Melhorar o Desempenho de Um Classificador de Imagem Utilizando Modelos Generativos: Um Estudo Experimental

Alice Mangara[1 [2020242411]] and Diana Mortágua[1 [2020242508]]

Universidade de Coimbra | FCTUC | DEI

Abstract. Este trabalho investiga o uso de modelos generativos para melhorar o desempenho de um classificadores de imagem. Em particular, exploramos o uso de autoencoders e redes adversariais generativas (GANs) para gerar dados adicionais que possam enriquecer o conjunto de treinamento de um classificador de imagens pré-definido. Avaliamos o impacto desses dados gerados no desempenho do classificador e comparamos os resultados com o modelo base.

1 Introdução

Modelos generativos permitem-nos aprender a distribuição subjacente de um conjunto de dados, permitindo gerar novos dados que seguem a mesma distribuição. Dois exemplos de modelo generativos são ***Autoencoders [AE]*** e ***Generative Adversarial Networks [GAN]***.

Os Autoencoders são uma classe de redes neurais que aprendem uma representação compacta dos dados de entrada (**espaço latente**), através de um processo de **codificação e decodificação**. Eles consistem em duas partes principais:

- **codificador** - mapeia os dados de entrada para o espaço latente
- **descodificador** -reconstrói os dados originais a partir do espaço latente

Os mesmos podem ser treinados de forma não supervisionada, minimizando a diferença entre a entrada e a saída reconstruída.

As GANs são constituídas por duas networks – **gerador e discriminador**. O discriminador recebe imagens e deve classificá-las como verdadeiras (originais) ou falsas (criadas pelo gerador). O seu objetivo é conseguir distinguir completamente estes dois grupos de imagens, não sendo enganado pelo gerador. O gerador gera imagens novas, aprendendo a distribuição das imagens reais originais, com o objetivo de enganar o discriminador, criando imagens tão parecidas com as reais que o discriminador não tenha a certeza qual a origem das imagens que lhe são dadas. Ambas estas networks melhoram simultaneamente pelo seu caráter adversário.

2 Problema

No nosso problema original, temos um classificador de imagens (Convolutional Neural Network – CNN) pré-determinada e um conjunto de imagens de treino reduzido e desequilibrado. Temos como objetivo melhorar a performance desta CNN ao realizar data augmentation com um AE e uma GAN nos dados fornecidos pelos professores, baseado no dataset "Traffic Sign Dataset", do qual usamos 10 classes. Devemos preparar o pipeline de Machine Learning [ML] das imagens originais, analisar a CNN a utilizar, preparar o pipeline de ML para treinar a CNN com os dados originais e com os dados criados pelos modelos generativos, explorar soluções de realização de data augmentation através dos dois métodos e comparar os resultados da CNN com e sem essas imagens criada.

3 Abordagem

Para analisar os nossos resultados, a nível de sucesso no nosso objetivo de criar imagens para auxiliar e melhorar o desemprenho da CNN iremos usar *precision, recall e f1-score* por cada classe e *macro average*, weighted average e accuracy para os dados de validação e de teste completos. Consideramos o uso das métricas escolhidas para cada classe uma vez que alguns sinais tem mais importância o seu recall ser alto e noutros é mais importante a precision ser alta (nomeadamente, o sinal de ser obrigatório virar à direita deve ser garantidamente seguido, pelo que deve ser recall alto, enquanto que no limite de velocidade ser 70 é menos importante ter o recall alto, podendo nós analisar o seu sucesso através da precision). Para analisar o quão bem as imagens geradas e os modelos generativos seguem a distribuição das imagens originais de treino usaremos ***Frechet Inception Distance [FID]***e ***Structural Similarity Index [SSIM]***.

3.1 AE

A abordagem inicial envolveu a procura do melhor modelo, para tal, implementámos diversas arquiteturas de autoencoders, tanto convencionais como com a adição de ruído gaussiano para a tarefa de denoising, com diferentes profundidades e configurações de camadas. O treino foi dividido em treino, validação e teste e testámos **funções de perda:** *nn.MSELoss e nn.L1Loss* e **otimizadores:** *optim.Adam e optim.SGD.* O melhor modelo foi selecionado com base na menor **perda de validação**(fig.1):

- **Best Autoencoder** : *DenoisingAutoencoder2* - Denoising e com mais camadas.
- **Best Criterion** : *MSELoss*
- **Best Optimizer** : *SGD*

Treinámos o modelo,aumentámos o número de neurónios e guardámos as melhores imagens reconstruídas, mas as mesmas continham muito ruído então optámos por realizar um novo experimento.

