Deep Learning - Relatório 2: *Semantic Segmentation*

Rian Freitas e Hanna Rodrigues

September 2023

1 Modificações no código

1.1 Pré-processamento dos Dados

Nesta etapa, realizamos o carregamento da imagem e aplicamos a normalização utilizando as funções fornecidas obtendo uma imagem de treinamento e sua de referência com as dimensões (3, 2565, 1919).

Além disso, implementamos a função *extract_patches*, que recebe a imagem e a referência como entrada, bem como um tamanho de patch, um stride e o número de classes desejado. No nosso caso, definimos o número de classes como 5, que corresponde às categorias do problema: Superfícies Impermeáveis, Edifícios, Vegetação Baixa, Árvores e Carros. Os tamanhos de patch escolhidos são os especificados no projeto: 32, 64 e 128, com um stride de mesmo valor respectivamente.

Os patches resultantes são representados como tensores com dimensões da seguinte forma:

(1, 2565/patch_size, 1919/patch_size, patch_size * patch_size * 3)

(1, 2565/patch_size, 1919/patch_size, patch_size * patch_size * 5)

para a imagem e referência respectivamente. Vale ressaltar que as dimensões da referência estão relacionadas ao número de classes, uma vez que foi utilizado o one-hot encoding para sua representação.

Após a criação dos patches, realizamos uma amostragem aleatória, selecionando 80% deles para o conjunto de treinamento, enquanto os 20% restantes são alocados para validação. Para preparar os dados para alimentar a rede neural, realizamos uma modificação no formato dos tensores. O tensor que representa a imagem e a referências de treino e validação são transformados para a forma:

(num_patches_treino, patch_size, patch_size, 3)

(num_patches_referencia, patch_size, patch_size, 5)

1.2 Modelo U-Net

A arquitetura da rede U-Net implementada é composta por um encoder que inclui três blocos contendo duas camadas de convolução 3x3 seguidas de camadas de max pooling 2x2. Em seguida, há um decoder que também possui três blocos, cada um consistindo de camadas de up-sampling 2x2 seguidas de duas camadas de convolução 3x3. Cada um desses blocos utiliza um número crescente de filtros, começando com 32 no primeiro bloco e chegando a 256 no último bloco, resultando em um total de 466.661 parâmetros treináveis na rede. Além disso, foram incorporadas duas conexões residuais, uma conectando o primeiro bloco do encoder ao primeiro bloco do decoder e a outra conectando o segundo bloco do encoder ao segundo bloco do decoder.

1.3 Treinamento

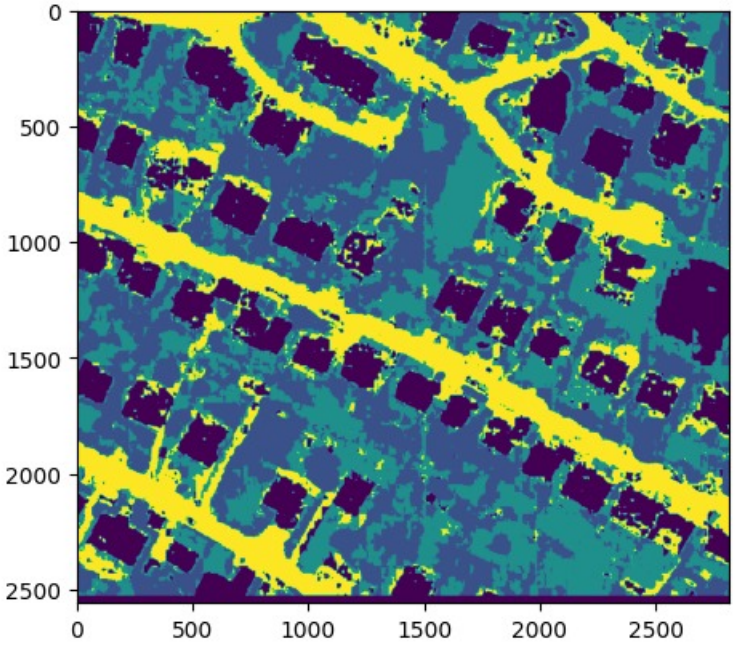
Na função de perda implementada, foi adotada a Cross-Entropy Categórica Ponderada, em que os pesos são calculados como a razão entre o número total de pixels e o número de pixels de cada classe. Isso significa que as classes com menor representação têm maior peso na perda, permitindo que o modelo se concentre mais nas classes menos frequentes. Além disso, para otimização, foi escolhido o otimizador Adam, com uma taxa de aprendizado de 0.0001 e parâmetro de decaimento momentum beta_1 igual a 0.9. Por fim, foi selecionado um batch_size de 64 amostras e o número de epochs definido como 100.

Utilizando a função fornecida, incorporamos o Early Stopping com uma paciência de 10, ou seja, estabelecemos um critério de parada no treinamento caso a função de perda não apresente melhoria durante 10 iterações consecutivas.

1.4 Resultados

Para diferentes tamanhos de patches, reconstruímos as imagens usando os patches gerados a partir das previsões da rede. Adicionalmente, calculamos várias métricas, incluindo a perda e a acurácia para os conjuntos de treinamento, validação e teste. Além disso, para cada uma das 5 classes, calculamos métricas adicionais, como o F1-score, recall e precisão.

