

Projeto Demonstrativo 4: Segmentação de Imagens Aéreas

Pedro Henrique Luz de Araujo

pedrohluzaraujo@gmail.com

Raphael Soares Ramos

raphael.soares.1996@gmail.com

Departamento de Ciência da Computação

Universidade de Brasília

Campus Darcy Ribeiro, Asa Norte

Brasília-DF, CEP 70910-900, Brazil,

Abstract

¹ O objetivo do Aprendizado de Máquina é transformar dados em informação. Após aprender de um conjunto de dados, nós queremos que a máquina seja capaz de responder questões sobre os dados. Neste projeto foram usados e avaliados dois algoritmos de aprendizado de máquina que usam dados de imagens aéreas e os transformam em informação, para então realizar a segmentação das imagens. Essa informação é obtida através de extração de regras e padrões dos dados da imagem, como análise de descritores de textura das imagens. Com o auxílio de uma imagem *ground truth*, foi gerada uma imagem de previsão das imagens aéreas, onde os pixels brancos indicam prédios e pretos não prédios (tanto na imagem *ground truth* quanto na gerada pelos dois métodos utilizados). A acurácia nula na imagem de avaliação, que é a acurácia obtida sempre prevendo a classe mais frequente, é igual a 65%. Ambos os métodos apresentados para a solução do problema obtiveram uma acurácia superior a 65%. O primeiro método é mais simples e obteve no melhor cenário 68% de acurácia e 37% de índice Jaccard. O segundo método é mais robusto e obteve 88% de acurácia e 64% de índice Jaccard.

1 Introdução

O Inria Aerial Image labeling dataset [1] é um conjunto de dados de segmentação de imagens aéreas. A tarefa consiste em classificar pixéis em edifício e não edifício. Ao todo é coberta uma área de 810 km^2 , metade para treinamento e metade para teste, de diversas cidades, como Austin, Chicago, Viena e Innsbruck. O dataset é composto de imagens coloridas de 5000 por 5000 pixéis com resolução espacial de 30 centímetros.

Para o presente trabalho, lidaremos com apenas três imagens do dataset, todas de Viena. Uma será usada para fins de treinamento, outra para fins de validação e a última para fins de teste, como explicamos na Seção 2. Compararemos dois modelos diferentes: um baseado em técnicas de visão computacional e aprendizado de máquina clássico; e outro a partir do uso de redes neurais convolucionais.

© 2018. The copyright of this document resides with its authors.

It may be distributed unchanged freely in print or electronic forms.

¹Pedro contribuiu com a implementação, experimentos e testes do requisito 2, além da escrita de parte da introdução e partes da metodologia, resultados e conclusões referentes ao requisito 2. Raphael contribuiu com a implementação, experimentos e testes do requisito 1, além da escrita de parte da introdução e partes da metodologia, resultados e conclusões referentes ao requisito 2.

2 Metodologia

2.1 Ferramentas

Usamos a biblioteca OpenCV [4], versão 3.3.0, para carregar as imagens. Para realizar operações sobre matrizes e vetores, utilizamos a biblioteca de computação numérica em Python, NumPy. Usamos ainda, como linguagem, Python 3.5.2, e o gerenciador de bibliotecas Anaconda 3. Por fim, foram usadas as bibliotecas Keras [5] e Tensorflow [6] para o treinamento e construção de redes neurais. O treinamento foi realizado com o uso de uma placa de vídeo GeForce GTX 750 Ti.

Para ambos os requisitos, selecionamos uma imagem do *dataset* para treinamento, uma para validação (escolha de hiper-parâmetros), e uma para o teste final de avaliação dos resultados. A Figura 1 exibe as imagens com os respectivos valores de *ground truth*.



(a) Imagem de treinamento.



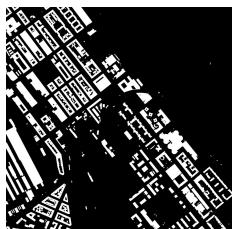
(b) Imagem de validação.



(c) Imagem de teste.



(d) Rótulo de treinamento.



(e) Rótulo de validação.



(f) Rótulo de teste.

Figure 1: Dataset.

2.2 Requisito 1

2.2.1 O modelo

Foi utilizada a técnica *K-nearest Neighbors (KNN)* [7] para a segmentação das imagens aéreas, com os descritores de textura da *Grey Level Co-Ocurrence Matrix (GLCM)* [8] sendo “pontos” do conjunto de treinamento.

