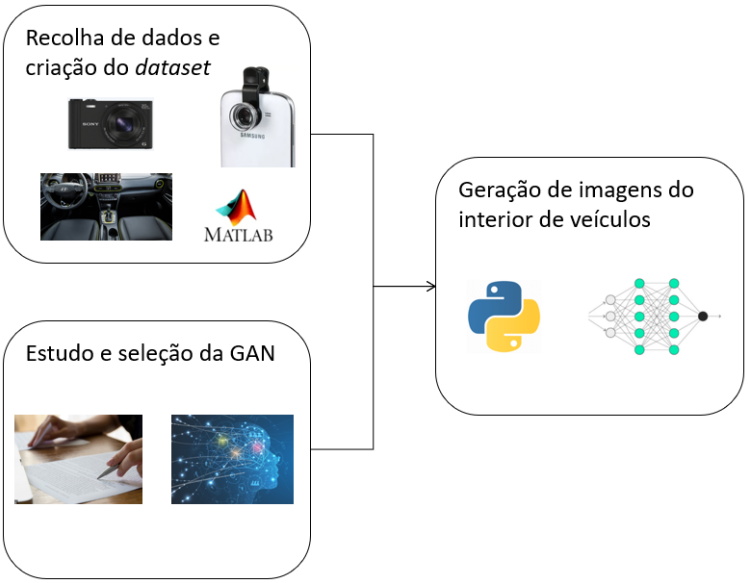


Figura 1: Representação do processo seguido para a geração de imagens com recurso a GANs.

O primeiro capítulo, Introdução, enquadra o leitor naquilo que foi desenvolvido, na motivação e objetivo da dissertação e a sua estrutura.

O segundo capítulo, Estado de Arte, explora os diferentes conteúdos associados ao projeto. Neste capítulo, é apresentada uma revisão bibliográfica focada nas GANs e noutros métodos associados à recolha e aumento da quantidade de dados.

O terceiro capítulo, Métodos, apresenta detalhadamente os algoritmos, materiais e técnicas que foram utilizados para a geração do *dataset*. Apresenta também a GAN selecionada para geração automática de imagens, bem como outras técnicas para melhoria do treino. São, ainda, apresentadas as métricas, com maior detalhe, que serão utilizadas na análise e avaliação dos resultados.

O quarto capítulo, Testes e Resultados, expõe os diferentes ensaios realizados e os resultados, qualitativos e quantitativos, obtidos. Neste capítulo, também se realiza uma análise e discussão dos resultados.

O quinto capítulo, Conclusões e Trabalho Futuro, apresenta as conclusões retiradas ao trabalho realizado, bem como diferentes possibilidades a implementar futuramente partindo do que foi desenvolvido até ao momento.

2

ESTADO DE ARTE

2.1 Veículos autónomos partilhados

Recentemente, a indústria automóvel e as companhias de *software* apresentaram protótipos para veículos autónomos e que serão introduzidos no mercado num futuro muito próximo. A maior característica que esta tecnologia apresenta é a capacidade da navegação do veículo ser totalmente independente, ou seja, não necessitar de um condutor para controlar o veículo. Sendo assim, vai permitir considerar os condutores como passageiros, os quais poderão realizar outras atividades (dormir, ler, trabalhar, etc.) sem a necessidade de se focar na condução. Começou então a surgir o conceito de veículos autónomos.

Com a sua evolução, surgiu uma ideia de negócio, os SAVs. O conceito de SAV surge da ligação de *carsharing* com serviços de táxi e veículos autónomos. Isto permite associar os diferentes estados de mobilidade atual, juntando tecnologia, economia e autonomia no mesmo conceito. Observando a figura 2, identificam-se as características que estão por base da mobilidade atual para cada estado de mobilidade.

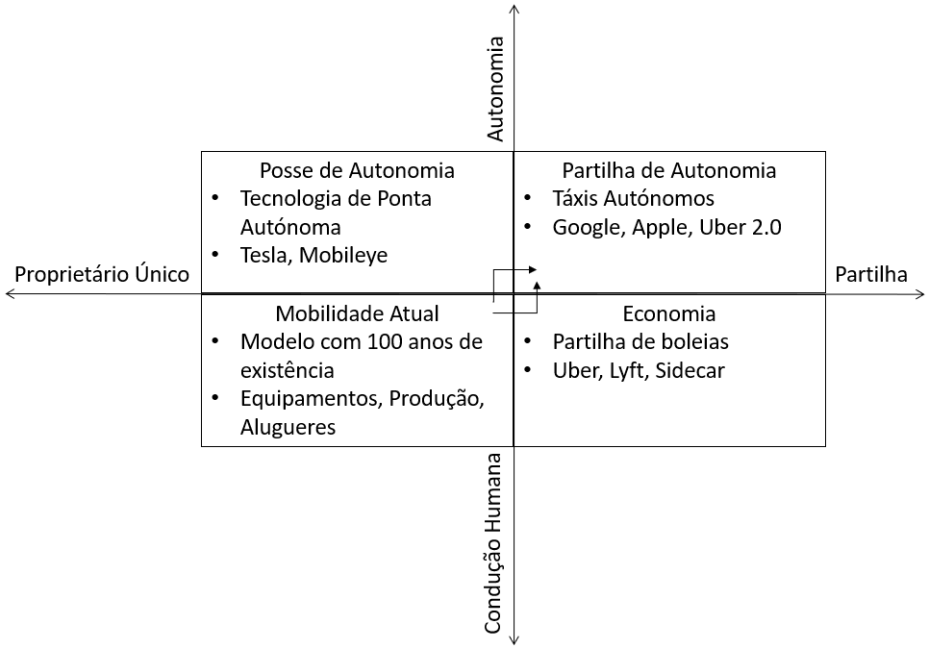


Figura 2: Diferentes estados de mobilidade. Imagem adaptada de Mingyang (2019).

Os SAVs podem proporcionar serviços de transporte convenientes a nível monetário e ambiental. Este conceito permite reduzir o número de veículos em trânsito, trazendo diversas vantagens. A nível ambiental,

a adoção desta opção de mobilidade permite reduzir emissões e diminuir a poluição, graças à redução do número de veículos a circular. A nível monetário, reduz-se nas despesas diárias comuns dos utilizadores, associadas a combustível, parquímetros e manutenção do veículo. A longo prazo, pode reduzir a compra de veículos, associada à falta de necessidade de ter um veículo próprio. Também a longo prazo e com um grande desenvolvimento nesta tecnologia, os próprios veículos podiam poupar muito tempo em viagens, comunicando entre si os seus percursos e assim adaptando os seus percursos para evitar trânsito (*Dynamic-Ride Sharing*) (Krueger et al., 2016).

A introdução deste negócio no mercado levanta várias questões. Sendo algo que pode ser utilizado por vários utilizadores, têm que ser garantidas certas políticas de utilização, onde surgem certas situações que se precisam de salvaguardar. Uma delas passa pela segurança dos utilizadores e do próprio veículo. Deste modo, é necessário sistemas de monitorização para controlar as diferentes situações e detetar casos que possam pôr em causa a integridade do veículo ou dos passageiros. Estas situações podem ser detetadas através da deteção de objetos considerados perigosos ou de gestos considerados ameaçadores, sendo necessário intervir nessas situações. No caso de deteção de objetos, também se podem incluir casos em que se perdem objetos. Quando se perdem objetos e sendo estes detetados no interior do veículo, é possível saber-se quem foram os últimos utilizadores do veículo e assim, poder-se devolver os objetos ao seu dono.

Estas novas necessidades requerem portanto o desenvolvimento de sistemas de monitorização, baseados em sensores de imagem (por exemplo, RGB, NIR, Depth), os quais recorrem a algoritmos de processamento de imagem para inferir a presença de objetos. Uma possibilidade no desenvolvimento destes algoritmos é o uso de técnicas de DL, as quais permitem obter uma precisão elevada e um baixo tempo computacional.

2.2 Deep learning

Ao longo dos anos, tem existido uma grande evolução na área de inteligência artificial (AI, do inglês *Artificial Intelligence*). A AI tem várias aplicações e várias áreas de investigação ainda por explorar e terá provavelmente uma expansão ainda maior no futuro próximo. No entanto, já se observam diferentes aplicações no dia-a-dia que recorrem a estas tecnologias.

Com a constante evolução tecnológica nesta área, tem-se tentado diversificar as áreas aonde esta é aplicada, focando-se nomeadamente em cenários cada vez mais complexos. Contudo, a aplicação desta tecnologia em cenários muito específicos acarreta diversos problemas, sendo que um dos principais é a falta de dados relativos ao processo a ser implementado, os quais são fundamentais para a implementação de técnicas de AI robustas e generalistas.

Observando a figura 3, DL constitui uma pequena parte daquilo que é AI. DL é um método de *representation learning* que consegue extrair a um nível alto características muito mais abstratas do que outros métodos. É um subconjunto de ML e recorre a redes neuronais artificiais profundas. A máquina na qual é aplicada a rede neuronal usa diferentes camadas para aprender características de uma certa representação de um conjunto de dados.

Como ilustrado na figura 4, um processo associado a ML usa como dados de entrada as características/informações previamente extraídas dos dados de entrada primários por um utilizador, baseando-se em regras definidas por este. Já o DL trabalha diretamente sobre os dados de entrada

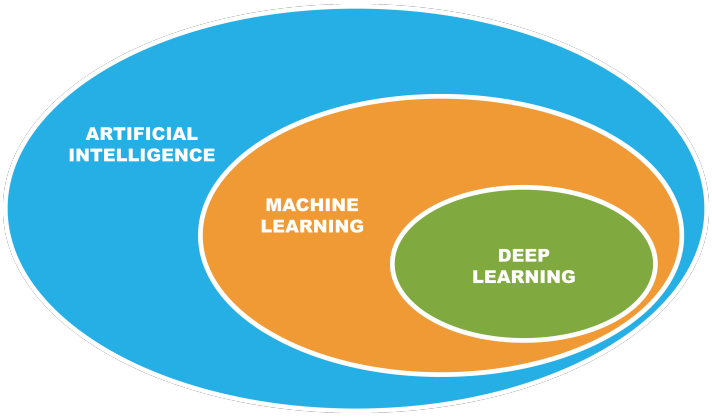


Figura 3: Sub-áreas de Inteligência Artifical. Imagem adaptada de PIKKART.

primários, extraindo automaticamente as características que melhor os descrevem, e procede de seguida ao seu processamento.

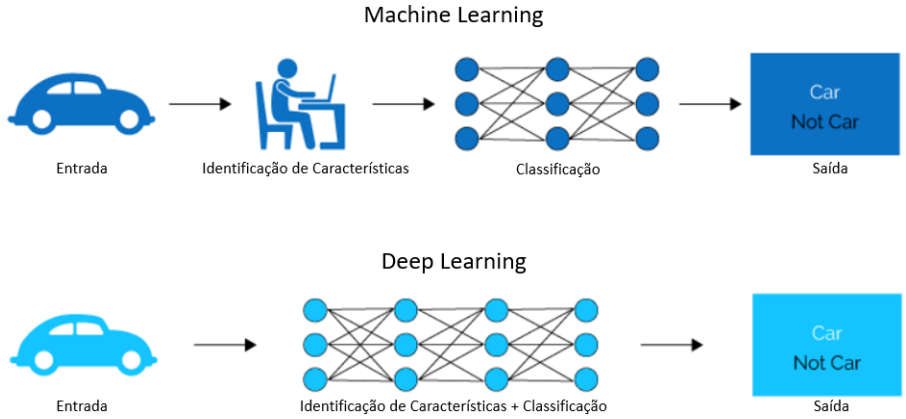


Figura 4: *Machine Learning* vs *Deep Learning*. Imagem adaptada de Coelho (2019).
A ideia de DL passa por a rede ser capaz de extrair automaticamente as características e proceder à classificação. No caso de ML, a extração é efetuada com base em regras definidas pelo desenvolvedor/utilizador, enquanto que a rede apenas aprende a classificar.

Uma iteração num processo de treino passa por uma rede neuronal receber uma entrada na sua primeira camada (*input layer* na figura 5) e aplicar as diferentes transformações nas camadas que lhe seguem (*hidden layer* na figura 5). Ao longo de um processo de treino, é possível treinar a rede para realizar um certo propósito (geração de amostras ou classificação de dados) com sucesso, apresentando o resultado deste processo (*output layer* na figura 5). Deste modo, depois de a rede ser bem treinada, esta é capaz de operar para qualquer distribuição de dados, com características tais que sejam semelhantes aos dados de entrada com que foi treinada.

2.3 Aprendizagem supervisionada vs não supervisionada

Na AI, especificamente no ML, podem-se destacar duas categorias para caracterizar os modelos, modelos supervisionados e modelos não supervisionados.

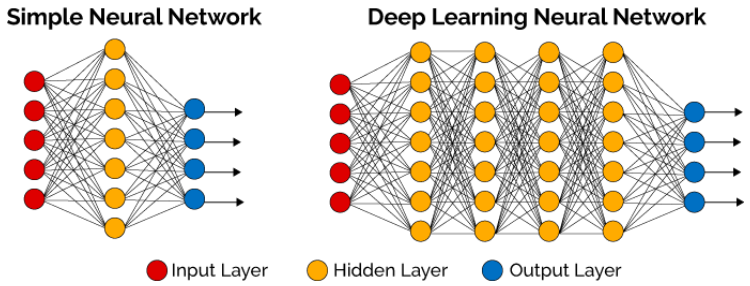


Figura 5: Diferença entre uma rede neuronal simples para uma rede neuronal baseada em *Deep Learning*. Imagem adaptada de Coelho (2019).
A principal diferença está no número de camadas que constituem a rede.

Num modelo baseado em aprendizagem supervisionada, o treino realiza-se com recurso a *datasets* que contemplam dados de entrada e também de saída, isto é, com *datasets* já anotados, sendo que o método de aprendizagem visa estabelecer relações entre entradas e saídas, de modo a ser capaz de depois inferir sobre a saída para dados de entrada previamente desconhecidos. Estes métodos têm uma percentagem elevada de acerto mas o método de aprendizagem obriga à existência de um *dataset* no qual as variáveis de saída (por exemplo, classificação ou regiões de interesse) são conhecidas (Goodfellow et al., 2016).

Quanto a um modelo baseado em aprendizagem não supervisionada, a rede neuronal foca-se em aprender a distribuição dos dados de entrada, para assim ser capaz de encontrar diferenças e/ou similaridades entre amostras destes dados ou conseguir novas representações para estes mesmos dados (podendo depois focar-se na descrição dos dados por novas características ou na redução da complexidade/quantidade das características originais). (Goodfellow et al., 2016)

Na tarefa de aprendizagem não supervisionada, destacam-se os modelos generativos. Estes modelos permitem representar distribuições de dados, onde estes dados têm um certo número de características semelhantes ao *dataset* usado para treinar a rede. Os modelos generativos são normalmente baseados em cadeias de *Markov, maximum likelihood* (Goodfellow et al., 2016) e *approximate inference*. O modelo base é o *Restricted Boltzmann Machine* (Palit, 2010). Deste modelo, baseando-se em *maximum likelihood*, surgiram as *Deep Belief Networks* (Salakhutdinov and Hinton, 2009) e as *Deep Boltzmann Machine* (Palit, 2010; Salakhutdinov and Hinton, 2009; Eslami et al., 2014). Na figura 6, observam-se as principais características que permitem distinguir estes modelos uns dos outros. Estes modelos apresentavam sérias limitações, o que levou à sua evolução. Resultados obtidos aplicando estes modelos apesar de satisfatórios, foram realizados em *datasets* básicos e sem complexidade o que seria algo a melhorar e evoluir para a geração de melhores imagens (Denton et al., 2015). Atualmente, podem-se distinguir os modelos generativos em três categorias principais: GANs, VAE (*Variational Autoencoder* (Kingma and Welling, 2013; Rezende et al., 2014)) e *AutoRegressive Networks*. Os modelos VAE geram imagens mais desfocadas comparativamente com as GANs. Já as *AutoRegressive Networks*, mais propriamente as PixelRNN, geram uma imagem píxel a píxel, correspondendo a um processo muito demorado, enquanto as GANs geram uma imagem muito mais rapidamente.

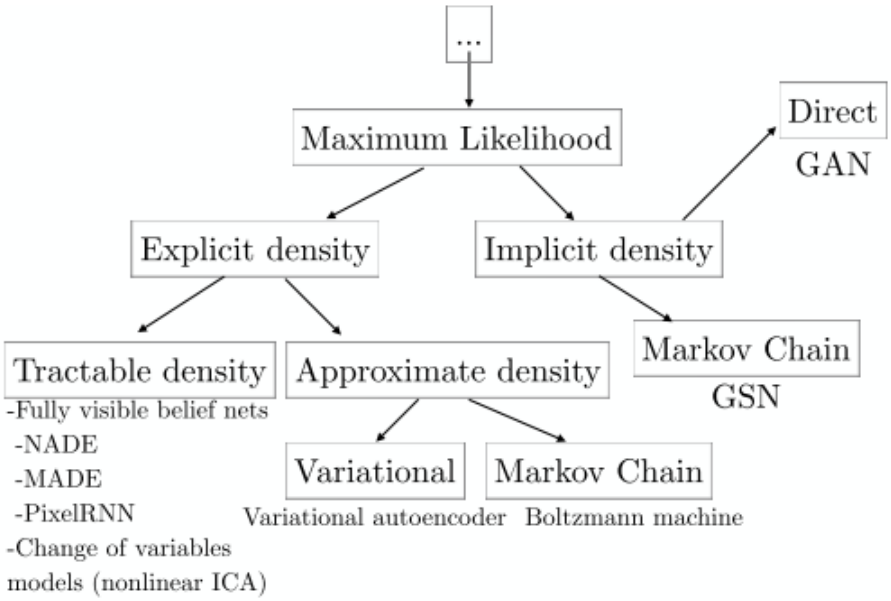


Figura 6: Modelos generativos baseados em ML. Imagem adaptada de Goodfellow (2016).

2.4 Generative Adversarial Networks

Para abordar os diferentes problemas mencionados acima, os modelos generativos revelam-se como uma alternativa para ultrapassar alguns obstáculos. No caso de imagens, os modelos generativos tornam-se capazes de gerar imagens a partir de um conjunto de dados previamente fornecido, sendo assim possível gerar novos dados.

Um modelo generativo devidamente treinado é capaz de representar e manipular distribuições de dados, como a sua transformação ou geração. Este modelos conseguem, através de um conjunto bastante limitado de dados, realizar previsões e gerar novos dados baseados naqueles já existentes. Permite também, igualmente deparado com um conjunto limitado de dados, identificar diferentes características nos dados, de tal modo que permite auxiliar o utilizador no *labelling* que distingue e identifica os dados, muitas vezes com pormenores imperceptíveis e difíceis de identificar.

Por estas razões, neste projeto, utilizou-se as GANs para a geração de imagens artificiais realistas permitindo assim aumentar o *dataset* mais rapidamente (Pan et al., 2019). As GANs são um dos modelos de redes generativas profundas, ou seja, são algoritmos capazes de gerar algo através de uma entrada aleatória. O seu potencial passa por aprender a imitar qualquer distribuição de dados nas quais sejam treinadas e, ao mesmo tempo, potenciar a sua capacidade de distinguir imagens verdadeiras de falsas dentro da distribuição de dados. Deste modo, podem-se distinguir duas redes neuronais, o gerador e o discriminador. Estas redes são treinadas até atingir um Equilíbrio de *Nash*, isto é, o treino continua até que ambas as redes atinjam dois níveis de treino que se possam considerar aceitáveis e satisfatórios consoante o objetivo do treino.

Numa GAN básica, como se observa na figura 7, o gerador recebe números aleatórios e gera uma imagem. A imagem gerada é atribuída como uma entrada no discriminador juntamente com o conjunto de imagens retiradas do conjunto de dados reais. O discriminador determina uma probabilidade da imagem gerada pertencer à distribuição de dados reais. Deste modo, pode-se treinar o discriminador para classificar

as imagens como pertencentes à distribuição real (probabilidade próxima de 1) ou à distribuição gerada (probabilidade próxima de 0). O gerador gera imagens através de entradas aleatórias e recebe feedback do discriminador, o que permite melhorar a imagem gerada e aproximá-la da distribuição real. Deste modo, ambas as redes competem entre si para atingir melhores resultados, podendo-se comparar com um jogo em que ambos são adversários, no qual competem um contra o outro para obter o melhor resultado possível de ambos. Por outras palavras, os pesos do gerador são atualizados para aumentar o erro final da classificação efetuada pela discriminador, enquanto os pesos do discriminador são atualizados para diminuir o erro final da classificação.

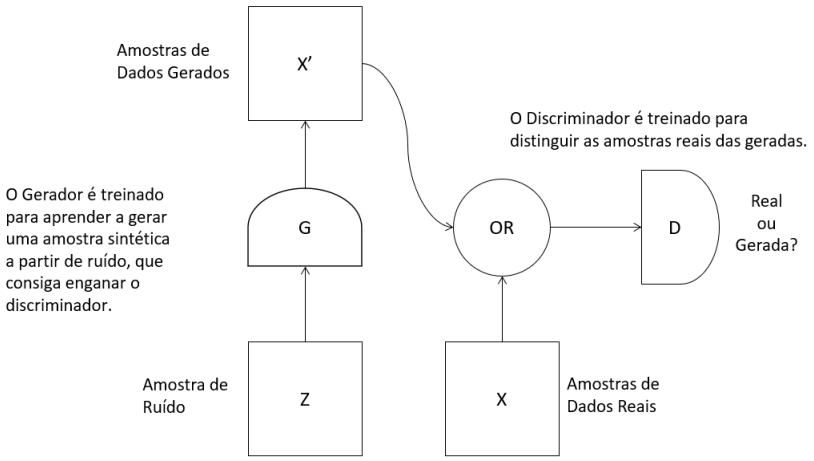


Figura 7: Processo da GAN. Imagem adaptada de Creswell et al. (2018).
O gerador (G) e o discriminador (D) são, normalmente, implementados com redes neuronais. O gerador recebe como entrada um vetor de valores aleatórios (Z) e gera uma imagem artificial (X'). Esta imagem artificial e as imagens reais (X) entram no discriminador e este visa distinguir as imagens reais das artificiais.

2.4.1 *Arquitetura base*

A primeira GAN, figura 8, usava *fully-connected neural networks*, tanto para o discriminador como para o gerador (Goodfellow et al., 2014). Esta descreve dois modelos, um modelo generativo G que gera uma imagem e um modelo discriminativo D que estima a probabilidade da imagem ser do *dataset* de treino ou do gerador. Antes das GANs, os modelos generativos apresentavam imagens geradas desfocadas e muito diferentes do *dataset* e apresentavam também problemas relacionados com a linearidade. Este novo modelo generativo propôs-se a resolver este problema.

Neste modelo, introduziu-se o conceito de redes adversárias (Goodfellow et al., 2014), isto é, ambas as redes treinam de modo a atingir objetivos opostos uma à outra. Este problema de otimização pode ser descrito pela seguinte equação:

$$min_G \sim max_D \sim V(D,G) = E_{x\sim pdata(x)}[logD(x)] + E_{z\sim p(z)}[log(1 - D(G(z)))], \quad (1)$$

sendo que D e G correspondem ao discriminador e ao gerador, respetivamente, E corresponde ao valor expectável da imagem pertencer à distribuição de dados real ou gerada, observando-se que o gerador procura minimizar esta função e o discriminador maximizar (opostos).

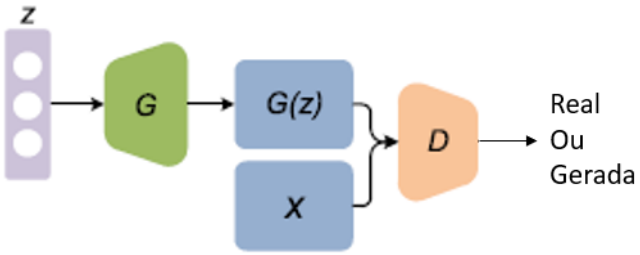


Figura 8: Arquitetura base de uma GAN. Imagem adaptada de Pan et al. (2019).
Z corresponde ao vetor aleatório, G ao gerador, G(Z) à imagem artificial, D ao discriminador e X às imagens reais.

Durante o treino, os parâmetros do gerador são atualizados ao mesmo tempo que os parâmetros do discriminador.

No inicio do treino, quando as imagens geradas ainda são fracas, o discriminador identifica as imagens falsas com muita confiança porque são muito diferentes do *dataset*. Como se observa no primeiro gráfico da figura 9, o discriminador distingue as distribuições uma da outra. Neste caso, a componente log(1-D(G(Z))) da equação 1 satura.

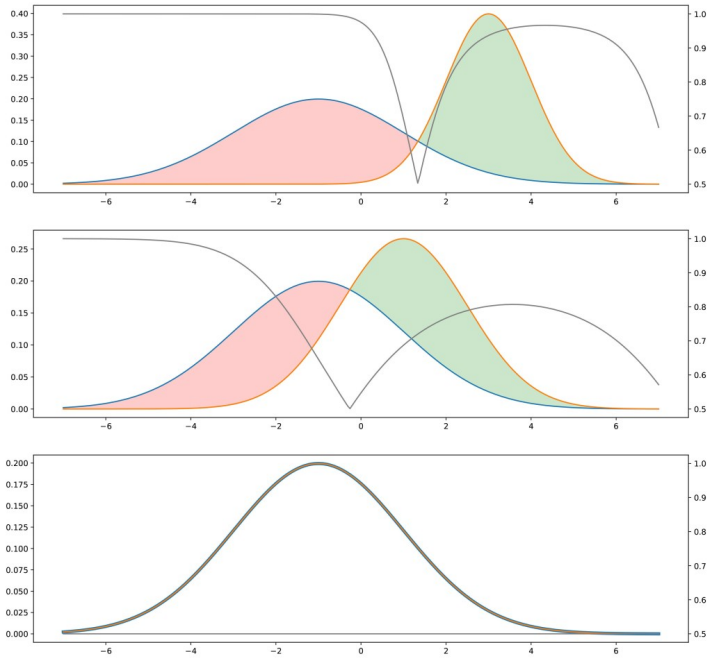


Figura 9: Evolução da distribuição de dados durante o treino. Imagem adaptada de Academy (2019).
Numa primeira fase (primeiro gráfico) observa-se que o discriminador consegue distinguir as duas distribuições de dados, a real (azul) e a artificial (laranja). Numa fase intermédia, o discriminador não tem a mesma acurácia que tinha numa fase inicial, pois à medida que as redes foram treinadas, as imagens artificiais aproximaram-se das reais sendo que o discriminador começou a ter mais dificuldade em distinguir a distribuição de dados. Na fase final do treino, quando o gerador atingiu um valor aceitável de otimização, as duas distribuições de dados são, idealmente, iguais o que leva à impossibilidade do discriminador distinguir imagens reais das geradas.

Durante o treino, a distribuição de dados gerada vai-se aproximando da distribuição de dados real (segundo gráfico da figura 9), o que leva a que o discriminador tenha mais dificuldade a distinguir as imagens.

Na fase final do treino, idealmente, o discriminador não consegue distinguir a distribuição real da gerada (terceiro gráfico da figura 9). Como o gerador tenta enganar o discriminador, idealmente chega-se a um ponto em que é muito difícil para o discriminador distinguir a distribuição verdadeira da falsa (Academy, 2019). Deste modo, o discriminador devolve valores aproximados de 0.5, pelo que se atinge o máximo de otimização (Pan et al., 2019).

Comparativamente com outros modelos generativos, nas GANs apenas se usa *back propagation* para se obter os valores dos gradientes, não sendo necessário fazer deduções e tendo grande flexibilidade no uso de funções. Porém, o gerador e o discriminador têm que estar sempre em sintonia durante o treino e tem que se manter o conhecimento dos gradientes ao longo do treino, entre cada passo.

Este modelo serviu de base para todas as GANs que lhe seguiram. As GANs que derivaram deste modelo podem-se distinguir em duas grandes categorias: as que procuram melhorar a arquitetura e as que procuram melhorar a função objetivo (Tabela 1).

