**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS6.071  
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA**

**Pós-graduação *Lato Sensu* em Inteligência Artificial e Aprendizado de Máquina**

**André Luís Baptista da Silva**

**USO DE VISÃO COMPUTACIONAL PARA DETECÇÃO DE VAGAS EM ESTACIONAMENTOS**

Macaé/RJ - Brasil

2021

**André Luís Baptista da Silva**

**USO DE VISÃO COMPUTACIONAL PARA DETECÇÃO DE VAGAS EM ESTACIONAMENTOS**

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Inteligência Artificial e Aprendizado de Máquina como requisito parcial à obtenção do título de especialista.

Macaé/RJ - Brasil

2021 **SUMÁRIO**

[1. Introdução 4](#_Toc67349686)

[1.1. Contextualização 4](#_Toc67349687)

[1.2. O problema proposto 5](#_Toc67349688)

[2. Coleta de Dados 6](#_Toc67349689)

[2.1 Estrutura da base de dados CNRPARK+EXT 6](#_Toc67349690)

[2.2 Pré-processamento das imagens 7](#_Toc67349691)

[3. Processamento/Tratamento de Dados 9](#_Toc67349692)

[3.1 Redimensionamento de imagens 9](#_Toc67349693)

[3.2 Criação dos vetores de características com ResNet18 10](#_Toc67349694)

[3.3 Cálculo da similaridade de cosseno 10](#_Toc67349695)

[3.4 Preparando as listas de top-k imagens mais semelhantes 11](#_Toc67349696)

[3.5 Testando a similaridade entre as imagens de exemplo 11](#_Toc67349697)

[3.6 Testando a similaridade entre as imagens de exemplo e as de busca 12](#_Toc67349698)

[4. Análise e Exploração dos Dados 13](#_Toc67349699)

[4.1 Obtendo e pré-processando dados 13](#_Toc67349700)

[4.2 Usando redes neurais convolucionais como geradores de vetores de características 14](#_Toc67349701)

[4.3 Cálculo de similaridades usando diferença dos vetores de características 15](#_Toc67349702)

[4.4 Usando listas de top-k mais semelhantes 17](#_Toc67349703)

[5. Criação de Modelos de Machine Learning 18](#_Toc67349704)

[6. Apresentação dos Resultados 19](#_Toc67349705)

[7. Links 23](#_Toc67349706)

[REFERÊNCIAS 24](#_Toc67349707)

# 1. Introdução

## 1.1. Contextualização

Uma aplicação da visão computacional com inúmeras possibilidades é a busca por imagens semelhantes. Neste trabalho apresento o uso de uma rede neural convolucional (convolutional neural network - CNN) pré-treinada para identificação de vagas vazias e ocupadas em estacionamentos.

Existem muitos artigos e trabalhos sobre este tema, normalmente usando uma abordagem de comparação com imagens de automóveis, buscando características comuns nos veículos, como retrovisores, rodas, curva do farol, reflexo do céu no parabrisa, etc. Inicialmente pensei em usar uma estratégia diferente, buscando características de vagas vazias, com vista em uma menor variabilidade (vagas vazias são sempre mais ou menos iguais, diferenciando-se apenas por alguma condição climática, iluminação, sombra, etc., ao passo que as vagas ocupadas variam muito entre si, devido à imensa quantidade de modelos diferentes de carros, cores, etc.).

Essa estratégia rapidamente se mostrou ineficaz, dado o fato que as vagas vazias possuem poucas características, levando a criação de vetores de features (características) muito pequenos, tornando a comparação muito difícil.

## 1.2. O problema proposto

A automatização de processos em geral é uma tendência mundial e irreversível. Existe enorme demanda por automatização de estacionamentos de diversos tipos, como supermercados, shoppings, empresas, etc. Uma abordagem tradicional é o uso de sensores de presença individuais para cada vaga, com custo e complexidade consideráveis, conforme figura:



No presente trabalho apresentaremos um sistema de identificação de vagas baseado em visão computacional (machine vision), que oferece vantagens importantes em custo e complexidade de instalação e manutenção.