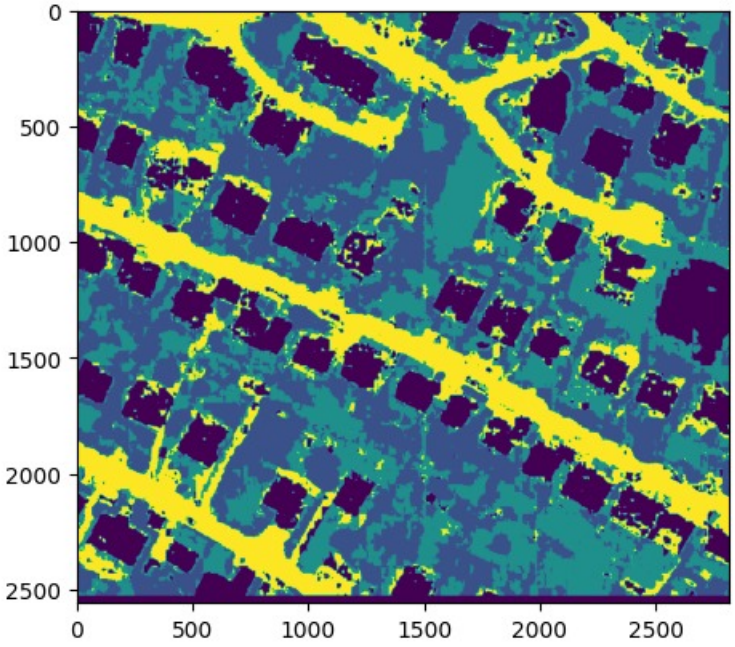
1.5 Patches 32 x 32



	Treino	Validação	Teste
Loss	0.74	2.19	-
Acurácia (%)	79.72	78.52	71.91

	F1-score	Recall	Precisão
Prédio	80.54928079	89.17558614	73.44468721
Árvore	74.2160158	75.38196507	73.08558513
Vegetação baixa	63.40663837	55.95832889	73.14219964
Carro	16.2754584	10.07691739	42.28739913
Superfície impermeável	73.84755757	82.35455914	66.93350852

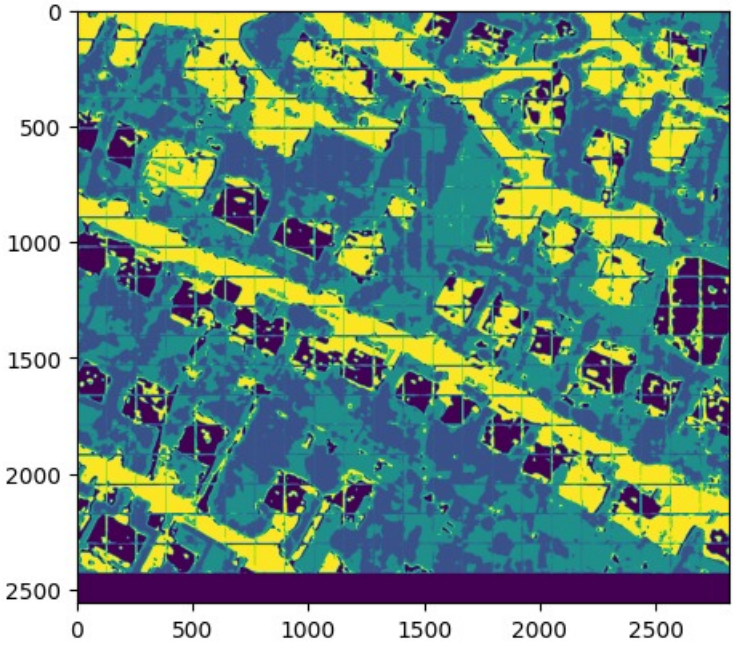
1.6 Patches 64 x 64



	Treino	Validação	Teste
Loss	1.09	2.09	-
Acurácia (%)	73.58	71.60	68.12

	F1-score	Recall	Precisão
Prédio	71.64444795	83.75190364	62.59544131
Árvore	72.27637941	75.31625617	69.47237062
Vegetação baixa	59.19451182	51.45835556	69.66832146
Carro	22.61399391	16.74432364	34.82005537
Superfície impermeável	73.09227395	74.93098351	71.3416422

1.7 Patches 128 x 128



	Treino	Validação	Teste
Loss	1.98	nan	-
Acurácia (%)	62.28	nan	57.87

	F1-score	Recall	Precisão
Prédio	48.81258318	50.01627304	47.66546771
Árvore	66.0421029	61.98914707	70.66211193
Vegetação baixa	57.26444758	52.04307082	63.65036039
Carro	27.79499716	23.54485615	33.91753676
Superfície impermeável	56.37429084	72.35737717	46.17471613

2 Conclusões Finais

A análise dos resultados destacou diferenças significativas no desempenho dos modelos de acordo com o tamanho dos patches empregados. Observou-se que os patches de menor dimensão (32x32) resultaram em uma acurácia superior no conjunto de teste, enquanto os patches maiores (128x128) conduziram a uma acurácia inferior. No entanto, um aspecto notável foi a melhora nas métricas relacionadas à classe de carros quando se utilizaram patches maiores. Com patches maiores, há uma maior representação de carros, permitindo que o modelo aprenda de forma mais eficaz essa classe específica que é rara em nossa amostra como podemos observar no F1-score e Recall desta classe.

Aqui, segue o [link do Colab].