FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE
COIMBRA

Advanced Machine Learning
2023/2024 - 2º Semester

"Do you need more signs?"
TP2 Report

Leonardo Cordeiro
2020228071

Gonçalo Bastos
2020238997

Abstract. Neste trabalho, desenvolvemos um método para aumentar o desempenho de um classificador de imagens de sinais de trânsito utilizando modelos generativos. A tarefa envolveu a análise de um modelo de CNN preexistente e a aplicação de técnicas de geração de dados para ampliar o conjunto de treinamento. Utilizamos AutoEncoders e Redes Adversariais Generativas (GANs) para gerar novas amostras de imagens, integrando-as ao conjunto de dados original. A eficácia dos modelos foi avaliada utilizando métricas de validação apropriadas, como o Structural Similarity Index para o AutoEncoder e o Inception Score para a GAN, além de comparar o desempenho do classificador com e sem os dados gerados. Os resultados mostraram uma melhoria significativa na accuracy do classificador ao utilizar os dados aumentados, comprovando a eficácia dos modelos generativos no aumento da diversidade e quantidade de dados de treinamento.

1 Introdução

Atualmente, a classificação de imagens atingiu o estado da arte usando redes neuronais profundas, particularmente em tarefas como a identificação de sinais de trânsito para veículos autónomos, no entanto, a eficácia desses sistemas é limitada sobretudo pela falta de dados de treino, como uma solução para essa lacuna,usa-se modelos generativos como por exemplo, AutoEncoders e GANs para gerar novas imagens para expandir o conjunto de dados de sinais de trânsito com o objetivo de melhorar o desempenho do classificador CNN base, e mostrar que o treino de dados aumentados com exemplos artificiais pode levar a um modelo de classificação mais preciso e resiliente.

2 Descrição do Problema

O problema central deste trabalho é melhorar o desempenho de um classificador de imagens de sinais de trânsito utilizando técnicas de aumento de dados geradas por modelos generativos. O classificador de base (fornecido pelo professor) é uma CNN, onde a inicial é limitada pela quantidade de dados disponíveis para treino, é esperado para sistemas de classificação de imagens, especialmente em aplicações críticas como esta, contar com um grande volume de dados diversificados para garantir bons resultados.

O conjunto de dados de sinais de trânsito que temos é pequeno e desbalanceado, o que pode levar ao desempenho não ótimo da CNN, para abordar esse desafio, utilizamos então modelos generativos para gerar novas imagens de sinais de trânsito, essas novas imagens serão adicionadas ao conjunto de dados original com o objetivo de aumentar a quantidade e diversidade dos dados de treino, esperando assim melhorar o desempenho do classificador.

O conjunto de dados , algo idêntico ao ultimo trabalho é uma versão do "Traffic Sign Dataset" contendo 54 classes de sinais de trânsito, no entanto, para este projeto, trabalhamos com um subconjunto de 10 classes, cada uma com imagens de 75x75 pixels. Como podemos ver na Figura abaixo as classes selecionadas incluem sinais como limite de velocidade, direções proibidas e curvas perigosas, entre outros. O total de imagens no conjunto de dados é de 277, com diferentes distribuições de exemplos por classe.

Para avaliar a eficácia dos dados gerados, utilizamos métricas específicas como Inception Score, MSE, MAE, Peak Signal Noise Ratio e Structural Similarity Index (SSIM) e conforme o enunciado este relatório serve para explicar as metodologias seguidas em cada um dos passos principais para resolver o problema: - Preparar o pipeline ML para o conjunto de dados de classificação de imagens. - Analisar o modelo de CNN base fornecido. - Treinar modelos generativos (AutoEncoders e GANs) para gerar novas amostras de dados. - Integrar as amostras geradas ao conjunto de dados original e treinar novamente o modelo de CNN. - Avaliar e comparar o desempenho do modelo de CNN com e sem os dados gerados.

Fig. 1: Amostras de diferentes classes

Este trabalho visa demonstrar que a utilização de modelos generativos pode efetivamente aumentar a quantidade de dados de treino, resultando em um classificador mais preciso e robusto para a tarefa de identificação de sinais de trânsito.

