Capítulo 3

Utilização de Técnicas de *Data Augmentation* em Imagens: Teoria e Prática

Maíla Claro, Luis Vogado, Justino Santos and Rodrigo Veras

Abstract

Recent advances in the acquisition and processing of digital images have made it possible to use the data provided from medical images in new and revolutionary ways, especially when associated with artificial intelligence. However, there is still a certain deficiency in obtaining these image databases and certain computational techniques require a large volume of data to provide reliable solutions and the ability to generalize different inputs. It is in this context that the Data Augmentation techniques are inserted into images. These techniques are intended to generate new virtual data from real examples to provide more volume and support the development of more robust applications. In this mini-course, the main techniques of Data Augmentation in images will be approached in a theoretical and practical way, from the most traditional to those involving Deep Learning.

Resumo

Recentes avanços na aquisição e processamento de imagens digitais permitiram o uso eficiente e inovador do dados fornecidos por meio de bases de dados em aplicações de inteligência artificial. Entretanto, ainda há uma certa deficiência na obtenção dessas bases de imagens e determinadas técnicas computacionais necessitam de um grande volume de dados para fornecer soluções confiáveis e com a capacidade de generalizar diferentes entradas. É nesse contexto que se inserem as técnicas de Data Augmentation em imagens. Essas técnicas tem como intuito gerar novos dados virtuais a partir de exemplos reais para prover mais volume e dar mais suporte ao desenvolvimento de aplicações mais robustas. Neste minicurso serão abordadas de forma teórica e prática, as principais técnicas de Data Augmentation em imagens, desde as mais tradicionais até as que envolvem Deep Learning.

3.1. Introdução

O Processamento Digital de Imagens (PDI) não é uma tarefa simples, na realidade envolve um conjunto de tarefas interconectadas. Tudo se inicia com a captura de uma imagem, a qual, normalmente, corresponde à iluminação que é refletida na superfície dos objetos, realizada através e um sistema de aquisição. Após a captura por um processo de digitalização, uma imagem precisa ser representada de forma apropriada para tratamento computacional. Imagens podem ser representadas em duas ou mais dimensões [Acharya and Ray 2005].

A classificação de imagens é uma das grandes áreas de aplicação do Aprendizado de Máquina, a quantidade de exemplos rotulados disponíveis e os resultados estão relacionados, onde quanto maior a quantidade de exemplos, maior a capacidade de predição dos classificadores gerados. No entanto, em situações da vida real, possuir bases de dados com uma grande quantidade de exemplos rotulados nem sempre é uma tarefa fácil, e muitas vezes, custosa. Desse modo, podemos utilizar abordagens de *Data Augmentation* (DA), que é um conceito fundado por técnicas computacionais com o objetivo de aumentar a quantidade de exemplos rotulados em um conjunto de dados e assim, melhorar os resultados obtidos [Taylor and Nitschke 2018].

A classificação de imagens possui diversas aplicações, por exemplo, a análise de dados de satélites [Penatti et al. 2015], reconhecimento facial [Vargas et al. 2016], detecção de doenças ([Claro et al. 2020]), dentre outras funções. Porém, em aplicações como imagens aéreas e médicas, os dados são limitados e, dessa forma é evidente a necessidade de utilização de técnicas de DA, já que a classificação de imagens oferece uma boa quantidade de aplicações e utilidade ao ser humano.

3.1.1. Representação Computacional de Imagens

Para representar e manipular imagens em um computador é necessário definir um modelo matemático apropriado [Gomes and Velho 1997]. Como uma imagem é resultado de um estímulo de luz, é possível estabelecer um universo matemático no qual se possa definir modelos abstratos de imagens de forma que permitam sua representação discreta com o propósito de possibilitar uma codificação da mesma em um computador [Gomes and Velho 1997].

Considere um escala de cinza, tal como mostrado na Figura 3.1, para representar uma imagem monocromática, i e j são dois números inteiros tais que $1 \leq \mathrm{i} \leq m$ e $1 \leq \mathrm{j} \leq n$. *f(i, j)* é a função inteira tal que $1 \leq f(i, j) \leq X$, onde X indica o valor branco em uma escala de cinza. Nesta situação uma matriz F (Figura 3.1) é chamada de imagem digital [Batchelor and Waltz 2012].

Uma imagem digital é considerada como uma matriz no qual os índices de linhas e colunas identificam um ponto na imagem, sendo que o respectivo valor do elemento dessa matriz identifica a intensidade (brilho) da imagem naquele ponto. A esses elementos de uma matriz digital dar-se o nome de “pixels". Na Figura 3.2 vemos um exemplo de uma imagem digital [Gonzalez and Woods 2000].

Em PDI trabalhamos basicamente com três tipos de imagens: Imagem Colorida (três bandas de cores), Imagem em Escala de Cinza (uma banda de cor) e Imagem Binária

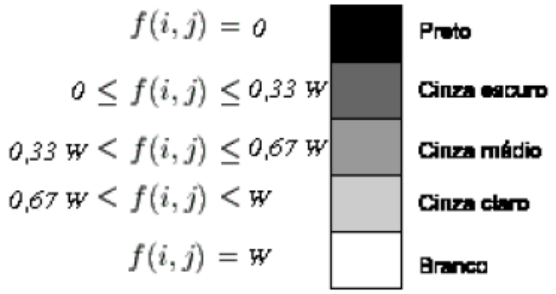

Figura 3.1. Demonstração de um mapeamento entre os valores armazenados na matriz F e as suas respectivas representações em tons de cinza [Batchelor and Waltz 2012]

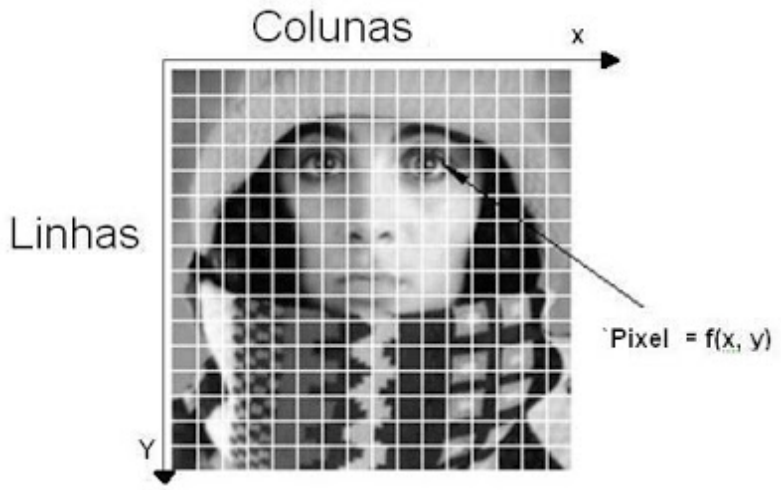

Figura 3.2. Exemplo de uma representação de Imagem Digital.

(Preto e Branco). No caso, da imagem em escala de cinza, a intensidade dos pontos da imagem, ou seja o valor do pixel, varia numa escala entre 0 e 255, onde o 0 (zero) indica a cor preta e o 255 a cor branca, tendo nesse intervalo o intensidade do cinza, como mostra a Figura 3.3.

Figura 3.3. Níveis de intensidade de Cinza.

Existe outro padrão para se trabalhar a intensidade do pixel, que é o intervalo entre 0 e 1, por exemplo: 0,15, 0,52, onde o 0 (zero) pode ser considerado como branco e o 1 (um) preto. É o chamado padrão com valores contínuos (fracionários), e no caso do intervalo 0 à 255 valores discretos (inteiros). A Figura 3.4 demonstra como seria a representação de uma imagem monocromática e os valores de seus respectivos pixels [Marques Filho and Neto 1999].