Tabela 1: Classificação das GANs, adaptado de Pan et al. (2019)

Otimização da Arquitetura	Convolucionais	DCGAN
	Condicionais	CGAN; InfoGAN; ACGAN
	Autoencoder	AAE; BiGAN; ALI; AGE; VAE-GAN
Otimização da Função	unrolled GAN; f-GAN; Mode-Regularized GAN; Least-Square GAN; Loss-Sensitive GAN; EBGAN; WGAN; WGAN-GP; WGAN-LP	

2.4.2 *Deep Convolutional GANs*

As redes neuronais convolucionais (CNN, do inglês *Convolutional Neural Networks*) são dos melhores modelos de aprendizagem supervisionada e são um dos tipos de redes mais comuns no processamento de imagem e vídeo (para fins de classificação, deteção, segmentação, etc.).

Usando como exemplo a tarefa de classificação de imagens, tipicamente, os modelos CNN processam as entradas por um conjunto de camadas convolucionais e de *pooling* para a tarefa de *feature learning* (isto é, para a extração de características relevantes para a tarefa em causa), aplicando no fim um conjunto de camadas adicionais para pós-processar as características extraídas (denominadas de *fully connected*) e uma função de ativação (por exemplo, *softmax*) para a tarefa de classificação (figura 10) (Goodfellow et al., 2016).

A primeira camada, a aplicação de filtros por convolução, procura identificar características nas imagens que ajudem a classificá-las, podendo-se tratar de limites de objetos nas imagens, contornos, entre outros. Deste modo, o primeiro processo passa por aplicar um determinado filtro, podendo-se definir qual o *stride* da operação, isto é, qual o "passo" entre posições que define se vamos ter o mesmo tamanho à saída ou se vamos "saltar" píxeis e daí "compactar" a informação (reduzir o tamanho dos mapas de características gerados). Por fim, de notar que é possível o recurso a *padding* para se ajustar os dados de entrada aos parâmetros da camada de convolução, de modo a garantir que a saída tem o tamanho pretendido.

Na segunda camada, *pooling*, entra o conceito de *downsampling*. Este processo é aplicado para diminuir o número de parâmetros, enquanto se tenta manter a informação relevante. Este processo permite também que

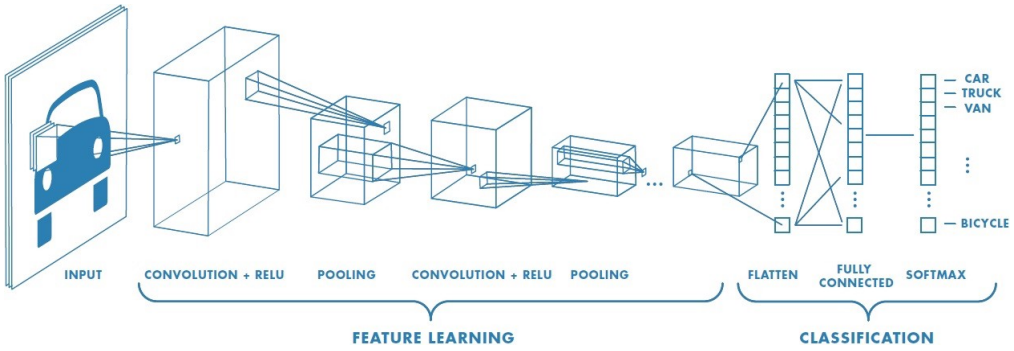


Figura 10: Rede neuronal baseada em CNN. Imagem retirada de Raghav (2018).
Nesta rede neuronal, distinguem-se duas tarefas. Na primeira, o objetivo é extrair as características da distribuição de dados através de camadas convolucionais e *pooling*. Na segunda, depois de extrair as características, procede-se à classificação da imagem através de camadas *fully connected* e uma função de ativação (por exemplo, *Softmax*).

os filtros aplicados em camadas seguintes incidão sobre uma região maior da imagem original, aumentando assim o campo receptivo durante o processo de extração de características.

Não é definido o número de iterações a que se pode sujeitar a entrada ao processo anterior. Quando o processo acaba, os parâmetros são dados como entrada à camada *fully connected* que procede com o processo de classificação.

Com a introdução das GANs, aplicou-se o conceito das CNN no gerador e no discriminador. Porém, em testes iniciais, era mais difícil treinar o gerador e o discriminador com o mesmo nível de capacidade e representação que as CNN apresentavam nas tarefas de aprendizagem supervisionada (Creswell et al., 2018).

Verificando-se esta dificuldade, a LAPGAN (*Laplacian pyramid* GAN) solucionou este problema usando uma estrutura *multi-layer perceptron* (MLP) para o processo de geração (Denton et al., 2015). A imagem é decomposta numa pirâmide de Laplace e numa GAN condicional e convolucional que era treinada para produzir uma camada a seguir à outra (Creswell et al., 2018). Apesar de obter imagens com melhor qualidade, os objetos nas imagens ainda não apareciam bem definidos.

Dado que as CNN são melhores que as MLP para extrair características das imagens, seguindo a LAPGAN, mas procurando imagens mais confiáveis, foi proposta a DCGAN (*Deep Convolutional* GAN) (Radford et al., 2015). Pretendia substituir as redes *fully connected* por redes desconvolucionais, obtendo melhores resultados na geração de imagens (figura 11) (Pan et al., 2019).

Este modelo segue um modelo CNN fazendo as seguintes alterações (Radford et al., 2015):

- Substituir as camadas *pooling* por *strided convolutions* no discriminador e por *fractional-strided convolutions* no gerador.
- Remover quaisquer camadas *fully connected*.
- Usar *batch normalization* (Ioffe and Szegedy, 2015; Goodfellow et al., 2016) após as convoluções, tanto no gerador como no discriminador.
- Usar a ativação Rectified Linear Unit (ReLU) (Nair and Hinton, 2010) no gerador em todas as camadas exceto no *output*, que usa *tanh*.

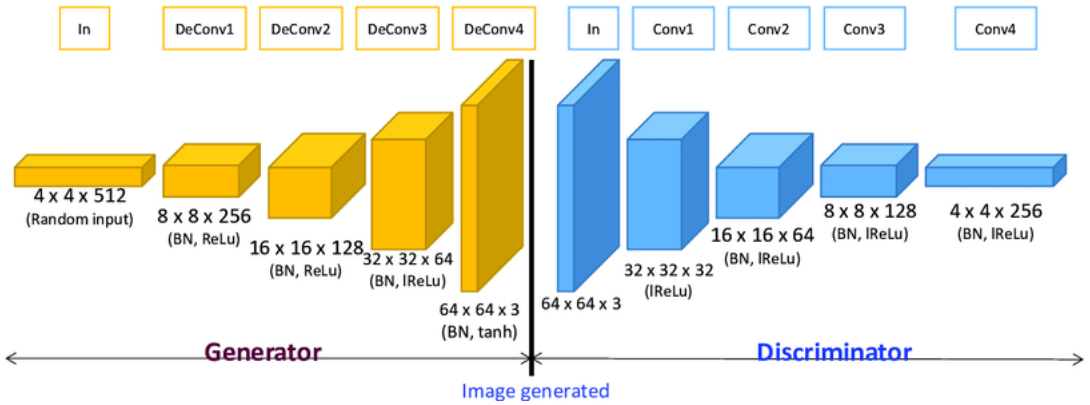


Figura 11: *Deep Convolutional* GAN. Imagem retirada de Suh et al. (2019).
Tanto o gerador como o discriminador seguem o modelo CNN. No caso do gerador, é utilizada a ativação ReLU exceto na camada final, na qual é utilizada a ativação *tanh*. No caso do discriminador, todas as camadas usam a ativação *leaky* ReLU.

- Usar a ativação *Leaky* ReLU no discriminador em todas as camadas.

A ativação ReLU (Nair and Hinton, 2010) permite introduzir não-linearidade ao processo. Existem outras funções de ativação mas a ReLU é a que apresenta melhores resultados (Neapolitan and Neapolitan, 2018).

Batch Normalization é uma camada utilizada em redes neuronais profundas, que normaliza as entradas para cada *mini-batch* (conjunto de dados de entrada para um dado passo do processo de aprendizagem). Esta técnica visa regularizar os valores das ativações (pesos e bias) de uma dada camada, de tal modo que estas se mantenham em valores equilibrados e se evite a "explosão" ou o "desaparecimento" dos gradientes. Este processo permite assim estabilizar e facilitar o processo de aprendizagem. Em simultâneo, esta técnica permite acelerar o treino (por vezes, para cerca de metade).

Como extensão, Wu et al. (2016) apresentou uma GAN capaz de sintetizar imagens 3D usando convoluções volumétricas.

2.4.3 *Conditional GAN*

Em modelos generativos incondicionais, não existe controlo nos dados gerados. A falta de restrições nas entradas pode levar a que o treino entre em *mode collapse*. Este fenómeno ocorre quando o gerador, ao invés de gerar um conjunto variado de imagens, começa a gerar constantemente uma imagem (ou um conjunto delas mas com muito pouca variabilidade). Isto acontece quando, durante o processo de treino, o gerador fica "preso" na tentativa de enganar o discriminador com uma imagem que este considera ser da distribuição real, não havendo mais melhoria no restante treino.

No entanto, condicionando um modelo com informação adicional, é possível orientar o processo de geração de dados. Tal informação pode ser fornecida através de *labels* por dados associados a processos de segmentação, entre outros.

A primeira GAN a introduzir o conceito acima citado foi a CGAN (Mirza and Osindero, 2014). Utilizando como base a primeira arquitetura (Goodfellow et al., 2016), surgiu a primeira GAN que condicionava a camada

de entrada e para isso fornecia, para além do ruído, a *label* a que a imagem pertencia, tanto no discriminador como no gerador (figura 12 a)).

A função de otimização é similar à GAN, apenas introduz a *label* c como entrada:

$$min_G \sim max_D \, V(D,G) = E_{x \sim pdata(x)}[logD(x|c)] + E_{z \sim p(z)}[log(1 - D(G(z|c)))] \quad (2)$$

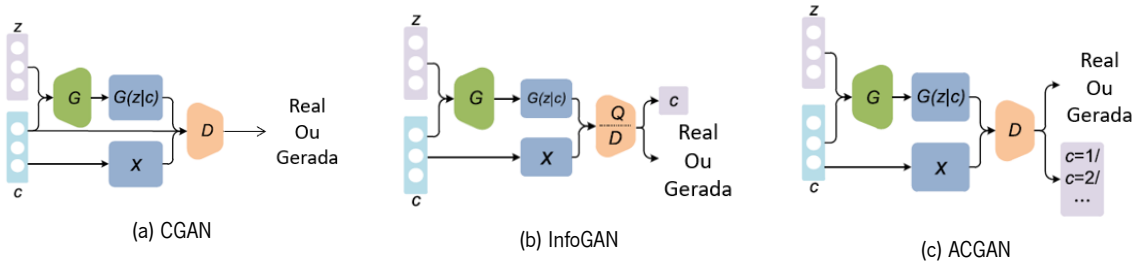
Em adição à CGAN, foi proposta a InfoGAN, introduzindo *mutual information* ao processo, permitindo um processo de geração mais controlado e um resultado interpretável Chen et al. (2016). Esta GAN introduziu uma melhor estruturação na maneira como os dados eram fornecidos às redes. Em vez de fornecer um vector contendo o ruído e a informação relativa à *label*, o vector era dividido em dois e introduzido na camada de entrada de maneira mais estruturada (ver figura 12 b)). A informação partilhada estabelecia uma relação entre o ruído e a *label* associada. Esta regularização pode ser aplicada pela seguinte equação:

$$min_{G,Q} max_D V_{InfoGAN}(D,G,Q) = V(D,G) - \lambda L_I(G,Q), \quad (3)$$

sendo que a componente Q corresponde à camada que precede a regularização das entradas, o ruído e a *label*.

Contudo, as GANs continuaram a ter problemas na geração de imagens de alta resolução com coerência, principalmente quandos aplicadas a datasets de enorme variabilidade. Então, Odena et al. (2017) propôs a ACGAN, na qual não seria fornecida a *label* ao discriminador, mas sim adicionado um classificador que analisava a probabilidade de pertencer a uma certa *label* e este valor era ajustado na função de *loss* da GAN. Estas novas alterações permitiram obter imagens de resolução maior, com mais qualidade (ver figura 12 c)).

O modelo acima citado permitiu treinos com sucesso ignorando a componente de *loss* associado a *labels*. Também se chegou à conclusão que obter amostras de boa qualidade pode ser independente de realizar um treino bem sucedido, podendo acontecer se a rede apenas aprender para algumas *labels*.



(a) CGAN (b) InfoGAN (c) ACGAN

Figura 12: Diferentes arquiteturas de modelos condicionais. Imagem adaptada de Pan et al. (2019).
Na CGAN, podemos observar a *label* associada a uma imagem da distribuição X como entrada no discriminador e como entrada no gerador para a geração de imagens. No caso da InfoGAN, esta informação é pré-processada e realiza-se uma regularização através de Q à função de otimização. Para a ACGAN, sabendo a *label* da imagem introduzida no discriminador, é aplicado uma *loss* associada à *label* da imagem real e aquela que é prevista por um classificador, condicionando assim o treino.

2.4.4 *Adversarial Auto-Encoders*

Auto-encoders são redes compostas por um *encoder* e um *decoder* que aprendem a mapear os dados numa representação mais específica. O *encoder* usa os dados e procura criar uma representação mais estruturada e objetiva desses mesmo dados, num processo de redução da dimensionalidade destes. Inversamente, o *decoder* usa a informação estruturada e procura gerar dados similares à distribuição fornecida pelos dados originais. Este processo permite aprender representações não lineares e tem boa flexibilidade em termos de arquitetura, podendo ser aplicado para diferentes situações. As principais aplicações de *auto-encoders* são o *denoising* e a redução de dimensionalidade, sendo que são redes treinadas para preservar o máximo de informação possível (Academy, 2019).

Porém, apesar de procurar estruturar as informações principais, apresenta falhas de organização da informação nas camadas ocultas, níveis mais profundos das redes, no qual a informação se encontra comprimida ao máximo.

As *adversarial networks* iriam se juntar mais tarde com os *auto-encoders* Makhzani et al. (2015). A ideia fundamental passava por o *encoder* receber um conjunto de imagens associadas a uma distribuição e, a partir da informação retirada, usar como entrada no *decoder* e gerar novas imagens, com as principais características da distribuição. As imagens geradas e as imagens reais são introduzidas no *encoder* e a informação extraída será introduzida num discriminador que tentará determinar se a informação está associada a uma imagem falsa ou verdadeira. Deste modo, o conceito de redes adversárias é associado ao *decoder* (semelhante ao gerador) e ao discriminador. Como o *encoder* permite obter informações mais simplificadas das imagens, usa-se esta informação para analisar a origem da imagem e, também, para complicar mais o processo de classificação do discriminador, permitindo assim equilibrar melhor o treino (figura 13).

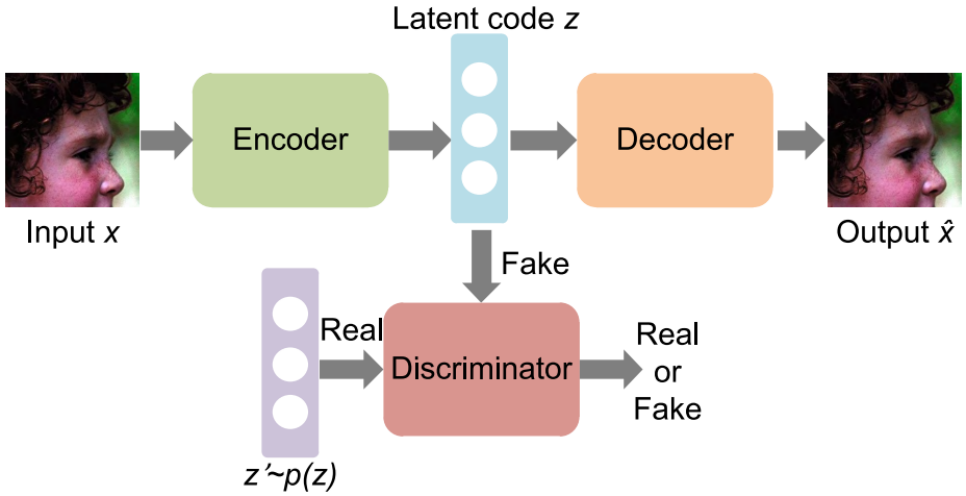


Figura 13: Arquitetura da *Adversarial Auto-Encoder*. Imagem adaptada de Pan et al. (2019).
As imagens das distribuições reais e geradas passam pelo *encoder* no qual serão extraídos vetores com as características mais simplificadas e objetivas. Estes vetores são introduzidos no discriminador, para realizar o processo de distinção entre imagens reais ou geradas e, no *decoder*, são geradas imagens com os dados fornecidos pelo *output* do *encoder*.

Alguns modelos optaram por apenas introduzir o *encoder* à sua arquitetura, realizando pequenas alterações. No entanto, depararam-se com problemas na aprendizagem aquando da passagem da distribuição de amostras para o espaço latente. Donahue et al. (2016) propôs a *Bidirectional Generative Adversarial Network* (BiGAN), para resolver o problema citado (figura 14).

A BiGAN (figura 14), considerando o modelo inicial da GAN, adiciona um *encoder* ao gerador que extrai das amostras geradas informações que serão organizadas em representações mais específicas e estruturadas. Deste modo, realizando o mesmo processo para os dados, a entrada do discriminador são *tuples* compostos pela imagem, gerada ou real, e a sua representação extraída do *encoder* correspondente. O *encoder* deve ser capaz de realizar o processo inverso ao gerador, ou seja, tal como o gerador gera imagens a partir de uma representação aleatória, o *encoder* deve ser capaz de processar a imagem gerada e obter, idealmente, a mesma entrada do gerador. Esta GAN também sugere que um *encoder* bem treinado deverá ter a capacidade de extrair características e atributos específicos a diferentes representações, sendo que as imagens que sejam similares apresentam saídas no *encoder* muito semelhantes e associam *labels* às diferentes representações.

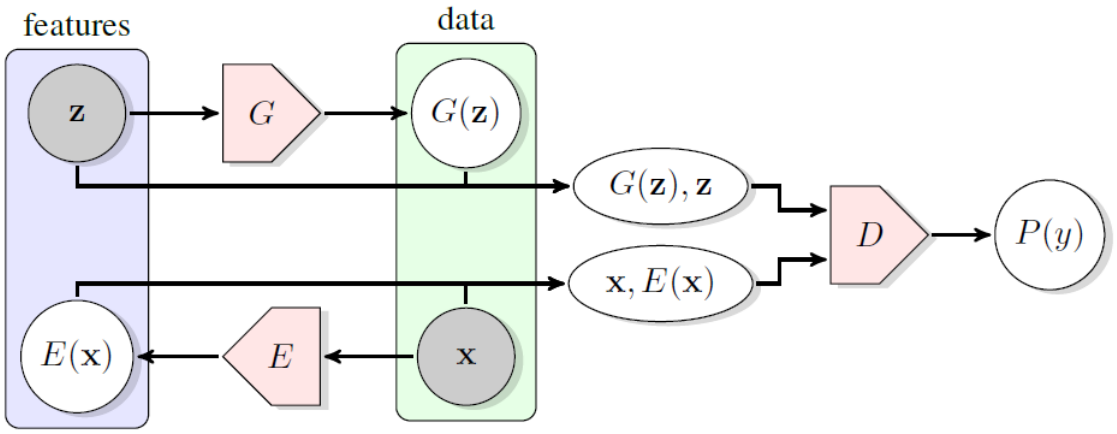


Figura 14: Arquitetura da BiGAN. Imagem retirada de Donahue et al. (2016).
O *encoder* apenas se aplica na distribuição real, extraindo assim as características associadas às imagens. No caso do gerador, com ruído de estrutura igual à saída do *encoder*, são geradas imagens. Ao discriminador são fornecidas as imagens e as correspondentes características iniciais, no caso das geradas o ruído z e no caso das imagens reais o vetor com as características simplificadas e objetivas.

Considerando os conceitos mencionados acima, Ulyanov et al. (2016) propôs a *Adversarial Generator-Encoder Network*, onde as redes adversárias ocorrem entre o gerador e o *encoder*, abdicando do discriminador (figura 15). Neste modelo, o gerador procura minimizar a divergência entre a distribuição latente e a distribuição das imagens geradas, enquanto que o *encoder* procura maximizar a divergência associada ao gerador e minimizar a divergência às imagens reais.

2.4.5 *GANs mais complexas e recentes*

Desde a primeira GAN, foram desenvolvidas diversas GANs, como já se verificou. Todas elas introduziram novos conceitos podendo-se distinguir diferentes focos no desenvolvimento das GANs, como o objetivo de melhorar a estabilidade do treino, através de alterações na função objetivo para melhorar a convergência ou normalizar os gradientes das redes, bem como mudanças na arquitetura (Brock et al., 2019).

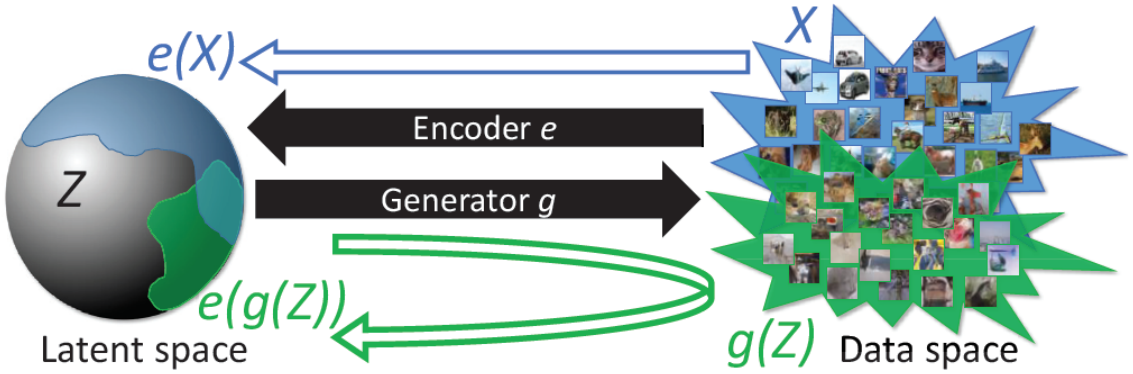


Figura 15: Arquitetura do AGE. Imagem retirada de Ulyanov et al. (2016).
Este modelo contêm apenas duas componentes, o gerador G e o *encoder* e. O processo de atingir duas distribuições similares é atingido com treino adversário, no qual o gerador tenta minimizar a diferença entre Z e e(g(Z)) enquanto que o *encoder* procura minimizar a diferença entre e(X) e Z, e maximizar a diferença entre Z e e(g(Z)). Esta implementação pode incluir uma *loss* de reconstrução para garantir que o modelo atua como *auto-encoder*.

Deste modo, recentemente, começaram a surgir GANs mais complexas, que procuram juntar os diferentes conceitos e não isolar cada técnica à GAN mais simples e ver a diferença de resultados, ou seja, procuram obter o melhor resultado possível independemente da complexidade da GAN.

A GAN que introduziu mais drasticamente diversos conceitos e mudanças de arquitetura foi a BigGAN (Brock et al., 2019). Esta *framework* apresentou os melhores resultados para a síntese de imagens baseadas no *dataset* do ImageNet. Esta arquitetura servirá de base para a presente tese, e será descrita em detalhe no capítulo 3.

Abordando mais concretamente a evolução das GANs, desconsiderando a BigGAN, algumas destacaram-se mais que outras.

Karras et al. (2018) introduziu o conceito de *progressive growth* com a PGAN (figura 16), permitindo aumentar a resolução das imagens geradas. Esta GAN permitiu obter imagens com qualidade para resoluções de 1024 e simplificar e acelarar consideravelmente o processo de treino.

Com a SNGAN, Miyato et al. (2018) introduziu um novo método, *spectral normalization*. Este método marcou a evolução das GANs tratando-se de um método que permitiu melhorar bastante a estabilidade do treino. Fundamentalmente, este método permite a cada atualização da rede, regularizar os pesos e *bias* para que estes não explodam para valores anormais.

A *Boundary-Seeking* GAN, BSGAN, introduziu certas alterações no algoritmo que permitiram tornar as variáveis discretas e contínuas deriváveis, permitindo assim realizar atualizações nas redes não só em termos de funções mas também de modificações nas variáveis.

A *Self-Attention* GAN, SAGAN (Zhang et al. (2019a)), introduziu novas técnicas que permitiram estabelecer ligações entre as duas redes e melhorar assim a comunicação entre elas.

Outras GANs introduziram diferentes métodos e técnicas novas, mas focadas em tradução imagem para imagem, isto é, transformar certas imagens noutras, não existentes na distribuição original. Em termos de técnicas para tradução imagem para imagem, destacam-se então StarGAN (Translation), Pix2PixHD (Wang et al. (2018)), MUNIT (Huang et al.), SPADE (Park et al. (2019)) e FUNIT (Liu et al. (2019)).

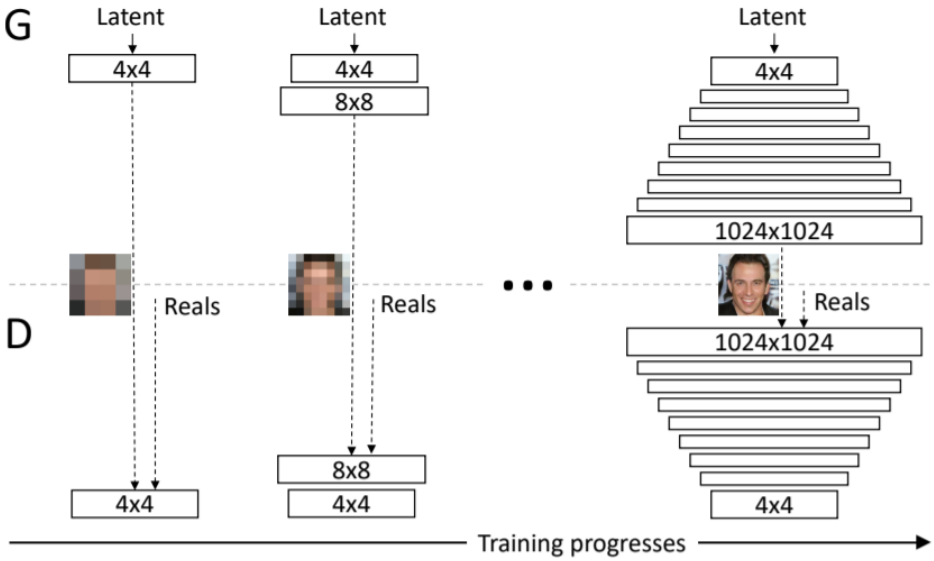


Figura 16: Processo de treino da PGAN. Imagem retirada de Karras et al. (2018).
O treino de ambas as redes começa numa resolução muito baixa (4x4). Com o progresso do treino, são adicionadas camadas ao gerador e ao discriminador para aumentar o tamanho das imagens geradas. Durante este processo, todas as camadas vão sendo treinadas ao longo do treino. Isto permite obter imagens de resoluções altas e acelerar o processo de treino consideravelmente.

A StyleGAN (Karras et al. (2019)) pode-se considerar uma GAN "híbrida". É uma GAN com um processo complexo, na qual inclui processos que têm como *input* ruído e imagens. Ou seja, recorre ao processo tradicional de geração de imagens através de ruído e incorpora componentes de tradução de imagens.

2.4.6 *Técnicas para melhorar o treino*

O objetivo principal do treino das GANs é o Equilíbrio de *Nash*, mas a sua implementação revelou-se um processo muito complicado. No entanto, com a evolução desta área, foram "estabelecidos" alguns conceitos comuns às diferentes GANs relativos a melhores métodos de treino que permitem obter melhores resultados em termos de amostras e desempenho do treino.

Salimans et al. (2016), no seu artigo *"Improved Techniques for GAN Training"*, sugeriu 4 ideias fundamentais que permitem melhorar o desempenho e a estabilidade do processo de treino:

- *Feature Matching* - nova função objetivo dada ao gerador na qual o orienta a gerar imagens com valor estatístico à saída do discriminador quando se trata de imagens reais, ou seja, o gerador tenta prever a saída do discriminador quando a entrada são imagens reais e recriar uma imagem que, quando seja entrada no discriminador, obtenha a mesma saída, permitindo obter amostras mais consistentes e um treino mais estável.

