Relatório Deep Learning
Assignment 2: Semantic Segmentation

Ana Carolina Erthal & Felipe Lamarca

Link para o Google Colab

1 Modificações no código

Iniciamos carregando as imagens a partir do Google Drive utilizando funções para que não fosse necessário repetir o processo manual a cada vez que o notebook fosse aberto, e normalizando as imagens. Foi possível, então, extrair os patches das imagens principais e referências, utilizando a função que construímos `extract_patches` utilizando tamanho e stride definidos para iterar pela imagem original.

Tendo todos os patches, dividimos o conjunto de treino em treino e validação, e avançamos para a transformação dos dados em one-hot encoding. Como tínhamos imagens que possuem canal `rgb`, não foi possível aplicar o one-hot encoding diretamente, e realizamos o passo intermediário de transformar as imagens em grayscale e utilizar um label encoder para termos um canal apenas representando a cor.

Inicializamos os pesos para fornecer à rede utilizando $w_i = \frac{\text{\# total pixels}}{\text{\# pixels of class i}}$ e partimos para a definição do modelo. Construímos a U-net seguindo a arquitetura padrão do modelo conforme a Figura 1, utilizando camadas de downscaling e posteriormente de upscaling, com skip connections. Nas camadas convolucionais, utilizamos ativação ReLU e zero-padding 1. Nas camadas de pooling, realizamos um Max Pooling com filtro 2x2.

Para o treinamento, completamos a função definindo um early stopping com patience 10 utilizando a função de perda. Isto é, avaliamos quando a loss está há 10 epochs sem melhora em relação à melhor loss para realizarmos uma parada, evitando que o modelo perca capacidade de generalização (isto é, evitando overfitting) e reduzindo tempo de treinamento despropositado.

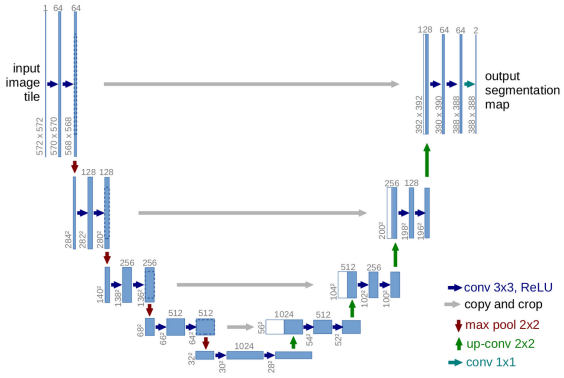


Figura 1 – Arquiterura da Unet Towards Data Science

Computamos as métricas (F1 score, acurácia, recall e precisão) do melhor modelo em relação ao conjunto de teste, e então o utilizamos para reconstrução da imagem de referência, exibindo as classificações realizadas pelo modelo.

Para evitar repetição de código ao longo do treinamento dos modelos, adicionamos todo o pipeline à classe `ModelPipeline()`, que realiza o processo descrito acima de maneira condensada.

Discutiremos mais adiante os hiperparâmetros utilizados e resultados obtidos.

2 Hiperparâmetros

Para todos os resultados apresentados nas tabelas de resultados abaixo, utilizamos os seguintes parâmetros:

Hiperparâmetro	**Valor**
`patch size`	(32x32), (64x64) ou (128, 128)
`stride`	32, 64 e 128
`# epochs`	20
`batch size`	10
`early stopping patience`	10
`loss`	Weighted Categorical Cross Entropy
`optimizer`	Adam
`learning rate`	0.001

Tabela 1 – Hiperparâmetros

3 Resultados

Métrica geral obtida a partir dos modelos utilizados (acurácia):

	Modelo 32x32	**Modelo 64x64**	**Modelo 128x128**
Acurácia	67.92%	63.72%	62.01%

Tabela 2 – Acurácia por modelo

Métricas classe-específicas obtidas a partir dos modelos utilizados:

Métrica	**Classe 1** building	**Classe 2** tree	**Classe 3** low vegetation	**Classe 4** car	**Classe 5** impervious surfaces
Recall	83.55%	73.82%	51.57%	5.63%	76.73%
Precision	71.23%	67.85%	67.49%	31.78%	65.18%
F1 Score	76.90%	70.71%	58.47%	9.57%	70.48%

Tabela 3 – Métricas por classe para patches de tamanho 32x32, stride = 32

Métrica	**Classe 1** building	**Classe 2** tree	**Classe 3** low vegetation	**Classe 4** car	**Classe 5** impervious surfaces
Recall	63.35%	71.62%	49.53%	0.0%	81.82%
Precision	68.53%	71.62%	49.53%	0.0%	81.82%
F1 Score	65.84%	69.44%	56.41%	0.0%	65.12%