Já para imagens coloridas, ou seja, que possuem mais de uma banda de frequên-

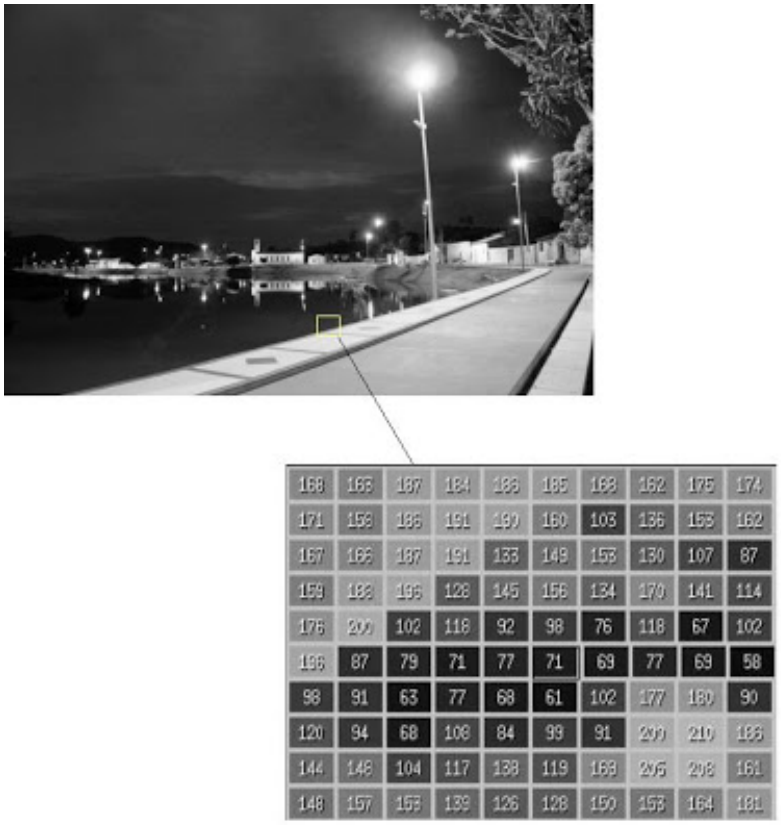

Figura 3.4. Imagem em monocromática e seus pixels.

cias, cada banda possui sua própria função de intensidade de brilho. Um exemplo de imagens coloridas, e é a mais utilizada, é o padrão RGB, apesar de existirem outros como por exemplo o HSI (*hue, saturation, intensity*). No modelo RGB a imagem possui três bandas que compõem as três cores primárias, Vermelho (*Red*), Verde (*Green*) e Azul (*Blue*).

Pode-se considerar, portanto, que uma imagem colorida é a composição de três imagens monocromáticas, denominadas, respectivamente, de banda vermelha (ou banda R), banda verde (ou banda G), e banda azul (ou banda B) e que a sobreposição destas três imagens, que individualmente são monocromáticas, compõe uma imagem colorida (ver Figura 3.5). E com a sobreposição/mistura de cada uma dessas cores e suas respectivas intensidades obtêm-se a formação de outras cores. Já no caso das imagens binárias, as opções de intensidade do pixel são apenas o preto ou branco, onde dependendo da imagem e do problema o preto pode representar o fundo da imagem e o branco os objetos ou vice-versa

Existem várias maneiras de lidar com complicações associadas a dados limitados no aprendizado de máquina. O aumento da imagem é uma técnica útil na construção de redes neurais convolucionais que podem aumentar o tamanho do conjunto de treinamento sem a aquisição de novas imagens. A ideia é simples, será preciso duplicar as imagens com algum tipo de variação para que o modelo possa aprender com mais exemplos. Idealmente, podemos aumentar a imagem de maneira a preservar os recursos essenciais para

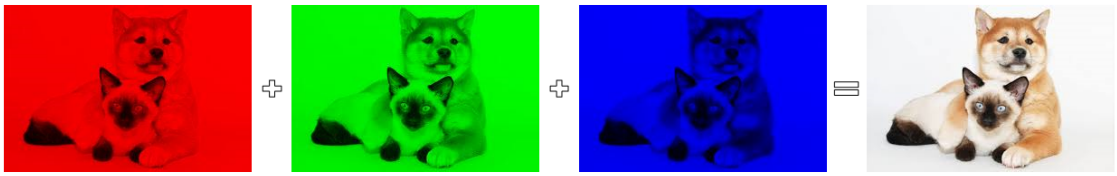

Figura 3.5. Imagens das Bandas R, G, B e RGB.

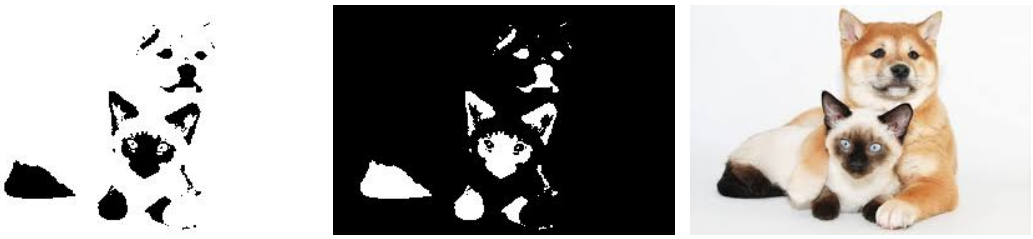

Figura 3.6. Exemplos de imagens em Binário com o plano de fundo branco, Binário com o plano de fundo preto e uma Imagem colorida.

fazer previsões, mas reorganiza os pixels o suficiente para adicionar algum ruído. O aumento será contraproducente se produzir imagens muito diferentes daquilo em que o modelo será testado, portanto, esse processo deve ser executado com cuidado.

3.2. Técnicas de *Data Augmentation*

Dados limitados são um grande obstáculo na aplicação de modelos de aprendizagem profunda, como redes neurais convolucionais. Frequentemente, classes desequilibradas podem ser um obstáculo adicional, embora possa haver dados suficientes para algumas classes, igualmente importantes, mas classes abaixo da amostra sofrerão com baixa precisão específica da classe. Esse fenômeno é intuitivo. Se o modelo aprender com alguns exemplos de uma determinada classe, é menos provável prever a invalidação da classe e os aplicativos de teste [Shorten and Khoshgoftaar 2019].

As primeiras demonstrações que mostram a eficácia das técnicas de *Data augmentation* vêm de transformações simples, como inversão horizontal, aumento de espaço de cores e corte aleatório. Essas transformações codificam muitas das invariâncias discutidas anteriormente que apresentam desafios para as tarefas de reconhecimento de imagem [Shorten and Khoshgoftaar 2019]. As técnicas abordadas nesta pesquisa são transformações geométricas, transformações de espaço de cores, filtros de núcleo, imagens misturadas, *random erasing*, aumento de espaço de recursos, treinamento *adversarial*, aprimoramento baseado em *Generative Adversarial Networks* (GAN) e transferência de estilo neural. Esta seção irá explicar como cada algoritmo de aumento funciona.

3.2.1. Transformações Geométricas

Esta seção descreve diferentes técnicas baseadas em transformações geométricas e muitas outras funções de processamento de imagem. Estes métodos de transformações geométricas são caracterizadas por sua facilidade de implementação. A compreensão dessas transformações fornecerá uma boa base para uma investigação mais aprofundada das téc-

nicas de aumento de dados.

Flipping

Flipping é uma técnica em que é feito um inversão na imagem original, onde essa inversão pode ser na horizontal ou na vertical. Inverter o eixo horizontal é muito mais comum do que inverter o eixo vertical. Esse aumento é um dos mais fáceis de implementar e provou ser útil em conjuntos de dados como CIFAR-10 e ImageNet. Em conjuntos de dados que envolvem reconhecimento de texto como MNIST [Deng 2012] ou SVHN, essa não é uma transformação de preservação de rótulo [Shorten and Khoshgoftaar 2019]. Como por exemplo na base MNIST em que é uma identificação dos números de 0 (zero) a 9 (nove), o número 6 (seis) poderia ser confundido com o número 9, depois de ser realizada uma inversão de horizontal e depois uma vertical. As Figuras 3.7 e 3.8 apresentam exemplos de imagens invertidas.

Figura 3.7. Exemplos da utilização da técnica Flip Horizontal.[1]

Figura 3.8. Exemplos da utilização da técnica Flip Vertical.[1]

Rotação

Os aumentos de rotação são feitos girando a imagem para a direita ou esquerda em um eixo entre 1º e 359º. A segurança dos aumentos de rotação é fortemente determinada pelo parâmetro do grau de rotação. Rotações leves, como entre 1 e 20 ou -1 a -20, podem ser úteis em tarefas de reconhecimento de dígitos, como MNIST, mas à medida que o grau de rotação aumenta, o rótulo dos dados não é mais preservado após a transformação [Shorten and Khoshgoftaar 2019]. A Figura 3.9 demonstra exemplos de imagens quadradas giradas em ângulo reto.