Essa técnica consiste em guardar todos os dados dos pontos de treinamento e rotular novos pontos baseados na proximidade com eles. Para classificar um novo ponto, o KNN busca nos N_k pontos mais próximos que foram salvos e então rotula os novos pontos de acordo com a classe do conjunto de treinamento que contém a maior parte dos N_k vizinhos².

²Alternativamente, o KNN pode ser usado para regressão. Neste caso, o resultado retornado é a média dos valores associados com os N_k vizinhos mais próximos.

092 O método de classificação KNN pode ser muito efetivo, mas requer que todo o conjunto
093 de treinamento seja salvo. Logo, ele pode usar muita memória e se tornar lento, apesar de se
094 beneficiar do paralelismo oferecido por *Field Programmable Gate Arrays (FPGAs)*.

095 Foram derivadas três estatísticas/features da GLCM para serem utilizadas no KNN:

- 096
- 097 • Contraste: Mede as variações na GLCM e a presença de transição abrupta de níveis de
098 cinza (bordas).
 - 099 • Energia: Fornece a soma dos quadrados dos elementos na GLCM. Também é con-
100 hecida como uniformidade ou segundo momento angular;
 - 101 • Homogeneidade: Mede a proximidade da distribuições dos elementos na GLCM com
102 a diagonal da GLCM;

104 Apenas uma distância (1 pixel) e um ângulo (0 radiano) foi considerado.

105 Foi utilizado os 3 canais de cores da imagem com o objetivo de melhorar o resultado,
106 visto que os pixels em nível de cinza de um prédio podem se assemelhar a pixels de um outro
107 elemento da imagem que não corresponde a um prédio. Além disso, foi possível aumentar
108 o número de exemplos de treinamento considerando os 3 valores de intensidade para cada
109 pixel da imagem.

111 2.2.2 Experimentos

113 O KNN foi executado usando 5 e 10 vizinhos (a escolha de 5 vizinhos foi recomendada
114 como uma boa opção por [2]), por motivos de comparação. Os descritores GLCM usados
115 como features para classificação foram obtidos dos valores dos 10734639 pixels de prédios
116 da imagem de treino. Esses 32203917 valores de pixels foram separados em 83 matrizes para
117 que as features destas matrizes fossem calculadas e usadas como exemplos de treinamento
118 para o classificador. Os pixels correspondentes a não prédios foram separados de forma
119 similar em 87 matrizes, e alimentaram também os exemplos de treinamento do classificador.

120 Para a classificação da imagem de teste foi passado uma janela de tamanho 50x50 e
121 100x100 na imagem. Cada uma dessas janelas foram classificadas como prédio ou não
122 prédio, e então foi feita a contagem dos pixels que realmente são prédios.

123 Assim, a acurácia foi calculada através da divisão do número de acertos pelo número
124 total de pixels na imagem (25 milhões) e o *IoU/Jaccard Index* foi calculado usando a seguinte
125 razão:

$$126 \quad \text{IoU} = \frac{vp}{vp + fn + fp}, \quad (1)$$

127 em que vp é a quantidade de verdadeiros positivos (pixels que são prédios e foram clas-
128 sificados como prédios); fn é a quantidade de falsos negativos (pixels que o classificador
129 incorretamente previu como não sendo prédio); e fp são os falsos positivos (pixels que o
130 classificador incorretamente previu como sendo prédio).

132 2.3 Requisito 2

133 2.3.1 O modelo

134 Como segundo método de segmentação das imagens aéreas, escolhemos treinar o modelo
135 DeepLabv3+ [3] com a rede MobileNetV2 [10] como extrator de características de imagem.

MobileNetV2 é uma rede convolucional baseada em uma estrutura residual invertida, em que existe conexões atalho (*shortcut connections*) entre camadas de gargalo (*bottleneck layers*). Em vez de convoluções completas, a rede usa convoluções separáveis em profundidade (*Depthwise separable convolutions*). A convolução separável em profundidade baseia-se em fatorizar a operação de convolução em duas camadas: uma convolução em profundidade que aplica um filtro para cada canal da entrada; e uma convolução por pontos (*pointwise*), de tamanho 1x1, responsável por criar novas características por combinações lineares dos canais de entrada. Como resultado, são mais baratas computacionalmente sem perdas de performance significativas em relação às convoluções completas.

