

Assignment 2: Semantic Segmentation

Juan Belieni e Juliana Carvalho

29 de setembro de 2023

1 Comentários de código

Link para o notebook: [Collab](#).

Link para o repositório do GitHub: [Repositório](#)

Alteramos o código da seguinte maneira:

Realizamos o download dos dados da URL recebida e carrega as imagens com a função `load_tiff_image`. Modificamos a função para ser possível converter a imagem em escala de cinza, utilizando `image.convert("L")` para simplificar a análise.

1.1 Patches

Por meio da função `extract_patches(image, size, stride)` extraímos patches (recortes) da imagem original, considerando o tamanho especificado e a distância (stride) entre eles. Utilizamos a função `tf.image.extract_patches` do tensorflow para cortar os patches da imagem com os tamanhos e passos especificados, dentre 32×32 , 64×64 e 128×128 , como orientação do assignment. Retorna-se um array multidimensional contendo os patches.

1.2 One-hot-encoding

Criamos a função `one_hot_encode` para corresponder a categoria selecionada dentre 5 classes existentes (legendadas no assignment como "Impervious surface, Building, Low vegetation, Tree, Car") com base na comparação com as cores únicas, criando assim uma codificação categórica.

1.3 U-Net: implementação e conceitos

1.3.1 U-Net

A U-net [1] é uma rede convolucional para segmentação semântica de imagens. Sua arquitetura é baseada em um padrão de *encoder* e *decoder*, que contrai a imagem em direção a um espaço latente para depois expandi-lá em outra imagem que representa sua segmentação.

Sua arquitetura pode ser dividida em blocos de *downsample* e *upsample*. Cada bloco de *downsample* é constituído por:

- 2 camadas convolucionais 3×3 com o dobro do número de canais em relação ao bloco anterior;
- aplicação de ReLU em cada uma das camadas convolucionais 3×3 ;
- operação final de *pooling* com *stride* 2.

Por sua vez, cada bloco de *upsample* é constituído por:

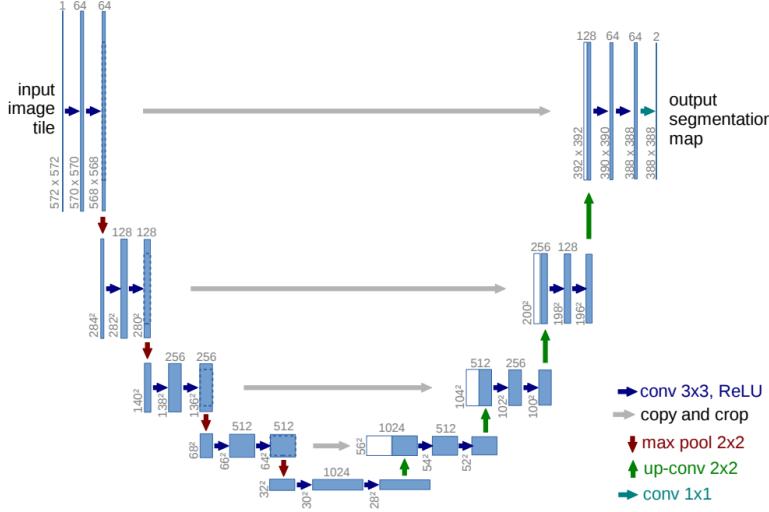


Figura 1: U-Net: imagem retirada do artigo de Ronneberger, Fischer e Brox [1].

- operação inicial de *upsampling* com uma convolução 2x2 que divide pela metade o número de camadas;
- concatenação com as *features* do respectivo bloco de *downsample*;
- 2 camadas convolucionais 3x3;
- aplicação de ReLU em cada uma das camadas convolucionais 3x3.

Definimos a função `unet(input_shape, n_classes)`: que implementa a arquitetura da U-Net, a qual possui um encoder com 4 camadas de downsampling (down_conv1 a down_conv4), uma camada convolucional intermediária, a middle_conv, e um decoder com 4 camadas de upsampling (up_conv1 a up_conv4).

A saída (`out_layer`) utiliza a ativação *softmax* para a segmentação da imagem de entrada em '`n_classes`' classes.

Ademais, a variável `n_layers` é uma lista que contém os números de filtros (potências de 2) a serem usados em cada camada de down e upsampling.