A base de dados em que o trabalho se baseia é o dataset CNRPARK+EXT (AMATO et al., 2015), composto de aproximadamente 150.000 patches (pedaços) de imagens de estacionamentos. Estas imagens foram coletadas entre julho de 2015 e fevereiro de 2016, sob várias condições meteorológicas e ângulos de visão, com diferentes iluminações e oclusões, no estacionamento da CNR Research Area em Pisa, Itália.

O objetivo é comprovar a viabilidade do processo, em termos de acurácia (taxa de acerto em relação à detecção de vagas vazias e ocupadas) e performance (conseguir executar a detecção em curto espaço de tempo).

# 2. Coleta de Dados

## 2.1 Estrutura da base de dados CNRPARK+EXT

O dataset escolhido para análise, CNRPARK+EXT (AMATO et al., 2015) é subdividido em dois: o original (CNR-PARK, menor, com 242 frames, coletado em apenas dois dias de julho de 2015) e o posterior (CNR-EXT, maior, com 4.081 frames, coletado em 23 dias entre novembro de 2015 e fevereiro de 2016, com grande variedade climática). Cada frame é subdividido em vários patches (segmentos de imagem) que representam uma vaga cada.

O subset CNR-PARK é composto por duas câmeras (A e B), e os arquivos seguem o seguinte padrão de nomenclatura:

<CAMERA>/<CLASSE>/YYYYMMDD\_HHMM\_<NUM\_SLOT>.jpg

Onde:

* <CAMERA> pode ser A ou B;
* <CLASSE> pode ser “free” ou “busy”, repectivamente “livre” ou “ocupada”;
* YYYYMMDD\_HHMM\_ representa a data e hora da captura;
* <NUM\_SLOT> é um número identificador da vaga para aquela câmera.

Já o segundo subset, CNR-EXT, é composto por 9 câmeras (1 a 9), e as imagens seguem este padrão de nomenclatura:

PATCHES/<CLIMA>/<DATA\_CAPTURA>/camera<NUM\_CAM>/<W\_ID>\_< DATA\_CAPTURA >\_<HORA\_CAPTURA>\_C0<NUM\_CAM>\_<NUM\_SLOT>.jpg

* <CLIMA> pode ser “sunny”, “overcast” ou “rainy”, respectivamente: “ensolarado”, “nublado” ou “chuvoso”;
* <DATA\_CAPTURA> é a data de captura da imagem no formato YYYY-MM-DD;
* <NUM\_CAM> é o número identificador da câmera, entre 1 e 9;
* <W\_ID> é o identificador do tipo de clima, “S”, “O” ou “R”;
* <HORA\_CAPTURA> é a hora da captura da imagem no formato HH.MM, com horas de 0 a 23;
* <NUM\_SLOT> é o identificador global de cada vaga, pode ser usado para identificar unicamante daterminada vaga no dataset CNR-EXT.

## 2.2 Pré-processamento das imagens

Para o presente trabalho foi escolhida a câmera “A” do subset “CNR-PARK”, que possui um total de 6.171 patches (pedaços de frames, cada um representando uma vaga individual), sendo 3.621 vagas ocupadas e 2.550 vagas vazias.

Através de um pequeno utilitário Console Application feito em C# (disponível na seção Links), foram sorteados 100 patches (50 vagas ocupadas e 50 vazias) para servirem como dados de exemplo e referência posterior. Essas 100 imagens (chamadas “dataset de exemplo”) foram copiadas para um diretório específico (“A - Imagens tratadas – exemplo”), enquanto todas as outras 6.071 imagens (ou seja, as 6.171 originais menos as 100 de exemplo) foram copiadas para o diretório “A - Imagens tratadas - busca”, também chamadas de “dataset de busca”.

Em outras palavras, temos:

* Dataset de exemplo: composto de 100 imagens escolhidas aletoriamente do conjunto total. É usado como base, ou seja, exemplo para a definição do estado de cada vaga que será buscada;
* Dataset de busca: são as 6.071 imagens restantes que simulam o caso de uso real, ou seja, cada uma delas é comparada contra cada uma das 100 imagens do “dataset de exemplo” para a definição do estado (vazia/ocupada).