É aplicada a cada camada de convolução uma camada de gargalo linear, responsável por reduzir a dimensionalidade da entrada, melhorando a performance computacional da rede. Entre camadas de gargalo são usadas conexões de atalho, a fim de contribuir para a propagação do gradiente através das camadas, permitindo o treinamento de redes mais profundas. Tal estrutura, composta de convolução, gargalo e conexões residuais é o bloco de construção principal da MobileNetV2, cuja arquitetura pode ser resumida em uma camada convolucional inicial de 32 filtros, seguida por 19 blocos de gargalo e finalizada por mais duas convoluções e *average pooling*.

O modelo DeepLabv3+, por sua vez, é uma rede codificadora-decodificadora (*encoder-decoder*), no qual o módulo de codificação é responsável por reduzir gradualmente o mapa de características e capturar informações semânticas complexas, enquanto o módulo de decodificação tem como objetivo recuperar gradualmente as informações espaciais.

O módulo de codificação do DeepLabv3+ aplica convoluções dilatadas para extrair as características obtidas de uma rede convolucional em resolução arbitrária - no nosso caso, as características geradas pela MobileNetV2. Convoluções dilatadas permitem o controle da resolução e do campo de visão da entrada, generalizado a operação de convolução [1]. O mapa de características final do módulo de codificação possui 256 canais, permitindo a captura de informações semânticas de alto nível.

Já a parte de decodificação consiste no aumento gradual da resolução do mapa de características obtido do módulo de codificação combinado com a concatenação do mapa de características de baixo nível de resolução correspondente da MobileNetV2 e camadas de convolução. A arquitetura final do DeepLabv3+ é ilustrada pela Figura 2.

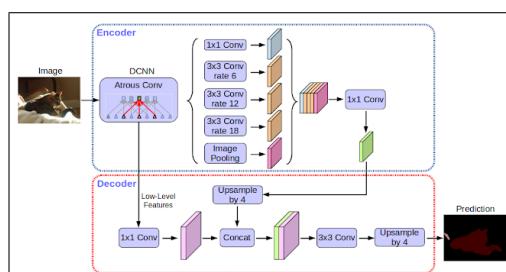


Figure 2: Arquitetura DeepLab3+. DCNN corresponde ao MobileNetV2. *Atrous Conv* representa as convoluções dilatadas, sendo *rate* o fator de dilatação. Imagem retirada de [5].

Utilizamos uma implementação em Keras do modelo descrito disponível no seguinte repositório: <https://github.com/bonlime/keras-deeplab-v3-plus>.

184 **2.3.2 Experiments**

185 Avaliamos o modelo no teste em nove cenários diferentes, envolvendo três variáveis: o
186 tamanho da entrada, o uso de pesos pre-treinados e, neste último caso, se fixariam os
187 pesos do extrator de características, isto é, da MobileNetV2.

188 A imagem de treinamento original é de 5000x5000 pixels, o que acarreta dois proble-
189 mas. Primeiramente, não caberia na memória da GPU. Em segundo lugar, dificilmente a
190 rede apresentaria um bom desempenho com apenas um exemplo de treinamento. Assim,
191 resolvemos fazer experimentos dividindo as imagens de treino originais em: 2500 imagens
192 de tamanho 100x100; 625 de tamanho 200x200; e 400 de tamanho 250x250. A intuição é
193 que haja um trade-off entre quantidades de exemplos e tamanho da imagem, afinal, quanto
194 maior a imagem maior a carga semântica.

195 Para cada um desses três tamanhos de imagem, treinamos três modelos diferentes: um
196 com pesos inicializados aleatoriamente; um com pesos pre-treinados no dataset COCO [8];
197 e outra também com pesos pre-treinados, mas com os pesos das camadas do extrator de
198 características fixados. Espera-se que o modelo apresente melhor resultados nos dois últimos
199 casos, uma vez que o uso de pesos pre-treinados, um caso de transferência de aprendizado,
200 empiricamente resulta em modelos com maior capacidade de generalização.

201 Para cada um dos nove cenários treinamos com batches de tamanho 16, no caso de pe-
202 sos do extrator fixados, e quatro nos demais casos. Como otimizador usamos *Stochastic*
203 *Gradient Descent* com momentum 0.9 e corte de gradientes com norma superior a um. A
204 taxa de aprendizado tem como valor inicial 0.1, diminuindo pela metade após cinco *epochs*
205 sem diminuição da função de custo, que é a função de entropia cruzada binária. Sejam p a
206 previsão da rede e y o rótulo que se deseja prever, a função de entropia cruzada binária é

207
$$\text{cross-entropy} = -(y \log(p) + (1 - y) \log(1 - p)). \quad (2)$$

208 Cada modelo foi treinado em um máximo de 100 *epochs*, sendo que o treinamento é
209 interrompido após 50 *epochs* sem melhoria da acurácia do conjunto de validação. Por fim,
210 salvamos o modelo apenas quando houvesse melhora do coeficiente de Jaccard no conjunto
211 de validação.