Funções auxiliares foram definidas:

- `convolution_layers(layer, n_filters)`: Define duas camadas de convolução 2D com ativação ReLU e as aplica à camada de entrada 'layer'. É utilizada para criar as duas camadas convolucionais nos blocos de downsampling e upsampling.
- `downsample_block(layer, n_filters)`: Cria um bloco de downsampling que consiste em duas camadas convolucionais seguidas por uma camada de max pooling 2D.
- `upsample_block(layer, conv_features, n_filters)`: Define um bloco de upsampling que começa com uma camada de upsampling 2D, concatena o bloco de downsampling correspondente 'conv_features', realiza assim, o "Copy and Crop" e o aplica como entrada das camadas convolucionais.

1.4 Treino

Renomeamos os argumentos da função `train_model` para ficarem mais descritivos e adicionamos o argumento de nome do modelo. Dentro dessa função, simplificamos trechos de código repetidos.

Ainda dentro dessa função, declaramos variáveis `best_val_loss`, `last_val_loss` e `early_stopping_counter` para implementar uma estratégia de "early stoping" com paciência igual a 10 e uma tolerância de $1e^{-3}$.

A estratégia observa a quantidade de épocas consecutivas em que o valor de perda na validação não melhora. Para isso, verificamos se a diferença da perda na época anterior e na época atual, e caso seja menor ou igual a -0.0001 , significa que a perda está aumentando, ou seja, o modelo pode estar piorando conforme passam as épocas.

1.5 Teste

Para testar a saída, criamos uma função `get_model(size)` com as definições dos modelos descritas abaixo.

Além disso, declaramos uma função `reconstruct_image` para reconstruir as imagens de acordo com o tamanho dos patches. Ela itera sobre os patches, os colocando nas posições corretas da imagem final.

Declaramos também uma `show_results(model, patch_size)` para mostrar as imagens lado a lado, além das métricas.

2 Dados

Os dados consistem em quatro imagens .tiff de tamanho 2565x191. São elas:

- imagem de treino;
- segmentação de referência da imagem de treino;
- imagem de teste;
- segmentação de referência da imagem de teste.

Essas imagens foram divididas em patches.

As imagens de referência determinavam a classificação nas seguintes classes:

- classe 1: building;
- classe 2: tree;
- classe 3: low vegetation;
- classe 4: car;
- classe 5: impervious surface.

3 Experimentos

Realizamos 5 experimentos diferentes descritos a seguir. Procuramos variar os strides entre 16 e 32 para comparar os resultados entre diferentes patch sizes. Entretanto, com o patch-size igual a 128 só foi possível realizar experimentos com o stride igual a 32, uma vez que 16 exigia maior tempo.

Os pesos de cada uma das classes (utilizados na função de entropia "weighted categorical cross-entropy") são:

Classe	<i>Building</i>	<i>Tree</i>	<i>Low vegetation</i>	<i>Car</i>	<i>Imp. surface</i>
Peso	5.67350328	2.97682037	3.92124661	374.0300152	4.34558461

Tabela 1: Pesos das classes

Exp.	Stride	Patch size	Otimizador	Learning rate	Epoches	Batch Size
1	16	32x32	Adam	10^{-3}	100	128
2	32	32x32	Adam	10^{-3}	100	128
3	16	64x64	Adam	10^{-3}	100	128
4	32	64x64	Adam	10^{-3}	100	128
5	32	128x128	Adam	10^{-3}	100	128

Tabela 2: Detalhes do experimento.

4 Resultados

Na tabela 3 abaixo, vemos que o experimento 3 alcançou a maior acurácia, seguido pelos experimentos 1 e 5. O experimento 4 apresentou resultados medianos enquanto o experimento 2 apresentou uma baixa acurácia. As acuráncias se tornam mais evidentes na comparação das imagens obtidas com as originais. O fato do stride de tamanho 32 no patch size 32×32 explica a baixa acurácia apresentada no experimento 2 pois com o stride desse tamanho não há superposição dos pixels.

Experimento	Experimento 1	Experimento 2	Experimento 3	Experimento 4	Experimento 5
Acurácia	67.3131	30.8413	<u>69.8699</u>	52.2497	66.8563

Tabela 3: Acurácia dos Experimentos

4.1 Experimento 1

Métrica	<i>Building</i>	<i>Tree</i>	<i>Low vegetation</i>	<i>Car</i>	<i>Imp. surface</i>
Accuracy	67.3131				
F1 Score	75.7668	72.5362	52.4328	30.8874	72.0625
Recall	77.2846	88.7902	42.1315	24.8983	71.9294
Precision	74.3074	61.3123	69.4020	40.6705	72.1960