No segundo experimento realizado, após intensa pesquisa decidimos implementar um ***Variational Autoencoder (VAE)*** com uma arquitetura específica composta por um codificador e um decodificador, baseado nos modelos da biblioteca keras. No codificador, a entrada passa por uma série de camadas convolucionais, seguidas por camadas totalmente conectadas, que mapeiam as imagens para uma distribuição latente, capturando suas características essenciais. Cada camada convolucional é seguida por uma camada de normalização por lotes *(Batch Normalization)* para estabilizar o treinamento e uma função de ativação ReLU para introduzir não linearidades. A reparametrização é então aplicada, gerando amostras aleatórias na distribuição latente. O decodificador, por sua vez, recebe as amostras na distribuição latente e as classes correspondentes, reconstruindo as imagens a partir dessas informações. Isso é feito através de camadas totalmente conectadas e camadas desconvolucionais, que transformam a representação latente de volta para uma forma dimensional espacial, produzindo a reconstrução final da imagem. Durante o treino (800 *epochs*), otimizámos uma função de perda que combina o erro de reconstrução *(Binary Cross-Entropy)* e a divergência *KL (Kullback-Leibler)* entre as distribuições latentes e uma distribuição normal padrão. O modelo é treinado em lotes de imagens, onde cada imagem é associada à sua classe correspondente por meio de codificação *one-hot*. Treinamos e salvamos o modelo individualmente para garantir os melhores modelos para cada classe.Experimentámos (lr = 1e-3 e lr = 1e-4) para observar se haviam mudanças na qualidade das imagens geradas.

3.2 GAN

No caso da GAN procuramos testar diferente valores de epochs e perceber que parâmetros funcionam melhor ao ver como a GAN reage após cada tentativa e analisando trabalhos semelhantes [1] e recomendações da literatura [2,3]. Esperamos que um maior número de epochs dê resultados melhores, obviamente, e procuraremos analisar o efeito de diferentes parâmetros em diferentes métricas. A nível da arquitetura do gerador e do discriminador, provocamos algumas alterações no gerador ao longo do tempo, nomeadamente no número máximo de neurónios e na quantidade de layers, de modo a obter uma network que originasse resultados mais próximos dos esperados (gostaríamos de ainda ter aumentado mais o número máximo de neurónios mas o custo computacional levava a uma duração de treino extremamente longa, pelo que nos focamos mais em alterações de parâmetros fora da network).

4 Setup experimental

Os nossos dados foram divididos em 80porcento treino, 10porcento validação e 10porcento teste.

4.1 AE

Na primeira abordagem/primeiro experimento, cada classe foi treinada individualmente, com um número fixo de épocas definido para cada iteração (600 epochs para classes com muitas samples e 800 para as com menos samples). Após cada epoch, o modelo foi avaliado no conjunto de validação, e se uma diminuição na loss function fosse observada, os pesos do modelo eram salvos. Além disso, as últimas 80 imagens reconstruídas foram armazenadas para análise posterior, pois eram as que apresentavam menor ruído e tinham potencial para melhorar o modelo. De seguida balanceou-se o dataset, verificou-se os resultados e calculou-se o SSIM e o FID.

Contudo, achámos que seria melhor experimentar uma abordagem que fosse mais recomendada pela literatura e que pudesse gerar imagens com maior qualidade.

Na segunda abordagem/ segundo experimento, para aumentar a quantidade de dados disponíveis no dataset de sinais de trânsito, utilizamos um modelo ***Variational Autoencoder (VAE)*** para gerar imagens sintéticas. Primeiramente, configuramos as diretorias para armazenar dados originais, imagens geradas e dados balanceados, organizados conforme as classes do dataset. O modelo VAE foi carregado e colocado em modo de avaliação. Para cada classe, geramos 150 novas imagens e calculamos a similaridade estrutural ***(SSIM)*** entre as imagens geradas e as imagens reais da mesma classe, como altera o número de epochs não variava de forma significativa a qualidade das imagens, usámos 800 epochs para todas as classes. Selecionámos e salvámos as 80 imagens geradas com maior SSIM para cada classe. Depois, para classes com menos de 80 imagens, adicionámos imagens geradas até que cada classe tivesse um número uniforme de imagens. Todas as imagens foram copiadas para o diretório balanceado, garantindo que cada classe tivesse até 80 imagens. Finalmente, visualizámos algumas das melhores imagens geradas para cada classe, assegurando a qualidade e relevância das imagens sintéticas.