Translação

Mudar as imagens para a esquerda, direita, para cima ou para baixo pode ser uma transformação muito útil para evitar distorções posicionais nos dados. Por exemplo, se todas as imagens em um conjunto de dados estiverem centralizadas, o que é comum em conjuntos de dados de reconhecimento de face, isso exigiria que o modelo também fosse testado em imagens perfeitamente centralizadas. Como a imagem original é traduzida em

[1]Fonte: https://towardsdatascience.com/exploring-image-data-augmentation-with-keras-and-tensorflow-a8162d89b844

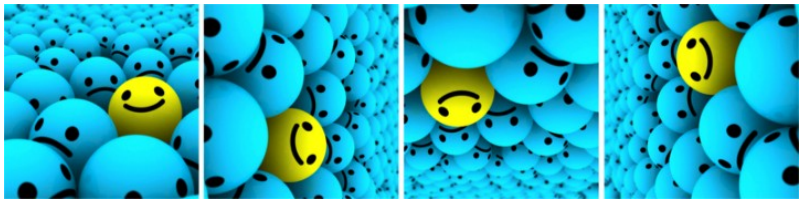

Figura 3.9. Da esquerda para à direita as imagens são giradas 90º graus no sentido horário em relação à anterior.[2]

uma direção, o espaço restante pode ser preenchido com um valor constante, como 0 ou 255, ou pode ser preenchido com ruído aleatório ou gaussiano. Esse preenchimento preserva as dimensões espaciais da imagem pós-aumento [Shorten and Khoshgoftaar 2019]. Esse método de aprimoramento é muito útil, pois a maioria dos objetos pode ser localizada em praticamente qualquer lugar da imagem. Isso força a rede neural convolucional a procurar em todos os lugares da imagem como é apresentado um exemplo na Figura 3.10.

Figura 3.10. Da esquerda, temos a imagem original, a imagem traduzida para a direita e a imagem traduzida para cima.[2]

Cropping ou Corte

Cortar imagens pode ser usado como uma etapa prática de processamento para dados de imagem com dimensões métricas de altura e largura cortando uma área central de cada imagem. Além disso, o corte aleatório também pode ser usado para fornecer um efeito muito semelhante às traduções. O contraste entre o corte aleatório e as traduções é que o corte reduzirá o tamanho da entrada, como (256, 256) para (224, 224), enquanto as traduções preservam as dimensões espaciais da imagem. Dependendo do limite de redução escolhido para o corte, isso pode não ser uma transformação de preservação do rótulo. A Figura 3.11 apresenta exemplos de cortes nas imagens.

Zoom / escala

Um zoom aleatório é obtido pelo argumento *zoom_range*. Um zoom menor que 1.0 amplia a imagem, enquanto um zoom maior que 1.0 diminui o zoom da imagem, como podemos observar na Figura 3.12.

Shear / cisalhamento

A transformação de cisalhamento inclina a forma da imagem. Isso é diferente da rotação no sentido de que, na transformação de cisalhamento, fixamos um eixo e esticamos a imagem em um determinado ângulo conhecido como ângulo de cisalhamento.

[2]Fonte: https://nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2/

Figura 3.11. Da esquerda para a direita, temos a imagem original, uma seção quadrada cortada da parte superior esquerda e, em seguida, uma seção quadrada cortada da parte inferior direita. As seções cortadas foram redimensionadas para o tamanho da imagem original.[2]

Figura 3.12. Exemplos da utilização da técnica Zoom.[1]

Isso cria um tipo de “alongamento"na imagem, que não é visto em rotação, como podemos visualizar na Figura 3.13. Temos a função *Shear_range* que especifica o ângulo da inclinação em graus.

Figura 3.13. Exemplos da utilização da técnica de Shear.[1]

3.2.2. Transformações Fotométricas

Além de transformações geométricas capazes de mudar a forma, tamanho e posição de elementos dentro imagem, outros tipos de transformações atuam sobre a composição pictórica delas. Mudanças aplicadas sobre o valor armazenado nos pixels e em alguns casos a substituição parcial de conteúdo são encontradas em abordagens como adição de ruído, mudanças no espaço de cor e processamentos de imagem, filtragem com kernel, combinação de imagens e *random erasing*.

Adição de ruído

Ruídos em imagens são caracterizados por valores aleatórios em pixels. Geralmente são provocados por fatores relacionados ao processo de obtenção da imagem. Em imagens analógicas, ruídos podem ser causados pelo próprio filme (aspereza da superfície), já em imagens digitais, variações de temperatura, falhas no sensor, entre outros podem causar esse efeito.

Inserir ruído em imagens digitais como técnica de DA foi testado por [Moreno-Barea et al. 2018] em nove base de dados. A adição de ruído em imagens pode ajudar

CNNs a aprender características mais robustas. A Figura 3.14 exemplifica dois tipos de ruídos comuns o gaussiano e o impulsivo.

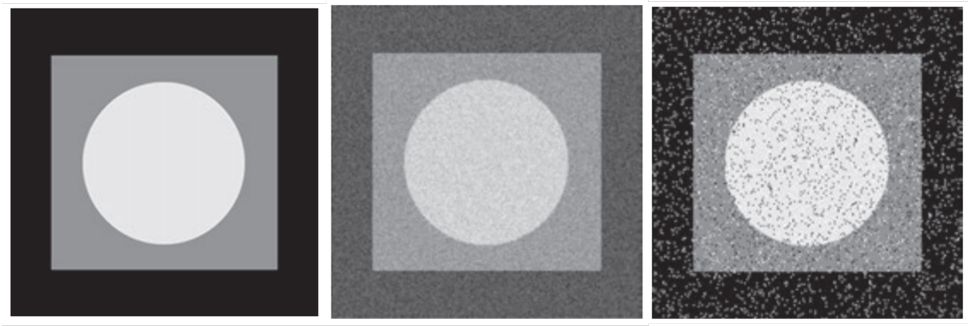

Figura 3.14. Ruídos gaussiano (b) e impulsivo (c) aplicados sobre imagem modelo (a). [Gonzalez and Woods 2000]

Transformações no espaço de cores

Os dados de uma imagem digital podem ser representados como um array de três dimensões (altura $\times$ largura $\times$ canais de cores). O aprimoramento no âmbito das cores é outra estratégia prática para implementação de *Data Augmentation*.

Os aprimoramentos de cores mais simples incluem o isolamento de um único canal de cor, como R, G ou B do modelo de cor RGB. Uma imagem pode ser rapidamente convertida em sua representação em um dos canais de cores, isolando essa matriz e adicionando duas matrizes povoadas com o valor zero nos outros canais de cores. Além disso, os valores RGB podem ser facilmente manipulados com operações simples de matriz para aumentar ou diminuir o brilho da imagem [Shorten and Khoshgoftaar 2019]. A Figura 3.15 ilustra essas técnicas.

Outros tipos de aumento podem derivar de ajustes de histograma de cores que descreve a imagem. A alteração dos valores de intensidade nesses histogramas resulta em alterações de iluminação e contraste, como as usadas nos aplicativos de edição de fotos.

É possível pensar em muitas formas de implementar transformações aplicáveis sobre o espaço de cores. Aumentar ou reduzir valor de todos os pixel em um valor fixo - simulando um ambiente com mais ou menos luz, iluminar cores específicas - mudança na temperatura de cor, a utilização de escala de cinza são só alguns exemplos.

Há de se observar que algumas aplicações são sensíveis à transformações no espaço de cores podendo gerar efeitos negativos sobre a tarefa. A utilização de escala de cinza, por exemplo, é uma opção capaz de gerar ganhos no processamento, mas que pode causar perdas na acurácia [Chatfield et al. 2014]. Outro caso, o escurecimento pode tornar objetos da imagem indetectáveis ou suas bordas se fundirem com a cena, dificultando uma possível tarefa de segmentação.

Filtragem com *kernel*

Também conhecida por filtragem espacial, consiste em deslisar uma janela sobre a imagem original realizando operações em cada posição ocupada, similar ao mecanismo

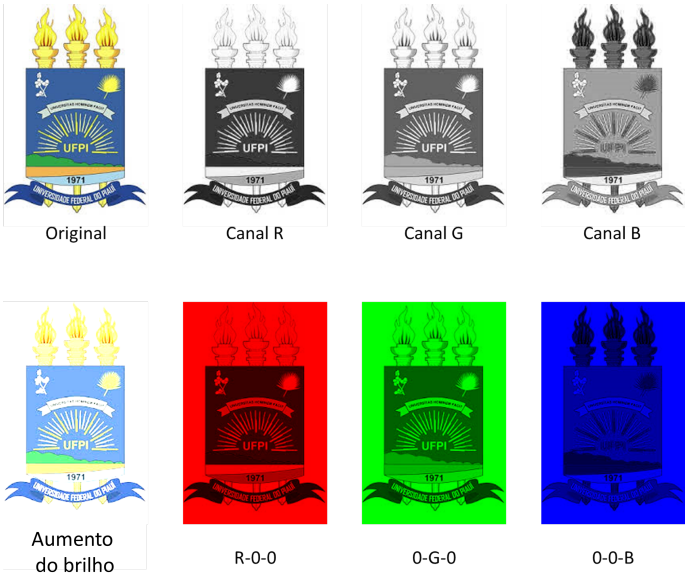

Figura 3.15. Transformações simples baseadas no espaço de cores.

de convolução. Estruturas internas de uma CNN já são capazes de realizar tais operações, portanto para alguns casos, não compensa aplicar esse tipo de aumento.

