Um Cache de Imagens Urbanas Auxiliado por Redes Adversárias Generativas

Guilherme B. Souza, Roberto G. Pacheco, Rodrigo S. Couto *

[1]Universidade Federal do Rio de Janeiro - PEE/COPPE/GTA

{bergman, pacheco, rodrigo}@gta.ufrj.br

Abstract. *Several smart city applications use the cloud to classify captured images. However, Internet's network delay can be prohibitive for such applications. Thus, installing an image cache on the Internet's edge to avoid the cloud reduces the delay. These caches are sensitive to the different lighting conditions of the images. It reduces their hit rate in urban scenarios, where the illumination changes during the day. This work proposes to use a generative adversarial network to change image illumination. This approach allows the cache to have the same image with different illuminations. The results show that the proposal achieves a higher hit rate than a standard cache, reducing the classification delay.*

Resumo. *Diversas aplicações em cidades inteligentes utilizam a nuvem para classificar imagens capturadas. Contudo, o atraso de rede da Internet pode ser proibitivo para tais aplicações. Para evitar a nuvem, é possível instalar um cache de imagens na borda da Internet, reduzindo o atraso. Esses caches são sensíveis às diferentes condições de iluminação das imagens. Isso reduz sua taxa de acerto em cenários urbanos, nos quais a iluminação se altera ao longo do dia. Este trabalho propõe utilizar uma rede adversária generativa para alterar a iluminação das imagens. Isso permite que o cache possua uma mesma imagem com diferentes iluminações. Os resultados mostram que a proposta atinge uma alta taxa de acerto em comparação ao cache comum, reduzindo o atraso da classificação.*

1. Introdução

As câmeras em dispositivos móveis podem estender a compreensão do ambiente ao redor dos seus usuários, permitindo aplicações dentro do contexto de cidades inteligentes. Por exemplo, aplicações de *smartphones*, como o *Google Lens*[1], são capazes de identificar objetos ou pontos turísticos a partir das imagens da câmera. Como base na identificação, o *Google Lens* retorna ao usuário informações detalhadas sobre os pontos de interesse. Nesse caso, para um dado ponto turístico, a aplicação retorna o nome e uma breve descrição histórica daquele local. Outro exemplo é o uso de câmeras em veículos inteligentes, que aproveitam informações visuais para ajudar na tomada de decisões [Bechtel et al., 2018, Kim et al., 2016]. Entretanto, o processamento de imagens demanda uma alta capacidade computacional e alto consumo de energia, que normalmente não estão disponíveis em dispositivos móveis [Cuervo et al., 2010].

*O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior Brasil (CAPES) - Código de Financiamento 001. O trabalho também foi financiado pelo CNPq, pela FAPERJ com os auxílios E-26/203.211/2017 e E-26/211.144/2019, e pela Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), auxílio no. 2015/24494-8.

[1]https://lens.google.com

A computação em nuvem é uma das soluções mais comuns para o problema da escassez de recursos em dispositivos móveis. Nessa abordagem, os dispositivos finais enviam requisições para um servidor com alta capacidade de processamento e de memória. O servidor, por sua vez, realiza a inferência e retorna o resultado da requisição para o dispositivo. Contudo, muitas aplicações são de natureza interativa. Isso significa que os resultados das requisições precisam ser obtidos com um baixo tempo de resposta para poderem ser aproveitados. No caso de aplicações em veículos inteligentes, há um intervalo de tempo limitado no qual as decisões devem ser tomadas [Bechtel et al., 2018]. Uma aplicação de reconhecimento de objetos é um outro exemplo, na qual espera-se que as informações sejam retornadas ao usuário enquanto ele ainda observa o mesmo alvo [Drolia et al., 2017b]. Portanto, por serem sensíveis à latência, essas aplicações podem ser inviabilizadas pelo aumento no tempo de resposta causado pela comunicação com um servidor remoto.

Soluções de computação na borda visam reduzir problemas causados pela nuvem, como o alto tempo de reposta [Satyanarayanan, 2017]. Nessas soluções, utilizam-se recursos computacionais na borda da Internet, como em pontos de acesso IEEE 802.11 e estações rádio base. Assim, objetiva-se trazer esses recursos para regiões geograficamente mais próximas dos usuários.

Caches de imagens são exemplos de soluções de computação na borda, buscando reduzir o tempo de resposta e o uso da rede. Trabalhos como o Cachier [Drolia et al., 2017b] e o FoggyCache [Guo et al., 2018] sugerem sistemas de cache de imagens na borda. Esses sistemas atuam como intermediários de requisições de vários dispositivos antes de serem transferidas para a nuvem. As requisições podem ser, por exemplo, de informações sobre um determinado local fotografado pelo usuário ou de reconhecimento de um objeto. Cada nova requisição recebida na borda é comparada com as imagens armazenadas no cache. Se o sistema identificar que uma das imagens é suficientemente semelhante à da requisição, retorna as informações armazenadas localmente para aquela imagem. Assim, evita-se a transferência para a nuvem e o sistema consegue reduzir o tempo de resposta da aplicação. Caso não seja possível reaproveitar as respostas armazenadas, a borda encaminha a requisição à nuvem. A resposta recebida da nuvem é enviada ao dispositivo final e pode ser armazenada no cache para uso em futuras requisições.

Para realizar as comparações entre as imagens das requisições e as imagens armazenadas no cache, utilizam-se extratores de características, como o ORB (*Oriented FAST and Rotated BRIEF*) [Rublee et al., 2011] e o SIFT (*Scale-invariant feature transform*) [Lowe, 1999]. Esses métodos detectam pontos notáveis (*keypoints*) presentes nas imagens e geram vetores capazes de representá-los. Esses vetores, conhecidos como descritores, podem ser comparados entre si para a obtenção de uma métrica de proximidade entre dois pontos diferentes. Os sistemas de cache utilizam essa comparação para determinar o grau de semelhança entre duas imagens, verificando se a imagem da requisição é semelhante a alguma imagem armazenada.