4.2 GAN

Para a ***GAN***, fizemos 3 experimentos na geração de imagens (inicialmente tínhamos treinado apenas o modelo e guardado as últimas 50 imagens mas, devido ao treino de ***GANs*** não convergir, abandonámos essa estratégia, passando a guardar imagens com os valores de perda mais satisfatórios (nomeadamente os loss values do gerador e do discriminador durante o treino) após o epoch 100 (uma vez que inicialmente, só com a primeira restrição seriam guardadas muitas imagens com pouco mais do que ruído). Inicialmente usamos 500 *epochs* para todas as classes, na segunda tentativa usámos 800 e por fim usamos 1800 para classes com um número relativamente grande de imagens de treino, 2500 para aquelas com menos de 15 e 2000 para as restantes (valores originários de opinião subjetiva após testes isolados em diferentes classes), uma vez que classes com mais imagens de treino precisavam de menos epochs para resultados visualmente semelhantes aos obtidos nas classes com menos imagens em que se usaram mais

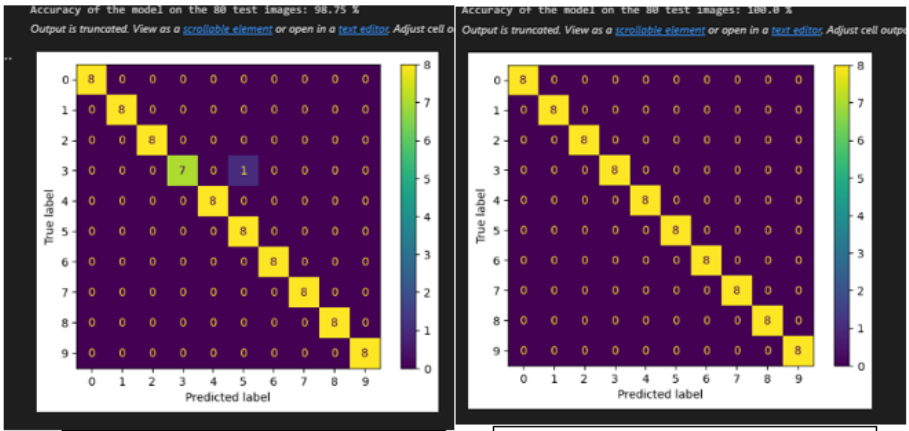
epochs. Na primeira ronda de treino utilizamos o optimizer do discriminador e do gerador com *learning rate* 0.0002 e *betas* (0.5,0.999). Na segunda ronda, adicionamos um *scheduler* no *learning rate* dos optimizers com *step size* 30 e *gamma* 0.95. Por fim, na última tentativa adicionámos *weight decay* de 0.00005 (em experiências pontuais experimentámos valores mais altos, mas considerámos um valor mais pequeno como este mais vantajoso) e alterámos o *gamma* do *scheduler* do l*earning rate* para 0.99 em vez de 0.95, de modo a levar o *learning rate* a diminuir de forma menos drástica após cada step. Na 1ª tentativa o gerador tinha 5 layers, sendo o número máximo de neurónios 256 do modo representado abaixo, enquanto que, na segunda e terceira tentativa usamos a estrutura no ficheiro de código entregue.

Cada classe foi treinada isoladamente, sendo as imagens obtidas do modo descrito acima guardadas. Após o treino de todas as classes, estas imagens eram colocadas nas respetivas pastas junto com os dados de treino originais, seguindo-se o treino da ***CNN*** para avaliar o progresso realizado, e calculado o ***FID*** e o ***SSIM***.