213

214 **3 Resultados**

215

216 **3.1 Requisito 1**

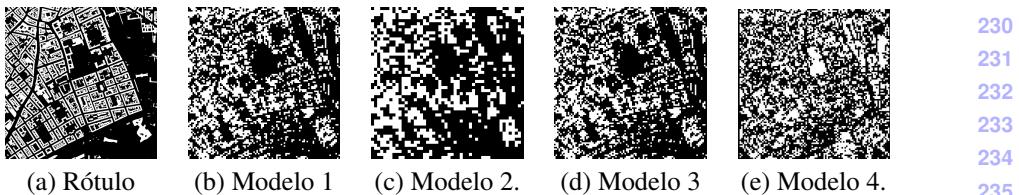
217

218 A Tabela 1 exibe os valores de acurácia e de coeficiente de Jaccard na imagem de teste para
219 os 4 cenários.

Modelo	Acurácia	IoU
janelaTamanho-50-rgb-vizinhos-5	0.679340	0.370197
janelaTamanho-100-rgb-vizinhos-5	0.664897	0.352915
janelaTamanho-50-grey-vizinhos-5	0.679874	0.366998
janelaTamanho-50-rgb-vizinhos-10	0.548790	0.274196

220 Table 1: Resultados no conjunto de teste usando o método 1.
221

222 A Figura 3 mostra as saídas dos modelos usando janela de tamanho 50 e 100 em rgb com
223 5 vizinhos para o *KNN*; usando janela de tamanho 50 em tons de cinza com 5 vizinhos e
224 janela de tamanho 50 em rgb com 10 vizinhos.



236
237
238
239
240
241
242
243
244
245

Figure 3: Resultados do Requisito 1.

O programa demorou 4 minutos para ser executado ao usar janela de tamanho 100, e 5 ao usar janela de tamanho 50.

3.2 Requisito 2

A Tabela 2 exibe os valores de acurácia e de coeficiente de Jaccard na imagem de teste para cada um dos nove cenários.

Modelo	Acurácia	IoU
semPretreino-naoFixo-100	0.86	0.54
preTreino-Fixo-100	0.65	0.09
preTreino-naoFixo-100	0.88	0.56
semPretreino-naoFixo-200	0.79	0.54
preTreino-Fixo-200	0.55	0.24
preTreino-naoFixo-200	0.88	0.64
semPretreino-naoFixo-250	0.84	0.59
preTreino-Fixo-250	0.35	0.33
preTreino-naoFixo-250	0.88	0.64

246
247
248
249
250
251
252
253
254
255
256

Table 2: Resultados no conjunto de teste.

A Figura 4 mostra a imagem de teste, sua imagem de rótulo, e as saídas dos modelos com pre-treino, sem pesos fixos, e treinados com imagens de tamanho 200 (modelo 1) e 250 (modelo 2), por terem obtidos os melhores resultados de acurácia e de coeficiente de Jaccard.



270
271
272
273
274
275

Figure 4: Resultados do Requisito 2.

4 Discussão e Conclusões

4.1 Requisito 1

Nota-se pela Tabela 1 que o uso de 10 vizinhos para o algoritmo *KNN* prejudicou bastante a classificação. O que faz sentido, visto que aumentar o número de vizinhos diminui a ca-

276 pacidade do modelo. A grosso modo, a capacidade do modelo descreve o quanto complexa
277 é a relação que o modelo pode modelar. Um modelo com maior capacidade tem capaci-
278 dade de modelar mais relacionamentos entre mais variáveis do que um modelo com menor
279 complexidade.

280 Além disso, o uso da janela de tamanho 100 afetou de forma significativa o resultado
281 visual na imagem de teste. Apesar da acurácia ter diminuído em apenas 1% e o IoU em
282 2. Este resultado negativo já era esperado, visto que estamos classificando janelas inteiras
283 como pertencentes a prédios ou não. Entretanto, o uso de janelas muito pequenas também
284 não ajudou, visto que precisamos de mais pixels para obter descritores mais precisos e que
285 caracterizam melhor o conjunto em questão.