- *Minibatch Discrimination* - quando ocorre *collapse* no gerador, cada saída gerada tem características muito semelhantes de maneira que o discriminador aprende que aquela imagem vem do gerador. Deste modo, o método de gradiente não consegue separar saídas idênticas. Usando *minibatch discrimination*,

o discriminador observa vários exemplos em combinação e não isolados (aumento de diversidade), o que permite minimizar a probabilidade de ocorrência de colapso do gerador.

- *Historical Averaging* - este método regulariza os parâmetros da rede considerando valores médios de todo o treino. Isto torna o processo de treino mais robusto porque para variações bruscas, os parâmetros das redes não variam drasticamente a ponto do treino se destabilizar.

- *One-Sided Label Smoothing* - Alterar a classificação entre 0 e 1 para 0.1 e 0.9, respetivamente, de modo a que a confiança do discriminador não seja alta e balance mais a disputa entre o discriminador e o gerador.

Mais tarde, Heusel et al. (2017) concluiu que usando duas *learning rate* (LR, isto é, a taxa de aprendizagem das redes) diferentes para o discriminador e o gerador permitia à rede atingir um Equilíbrio de *Nash* com mais facilidade, conceito conhecido por Two Time-Scale Update Rule (TTUR). Quando utilizados LR em separado, e sendo considerado um dos pârametros mais importantes do processo de treino, consegue-se definir a "confiança" com que a rede realiza a atualização, sendo que como o discriminador terá mais facilidade em realizar a sua tarefa, distinguir imagens reais de falsas, a rede poderá dar passos mais confiantes (LR mais alto). Já o gerador, como inicialmente só gera imagens com muito ruído, terá que dar passos mais "pequenos" (LR mais baixo), mais calculados, para que continue a convergir para a saída pretendida e não entre em *mode collapse*.

Miyato et al. (2018), com a SNGAN, propôs aplicar *Spectral Normalization* no discriminador, técnica que permite normalizar os pesos da rede e estabilizar o treino da GAN, sendo que também se verificou, mais tarde, que aplicando no gerador obtia melhores resultados ainda (Zhang et al., 2019a).

Quanto maior for a resolução das imagens, mais instável e difícil se torna o treino. E um dos problemas mais associado à instabilidade é os gradientes que são transmitidos do discriminador para o gerador se tornarem desinformativos relativamente às distribuições reais e geradas. Recentemente, foi proposto o conceito de *Multi-Scale Gradients* nas GANs (Karnewar and Wang, 2020), mais propriamente na ProGAN (*Progressive Growing* GAN). Esta técnica permite estabelecer uma ligação entre os gradientes do discriminador e do gerador, sendo que os gradientes fluem de um para outro graças a múltiplas *skip connections* entre as redes (figura 17).

2.5 Métricas

Com a evolução das GANs, também foram sendo introduzidas métricas para se poder realizar comparações e análises aos diferentes comportamentos que as GANs têm durante o treino e aos resultados obtidos. Por exemplo, é necessário verificar se o processo de treino é estável, se as imagens geradas têm a devida qualidade, entre outras.

Deste modo, o processo de avaliação das amostras geradas, observável na figura 18, consiste em passar as amostras geradas por uma rede pré-treinada (*feature extractor*) e chegar a um indicador quantitativo que nos permita comparar a distribuição das imagens geradas com as imagens reais (idealmente, iguais).

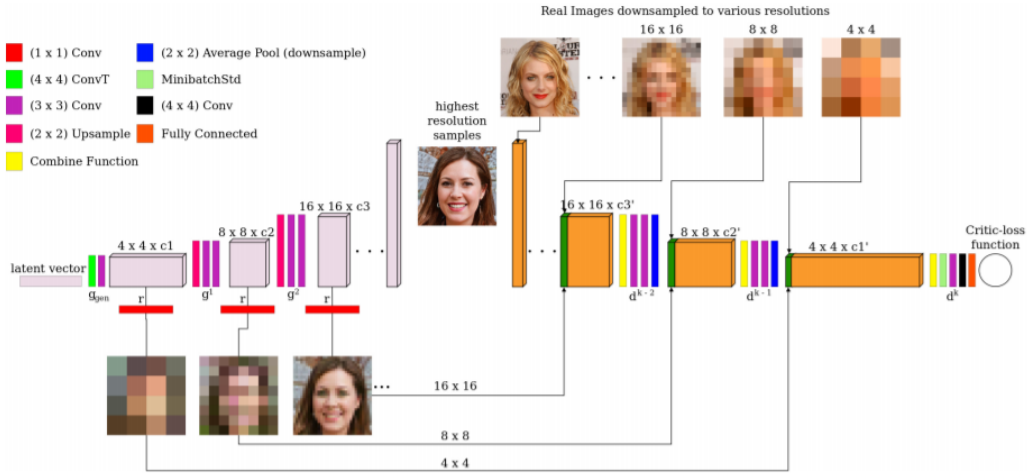


Figura 17: Arquitetura da MSG-GAN, baseada no modelo da ProGAN. Imagem adaptada de Karnewar and Wang (2020). São estabelecidas ligações entre camadas intermédias, a diferentes escalas, permitindo avaliar as imagens a diferentes pontos de detalhe e realizando um treino mais pormenorizado.

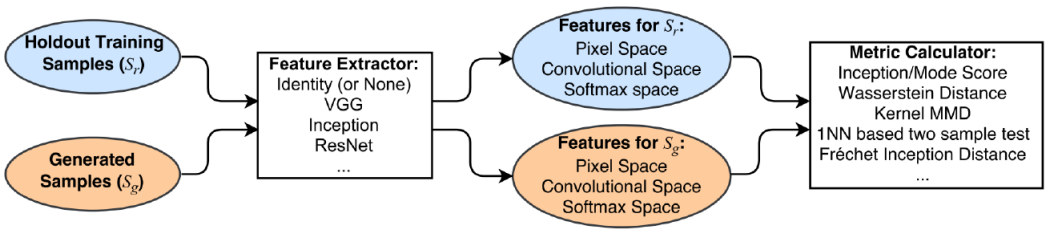


Figura 18: Método mais comum para avaliação de amostras geradas por GANs. Imagem adaptada de Xu et al. (2018). Sr e Sg correspondem a imagens reais e geradas, respetivamente.

Existem diferentes ideias e métodos que abordam as métricas utilizadas para avaliação das GANs. No entanto, há um conjunto de conceitos comuns às diferentes métricas, citados por Borji (2019) num estudo realizado às diferentes métricas, que são a base para uma métrica ideal e eficiente:

- Discriminabilidade das amostras (qualidade em distinguir as imagens reais daquelas geradas).
- Gerar imagens com diversidade (ter sensibilidade ao *overfitting* e *mode collapse*).
- Amostragem controlada, com os espaços latentes bem estruturados.
- Limites bem definidos.
- Robustez contra distorções e transformações das imagens, conseguindo associar estas à mesma *label* mesmo que sofram pequenas alterações.
- Ser consensual com a avaliação e julgamento do utilizador.
- Ter baixo consumo computacional e amostragem.

Nesse mesmo estudo, Borji (2019) reviu e comparou as diferentes métricas e procurou identificar as melhores, tendo concluído que não havia uma métrica que preenchesse todos os requisitos acima citados.

Borji (2019) sugere a seleção da métrica consoante a aplicação dada às GANs. Deste modo, na presente tese foram seleccionadas duas métricas, *Inception Score* (IS) e *Fréchet Inception Distance* (FID) que cumprem com as características que se considerou fundamentais, nomeadamente a fidelidade do *dataset*, a diversidade das imagens geradas e o controlo sobre a amostragem. Estas métricas serão descritas em detalhe no capítulo 3.

3

MÉTODOS

Neste capítulo, detalham-se inicialmente os diferentes métodos utilizados para a recolha de imagens do interior de veículos e criação dos *datasets* utilizados. Numa segunda parte, são detalhadas as diferentes técnicas e algoritmos utilizados com o objetivo de gerar imagens do interior de veículos. Na fase final, são apresentadas as métricas utilizadas para monitorizar os diferentes treinos e avaliar os resultados obtidos.

3.1 Datasets

Como foi mencionado, todos os métodos/técnicas associados a AI necessitam de um conjunto de dados orientados ao que se pretende fazer. Deste modo, foi necessário recolher imagens do interior de veículos.

Na fase inicial, observaram-se problemas associados à quantidade de dados. As imagens recolhidas não eram suficientes para a realização de ensaios com sucesso. Como o processo de recolha de dados e organização revelou-se um processo demorado e bastante árduo, procuraram-se alternativas para suprimir estas dificuldades. Diferentes estudos sugerem a realização de *fine-tuning* ou a utilização de outros *datasets* públicos.

O *fine-tuning* consiste em refinar um modelo pré-treinado num *dataset* através de um novo treino no qual as imagens de uma das *labels* do treino original são substituídas por imagens da nova classe de interesse, a qual no nosso caso seriam as imagens recolhidas no interior de veículos. O processo passa por apenas realizar atualizações aos pesos do gerador na fase de treino, mantendo constante o discriminador, ensinando assim o gerador a gerar novas imagens, agora com as propriedades das imagens do novo *dataset*, com uma *label* associada às imagens pretendidas. Este processo força o gerador a adaptar o processo de geração para as novas imagens, substituindo o conteúdo que estava a gerar anteriormente, associado a outra *label*.

A utilização de outros *datasets* públicos consiste em utilizar *datasets* livres e gerar os próprios *datasets* com as imagens de interior de veículos atribuídas a uma *label*. Deste modo, pode-se controlar a quantidade de dados utilizados e o tipo de dados com que a rede é treinada.

No contexto do problema, a aplicação de *fine-tuning* tem uma certa utilidade em redes CNN, no entanto, quando aplicado em GANs, é muito difícil obter bons resultados. A alteração de uma *label* provoca destabilização no treino, não sendo fácil manter o equilíbrio entre gerador e discriminador, um tem que aprender a gerar outro tipo de imagens e o outro começa a ver imagens completamente diferentes. Posto isto, decidiu-se complementar as imagens do interior de veículos com imagens de outros *datasets* públicos, aumentando assim a quantidade de dados total.

Deste modo, utilizando o *dataset* CIFAR10, substituiu-se uma das *labels* pelas imagens do interior de veículos, formando todas uma *label* só. A utilização deste *dataset* apresentava certas limitações, como a baixa resolução e a quantidade de dados limitada. Para suprimir estas limitações e até potenciar melhores resultados, recorreu-se ainda ao *dataset* ImageNet.

De notar que será possível o uso de imagens não relacionadas com o interior de veículos pelo facto de se ter optado por uma GAN condicional, a BigGAN. Este tipo de GAN permite controlar as imagens geradas (por recurso à *label* dada como entrada), mesmo que o *dataset* contemple uma panóplia de *labels*. Caso se tivesse optado por uma GAN não condicional, esta seria treinada para gerar imagens de qualquer uma das *labels* (pois esta informação não é partilhada com a rede), não havendo qualquer controlo sobre a saída.

Em suma, o uso de uma GAN condicional e o recurso a *datasets* externos para aumento da quantidade de dados disponível permite suprimir as dificuldades associadas à falta de dados no contexto em causa.

3.1.1 *Dataset In-Vehicle*

Para o *dataset in-vehicle*, procurou-se obter o máximo de diversidade de veículos e em diferentes estados de conservação. Deste modo, foram retiradas imagens de sucatas e stands de automóveis.

No processo de obtenção de imagens, foram utilizados dois sensores, apresentando resoluções de 3264x2448 e 1920x1080. Os sensores apresentam campos de visão *ultrawide* (maiores que 110º). Os pontos de captura definidos para todos os carros são apresentados na figura 19.

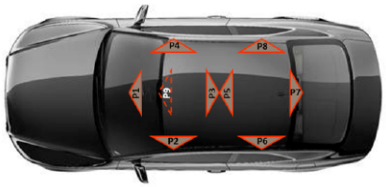


Figura 19: Diferentes pontos de captura do interior de veículos.
As vistas P1 a P8 são todas de perspetiva descendente. A vista P9 é de uma perspectiva ascendente. Foram retiradas 3 fotos de diferentes ângulos para as vistas P1 e P5.

No total, foram adquiridas 13 imagens por veículo. No entanto, em certos carros de sucatas, tornou-se impossível tirar certas fotos devido aos danos que os carros apresentavam. Deste modo, o *dataset in-vehicle* é composto por 135 carros num total de 3477 imagens.

Para se garantir a geração, gestão e exploração do *dataset*, definiu-se o seguinte *pipeline* apresentado na figura 21.

Numa fase inicial, à medida que se iam recolhendo imagens, estas eram estruturadas por automóvel e posteriormente por vista para cada automóvel. Neste processo, todas as imagens eram revistas para verificar a não existência de erros. Depois, esta informação foi estruturada num ficheiro json, passando-se de seguida ao processo de *labelling* manual. Criou-se um método (*Fusion2Json.m*) que permite unir dois ficheiros jsons diferentes sem a perda de informação e um método (*CheckLabel.m*) que permite identificar quaisquer erros no *labelling*. O método *UpdateJson.m* permite atualizar o ficheiro json mais atual adicionando quaisquer imagens que tenham sido adicionadas à versão V1 sem perder o trabalho realizado até ao momento.

Figura 20: Exemplo de imagens retiradas das diferentes vistas.

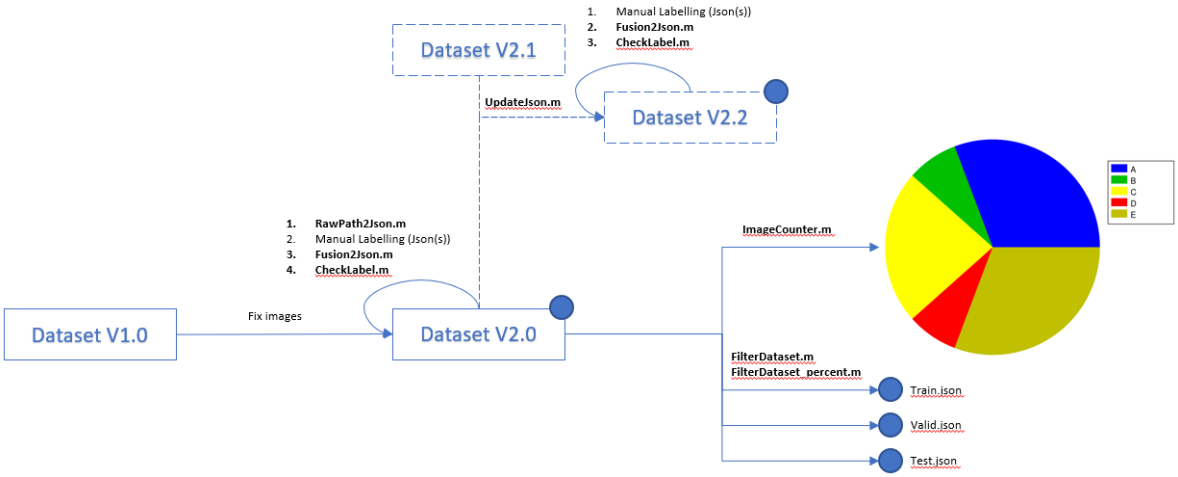


Figura 21: *Pipeline* de gestão e organização do *dataset*.
São apresentados os diferentes *scripts* e quais aplicar consoante o estado do *dataset*. A versão V1 apenas contém as diferentes imagens divididas por vista para cada carro. A versão V2 trata dos diferentes conjuntos de dados, desde a sua filtragem, tratamento e preparação para ser utilizado.

3.1.2 *Dataset CIFAR10 + In-Vehicle*

Inicialmente, recorreu-se ao *dataset* CIFAR10. Este é um dos *datasets* mais comuns e mais usados para ML. É normalmente utilizado para rapidamente validar algoritmos e perceber se estão a funcionar corretamente ou não. Este *dataset* contém as seguintes características:

- Resolução 32x32.
- 10 *labels*.
- *Dataset* com 60 000 imagens, 50 000 para treino e 10 000 para teste.
- Imagens centradas no objeto.

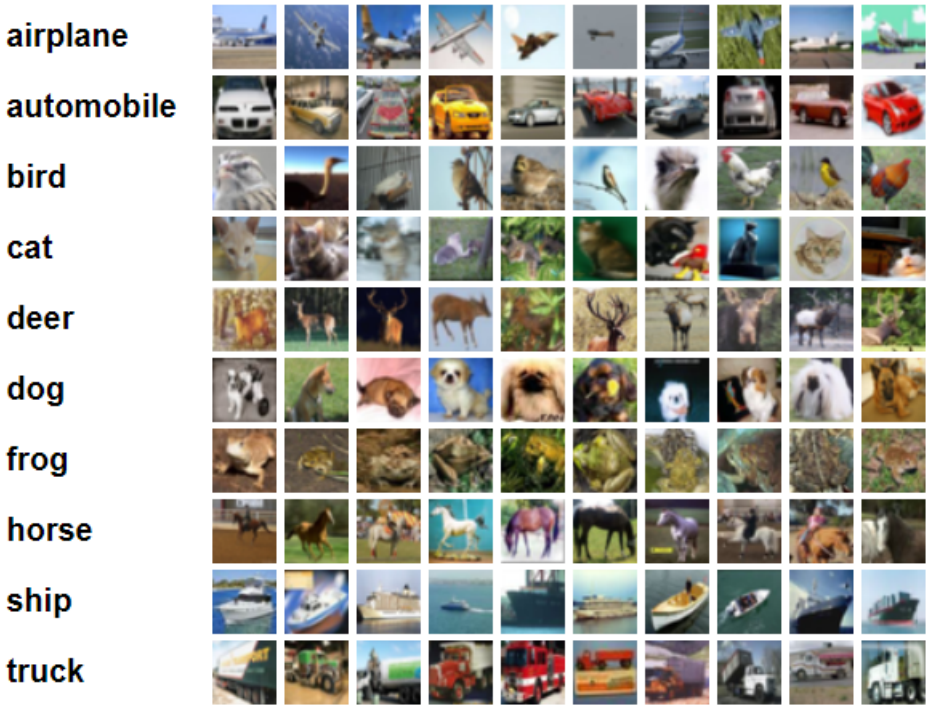


Figura 22: Descrição das 10 *labels* e 10 exemplos de imagens para cada *label* do *dataset* CIFAR10. Imagem adaptada de Alex Krizhevsky and Hinton.

A adaptação realizada passou por escolher uma *label* (*Frog*) e remover todas as imagens associadas a essa *label*. Depois, criou-se uma nova *label* (int) com todas as imagens que compunham o *dataset In-Vehicle*. Estas foram modificadas de modo a coincidir com as restantes imagens que faziam parte do *dataset*, aplicando o método *resize* da biblioteca *OpenCV*. Por comparação dos diferentes tipos de interpolação disponíveis, optou-se pelo uso da opção INTER_AREA uma vez ser aquela que originava o mínimo de artefatos/alterações nas imagens. Assim, foi possível redimensionar todas as imagens do interior de veículos para uma resolução de 32x32.

Deste modo, criou-se um *dataset* de treino com 48 477 imagens e 10 *labels*, correspondendo à segunda linha da tabela 2, com o nome C10_AllViews. De notar que apenas existem 3 477 imagens do interior de veículos, daí a diferença para o CIFAR10 em termos de quantidade de imagens.

3.1.3 *Dataset ImageNet + In-Vehicle*

Como já foi mencionado, para se superar algumas limitações e potenciar mais os resultados, criou-se um *dataset* baseado no *ImageNet*. Procurou-se elaborar um *dataset* mais equilibrado, com mais imagens e um maior número de *labels*.

O *dataset ImageNet* foi criado por uma equipa de investigação com o objetivo de promover a pesquisa associada à imagem e visão por computador. Um dos grandes problemas, como já se mencionou, é a falta de dados. Para investigação, são necessários bastantes dados e com qualidade. Uma equipa de investigação decidiu então criar um *dataset* com dados livres e existentes na internet, organizados por *synsets* específicos, isto é, palavras que descrevem o contexto das imagens que o compõe.

O *dataset ImageNet* contém 32326 *synsets*. Tentaram juntar 1000 imagens por *synset*, de modo a obter um *dataset* equilibrado. A recolha destas imagens começou em 2009 e desde então, têm sido recolhidas e tratadas de modo a aumentar o *dataset* e a sua qualidade.

Neste trabalho, recorreu-se a uma *framework* que permite obter as imagens dos diferentes *synsets*. Todas as imagens que constituem o *dataset* têm associado um URL. As imagens são organizadas por *synsets* e manipuladas consoante a intenção dos utilizadores.

Posto isto, selecionaram-se *synsets* considerados benéficos para o trabalho realizado. Procuraram-se *synsets* com características associadas às *labels* do CIFAR10, como pássaros, carros, cães, entre outros. O objetivo passou por selecionar os *synsets* com maior quantidade de imagens e que acrescentassem diversidade ao *dataset*.

Visto que um maior número de imagens promove a melhor qualidade de treino, pretendeu-se aumentar para o dobro de imagens, aproximadamente. Reunidos os *synsets*, foram reunidas 105167 imagens em 134 *labels*, surgindo o *dataset* ImageNet_Balance, linha 5 da tabela 2. Este *dataset* já continha as imagens do interior de veículos divididas em 3 *labels*: uma corresponde às imagens das vistas laterais do veículo, outra às imagens das vistas frontais e traseiras e a última contendo apenas imagens do teto do veículo.

Realizando o mesmo processo, criou-se outro *dataset* com aproximadamente o dobro de imagens e *labels* do anterior. Realizou-se este processo para se perceber até que ponto a maior quantidade de dados podia ser benéfica para o treino. Deste modo, o novo *dataset* contém 197508 imagens com 276 *labels*, correspondendo à linha 6 da tabela 2 (ImageNet_Balance_Double).

A diferença de quantidade de imagens e *labels* está associado ao facto de diferentes *labels* do *dataset* ImageNet conterem diferentes quantidades de imagens. Foram escolhidas *labels* nas quais a quantidade de imagens rondava aproximadamente 800 imagens.

3.1.4 *Datasets utilizados nos diferentes testes*

A utilização das diferentes combinações e *datasets* mencionados acima para os ensaios resultou no conjunto de *datasets* descrito na tabela 2. Os nomes atribuídos aos *datasets* são meramente simbólicos e procuram fornecer certa informação para fácil identificação por parte do leitor.

O *dataset* CIFAR10 está detalhado acima. Para a utilização deste como base para os ensaios realizados para a resolução 32x32, foi necessário introduzir as imagens do interior de veículos. Para isto, removeu-se a *label frog* e colocou-se todas as imagens do interior de veículos numa única classe, surgindo o *dataset* C10_AllViews. Para estudo da influência da variedade de imagens dentro da própria classe, criou-se um *dataset* C10_P1P5Views, no qual apenas se substitui a classe *frog* pelas imagens das vistas P1 e P5.

O *dataset* ImageNet_128_Double é composto por imagens do *dataset* ImageNet, tendo por base estrutural o C10_AllViews. No entanto, aumentou-se o número de imagens que compõe cada *label* para aproximadamente o dobro, mantendo todas as imagens do interior de veículos na mesma *label*. Como o ImageNet é composto por *synsets* mais específicos, na criação deste *dataset* selecionou-se diferentes *synsets* e agruparam-se todas as imagens segundo a respetiva categoria-mãe, associando-se esta a uma *label*. Por exemplo, agruparam-se as imagens dos *synsets* de diferentes raças de cães na mesma classe, cão.

O ImageNet_Balance é um *dataset* mais equilibrado, no qual todas as *labels* têm aproximadamente o mesmo número de imagens. Na classe de interior de veículos foram incluídas todas as vistas. O *dataset* ImageNet_Balance_Double segue a mesma ideologia da criação do *dataset* ImageNet_Balance, no entanto é composto por aproximadamente o dobro de classes e, consequentemente, imagens.

Tabela 2: Descrição e composição dos diferentes *datasets* utilizados nos diferentes ensaios

Dataset	# Labels	# Imagens	Observações
CIFAR10	10	50 000	*Dataset* oficial CIFAR10
C10_AllViews	10	48 477	CIFAR10 sem a *label frog*, trocando pelas imagens do interior dos veículos
C10_P1P5Views	10	45 801	CIFAR10 sem a *label frog*, trocando pelas imagens do interior dos veículos das vistas P1 e P5
ImageNet_128_Double	10	97 792	Criação de dataset com resolução 128x128 similar ao C10_AllViews com o dobro de imagens por *label*
ImageNet_Balance	134	105 167	*Dataset* mais equilibrado com um número aproximado de imagens por *label*, incluindo o interior de veículos
ImageNet_Balance_Double	276	197 508	*Dataset* com maior número de imagens e *labels* para estudar o comportamento do treino com o dobro de quantidade da dados

3.2 BigGAN

Depois de se explorar um grande conjunto de GANs e analisar as diferentes funcionalidades que cada uma tinha para oferecer, optou-se pela utilização de uma *framework*, em *PyTorch*, baseada na BigGAN (Brock et al., 2019).

A BigGAN apresentou 3 grandes contribuições que serviram de base para novos avanços nesta área:

- Realizaram alterações simples na arquitetura que permitiram melhorar a escala da rede e comprovaram que as GANs beneficiam destas alterações;
- Conseguiram melhorar o treino quando se utilizam redes com muitos parâmetros, melhorando assim a estabilidade do processo de geração de imagens;
- Modificaram a regularização melhorando condicionalmente o treino, obtendo melhor desempenho.

A BigGAN baseou-se na arquitetura das ResNET (figura 23) e SAGAN. No entanto, inclui diferentes métodos e técnicas que foram sendo introduzidas e convencionadas por diferentes equipas de investigação, até ao surgimento da BigGAN.

As ResNET são redes com diferentes camadas, organizadas em *residual blocks*. Normalmente, numa rede neuronal, a informação passa de camada em camada. Numa rede com *residual blocks*, a informação passa para a próxima camada e 2 ou 3 camadas depois, como se observa na figura 24. Quando as conexões entre camadas eram lineares, e quanto maior fosse a rede e mais complexa, o desempenho do modelo era cada vez mais degradado, havendo muitas perdas de informação, muitas transformações e a precisão da rede saturava.

$z \in \mathbb{R}^{120} \sim \mathcal{N}(0, I)$ Embed$(y) \in \mathbb{R}^{128}$
Linear $(20 + 128) \rightarrow 4 \times 4 \times 16ch$
ResBlock up $16ch \rightarrow 16ch$
ResBlock up $16ch \rightarrow 8ch$
ResBlock up $8ch \rightarrow 4ch$
ResBlock up $4ch \rightarrow 2ch$
Non-Local Block (64×64)
ResBlock up $2ch \rightarrow ch$
BN, ReLU, 3×3 Conv $ch \rightarrow 3$
Tanh

(a) Generator

RGB image $x \in \mathbb{R}^{128 \times 128 \times 3}$
ResBlock down $ch \rightarrow 2ch$
Non-Local Block (64×64)
ResBlock down $2ch \rightarrow 4ch$
ResBlock down $4ch \rightarrow 8ch$
ResBlock down $8ch \rightarrow 16ch$
ResBlock down $16ch \rightarrow 16ch$
ResBlock $16ch \rightarrow 16ch$
ReLU, Global sum pooling
Embed$(y) \cdot h$ + (linear $\rightarrow 1$)

(b) Discriminator

Figura 23: Arquitetura da BigGAN. Imagem adaptada de Brock et al. (2019).
ch corresponde ao pârametro de multiplicador do canal. É possível observar os diferentes *residual blocks* que compõe as redes ResNET.

A utilização dos *residual blocks* permitiu melhor convergência e desempenho, permitindo preservar melhor a informação relevante entre camadas.