Tabela 4: Resultados do Experimento 1

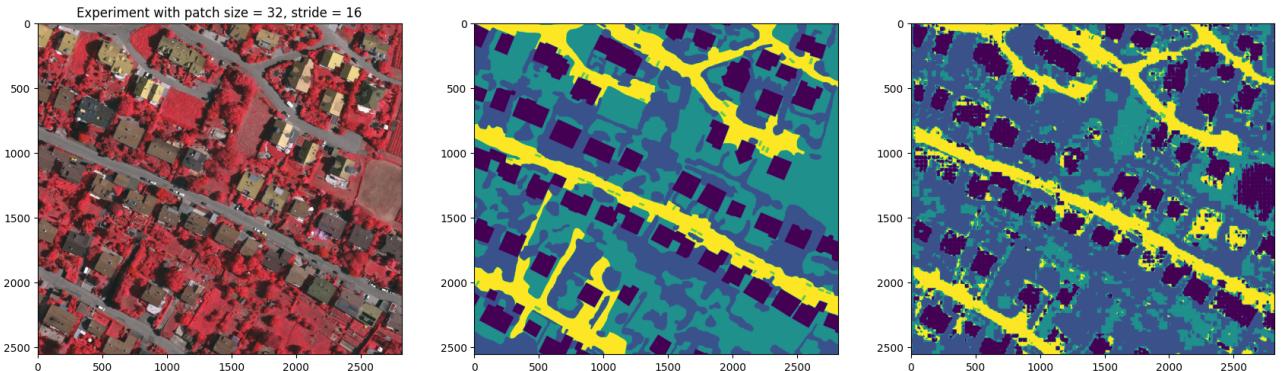


Figura 2: Experimento 1

Interpretação da imagem: Em relação à referência, a segmentação semântica produzida é mais granular. Contudo, apresenta bem os contornos das "impervious surfaces". Podemos ver pela imagem que a nossa classificação errou a "Low vegetation" no meio do lado direito (azul-escuro), classificando-a como "building". Percebemos também uma dificuldade de diferenciar "Tree" de "Low vegetation", já que a maioria das "Low vegetation" são classificadas como "Trees". A classe "Car" é percebida com dificuldade, sendo confundida como "impervious surfaces". Esses resultados se refletem na tabela, já que "Trees" e "Cars" apresentam os menores scores.

Uma imperfeição aparente nessa imagem é o efeito de borda, em que se vê vários quadradinhos na imagem.

4.2 Experimento 2

Métrica	Building	Tree	Low vegetation	Car	Imp. surface
Accuracy	30.8413				
F1 Score	0.7112	47.1947	2.7868	0.1108	11.2833
Recall	0.3597	99.1435	1.4942	0.0555	6.7142
Precision	30.8430	30.9681	20.6515	38.4615	35.3173

Tabela 5: Resultados do Experimento 2

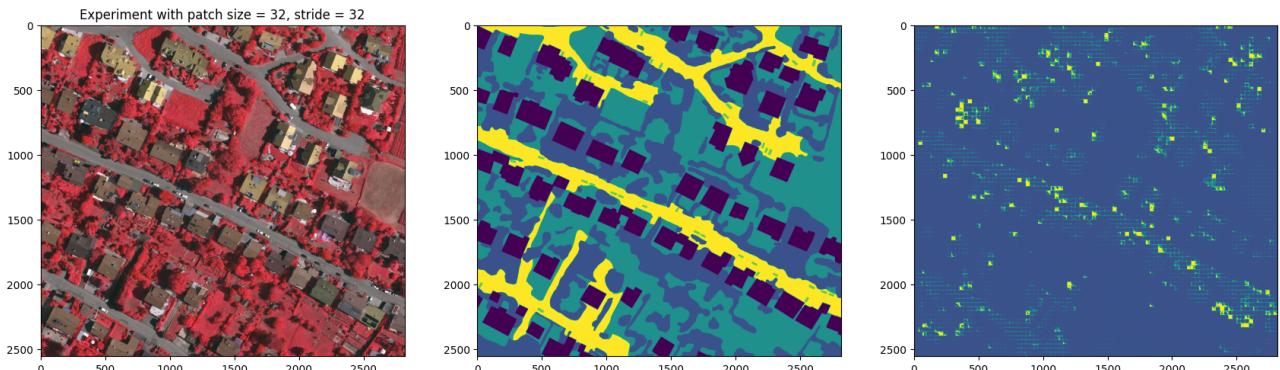


Figura 3: Experimento 2

Interpretação da imagem: Este experimento classificou a maior parte da imagem como "Tree-- justificando um alto valor de "Recall", e embora apresente alguns pontos amarelos correspondentes às "impervious surfaces", têm dificuldade de acertar quase tudo. O fato do stride ser igual ao tamanho do patch fez com que não houvessem pixeis de interseção para detectar as bordas. Nessa imagem também é possível ver vários quadradinhos do efeito de bordas.