Os extratores de características apresentam baixo desempenho ao comparar imagens de um mesmo local ou objeto em condições de iluminação muito distintas [Sattler et al., 2018]. Assim, duas imagens de um mesmo elemento podem ser consideradas como diferentes pelo cache, se forem capturadas com diferentes condições de iluminação. Por exemplo, um cache de imagens urbanas que possui imagens capturadas à noite não é eficaz ao receber requisições de imagens diurnas. Consequentemente, a borda

pode desnecessariamente enviar uma requisição à nuvem, mesmo quando o cache possui as informações do local de interesse.

Este trabalho propõe uma solução para tornar um cache robusto às condições de iluminação das imagens. Mais especificamente, considera-se um cenário urbano no qual imagens noturnas estão no cache e deseja-se realizar requisições com imagens diurnas. Esse cenário é típico de aplicações de veículos inteligentes, nas quais as requisições com imagens de ambientes externos podem ser geradas a qualquer momento do dia [Bechtel et al., 2018]. A robustez é alcançada por uma técnica de tradução de imagens, que transforma imagens noturnas em imagens diurnas. O resultado dessa transformação é armazenado no cache, aumentando a chance da requisição ser atendida na borda.

A tradução de imagens deste trabalho utiliza redes adversárias generativas (GANs - *Generative Adversarial Networks*) [Goodfellow et al., 2014]. As GANs são redes neurais que, no contexto deste trabalho, são utilizadas para gerar imagens artificiais, alterando as condições de iluminação da imagem original. Na análise deste trabalho, utiliza-se um *dataset* da literatura com imagens capturadas por veículos e comparam-se dois caches quando submetidos a requisições de imagens diurnas. Um cache contém imagens originalmente noturnas. O outro cache contém imagens diurnas artificialmente geradas a partir dessas imagens noturnas.

Os resultados mostram que o armazenamento de imagens artificiais reduz a quantidade de imagens enviadas à nuvem, reduzindo a latência da requisição. Dessa forma, a técnica é capaz de melhorar a qualidade do serviço oferecido pelas aplicações, e tornar a experiência dos seus clientes mais satisfatória. Além disso, o uso de soluções de cache como esta evita uma sobrecarga desnecessária na rede, o que beneficia outras aplicações no contexto de cidades inteligentes.

Este trabalho está organizado da seguinte forma. A Seção 2 apresenta uma breve revisão da literatura relacionada a este trabalho. Já a Seção 3 descreve o funcionamento de um cache de imagens. A Seção 4 explica os conceitos básicos do uso de GANs na geração de imagens artificiais, enquanto a Seção 5 detalha o *dataset* utilizado. Em seguida, a Seção 6 avalia o desempenho da solução proposta. Por fim, a Seção 7 apresenta conclusões e trabalhos futuros.

2. Trabalhos relacionados

Diversos trabalhos aproveitam a localidade de aplicações em dispositivos móveis e propõem sistemas de cache na borda da internet. Essa localidade pode ser observada, por exemplo, em aplicações de reconhecimento de imagens, nas quais usuários geograficamente próximos frequentemente realizam requisições dos mesmos objetos e pontos turísticos. O Cachier [Drolia et al., 2017b] é um cache de imagens que utiliza o extrator ORB para obter características das imagens das requisições. Os descritores dessas características são usados para treinar um modelo de classificação, que é usado para identificar imagens que contém objetos em comum. O Cachier consegue reaproveitar requisições, e obtém tempo de resposta médio até três vezes menor em relação ao uso exclusivo da nuvem. O Precog [Drolia et al., 2017a] é uma extensão do Cachier, no qual partes do cache são espalhadas na borda e nos próprios dispositivos móveis. Essa solução utiliza um método de predição de dados para armazenar previamente no cache do dispositivo móvel uma parcela do conteúdo da borda. Dessa forma, esse sistema consegue obter uma latência

ainda menor do que o Cachier, pois pode aproveitar com eficiência o resultado armazenado no próprio dispositivo.

O FoggyCache [Guo et al., 2018] é uma solução que pode ser utilizada para cache de imagens ou de outros tipos de conteúdo. Essa proposta é um sistema capaz de reutilizar resultados de diversos tipos de redes neurais profundas (DNNs - *Deep Neural Networks*). Para isso, utilizam-se métodos consolidados na literatura que codificam as entradas da DNN em vetores de alta dimensão. Esses vetores são armazenados no cache, e podem ser comparados com novas entradas para identificar requisições semelhantes. Para isso, é proposto um método de comparação desses vetores, que permite um ajuste fino entre a precisão e a taxa de acerto do cache. Além disso, os autores propõem também um algoritmo que otimiza o tempo de busca no cache quando a distribuição de requisições é desconhecida. Já o Shadow Puppets [Venugopal et al., 2018] difere dos outros caches de imagens na forma como as características das imagens são extraídas. Ao invés de utilizar os extratores de atributos tradicionais, como o ORB e o SIFT, o Shadow Puppets sugere o uso de uma rede neural compacta que funciona como uma função *hash*. Essa rede neural gera uma codificação dos dados de entrada do sistema, que pode ser usada para comparações de forma similar aos descritores de características. O argumento usado para essa proposta é a capacidade das redes neurais de identificar similaridades entre imagens de forma mais acurada que os extratores tradicionais. Consequentemente, as redes neurais utilizadas podem aumentar a taxa de acerto do cache.

Os trabalhos citados focam técnicas e modelos matemáticos que otimizam o tempo de processamento das etapas que compõem o funcionamento do cache. Ao invés disso, este trabalho lida com o problema da redução na taxa de acerto de um sistema de cache de imagens ao receber consultas com condições de iluminação muito distintas. O cache desenvolvido neste trabalho é baseado no Cachier, utilizando um processo similar de reaproveitamento de imagens, incluindo os algoritmos usados para extrair, armazenar e comparar os descritores. Entretanto, os resultados deste trabalho são aplicáveis a qualquer cache que atue com comparação de imagens. Isso ocorre pois sua proposta é adaptar a iluminação das imagens do cache com geração de imagens artificiais. Essa proposta pode ser realizada de forma independente dos métodos de comparação de imagens.