Tabela 4 – Métricas por classe para patches de tamanho 64x64, stride = 64

Métrica	**Classe 1** building	**Classe 2** tree	**Classe 3** low vegetation	**Classe 4** car	**Classe 5** impervious surfaces
Recall	66.25%	60.11%	53.50%	0.0%	80.10%
Precision	60.74%	71.72%	61.57%	0.0%	54.00%
F1 Score	63.37%	65.40%	57.25%	0.0%	54.00%

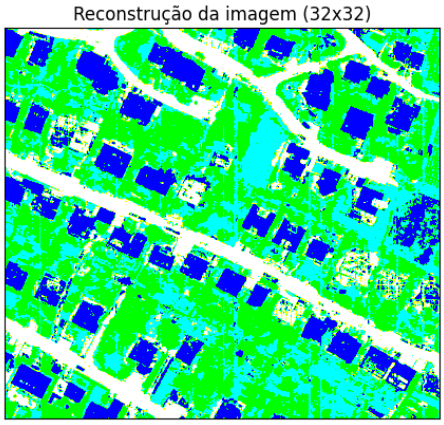
Tabela 5 – Métricas por classe para patches de tamanho 128x128, stride=128

Na prática, sabemos que a precisão representa o quanto o modelo acertou (True positive) dentre o total de vezes em que classificou algo como positivo (True positive + False positive). Enquanto isso, o recall representa o quanto o modelo acertou (True positive) dentre o total de vezes em que deveria ter classificado algo como positivo (True positive + False negative). Apesar da importância das duas métricas, não é claro qual delas deve prevalecer para tomada de decisões sobre o modelo. Por isso, utilizamos também o F1-Score, que representa uma combinação equilibrada das duas métricas.

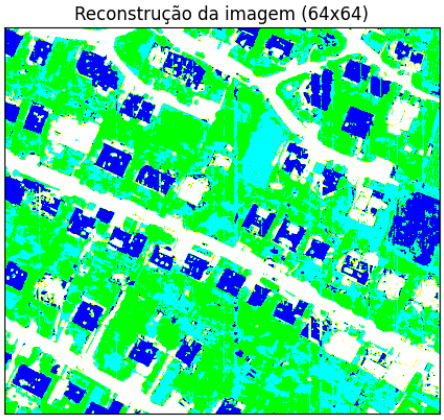
Nas tabelas acima, foi possível observar que o modelo que obteve maior acurácia foi o Modelo 32x32 (isto é, com patches de tamanho 32x32 e stride 32), e piorou conforme aumentamos o tamanho dos patches e stride. É importante destacarmos que esse desempenho era esperado, já que aumentando tamanho do patch e o stride conjuntamente, passamos a ter menos patches para o treinamento. Tínhamos o plano inicial de testar também o desempenho utilizando tamanhos de patch maiores com strides menores - já que acreditávamos que assim obteríamos performances boas, uma vez que teríamos mais imagens de treino -, mas tivemos problemas de falta de RAM.

Podemos interpretar, também, os resultados obtidos nas tabelas. Em geral, obtivemos resultados muito bons na classe 5, o que é bem justificado, já que as estradas são muito bem delimitadas por cor, e há muitas amostras nos patches. Para as classes 1, 2 e 3 também temos resultados bons. É interessante destacar que a classe 3, em geral, obtém maior precision que recall, já que não devolve tantos labels positivos para essa classe quanto deveria, reduzindo a sua taxa de $\frac{\text{TP}}{\text{TP+FN}}$ (recall) e aumentando a taxa de $\frac{\text{TP}}{\text{TP+FP}}$ (precision).

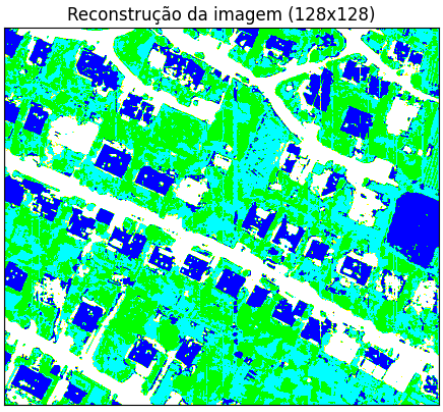
A classe 4 é, claramente, a de pior desempenho. Esse fato se deve principalmente à falta de amostra de carros na imagem de treinamento. Como o modelo quase não vê a classe nos patches de treino, não consegue classificar com sucesso na imagem de teste (e para os modelos 64x64 e 128x128, nem faz classificações de classe 4).



(a) Modelo 32x32



(b) Modelo 64x64



(c) Modelo 128x128

Figura 2 – Reconstrução da imagem de referência

As imagens de teste reconstruídas refletem as características das métricas dos modelos. Por exemplo, verificamos que a imagem gerada pelo modelo 32x32 é a melhor de todas e, por exemplo, consegue identificar alguns elementos de classe 4, ainda que poucos. Nos outros dois modelos, não identificamos qualquer elemento dessa classe. Também é possível observar a queda na precisão da classe 5 e nos scores da classe 1 no último modelo, em que passamos a ter buildings classificados como impervious surfaces.