4.3 Experimento 3

Métrica	<i>Building</i>	<i>Tree</i>	<i>Low vegetation</i>	<i>Car</i>	<i>Imp. surface</i>
Accuracy	69.8699				
F1 Score	76.0661	73.4842	57.1361	25.7504	77.2497
Recall	73.8863	90.0103	46.7943	15.5610	79.3287
Precision	78.3785	62.0853	73.3458	74.5967	75.2768

Tabela 6: Resultados do Experimento 3

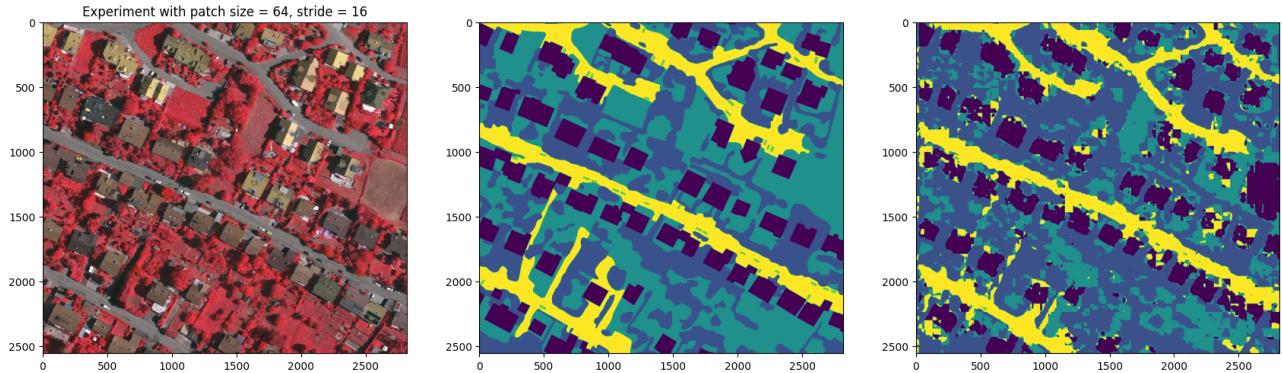


Figura 4: Experimento 3

Interpretação da Imagem de forma geral, este experimento foi semelhante ao primeiro, mas demonstrou-se mais capaz de eliminar mais granulações que o experimento 1, como podemos ver pelos contornos das "impervious Surfaces". Isso se refletiu na maior acurácia em relação ao 1.

4.4 Experimento 4

Métrica	<i>Building</i>	<i>Tree</i>	<i>Low vegetation</i>	<i>Car</i>	<i>Imp. surface</i>
Accuracy	52.2497				
F1 Score	68.0030	58.4500	47.3802	27.5211	0.6906
Recall	58.4177	94.4205	37.6261	26.9507	0.3483
Precision	81.3513	42.3257	63.9613	28.1162	40.2104

Tabela 7: Experimento 4

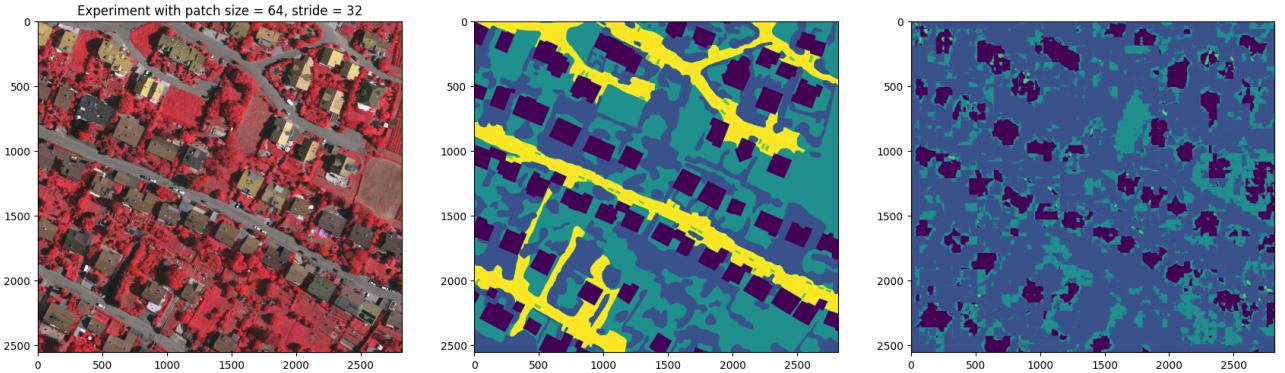


Figura 5: Experimento 4

Interpretação da Imagem: Este experimento classificou a maioria da área como "Trees", conforme vemos pela cor azul-marinho. Isso se reflete no maior Recall para as árvores. Além disso, os "building"s foram detectados em algum grau, refletindo numa maior precision entre as outras classes. As low vegetaions também foram identificadas em parte. De forma geral, os carros e impervious surfaces quase não aparecem na segmentação gerada. Como a área da impervious surfaces é bem maior, isso se refletiu no menor F1 score e recall.