Para a definição do tamanho da base de teste, alguns fatores precisaram ser levados em conta: cada imagem do “dataset de busca” será comparada contra cada uma do “dataset de exemplo”. Portanto quanto maior a quantidade das imagens de exemplo maior será o tempo de processamento de cada imagem de busca. Com o “dataset de exemplo” composto por 100 imagens o tempo para cada imagem de busca ficou em 2,3 segundos, o que é razoavelmente alto. Este tempo e outras métricas podem ser verificados na seção Apresentação de Resultados. Por outro lado, com menos imagens temos uma perda em termos de acertos de estado da vaga (vazia/ocupada), sendo assim encontrei no valor de 100 imagens um equilíbrio entre performance e acurácia.

Todas essas imagens de patches de vagas tiveram os nomes prefixados com “busy\_” ou “free\_” para facilitar a análise do resultado.

# 3. Processamento/Tratamento de Dados

Nesta seção apresentarei breve explanação sobre cada parte do script.

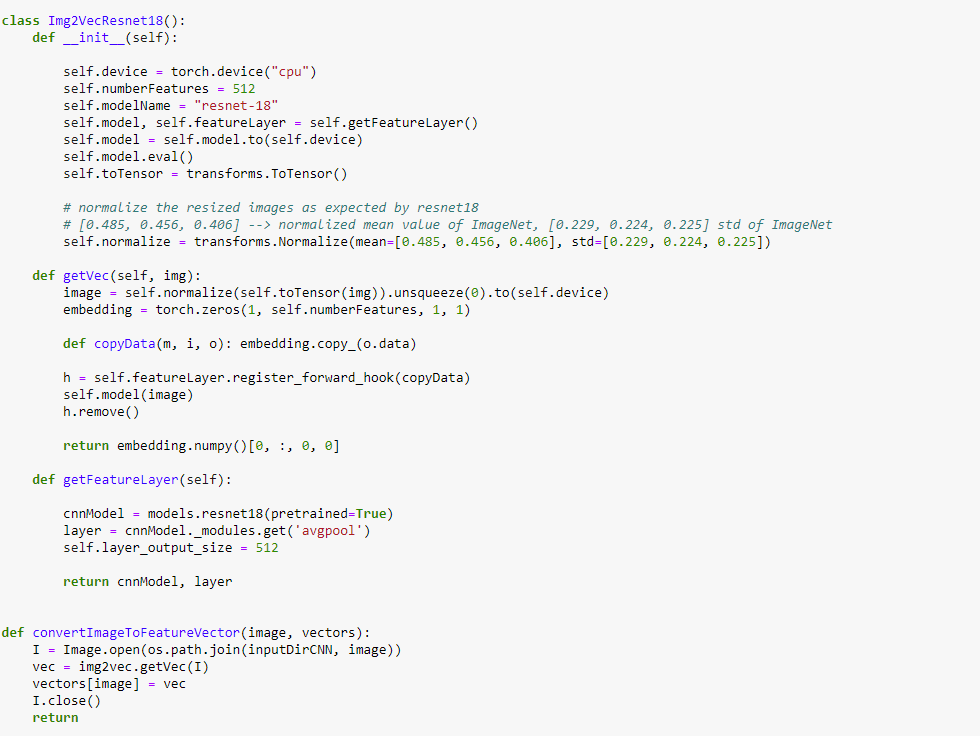
## 3.1 Redimensionamento de imagens

Pegamos cada imagem no diretório de entrada e as redimensionamos para o tamanho de 224 x 224 pixels, que é o tamanho esperado pela CNN ResNet18. O detalhe relevante aqui é que devemos tomar cuidado para preservar a informação de metadados EXIF da imagem original, pois ela contém a orientação da imagem. A seguir a parte do código mais significativa:



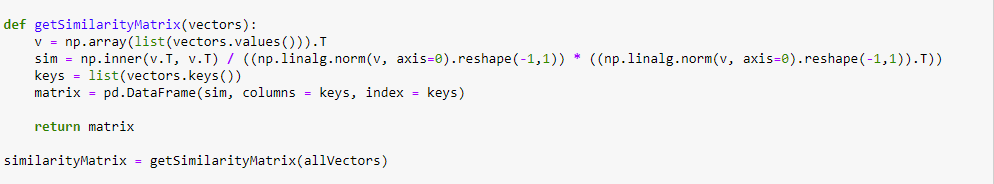
## 3.2 Criação dos vetores de características com ResNet18