Duas aplicações de kernels muito comuns no processamento de imagens, são os filtros de borramento e de aguçamento (veja a Figura 3.16). Intuitivamente, realizar DA por meio de borramento das imagens de treino pode aumentar a resistência do algoritmo sobre imagens borradas por movimento no conjunto de testes. Adicionalmente aguçar imagens pode destacar detalhes sobre o objeto de interesse [Shorten and Khoshgoftaar 2019].

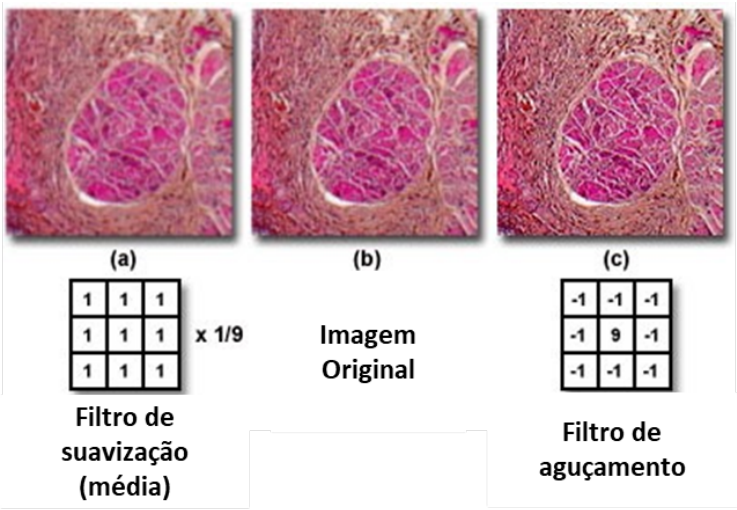

Figura 3.16. Filtros de borramento e aguçamento.[2]

[2]Fonte: https://static3.olympus-lifescience.com/data/olympusmicro/primer/digitalimaging/images/processintro/processintrofigure7.jpg

Uma técnica de DA avaliada por [Kang et al. 2017], denominada de *PatchShuffle Regularization* utiliza um mecanismo semelhante, no entanto em vez de operar sobre o nível de intensidade, em sua abordagem o filtro realiza trocas aleatórias nas posições dos pixels, com uma determinada taxa de probabilidade de mudança p. A Figura 3.17 ilustra o seu funcionamento.

Figura 3.17. ***PatchShuffle Regularization.*** ***a*****: aplicação do um filtro 2×2 sobre uma imagem 4×4.** ***b*****: Exemplos da aplicação sobre imagens reais com diferentes tamanhos de filtros. [Kang et al. 2017]**

Combinação de imagens

O aumento de dados por meio da combinação consiste em gerar imagens cujo conteúdo venha de um conjunto de outras imagens da base. O conteúdo gerado pode não fazer muito sentido com a realidade (Figura 3.18), divergindo dos exemplares que serão naturalmente fornecidos na etapa de teste.

Métodos lineares foram avaliados por [Inoue 2018]. Sua abordagem combinou pares de imagens por meio da média das intensidades dos pixels correspondentes (semelhante a Figura 3.18-b). Apesar de contraintuitivo, a inclusão de imagens de classes diferentes para serem combinadas (mesmo preservando apenas um dos rótulos) promoveu resultados melhores no conjunto de validação quando comparados com as técnicas tradicionais. Há de se observar que sua metodologia alternou épocas de treinamento com e sem imagens combinadas, e um ajuste fino com imagens puras finalizou o treinamento.

A (Figura 3.18-c) ilustra métodos não lineares de combinar imagens, esses métodos foram avaliados por [Summers and Dinneen 2019]. Utilizando estas abordagens eles obtiveram resultados de com performance melhor quando comparados aos métodos tradicionais.

Outra forma de combinar imagens também de maneira não linear, encontradas no trabalho de [Takahashi et al. 2019], consiste em recortes aleatórios de imagens que serão

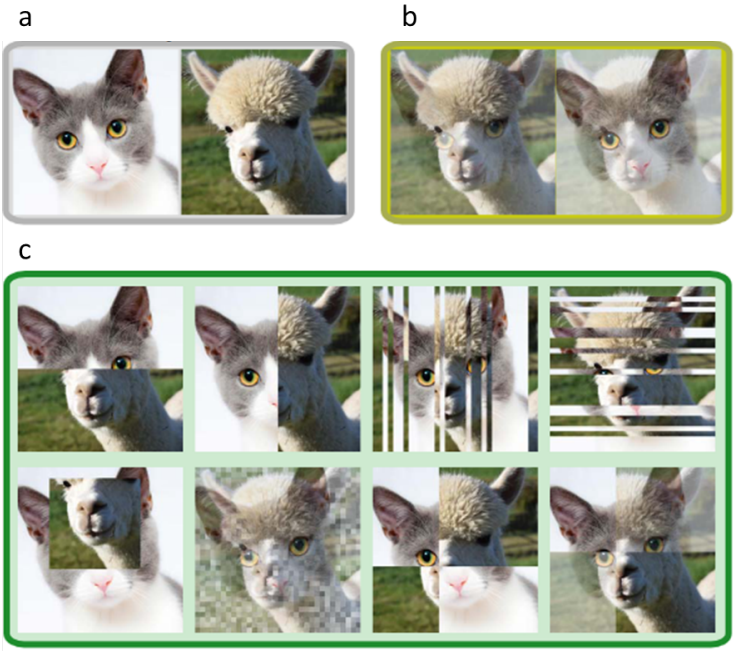

Figura 3.18. Combinação de imagens. *a*: imagens de entrada, *b*: combinação com método linear, *c*: combinação por métodos não lineares. [Summers and Dinneen 2019]

concatenados uns aos outros, a montagem decorrente desse processo forma uma nova imagem (Figura 3.19).

Random erasing

Essa técnica introduzida por [Zhong et al. 2020] consiste em “apagar"regiões aleatórias da imagem. Na prática uma parte da imagem tem seus pixels substituídos por um valor fixo ou um ruído conforme ilustrado na Figura 3.20.

A abordagem de remover uma parte da imagem ataca um problema conhecido com oclusão. Com essa técnica algoritmos de reconhecimento passam a se ater a detalhes gerais da imagem se tornando mais robustos sobre situações comuns onde algum objeto

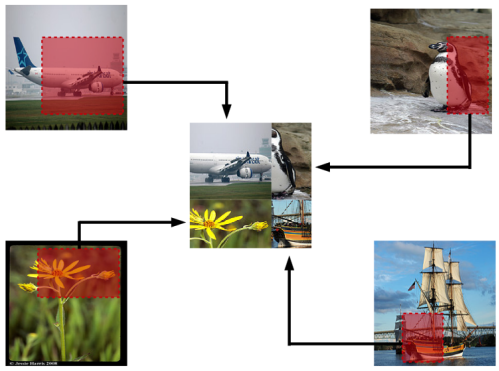

Figura 3.19. combinação com métodos não lineares. [Takahashi et al. 2019]

toma parcialmente a frente de outro a ser reconhecido. Por exemplo, uma CNN para reconhecimento de pessoas pode se ater apenas aos olhos ou ao rosto, treinar a rede com partes excluídas vai forçar a rede a dar importância a outras características.