5 Análise dos Resultados

5.1 VAE

Para ambas as abordagens denotámos uma melhoria da accuracy do modelo pré-definido, para 98,75 e 100 respetivamente. (figura1)



Abordagem1 Abordagem2

Fig. 1: Resultados da accuracy do VAE

Podemos ver os valores das métricas de avaliação (*precision* e *recall*) , onde os valores representam médias de 3 treinos na CNN para evitar valores outliers resultantes da divisão treino-validação-teste. (figura2)

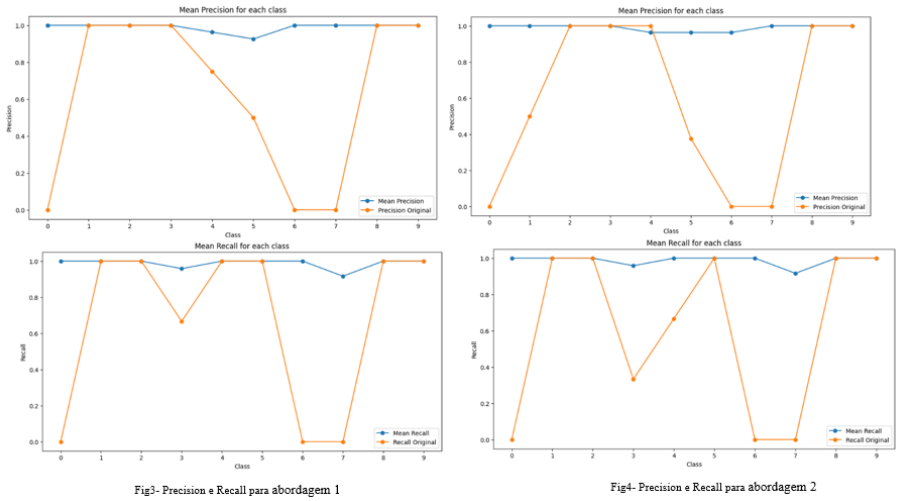


Fig3- Precision e Recall para abordagem 1

Fig4- Precision e Recall para abordagem 2

Fig. 2: *Precisão e Recall* por classe relativos ao VAE

Ao analisar estes valores, podemos afirmar que tanto para a primeira e segunda abordagem os valores de precision e recall melhoram, sendo mais eficazes para quase todas as classes do que o modelo original. Individualmente podemos ver que a VAE na primeira abordagem conseguiu aprender bem todas as classes, porém na segunda abordagem não conseguiu acertar tão bem na quinta classe(44). Em geral ambas as abordagens conseguiram bons resultados,o que não era esperado pois a segunda abordagem possui imagens com melhor qualidade, selecionadas com um critério melhor,portanto uma maior discrepância de resultados era esperado. Uma possível justificação é o modelo estar a fazer *overffitting*. Destacamos a segunda abordagem, pois é a que obteve melhor accuracy. Exemplos de imagens geradas por estas abordagens encontram-se em anexo.

Podemos analisar também os valores de FID e SSIM(figura3):

Podemos observar que em relação ao FID, ambas as abordagens apresentam um comportamento semelhante, a classe 13 apresenta um valor elevado, o que indica uma qualidade de imagens pior, e também podemos verificar que as classes com mais dados originais possuem imagens com mais qualidade, como esperado. Outro aspeto surpreendente é o facto de ambas as abordagens terem valores similares, pois tal como foi referido, a métrica de seleção e a qualidade de imagens geradas são bem diferentes. Com o SSIM observamos a preservação da estrutura das imagens e analisando ambas as abordagens, a abordagem 2 preserva melhor

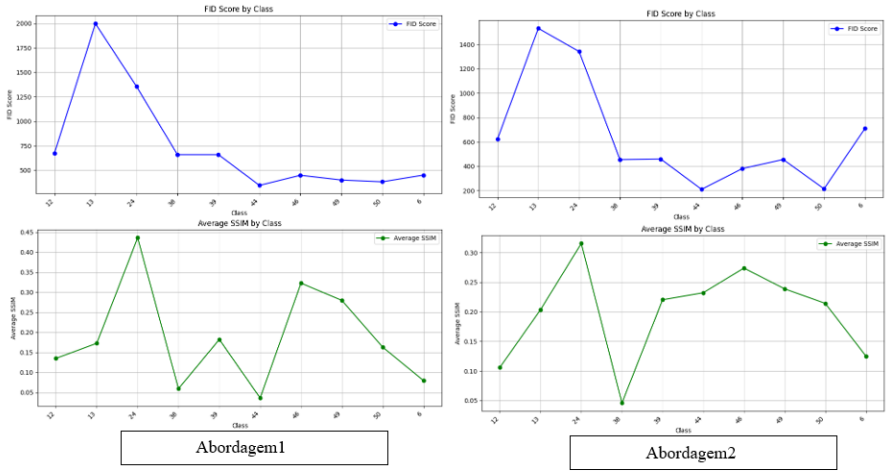


Fig. 3: *FID e SSIM* por classe relativos à VAE

a estrutura das imagens para todas as classes, mas para ambas, a classe 24 é a que o modelo preservou melhor a estrutura.