Esta é a parte principal do script, onde são criados os vetores de features (características) para cada imagem do dataset de exemplo. A entrada é normalizada para os valores de média [0.485, 0.456, 0.406] e desvio padrão [0.229, 0.224, 0.225] da ImageNet. Segue trecho principal do código:



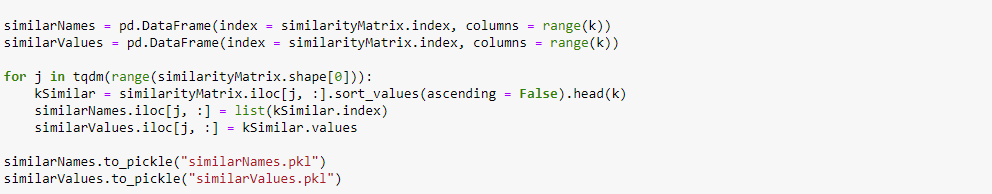
## 3.3 Cálculo da similaridade de cosseno

Esta função calcula a similaridade cosseno de cada vetor para todos os outros. Para conjuntos de imagens muito grandes (da ordem de milhares) esta matriz pode ficar muito grande e tornar o processamento lento, sendo desejável o uso de uma outra estrutura de dados.



## 3.4 Preparando as listas de top-k imagens mais semelhantes

Baseado na matriz de similaridade criada no passo anterior, armazenamos em outra estrutura de dados as k imagens mais semelhantes à cada uma do dataset de exemplo.



## 3.5 Testando a similaridade entre as imagens de exemplo

Neste passo iremos testar e mostrar as top-k imagens mais semelhantes à cada uma das 100 imagens que compõem o dataset de exemplo. Também usamos uma simples lógica para calcular a acurácia (taxa de acertos) percentual. Segue trecho do código:



## 3.6 Testando a similaridade entre as imagens de exemplo e as de busca

Este passo é bastante similar ao anterior, porém é o que mais se aproxima do caso de uso real. Também requer um tempo bem maior para a execução, dado a diferença de tamanho dos datasets (6.071 deste contra apenas 100 do anterior):