Para gerar as imagens do cache, o método proposto neste trabalho utiliza GANs para transformar a iluminação da imagem original. O uso de GANs em aplicações de visão computacional é um tema vastamente explorado na literatura. Essa junção é responsável por importantes avanços em áreas como restauração de imagens [Yeh et al., 2017] e ampliação artificial de conjuntos de treino [Shrivastava et al., 2017]. Mais especificamente, modelos estruturados a partir de GANs são utilizados para auxiliar aplicações que sofram problemas de adaptação de domínio [Tzeng et al., 2017]. Neste trabalho, os resultados mostram que as GANs são capazes de melhorar o desempenho de um sistema de cache que possui imagens de treino de um domínio (isto é, imagens noturnas), e é testado com imagens pertencentes a outro domínio (isto é, imagens diurnas).

3. Funcionamento do cache

Os sistemas de cache de imagens funcionam de forma semelhante a um web cache, como os utilizados por soluções de *proxy* HTTP (*HyperText Transfer Protocol*) [Eiras et al., 2018]. Em um web cache, armazenam-se os conteúdos de páginas web

requisitadas recentemente por diversos clientes de uma rede. Cada conteúdo armazenado no cache é associado à sua URL (*Uniform Resource Locators*). Para verificar se um conteúdo solicitado está disponível localmente, o *proxy* busca a URL da requisição no seu cache. Caso encontre, o *proxy* envia o conteúdo ao cliente, sem necessidade de comunicação com um servidor na Internet. Diferentemente de um web cache, caches de imagens não realizam uma busca exata de identificadores como a URL. Esses caches calculam métricas de semelhança entre imagens para verificar se o conteúdo de uma requisição está disponível localmente. Isso ocorre pois é pouco provável que diferentes clientes de uma mesma aplicação realizem requisições com imagens idênticas. Por outro lado, requisições com imagens semelhantes podem estar associadas a um mesmo conteúdo. Em uma aplicação de geolocalização, por exemplo, um cliente realiza uma requisição contendo uma foto de um determinado local, e espera que o servidor, ou o cache, retorne a região ou localização exata do local capturado na foto. Duas fotos de um mesmo local, embora diferentes em relação a aspectos como posicionamento da câmera e condições de iluminação, recebem uma mesma resposta da aplicação.

O sistema implementado para este trabalho é baseado no Cachier [Drolia et al., 2017b]. Para verificar a semelhança entre imagens, o sistema emprega um classificador treinado com descritores obtidos pelo extrator ORB. Esse classificador armazena uma lista de descritores extraídos de imagens de requisições anteriores. Armazena-se também o conteúdo da requisição relacionada à imagem da qual cada descritor foi obtido. Neste trabalho, o conteúdo consiste em uma *string* com informações sobre o local da imagem. Ao receber uma nova requisição, o ORB extrai 1.600 características da imagem recebida. Para o descritor de cada característica extraída, procura-se o vizinho mais próximo dentre os descritores armazenados no cache. A busca por vizinhos é realizada pelo algoritmo LSH (*Locality Sensitive Hashing*) [Lv et al., 2007], que visa reduzir o tempo de busca. Tanto a extração de características do ORB quanto a busca de dados com o LSH são implementados em Python 3 [2], por meio da biblioteca de visão computacional OpenCV [3]. O classificador, ao final da sua execução, indica o conteúdo em cache mais semelhante à imagem solicitada. Essa decisão considera a similaridade entre os descritores das imagens em cache e a solicitada pelo usuário.

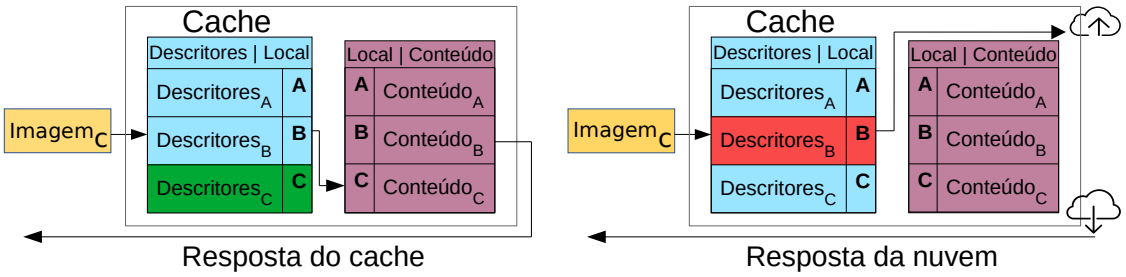
Após a classificação, o sistema verifica o número de descritores da imagem que são semelhantes aos descritores relacionados ao local escolhido. Essa verificação é necessária para decidir se o conteúdo deve ser reaproveitado na requisição. Caso o número de descritores seja superior a um limiar, considera-se que o resultado do classificador está correto e então há um acerto do cache. A Figura 1(a) ilustra essa situação de acerto. Nesse caso, o próprio cache responde à requisição com o conteúdo associado. Por outro lado, caso o número de descritores encontrados seja inferior ao limiar, considera-se que o resultado do classificador está incorreto, e então há uma falha no cache. Em uma situação de falha, a borda encaminha a imagem para nuvem, que retorna o conteúdo ao cliente. Essa situação pode ocorrer em dois casos distintos. O caso mais comum é aquele no qual o conteúdo referente à imagem não está no cache. O outro caso, ilustrado na Figura 1(b), ocorre quando o classificador não consegue associar a imagem ao seu local correto mesmo quando seu conteúdo está no cache. Isso pode ocorrer, por exemplo, quando as condições

[2] http://www.python.org

[3] https://opencv.org

de iluminação da imagem recebida são diferentes da armazenada no cache. Assim, a estratégia do uso de GANs deste trabalho visa reduzir esse último caso de falha.

Idealmente, o limiar do número de descritores deve ser escolhido de forma a separar todas as classificações erradas como falhas e todas as classificações corretas como acertos. Na ausência de um modelo matemático que encontre um limiar ideal, esse valor é variado experimentalmente e escolhido de forma manual na Seção 6.