4.5 Experimento 5

Métrica	Building	Tree	Low vegetation	Car	Imp. surface
Accuracy	66.8563				
F1 Score	69.5862	72.3037	53.9834	24.2111	75.4636
Recall	65.5941	88.6303	45.9579	15.9604	75.3080
Precision	74.0958	61.0566	65.4050	50.1219	75.6198

Tabela 8: Experimento 5

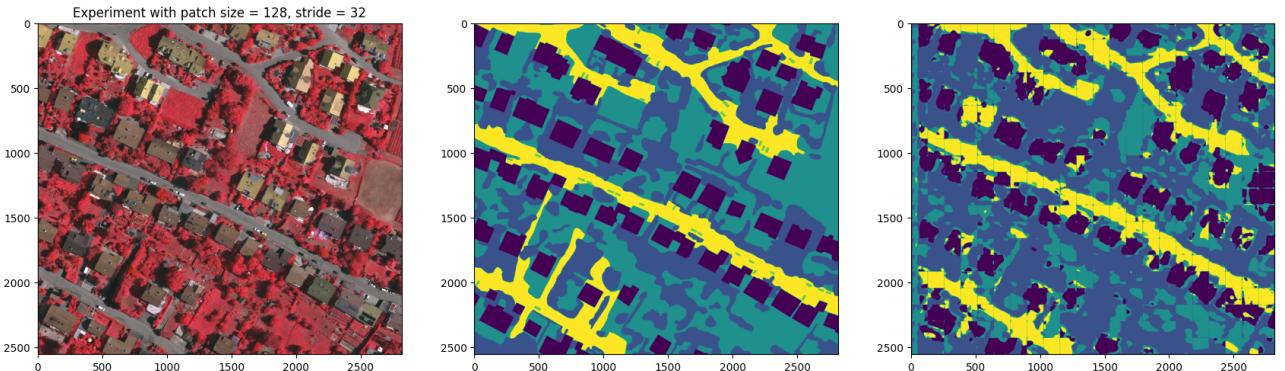


Figura 6: Experimento 5

Interpretação da imagem: de forma geral, detecta bem as classes, com certa granulação. Contudo, confunde mais as bordas de buildings com impervious surfaces. O efeito de borda é menos visível nesse experimento em relação ao experimento 1, devido ao patch size ser maior.

4.6 Piores classes

As piores classes para se detectar (considerando a soma das métricas "F1 Score", "Recall", e "Precision") (uma vez que a melhora ou piora de um dos scores para uma dada classe parece se refletir nos outros) foram:

- Experimento 1: car
- Experimento 2: car
- Experimento 3: car
- Experimento 4: impervious surface
- Experimento 5: car

Considerando a análise visual das imagens, as classes mais erradas foram:

- Experimento 1: car, já que quase não é possível identificar os carros originais da referência;
- Experimento 2: cars, buildings, low vegetation, impervious surfaces, uma vez que a maioria dos patches são classificados como tree.
- Experimento 3: car
- Experimento 4: impervious surface
- Experimento 5: car

Ou seja, as árvores e os carros foram mais difíceis de serem encontrados na segmentação semântica. Isso pode acontecer devido a seus tamanhos.

5 Conclusão

Neste trabalho, exploramos como realizar segmentação semântica utilizando a rede convolucional U-Net, que utiliza uma arquitetura de *encoder* e *decoder*. Percebemos que, com strides menores, obtivemos os melhores resultados de acurácia.

Como foi possível observar nos experimentos, o feito de borda causou uma deterioração da acurácia. Esse processo pode estar associado ao argumento "same" da função `extract_patches` do Tensorflow, que resulta na adição de zeros. Uma maneira de suavizar o efeito de bordas seria fazer a média dos pontos na área de interseção de dois patches vizinhos ao utilizar um stride menor que o patch size na hora de gerar a imagem.

Referências

- [1] Olaf Ronneberger, Philipp Fischer e Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". Em: *Medical Image Computing and Computer-Assisted Intervention-MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III* 18. Springer. 2015, pp. 234–241.