286 O uso da imagem de teste em nível de cinza em vez das cores rgb pouco alterou o re-
287 sultado visual e estatísticos. O que indica que os pixels referentes aos prédios mantém boa
288 distinção com os pixels referentes a não prédios, mesmo em nível de cinza.

289 Para melhoria do algoritmo de Machine Learning utilizado poderia ter sido utilizado
290 *cross-validation* para melhor ajuste dos parâmetros. Entretanto, os resultados provavelmente
291 não seriam muito diferentes dos obtidos.

292 4.2 Requisito 2

293 O método elaborado no Requisito 2 apresentou bons resultados. A Figura 4 demonstrou
294 que as previsões dos melhores modelos são extremamente semelhantes ao rótulo da imagem
295 teste. A maior fonte de erro se deu no canto inferior direito da imagem. Nota-se da imagem
296 original que nessa região há uma espécie de construção sobre trilhos de trem, os quais não são
297 considerados como edifícios pelo *ground truth*, embora sejam muito semelhantes a prédios
298 quando vistos de cima.

299 Temos como vantagem do uso de aprendizado profundo a desnecessidade de engenharia
300 de características - a própria rede o faz, podendo atingir resultados superiores a caracterís-
301 ticas criadas por humanos. Como contrapartida, necessita-se de uma grande quantidade de
302 exemplos de treinamento. Entretanto, isso pode ser mitigado pelo uso de transferência de
303 aprendizado, como o fizemos ao usar pesos pre-treinados.

304 Percebe-se que os modelos com pesos do extrator de características fixos apresentaram
305 baixa performance. Isso pode ser explicado, talvez, em razão da diferença do domínio de
306 imagens aéreas do domínio onde foram treinados os pesos, que são imagens de objetos em
307 contextos ordinários. Não obstante, ficou claro que o uso de pesos pre-treinados melhorou
308 os resultados quando permitiu-se o *fine-tuning* em todas as camadas da rede. Quanto ao
309 *trade-off* número de exemplos x tamanho das imagens, nota-se que modelos treinados com
310 imagens de diferentes tamanhos apresentaram aproximadamente a mesma performance, com
311 uma ligeira vantagem de imagens maiores quanto ao índice Jaccard.

312 Um problema do Requisito 2 é o alto custo computacional: foi necessária um pouco mais
313 de uma hora para treinar cada um dos modelos. Caso não fosse usada uma GPU tal tempo
314 seria muito superior. Por outro lado, uma vez treinada a rede a inferência é rápida, sendo
315 necessário menos de um minuto para classificar os 25 milhões de pixels da imagem de teste
316 sem o uso de GPU e contando o tempo do carregamento inicial da rede.

317 Como melhoria, há a possibilidade do uso de técnicas de aumento de dados (*data aug-
318mentation*) para aumentar o número de exemplos. Isso inclui transformações aleatórias nas
319 imagens, como rotação, inversões horizontais e verticais, e zooms. Como trabalho futuro
320 mostra-se interessante o treinamento da rede no conjunto inteiro de imagens aéreas com o
321 uso de *data augmentation*.

References

- [1] Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL <https://www.tensorflow.org/>. Software available from tensorflow.org. 322
323
- [2] GEAPA Batista and Diego Furtado Silva. How k-nearest neighbor parameters affect its performance. In *Argentine symposium on artificial intelligence*, pages 1–12. sn, 2009. 324
325
326
327
328
329
- [3] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 330
331
- [4] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 40(4): 332
834–848, April 2018. ISSN 0162-8828. doi: 10.1109/TPAMI.2017.2699184. URL 333
<doi.ieeecomputersociety.org/10.1109/TPAMI.2017.2699184>. 334
335
336
- [5] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *CoRR*, abs/1802.02611, 2018. URL <http://arxiv.org/abs/1802.02611>. 337
338
339
340
- [6] François Chollet et al. Keras. <https://keras.io>, 2015. 341
342
- [7] Robert M Haralick, K Shanmugam, Its’Hak Dinstein, et al. Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*, 3(6):610–621, 1973. 343
344
345
- [8] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10602-1. 346
347
348
349
350
351
- [9] Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, and Pierre Alliez. Can semantic labeling methods generalize to any city? the inria aerial image labeling benchmark. In *International Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE, 2017. doi: 10.1109/IGARSS.2017.8127684. URL <https://project.inria.fr/aerialimagelabeling/>. 352
353
354
355
356
- [10] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018. URL <http://arxiv.org/abs/1801.04381>. 357
358
359
360
361
362
363
364
365
366
367