(a) Acerto do cache. (b) Falha do cache.

Figura 1. Funcionamento de um cache de imagens. Adaptado de [Drolia et al., 2017b].

4. Geração de imagens artificiais com GANs

As Redes Adversárias Generativas (GANs), introduzidas em [Goodfellow et al., 2014], são uma classe de redes neurais que formam um modelo de aprendizado de máquina não supervisionado capaz de gerar imagens artificiais. Esse modelo funciona com base no treinamento simultâneo de duas redes antagônicas. Há uma rede discriminativa $\mathcal{D}$, que aprende a diferenciar imagens reais das imagens artificiais criadas pela outra rede geradora $\mathcal{G}$. A rede $\mathcal{G}$ é treinada para gerar imagens que minimizem as chances de $\mathcal{D}$ diferenciar corretamente. Essa dinâmica, na qual ambas as redes são treinadas para minimizar o acerto da outra, permite com que a rede $\mathcal{G}$ aprenda a gerar imagens cada vez mais realistas. Após o treino, $\mathcal{G}$ pode ser usada para gerar novas imagens aleatórias, seguindo a distribuição do conjunto de treino.

Em vez de gerar imagens aleatoriamente, a CycleGAN [Zhu et al., 2017] propõe estender o modelo das GANs para realizar a tradução de imagens específicas de um domínio para um outro, tarefa conhecida como tradução imagem-para-imagem. Assim como as GANs, a CycleGAN é treinada de forma não supervisionada. O modelo recebe imagens não pareadas de dois domínios, $\mathcal{X}$ e $\mathcal{Y}$, e aprende a realizar a tradução do domínio $\mathcal{X}$ para o $\mathcal{Y}$, e vice-versa. Por exemplo, em [Zhu et al., 2017] a CycleGAN é utilizada para transformar imagens de zebras em imagens de cavalos. Para realizar essa tradução, a CycleGAN consiste em dois pares de redes geradoras e discriminativas. Cada rede geradora realiza a tradução de domínios em um dos dois sentidos possíveis. Cada rede discriminativa é treinada para distinguir imagens artificiais de outras imagens reais de um mesmo domínio. Essas redes são treinadas de forma cíclica, permitindo que uma imagem que seja traduzida para um domínio $\mathcal{Y}$, possa ser traduzida de volta para o domínio $\mathcal{X}$ inicial, de forma a gerar um resultado semelhante à imagem original.

O modelo de tradução de imagens utilizado neste trabalho, a TodayGAN [Anoosheh et al., 2019], tem sua estrutura baseada na CycleGAN e realiza a tradução

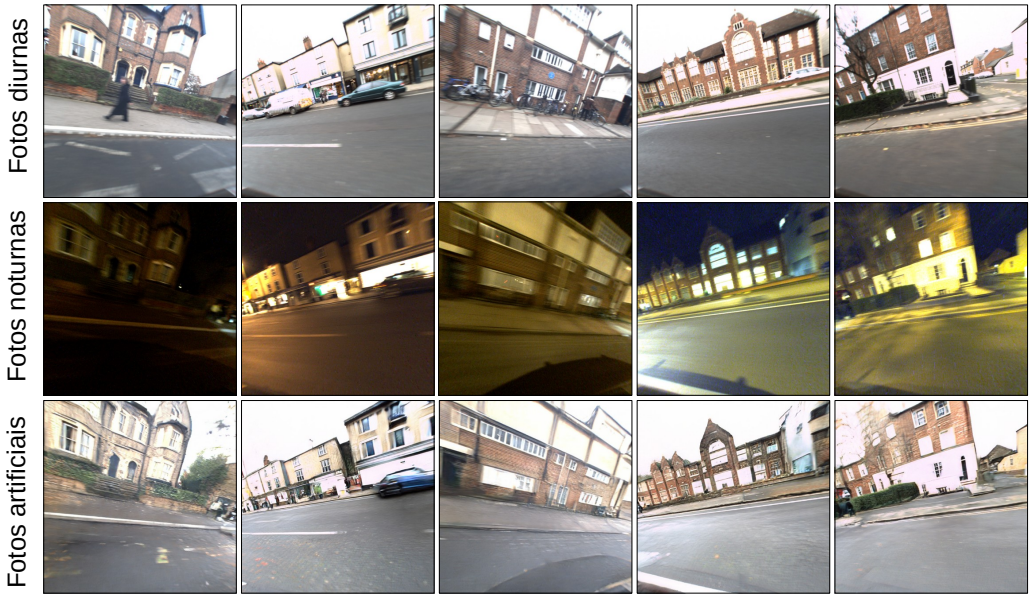


Figura 2. Exemplo de uso da TodayGAN nos experimentos.

entre os domínios de imagens diurnas e noturnas. Esse modelo é treinado com um *dataset* de imagens urbanas, captadas por um veículo autônomo à noite e de dia. Esse *dataset*, que também é utilizado neste trabalho, é detalhado na Seção 5. A Figura 2 mostra um exemplo de uso da TodayGAN no *dataset* empregado. Na primeira linha da figura, estão presentes exemplos de imagens reais do período diurno de cinco locais distintos. Na segunda linha, apresentam-se imagens noturnas e reais dos mesmos locais. A terceira linha apresenta as imagens artificiais diurnas geradas pela TodayGAN, a partir das imagens noturnas da segunda linha. Nota-se que as imagens artificiais conseguem representar os locais no período diurno utilizando imagens noturnas.