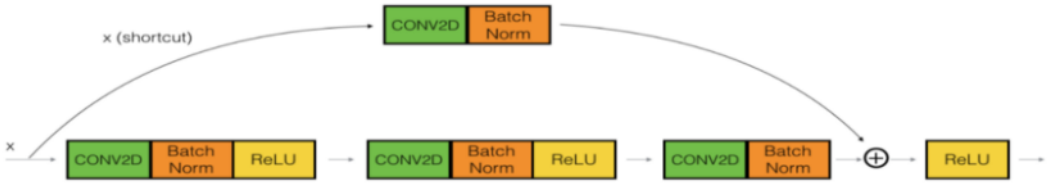


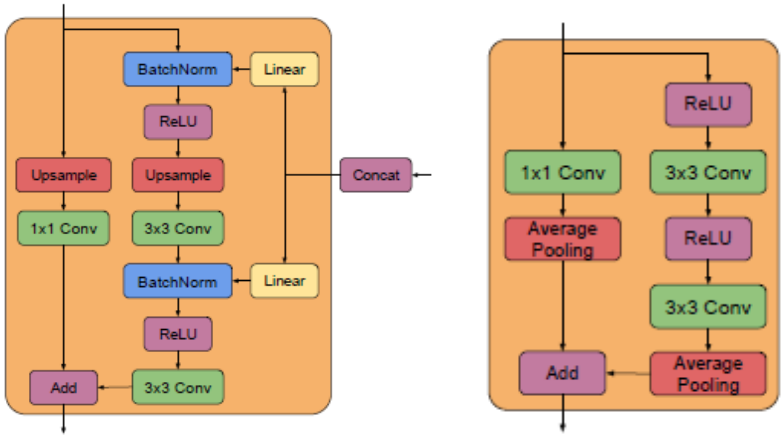
Figura 24: Exemplo do processo associado aos *residual blocks*. Imagem retirada de Nandepu (2019).
A mesma entrada passa por camadas diferentes, sendo a saída de ambos os processos adicionadas no fim do *residual block*.

Na figura 25, observa-se um processo similar ao da figura 24. Analisando a figura 25a, observa-se que a entrada na camada de *BatchNorm* é comum à camada de *upsample*, sendo que ambas as saídas se juntam numa fase final do bloco.

A BigGAN realizou algumas modificações na introdução de entradas na rede do gerador. Como se observa na figura 26, são fornecidos duas entradas, ruído z e a *label*, mas de maneira conjunta.

A BigGAN utiliza espaços latentes hierárquicos que consiste em introduzir o vetor de ruído em múltiplas camadas ao longo do gerador, em vez de ser só na inicial. Deste modo, o ruído é dividido em conjuntos mais pequenos de tamanho igual (neste caso, 6 conjuntos, a entrada inicial mais os 5 blocos residuais). A este vetor de ruído junta-se a informação da *label*, que através de um processo de *shared embedding*, converte-se em vetor. Este vetor (ruído e *label*) entra na rede através das camadas que aplicam *Batch Normalization* (BatchNorm), sendo projetado nos seus parâmetros, pesos e *biases* (observar figura 26).

O processo de *shared embedding* permite mapear variáveis discretas (cada *label* tem um valor inteiro atribuído) num vetor de números contínuos, e a camada correspondente é partilhada por todas as camadas. Isto é, para cada *label* estão associados parâmetros constantes que são partilhados por toda a rede consoante a entrada associada à *label*, o que permite reduzir significamente a dimensão das variáveis e os parâmetros da rede.



(a) (b)

Figura 25: *Residual Blocks* no Gerador e Discriminador na arquitetura da BigGAN. Imagem adaptada de Brock et al. (2019).
a) corresponde ao *Residual Block* no gerador; b) corresponde ao *Residual Block* no discriminador. Ambos os *residual blocks* são constituídos por diferentes camadas. Estas aplicam as diferentes técnicas mencionadas na BigGAN, que permitem normalizar e regularizar os dados processados nas redes.

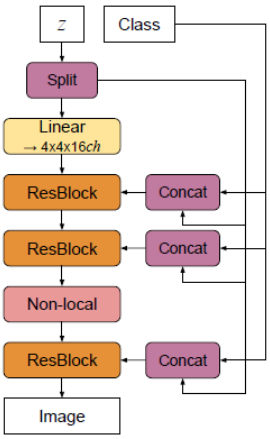


Figura 26: Exemplo do gerador aplicado na BigGAN. Imagem adaptada de Brock et al. (2019).
No bloco *split*, realiza-se a separação do ruído em diferentes conjuntos que depois serão introduzidos ao longo da rede (espaços latentes hierárquicos). No bloco *concat*, depois da informação associada à *label* ser processada (*shared embedding*), esta é concatenada com o ruído de entrada e projetada na camada de BatchNorm (figura 25a).

Na SAGAN, Zhang et al. (2019a) introduziu o conceito de *self-attention*, o que permitiu ao gerador e discriminador criar relações entre diferentes camadas dentro da rede (figura 27), associadas a diferentes características, muito distintas umas das outras (figura 28). Na figura 27, observa-se o processo relacional associado a *self-attention*. A cada camada da rede, segundo a sua função, é estabelecida uma relação com as outras camadas através de diferentes processos que permitem preservar a informação e melhorar o processo da rede.

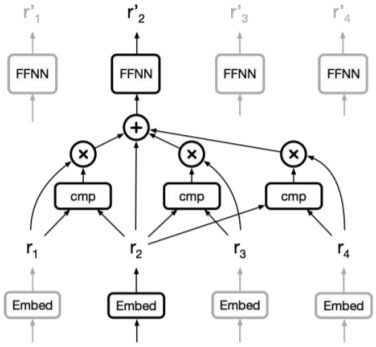


Figura 27: Exemplo de relações estabelecidas entre diferentes camadas para o *update* dos gradientes de uma camada. Imagem retirada de Vaswani and Huang (2019).
A cada camada realiza-se uma relação com as restantes sempre com o mesmo processo matemático, normalmente através de multiplicações e *softmax*. Depois, o resultado das relações existentes com as diferentes camadas é usado para calcular os gradientes nessa camada. Isto permite estabeler relações entre as diferentes camadas, estando normalmente associada a cada camada uma característica da distribuição correspondente.

Observando o exemplo na figura 28, associa-se as diferentes regiões (conjunto de camadas) a diferentes características correspondentes à distribuição de cada *label*. Assim, torna-se mais fácil para a rede identificar as características que correspondem a certa *label* e consegue gerar imagens com melhor qualidade.

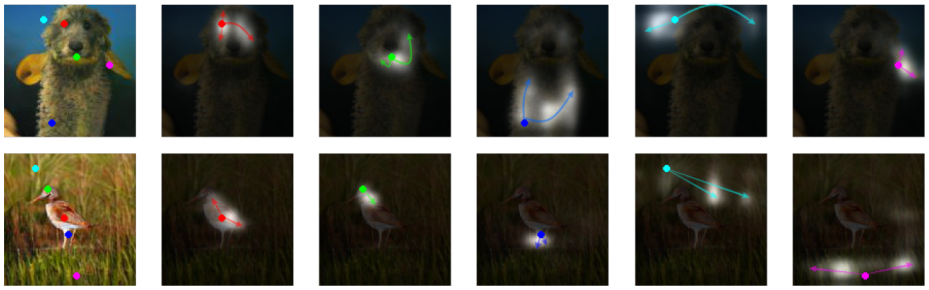


Figura 28: Exemplo das diferentes características associadas a cada imagem. Imagem adaptada de Zhang et al. (2019a). Em cada coluna, observam-se diferentes mapeamentos das características associadas às diferentes regiões dentro das redes, as quais ficam identificadas para estabelecer as tais relações mencionadas acima.

Seguindo o método da SAGAN, depois de verificar que melhorava consideravelmente o treino, a BigGAN também aplica *Spectral Normalization* e TTUR. No entanto, no processo de otimização, diminui as LR para metade e optaram por realizar 2 passos no discriminador a cada passo do gerador.

A BigGAN aplicou ainda regularização *Orthogonal*. Isto permite, ao longo das camadas, com as diferentes multiplicações e operações realizadas, manter os gradientes estáveis, sem explodir para valores muito elevados e desagradáveis para a rede. Pelo facto de conter certas limitações (Miyato et al. (2018)), procuraram diferentes variantes e chegaram à seguinte fórmula:

$$R_B(W) = B||W^T W \odot (1 - I)||_F^2 \tag{4}$$

Brock et al. (2019) recorreu a um processo iterativo com o objetivo de atingir o melhor resultado possível, realizando diferentes testes com pequenas alterações e observando os resultados. Neste processo, foram aplicados os diferentes conceitos e técnicas acima mencionados, tendo obtido os melhores resultado para um

batch size (BS) de 2048 e canal multiplicador de unidades para cada camada de 96, com *shared embedding*, *skip connections* e regularização *orthogonal*, com um FID de 8,51 e IS de 99,31 no *dataset ImageNet*.

3.3 *Differentiable Augmentation*

Uma vez que a quantidade de dados influencia o treino das GANs e é tipicamente o fator limitante para a obtenção de melhores resultados, uma estratégia muito utilizada é o recurso a técnicas de *augmentation* dos dados. Com vista a maximizar o potencial destas técnicas no contexto das GANs, Zhao et al. (2020) propôs a técnica de *Differentiable Augmentation* (DiffAug).

Observando a figura 29, quando existe um conjunto limitado de dados, o discriminador tende a memorizar o *dataset* e o desempenho do treino piora drasticamente, entrando facilmente em *mode collapse*.

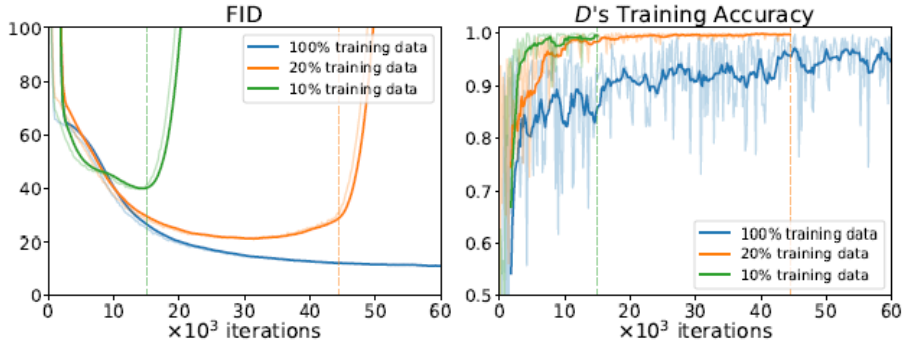


Figura 29: Análise ao treino da BigGAN em CIFAR10 com quantidades limitadas de dados. Imagem adaptada de Zhao et al. (2020).
Quando os dados são limitados para 10%, o treino entra em *mode collapse* em fases precoce de treino e o discriminador memoriza o *dataset*, ganhando confiança e sobrepõe-se ao gerador. Quando reduzidos a 20%, o treino melhora, prolonga-se durante mais tempo mas acaba por acontecer o mesmo que com 10%, entra em *mode collapse*. Quando se utilizam todos os dados do *dataset*, o treino fica estável e é muito mais difícil para o discriminador distinguir imagens reais de falsas.

Certos grupos procuraram desenvolver esta temática aplicando *augmentation* diretamente ao *dataset*, atingindo pequenas melhorias apesar de modificarem a distribuição real. No entanto, o problema de aplicar apenas no *dataset* está no gerador gerar imagens com esse tipo de *augmentation*. Observando a figura 30, as imagens geradas contêm propriedades associadas à *augmentation* e não ao *dataset*, realizando a transformação à distribuição real provocando perda de qualidade nas imagens geradas.

DiffAug permitiu melhorar consideravelmente os resultados obtidos, regularizando tanto o gerador como o discriminador sem modificar a distribuição real, mantendo o treino fluído. Para tal, além de aplicar *augmentation* no *dataset*, o método aplica também às imagens geradas (ver figura 31).

A figura 31 pode ser traduzida matematicamente nas equações de *loss* da GAN pelas seguintes modificações:

$$L_D = E_{x\sim pdata(x)}[f_D(-D(T(x)))] + E_{z\sim p(z)}[f_D(D(T(G(z))))] \tag{5}$$

$$L_G = E_{z\sim p(z)}[f_G(-D(T(G(z))))] \tag{6}$$

Figura 30: Diferentes tipos de *augmentation* aplicados ao *dataset* CIFAR10. Imagem adaptada de Zhao et al. (2020). Podem-se observar nas diferentes linhas, transformações provocadas por *augmentation*. Na primeira linha, observam-se translações da imagem. Na segunda linha, observam-se translações e cortes neutros às imagens. Na terceira linha, observam-se translações, cortes neutros e mudanças em termos de contraste, brilho e saturação nas imagens.

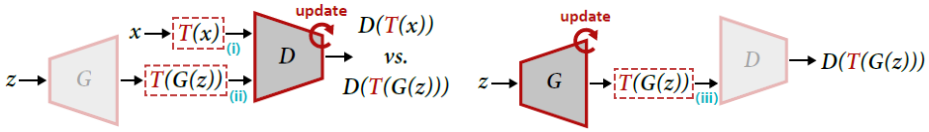


Figura 31: *DiffAug* para atualizar o discriminador e o gerador. Imagem adaptada de Zhao et al. (2020). A *augmentation* é aplicada às imagens reais e geradas, atualizando o discriminador. Quando o gerador é atualizado, os gradientes são propagados através de T, daí a necessidade da operação de transformação da imagem ser diferenciável.

No trabalho original, foram propostos e utilizados 3 tipos de *augmentation* como é descrito na figura 30. Estes 3 tipos têm impactos diferentes nas imagens e podem ser benéficas ou não para o processo de treino. Na figura 32, é possível observar que a quantidade de *augmentation* aplicado não tem grande impacto em termos de FID, no entanto, o discriminador tem mais dificuldade em distinguir as duas distribuições quanto mais alterações forem aplicadas, o que já se torna benéfico para o treino.

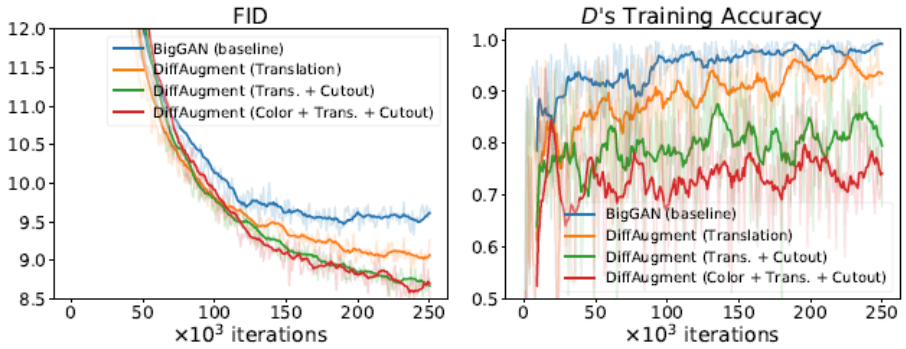


Figura 32: Impacto quantitativo da aplicação de *augmentation*. Imagem adaptada de Zhao et al. (2020). Analisando o FID, observa-se que aplicar alterações à cor não tem grande impacto em termos de qualidade de imagens, no entanto, em termos de treino da GAN, é possível observar que o discriminador tem mais dificuldade em distinguir as duas distribuições de imagens.

No entanto, quando se realizam treinos com uma quantidade de dados limitada, os modelos treinados beneficiam com a utilização de *DiffAug*. Na figura 33, é possível observar que, utilizando o modelo StyleGAN2, o resultado do treino torna-se pior cada vez que se reduzem os dados disponíveis. No entanto, com a aplicação de *DiffAugmentation*, apesar da quantidade de dados ser reduzida consideravelmente, obtiveram-se resultados bons e muito próximos entre as diferentes quantidades de dados, ao contrário do que acontece quando não é aplicada qualquer tipo de *augmentation*.

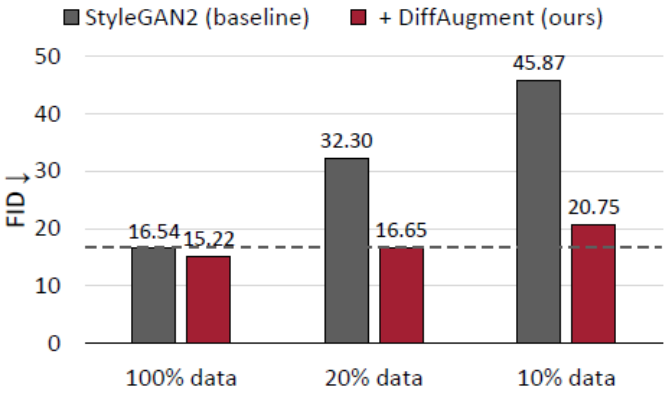


Figura 33: Impacto da *DiffAug* em quantidades limitadas de dados. Imagem adaptada de Zhao et al. (2020). Observa-se que um modelo, quando presente a quantidades limitadas de dados, beneficia da utilização de *DiffAug*. Quando o modelo é treinado sem esta *augmentation*, obtém resultados mais fracos devido ao número reduzido de dados. Aplicando esta técnica, o treino é capaz de atingir valores bastantes melhores (menor FID) apesar da menor quantidade de dados utilizados.

3.4 *Consistency Regularization*

Um dos grandes problemas, como já identificado, é estabilizar o treino. Ao longo da evolução das GANs, diferentes tipos de regularização foram surgindo, mas sempre com problemas de interação com outras técnicas.

Consistency Regularization (CR) é um técnica simples que surgiu para *semi-supervised learning*. Zhang et al. (2019b) introduziu esta técnica que aplica *augmentation* nas entradas do discriminador com vista a

penalizá-lo e tornar o treino mais complicado para este, de modo a não superar o gerador. No entanto, os tipos de *augmentation* têm que ser focados em manter as características das imagens que sofrem *augmentation*, de tal modo que preserve a classificação das imagens originais. Aplicando CR, as imagens antes e depois da *augmentation* tentam manter a mesma *label* e regularizar o discriminador para que tal seja verdade. Isto só acontece se a *augmentation* preservar as características das imagens.

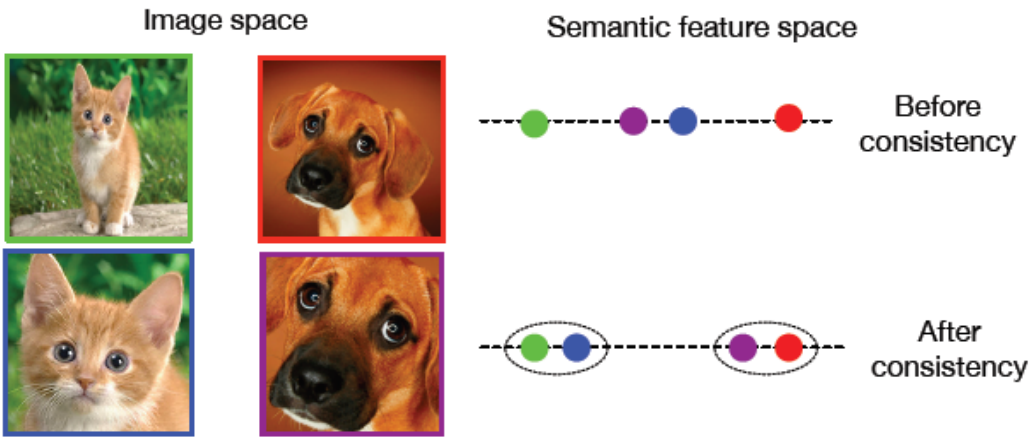


Figura 34: *Consistency Regularization* para GANs. Imagem adaptada de Zhang et al. (2019b).
No espaço da imagem, considerámos 2 conjuntos, gatos e cães, sendo as de cima, as imagens antes de sofrer *augmentation* e as de baixo, depois de sofrer *augmentation*. Sem aplicar CR, no espaço semântico, as imagens a azul e roxo (depois de aplicado *augmentation*) aparecem muito próximas apesar de serem duas imagens de *labels* completamente diferentes. Quando aplicado CR, consegue-se que a distância das imagens no espaço semântico, com ou sem *augmentation*, seja menor, permitindo ao discriminador associar as imagens à mesma *label*, independentemente do tipo de transformação aplicada.

O método de regularização traduz-se na seguinte equação:

$$min_D L_{CR} = min_D \sum_{j=m}^{n} \lambda_j ||D_j(x) - D_j(T(x))||^2 \tag{7}$$

onde j corresponde às camadas, m corresponde à camada inicial e n à camada final, onde CR é aplicado. λ corresponde ao peso da CR. Isto encoraja o discriminador a produzir a mesma saída para o mesmo conjunto de imagens que sofrem diferentes *augmentation*.

Comparativamente com outros modelos implementados até à data, a CR-BigGAN apresentou os melhores resultados relativos a FID. Quando aplicado nos *datasets* CIFAR10 e ImageNet, este modelo atingiu um FID de

11,48 e 6,66, respetivamente. Sem esta regularização, para CIFAR10 o FID era de 14,73 e para o ImageNet era de 7,75, como indica Zhang et al. (2019b).

Algoritmo 1: CR-GAN

Entrada: Parâmetros do gerador e do discriminador θ_G e θ_D, coeficiente CR λ, hyper-parâmetros do otimizador Adam α, β_1, β_2, *batch size* M, número de iterações do discriminador por iteração de gerador N_D;

para *número de iterações de treino* **faça**

  para *t = 1,...,N_D* **faça**

    para *i = 1,...,M* **faça**

      Sample $z \sim p(z), x \sim p_{data}(x)$

      Augment x para obter T(x)

      $L_{CR}^{(i)} \leftarrow ||D(x) - D(T(x))||^2$

      $L_D^{(i)} \leftarrow D(G(z)) - D(x)$

    $\theta_D \leftarrow Adam(\frac{1}{M}\sum_{i=1}^{M}(L_D^{(i)} + \lambda L_{CR}^{(i)}), \alpha, \beta_1, \beta_2)$

  Amostra de um conjunto de variáveis z $\{z^{(i)}\}_{i=1}^{M} \sim p(z)4$

  $\theta_G \leftarrow Adam(\frac{1}{M}\sum_{i=1}^{M}(-D(G(z))), \alpha, \beta_1, \beta_2)$

No estudo realizado por Zhang et al. (2019b), relativamente à utilização de *augmentation*, obtiveram-se melhores resultados quando esta é aplicada através de CR. Como se observa na figura 35, numa GAN simples, quando aplicado *augmentation*, os resultados finais rondam os mesmo valores que sem *augmentation*, sendo que a maior diversidade e quantidades de dados pode evitar o *overfitting* da rede. Quando aplicado *augmentation* através de CR, obtêm-se melhores resultados quantitativos.

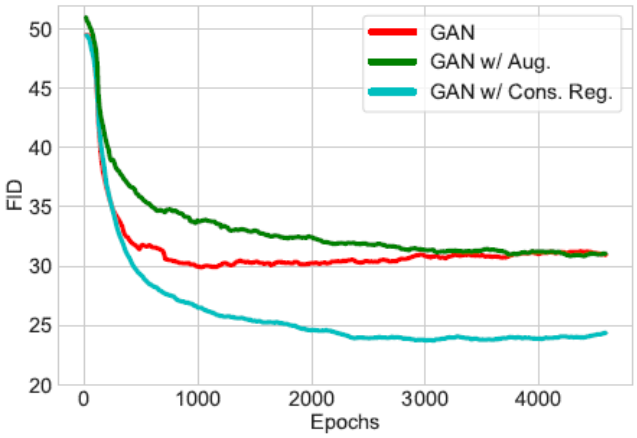


Figura 35: Valor de FID para um modelo base com o recurso a diferentes técnicas de *augmentation*. Imagem adaptada de Zhang et al. (2019b).
Foram realizados 3 treinos numa GAN para CIFAR10, a vermelho um treino com a GAN base, a verde um treino no qual os dados sofrem *augmentation* e a azul um treino com CR. Os dois modelos sem CR obtiveram resultados finais similares, sendo que com *augmentation*, atingiu o seu melhor resultado mais cedo. Quando aplicada *augmentation* através de CR, foram obtidos resultados muito melhores.

Quanto ao tipo de *augmentation*, Zhang et al. (2019b) indica que a aplicação de translações e *flips* nas imagens apresentam melhor FID. Menciona, também, que a aplicação de cortes neutros juntamente com outros tipos de *augmentation* deveria apresentar melhores resultados. No entanto, como o gerador aprende

a gerar imagens com essas transformações e sendo os cortes neutros uma transformação nas imagens mais invasiva, isto é, acrescenta uma característica que não faz parte do *dataset* e, consequentemente, altera significamente as imagens, resulta em valores piores de FID.

3.5 Métricas

Quanto a métricas, como já foi mencionado, foram seleccionadas duas: a *Inception Score* (IS) e *Fréchet Inception Distance* (FID). A seleção destas métricas focou-se em 3 aspetos fundamentais considerados relevantes para o evoluir do trabalho e desenvolvimento do processo, na procura do melhor resultado possível:

- Distribuição das imagens geradas similar à distribuição das imagens reais.
- Diversidade das imagens geradas.
- Controlo sobre a amostragem realizada.
- Estabilidade ao longo do treino.

3.5.1 *Inception Score*

Inception Score (IS) usa uma rede neuronal, *Inceptionv3*, pré-treinada no *dataset ImageNet* para capturar as características das imagens geradas e classificá-las consoante as *labels* de ImageNet e CIFAR10. Esta rede é utilizada para realizar a seguinte operação:

$$IS = exp(E_{x \sim p_g(x)} D_{KL}(p(y|x)||p(y)) \tag{8}$$

Algoritmo 2: Cálculo de *IS*

Entrada: Conjunto de imagens geradas $images$, número de divisões n_splits;
Saída: Média e Desvio padrão de *Inception Score*, IS_mean e IS_std;
Carregar a rede Inceptionv3;
Processar $images$ pela rede e extrair as ativações necessárias para o cálculo;
para *número de divisões* n_splits **faça**
  Obtenção do conjunto de ativações para o *split* atual
  Previsão de $P(y|x)$;
  Cálculo de $P(y)$;
  $KL \leftarrow P(y|x) * (\log P(y|x) - \log P(y))$;
  Soma sobre as classes: $KL = sum(KL, axis = 1)$;
  Média sobre as imagens: $KL = mean(KL)$;
  IS score $\leftarrow \exp(KL)$;
  Armazenamento de IS em *IS score*
$IS\_mean \leftarrow$ média de *IS score*
$IS\_std \leftarrow$ desvio-padrão de *IS score*

Apesar de ser uma métrica bastante utilizada pela comunidade, existem algumas limitações associadas (Barratt and Sharma, 2018), nomeadamente:

- Favorece GANs que memorizam o *dataset*, não sendo sensível a *overfitting*.
- Falhas na deteção de *mode collapse*.
- Favorece modelos que geram bons objetos e não imagens realistas.
- Favorece modelos que aprendem a forma e diversificam imagens.
- É uma métrica assimétrica.
- É afetada pelo resolução da imagem.

O principal problema associado a esta métrica é que apresenta valores bons de IS mesmo depois de ocorrer *mode collapse*. Normalmente, quando as imagens geradas apresentam grandes perturbações e divergência depois de apresentar imagens com qualidade, ocorreu *mode collapse*. No entanto, o IS continua com valores similares ao momento de *mode collapse*, não detetando assim o problema. Esta limitação obriga à avaliação qualitativa por um humano para detetar os problemas no processo de geração.

3.5.2 *Fréchet Inception Distance*

Fréchet Inception Distance (FID) utiliza a mesma rede que a IS para medir a qualidade das imagens e tem muitos dos problemas associados a IS. A diferença está na comparação das diferentes características das imagens reais e geradas, sendo que, neste caso, utiliza distribuições Gaussianas, menos sensíveis aos pequenos detalhes, em imagens de maior resolução.

Esta métrica é calculada pela média e covariância de ambas as distribuições, a de imagens geradas e a do *dataset*. Este cálculo é realizado consoante a seguinte equação:

$$FID = (m - m_w)^2 + Tr(C + C_w - 2(C * C_w)^{1/2}), \tag{9}$$

onde m/c e m_w/c_w correspondem às médias e covariâncias das distribuições das imagens geradas e do *dataset*, respetivamente.