3 Metodologia

3.1 AutoEncoder

Os autoencoders são um tipo de rede neuronal utilizada para aprendizagem não supervisionado de representações eficientes dos dados, geralmente para fins de redução de dimensionalidade ou aprendizagem de características. Um autoencoder consiste em duas partes principais:

– **Encoder**: Mapeia os dados de entrada para um espaço de menor dimensionalidade, chamado de espaço latente. Formalmente, o codificador pode ser representado como uma função $h = f(x)$, onde x é o dado de entrada e h é a representação do espaço latente.

– **Decoder**: Reconstrói os dados de entrada a partir da representação do espaço latente. O Decoder pode ser representado como uma função $\hat{x} = g(h)$, onde $\hat{x}$ é a reconstrução do dado de entrada original.

O objetivo do treinamento de um autoencoder é minimizar a diferença entre a entrada x e a saída reconstruída $\hat{x}$. A função de perda usada é o erro quadrático médio (MSE):

$$\mathcal{L}(x, \hat{x}) = \|x - \hat{x}\|^2$$

Arquitetura

A arquitetura do autoencoder proposto é composta por um encoder e um decoder, ambos implementados como redes neuronais feedforward. A seguir, detalhamos a arquitetura específica utilizada. O encoder é responsável por reduzir a dimensionalidade da entrada. Para uma imagem de entrada com 3 canais de cor (RGB) e dimensões 75x75, o codificador comprime os dados através de várias camadas (*fully connected*) seguidas por funções de ativação não lineares (ReLU). Este processo resulta em uma representação latente de dimensionalidade muito menor. O decoder reconstrói a entrada a partir da representação latente. Ele utiliza várias camadas (*fully connected*) e funções de ativação não lineares (ReLU), terminando com uma função de ativação Sigmoid para produzir uma imagem de saída com valores normalizados entre 0 e 1.

$$x \rightarrow \text{Encoder} \rightarrow h \rightarrow \text{Decoder} \rightarrow \hat{x}$$

Treino

O algoritmo de treino do autoencoder ajusta os pesos da rede para minimizar a diferença entre as imagens de entrada e as reconstruídas. O otimizador utilizado foi o ADAM e a função de perda a MSE. Durante cada epoch, o modelo é colocado em modo de treino e, para cada minibatch de dados, os gradientes são zerados, os dados são passados pelo modelo, a loss é calculada e os pesos são atualizados por backpropagation.

Métricas de Avaliação

Para avaliar o desempenho de um autoencoder, utilizamos várias métricas que quantificam a qualidade da reconstrução das imagens.

O **Erro Quadrático Médio (MSE)** é uma métrica que calcula a média dos quadrados das diferenças entre os valores dos pixeis das imagens originais e reconstruidas. Matematicamente, é definido como:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (x_i - \hat{x}_i)^2$$

onde x_i é o valor do pixel original, $\hat{x}_i$ é o valor do pixel reconstruido, e N é o número total de pixeis. O MSE é uma medida comum de erro que penaliza grandes discrepâncias, tornando-o sensível a grandes diferenças entre a imagem original e a reconstruida.

O **Erro Médio Absoluto (MAE)** calcula a média das diferenças absolutas entre os valores dos pixels das imagens originais e reconstruídas. É definido como:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|x_i - \hat{x}_i|$$

O MAE é menos sensível a grandes discrepâncias em comparação com o MSE, proporcionando uma visão complementar da qualidade da reconstrução.