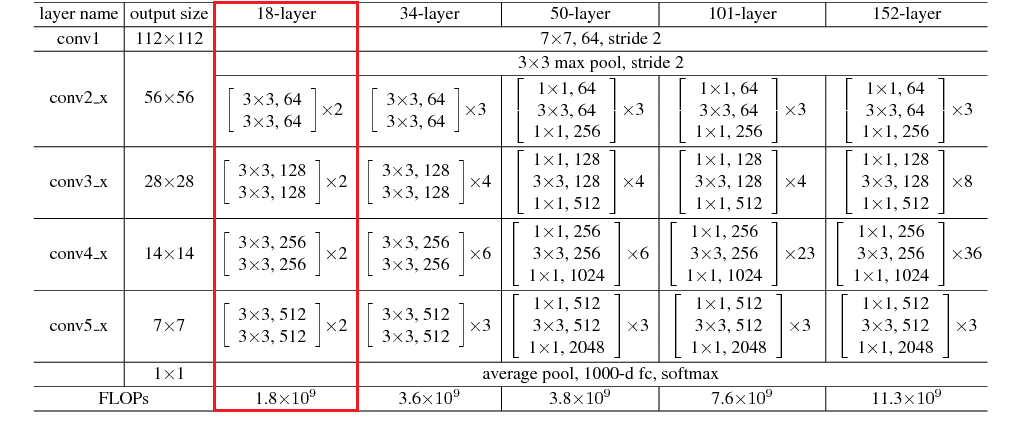
# 4. Análise e Exploração dos Dados

## 4.1 Visão geral sobre a ResNet18

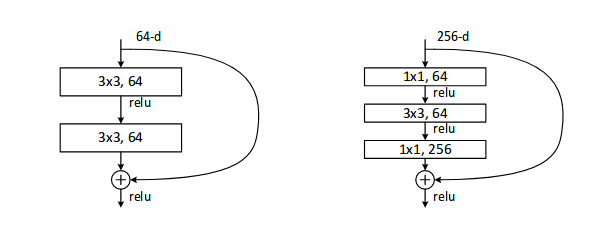
Seguindo a abordagem vista em (KORZEC, 2020), usamos uma rede neural convolucional (CNN, na sigla em inglês) pré-treinada chamada ResNet18 (PYTORCH TEAM, 2017) incluída na biblioteca PyTorch. A ResNet ganhou primeiro lugar no concurso de classificação de imagens ILSVRC 2015 com taxa de erro no top-5 de apenas 3,57%. Também ganhou primeiro lugar nas competições ILSVRC e COCO 2015 nas categorias ImageNet Detection, ImageNet localization, Coco detection e Coco segmentation (JAY, 2018)

A ResNet18 é uma CNN pré-treinada com 18 camadas de profundidade. A rede neural pode classificar imagens em mais de 1000 categorias. Ela requer como entrada imagens RGB de resolução 224x224 pixels, com valores resultantes no intervalo [0, 1]. As imagens são normalizadas para a média [0,485, 0,224, 0,225] e desvio padrão [0,229, 0,224, 0,225], então são todas redimensionadas para este formato, cuidando para que não se perca informação de orientação da imagem. Estas ponderações foram obtidas através do treinamento da ResNet18 com a ImageNet (DENG et al., 2009), em caso de treinamento com outros datasets, os valores serão outros.

Na figura abaixo podemos observar um diagrama da arquitetura das ResNets, com a ResNet18 em destaque, conforme disponível em (PYTORCH TEAM, 2017):



Cada bloco da ResNet pode ser de duas camadas de profundidade (para redes pequenas como a ResNet18 ou 34) ou de três camadas de profundidade (no caso de redes maiores como as ResNets 50, 101 ou 152), conforme figura abaixo (JAY, 2018):



## 4.2 Usando redes neurais convolucionais como geradores de vetores de características

Métodos tradicionais de classificação por características normalmente envolvem busca por bordas (edge detection) em diferentes direções (orientações) e densidades locais. Ao invés de fazer todo este processo trabalhoso e maçante, já existem modelos pré-treinados em bases de imagens gigantescas. A Imagenet, usada neste caso pela ResNet18, por exemplo possui mais de 14 milhões de imagens. Não é necessário o entendimento do processo por trás, funcionando como uma abordagem caixa-preta. Só é necessário saber que imagens semelhantes geram entradas semelhantes na tabela de vetores de características.

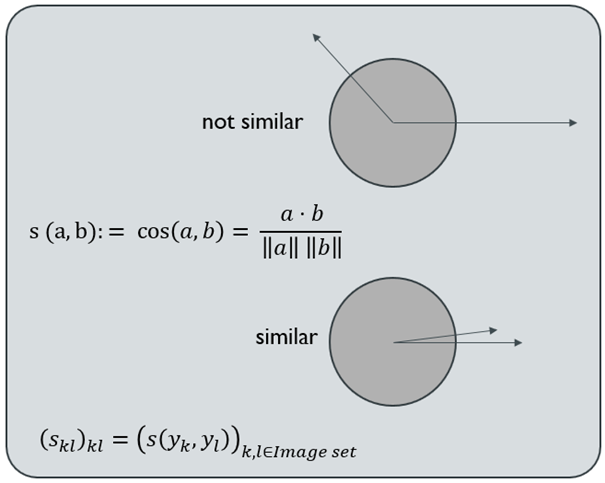
Quando usamos uma rede pré-treinada, como a ResNet18, todos os parâmetros já estão fixos, não sendo necessário se preocupar com eles. Apenas deve-se pré processar as imagens para se adequar ao formato de entrada usado pela rede, como visto anteriormente.

## 4.3 Cálculo de similaridades usando diferença dos vetores de características

Para o cálculo do nível de similaridade entre as imagens uma abordagem simples e eficiente é o cálculo da similaridade de cosseno para todos os pares de imagens, conforme demonstrado no artigo por (KORZEC, 2020).