A TodayGAN é originalmente proposta para auxiliar na recuperação de imagens similares em um banco de dados, com objetivo de retornar ao usuário a posição geográfica das imagens das requisições. Dada uma imagem de entrada, o sistema retorna a sua localização com base na imagem mais similar encontrada no banco de dados. As imagens de teste utilizadas são noturnas, enquanto as imagens que formam o banco de dados são diurnas. Portanto, antes de realizar a busca, a TodayGAN realiza a transformação das imagens imagens noturnas em imagens diurnas para obter um melhor desempenho. Este trabalho adapta a TodayGAN para o uso em uma tarefa semelhante de geolocalização, porém com uma camada adicional de cache. O cache funciona de forma similar ao modelo de recuperação de imagens usado para avaliar a TodayGAN em [Anoosheh et al., 2019]. Nesse modelo, utiliza-se o extrator DenseVLAD [Torii et al., 2015]. Já neste trabalho, o extrator ORB é utilizado por ser mais simples e mais rápido. Além disso, o cache possui um número de imagens armazenadas menor em relação ao banco de dados do modelo de recuperação. Essas diferenças são motivadas pela necessidade de manter um tempo de processamento pequeno no cache. A TodayGAN aprende as traduções de imagens noturnas para diurnas e vice-versa. Entretanto, este trabalho segue a mesma metodologia usada em [Anoosheh et al., 2019], considerando a tradução em apenas um sentido. Ou seja, transformam-se imagens noturnas em imagens diurnas artificiais. Neste trabalho são

traduzidas as imagens armazenadas no cache, ao invés de traduzir em tempo real as imagens que serão buscadas.

5. Dataset

Este trabalho utiliza um conjunto de imagens selecionadas do *dataset* Oxford RobotCar [Maddern et al., 2017]. O treinamento da TodayGAN utiliza um conjunto de imagens desse mesmo *dataset*. Embora originem do mesmo *dataset*, as imagens utilizadas no treino da TodayGAN não fazem parte do conjunto de teste deste trabalho.

O *dataset* Oxford RobotCar é composto de quadros de vários vídeos gravados em um mesmo trajeto na cidade de Oxford, Inglaterra. Os quadros são coletados por um veículo autônomo por meio de câmeras acopladas nos lados esquerdo e direito, e na traseira do carro. O veículo repete o mesmo trajeto durante vários dias e em diferentes horários, gravando assim diferentes condições de clima e iluminação.

Os experimentos deste trabalho são realizados para uma aplicação urbana que, ao receber uma nova requisição, retorna ao usuário a identificação do local da imagem dessa requisição. Contudo, o cache proposto pode ser utilizado também para outras aplicações em dispositivos móveis e sensores que visam extrair informações do ambiente ao seu redor. Para analisar o funcionamento do cache para essa aplicação, é necessário que as imagens do conjunto de teste estejam agrupadas por local. Além disso, as imagens de um mesmo local precisam ser visualmente semelhantes entre si. Com isso, o cache pode reaproveitar o resultado de uma imagem quando existe outra do mesmo local armazenada. Portanto, este trabalho opta por utilizar apenas uma fração do Oxford RobotCar, constituída de imagens selecionadas manualmente. Para construir o conjunto de imagens usado neste trabalho, selecionam-se 70 locais presentes nas filmagens. As imagens selecionadas possuem fachadas de edifícios, sem obstáculos na frente das câmeras nos momentos das gravações. É importante ressaltar que escolhem-se diferentes locais ao longo da trajetória do veículo, evitando que os resultados estejam relacionados a apenas uma região específica da cidade. Além disso, utilizam-se gravações realizadas no outono[4], devido ao número reduzido de folhagens das árvores que atrapalham a visibilidade das cenas. Da mesma forma, este trabalho utiliza imagens de dias nublados, evitando o brilho forte do sol que satura as lentes das câmeras. A seleção manual de imagens visa tornar mais simples a validação da proposta deste trabalho, focando o uso de GANs em caches. Entretanto, uma aplicação real de visão computacional deve possuir outros mecanismos para lidar com adversidades da captura de imagens, como a presença de obstáculos.

Para cada local escolhido, utilizam-se dez imagens reais noturnas e dez imagens reais diurnas. Cada um dos dois conjuntos de dez imagens representa fotografias de um local capturadas em sequência. As imagens noturnas são utilizadas para gerar dez imagens diurnas artificiais com a TodayGAN. Assim, cada local possui 30 imagens, sendo dez diurnas, dez noturnas e dez diurnas artificiais. Como há 70 locais, o conjunto deste trabalho utiliza 2.100 imagens. Como exemplificado na Figura 2, escolhem-se as imagens diurnas mais semelhantes possíveis às imagens capturadas à noite em relação à posição e ao ângulo das fotos. Assim, outros efeitos sobre os resultados são minimizados, exceto a diferença de iluminação entre as imagens, que é a característica analisada neste trabalho.

[4]Como o *dataset* é rotulado com as datas de captura das imagens, é possível conhecer a estação do ano, verificando-se a data na qual a imagem foi obtida.

6. Análise experimental

Os experimentos utilizam um conjunto de teste com todas as 700 imagens diurnas reais selecionadas do *dataset*. Essas imagens são divididas em sete grupos de cem imagens. Um grupo possui imagens de 10 dos 70 locais possíveis. Em um grupo, cada local possui 10 imagens. Utiliza-se cada uma das imagens como entrada do modelo de classificação do cache. À exceção do gráfico de dispersão da Seção 6.2, todos os experimentos consistem em uma média dos sete grupos, com intervalos de confiança de 95%.

O conjunto de treino do classificador consiste nas imagens armazenadas no cache. Essas imagens são usadas para gerar o modelo de busca de vizinhos próximos do LSH. Dessa forma, para cada imagem de teste, buscam-se as imagens de treino mais semelhantes. O classificador é treinado com um número fixo de imagens para cada local que se deseja armazenar no cache. Os experimentos avaliam dois conjuntos de treino, obtidos seguindo a metodologia da Seção 5. No primeiro, o classificador do cache é treinado apenas com imagens reais noturnas. No segundo, utilizam-se apenas imagens diurnas artificiais geradas pela TodayGAN. Todas as imagens artificiais são geradas previamente, utilizando o código da TodayGAN[5] no Google Colaboratory[6].