Algoritmo 3: Cálculo de *FID*

Entrada: Conjunto de imagens geradas, conjunto de imagens do *dataset*;
Saída: Valor de FID;
Carregar a rede Inceptionv3;
Processar imagens geradas pela rede e extrair as ativações necessárias para o cálculo;
Processar imagens do *dataset* pela rede e extrair as ativações necessárias para o cálculo;
Cálculo da média dos dois conjuntos de ativações $mu\_amostras$ e $mu\_dataset$;
Cálculo da matriz de covariância dos dois conjuntos de ativações $\sigma\_amostras$ e $\sigma\_dataset$;
$Diff \leftarrow mu\_amostras - mu\_dataset$;
$cov\_media \leftarrow \sqrt{\sigma\_amostras * \sigma\_dataset}$;
$FID \leftarrow Diff^2 + \mathrm{Tr}(\sigma\_amostras + \sigma\_dataset - 2 * cov\_media)$;

Esta métrica apresenta boas avaliações em termos de discriminabilidade, robustez do modelo e eficiência computacional. A sua avaliação é consistente com a decisão humana e é mais robusto e sensível a perturbações nas imagens que o IS.

Analisando a figura 36, pode-se observar o comportamento das duas métricas quando são aplicadas diferentes transformações nas imagens. Globalmente, o FID é mais sensível a estas alterações nas imagens. No caso do IS, apenas quando há mudanças significativas do *dataset*, nomeadamente junção de imagens de diferentes *datasets*, é que se verifica variações na métrica. Quando aplicadas outras transformações, o IS tem pouca sensibilidade, não se verificando um efeito visível no valor da métrica e falhando portanto a detecção do efeito prejudicial das modificações realizadas. No entanto, estas mesmas transformações provocam variações no valor de FID. Nos diferentes casos de perturbação, quanto maior é a intensidade da perturbação, maior a variação do FID (para valores superiores, ou seja, piores), concluindo-se, assim, ser uma métrica mais sensível e robusta.

Em suma, o FID é uma métrica mais robusta a alterações na qualidade das imagens geradas, e é capaz de identificar quando o modelo generativo entra em *mode collapse*. Deste modo, definiu-se o FID como métrica de referência para se comparar os diferentes resultados e tirar conclusões. Não obstante, o valor de IS será também reportado para permitir comparações com outros trabalhos, técnicas e metodologias.

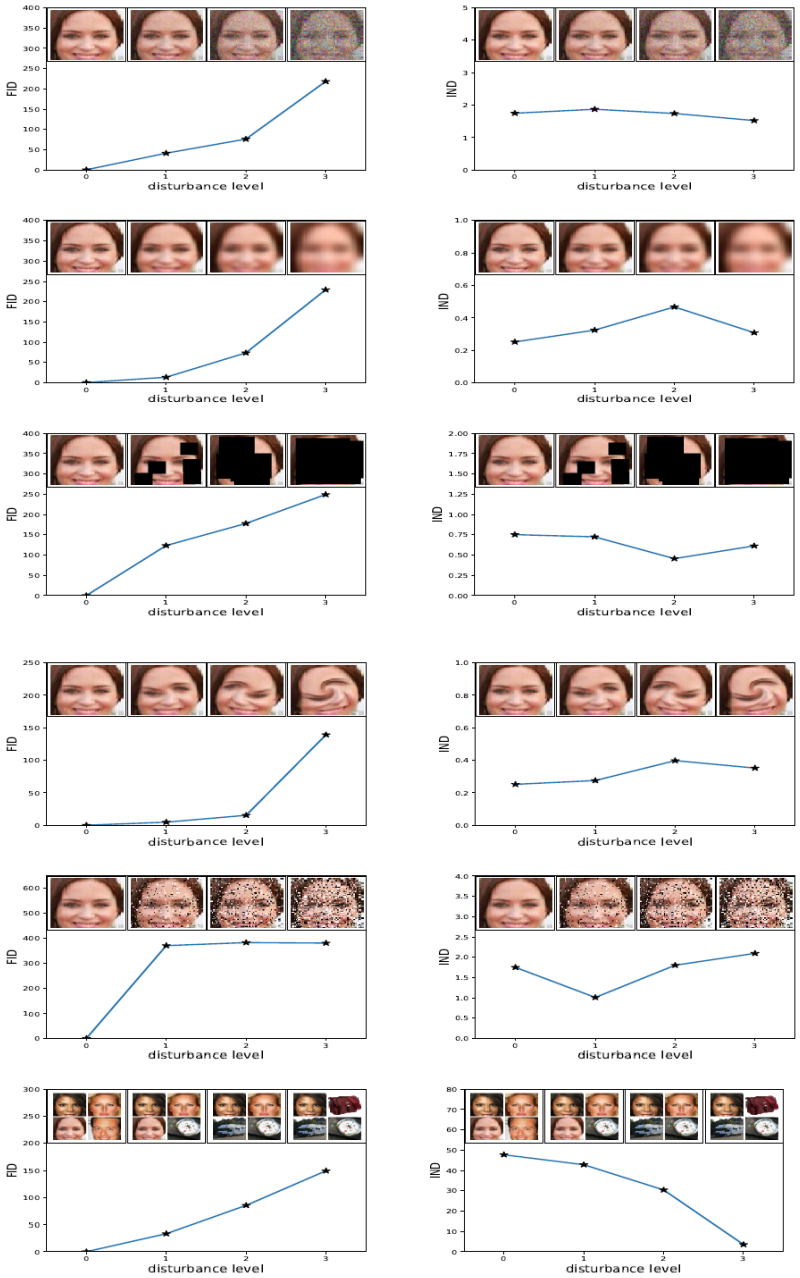


Figura 36: Comportamento do FID e IND após diferentes alterações nas imagens. Imagem adaptada de Heusel et al. (2017).
IND corresponde a *Inception Distance*, uma métrica que traduz o valor de *Inception Score* numa distância. Ambas as métricas aumentam quanto maior for o nível de perturbação. Por ordem, as perturbações correspondem ao seguinte: ruído gaussiano, efeito de névoa gaussiana, retângulos pretos, rotação, ruído tipo sal e pimenta e mistura de 2 *datasets*. O FID captura melhor estas perturbações nas imagens, verificando-se maior variação no seu valor.

4

TESTES E RESULTADOS

Este capítulo é dedicado à apresentação dos testes realizados e respetivos resultados. Inicialmente, serão apresentados os testes e resultados relativos aos primeiros ensaios efetuados na resolução 32x32, na tentativa de gerar imagens com sucesso apesar da pequena resolução e pouco detalhe nas mesmas, e numa segunda fase, serão apresentados os testes e resultados para a resolução 128x128.

Numa primeira fase, foram realizados pequenos testes e ensaios para estudar e validar a *framework* que iria servir de base para o trabalho. Estes testes eram focados em perceber as funcionalidades e o potencial que a *framework* oferecia, de modo a se perceber que métodos e alterações seriam necessárias realizar.

4.1 Resolução 32x32

4.1.1 *Resultados*

Depois de criado o *dataset* C10_AllViews, composto pelos *datasets* CIFAR10 e *In-Vehicle*, realizaram-se diferentes ensaios. À medida que se avançava nos ensaios, tiravam-se conclusões e programavam-se os seguintes.

Todos os ensaios descritos na tabela 3 foram realizados com o mesmo *setup* de parâmetros. Utilizou-se uma configuração mais simples, com tamanho de *batch* (BS) de 50, 4 passos de discriminador, LR para ambas as redes de $2x10^{-4}$, ch de 64, sem aplicar quaisquer técnicas apresentadas na BigGAN, pois trata-se de treinos relativamente simples, com imagens de pequena resolução, não exigindo redes grandes. Esta configuração foi a apresentada no artigo da BigGAN (Brock et al. (2019)), apesar de este não expôr muito detalhe relativamente aos ensaios realizados em CIFAR10.

Tabela 3: Ensaios realizados para resolução a 32x32

Ensaio	Modelo	Dataset	Observações
EV1	BigGAN	CIFAR10	Ensaio com CIFAR10
EV2	BigGAN	C10_AllViews	*Label* do interior de veículos com todas as imagens
EV3	BigGAN	C10_P1P5Views	*Label* apenas com imagens das vistas P1 e P5
EV4	BigGAN	C10_AllViews	Ensaio com dados *augmented* (*flips* e translações)
EV5	BigGAN	C10_AllViews	Ensaio com dados *augmented* (*flips*)

Numa primeira iteração (EV1), foi realizado um ensaio com o *dataset* CIFAR10 como ponto de partida. No segundo ensaio (EV2), utilizaram-se todas as imagens do *dataset in-vehicle* numa *label* apenas, substituindo

uma das *labels* do *dataset* CIFAR10. Pelo facto de imagens de diferentes vistas apresentarem características distintas, decidiu-se num terceiro ensaio (EV3) reduzir as imagens da *label in-vehicle* para duas vistas com características semelhantes. Devido à reduzida quantidade de dados existentes para a *label* e à diferença quando comparado com as outras *labels*, decidiu-se aplicar um ensaio (EV4) com recurso a técnicas simples de *augmentation*, nomeadamente *flips* e translações, utilizando novamente o *dataset* com todas as vistas. Para o quinto e último ensaio com resolução de 32x32, recorreu-se novamente ao *dataset* com todas as vistas, mas apenas aplicando *flips* como *augmentation*. Os resultados obtidos estão sumariados na tabela 4.

Tabela 4: Resultados obtidos nos ensaios descritos na tabela 3

Ensaio	Itr. Collapse	Métricas					
		FID	IS	Max FID	IS	FID	Max IS
EV1	Estável	6,67	8,22	6,44	8,37	6,69	8,37
EV2	40000	25,28	8,69	9,82	8,05	25,28	8,69
EV3	18000	18,16	7,84	18,16	7,84	19,39	8,00
EV4	Estável	6,74	6,94	6,74	6,94	6,86	6,98
EV5	Estável	6,54	8,30	6,43	8,27	6,67	8,33

A coluna *Itr. Collapse* corresponde à iteração em que ocorreu *collapse*. Caso não tenha ocorrido, o treino é identificado como estável. O primeiro par de colunas corresponde ao valor no momento da última iteração registada; o segundo par de valores corresponde aos valores para a iteração com menor valor de FID; e o terceiro par de valores corresponde aos valores para a iteração com maior valor de IS.

Na figura 37, observam-se amostras das diferentes *labels* geradas para o ensaio EV5. Na oitava linha, é possível observar imagens do interior de veículos com certa qualidade, apesar das amostras terem uma resolução baixa. Observam-se certos detalhes nas imagens como os bancos, volante, teto e janelas.

4.1.2 *Discussão*

Ao longo dos ensaios realizados, foram tomadas decisões e definidos novos ensaios com o objetivo de melhorar os resultados quantitativos e qualitativos. Consoante os problemas que iam surgindo, estudavam-se os resultados e procuravam-se alternativas que permitissem melhorar. Foi considerada a métrica FID como métrica discriminativa para os melhores resultados, como mencionado acima, por ser mais robusta comparativamente ao IS.

Nas figuras 38 e 39, é possível observar a evolução das métricas ao longo dos processos de treino. O ensaio EV5 foi aquele que mais semelhanças teve com o ensaio realizado com o *dataset* CIFAR10, quer em termos de métricas, como estabilidade ao longo do treino. Observando qualitativamente as amostras produzidas por este ensaio (figura 37), não se observam transformações resultantes do processo de *augmentation* utilizado durante o treino, melhorando as amostras qualitativamente. Podemos concluir, através das imagens geradas e a evolução das métricas, que o ensaio EV5 apresenta o melhor resultado para a resolução 32x32.

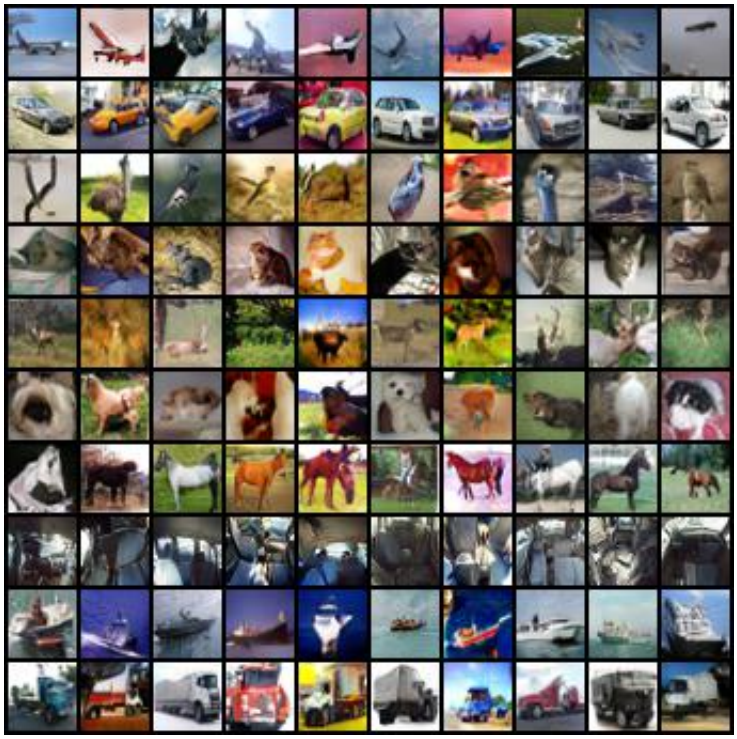


Figura 37: Amostras das diferentes *labels* para o ensaio EV5.
Na oitava linha, observam-se imagens geradas do interior de veículos.

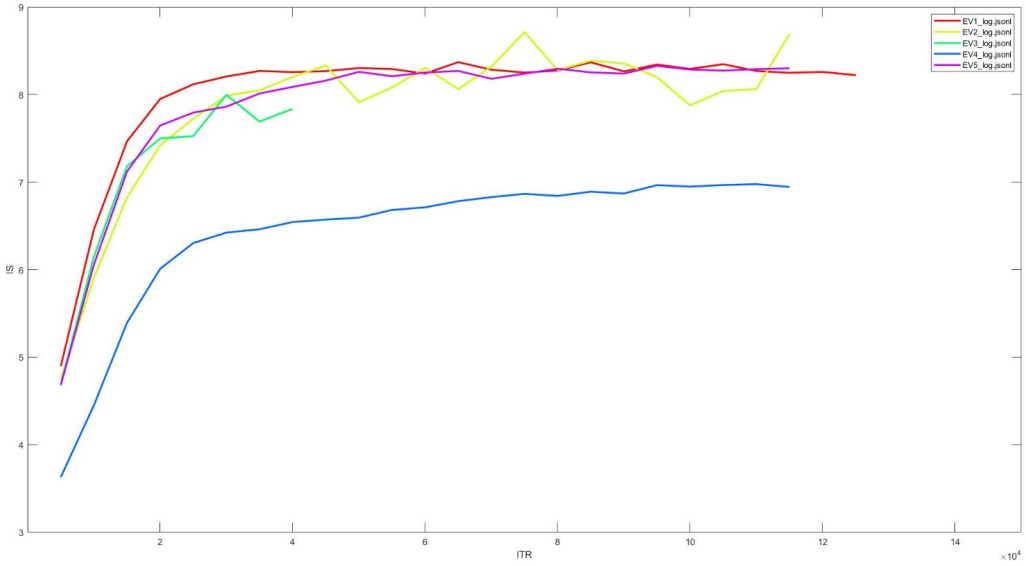


Figura 38: Evolução do IS ao longo do treino e comparação entre os ensaios descritos na tabela 3.
Para os ensaios EV2 e EV3, observa-se uma evolução muito inconstante e variável ao longo do processo de treino, associando-se à instabilidade do treino. Quando aplicado *augmentation* (EV4), verifica-se uma melhoria quanto à estabilidade de treino, mas não atinge os melhores resultados possíveis em termos de IS. No ensaio realizado sem aplicação de translações (EV5), pode-se verificar estabilidade no treino e valores muito próximos dos resultados obtidos para o *dataset* original CIFAR10 (EV1).

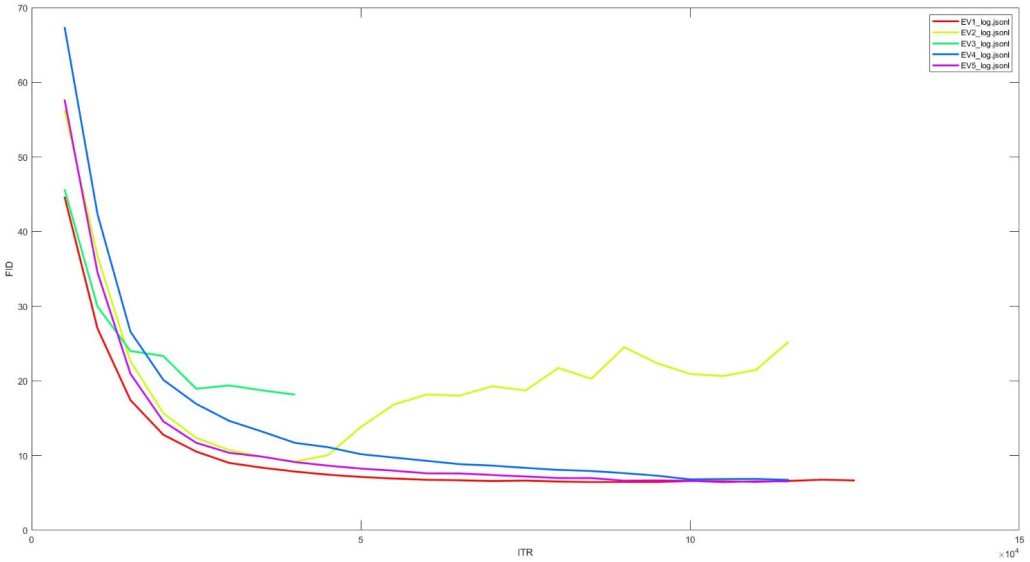


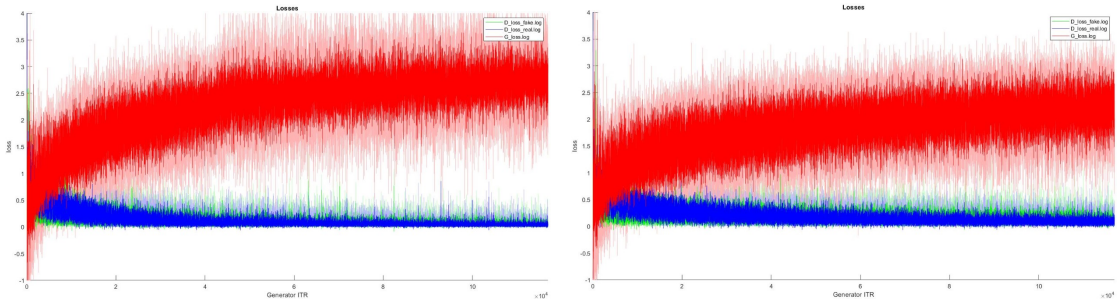
Figura 39: Evolução do FID ao longo do treino e comparação entre os ensaios descritos na tabela 3.
Nos ensaios EV2 e EV3, tal como para IS, observa-se uma evolução instável do FID. Para os outros 3 ensaios, observa-se estabilidade e melhorias ao longo do treino, sendo que para EV4 (*augmentation* com *flips* e translações), a rede aparenta ser mais lenta no processo de aprendizagem, pois demora mais iterações a convergir para o valor de FID obtido no ensaio EV5 (*augmentation* com *flips* apenas).

Na figura 40, observam-se amostras do ensaio EV4, na qual são visíveis amostras com barras pretas. Neste ensaio, aplicou-se *augmentation*, nomeadamente *flips* e translações. Nas amostras geradas, observam-se o efeito das translações (deslocação da imagem deixando espaços com valor nulo, ou seja pretos, na imagem alterada), sendo que aplicando este tipo de *augmentation* às imagens do *dataset*, as redes assumem que as barras pretas fazem parte do conjunto de características das imagens. Este tipo de *augmentation* revelou-se prejudicial na geração das imagens, alterando as propriedades destas.

Como indica a tabela 3, os ensaios EV2 e EV5 foram realizados sobre o mesmo *dataset*. No entanto, no ensaio EV5 aplicou-se técnicas de *augmentation* para aumentar a quantidades de dados, aplicando *flips* às imagens. Analisando a tabela 4, observa-se estabilidade nos resultados obtidos para o ensaio EV5. Reproduzindo a evolução de *loss* para os dois ensaios, observa-se que o valor médio e a variação da *loss* do ensaio EV5 (figura 41b), com maior quantidade de dados, é menor, promovendo a estabilidade do treino do modelo.

Globalmente, foram obtidos bons resultados nesta resolução. Comparativamente com o *dataset* CIFAR10, foram obtidos resultados muito similares, um bom indicador do sucesso destes ensaios. Possibilitou-nos ainda observar o problema proveniente da quantidades de dados, sendo que quando aplicados métodos de *augmentation*, os resultados melhoraram consideravelmente. No entanto, nem todo o tipo de *augmentation* é benéfico para o treino, observando-se que a aplicação de translações prejudica os resultados obtidos (não só quantitativamente, mas também pelo facto de as imagens geradas apresentarem as mesmas bandas pretas criadas pelo processo de translação durante a *augmentation*).

Figura 40: Amostras das diferentes *labels* para o ensaio EV4.



(a) Ensaio EV2. (b) Ensaio EV5.

Figura 41: Evolução da *Loss* para os ensaios EV2 e EV5.
Analisando os dois gráficos, observa-se que a loss no ensaio EV5 tem um valor médio mais baixo e menor variação na sua evolução, fator que influencia a estabilidade do treino.

4.2 Resolução 128x128

Depois de realizados os ensaios a baixa resolução, procurou-se aumentar a complexidade e, consequentemente, a qualidade das imagens geradas. Procedeu-se assim a ensaios numa resolução de 128x128.

4.2.1 *Resultados*

Tendo em conta o *dataset* utilizado na resolução mais baixa, criou-se um *dataset*, composto por imagens do ImageNet e *In-Vehicle*. O *dataset* criado apresenta semelhanças com o CIFAR10, nomeadamente no número de imagens e nas *labels* que o constituem.

Depois de criado o *dataset*, realizaram-se ensaios/testes iterativos de modo a encontrar a melhor parametrização, isto é, aquela que permite realizar um treino estável e, consequentemente, melhor. Estes ensaios foram realizados com recurso ao *dataset* ImageNet_128_Double e ao modelo BigGAN (Brock et al., 2019).

Desde modo, tal como se observa na tabela 5, mantendo uma *batch size* (BS) de 50, fez-se variar as diferentes propriedades da GAN que, segundo Brock et al. (2019), permitia obter melhorias nos resultados. Partiu-se, inicialmente, da parametrização utilizada para a resolução mais baixa (EV6). Numa segunda iteração (EV7), foi aumentado o tamanho da rede. Num terceiro ensaio (EV8), reduziu-se novamente ao tamanho da rede, mas alterou-se a regularização, usando-se agora a sugerida no artigo que apresentaria melhor resultado. Na iteração seguinte (EV9), introduziu-se a técnica *shared embedding*. No ensaio seguinte (EV10), aplicou-se espaços latentes hierárquicos. Para terminar esta primeira fase de testes, realizaram-se mais 2 ensaios nos quais se variou a LR para os valores sugeridos no artigo. Durante o processo de treino, a convergência da rede foi verificada, e sempre que ocorreu *mode collapse*, o treino foi interrompido.

Tabela 5: Evolução dos ensaios para seleção da melhor parametrização na resolução 128x128

Ensaio	BS	D *Steps*	LR D	LR G	CH	Init	*Shared*	Hier	FID	IS
EV6	50	4	$2x10^{-4}$	$2x10^{-4}$	64	N02			78,97	9,41
EV7	50	4	$2x10^{-4}$	$2x10^{-4}$	96	N02			75,00	9,27
EV8	50	4	$2x10^{-4}$	$2x10^{-4}$	64	ortho			70,27	9,89
EV9	50	4	$2x10^{-4}$	$2x10^{-4}$	64	ortho	ativo		64,45	9,99
EV10	50	4	$2x10^{-4}$	$2x10^{-4}$	64	ortho	ativo	ativo	**61,63**	10,13
EV11	50	2	$2x10^{-4}$	$2x10^{-4}$	64	ortho	ativo	ativo	76,08	10,45
EV12	50	2	$2x10^{-4}$	$1x10^{-4}$	64	ortho	ativo	ativo	68,25	10,49

BS: *batch size*; D *Steps*: número de passos do discriminador por cada passo do gerador; LR D: LR do discriminador; LR G: LR do gerador; CH: número de canais da rede (aumento da rede); *Init*: tipo de inicialização (N02 corresponde a inicialização aleatória por amostragem de uma distribuição normal e ortho a inicialização *orthogonal*); *Shared*: aplicação de *shared embeddings*; *Hier*: aplicação de espaços latentes hierárquicos.

Como já mencionado, a utilização de *datasets* equilibrados melhora consideravelmente o treino e, consequentemente, os resultados obtidos. Então, no ensaio EV13 decidiu-se remover a *label* com as imagens do interior de veículos, passando o *dataset* a ter apenas 9 classes mas um número mais equilibrado de imagens por classe. Este ensaio obteve 44,47 para FID e 11,23 para IS, verificando-se melhoria no resultado obtido e comprovando assim a importância do equilíbrio no número de imagens por classe.

Deste modo, procedeu-se à realização de novos ensaios no quais se fez variar o *dataset*. A ideia passou por criar *datasets* mais equilibrados, nos quais se distribuiu as imagens por *labels* mais distintas (com base nos *synsets* originais do *dataset* ImageNet), mantendo aproximadamente o mesmo número de imagens por *label* e com o mesmo estilo de variabilidade. Sendo assim, foram criados mais 2 *datasets* (linhas 5 e 6 na tabela 2 do capítulo Métodos, ponto 3.1), nomeadamente ImageNet_Balance (EV14) e ImageNet_Balance_Double (EV16).

Para o *dataset* ImageNet_Balance, considerou-se ainda uma segunda variante com as imagens do interior de veículos divididas em 3 *labels* (teto, vistas frontais e vistas laterais), sendo esta variante utilizada no ensaio EV15. Para estes ensaios, foi utilizada a configuração que apresentou os melhores resultados nos ensaios anteriores (EV10), com o *dataset* ImageNET_128_Double. Os resultados correspondentes estão expostos na tabela 6.

Tabela 6: Resultados obtidos para os ensaios realizados com diferentes variantes do *dataset ImageNet + In+vehicle* na resolução 128x128

Ensaio	Itr	FID	IS	IS Dataset	% IS
EV10	27500	61,63	10,13	21,52	47,07
EV14	39000	**27,43**	16,39	25,23	**64,96**
EV15	35000	35,13	16,01	25,23	**63,44**
EV16	52000	27,31	18,07	38,82	46,54

IS *Dataset* corresponde ao valor de IS para as imagens reais do *dataset* utilizado. Este valor serve de referência sendo que as imagens geradas tentam imitar o *dataset*, pelo que idealmente terá um valor aproximado de IS, daí a coluna % IS que dá um valor relativo de comparação entre o valor obtido para as imagens geradas e as imagens do *dataset*.

De modo a facilitar o processo de geração de amostras das nossas *labels*, decidiu-se utilizar o *dataset* do ensaio EV15 nos ensaios seguintes, pois representa as nossas classes de forma mais distribuída, permitindo assim obter uma maior número de amostras e com maior variabilidade. De notar que, apesar de este ter obtido um valor de FID pior, o valor de %IS é similar ao ensaio EV14 e superior ao ensaio EV16.

Depois de encontrada uma parametrização e um *dataset* que se enquadra no pretendido, foram acrescentadas novas técnicas ao algoritmo inicial (tabela 7). Inicialmente, aplicou-se *self-attention* ao modelo. Numa segunda abordagem, aplicou-se *consistency regularization*, com aplicação de *flips* e translações, alternando o uso de *cutout* de um ensaio para o outro. Na abordagem seguinte, retirou-se a implementação de *consistency regularization*, implementando agora *differentiable augmentation*, aplicando-se como *augmentation* translações e *cutout*. No último ensaio, aplicou-se todos os métodos em conjunto.

Tabela 7: Resultados obtidos para os ensaios realizados com os diferentes métodos implementados e diferentes tipos de *augmentation*, utilizando o *dataset* ImageNet_Balance e imagens com resolução de 128x128

Ensaio	Método	DiffAug	CR Augmentation	Métricas	
				FID	IS
EV17	BigGAN s/ Attention			27,43	16,38
EV18	BigGAN c/ Attention			25,90	16,38
EV19	CR - BigGAN		flips, translações	**18,23**	**17,19**
EV20	CR - BigGAN		flips, translações, cutout	24,23	16,40
EV21	BigGAN + DiffAug	cutout, translações		169,18	3,06
EV22	CR - BigGAN + DiffAug	cutout, translações	flip, translações	56,50	12,23

Na figura 42, observam-se amostras das diferentes *labels* geradas para o ensaio EV19. Nas primeiras três linhas é possível observar amostras do interior de veículo. Observam-se diferentes características associadas ao interior de veículos, nomeadamente o teto, bancos, encostos de cabeça, janelas, entre outros.