5.2 GAN

Relativamente à GAN, podemos ver os valores das métricas de avaliação nas figuras **4 e 5**. Podemos ver que, como é de esperar, os valores de todas as métricas se aproximaram mais do esperado ao longo das diferentes alterações a cada experimento. O valor da FID diminuiu a cada experimento, tal como o valor do SSIM diminuiu. A nível de *recall, accuracy e precision* também podemos ver como melhoraram, sendo estes valores médias de 5 treinos na CNN para evitar valores outliers resultantes da divisão treino-validação-teste. Podemos ver também que algumas classes foram mais difíceis para a GAN tentar aprender, continuando os seus valores a ser piores do que as outras classes quando usadas na ***CNN***. Para além disso, podemos ver que, as classes que têm melhores resultados na **CNN** são aquelas cujas criações da GAN tem distribuição mais distância às fotos originais. Um exemplo de foto de cada classe dos experimentos pode ser vista nos anexos, estando todas as fotos do experimento 3 que foram usadas juntamente com as de treino original na pasta entregue com o trabalho.

6 Conclusão

Ao analisar as métricas de *precision e recall* por classe na performance CNN, ficou evidente que as imagens adicionais produzidas pelas VAEs contribuíram para uma classificação mais precisa e abrangente dos sinais de trânsito. A variação na qualidade das imagens entre as duas abordagens ressalta a importância

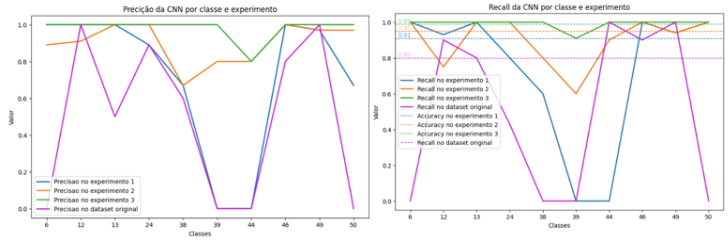


Fig. 4: *precision e recall* por classe por cada experimento da GAN

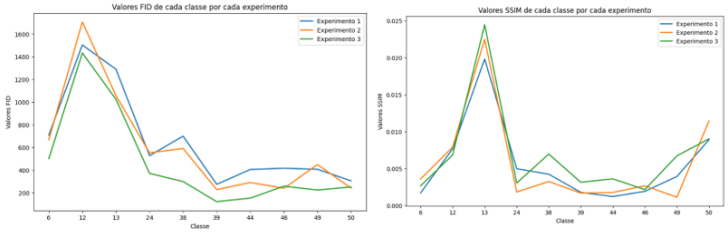


Fig. 5: Valores FID e SSIM por cada classe para cada experimento da GAN

da seleção cuidadosa de hiperparâmetros e da arquitetura do modelo. Sendo que a 2ª abordagem utiliza uma arquitetura específica de VAE, o que resulta em imagens de melhor qualidade e nitidez. Daí, a segunda abordagem ser a preferida, sendo todas as métricas relativas a ela vantajosas em relação à 1ª. Os seus resultados também apontam para o potencial significativo das VAEs como uma ferramenta viável para melhorar o desempenho dos classificadores de imagens. Relativamente à GAN, consideramos fazer sentido que as classes com melhores resultados da CNN sejam aquelas cuja distância à distribuição das imagens originais é maior (apesar de parecer contra intuitivo), uma vez que são as classes com mais dados de treino. Tendo mais imagens de base, as fotos criadas pela GAN têm mais diversidade, mas as prórpias fotos de treino têm maior diversidade entre elas. Por outro lado, classes com poucas fotos de treino não incentivam à diversidade nas fotos da GAN, o que leva a que, mesmo que as fotos não sejam tão distintas entre classes para favorecer a sua performance na CNN, tenham uma distribuição e estrutura mais semelhante às originais. Podemos ver que, efetivamente, as alterações dos parâmetros que fizemos seguindo a nossa experiência e a literatura ajudaram a melhorar as fotos criadas para serem mais reconhecíveis e distinguíveis entre classes, bem como em maior quantidade, melhorando o desempenho da CNN. Analisando os resultados, então, podemos ver que o experimento 3 têm os parametros e arquiteturas preferidas entre os três experimentos. Ambos os modelos mostraram valores semelhantes de FID e de SSIM, e melhoramos o desempenho da CNN com ambos, pelo que concluimos