Os primeiros experimentos deste trabalho têm como objetivo analisar a acurácia do classificador, selecionando os parâmetros mais adequados para o cache. Em seguida, realizam-se experimentos para avaliar a taxa de acerto da proposta e escolher o limiar do número de descritores. Por fim, os parâmetros escolhidos são mantidos fixos e analisa-se a eficácia do uso de GANs na redução da latência do cache. O cache utilizado na análise executa em uma máquina virtual do VirtualBox[7] com Ubuntu 20.4, 8GB de RAM e quatro CPUs virtuais de uma CPU AMD Ryzen 7 1700X.

6.1. Acurácia do classificador

A seguir, analisa-se a acurácia do classificador, desconsiderando a verificação do limiar do número de descritores. Ou seja, a ideia é verificar apenas o desempenho do ORB e do LSH em relação às condições de iluminação.

6.1.1. Impacto do número de imagens no treino

O conjunto de treino deste experimento contém apenas imagens dos mesmos locais referentes às imagens de teste. Como são sete grupos de teste, cada um contendo 10 locais, o conjunto de treino para cada grupo terá presente apenas imagens desses mesmos 10 locais. É importante enfatizar que as imagens são sempre diferentes entre si, mesmo quando pertencem a um mesmo local. Neste experimento, varia-se a quantidade de imagens de treino por local, sendo todos os locais treinados com o mesmo número de imagens. Em uma aplicação real, essa quantidade pode ser, por exemplo, o número mínimo de imagens que a borda precisa receber para armazenar o local no cache. Para cada grupo, obtém-se a acurácia do classificador, definida como a porcentagem do número de classificações corretas em relação ao total de classificações realizadas. Uma classificação correta é aquela que indica o local esperado para a imagem testada. A Figura 3 apresenta os resultados de

[5]https://github.com/AAnoosheh/ToDayGAN

[6]https://colab.research.google.com/

[7]https://www.virtualbox.org/

acurácia em função da quantidade de imagens de treino por local. Os resultados identificados como “Noite” representam o classificador treinado com imagens noturnas. Nota-se que esse cenário obtém baixo desempenho, corroborando a motivação deste trabalho. Por outro lado, o uso de imagens da TodayGAN, identificado como “GAN” na Figura 3, mostra-se eficaz em melhorar o desempenho da classificação.

A Figura 3 também mostra que a melhoria pela TodayGAN ocorre independentemente do número de imagens por local. Assim, mesmo com poucas imagens de treino, é possível tornar o classificador mais robusto às diferentes condições de iluminação. Os próximos experimentos utilizam um valor fixo de dez imagens de treino por local, pois essa é a configuração que apresenta o maior valor médio[8] de acurácia da Figura 3.

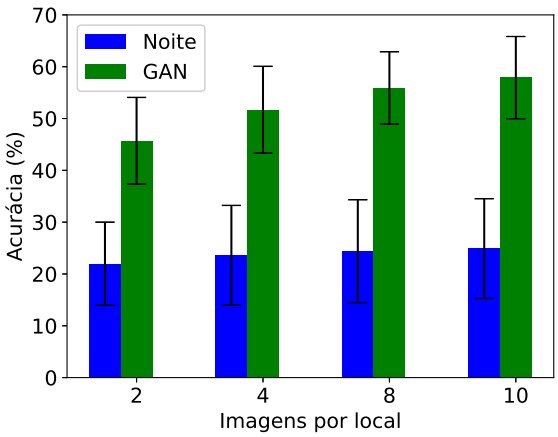
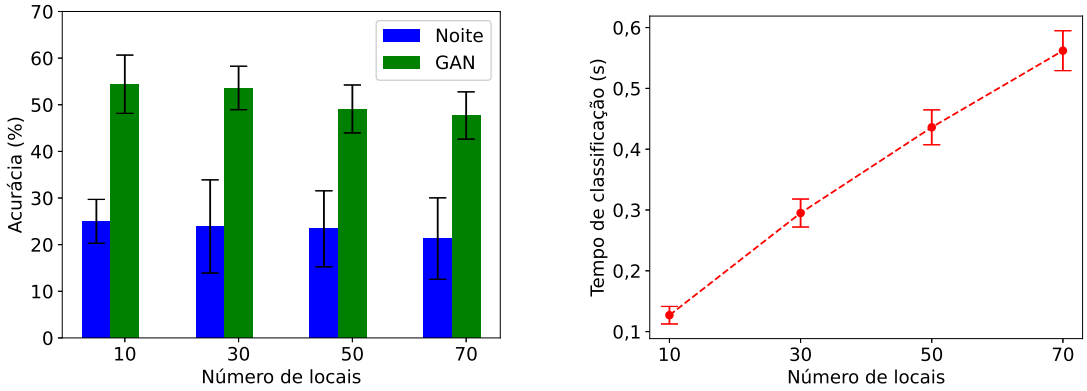


Figura 3. Acurácia para diferentes quantidades de imagens de treino por local.

6.1.2. Impacto do número de locais no cache

O segundo parâmetro analisado é o número de locais que estão armazenados no cache. No experimento da Seção 6.1.1, o conjunto de treino possui apenas imagens dos mesmos locais presentes nas imagens de teste. Dessa vez, o conjunto de treino possui um número de locais igual ou superior ao do conjunto de teste. O objetivo é verificar a escalabilidade da busca no cache para números maiores de locais armazenados. As imagens dos dez locais de interesse são sempre mantidas e adicionam-se ao cache imagens de locais distintos, que não auxiliam na classificação dos grupos de teste. A Figura 4(a) apresenta a acurácia em função do número de locais diferentes presentes no cache. Os resultados mostram que o classificador treinado com imagens da TodayGAN atinge desempenho superior àquele treinado com imagens noturnas para todos os números de locais testados. A Figura 4(b) mostra o tempo de classificação médio de uma imagem em relação ao número de locais, quando o cache é treinado apenas com imagens da TodayGAN. É possível notar que esse tempo aumenta de forma aproximadamente linear. Isso ocorre pois a inclusão de mais locais adiciona mais entradas ao classificador, aumentando o tempo de busca por vizinhos. Esse resultado mostra que deve-se armazenar poucos locais no cache para manter um baixo tempo de classificação.