Figura 42: Amostras das diferentes *labels* para o ensaio que apresenta melhor resultado, CR-BigGAN com *flips* e translações.
Nas primeiras três linhas, observam-se diferentes amostras geradas das *labels* do interior de veículos.

4.2.2 *Discussão*

Considerando os resultados expostos na tabela 5, a única diferença entre os 2 primeiros ensaios está no tamanho das redes. Apesar de se ter observado um melhor FID no ensaio com um maior número de canais, o aumento em termos de custo computacional não compensou o ganho observado, tendo-se optado por manter a parametrização do primeiro ensaio.

O ensaio com melhor resultado em termos de IS foi o EV12 e, em relação ao FID, o EV10. No entanto, pelo facto de o FID ser uma métrica mais robusta que a IS, considerou-se o ensaio EV10 como a melhor parametrização para os ensaios realizados posteriormente.

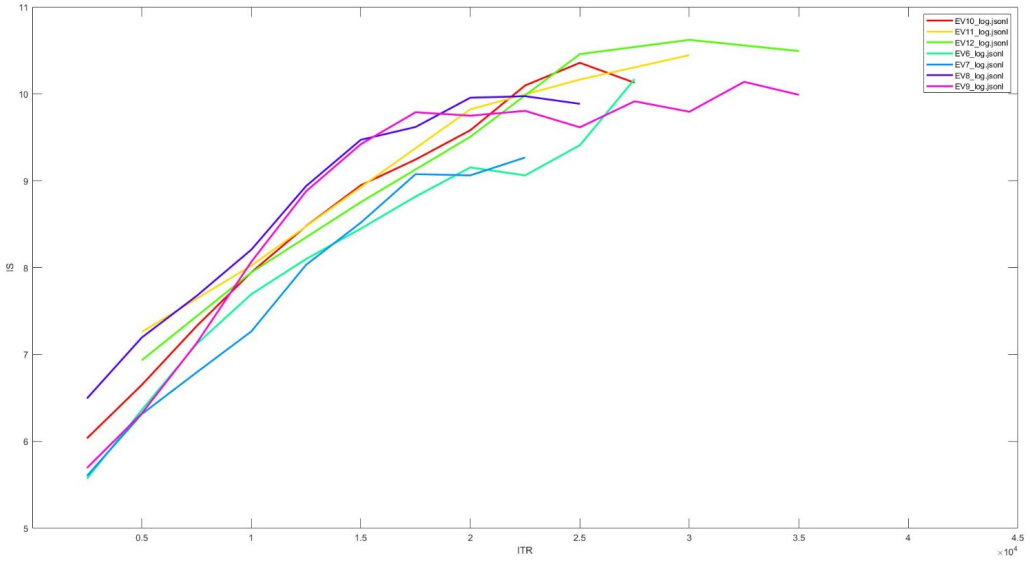


Figura 43: Evolução do IS ao longo do treino para os ensaios descritos na tabela 5.
A evolução do IS, para todos os ensaios, é relativamente estável, observando-se apenas para os ensaios EV6 e EV9 uma certa instabilidade. O ensaio EV12 apresenta melhor resultado em termos de IS máximo registado.

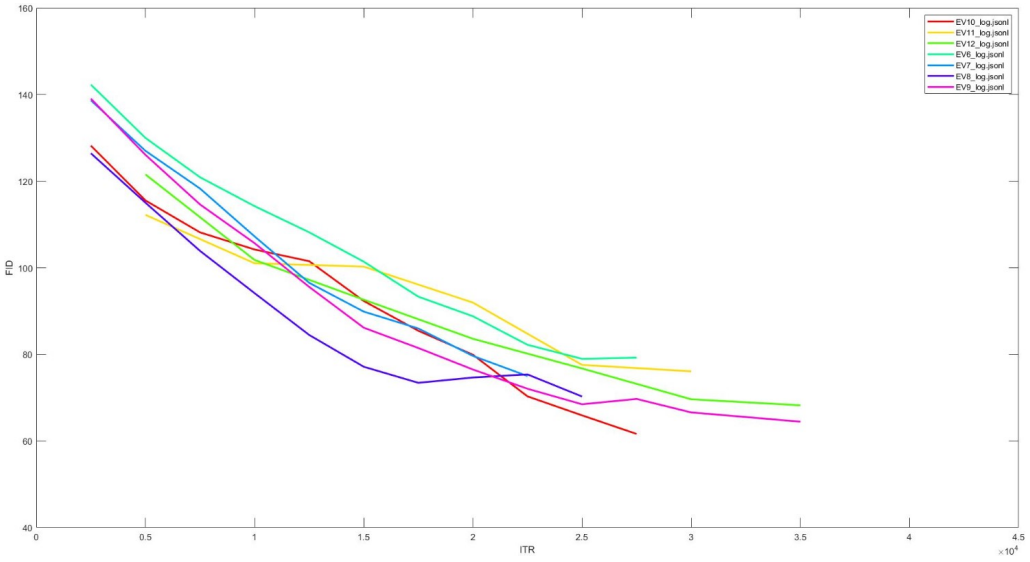


Figura 44: Evolução do FID ao longo do treino para os ensaios descritos na tabela 5.
Os ensaios convergiram normalmente, reduzindo o valor de FID gradualmente. EV10 apresentou o melhor resultado para FID.

Quando diminuído o número de passos do discriminador por passo do gerador (ensaio EV11), verificou-se uma ligeira melhoria do IS, mas com um aumento acentuado do FID (ou seja, pior diversidade e qualidade). Este resultado demonstra que, uma vez que não se alterou a LR do gerador para compensar a diferença do número de passos entre discriminador e gerador, o treino tornou-se desbalanceado. O discriminador converge assim relativamente mais devagar, ou seja, dá menos passos para o mesmo número de passos do gerador, sem alteração do LR, não existindo equilíbrio entre os dois, prejudicando consideravelmente o treino.

Não obstante ser o ensaio com melhor resultado, as imagens obtidas ainda apresentavam imensas debilidades, verificando-se *mode collapse* nas amostras da nossa *label* (ver figuras 45 e 46, correspondentes aos ensaios EV10 e EV12, respetivamente).

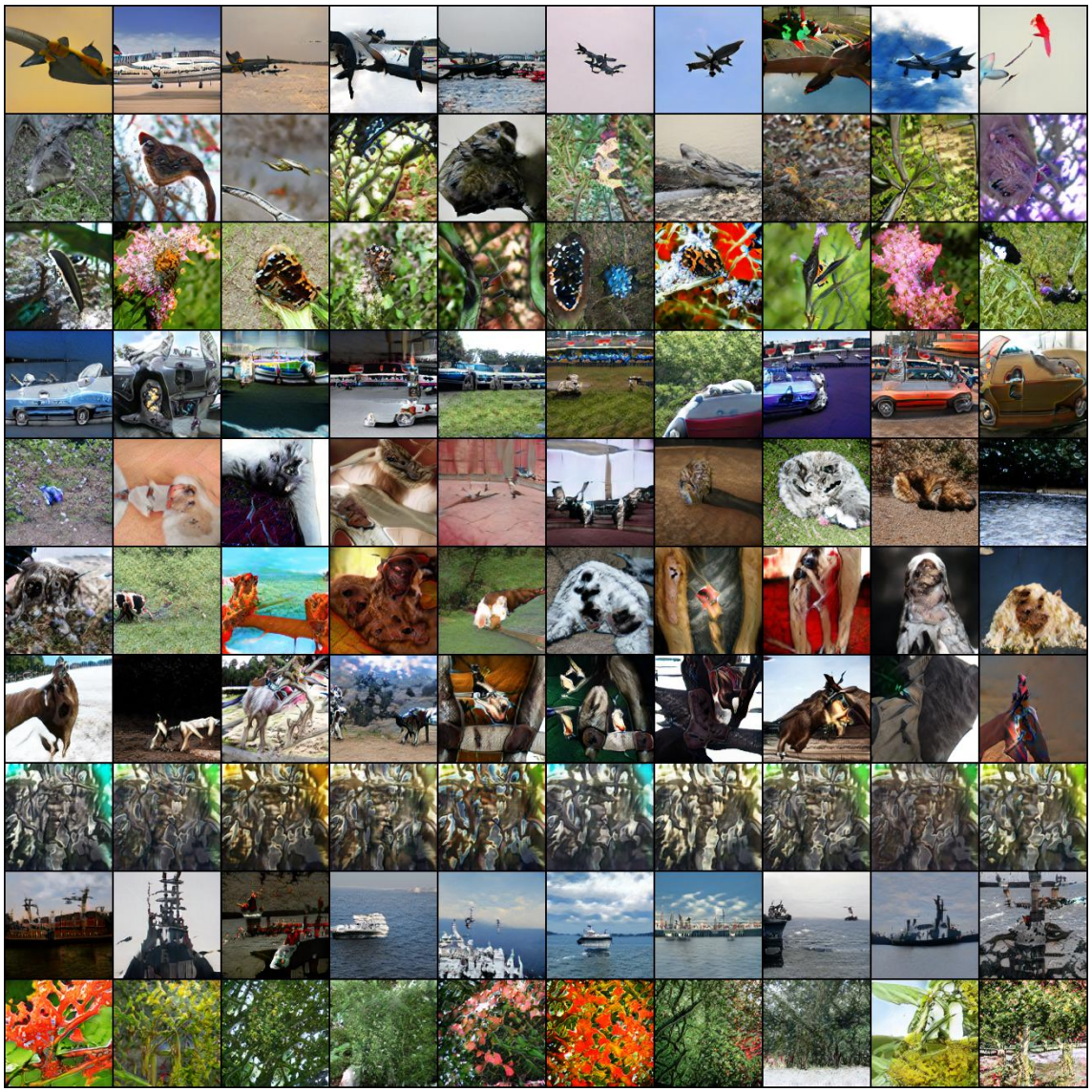


Figura 45: Amostras das diferentes *labels* para o ensaio EV10.
Observando a oitava linha, correspondente à *label* de interesse, observa-se *mode collapse* nas amostras geradas.

No ensaio EV13, decidiu-se então remover a classe de interesse e verificar o efeito que poderia ter a utilização de um *dataset* equilibrado. Este continha aproximadamente o mesmo número de imagens por classe e com o mesmo estilo de variabilidade, o que permitiu manter o treino mais estável durante o dobro de iterações, o que originou um FID menor e IS maior (apesar de não comparável com os ensaios anteriores pelo facto de utilizar um *dataset* distinto). Foi então que introduzimos a utilização de um *dataset* maior e mais equilibrado.

Tratando-se agora de diferentes *datasets*, não se pode comparar diretamente os valores das métricas. Deste modo, calculou-se o IS do *dataset* para se realizar uma análise percentual e assim retirar conclusões.

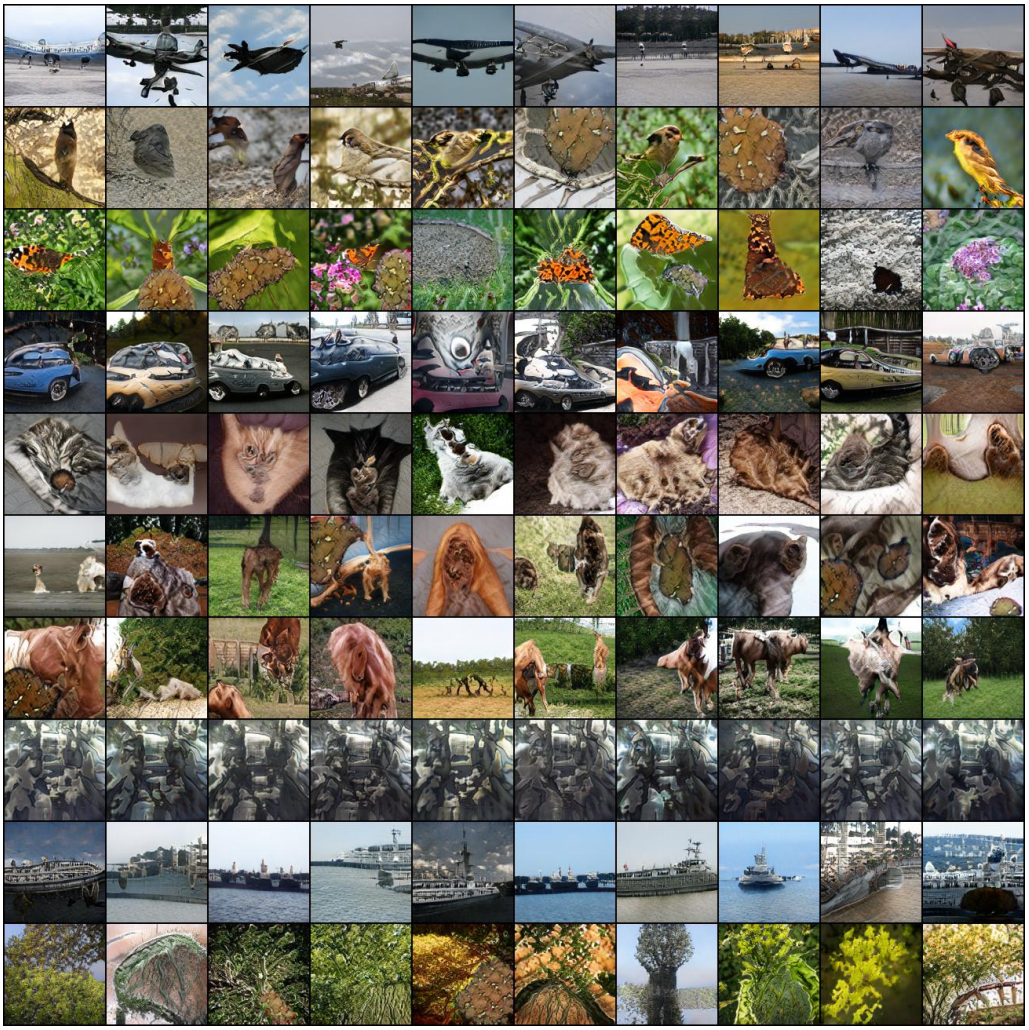


Figura 46: Amostras das diferentes *labels* para o ensaio EV12.
Observando a oitava linha, correspondente à *label* de interesse, observa-se *mode collapse* nas amostras geradas.

Analisando a tabela 6, o ensaio EV14 apresentou melhores resultados percentuais. Verifica-se também que com a maior quantidade de dados, o treino apresenta uma maior estabilidade e prolonga-se assim por mais tempo (antes de ocorrer *mode collapse*).

Comparando o ensaio EV14 com o ensaio EV16, no qual a diferença está na quantidade de imagens e *labels* (o dobro para EV16), observa-se um melhor valor de FID para EV16. No entanto, este resultado é obtido após um número de iterações maior, tal como mostra a figura 48. Tendo por base o tempo computacional exigido, a melhoria observada não foi considerada significativa, e optou-se por utilizar um *dataset* de menores dimensões nos restantes ensaios. Observando o valor percentual de IS, como o *dataset* é maior, consequentemente o seu valor de IS aumenta. No entanto, o valor percentual de IS no treino é muito inferior ao valor de IS do *dataset*, cerca de 18%.

Analisando os resultados relativos aos ensaios EV14 e EV15, pode-se identificar uma diferença considerável de FID. No entanto, analisando a evolução das métricas (figuras 47 e 48), pode-se observar que, para ambos os ensaios, as curvas evoluíram de forma semelhante. Como o ensaio EV15 foi parado numa fase anterior

(pois utilizou-se apenas a análise qualitativa da classe de interesse como referência para interromper o treino), muito provavelmente o FID continuaria a tender para valores similares aos do ensaio EV14.

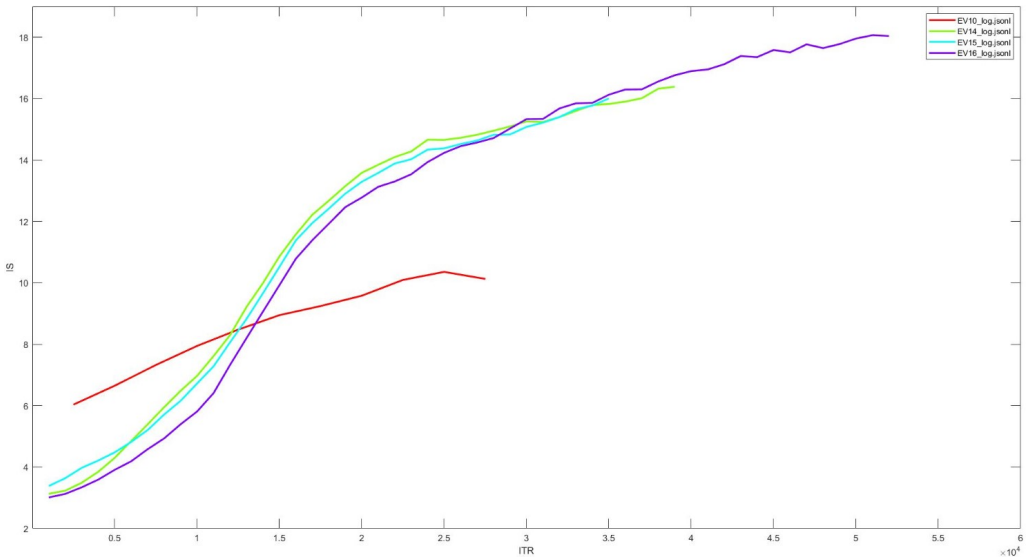


Figura 47: Evolução do IS ao longo do treino para os ensaios descritos na tabela 6.
O IS dos diferentes ensaios evoluiu normalmente, sendo que o ensaio com o *dataset* utilizado inicialmente apresenta o pior resultado.

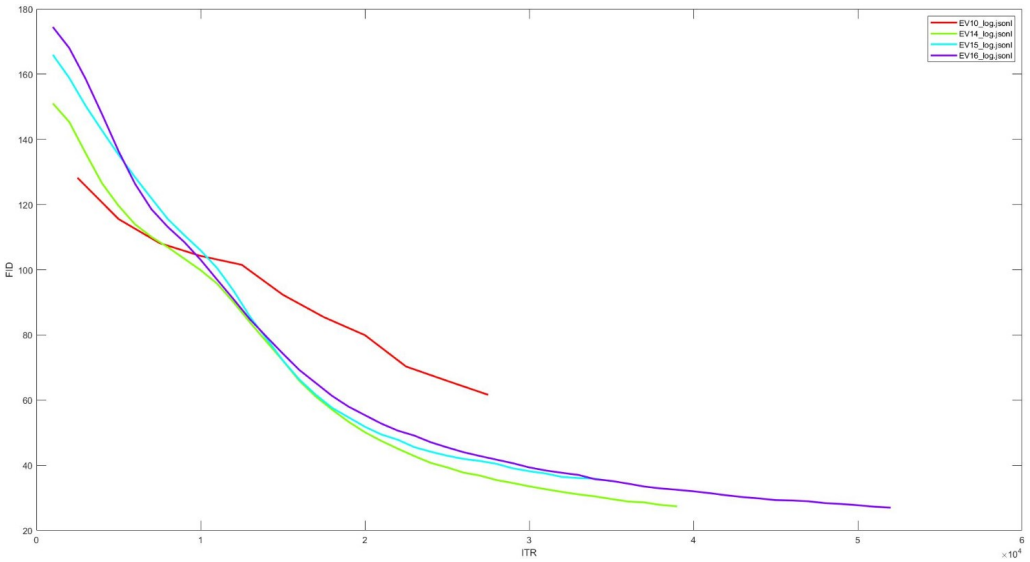
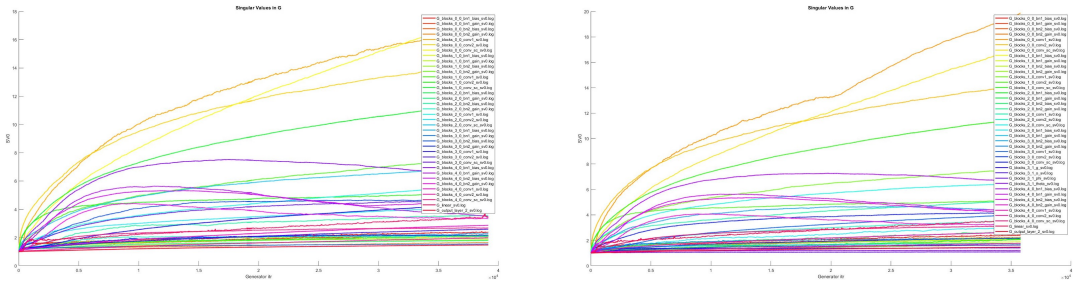


Figura 48: Evolução do FID ao longo do treino para os ensaios descritos na tabela 6.
O treino convergiu normalmente, havendo uma diminuição gradual do valor de FID em todos os ensaios. EV10 apresentou o melhor resultado para o FID.

Comparando agora a evolução dos gradientes do gerador para ambos os ensaios (figura 49), existe um pormenor que se destaca. Para o mesmo gradiente, observa-se, no ensaio EV15, que a curva, por volta das 20000 iterações, sofreu uma anomalia e diminuiu a taxa de crescimento com que estava a convergir até este ponto. Esta ocorrência pode estar associada a arredondamentos que ocorram no processo de atualização

dos pesos da rede, na possibilidade de a rede, neste passo, ter treinado com uma *batch* desequilibrada, entre outros fatores.



(a) Ensaio EV14. (b) Ensaio EV15.

Figura 49: Evolução dos gradientes do gerador para os ensaios EV14 e EV15.
Observa-se uma anomalia no ensaio EV15 perto das 20000 iterações, resultando numa variação na taxa de crescimento divergindo do resultado pretendido.

Considerando os resultados expostos na tabela 7, a única diferença entre os 2 primeiros ensaios está na aplicação de *self-attention* nas redes. A utilização de *self-attention* apresentou melhor resultado de FID, levando à sua utilização a partir deste ponto.

O ensaio com melhor resultado foi a utilização combinada de *consistency regularization* com a BigGAN. A combinação de *augmentation* por *flips* e translações apresentou um FID de $18,23$ e um IS de $17,19$, sendo este o melhor resultado obtido em todos os ensaios realizados.

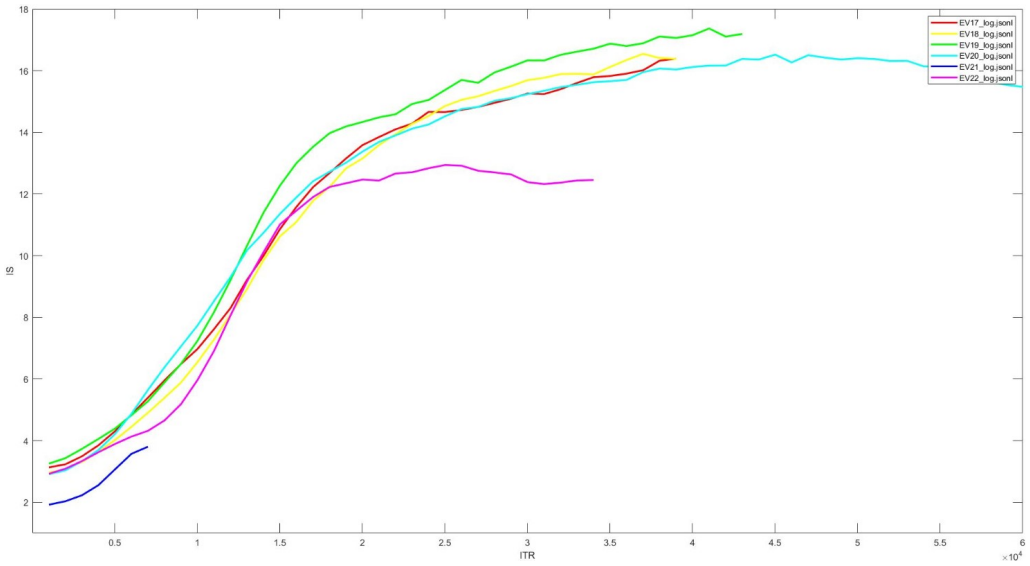


Figura 50: Evolução do IS ao longo do treino para os ensaios descritos na tabela 7.
A aplicação de *DiffAug* é pior quando comparada com os restantes métodos. Observa-se também a influência da *augmentation cutout* quando aplicada no método CR-BigGAN. O ensaio com o melhor valor de IS corresponde ao que aplica *consistency regularization* com BigGAN, com *flips* e translações como *augmentation*.

Analisando os gráficos das figuras 50 e 51, observa-se que a aplicação de *DiffAug* não é benéfica para a rede. Apresenta os piores resultados entre os diferentes métodos tanto para IS como para FID. De facto, nos ensaios com *DiffAug* verificou-se sempre uma menor estabilidade do treino, sendo que mesmo no ensaio conjunto com

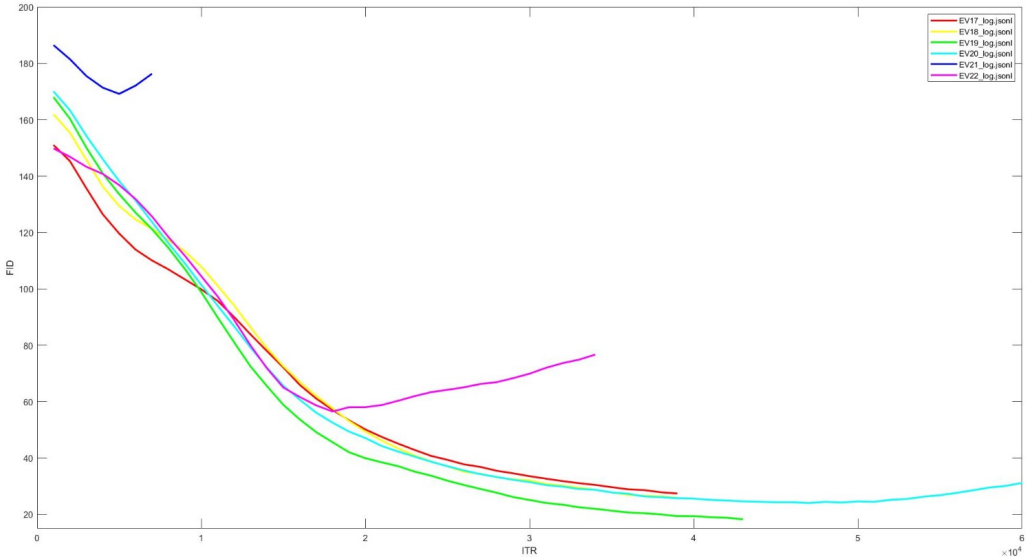


Figura 51: Evolução do FID ao longo do treino para os ensaios descritos na tabela 7.
O ensaio realizado com BigGAN e *DiffAug* apresenta uma linha mais curta pelo facto de ter progredido, a certo ponto, para valores piores de FID e, consequentemente, o seu treino ter sido interrompido. O mesmo acontece quando aplicado em conjunto a *consistency regularization* e a *differentiable augmentation*. É também observável a influência de *cutout* na *augmentation* entre os 2 ensaios de CR-BigGAN. O ensaio que aplica a BigGAN com *consistency regularization*, recorrendo apenas a *flips* e translações como *augmentation*, apresentou o melhor resultado.

consistency regularization (que melhora a estabilidade do treino) se verificou um pior resultado. No entanto, pelo facto de se utilizar muita *augmentation* com a aplicação de *DiffAug*, de modo a aumentar a quantidade de dados disponibilizados à rede, pode-se provocar um problema de *under-fitting*. Este acontece quando o modelo não aprendeu o suficiente dos dados de treino (seja pela falta de imagens para a complexidade do problema, ou pelo facto de a variabilidade destas ser muito grande para a capacidade da rede), resultando numa fraca generalização dos dados e amostras com pouca qualidade, confiança e variabilidade, tal como se observa na figura 52.