Tanto a MAE como a MSE fornecem medidas diretas do erro de pixel, com MSE sendo mais sensível a grandes erros e MAE proporcionando uma visão complementar menos influenciada por outliers

A **Relação Sinal-Ruído de Pico (PSNR)** é uma métrica que mede a qualidade de reconstrução das imagens em termos de decibéis (dB). É calculada com base no MSE, mas fornece uma interpretação mais intuitiva em termos de qualidade percetual:

$$\text{PSNR} = 20\log_{10}\left(\frac{\text{MAX}_I}{\sqrt{\text{MSE}}}\right)$$

onde MAX_I é o valor máximo possível de um pixel na imagem (por exemplo, 255 para imagens de 8 bits). Um valor de PSNR mais alto indica melhor qualidade de reconstrução. Esta metrica traduz a qualidade de reconstrução em uma escala logarítmica que é mais intuitiva para humanos, facilitando a interpretação de pequenas mudanças na qualidade de reconstrução.

O **Índice de Similaridade Estrutural (SSIM)** mede a similaridade perceptual entre duas imagens, considerando mudanças de luminosidade, contraste e estrutura. É definido como:

$$\text{SSIM}(x,\hat{x}) = \frac{(2\mu_x\mu_{\hat{x}} + C_1)(2\sigma_{x\hat{x}} + C_2)}{(\mu_x^2 + \mu_{\hat{x}}^2 + C_1)(\sigma_x^2 + \sigma_{\hat{x}}^2 + C_2)}$$

onde μ_x e $\mu_{\hat{x}}$ são as médias de x e $\hat{x}$, σ_x^2 e $\sigma_{\hat{x}}^2$ são as variâncias, e $\sigma_{x\hat{x}}$ é a covariância entre x e $\hat{x}$. Os termos C_1 e C_2 são constantes para evitar divisões por zero. O SSIM varia entre -1 e 1, onde 1 indica imagens idênticas.

Esta metrica avalia a similaridade estrutural e perceptual, fornecendo uma medida da qualidade visual que alinha melhor com a percepção humana do que MSE ou MAE.

Estas métricas combinadas permitem uma avaliação abrangente da capacidade do autoencoder de reconstruir imagens de maneira fiel ao original.

Resultados

As figuras abaixo são os resultados obtidos através dos Autoencoders.

(a) Figura 1 (b) Figura 2 (c) Figura 3 (d) Figura 4 (e) Figura 5

(f) Figura 6 (g) Figura 7 (h) Figura 8 (i) Figura 9 (j) Figura 10

Fig. 2: Imagens Geradas pelo AutoEncoder

3.2 GAN

As Generative Adversarial Networks (GANs) são uma classe de modelos de aprendizagem computacional composta por dois blocos: o Generator e o Discriminator. O processo de aprendizagem da GAN é adversarial, o gerador tena "enganar" o discriminador, e o discriminador tenta não se deixar "enganar".

- Generator: O objetivo do generator é gerar dados falsos que sejam indistinguìveis dos dados reais. Ele recebe um valor aleatório de ruído gaussiano, com uma distribuição normal e desvio padrão 1, e tenta transformar essa entrada em dados que possam enganar o Discriminator.
- Discriminator: O discriminator é uma rede neuronal que tenta distinguir entre os dados reais e os dados produzidos pelo generator. Ele recebe como entrada uma imagem e dá uma probabilidade desse dado ser real ou falso.

Arquitetura

A nossa arquitetura foi baseada em redes adversarias generativas condicionais (cGANs) são uma extensão das redes adversarias generativas (GANs) que incorporam informações condicionais no processo de geração de dados. Introduzidas por Mehdi Mirza e Simon Osindero em 2014, as cGANs têm como objetivo condicionar tanto o gerador quanto o discriminador a informações adicionais, como labels da classe, para controlar a forma dos dados gerados. Em uma cGAN, o gerador G aprende a mapear um vetor de ruído z juntamente com um vetor de condição y para o espaço dos dados X, enquanto o discriminador D tenta diferenciar entre dados reais e gerados considerando também y.

Matematicamente, a função de perda de uma cGAN é formulada como:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x|y)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z|y)|y))]$$

Onde:

- $p_{\text{data}}(x)$ é a distribuição real dos dados.
- $p_z(z)$ é a distribuição do vetor de ruído.
- x representa os dados reais.
- z é o vetor de ruído.
- y é o vetor de condição, como os rótulos de classe.
- $D(x|y)$ é a probabilidade do discriminador considerar x como real dado y.
- $G(z|y)$ é o gerador que produz dados sintéticos condicionados em y.

Nesta formulação, o gerador G tenta minimizar a probabilidade de o discriminador D distinguir os dados gerados dos reais, enquanto D tenta maximizar essa probabilidade, considerando também as condições y. As cGANs são particularmente úteis em tarefas onde o controle sobre as características dos dados gerados é necessário, como na geração de imagens condicionadas a labels específicas, permitindo uma maior flexibilidade.

Baseado nesta formulação definimos o nosso modelo que tem as seguintes layers:

O gerador é implementado pela classe `Generator`, que herda de `nn.Module`. A arquitetura do gerador é composta por camadas lineares, normalização em batch e ativações LeakyReLU. A última camada usa a função de ativação `Tanh` para produzir a imagem final.

O discriminador é implementado pela classe `Discriminator`, que também herda de `nn.Module`. A arquitetura do discriminador inclui camadas lineares, ativações LeakyReLU e dropout para prevenir overfitting. A última camada é uma camada linear que retorna a validade da imagem.

A camada `Embedding` é crucial pois transforma as labels da classe em vetores densos, permitindo que o modelo condicione a geração de imagens com base nas características específicas de cada classe.

Treino

A função de perda para o treinamento da cGAN é composta por duas partes: a perda do gerador e a perda do discriminador. Essas perdas são definidas de forma a capturar a natureza adversarial do treino.

A função de perda do Discriminator é projetada para maximizar a probabilidade de atribuir a label correta aod dados reais e falsos. É dada por:

$$\mathcal{L}_D = -\mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x|y)] - \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z|y)|y))]$$

onde:

- $p_{\text{data}}(x)$ é a distribuição real dos dados.
- $p_z(z)$ é a distribuição do vetor de ruído.
- x são as amostras reais.
- y são os vetores de condição.
- $G(z|y)$ são as amostras geradas pelo gerador.

– $D(x|y)$ é a probabilidade do discriminador considerar x como real dado y.

A função de perda do generator é projetada para minimizar a capacidade do discriminador de distinguir entre dados reais e geradas. É dada por:

$$\mathcal{L}_G = -\mathbb{E}_{z\sim p_z(z)}[\log D(G(z|y)|y)]$$

onde:

– $p_z(z)$ é a distribuição do vetor de ruído.
– $G(z|y)$ são as amostras geradas pelo generator.
– $D(G(z|y)|y)$ é a probabilidade do discriminador considerar $G(z|y)$ como real dado y.

O treinamento de uma cGAN envolve a atualização alternada do gerador e do discriminador.

Para cada iteração de treinamento:

– Amostrar um batch de vetores de ruído z e labels de classe y.
– Gerar um batch de imagens $G(z|y)$ usando o gerador.
– Calcular a perda do discriminador $\mathcal{L}_D$ usando dados reais e gerados.
– Atualizar os parâmetros do discriminador para maximizar $\mathcal{L}_D$.
– Calcular a perda do gerador $\mathcal{L}_G$.
– Atualizar os parâmetros do gerador para minimizar $\mathcal{L}_G$.

Ao alternar essas atualizações, o gerador melhora continuamente na criação de imagens que o discriminador considera reais, enquanto o discriminador melhora na distinção entre imagens reais e geradas.

Neste trabalho, utilizamos a função de perda MSE (*Mean Squared Error*) para medir a discrepância entre as previsões do modelo e as labels reais, e utilizámos o otimizador ADAM devido à sua capacidade de adaptar as taxas de aprendizagem para cada parâmetro individualmente, promovendo uma convergência mais rápida e estável durante o treino.

Resultados e Avaliação

O Inception Score (IS) é uma métrica utilizada para avaliar a qualidade e a diversidade das imagens geradas por modelos GAN. A métrica baseia-se na capacidade do modelo gerador de criar imagens realistas que pertencem a classes distintas. O Inception Score é uma métrica valiosa por várias razões:

– **Qualidade das Imagens**: Imagens geradas que são visualmente semelhantes a imagens reais resultam em distribuições de previsão de classe de alta confiança.
– **Diversidade das Imagens**: Imagens geradas que pertencem a uma variedade de classes resultam em uma alta entropia das previsões.

Matematicamente, o Inception Score é definido da seguinte forma:

$$IS(G) = \exp\left(\mathbb{E}_{\mathbf{x}\sim p_g}\left[\mathrm{KL}(p(y|\mathbf{x})\|p(y))\right]\right)$$

onde:

- p_g é a distribuição das imagens geradas pelo gerador G.
- $p(y|\mathbf{x})$ é a distribuição de probabilidade das classes y dadas as imagens $\mathbf{x}$ geradas, conforme predito pelo modelo Inception.
- $p(y)$ é a distribuição marginal das classes, calculada como $p(y) = \mathbb{E}_{\mathbf{x}\sim p_g}[p(y|\mathbf{x})]$.
- $\mathrm{KL}(p(y|\mathbf{x})\|p(y))$ é a divergência Kullback-Leibler entre $p(y|\mathbf{x})$ e $p(y)$.

O Inception Score avalia tanto a qualidade quanto a diversidade das imagens geradas. Um IS mais alto indica que as imagens mais realistas (alta confiança nas previsões de classe) quanto diversificadas (alta entropia).

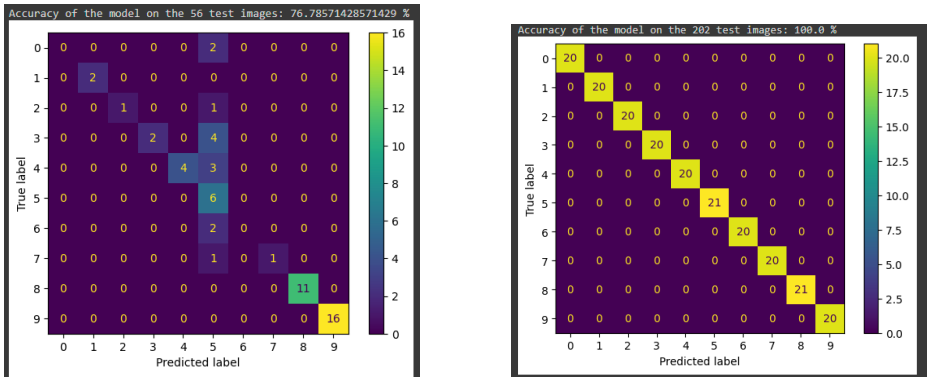
(a) Figura 1 (b) Figura 2 (c) Figura 3 (d) Figura 4 (e) Figura 5

(f) Figura 6 (g) Figura 7 (h) Figura 8 (i) Figura 9 (j) Figura 10

Fig. 3: Imagens Geradas pela GAN

4 Resultados

Para avaliar os resultados obtidos com os dados aumentados foi adicionado aos dados da pasta os dados gerados de forma a balancear o dataset, ficando com cada classe com 100 imagens, sendo as imagens geradas provenientes 50/50 dos Autoencoders e da GAN. Com isto conseguimos aumentar significativamente a performance do modelo no dataset de validação com a cnn treinada com 50 epochs com o dataset de treino aumentado.



(a) Acc: 76% — Dataset Inicial (b) Acc: 100% — Dataset Aumentado

Fig. 4: Comparação dos Resultados antes e depois dos dados Aumentados

5 Conclusão

Este relatório detalhou um método para superar a escassez de dados em classificação, neste caso, de sinais de trânsito usando AutoEncoders e GANs. Mais do que os resultados, destacamos o processo de implementação e avaliação dessas técnicas de geração de dados, sendo que o foco foi explorar como estes modelos podem ser integrados em pipelines existentes de Machine Learning para enriquecer conjuntos de dados de forma a melhorar os resultados de um modelo base sem um conjunto de dados enriquecido, e os resultados obtidos foram bastante positivos.

References

1. I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," *arXiv preprint arXiv:1411.1784*, 2014.