[8]Utiliza-se o valor médio apenas como heurística de escolha do parâmetro e não como fator comparativo entre diferentes quantidades de imagens por local, visto que os intervalos de confiança se sobrepõem.



(a) Acurácia. (b) Tempo médio de classificação.

Figura 4. Desempenho para diferentes números de locais armazenados.

Em uma aplicação real, a solução deste trabalho pode exigir que o classificador tenha uma entrada de imagem artificial para cada imagem original armazenada. Nesse caso, o uso de GANs necessita do dobro de entradas no classificador em comparação com um cache de imagens comum. Isso aumenta o tempo da classificação, da mesma forma que observado na Figura 4(b). Uma solução para esse problema é utilizar um cache que se adapte ao período do dia. Por exemplo, em um período noturno, a busca considera apenas imagens noturnas, evitando um aumento no tempo de classificação. Além disso, é possível utilizar políticas de substituição, que mantêm no cache apenas os locais mais populares [Drolia et al., 2017b]. Para tornar os próximos experimentos independentes da política de cache, treinam-se os classificadores apenas com as imagens dos dez locais utilizados no teste, como já realizado na Seção 6.1.1.

6.2. Impacto do limiar de número de descritores

Como mencionado na Seção 3, o sistema só considera que o local está no cache quando o número de descritores encontrado é maior do que um limiar. Um alto limiar prejudica a taxa de acerto, que é a porcentagem das requisições que o cache consegue satisfazer. As requisições satisfeitas são aquelas aceitas pelo limiar. Por outro lado, um limiar baixo reduz a precisão do cache. Neste trabalho, considera-se a precisão como a porcentagem de classificações corretas em relação ao total de classificações aceitas pelo limiar. O objetivo deste experimento é avaliar esse compromisso e determinar um limiar aceitável, levando em consideração a precisão e taxa de acerto desejáveis para o *dataset*. Para isso, realiza-se a classificação das imagens de cada um dos sete grupos e o resultado da classificação de cada imagem é comparado com um limiar. Em cada grupo, testam-se diferentes valores de limiar para obter uma larga faixa de valores de acurácia e precisão.

Os resultados para os cenários “Noite” e “GAN” são apresentados no gráfico de dispersão da Figura 5. Cada ponto na figura representa o resultado da precisão e da taxa de acerto para um determinado valor de limiar usando um grupo de imagens. É importante salientar que o caso ideal possui uma taxa de acerto de 100% com uma precisão de 100%, já que todos os locais de teste estão no cache. A Figura 5 mostra que a TodayGAN possui mais pontos no canto superior direito da figura. Isso indica que a TodayGAN pode aumentar a taxa de acerto do sistema sem reduzir significativamente sua precisão. Para

o próximo experimento, escolhe-se, para cada cenário, o limiar que obteve precisão de aproximadamente 90% com a maior taxa de acerto possível.

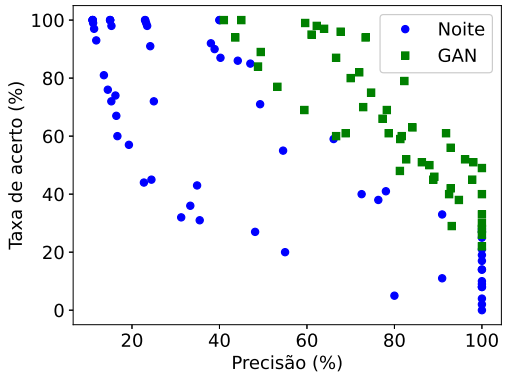


Figura 5. Compromisso entre precisão e taxa de acerto.

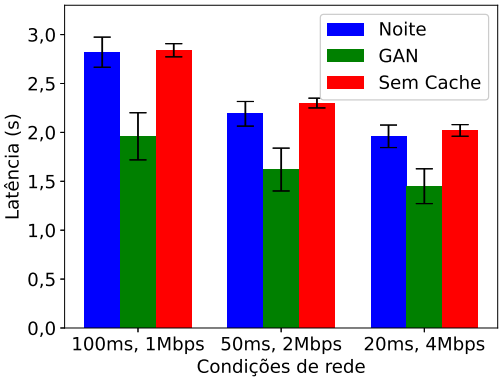


Figura 6. Latência para diferentes condições de rede.

6.3. Latência do cache

Finalmente, analisa-se a eficácia da TodayGAN em reduzir a latência do cache. Para tal, conecta-se a máquina virtual dos experimentos anteriores, que representa a borda, a uma outra máquina virtual que atua como nuvem. Essa última está na mesma máquina física que a borda e possui oito CPUs virtuais e 16GB de RAM. As condições da rede entre essas duas máquinas é emulada pelo TC [9] em conjunto com NETEM [10], utilizando os parâmetros *delay* e *tbf rate*. Essas ferramentas permitem controlar a banda e o atraso de rede entre as duas máquinas e executar o experimento para diferentes condições de rede.

Os experimentos consideram que as requisições começam diretamente no dispositivo na borda. O tempo de comunicação com um dispositivo final é ignorado, pois independe da solução de cache. O principal objetivo deste experimento é comparar a latência das requisições em três cenários: um cache com imagens reais noturnas, um cache com imagens artificiais geradas pela TodayGAN e um cenário sem cache de borda (isto é, com encaminhamento direto para nuvem).

O experimento segue a metodologia dos grupos de imagens da Seção 6.1, utilizando os parâmetros fixos escolhidos nessa seção. Além disso, utiliza-se a classificação com verificação dos limiares selecionados na Seção 6.2. A nuvem responde às falhas do cache utilizando o mesmo classificador. Contudo, esse classificador é treinado com todas as imagens do *dataset* e, portanto, atinge sempre 100% de acurácia, embora apresente um maior tempo de classificação. Para cada grupo de imagens, obtém-se, além da latência média por imagem, a taxa de acerto e a precisão do cache. Os valores de precisão e taxa de acerto, que são independentes das condições de rede, são apresentados na Tabela 1. Os dados da tabela mostram que os limiares são escolhidos de forma a manter uma precisão próxima nos dois cenários e, dessa forma, realizar uma comparação justa. A tabela mostra também que o cache treinado com imagens da TodayGAN consegue, na média, atingir uma taxa de acerto mais de cinco vezes maior que o cache normal quando define-se uma precisão alta. Isso corrobora as conclusões da Seção 6.2.