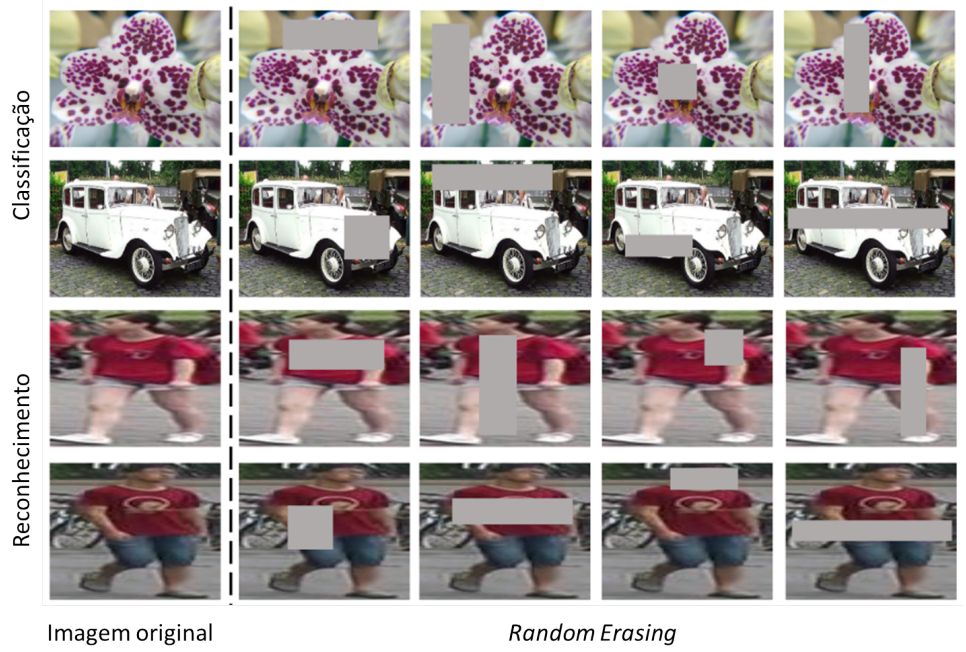

Figura 3.20. *Random erasing* [Zhong et al. 2020]

3.2.3. Transformações baseadas em *Deep Learning*

Feature Space Augmentation

As técnicas mais tradicionais utilizadas no aumento de dados são baseadas em imagens como entrada para as redes neurais convolucionais. No entanto, também é possível aplicar técnicas de aumento em um espaço de características gerado por camadas totalmente conectadas de uma CNN [DeVries and Taylor 2017].

Dentre as principais técnicas desse tipo de aumento, temos: *Synthetic Minority Oversampling Technique* (SMOTE) [Nitesh VC 2002] e *Autoencoders* [Pascal and Hugo 2010]. A SMOTE é uma técnica utilizada para gerenciar o desbalanceamento de classes em um determinado problema, aumentando a classe desbalanceada por meio da sintetização de novos exemplos a partir do algoritmo de agrupamento *K-nearest neighbor* (KNN) [Aha and Kibler 1991]. O funcionamento do SMOTE é iniciado com a escolha aleatória de um exemplo da classe desbalanceada. A partir do exemplo, os k vizinhos mais próximos são selecionados. Dentre eles, um dos vizinhos é escolhido também de forma aleatória e o novo exemplo é gerado a partir de um novo ponto entre o exemplo inicial e o seu vizinho selecionado. Esse procedimento é executado até que a classe esteja balanceada.

Os *Autoencoders* também são utilizados no aumento de dados para o espaço de características. Essa estrutura é dividida em duas partes, a primeira é o *encoder* que irá receber uma imagem como entrada e retorna um vetor de baixa dimensionalidade. Esse

vetor serve de entrada para o *decoder*, que irá reconstruir a imagem original a partir do vetor compactado pelo *encoder*. Sendo assim, o exemplo gerado não possuirá exatamente as mesmas características da original, formando uma imagem artificial. Na Figura 3.21 apresentamos um exemplo simples de *Autoencoder*.

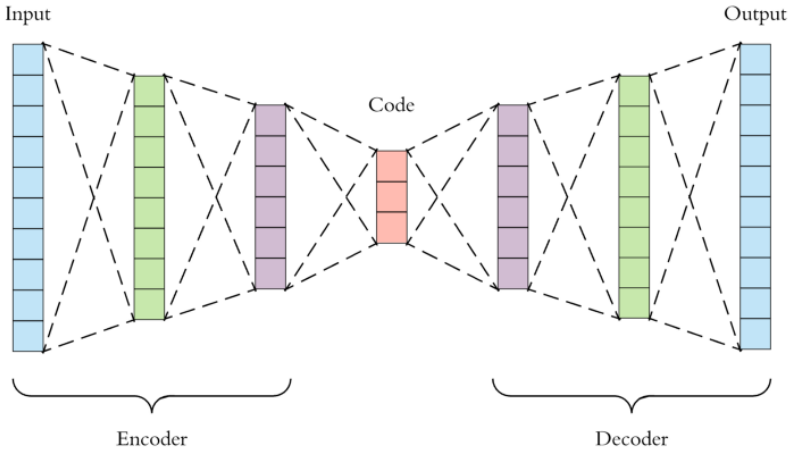

Figura 3.21. Exemplo de *Autoencoder*.[3]

Outra forma de aumento do espaço de características é a inserção ruídos, interpolações e extrapolação. No entanto, assim como as outras técnicas, a principal desvantagem desse tipo de aumento é a dificuldade na interpretação dos vetores. Já os *Autoencoders* para CNNs demandam um tempo excessivo para treinar devido a presença de inúmeros dados.

Treinamento Adversarial

Uma das soluções para buscas no espaço de características e para auxiliar no desenvolvimento de técnicas de aumento é o treinamento *Adversarial*. Essa técnica consiste na utilização de duas ou mais redes neurais que possuem diferentes objetivos. Um dos objetivos é realizar ataques como a inserção de ruídos nas imagens no intuito de desafiar completamente a intuição sobre como esses modelos representam imagens. Diante disso, esses ataques são realizados ao treinar uma rede rival com aumentos provenientes de outra rede [Moosavi-Dezfooli et al. 2015].

Um dos principais objetivos para utilizar esse tipo de aumento é a melhora proporcionada nos pontos fracos no limite de decisão aprendido. Sendo assim, esse tipo de treinamento também auxilia na descobertas de pontos fracos ou pertubações em modelos de *deep learning*. A eficácia do treinamento *Adversarial* na forma de busca por ruído ou aumento ainda é um conceito relativamente novo que não foi amplamente testado e compreendido.

No entanto não fica claro na literatura se essa técnica auxilia na redução do *overfitting*. Trabalhos futuros buscam expandir a relação entre resistência a ataques adversários e desempenho real em conjuntos de dados de teste.

[3]Fonte: `https://towardsdatascience.com/applied-deep-learning-part-3\-autoencoders-1c083af4d798`

Aumento com GANs

Outra técnica utilizada nos últimos anos para o aumento de dados são os modelos generativos. Esses modelos utilizam os dados da própria base de dados para gerar novas imagens com características similares à base de dados. Levando em consideração os ataques adversariais e a estrutura dos *Autoencoders*, foram propostas as *Generative Adversarial Networks* (GANs). Essas arquiteturas possuem duas redes adversárias, o gerador que gera novos exemplos a partir da base de dados e o discriminador que prediz se o exemplo gerado pode ser considerado real ou falso. O intuito do gerador é enganar o discriminador, ou seja, fazer com que os exemplos gerados estejam mais próximos possíveis de exemplos reais. A utilização dessas arquiteturas vem ganhando destaque ao longo dos anos, uma vez que seus resultados se mostraram surpreendentes.

No domínio das imagens, a GAN mais conhecida para aplicações é a *Deep Convolutional Generative Adversarial Network* (DCGAN) [Radford et al. 2015]. Essa arquitetura utiliza CNNs no lugar de redes neurais simples como em GANs mais simples. A partir de um ruído aleatório, as camadas deconvolucionais da DCGAN tendem a gerar exemplos correspondentes a base de dados e classe das imagens de entrada. O exemplo artificial servirá de entrada para o discriminador, que é nada mais que uma CNN comum com uma função de ativação em sua última camada.

Diferente dos *Autoencoders*, as DCGANs não precisam ser simétricas. Sendo assim, apenas a saída do gerador e do discriminador devem possuir tamanhos equivalentes. Na Figura 3.22 apresentamos um exemplo de DCGAN em uma aplicação da literatura.