A similaridade de cosseno é uma medida de comparação da orientação de dois vetores não nulos, independentemente da magnitude deles. Dois vetores com a mesma orientação possuem uma similaridade de cosseno igual a 1, dois vetores distantes por 90º tem a similaridade igual a zero, e dois vetores diametralmente opostos (ou seja, distantes por 180º) possuem a similaridade igual a -1, conforme (SINGHAL, 2001).

Na figura abaixo, retirada de (KORZEC, 2020), temos a visualização da similaridade de cosseno para duas dimensões, que é possível de ser representada e interpretada facilmente e em forma de diagrama:



Para uma quantidade maior de dimensões funciona da mesma forma, considere por exemplo os seguintes vetores, também extraídos de (KORZEC, 2020):



Intuitivamente, a similiaridade entre a e b deve ser maior do que entre a e c ou entre b e c. E, de fato, após aplicar os vetores na fórmula acima, encontramos:



Portanto, como esperado, temos:



O cálculo de similaridade é um tópico de estudo muito frequente na área correlata de sistemas de recomendações. Posteriormente poderiam ser avaliadas outras métricas de semelhança, em busca de resultados melhores em precisão (acurácia) e/ou performance. Para maior aprofundamento neste tópico, ver (AGGARWAL, 2016).

## 4.4 Usando listas de top-k mais semelhantes

De posse dos índices de similaridade entre todos os pares de imagens, dispostos em forma de matriz, basta selecionarmos as top-k mais semelhantes, com k=4 no nosso caso.

Não existe nenhuma razão específica para a escolha deste valor para k, apenas é um número arbitrário. A rigor, nem seria necessário já que o objetivo é encontrar apenas a imagem mais semelhante, sendo as outras k-1 semelhantes descartadas. Porém essas imagens se mostraram valiosas para avaliar a acurácia de detecção de similaridade, pois em alguns casos existe erro na primeira e acerto em outra, ou vice versa. Isso será demostrado posteriormente na seção de Apresentação de Resultados.

# 5. Criação de Modelos de Machine Learning

Para este trabalho foi desenvolvido um script em Pyhton, usando a ferramenta Jupyter Notebook, rodando localmente (o Google Colab não é tão prático quando se trata de uma quantidade grande de imagens).

O código foi baseado no artigo (KORZEC, 2020), com as devidas adaptações necessárias à diferente finalidade. Originalmente era um sistema de recomendação de produtos baseado na semelhança de imagens, ou seja, é possível um pré processamento e deixar as k imagens mais semelhantes já preparadas. No caso de um sistema de detecção de vagas em tempo real, existe a necessidade de trabalhar com imagens inéditas o tempo todo.

O script pode ser encontrado na seção Links.

# 6. Apresentação dos Resultados

Num primeiro momento foi feita uma comparação de cada uma das 100 imagens do dataset de exemplo com as demais (excluindo a própria imagem). O resultado foi este:

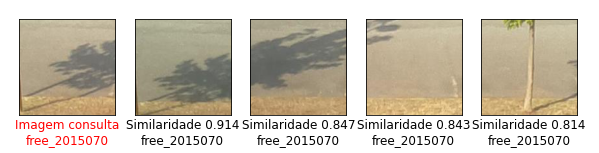
Total: 100

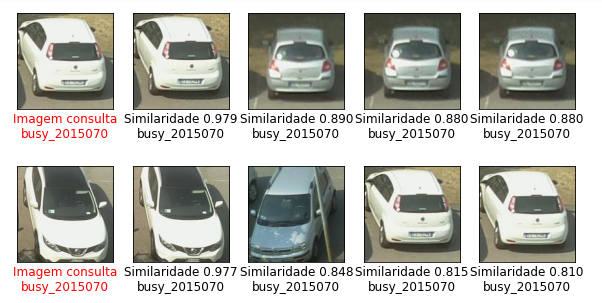
Taxa de acerto da imagem mais semelhante: 100.0%