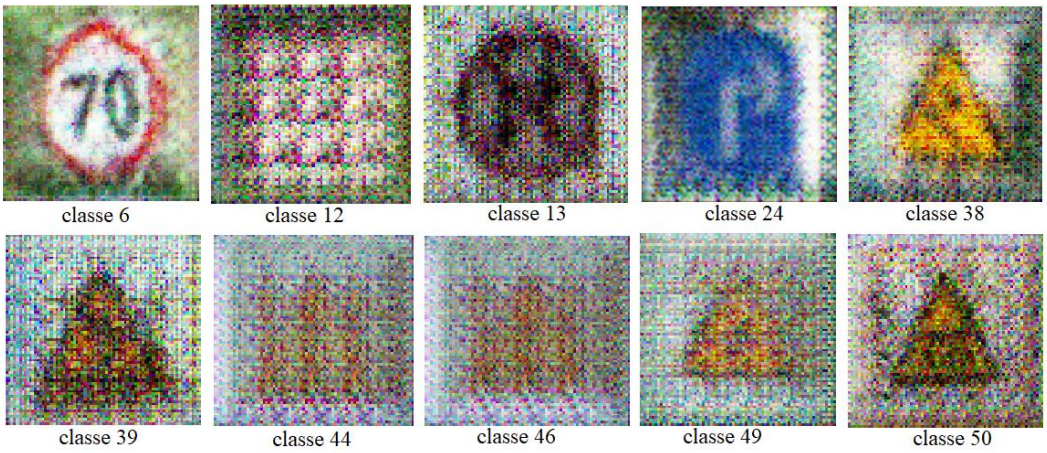
que a utilização de ambos os modelos é uma mais valia em caso de necessidade de mais imagens de treino.