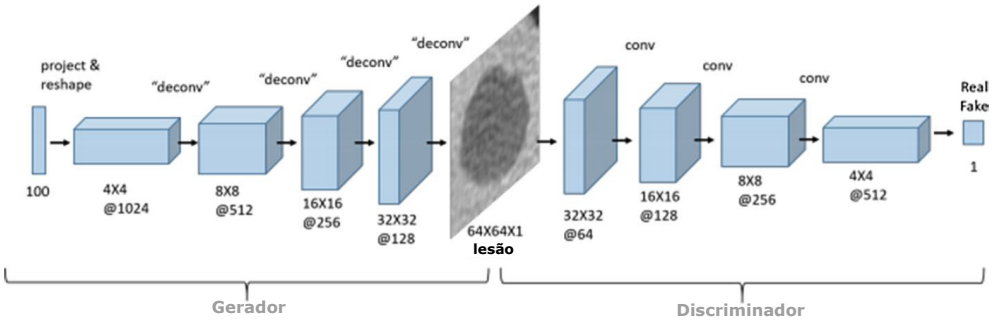

Figura 3.22. Exemplo de DCGAN aplicada a geração de imagens de lesão de fígado. [Frid-Adar et al. 2018]

Outros tipos de GANs também são empregados na literatura como diferentes finalidades, como as *Progressively Growing GANs* [Heljakka et al. 2018]. Essa arquitetura tem como principal objetivo trabalhar com imagens em alta resolução, partindo baixas resoluções para as mais altas. Outra arquitetura bastante utilizada é a CycleGAN [Zhu et al. 2017]. Essa GAN consegue transferir o domínio de uma base de imagens para outro. Isso permite que uma entrada de uma determinada classe seja remapeada para outra, fazendo com que o discriminador faça a predição de qual das classes pertence a imagem gerada. Após a predição, a imagem retorna para outro gerador que tentará retornar a mesma para sua classe original. Um segundo discriminador é então utilizado para predizer se essa imagem transformada é ou não da classe original, formando um ciclo.

Existe um grande potencial no uso de GANs para o aumento de dados. No entanto, elas ainda apresentam alguns desafios, como: dificuldade na obtenção de imagens com altas resoluções, instabilidade no treinamento e não convergência e a necessidade de grandes conjuntos de dados para treinamento.

Neural Style Transfer

Uma dentre as técnicas que melhor representam a capacidade de aprendizado das CNNs é a transferência de neural de estilo ou *Neural Style Transfer* [Gatys et al. 2015]. Essa técnica consiste na transferência de características intrínsecas de uma imagem para outra utilizando CNNs. Essa capacidade de transferir o estilo possibilita que essa técnica sirva como ferramenta para o aumento de dados. A transferência ocorre através da seleção de camadas convolucionais e da retro-propagação da *loss* através das épocas, permitindo que o conteúdo da imagem de entrada seja replicado para a imagem base.

A transferência de estilo permite que a imagem originada do processamento possua características de cor, textura e iluminação da imagem de estilo sem que o conteúdo original seja completamente descartado. Isso ocorre devido a possibilidade das CNNs conseguirem aprender tanto características mais gerais quanto mais específicas de uma imagem. Na Figura 3.23 demonstramos um exemplo de transferência de estilo.

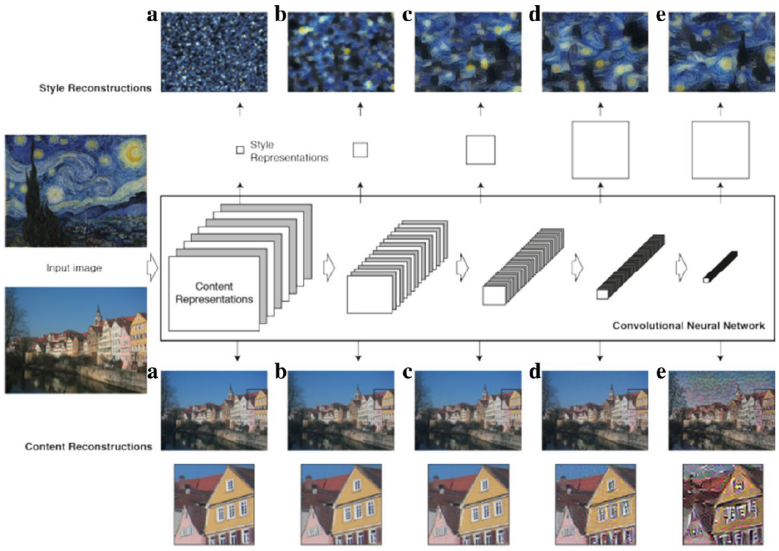

Figura 3.23. Imagens utilizadas para a realização da transferência de estilo e seus resultados. [Shorten and Khoshgoftaar 2019]

Uma das desvantagens desse aumento é o esforço necessário para selecionar os diferentes estilos para serem transferidos. Além disso, se a base utilizada para definir os estilos for pequena, pode ocorrer a inserção de um *bias* nos resultados. Outra desvantagem é o recurso computacional exigido para processar e gerar todas as imagens necessárias para o aumento, mesmo sendo uma alternativa que pode proporcionar uma melhor generalização para CNNs, a utilização de outras técnicas de aumento podem ser mais eficientes. Trabalhos recentes apresentaram formas mais rápidas de transferir o estilo para imagens de conteúdo. No entanto, são limitados pois os estilos disponíveis são predefinidos.

3.3. Aplicações de *Data Augmentation*

O aumento de dados é uma técnica popular amplamente usada para aprimorar o treinamento de redes neurais convolucionais. Embora muitos de seus benefícios sejam bem conhecidos por pesquisadores e profissionais de aprendizagem profunda, seus efeitos implícitos de regularização, em comparação com as técnicas populares de regularização explícita, como *weight decay* e *dropout*, permanecem praticamente sem estudo. De fato, as redes neurais convolucionais para classificação de objetos de imagem geralmente são treinadas com aumento de dados e regularização explícita, assumindo que os benefícios de todas as técnicas sejam complementares [Hernández-García and König 2018].

3.3.1. Estado da arte que aplicam *Data Augmentation*

Diversas técnicas de DA foram propostas na literatura, pois obter grandes quantidades de dados é uma tarefa difícil e de extrema importância para se obter bons resultados com os algoritmos de aprendizagem de máquina. Portanto, alguns trabalhos relacionados serão analisados e detalhados nesta seção.

Existem algumas técnicas de transformações de imagens tradicionais que são mais conhecidas e utilizadas, como vimos na Seção 3.2. No entanto, tem-se várias propostas de variações desses métodos e as novas abordagens utilizadas são os métodos gerativos, como as *Generative Adversarial Networks* (GANs).

Várias aplicações podem ser beneficiadas com as técnicas de DA. A visão computacional, por exemplo, se baseia em detectar objetos em imagens e é melhor utilizada com uma grande quantidade de dados. Dessa forma, diversos trabalhos utilizaram técnicas de DA em seus conjuntos de dados, como os trabalhos apresentados a seguir.

[Claro et al. 2020] aplicaram técnicas de DA para detecção de Leucemia Aguda em imagens de lâminas de sangue. Neste trabalho, os autores desenvolveram CNNs e aplicaram em um conjunto de imagens heterogêneas, utilizando rotação, translação, zoom, *Flipping* e *shear* como técnicas para aumento de dados. O resultado obtido foi de 97,18% de acurácia.

No trabalho de [Zhang et al. 2019], os autores utilizaram DA com três tipos de métodos: rotação de imagem, correção *gamma* e *noise injection* ou injeção de ruído. Nesse trabalho foi utilizado uma CNN como classificador e o *dataset* utilizado era composto por imagens para classificação de frutas. O melhor resultado encontrado foi 94,94%, sendo aproximadamente 5% a mais que os resultados encontrados na literatura.

Os autores [Taylor and Nitschke 2017] propuseram um estudo comparativo entre os esquemas populares de DA para indicar qual é a técnica mais indicada para tal base de dados. Foram testadas técnicas geométricas (ex. corte, rotações) e fotométricas (ex. *color jittering, fancy* PCA) sendo avaliadas por uma CNN. Os resultados utilizando 4-*fold cross validation* foram reportados com as acurácia top-1 e top-5, indicando que a técnica de corte aumentou os resultados obtidos pela CNN.

A proposta dos autores [Shijie et al. 2017] era aplicar diversas técnicas de DA e após a geração da base aumentada, classificar os dados utilizando a CNN Alexnet e nesse caso, considerou *subsets* das bases de dados CIFAR-10 e da *ImageNet* com 10 classes. As técnicas de DA utilizadas foram: GAN/WGAN, corte, rotação *flipping, shifting*, PCA

Tabela 3.1. Resumo dos Trabalhos Relacionados.