[9] http://lartc.org/manpages/tc.txt
[10] https://wiki.linuxfoundation.org/networking/netem

Cenário	Precisão	Taxa de acerto
Noite	93,9 $\pm$ 9,26 %	6,14 $\pm$ 4,2 %
GAN	94,8 $\pm$ 6,5 %	36,6 $\pm$ 7,85 %

Tabela 1. Precisão e taxa de acerto para os cenários no teste do cache.

Para a avaliação da latência, testam-se três condições de rede distintas. Cada condição possui um determinado valor de atraso e de banda. A Figura 6 apresenta a latência obtida para cada uma das condições de rede testadas. Os resultados mostram que, para todas as condições de rede testadas, o uso de imagens diurnas artificiais permite reduzir a latência do cache. Além disso, caso o cache possua apenas imagens noturnas, o desempenho se aproxima ao de uma solução sem cache.

7. Conclusões e trabalhos futuros

Este trabalho mostrou que, em um cenário urbano, o classificador de um cache treinado com imagens noturnas apresenta baixo desempenho ao receber imagens diurnas. A proposta deste trabalho é então utilizar redes adversárias generativas (GANs) para gerar imagens artificiais. Assim, é possível gerar imagens a partir de outras já recebidas pelo cache, adequando seu conteúdo a um determinado período do dia. Os resultados obtidos mostraram que o aumento na taxa de acerto pela TodayGAN reduz a latência geral do sistema. Em trabalhos futuros, pretende-se realizar testes para um cache com conteúdo dinâmico, já que neste trabalho manteve-se o conteúdo do cache estático para simplificar a análise e validar a hipótese da melhora de desempenho com a TodayGAN. Planeja-se também verificar se a melhoria de desempenho se mantém para outros *datasets*.

Referências

Anoosheh, A., Sattler, T., Timofte, R., Pollefeys, M. e Van Gool, L. (2019). Night-to-day image translation for retrieval-based localization. Em *IEEE International Conference on Robotics and Automation (ICRA)*, p. 5958–5964.

Bechtel, M. G., McEllhiney, E., Kim, M. e Yun, H. (2018). Deeppicar: A low-cost deep neural network-based autonomous car. Em *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, p. 11–21.

Cuervo, E., Balasubramanian, A., Cho, D.-k., Wolman, A., Saroiu, S., Chandra, R. e Bahl, P. (2010). Maui: making smartphones last longer with code offload. Em *International Conference on Mobile Systems, Applications, and Services (MobiSys)*, p. 49–62.

Drolia, U., Guo, K. e Narasimhan, P. (2017a). Precog: Prefetching for image recognition applications at the edge. Em *ACM/IEEE Symposium on Edge Computing (SEC)*, p. 1–13.

Drolia, U., Guo, K., Tan, J., Gandhi, R. e Narasimhan, P. (2017b). Cachier: Edge-caching for recognition applications. Em *IEEE International Conference on Distributed Computing Systems (ICDCS)*, p. 276–286.

Eiras, R. S. V., Couto, R. S. e Rubinstein, M. G. (2018). Avaliação de desempenho de um proxy HTTP implementado como função virtual de rede. Em *XXXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. e Bengio, Y. (2014). Generative adversarial networks. *arXiv preprint arXiv:1406.2661*.

Guo, P., Hu, B., Li, R. e Hu, W. (2018). FoggyCache: Cross-device approximate computation reuse. Em *International Conference on Mobile Computing and Networking (MobiCom)*, p. 19–34.

Kim, H., Lee, Y., Yim, B., Park, E. e Kim, H. (2016). On-road object detection using deep neural network. Em *IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, p. 1–4.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. Em *IEEE International Conference on Computer Vision (ICCV)*, p. 1150–1157.

Lv, Q., Josephson, W., Wang, Z., Charikar, M. e Li, K. (2007). Multi-probe LSH: efficient indexing for high-dimensional similarity search. Em *International Conference on Very Large Data Bases (VLDB)*, p. 950–961.

Maddern, W., Pascoe, G., Linegar, C. e Newman, P. (2017). 1 year, 1000 km: The oxford robotcar dataset. *The International Journal of Robotics Research*, 36(1):3–15.

Rublee, E., Rabaud, V., Konolige, K. e Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. Em *IEEE International Conference on Computer Vision (ICCV)*, p. 2564–2571.

Sattler, T., Maddern, W., Toft, C., Torii, A., Hammarstrand, L., Stenborg, E., Safari, D., Okutomi, M., Pollefeys, M., Sivic, J. et al. (2018). Benchmarking 6dof outdoor visual localization in changing conditions. Em *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 8601–8610.

Satyanarayanan, M. (2017). The emergence of edge computing. *Computer*, 50(1):30–39.

Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W. e Webb, R. (2017). Learning from simulated and unsupervised images through adversarial training. Em *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 2107–2116.

Torii, A., Arandjelovic, R., Sivic, J., Okutomi, M. e Pajdla, T. (2015). 24/7 place recognition by view synthesis. Em *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 1808–1817.

Tzeng, E., Hoffman, J., Saenko, K. e Darrell, T. (2017). Adversarial discriminative domain adaptation. Em *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 7167–7176.

Venugopal, S., Gazzetti, M., Gkoufas, Y. e Katrinis, K. (2018). Shadow Puppets: Cloud-level accurate AI inference at the speed and economy of edge. Em *USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*.

Yeh, R. A., Chen, C., Yian Lim, T., Schwing, A. G., Hasegawa-Johnson, M. e Do, M. N. (2017). Semantic image inpainting with deep generative models. Em *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 5485–5493.

Zhu, J.-Y., Park, T., Isola, P. e Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. Em *International Conference on Computer Vision (ICCV)*, p. 2223–2232.