Taxa de acerto das 4 imagens mais semelhantes: 94.5%

A acurácia (taxa de acerto) da primeira imagem (ou seja, a mais semelhante) foi de 100%. No acumulado das quatro imagens mais semelhantes houve uma taxa de acerto de 94,5%. Nos exemplos a seguir o nome do arquivo de imagem foi truncado em 12 caracteres. Segue abaixo uma amostra do resultado de busca de algumas imagens:









Posteriormente foi feita a análise com o dataset de busca, mais próximo do caso de uso real, composto por 6.071 imagens, com o seguinte resultado:

Total: 6071

Taxa de acerto da imagem mais semelhante: 98.17163564486904%

Taxa de acerto das 4 imagens mais semelhantes: 96.70564981057487%

Tempo total: 43min 52s

Tempo por imagem: 2,31s

Alguns exemplos de imagens que foram equivocadamente avaliadas, numa primeira análise os erros são bem mais comuns em vagas vazias do que em vagas ocupadas:





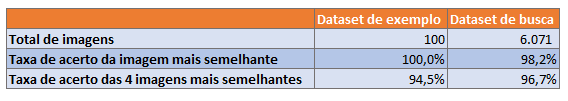






É possível notar que o erro se dá, normalmente, na imagem mais semelhante, sendo possível de tratar com algum simples algoritmo quando as quatro imagens mais semelhantes indicarem resultados diferentes entre si, aplicando pesos para cada uma das quatro imagens mais semelhantes.

A tabela a seguir mostra as métricas de erro consolidadas para ambos os datasets:



## 6.1 Comparação com outras redes ResNet

Foi efetuada uma comparação com as outras redes convolucionais ResNet: ResNet34, ResNet50, ResNet101 e ResNet152, sendo que este número representa o número de camadas de profundidade da rede neural convolucional.

[INSERIR TABELA E GRAFICO AQUI]

# 7. Links

* Repositório contendo o script, imagens, testes e o utilitário usado para o pré tratamento das imagens: <https://github.com/albs-br/tcc-machine-vision>
* Vídeo com demonstração do script em Python testado contra a base de exemplo com 100 imagens de vagas, metade delas ocupadas e metade vazia: <https://youtu.be/CW7c21cET5M>

# REFERÊNCIAS

KORZEC, Maciej D. Effortlessly Recommending Similar Images. *In:* **Towards Data Science**. [*S. l.*], 27 jun. 2020. Disponível em: <https://towardsdatascience.com/effortlessly-recommending-similar-images-b65aff6aabfb>. Acesso em: 11 jan. 2021.

AMATO, Giuseppe *et al.* **CNRPark+EXT**: A Dataset for Visual Occupancy Detection of Parking Lots. [*S. l.*], jul. 2015. Disponível em: <http://cnrpark.it/>. Acesso em: 4 jan. 2021.

PYTORCH TEAM. **RESNET**: Deep Residual Networks Pre-Trained on ImageNet. [*S. l.*], 2017. Disponível em: <https://pytorch.org/hub/pytorch_vision_resnet/>. Acesso em: 13 jan. 2021.

SINGHAL, Amit. **Modern Information Retrieval: A Brief Overview. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering**, [*s. l.*], v. 24, n. 4, p. 35-43, 2001. Disponível em: <http://sites.computer.org/debull/A01dec/A01DEC-CD.pdf>. Acesso em: 5 fev. 2021.

DENG, J. et al. ImageNet: A Large-Scale Hierarchical Image Database. **IEEE Computer Vision and Pattern Recognition (CVPR)**, [*s. l.*], 2009. Disponível em: <http://www.image-net.org/papers/imagenet_cvpr09.pdf>. Acesso em: 11 jan. 2021.

JAY, Prakash. **ResNet and ResNeXt**: Understanding and Implementing Architectures of ResNet and ResNeXt for state-of-the-art Image Classification: From Microsoft to Facebook [Part 1]. [*S. l.*], 2018. Disponível em: <https://medium.com/@14prakash/understanding-and-implementing-architectures-of-resnet-and-resnext-for-state-of-the-art-image-cf51669e1624>. Acesso em: 20 mar. 2021.

AGGARWAL, Charu C. **Recommender Systems**: The Textbook. *[S. l.]*: Springer, 2016.