Título	Autores e Ano	Base de Dados	Técnica de DA
Convolution Neural Network Models for Acute Leukemia Diagnosis	[Claro et al. 2020]	Imagens de Lâminas de Sangue	Rotação, Translação, Zoom, *Flipping* e *Shear*
Image based fruit category classification by 13-layer deep convolutional neural network and data augmentation	[Zhang et al. 2019]	Classificação de Frutas	Rotação, Correção de *Gamma* e *Noise injection*
Improving Deep Learning using Generic Data Augmentation	[Taylor and Nitschke 2017]	Caltech 101; Pascal VOC	Comparação manual de técnicas fotométricas e geométricas
Research on Data Augmentation for Image Classification Based on Convolutional Neural Networks	[Shijie et al. 2017]	Subsets da CIFAR-10 e ImageNet	Comparação DA tradicional e GANs
Synthetic Data Augmentation Using GAN for Improved Liver Lesion Classification	[Frid-Adar et al. 2018]	Tomografia do Fígado	Comparação DA tradicional e GANs para balancear *datasets*

jittering, color jittering, noise, e combinações desses métodos. Os autores concluíram que as técnicas de corte, *flipping*, WGAN e rotação obtiveram melhores resultados em relação às outras técnicas e que combinações de métodos de DA podem ajudar a se obter melhores resultados que os individuais.

Já em [Frid-Adar et al. 2018], os autores propuseram a utilização de métodos de DA tradicionais e a aplicação das GANs para gerar dados sintéticos. Os testes foram realizados com uma base de dados limitadas de tomografia computacional composta por 182 imagens de lesões no fígado. Os resultados mostraram uma melhora significativa. Com os métodos clássicos de DA obtiveram 78,6% de sensibilidade e 88,4% de especificidade. Com a adição dos dados gerados pelas GANs, foram obtidos 85,7% de sensibilidade e 92,4% de especificidade.

3.4. Estudo de Caso

A utilização de GAN's para o aumento de dados vem aumentando com o passar dos anos. Esse fato é justificado pelo aumento do poder computacional que proporciona o treinamento dessas arquiteturas. No entanto, a concepção das GAN's ainda é de difícil entendimento por parte dos estudantes. Tendo em vista essa problemática, nesta seção iremos demonstrar o funcionamento de GAN's por meio da implementação de uma DCGAN com o gerador, discriminador e o processo de treinamento e geração das imagens artificiais.

3.4.1. Base de dados

No problema abordado neste capítulo, apresentamos a base MNIST [Deng 2012], uma das bases mais populares encontrada em diversos experimentos na literatura.. Essa base de dados possui imagens contendo dígitos escritos a mão e conta com cerca de 70.000 imagens para treino e teste. A resolução das imagens de entrada é de 28×28. Na Figura 3.24, apresentamos alguns exemplos da base utilizada no experimentos.

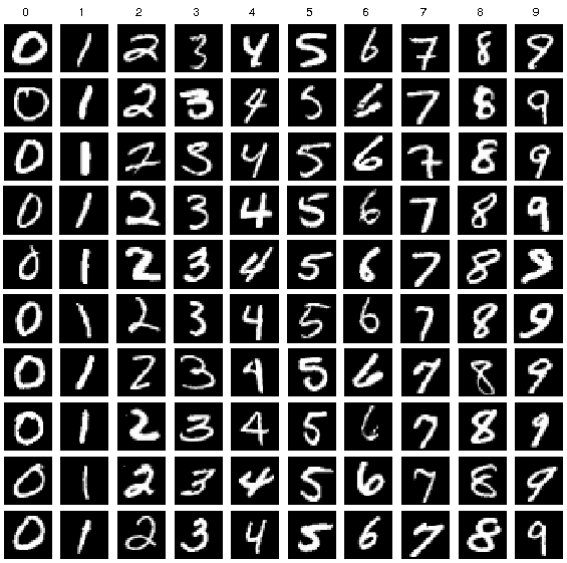

Figura 3.24. Exemplos de imagens provenientes da base de dados MNIST. [Deng 2012]

```
import os
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import numpy as np
from keras import optimizers

batch_size = 64
lr = 0.0003
epochs = 50

(x_treino, _), (x_teste, _) = keras.datasets.mnist.load_data()

all_digits = np.concatenate([x_treino, x_teste])

all_digits = all_digits.astype('float32') / 255

all_digits = np.reshape(all_digits, (-1, 28, 28, 1))

dataset = tf.data.Dataset.from_tensor_slices(all_digits)
dataset = dataset.shuffle(buffer_size=1024).batch(batch_size).prefetch
    (32)
```

3.4.2. Construíndo a GAN

Para construirmos a GAN proposta neste estudo, precisamos definir a arquitetura do gerador e do discriminador. Sendo assim, como entrada, o gerador deve receber um ruído de tamanho 128 que servirá como base para a geração da imagem artificial. A estrutura da arquitetura com os tamanhos dos filtros convolucionais é apresentada no código abaixo.

```

def make_generator_model():

    model = tf.keras.Sequential()
    model.add(layers.Dense(7*7*256, use_bias=False, input_shape=(128,))
    )
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Reshape((7, 7, 256)))
    assert model.output_shape == (None, 7, 7, 256)

    model.add(layers.Conv2DTranspose(128, (5, 5), strides=(1, 1),
    padding='same', use_bias=False))
    assert model.output_shape == (None, 7, 7, 128)
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Conv2DTranspose(64, (5, 5), strides=(2, 2),
    padding='same', use_bias=False))
    assert model.output_shape == (None, 14, 14, 64)
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Conv2DTranspose(1, (5, 5), strides=(2, 2), padding
    ='same', use_bias=False, activation='tanh'))
    assert model.output_shape == (None, 28, 28, 1)

    return model

generator = make_generator_model()
generator.summary()
```

```

def make_discriminator_model():

    model = tf.keras.Sequential()
    model.add(layers.Conv2D(64, (5, 5), strides=(2, 2), padding='same',
                                    input_shape=[28, 28, 1]))
    model.add(layers.LeakyReLU())
    model.add(layers.Dropout(0.3))

    model.add(layers.Conv2D(128, (5, 5), strides=(2, 2), padding='same'
    ))
    model.add(layers.LeakyReLU())
    model.add(layers.Dropout(0.3))

    model.add(layers.Flatten())
    model.add(layers.Dense(1))
```

```
    return model

discriminator = make_discriminator_model()

discriminator.summary()
```

3.4.3. Treinamento

```
class GAN(keras.Model):

    def __init__(self, discriminator, generator, latent_dim):
        super(GAN, self).__init__()
        self.discriminator = discriminator
        self.generator = generator
        self.latent_dim = latent_dim

    def compile(self, d_optimizer, g_optimizer, loss_fn):
        super(GAN, self).compile()
        self.d_optimizer = d_optimizer
        self.g_optimizer = g_optimizer
        self.loss_fn = loss_fn

    def train_step(self, real_images):
        if isinstance(real_images, tuple):
            real_images = real_images[0]
        batch_size = tf.shape(real_images)[0]
        random_latent_vectors = tf.random.normal(shape=(batch_size,
    self.latent_dim))

        generated_images = self.generator(random_latent_vectors)

        combined_images = tf.concat([generated_images, real_images],
    axis=0)

        labels = tf.concat(
            [tf.ones((batch_size, 1)), tf.zeros((batch_size, 1))], axis
    =0
        )

        labels += 0.05 * tf.random.uniform(tf.shape(labels))

        # Treinando o discriminador
        with tf.GradientTape() as tape:
            predictions = self.discriminator(combined_images)
            d_loss = self.loss_fn(labels, predictions)
        grads = tape.gradient(d_loss, self.discriminator.
    trainable_weights)
        self.d_optimizer.apply_gradients(
            zip(grads, self.discriminator.trainable_weights)
        )

        random_latent_vectors = tf.random.normal(shape=(batch_size,
    self.latent_dim))
```

```
        misleading_labels = tf.zeros((batch_size, 1))

        with tf.GradientTape() as tape:
            predictions = self.discriminator(self.generator(random_latent_vectors))
            g_loss = self.loss_fn(misleading_labels, predictions)
        grads = tape.gradient(g_loss, self.generator.trainable_weights)
        self.g_optimizer.apply_gradients(zip(grads, self.generator.trainable_weights))
        return {"d_loss": d_loss, "g_loss": g_loss}
```

```
class GANMonitor(keras.callbacks.Callback):
    def __init__(self, num_img=3, latent_dim=128):
        self.num_img = num_img
        self.latent_dim = latent_dim

    def on_epoch_end(self, epoch, logs=None):
        random_latent_vectors = tf.random.normal(shape=(self.num_img, self.latent_dim))
        generated_images = self.model.generator(random_latent_vectors)
        generated_images *= 255
        generated_images.numpy()
        for i in range(self.num_img):
            img = keras.preprocessing.image.array_to_img(generated_images[i])
            img.save("generated_img_{i}_{epoch}.png".format(i=i, epoch=epoch))

gan = GAN(discriminator=discriminator, generator=generator, latent_dim=latent_dim)

gan.compile(
    d_optimizer = optimizers.Adam(learning_rate=lr),
    g_optimizer = optimizers.Adam(learning_rate=lr),
    loss_fn = keras.losses.BinaryCrossentropy(from_logits=True),
)

gan.fit(
    dataset, epochs=epochs, callbacks=[GANMonitor(num_img=3, latent_dim=latent_dim)]
)
```

Com o treinamento concluído, na Figura 3.25 apresentamos alguns resultados obtidos pela arquitetura proposta. Observamos que em alguns exemplos é notável que o discriminador não conseguiu distinguir todos os exemplos entre reais e falsos. No entanto, em outros conseguimos definir bem qual número está presente na imagem. A quantidade de épocas utilizadas no treino é de fundamental importância na obtenção dos resultados, sendo necessárias inúmeras épocas para treinar apropriadamente uma GAN.