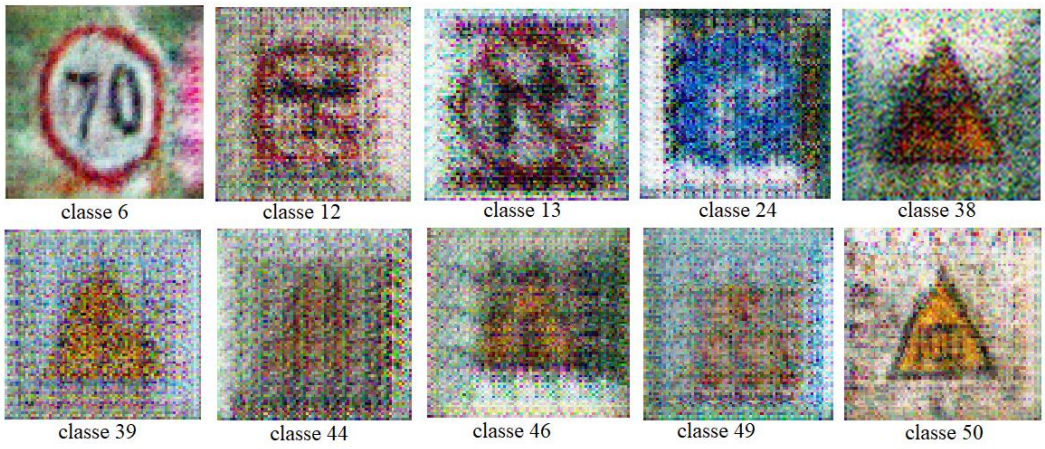
References

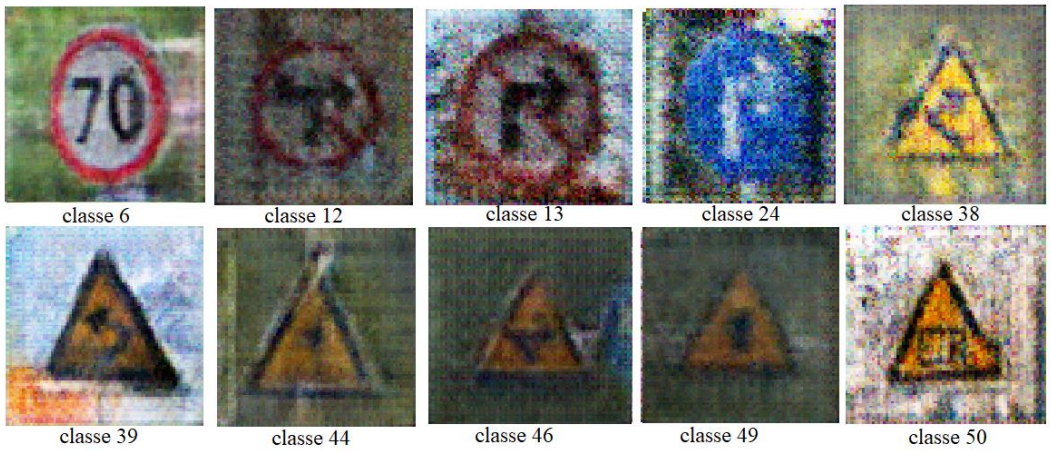
1. Dewi Christine, Chen Rung-Ching, Liu Yan-Ting, Tai Shao-Kuo - Synthetic Data Generation using DCGAN for Improved Traffic Sign Recognition (2021)
2. Bishop M. Christopher, Bishop Hugh – Deep Learning (Fundations and concepts) (2024)
3. Goodfellow Ian, Bengio Yoshua, Courville Aaron – Deep Learning , www.deeplearningbook.org
4. Chatgpt, https://openai.com/index/chatgpt/
5. Keras Model, https://blog.keras.io/building-autoencoders-in-keras.html
6. Inspiration code, https://github.com/chaitanya100100/VAE-for-Image-Generation?tab=readme-ov-file

1 Anexo

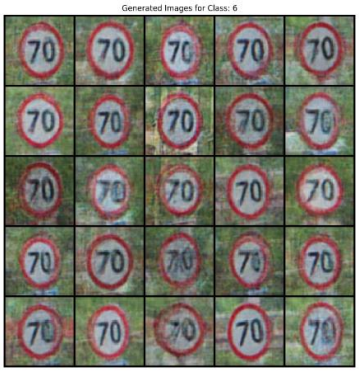
Exemplo Imagens Geradas GAN

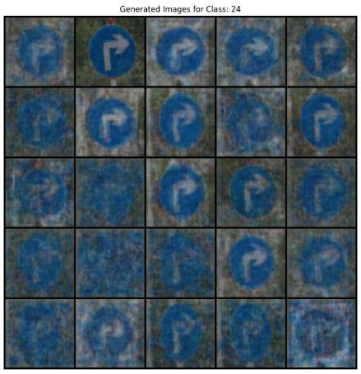


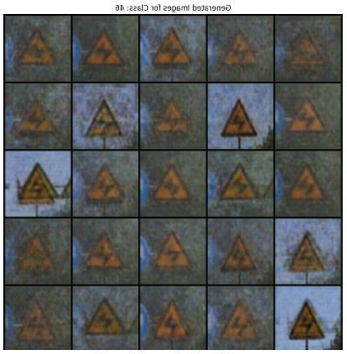




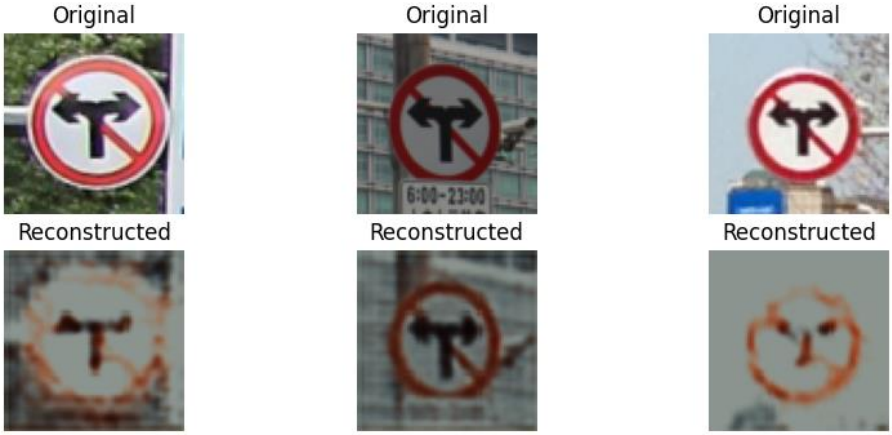
Exemplo Imagens Geradas VAE -> Abordagem2







Abordagem 1->


Original
Original
Original
6:00-23:00
Reconstructed
Reconstructed
Reconstructed


Original Original Original Original Original Original Original Original Original Original Original Original


Reconstructed