Comparando agora os ensaios realizados com a CR-BigGAN sem ou com aplicação de *cutout* (ensaios EV19 e EV20, respetivamente), observa-se quantitativamente, na tabela 7, que a sua não utilização se reflete num melhor resultado.

Qualitativamente, observa-se, através das amostras expostas na figura 54, que o detalhe do interior de veículos não fica tão claro como se verificou sem aplicar *cutout* (figura 42). Certos detalhes como as palas do teto, os limites dos bancos ou os encostos de cabeça não são observáveis de forma clara.

Observando a evolução da *loss*, verifica-se que a *loss* do discriminador apresenta maiores oscilações no ensaio com o recurso à *augmentation* por *cutout* (figura 53). A transformação aplicada pode ocultar pormenores fundamentais das imagens, confundindo o discriminador de tal modo que seja necessário uma compensação maior a cada iteração de treino. Todos as outras transformações não influenciam as características observáveis no *dataset*.

No entanto, como já mencionado, a aplicação de *augmentation* aos dados de entrada induz o gerador em erro levando-o a gerar amostras com as transformações aplicadas. Na figura 42 observam-se pequenas barras nos limites das amostras. Isto é o resultado da aplicação de translações na distribuição de dados de

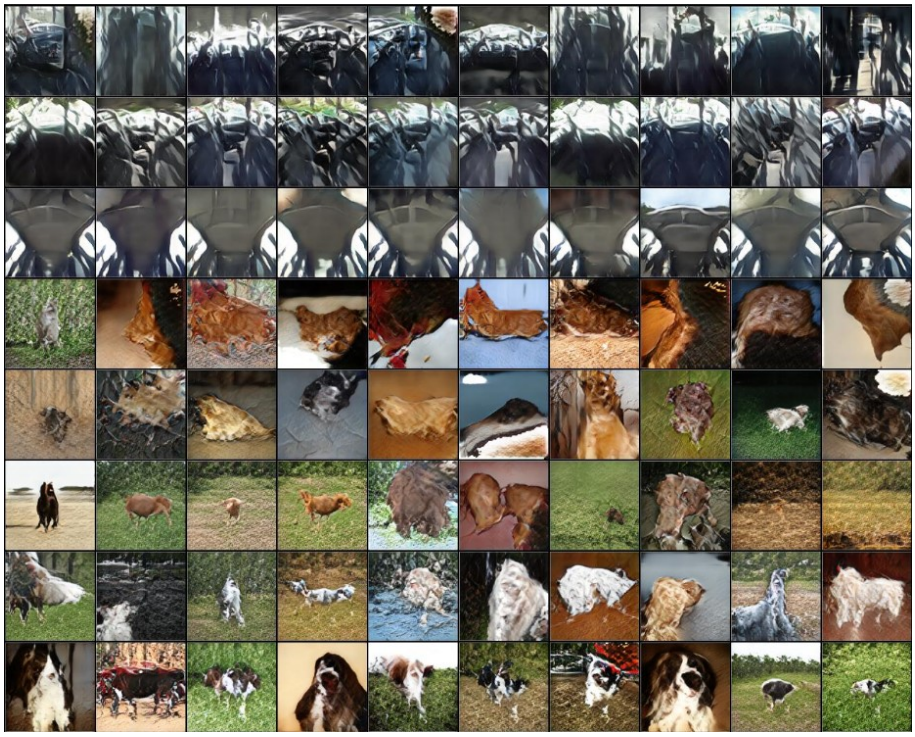
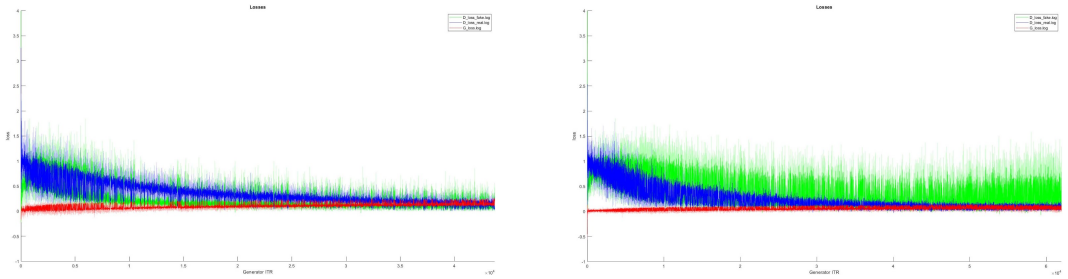


Figura 52: Amostras de diferentes *labels* para o ensaio EV22.
Estas amostras apresentam pouco detalhe das respetivas *labels*, demonstrando a dificuldade da rede em aprender a distribuição de dados. A aplicação de *augmentation* nas imagens do *dataset*, quando aplicado em demasia, pode-se tornar prejudicial ao treino da rede.

entrada. Tal efeito está descrito no capítulo Métodos, quando se descreve a *Differentiable Augmentation* e *Consistency Regularization*.



(a) *Loss* de CR-BigGAN com *flips* e translações (EV19). (b) *Loss* de CR-BigGAN com *flips, cutout* e translações (EV20).

Figura 53: Evolução das *losses* para os ensaios realizados com a CR-BigGAN.
Analisando os dois gráficos, observa-se que as losses do gerador e discriminador no ensaio em que se aplica *cutout* têm maiores oscilações, influenciando a estabilidade do treino.

Globalmente, foram obtidos resultados satisfatórios. Conseguiu-se gerar amostras nas quais é possível observar as características do interior de veículos. Com recurso a *augmentation*, as amostras são prejudicadas pelo tipo de transformações aplicadas, beneficiando daquelas menos invasivas ao conteúdo original da imagem. A variação da parametrização do modelo permitiu explorar diferentes aspectos, mas caso não existisse limitações de recursos computacionais, poderia-se estender os ensaios e explorar as configurações

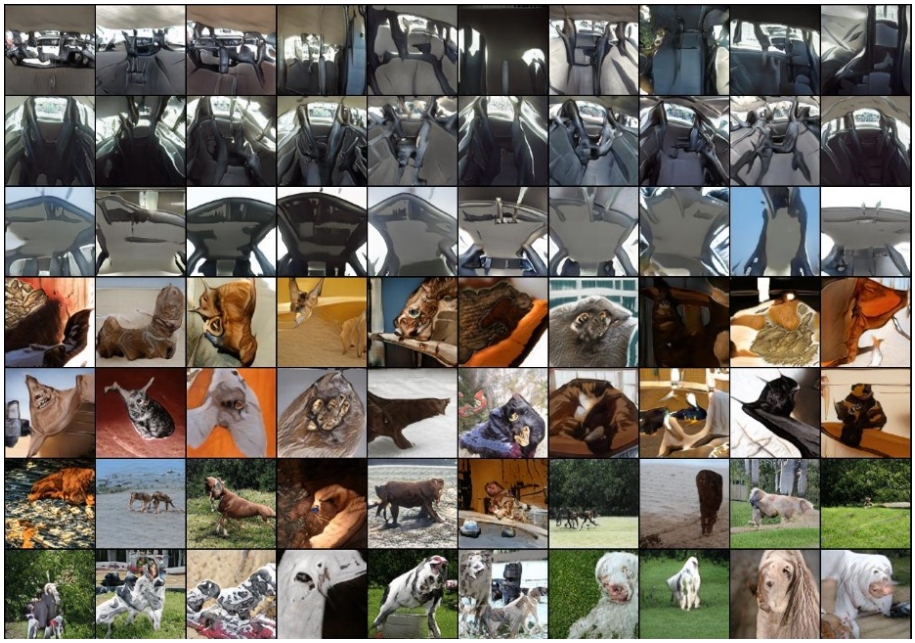


Figura 54: Amostras de diferentes *labels* para o ensaio com CR-BigGAN, aplicando *cutout*.
As amostras presentes nas primeiras três linhas, comparativamente com as da figura 42, apresentam algum detalhe, porém, não é tão claro certos pormenores como limites de bancos ou as palas presentes no teto.

que apresentaram os melhores resultados (nomeadamente com o *dataset* de maiores dimensões e um número de canais superior).

5

CONCLUSÕES E TRABALHO FUTURO

O trabalho desenvolvido nesta dissertação teve como objetivo estudar e avaliar diferentes GANs para a geração de imagens do interior de veículos.

Numa fase inicial, foi necessário recolher um *dataset* que representasse o interior de veículos. Deste modo, recolheu-se um pequeno conjunto de fotografias e foram criados métodos para processar as imagens recolhidas. As imagens recolhidas foram suficientes para atingir o objetivo pretendido, sugerindo-se contudo no futuro aumentar o *dataset* criado, em termos de quantidade e variabilidade das imagens, de modo a potenciar os resultados obtidos. Também a aplicação de *augmentation* permitiu aumentar a quantidade e variedade de dados, podendo-se futuramente explorar mais esta técnica.

Quanto à GAN utilizada, esta revelou-se uma boa escolha, permitindo explorar a influência de diferentes técnicas e parâmetros. Através de múltiplos ensaios, nos quais se testaram diferentes configurações do modelo, evoluiu-se na qualidade das amostras, obtendo-se resultados promissores. Tratando-se de uma temática sempre em evolução, vão surgindo novas técnicas que podem acrescentar melhorias ao modelo aplicado. Além disso, com um aumento dos recursos computacionais disponíveis, seria possível explorar todas as possibilidades existentes (maior *dataset*, rede maior/mais complexa, etc.) para atingir um melhor resultado.

Deste modo, conclui-se que é possível gerar dados artificiais capazes de serem utilizados no treino de sistemas que necessitam de enormes quantidades de dados e na qual a disponibilidade de dados é limitada, tal como em aplicações de monitorização no interior de SAVs.

REFERÊNCIAS

Data Science Academy. Deep learning book, 2019. URL http://deeplearningbook.com.br/. Last accessed November 2019.

Vinod Nair Alex Krizhevsky and Geoffrey Hinton. The cifar-10 dataset. URL https://www.cs.toronto.edu/~kriz/cifar.html. Last accessed August 2020.

Shane Barratt and Rishi Sharma. A Note on the Inception Score. 2018. URL http://arxiv.org/abs/1801.01973.

Ali Borji. Pros and cons of GAN evaluation measures. *Computer Vision and Image Understanding*, 179:41–65, 2019. ISSN 1090235X. doi: 10.1016/j.cviu.2018.10.009.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GaN training for high fidelity natural image synthesis. *7th International Conference on Learning Representations, ICLR 2019*, pages 1–35, 2019.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in Neural Information Processing Systems*, pages 2180–2188, 2016. ISSN 10495258.

Mateus Coelho. Inteligência artificial e deep learning, 2019. URL http://www.decom.ufop.br/imobilis/inteligencia-artificial-e-deep-learning/. Last accessed November 2019.

Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Deep learning for visual unDerstanDing: part 2 Generative Adversarial Networks. *IEEE SIgnal ProcESSIng MagazInE*, 35(January):53–65, 2018. doi: 10.1109/MSP.2017.2765202. URL http://arxiv.org/abs/1710.07035{%}0Ahttp://dx.doi.org/10.1109/MSP.2017.2765202.

Emily Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. *Advances in Neural Information Processing Systems*, 2015-Janua: 1486–1494, 2015. ISSN 10495258.

Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial Feature Learning. (2016):1–18, 2016. URL http://arxiv.org/abs/1605.09782.

S. M. Ali Eslami, Nicolas Heess, Christopher K. I. Williams, and John Winn. The shape boltzmann machine: A strong model of object shape. *International Journal of Computer Vision*, 107(2):155–176, Apr 2014. ISSN 1573-1405. doi: 10.1007/s11263-013-0669-1. URL https://doi.org/10.1007/s11263-013-0669-1.

Ian Goodfellow. NIPS 2016 Tutorial: Generative Adversarial Networks. 2016. URL http://arxiv.org/abs/1701.00160.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. urlhttpwww.deeplearningbook.org.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. *Advances in Neural Information Processing Systems*, 2017-Decem(Nips):6627–6638, 2017. ISSN 10495258.

Xun Huang, Ming-yu Liu, Serge Belongie, and Jan Kautz. Multimodal Unsupervised Image-to-Image Translation.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *32nd International Conference on Machine Learning, ICML 2015*, 1:448–456, 2015.

Animesh Karnewar and Oliver Wang. MSG-GAN: Multi-Scale Gradients for Generative Adversarial Networks. pages 7796–7805, 2020. doi: 10.1109/cvpr42600.2020.00782.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, pages 1–26, 2018.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:4396–4405, 2019. ISSN 10636919. doi: 10.1109/CVPR.2019.00453.

Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. (Ml):1–14, 2013. URL http://arxiv.org/abs/1312.6114.

Rico Krueger, Taha H. Rashidi, and John M. Rose. Preferences for shared autonomous vehicles. *Transportation Research Part C: Emerging Technologies*, 2016. ISSN 0968090X. doi: 10.1016/j.trc.2016.06.015.

Ming Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-shot unsupervised image-to-image translation. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-Octob:10550–10559, 2019. ISSN 15505499. doi: 10.1109/ICCV.2019.01065.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial Autoencoders. 2015. URL http://arxiv.org/abs/1511.05644.

Hao Mingyang. Shared Autonomous Vehicles : Preferences , Opportunities , and Future Implications. (September), 2019.

Mehdi Mirza and Simon Osindero. Conditional Generative Adversarial Nets. pages 1–7, 2014. URL http://arxiv.org/abs/1411.1784.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, Yuichi Yoshida, and Preferred Networks. S Pectral N Ormalization. 2018. ISSN 17445000. doi: 10.1080/09208119108944286. URL https://arxiv.org/pdf/1802.05957.pdf.

Vinod Nair and Geoffrey Hinton. Rectified linear units improve restricted boltzmann machines vinod nair. volume 27, pages 807–814, 06 2010.

Raghuram Nandepu. Understanding and implementation of residual networks(resnets), 2019. URL https://medium.com/analytics-vidhya/understanding-and-implementation-of-residual-networks-resnets-b80f9a507b9c. Last accessed September 2020.

Richard E. Neapolitan and Richard E. Neapolitan. *Neural Networks and Deep Learning*. 2018. doi: 10.1201/b22400-15.

Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. *34th International Conference on Machine Learning, ICML 2017*, 6:4043–4055, 2017.

Sukanchan Palit. A fast learning algorithm for deep belief nets. *International Journal of Environmental Science and Development*, 1554:341–346, 2010. ISSN 20100264. doi: 10.7763/ijesd.2010.v1.67.

Zhaoqing Pan, Weijie Yu, Xiaokai Yi, Asifullah Khan, Feng Yuan, and Yuhui Zheng. Recent Progress on Generative Adversarial Networks (GANs): A Survey. *IEEE Access*, 7:36322–36333, 2019. ISSN 21693536. doi: 10.1109/ACCESS.2019.2905015.

Taesung Park, Ming Yu Liu, Ting Chun Wang, and Jun Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:2332–2341, 2019. ISSN 10636919. doi: 10.1109/CVPR.2019.00244.

PIKKART. Computer vision and deep learning. URL http://www.pikkart.com/contenuto/contenuti--ecm/1computer-vision--deep-learning.ashx. Last accessed November 2019.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. pages 1–16, 2015. URL http://arxiv.org/abs/1511.06434.

Prabhu Raghav. Understanding of convolutional neural network (cnn) — deep learning, 2018. URL https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148. Last accessed November 2019.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *31st International Conference on Machine Learning, ICML 2014*, 4:3057–3070, 2014.

Ruslan Salakhutdinov and Geoffrey Hinton. Deep Boltzmann machines. *Journal of Machine Learning Research*, 5(3):448–455, 2009. ISSN 15324435.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016. ISSN 10495258.

Sungho Suh, Haebom Lee, Jun Jo, Paul Lukowicz, and Yong Oh Lee. Generative oversampling method for imbalanced data on bearing fault detection and diagnosis. *Applied Sciences (Switzerland)*, 9(4), 2019. ISSN 20763417. doi: 10.3390/app9040746.

Multi-domain Image-to-image Translation. StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. It Takes (Only) Two : Adversarial Generator-Encoder Networks. pages 1250–1257, 2016.

Ashish Vaswani and Anna Huang. Self-attention for generative models, 2019. URL http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture14-transformers.pdf. Lecture from web.stanford.edu. Last accessed September 2020.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local Neural Networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 7794–7803, 2018. ISSN 10636919. doi: 10.1109/CVPR.2018.00813.

Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T. Freeman, and Joshua B. Tenenbaum. Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. *Advances in Neural Information Processing Systems*, (Nips):82–90, 2016. ISSN 10495258.

Qiantong Xu, Gao Huang, Yang Yuan, Chuan Guo, Yu Sun, Felix Wu, and Kilian Weinberger. An empirical study on evaluation metrics of generative adversarial networks. 2018. URL http://arxiv.org/abs/1806.07755.

Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *36th International Conference on Machine Learning, ICML 2019*, 2019-June:12744–12753, 2019a.

Han Zhang, Zizhao Zhang, Augustus Odena, and Honglak Lee. Consistency Regularization for Generative Adversarial Networks. pages 1–19, 2019b. URL http://arxiv.org/abs/1910.12027.

Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable Augmentation for Data-Efficient GAN Training. 2020. URL http://arxiv.org/abs/2006.10738.

ANEXOS

Amostras para cada ensaio

Consideraram-se amostras de cada ensaio para análise qualitativa. Os critérios de seleção passaram por selecionar amostras de cada *label* de interesse. Os momentos escolhidos para a seleção das amostras foram o início, meio e momento anterior a entrar em *mode collapse*.

Nos ensaios em que não se observou qualquer convergência ao longo do treino, apenas foram selecionados dois momentos para recolha de amostras, o momento inicial e o momento em que outras *labels* apresentam amostras com alguma qualidade.

Figura 55: Amostra da *label* de interesse no ensaio EV2, na iteração 2000.

Figura 56: Amostra da *label* de interesse no ensaio EV2, na iteração 20000.

Figura 57: Amostra da *label* de interesse no ensaio EV2, na iteração 38000.

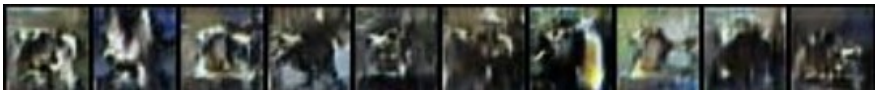


Figura 58: Amostra da *label* de interesse no ensaio EV3, na iteração 2000.

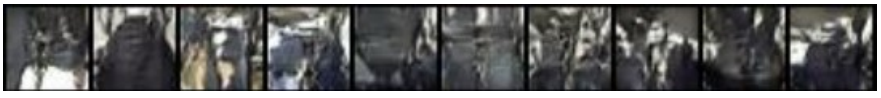


Figura 59: Amostra da *label* de interesse no ensaio EV3, na iteração 12000.

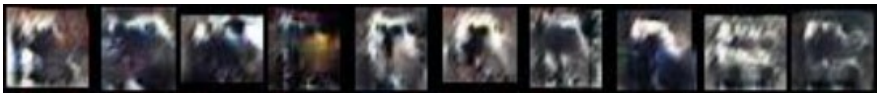


Figura 60: Amostra da *label* de interesse no ensaio EV4, na iteração 2000.

Figura 61: Amostra da *label* de interesse no ensaio EV4, na iteração 20000.

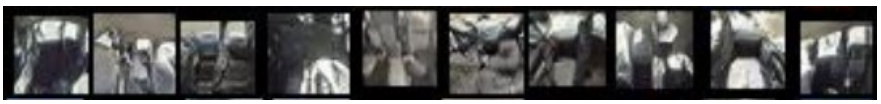


Figura 62: Amostra da *label* de interesse no ensaio EV4, na iteração 38000.

Figura 63: Amostra da *label* de interesse no ensaio EV5, na iteração 2000.

Figura 64: Amostra da *label* de interesse no ensaio EV5, na iteração 20000.

Figura 65: Amostra da *label* de interesse no ensaio EV5, na iteração 34000.

Figura 66: Amostra da *label* de interesse no ensaio EV6, na iteração 1250.

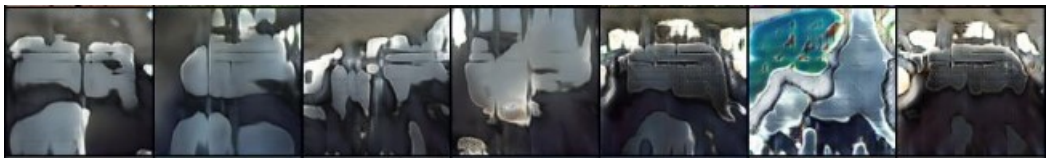


Figura 67: Amostra da *label* de interesse no ensaio EV6, na iteração 11250.

Figura 68: Amostra da *label* de interesse no ensaio EV7, na iteração 1250.

Figura 69: Amostra da *label* de interesse no ensaio EV7, na iteração 12500.

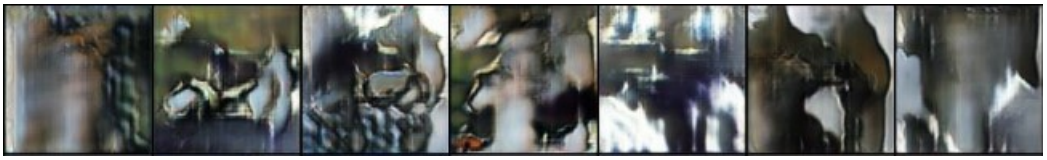


Figura 70: Amostra da *label* de interesse no ensaio EV8, na iteração 1250.

Figura 71: Amostra da *label* de interesse no ensaio EV8, na iteração 15000.

Figura 72: Amostra da *label* de interesse no ensaio EV9, na iteração 1250.

Figura 73: Amostra da *label* de interesse no ensaio EV9, na iteração 15000.

Figura 74: Amostra da *label* de interesse no ensaio EV10, na iteração 1250.

Figura 75: Amostra da *label* de interesse no ensaio EV10, na iteração 11250.

Figura 76: Amostra da *label* de interesse no ensaio EV11, na iteração 2500.

Figura 77: Amostra da *label* de interesse no ensaio EV11, na iteração 12500.

Figura 78: Amostra da *label* de interesse no ensaio EV12, na iteração 2500.

Figura 79: Amostra da *label* de interesse no ensaio EV12, na iteração 12500.

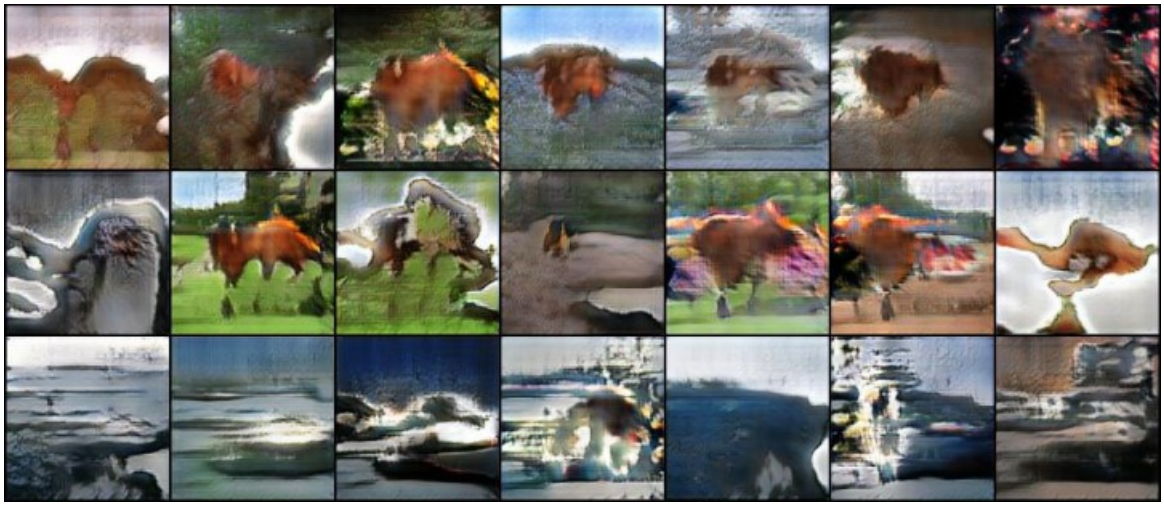


Figura 80: Amostras de outras *labels* no ensaio EV13, na iteração 2500.

Figura 81: Amostras de outras *labels* no ensaio EV13, na iteração 30000.

Figura 82: Amostras de outras *labels* no ensaio EV13, na iteração 60000.

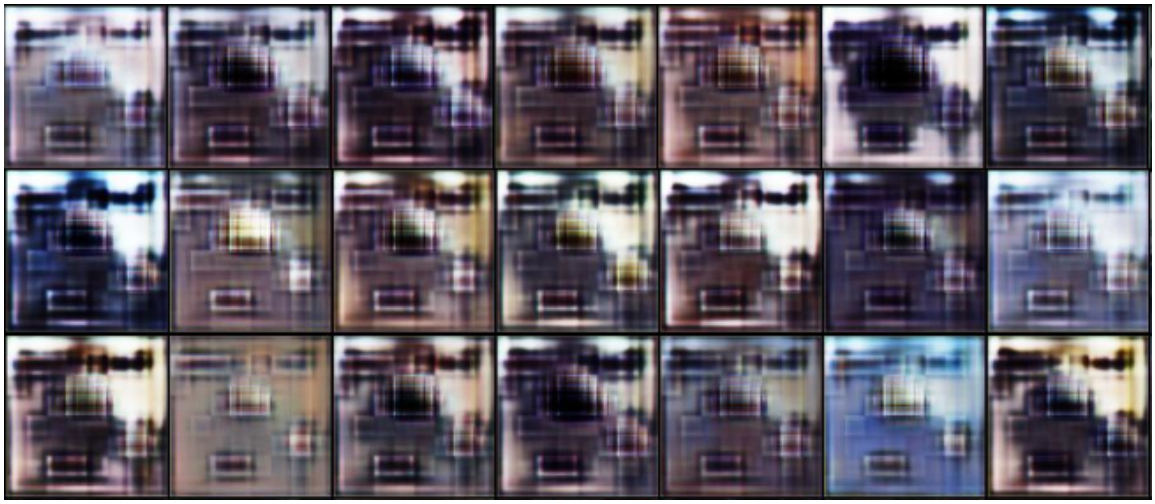


Figura 83: Amostra da *label* de interesse no ensaio EV14, na iteração 250.

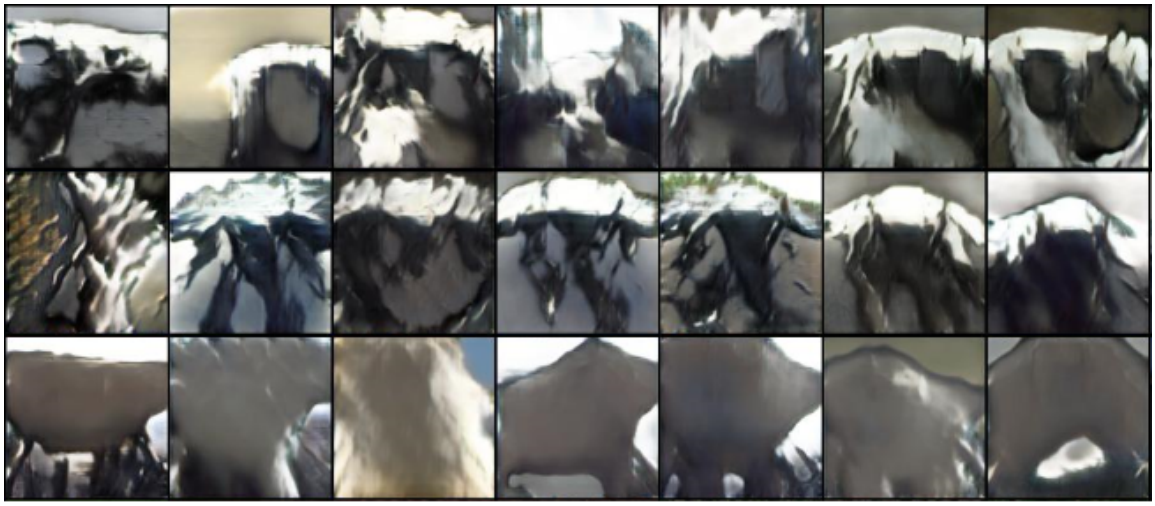


Figura 84: Amostra da *label* de interesse no ensaio EV14, na iteração 12500.

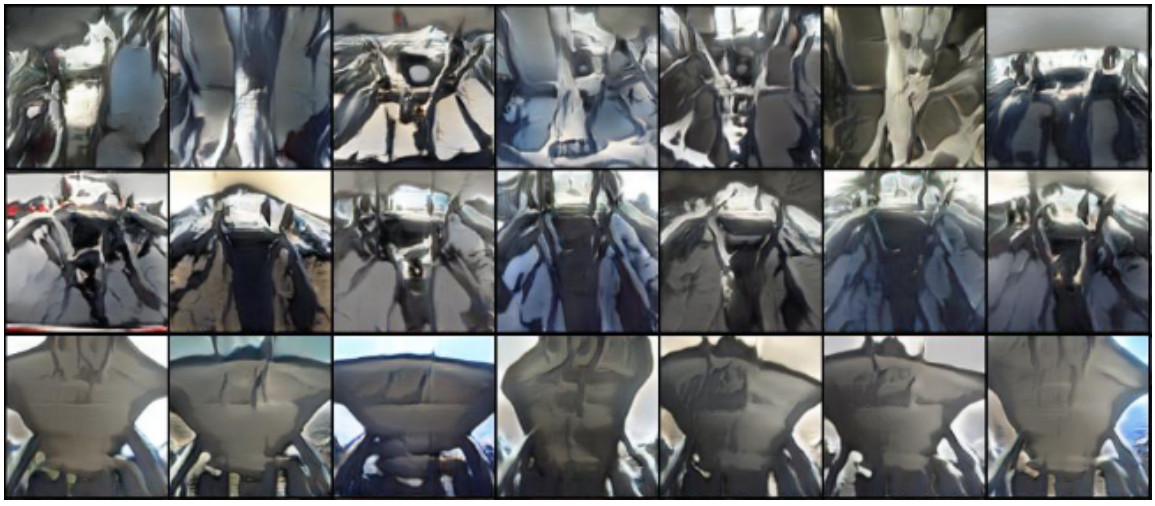


Figura 85: Amostra da *label* de interesse no ensaio EV14, na iteração 21500.

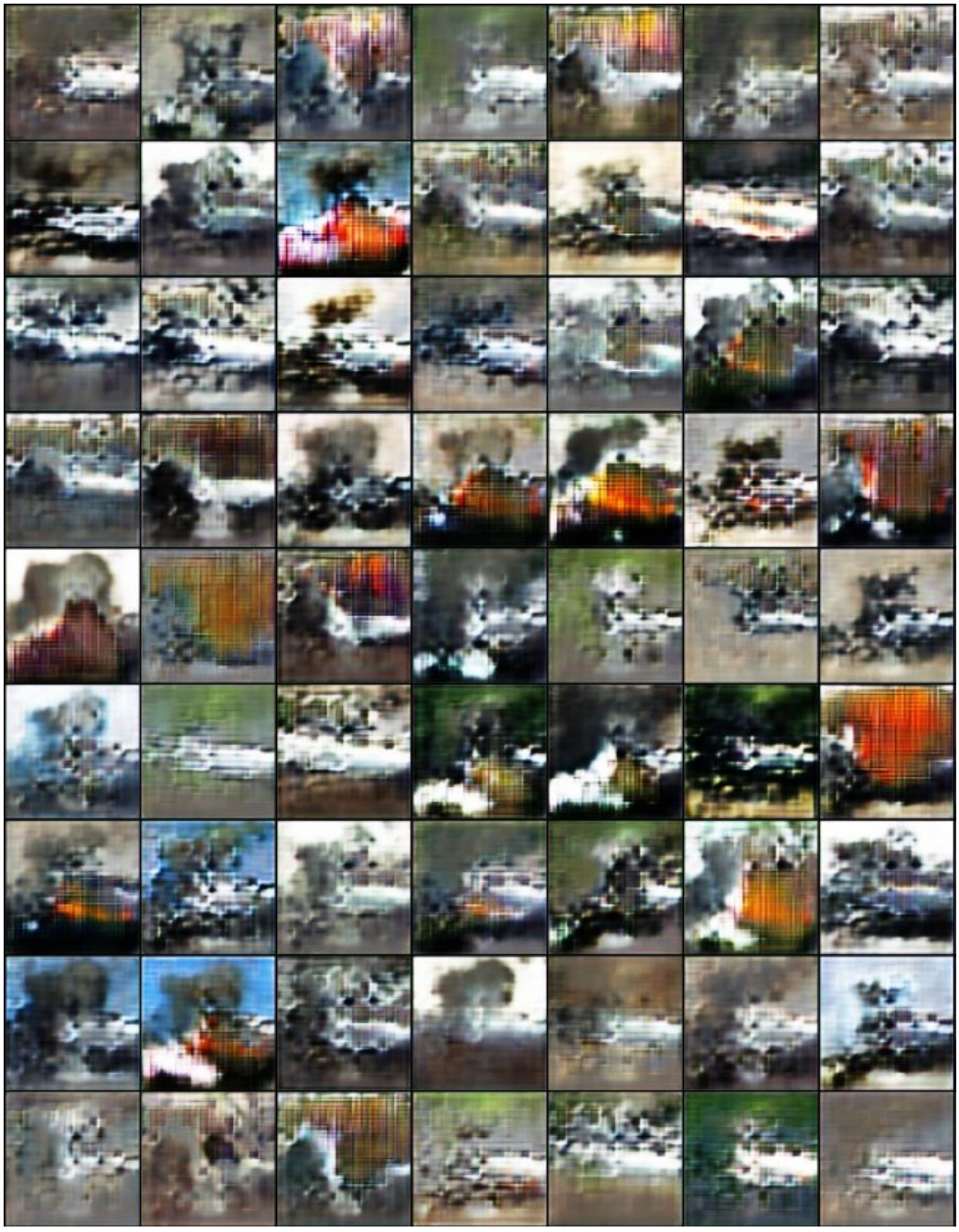


Figura 86: Amostra da *label* de interesse no ensaio EV15, na iteração 500.

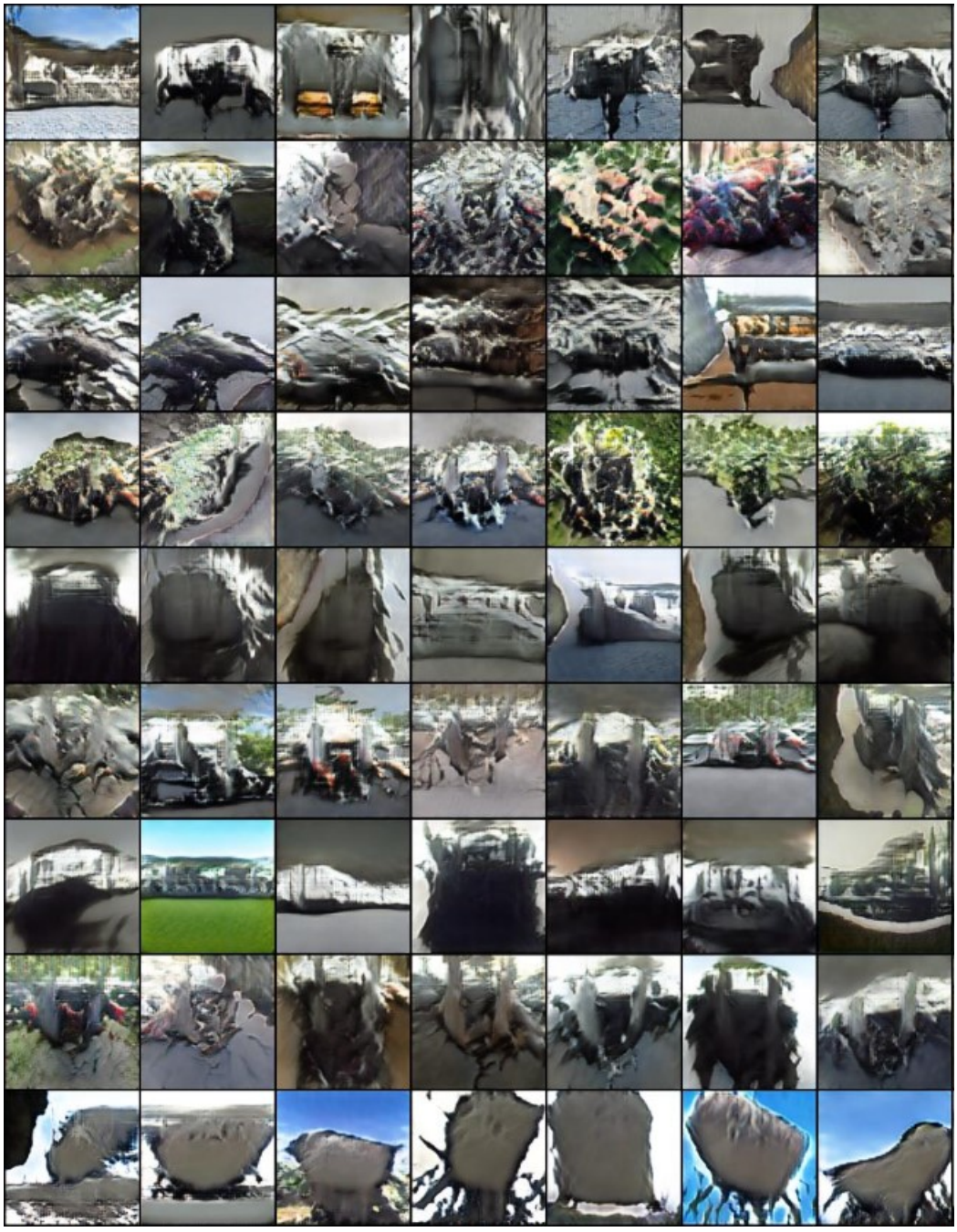


Figura 87: Amostra da *label* de interesse no ensaio EV15, na iteração 12500.

Figura 88: Amostra da *label* de interesse no ensaio EV15, na iteração 24000.

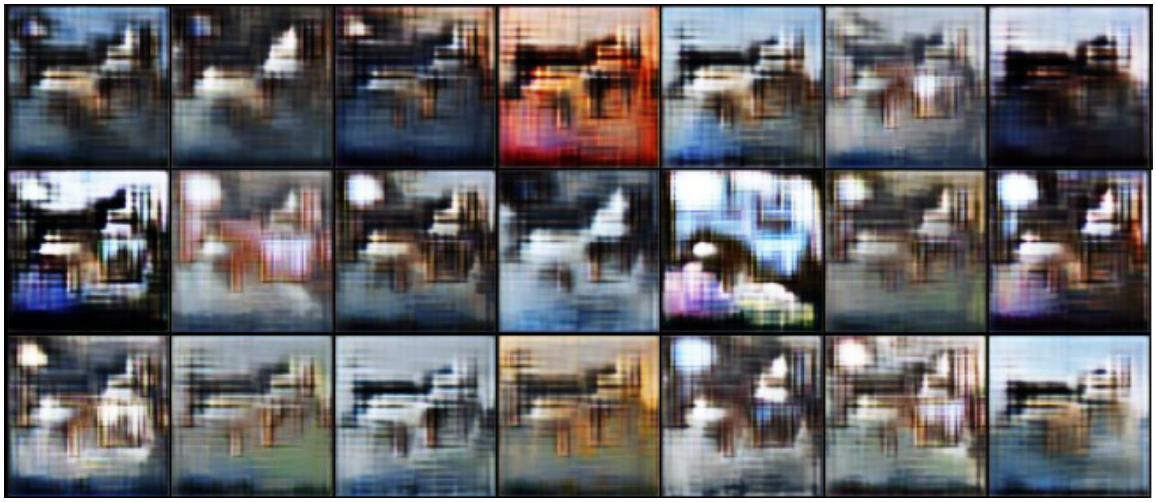


Figura 89: Amostra da *label* de interesse no ensaio EV16, na iteração 500.

Figura 90: Amostra da *label* de interesse no ensaio EV16, na iteração 26000.

Figura 91: Amostra da *label* de interesse no ensaio EV16, na iteração 45000.

Figura 92: Amostra da *label* de interesse no ensaio EV17, na iteração 1000.

Figura 93: Amostra da *label* de interesse no ensaio EV17, na iteração 13500.

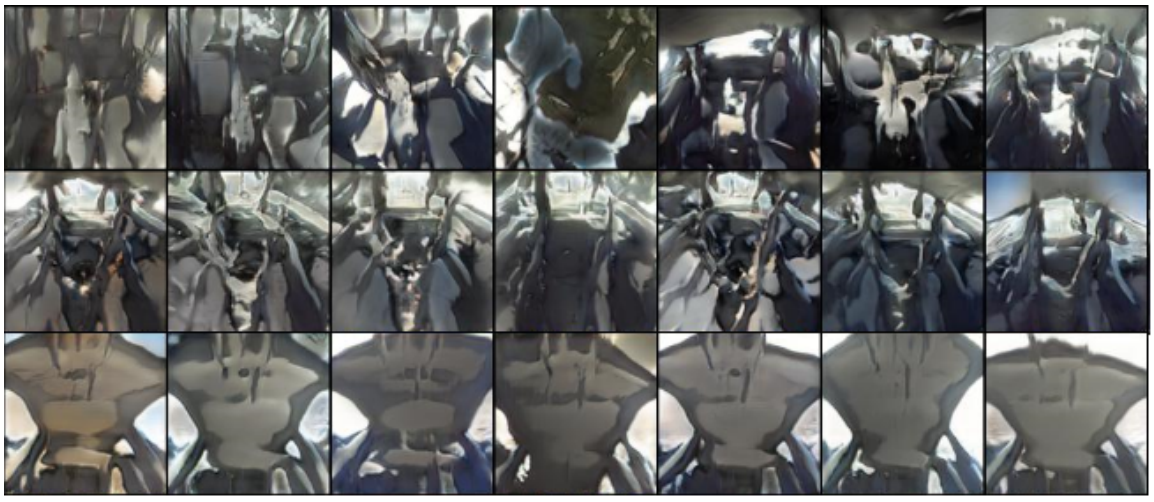


Figura 94: Amostra da *label* de interesse no ensaio EV17, na iteração 24500.

Figura 95: Amostra da *label* de interesse no ensaio EV18, na iteração 500.

Figura 96: Amostra da *label* de interesse no ensaio EV18, na iteração 12000.

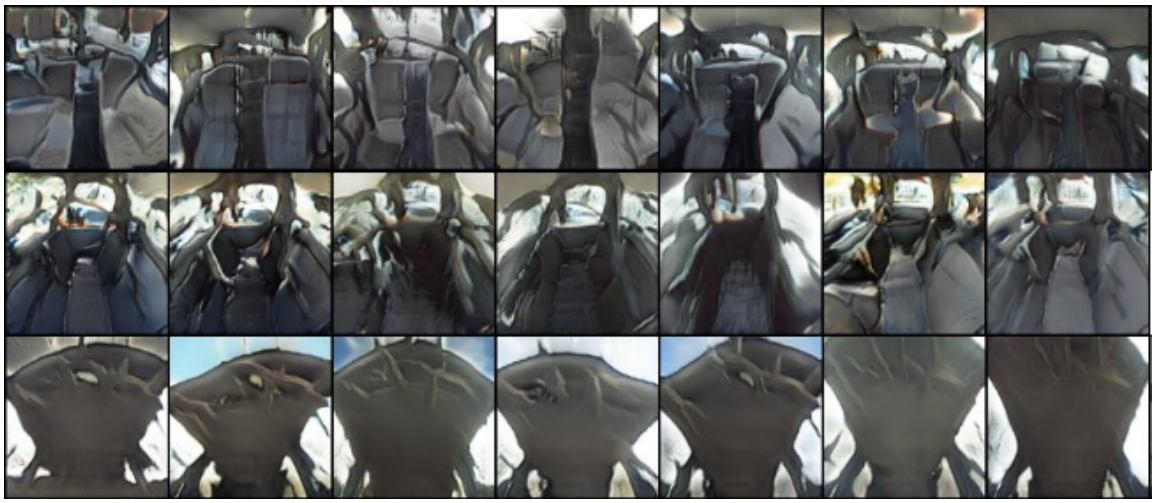


Figura 97: Amostra da *label* de interesse no ensaio EV18, na iteração 25500.

Figura 98: Amostra da *label* de interesse no ensaio EV19, na iteração 500.

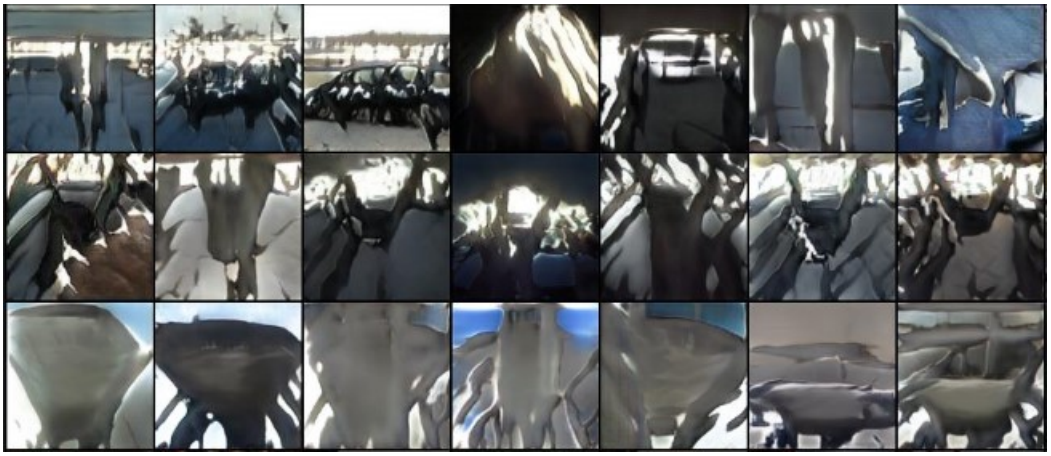


Figura 99: Amostra da *label* de interesse no ensaio EV19, na iteração 15000.

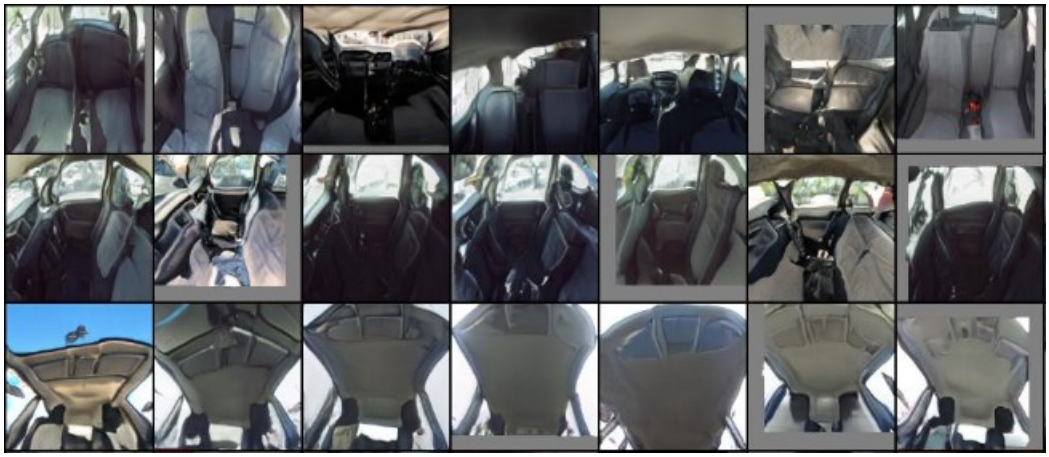


Figura 100: Amostra da *label* de interesse no ensaio EV19, na iteração 45000.

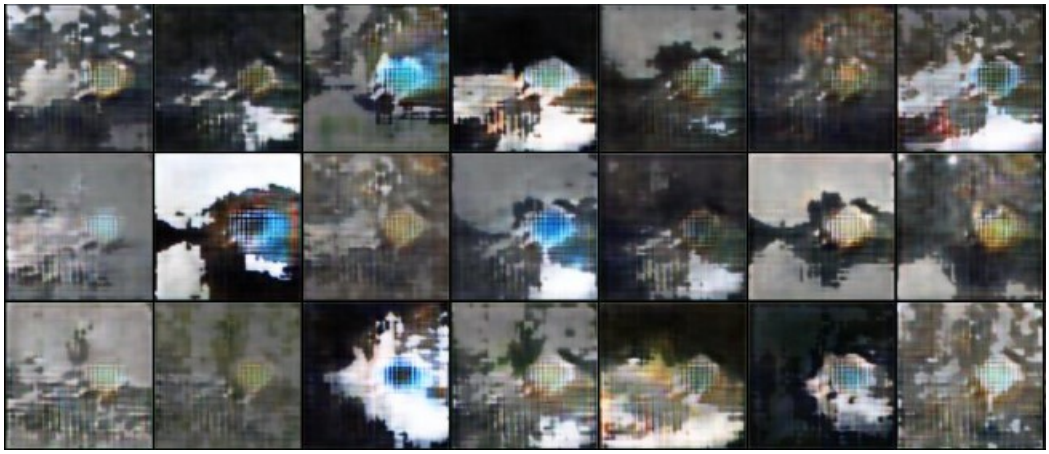


Figura 101: Amostra da *label* de interesse no ensaio EV20, na iteração 500.

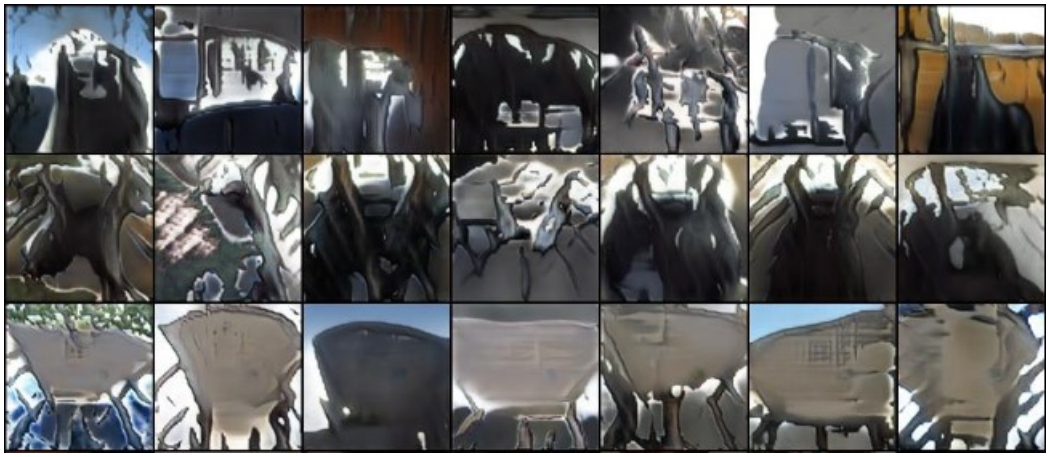


Figura 102: Amostra da *label* de interesse no ensaio EV20, na iteração 15000.

Figura 103: Amostra da *label* de interesse no ensaio EV20, na iteração 30000.

Figura 104: Amostra da *label* de interesse no ensaio EV21, na iteração 500.

Figura 105: Amostra da *label* de interesse no ensaio EV21, na iteração 10000.

Figura 106: Amostra da *label* de interesse no ensaio EV21, na iteração 20000.

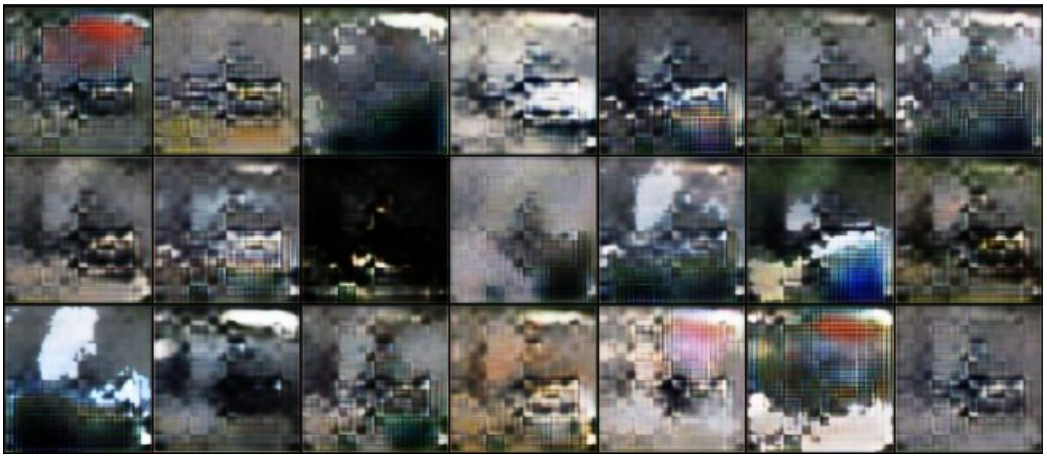


Figura 107: Amostra da *label* de interesse no ensaio EV22, na iteração 500.

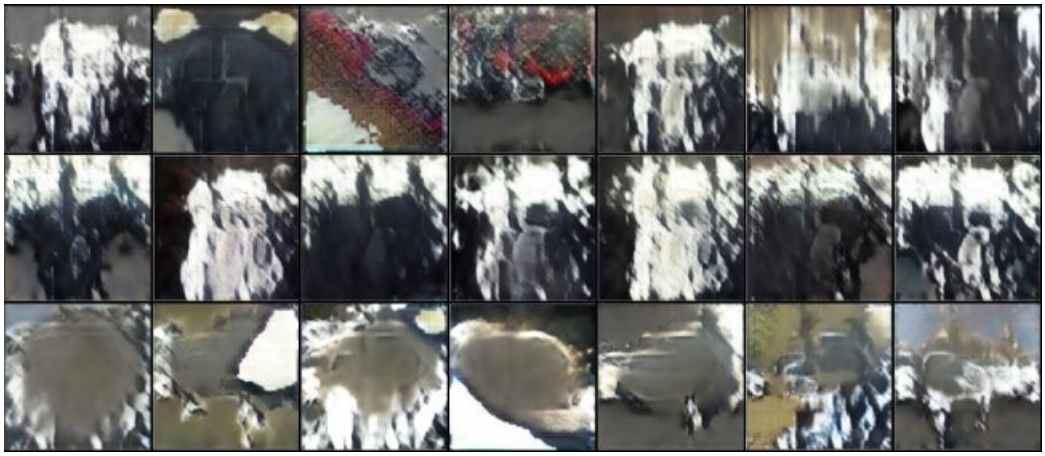


Figura 108: Amostra da *label* de interesse no ensaio EV22, na iteração 10000.

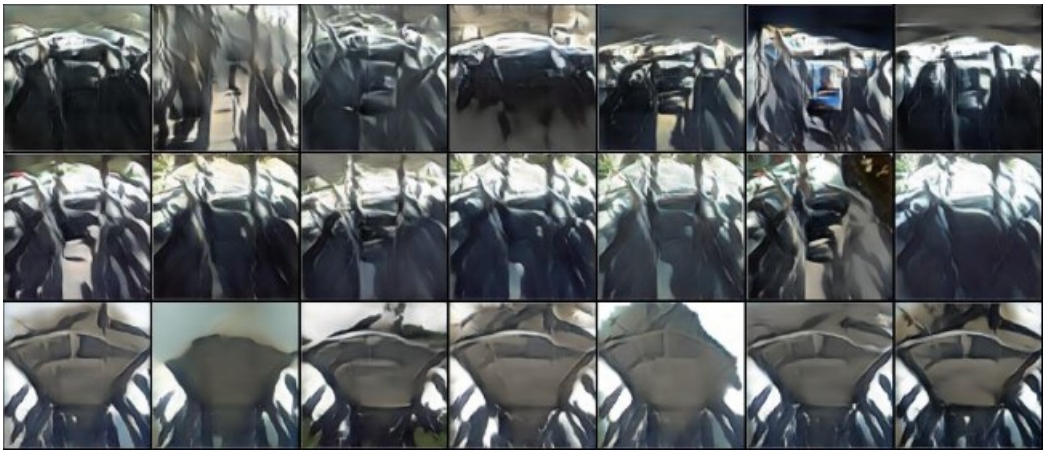


Figura 109: Amostra da *label* de interesse no ensaio EV22, na iteração 20000.