3.5. Considerações Finais

Dos aprimoramentos discutidos, transformações geométricas, transformações de espaço de cores, filtros de kernel, imagens misturadas e *random erasing*, quase todas essas trans-

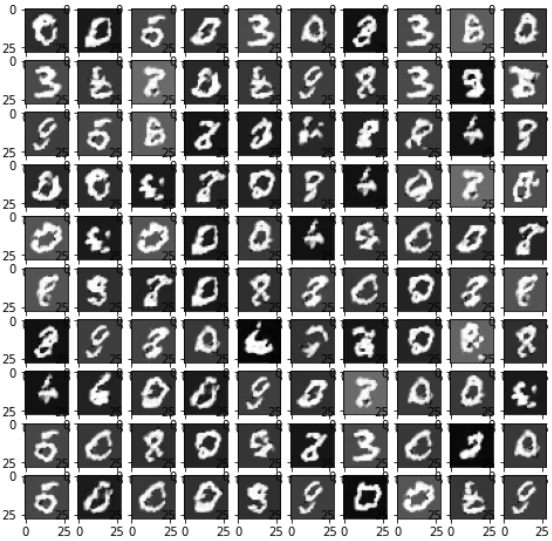

Figura 3.25. Resultado da geração a partir da DCGAN proposta.

formações também vêm com um parâmetro de magnitude de distorção associado. Com uma grande lista de potenciais aumentos e um espaço de magnitudes na maior parte contínuo, é fácil conceituar o enorme tamanho do espaço de pesquisa de aumento. A combinação de aprimoramentos como corte, inversão, mudança de cor e *random erasing* pode resultar em tamanhos de conjuntos de dados massivamente inflados. No entanto, isso não garante que seja vantajoso. Em domínios com dados muito limitados, isso pode resultar em *overfitting*.

Portanto, podemos concluir que DA é importante e pode ser considerado quando é preciso obter mais dados e tentar melhorar os resultados. Para isso, é necessário antes uma análise minuciosa sobre quais técnicas utilizar e ponderar, segundo as vantagens e desvantagens de cada método. Por exemplo, as técnicas de transformações de imagens são muito rápidas, mas é necessário analisar se tais efeitos não podem confundir o classificador. Já com as GANs, os resultados sintéticos podem ser excepcionais, mas necessita de muito processamento e tempo.

Referências

[Acharya and Ray 2005] Acharya, T. and Ray, A. K. (2005). *Image processing: principles and applications*. John Wiley & Sons.

[Aha and Kibler 1991] Aha, D. and Kibler, D. (1991). Instance-based learning algorithms. *Machine Learning*, 6:37–66.

[Batchelor and Waltz 2012] Batchelor, B. and Waltz, F. (2012). *Intelligent machine vision: techniques, implementations and applications*. Springer Science & Business Media.

[Chatfield et al. 2014] Chatfield, K., Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*.

[Claro et al. 2020] Claro, M., Vogado, L., Veras, R., Santana, A., Tavares, J., Santos, J., and Machado, V. (2020). Convolution neural network models for acute leukemia diagnosis. In *The 27th International Conference on Systems, Signals and Image Processing (IWSSIP 2020)*.

[Deng 2012] Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142.

[DeVries and Taylor 2017] DeVries, T. and Taylor, G. W. (2017). Dataset augmentation in feature space.

[Frid-Adar et al. 2018] Frid-Adar, M., Klang, E., Amitai, M., Goldberger, J., and Greenspan, H. (2018). Synthetic data augmentation using gan for improved liver lesion classification. In *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*, pages 289–293. IEEE.

[Gatys et al. 2015] Gatys, L. A., Ecker, A. S., and Bethge, M. (2015). A neural algorithm of artistic style.

[Gomes and Velho 1997] Gomes, J. and Velho, L. (1997). *Image processing for computer graphics*. Springer Science & Business Media.

[Gonzalez and Woods 2000] Gonzalez, R. C. and Woods, R. E. (2000). *Processamento de imagens digitais*. Editora Blucher.

[Heljakka et al. 2018] Heljakka, A., Solin, A., and Kannala, J. (2018). Pioneer networks: Progressively growing generative autoencoder. In *Asian Conference on Computer Vision*, pages 22–38. Springer.

[Hernández-García and König 2018] Hernández-García, A. and König, P. (2018). Further advantages of data augmentation on convolutional neural networks. In *International Conference on Artificial Neural Networks*, pages 95–103. Springer.

[Inoue 2018] Inoue, H. (2018). Data augmentation by pairing samples for images classification. *arXiv preprint arXiv:1801.02929*.

[Kang et al. 2017] Kang, G., Dong, X., Zheng, L., and Yang, Y. (2017). Patchshuffle regularization. *arXiv preprint arXiv:1707.07103*.

[Marques Filho and Neto 1999] Marques Filho, O. and Neto, H. V. (1999). *Processamento digital de imagens*. Brasport.

[Moosavi-Dezfooli et al. 2015] Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. (2015). Deepfool: a simple and accurate method to fool deep neural networks.

[Moreno-Barea et al. 2018] Moreno-Barea, F. J., Strazzera, F., Jerez, J. M., Urda, D., and Franco, L. (2018). Forward noise adjustment scheme for data augmentation. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 728–734. IEEE.

[Nitesh VC 2002] Nitesh VC, Kevin WB, L. O. K. W. (2002). Smote: synthetic minority over-sampling technique. In *Journal of Artificial Intelligence Research*, page 321–357.

[Pascal and Hugo 2010] Pascal, V. and Hugo, L. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408.

[Penatti et al. 2015] Penatti, O. A., Nogueira, K., and Dos Santos, J. A. (2015). Do deep features generalize from everyday objects to remote sensing and aerial scenes domains? In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 44–51.

[Radford et al. 2015] Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks.

[Shijie et al. 2017] Shijie, J., Ping, W., Peiyi, J., and Siping, H. (2017). Research on data augmentation for image classification based on convolution neural networks. In *2017 Chinese automation congress (CAC)*, pages 4165–4170. IEEE.

[Shorten and Khoshgoftaar 2019] Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60.

[Summers and Dinneen 2019] Summers, C. and Dinneen, M. J. (2019). Improved mixed-example data augmentation. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1262–1270. IEEE.

[Takahashi et al. 2019] Takahashi, R., Matsubara, T., and Uehara, K. (2019). Data augmentation using random image cropping and patching for deep cnns. *IEEE Transactions on Circuits and Systems for Video Technology*.

[Taylor and Nitschke 2017] Taylor, L. and Nitschke, G. (2017). Improving deep learning using generic data augmentation. *arXiv preprint arXiv:1708.06020*.

[Taylor and Nitschke 2018] Taylor, L. and Nitschke, G. (2018). Improving deep learning with generic data augmentation. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1542–1547. IEEE.

[Vargas et al. 2016] Vargas, A. C. G., Paes, A., and Vasconcelos, C. N. (2016). Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres. In *Proceedings of the xxix conference on graphics, patterns and images*, volume 1.

[Zhang et al. 2019] Zhang, Y.-D., Dong, Z., Chen, X., Jia, W., Du, S., Muhammad, K., and Wang, S.-H. (2019). Image based fruit category classification by 13-layer deep convolutional neural network and data augmentation. *Multimedia Tools and Applications*, 78(3):3613–3632.

[Zhong et al. 2020] Zhong, Z., Zheng, L., Kang, G., Li, S., and Yang, Y. (2020). Random erasing data augmentation. In *AAAI*, pages 13001–13008.

[Zhu et al. 2017] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks.