



Relatório da Atividade 3: Segmentação Semântica

1. Objetivo

Este trabalho tem como objetivo a aplicação e estudo de uma rede neural convolucional (CNN) aplicada à Segmentação Semântica.

2. Desenvolvimento

Alguns dos aspectos da experiência foram estabelecidos previamente na descrição da atividade. São eles:

- ❖ Ambiente de execução: Google Colab;
- ❖ Biblioteca para execução da CNN: Keras (Tensor Flow);
- ❖ Dataset: Imagem de satélite com recortes selecionados pelo professor da disciplina advindas do concurso de Classificação Semântica 2D do ISPRS (Sociedade Internacional de Fotogrametria e Sensoriamento Remoto) encontrado no link: <https://www.isprs.org/education/benchmarks/UrbanSemLab/semantic-labeling.aspx> ;
- ❖ Código: Parte do código utilizado nesta experiência já foi previamente estabelecido para o treino da CNN utilizando uma imagem '.tiff' de três bandas (Red, Green, Blue), sendo a escolha, aplicação da rede e adaptação do código para utilização de diferentes partes do dataset para treino de total responsabilidade do aluno. O código consiste nas seguintes etapas:
 - i. Importação de bibliotecas e funções a serem utilizadas na atividade;
 - ii. Definição do Google Drive do aluno como pasta de armazenamento para o código a ser executado;
 - iii. Carregamento e normalização das imagens trabalhadas;
 - iv. Criação das classes a partir da imagem de referência;
 - v. Extração dos *patches* (áreas selecionadas) da imagem para treinamento;
 - vi. Distinção entre os patches a serem executados para treinamento e validação da CNN;

vii. Data Augmentation, consistindo basicamente na rotação dos patches das imagens utilizadas no treinamento e suas respectivas referências em 90° e viradas na vertical e horizontal. No código existem outras possibilidades de manipulação dos patches, porém após um treinamento utilizando uma imagem e explorando essas possibilidades disponíveis de data augmentation as diferenças proporcionadas não se mostraram significativas o suficiente para justificar sua utilização nessa atividade em particular;

viii. Codificação pelo aluno do Modelo CNN escolhido (U-Net);

ix. Criação das funções utilizadas para a realização do treinamento da CNN;

x. Definição dos hiperparâmetros a serem utilizados;

xi. Treinamento da CNN utilizando as funções e o modelo previamente codificados;

xii. Exibição de um gráfico com os valores de acurácia e perda do modelo executado para os patches de treinamento e validação ao longo das épocas estabelecidas;

xiii. Extração da segmentação prevista para um *patch* aleatório no treinamento da CNN, com o cálculo da matriz de confusão, a acurácia, a precisão, o *f1 score* e o *recall* para os dados de treinamento e validação;

xiv. Inferência do modelo CNN para teste;

xv. Extração da segmentação prevista para um *patch* aleatório no teste da CNN, com o cálculo da matriz de confusão, a acurácia, a precisão, o *f1 score* e o *recall* para o *patch*;

xvi. Construção e exibição do mosaico a partir dos *patches* de teste, e cálculo da matriz de confusão, a acurácia, a precisão, o *f1 score* e o *recall* para o mosaico.

A rede escolhida para esta atividade foi a U-Net, e o treinamento foi realizado oito vezes. Originalmente esta atividade teria como objetivo o treinamento e avaliação de uma rede CNN com 4 grupos de seleções de imagens para o treinamento: uma imagem 'Train1_Image.tiff', duas imagens 'Train1_Image.tiff' e 'Train2_Image.tiff', uma imagem com quatro bandas 'Train1_Image.tiff' concatenada à 'Train1_DSM.tiff' e por fim duas imagens 'Train1_Image.tiff' concatenada à 'Train1_DSM.tiff' e 'Train2_Image.tiff' concatenada à 'Train2_DSM.tiff' resultando assim em quatro treinamentos, contudo, ao começar a fazer os treinamentos percebeu-se um comportamento peculiar da rede. Na seção ix do código está presente a função de Early Stop, com um hiperparâmetro posteriormente definido de 10 épocas para checagem, porém a presença do mesmo acarretava inicialmente em uma parada prematura do treinamento. Após uma investigação no código da rede implementada, chegou-se à conclusão de que esta situação somente acontecia quando a U-Net era utilizada com 'BatchNormalization()'. Desta forma decidiu-se estudar o comportamento da rede com ambas as situações: utilizando do Early Stop numa U-Net simples e utilizando de um Early Stop modificado de forma a forçar o

treinamento por todas as épocas tendo o 'BatchNormalization()' presente após cada camada de convolução da CNN.

Os hiperparâmetros utilizados nesta atividade foram previamente definidos pelo professor, tendo como exceção os pesos de cada classe. Estes por sua vez foram estabelecidos por meio de tentativa e erro a partir do cálculo da presença de cada classe na imagem de referência (este cálculo se encontra na seção iv do código). Os hiperparâmetros utilizados são explicitados a seguir:

```
batch_size = 16
epochs = 100
class_weights = [0.10, 0.05, 0.10 , 0.05, 4.00]
early_stopping_epochs = 10
early_stopping_delta = 0.0001
data_augmentation = True
number_samples_for_generator = 4
adam = Adam(learning_rate = 0.0001 , beta_1=0.9)
```

Cada um destes hiperparâmetros é explicado na subseção 3.5

Na próxima seção são apresentados alguns conceitos teóricos que são relevantes para este relatório.

3. Conceitos Teóricos

3.1. Segmentação Semântica

Na classificação de imagem é atribuído um rótulo para cada imagem de entrada. Em contrapartida a segmentação semântica atribui um rótulo para cada pixel da imagem de entrada. Isso significa que a imagem vai ser “fragmentada” em classes onde a segmentação semântica acaba não só identificando o objeto mas também onde o mesmo se encontra na imagem. Deve-se notar que diferentemente da segmentação de instâncias, a segmentação semântica identifica classes de objetos e, por tanto, não diferencia entre os objetos (instâncias) de uma mesma classe. A Figura 1 exemplifica o que a segmentação semântica faz em relação a segmentação de instâncias ao identificar a classe ‘balões’ ao invés de cada balão.

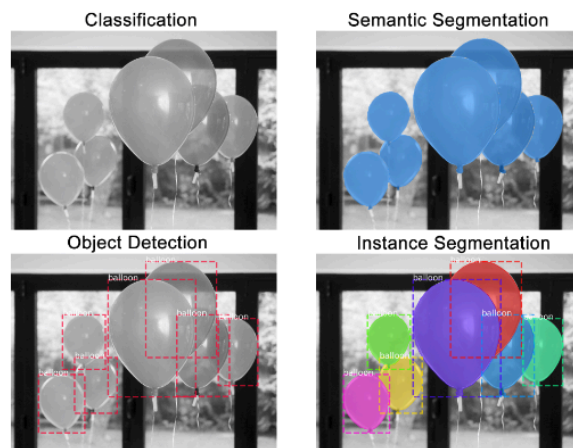


Figura 1: Segmentação Semântica x Segmentação de Instâncias

Ao trabalhar com segmentação semântica de imagens pode ocorrer, como é o caso deste trabalho, da imagem a ser processada ser muito grande. Para tanto, essa imagem pode ser tratada por inferência em *tiles*. Neste caso deve-se ter um cuidado com a disposição destes *tiles* na imagem pois as informações dos objetos presentes nas bordas normalmente são insuficientes para uma classificação satisfatória. Para remediar este problema os *tiles* são dispostos com uma porcentagem de sobreposição (*overlap*) na imagem. Assim, as informações das bordas são descartadas e o centro de cada *tile* é processado e os resultados destes processamentos formam o mosaico da imagem original processada. Um exemplo disto é visto na Figura 2.

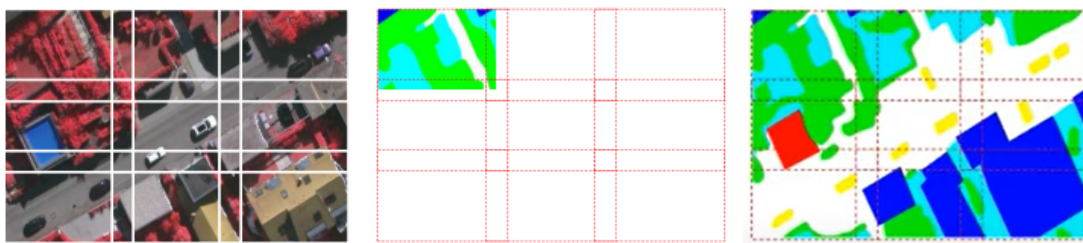


Figura 2: Esquema de processamento em *tiles*

Para a implementação da segmentação semântica neste trabalho foi escolhido a utilização da rede convolucional U-Net, que será discutida na próxima subseção.

3.2. Rede U-Net

Uma das redes mais famosas, a U-Net, foi a primeira rede a utilizar a ideia da convolução transposta e da chamada *skip-connections*. Sua arquitetura é simétrica, formando um “U” (daí o nome U-Net), onde o *encoder* segue o raciocínio de uma típica rede convolucional porém sem as camadas serem *fully connected*. A ideia por trás da U-Net é ter um grande número de canais com informações no momento de *upsampling*. Ao examinar a Figura 3, percebe-se que a cada *max pooling*, a camada vai ter sua resolução diminuída, ou seja, acontece que a camada vai ficando mais rica semanticamente porém mais pobre em relação às informações

espaciais. A U-Net então propõe o armazenamento destas informações da camada antes do *max pooling* e concatena-a à camada com sua respectiva resolução logo após a convolução transposta (up-conv) na fase do *decoder*. Desta forma, aquela subsequente etapa de convoluções do *decoder* vai ter dados adicionais que auxiliam na recuperação das informações referentes à resolução espacial da imagem.

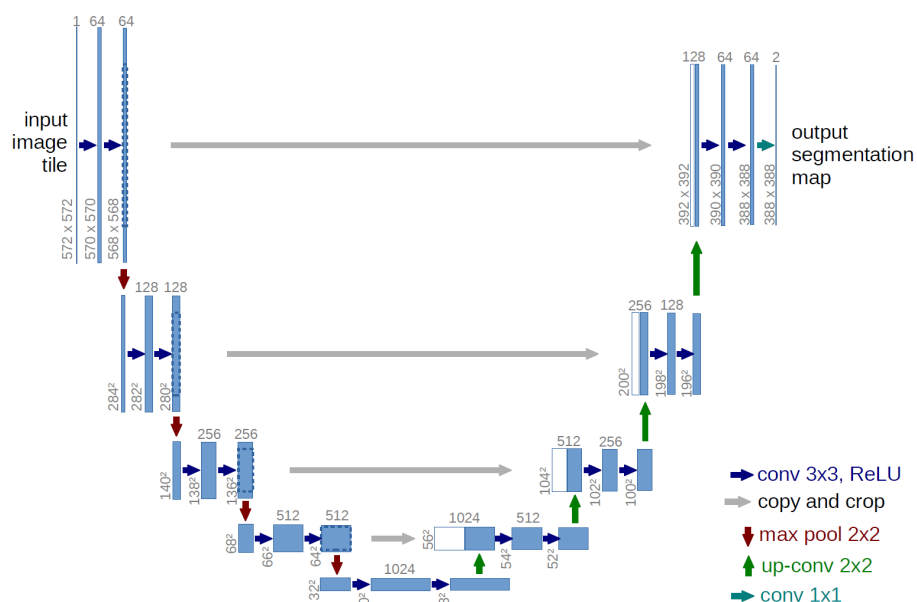


Figura 3: U-Net

A Figura 3 apresenta a arquitetura da U-Net, porém a mesma pode ter funções como *dropout* ou *batch normalization* incluídas em seu modelo. A presença (ou falta) desta última no modelo é discutida nesta atividade, e por tanto uma breve apresentação de seu conceito é feita na próxima subseção.

3.3. Batch Normalization (BN)

Primeiro deve-se entender para que serve a função *BatchNormalization()*. BN tem como objetivo suavizar a função de perda, permitindo assim que se trabalhe com *learning rates* maiores e, de maneira resumida, ela funciona como uma normalização das ativações.

A BN é realizada para cada dimensão individualmente:

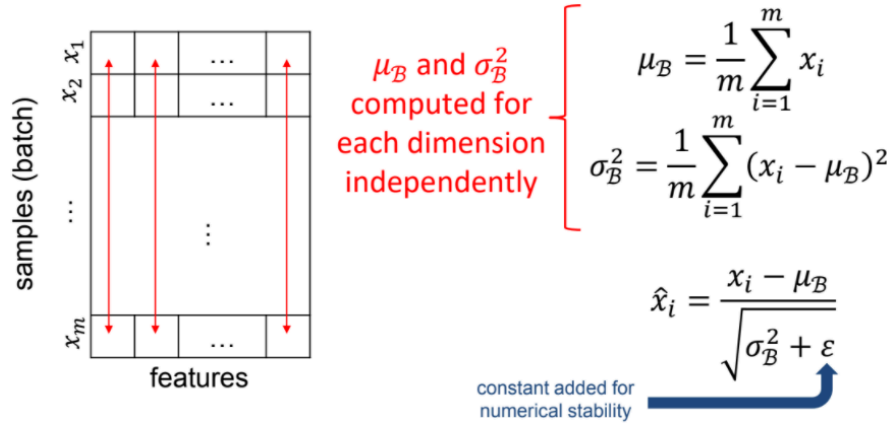
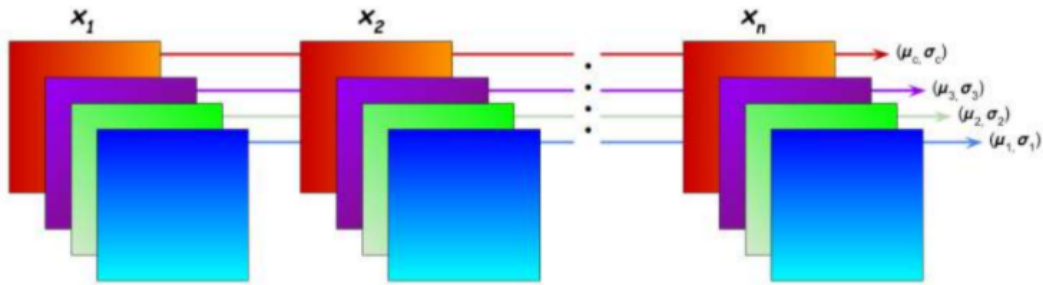


Figura 4: Equações de batch normalization em um vetor

Onde X_i é a entrada, \hat{X}_i é a normalização do vetor, σ^2 é a variância, μ é a média e ϵ é uma constante pequena que impede o denominador de zerar.

Para matrizes, ou mais especificamente imagens (normalmente com vários canais), tem-se que a média e a variância são calculados para cada canal ao longo de todas as amostras em ambas as dimensões espaciais:



$$\mu_c = \frac{1}{m \times h \times w} \sum_{i=1}^m \sum_{j=1}^h \sum_{k=1}^w x_{icjk}$$

$$\sigma_c^2 = \frac{1}{m \times h \times w} \sum_{i=1}^m \sum_{j=1}^h \sum_{k=1}^w (x_{icjk} - \mu_c)^2$$

$$\hat{x}_{icjk} = \frac{x_{icjk} - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}}$$

Figura 5: Fórmulas e esquema de batch normalization para imagens de vários canais

Onde m se refere às amostras do batch, h e w se referem às dimensões espaciais. Por último, é interessante comentar que o processo BN permite por meio de parâmetros aprendidos ao longo do treinamento a possibilidades de ‘desfazer’ a normalização caso a rede não considere-a boa para o problema proposto.

Existem então vantagens proporcionadas na utilização da função *BatchNormalization()*, são elas: a melhora do fluxo do gradiente através da rede,

permite maiores learning rates como comentado anteriormente, reduz a dependência da inicialização dos pesos e por fim, esta função tem um efeito de regularização na rede melhorando as propriedades de generalização, o que por sua vez, ajuda a prevenir *Overfitting*.

3.4. Alguns conceito básicos

Visando auxiliar a debate sobre os resultados obtidos, são apresentados a seguir alguns conceitos básicos:

- *Overfitting*: Ocorre quando o modelo fica demasiadamente especializado com a aprendizagem excessiva de detalhes e ruídos do treinamento. Isso é identificável quando o modelo apresenta uma acurácia de treinamento muito maior em relação a acurácia de validação. Em contrapartida, o valor de erro dos dados de treinamento vai se apresentar muito menor do que o erro dos dados de validação. Por consequência, um modelo sofrendo de *Overfitting* terá uma performance ruim nos dados de teste. Como muitos problemas tratados em deep learning demandam um modelo mais complexo existem várias técnicas que, quando aplicadas, ajudam a prevenir a ocorrência de *Overfitting*.
- *Data Augmentation*: Técnica que ajuda a melhorar a acurácia do modelo e consequentemente, ajuda a prevenir a ocorrência de *overfitting* ao aumentar a quantidade dos dados de treinamento por meio de tratamentos das imagens do dataset (Ex.: rotacionar, comprimir, alongar e etc).
- *Early Stop*: Técnica que observa as amostras pelos conjuntos de treinamento e validação de forma que, ao longo do treinamento do modelo se a performance da validação for deteriorando em comparação a performance do treinamento, ou não for identificado uma melhora significativa dentre essas performances após um período de n épocas o treinamento deste modelo é parado (*early stop*). Nesta técnica é mantido também o melhor resultado até o momento de parada.
- *Matrix Confusion*: Classificador $n \times n$ onde o n varia de acordo com a quantidade de classes a serem identificadas segmentação semântica. A partir dos valores contidos nesta matriz que se pode adquirir as métricas utilizadas na deliberação da performance do modelo.
- *Overall Accuracy*: Também chamada de acurácia global neste trabalho, é uma métrica referente a acurácia geral da classificação realizada pelo modelo. Obtida a partir da equação $OA = (true_positive + true_negative)/all_samples$. Sendo o *all samples* mensurado em pixels. Infelizmente pode levar ao erro e por tanto, outras métricas devem ser utilizadas para auxiliar na avaliação do modelo.

- **Precision:** Métrica onde, cada valor se refere a classe informando quantos itens selecionados são de fato relevantes. Obtida a partir da equação $precision = true_positive / (true_positive + false_positive)$.
- **Recall:** Métrica onde, cada valor se refere a classe informando quantos itens relevantes foram selecionados. Obtida a partir da equação $recall = true_positive / (true_positive + false_negative)$.
- **F1 Score:** Média harmônica da precision com recall. Obtida pela equação $F1 = 2 \times [(precision \times recall) / (precision + recall)]$. Normalmente é uma métrica muito utilizada em casos de segmentação semântica, e que nos casos de desbalanceamento das classes apresentará valores mais confiáveis do que a acurácia global. Vale observar que quando um de seus valores é zerado, significa que $true_positive + false_positive == 0$ e por tanto seu valor de *precision* é indefinido ou pode significar também que $true_positive + false_negative == 0$ e logo, seu valor de *recall* é indefinido. Ambas as situações são caracterizadas pelas colunas ou linhas da *matrix confusion* completamente zeradas, respectivamente.

3.5. Hiperparâmetros

Por fim, nesta subseção é explicitado o que significa os hiperparâmetros utilizados no código. São eles:

- `batch_size = 16`, quantidade de amostras por batch a serem trabalhadas, neste caso 16;
- `epochs = 100`, quantidade de épocas para cada treinamento;
- `class_weights = [0.10, 0.05, 0.10, 0.05, 4.00]`, é um hiperparâmetro com os pesos escolhidos para cada classe de forma arbitrária. Esses pesos são aplicados devido a um problema de desbalanceamento de classes, onde pelo menos uma das classes tem uma presença muito maior ou muito menor do que as outras no dataset de treinamento. Esses pesos tem portanto como objetivo melhorar performance para as classes dando peso maiores para classes má representadas e pesos menores para classes com grande representatividade.
- `early_stopping_epochs = 10`, intervalo máximo de n épocas (onde, neste caso n=10) em que caso não seja identificado uma melhora entre as performances de treinamento e validação, o *early stop* é acionado e o treinamento é interrompido, mantendo o melhor desempenho até aquele momento;
- `early_stopping_delta = 0.0001`, esta é a taxa de melhora do delta aceita pelo *early stop*, neste caso essa taxa é equivalente a 0.01%. Isso significa que qualquer valor menor do que 0.01% na melhoria entre as performances não será considerada como o suficiente para impedir um possível *early stop* ;

- `data_augmentation = True`, informa que ocorre *data augmentation*;
- `number_samples_for_generator = 4`, este hiperparâmetro define a quantidade de amostras a serem trabalhadas no *data augmentation*;
- `adam = Adam(learning_rate = 0.0001 , beta_1=0.9)`, Adam é um algoritmo de otimização da família do gradiente estocástico que, de forma simplificada, combina os algoritmos de otimização AdaGrad e RMSProp, onde os valores de `learning_rate = 0.0001` e `beta_1=0.9` são sugeridos como padrões iniciais em caso de falta de melhores escolhas. O primeiro é uma taxa de aprendizado de 0.01% e o segundo controla a taxa de decaimento da média móvel exponencial do gradiente, o valor inicial próximo de 1.0 resulta num bias dos momentos tendendo para zero (em conjunto com o `beta_2 = 0.999` que é default do algoritmo).

A seguir serão apresentados os resultados obtidos neste experimento.

4. Resultados

O experimento proposto nesta atividade consiste da codificação de uma rede neural convolucional, o estudo de seu comportamento e resultados para diferentes quantidades de dataset de treinamento. Como dito antes, a CNN escolhida a ser aplicada foi a U-Net. Contudo devido ao comportamento peculiar descoberto durante o treinamento, este experimento foi aplicado em duas situações distintas: U-Net simples sem a aplicação da função Batch Normalization e uma U-Net com aplicação da função Batch Normalization após cada Convolução.

Para o primeiro caso, doravante referido como 's/ BN', o modelo gerou um total de 31,082,798 parâmetros, sendo todos eles treináveis. Para o segundo caso, doravante referido como 'c/ BN', foram achados um total de 31,106,366 parâmetros dos quais 31,094,582 são treináveis e 11,784 são não treináveis advindos do BN.

4.1. Curvas e Patches do Treinamento

Para cada treinamento foram gerados dois gráficos ilustrando o comportamento do modelo por meio de duas curvas, uma para o conjunto de treinamento e outra para o conjunto de validação. Sendo os gráficos a esquerda das Figuras 6, 7, 8, 9, 10, 11, 12, e 13 referentes a acurácia do modelo ao longo do treinamento e os gráficos a direita destas mesmas Figuras referentes a perda do modelo durante esse mesmo treinamento. Todos os treinamentos foram realizados onde das amostras utilizadas para cada treinamento 20% foi dedicado ao conjunto de validação e 80% ao conjunto de treinamento. Outro fator importante a ser comentado é de que para treinamento foram utilizadas 16 amostras advindas de data augmentation, geradas a partir de 4 patches aleatórios, ou seja, 4 amostras geradas para cada patch.

Ao analisar os gráficos, o primeiro fator a ser discutido é o comportamento das curvas. No geral, todos os gráficos referentes ao modelo s/ BN (Figuras 6, 8, 10 e 12) apresentam curvas tradicionais onde a acurácia e a perda do conjunto de treino e validação com valores muito próximos, muitas vezes se sobrepondo e, ao

final, com o *early stop* parando o treinamento dentro do intervalo de 65 a 85 épocas. Por outro lado, o modelo c/ BN, apresenta gráficos com diferenças muito grandes entre as curvas de treinamento e validação nos intervalos entre a origem e os pontos de ~32 épocas (Figura 7), ~16 épocas (Figura 9) e ~30 épocas (Figura 11) e um pico maior no ponto de ~55 épocas (Figura 13). Um estudo maior foi feito em cima deste modelo para entender este comportamento. Como três dos quatros treinamentos com BN apresentaram o mesmo comportamento em suas curvas, o código da rede e as funções implementadas foram revistos para que quaisquer possíveis erros de programação fossem descartados, como o comportamento persistiu voltou-se o foco para outros fatores que pudessem explicá-lo.

Como explicado anteriormente, o data augmentation aumenta a variedade das amostras de treinamento por meio de manipulações da imagem. Diminuir a quantidade de amostras não faz sentido pois a pouca quantidade de amostras aumenta a chances do treinamento ter problemas como *Overfitting* e por tanto seria contra-produtivo realizar este treinamento. Em contrapartida, aumentar a quantidade de amostras poderia resultar numa mudança positiva do comportamento, logo um treinamento utilizando 1 imagem (Train1.tiff) e seu DSM com 32 amostras por batch foram realizadas. Infelizmente, a ‘distorção’ que antes só ocorria nas primeira épocas, neste caso se estendeu para todo o treinamento, o que resultou na decisão final de se aplicar o valor inicial de 16 amostras por batch. Os outros valores de hiperparâmetro: epochs, learning_rate, beta_1 e class_weights tiveram seus valores alterados tanto para mais quanto para menos e ainda assim esse comportamento persistiu.

Foi feito então uma pesquisa mais profunda sobre o que é de fato o comportamento de um curva com batch normalization e percebeu-se que essa distorção no início da curva existe em outros experimentos, porém este pico é mais acentuado e menos largo o que leva-se a crer em duas possibilidades: em outros experimento essa distorção não é tão distinguível devido a quantidade de épocas com as quais são feitos esses treinamentos permitindo assim uma ‘compressão’ na representação desse pico devido a quantidade de épocas que normalmente são plotadas no eixo x dos gráficos, e que, como previamente comentado, a variabilidade do dataset tem uma relação íntima com essa distorção inicial. Esta segunda afirmação é corroborada ao compararmos as Figuras 7 e 9, que representam treinamentos com amostras providas de uma imagem (Train1.tiff) e com amostras providas de duas imagens (Train1.tiff e Train2.tiff) respectivamente.

Outro fator interessante observado é que a função *Early Stop* ajuda, devido a estabilização mais rápida do BN, a interromper o treino e assim além de prevenir *Overfitting*, a função auxilia o treinamento a terminar mais rápido. Contudo, para que ambas possam funcionar em conjunto o *Early Stop* deve ser ajustado para considerar o início do treinamento e não causar uma parada prematura.

Considerando que a função *Early Stop* guarda o melhor desempenho do modelo para ser aplicado com os dados de test, decidiu-se configurar a função de forma ao treinamento c/ BN ocorrer ao longo das 100 épocas, porém continuando a

armazenar o melhor modelo, assim provendo gráficos durante as 100 épocas sem afetar o resultado final.

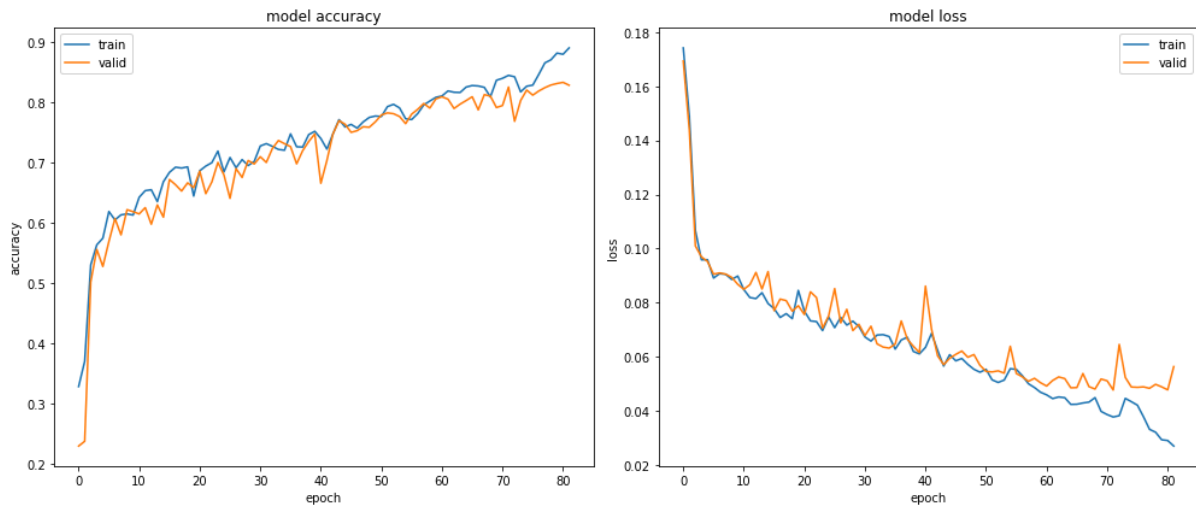


Figura 6: Curva de acurácia e perda do treinamento do modelo sem BN com uma imagem

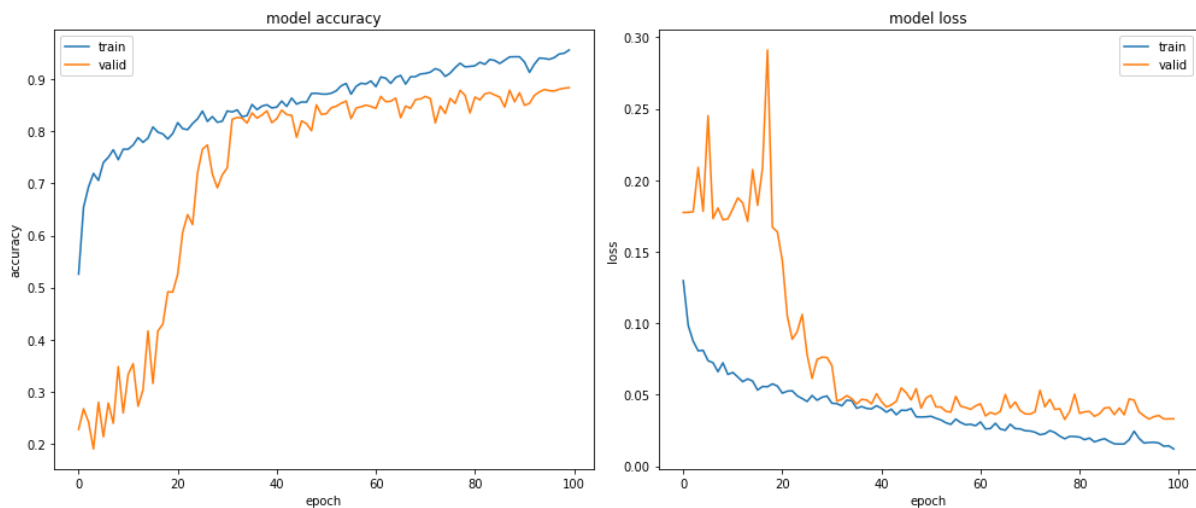


Figura 7: Curva de acurácia e perda do treinamento do modelo com BN com uma imagem

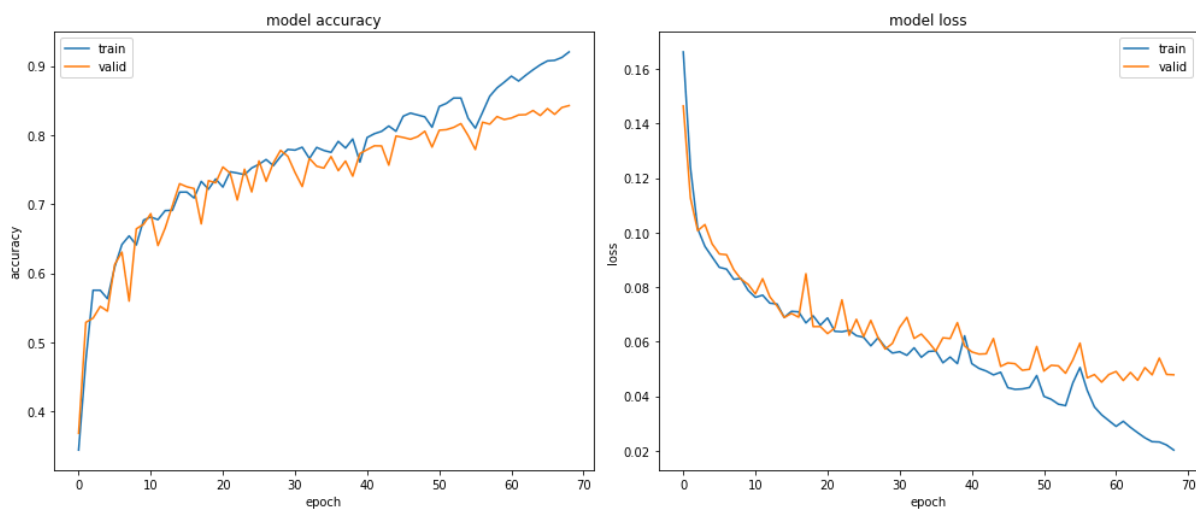


Figura 8: Curva de acurácia e perda do treinamento do modelo sem BN com duas imagens

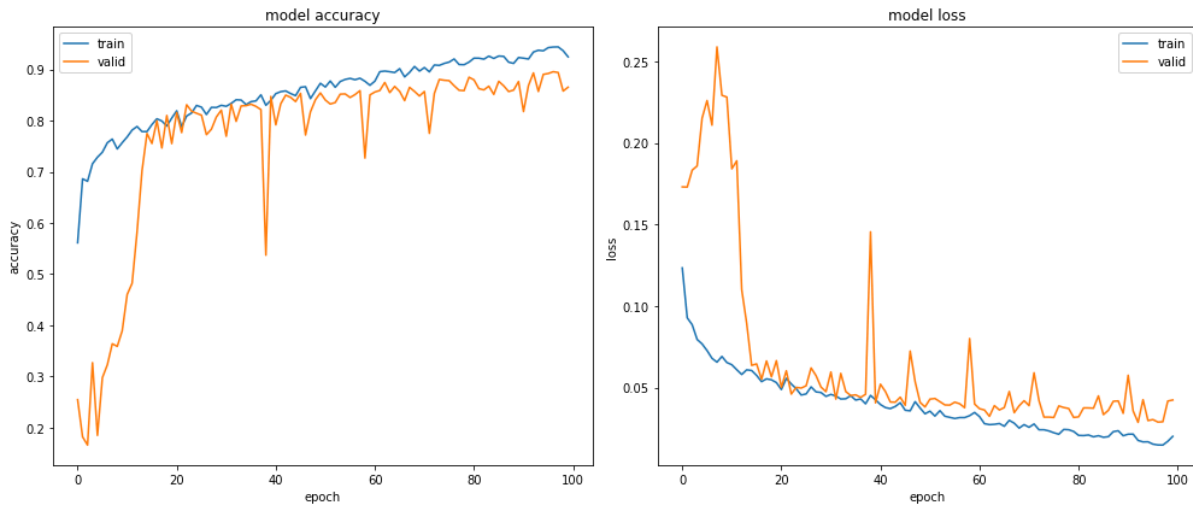


Figura 9: Curva de acurácia e perda do treinamento do modelo com BN com duas imagens

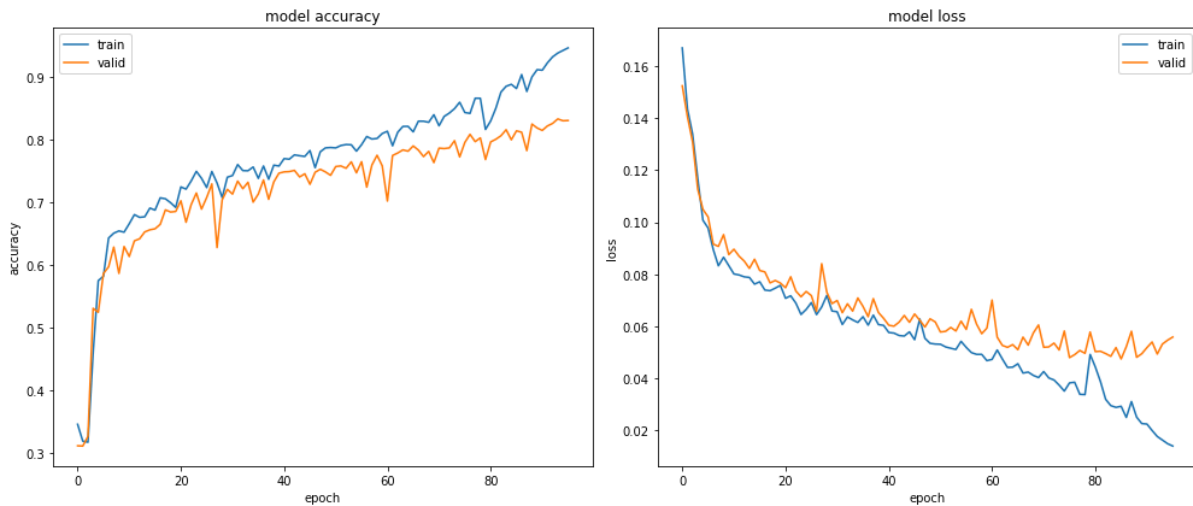


Figura 10: Curva de acurácia e perda do treinamento do modelo sem BN com uma imagem + DSM

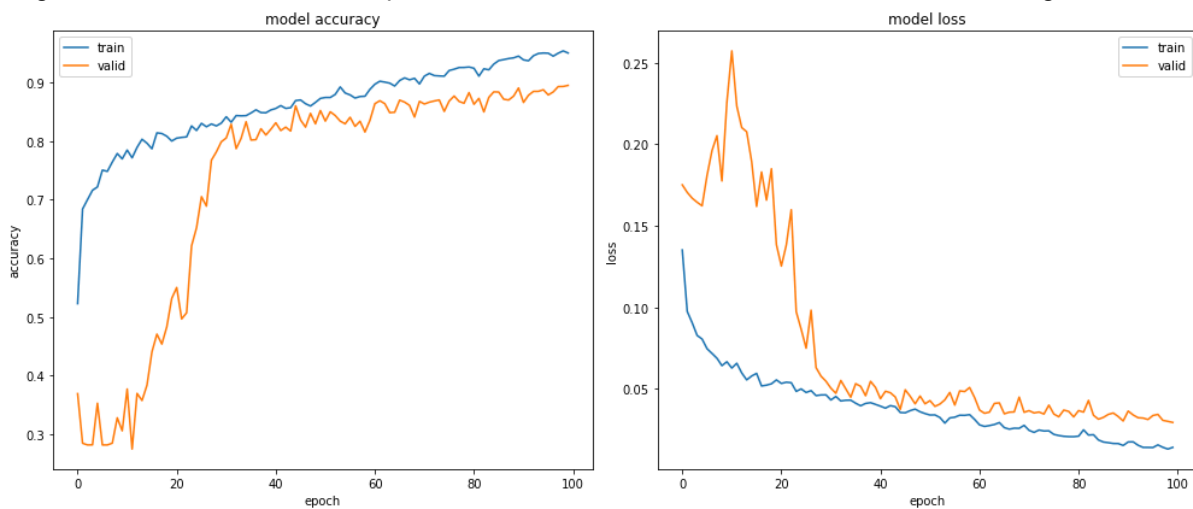


Figura 11: Curva de acurácia e perda do treinamento do modelo com BN com uma imagem + DSM

Deve-se reparar que, dentre todas as curvas apresentadas até este ponto, para treinamentos c/ BN (Figura 7, 9 e 11), a curva de validação estabiliza em um dado momento e então não apresenta um crescimento aparente na perda ou um decrescimento aparente na acurácia, na verdade, o que parece encaminhar o modelo para um *Overfitting* é a curva de treinamento que fica cada vez mais precisa na acurácia e tendendo mais ao zero na perda. Por outro lado, a parada dos treinamentos s/ BN (Figura 6 e 8) apresenta um crescimento da curva de validação nos gráficos de perda e um decrescimento nos gráficos de acurácia.

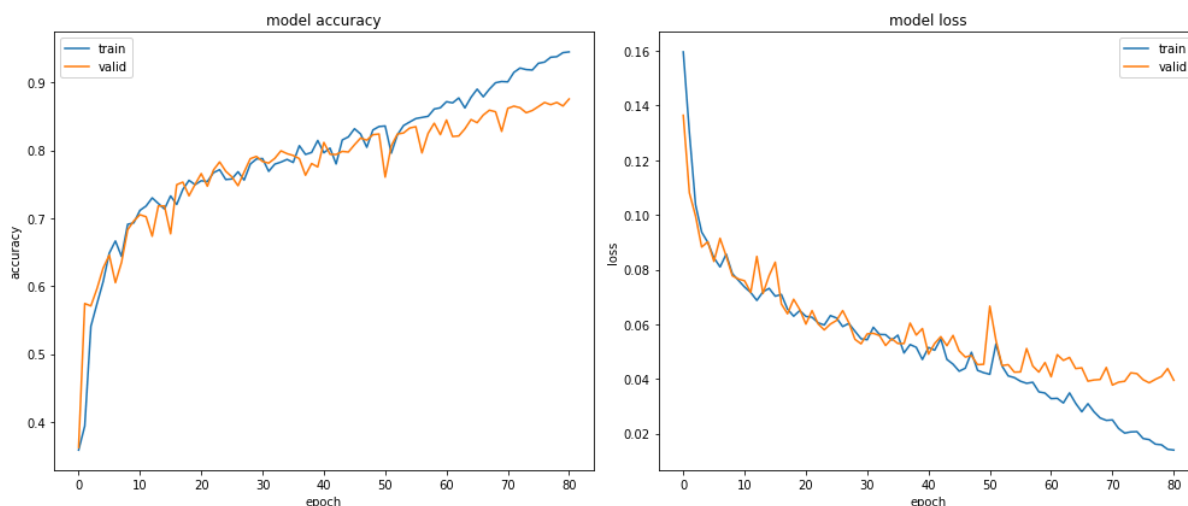


Figura 12: Curva de acurácia e perda do treinamento do modelo sem BN com duas imagens + DSM

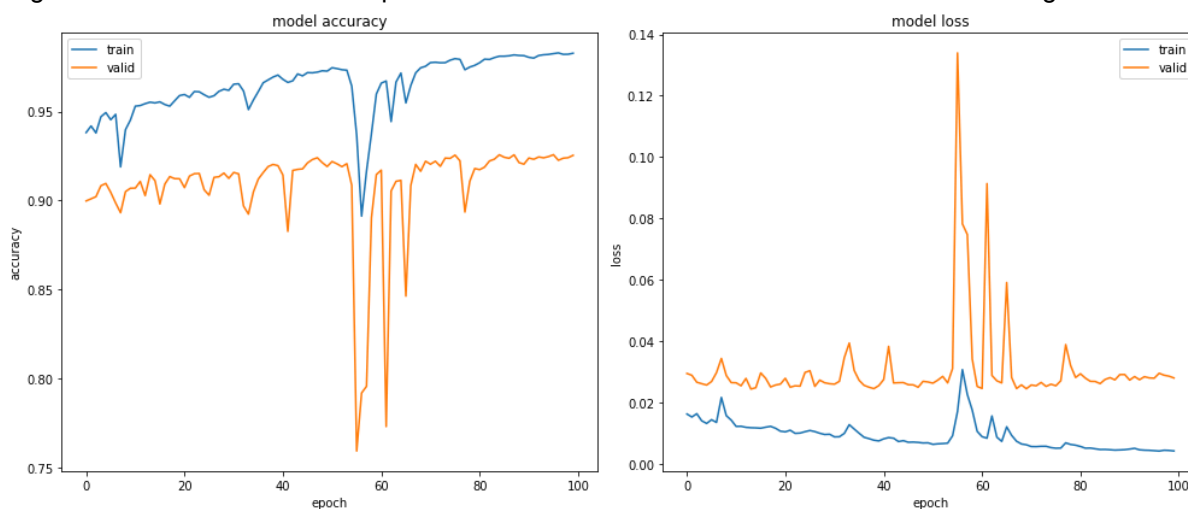



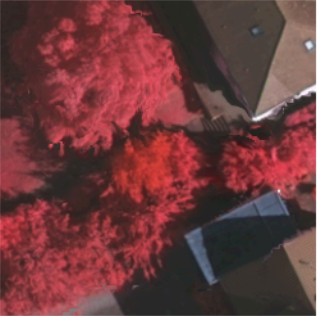

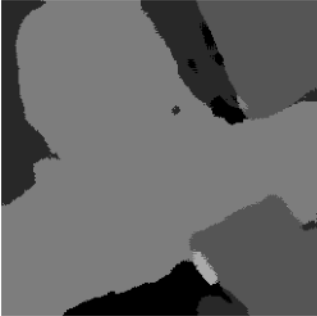





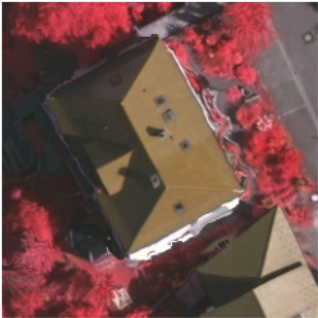








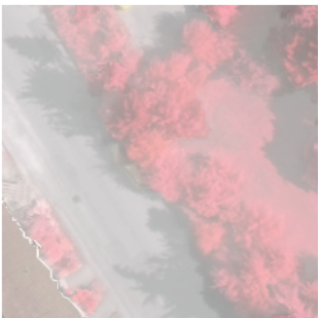


Figura 13: Curva de acurácia e perda do treinamento do modelo com BN com duas imagens + DSM

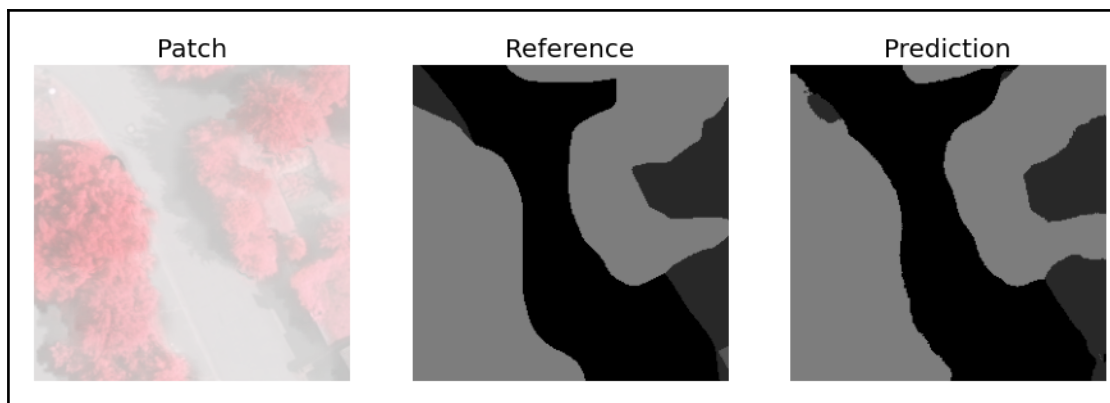
Nota-se alguns 'picos acentuados' (flutuações de valores) ao longo dos gráficos que, enquanto não desejados, podem ser justificados devido a presença de ruídos. Vale observar que caso essa flutuação fosse constante, isso poderia ser indício de que o aprendizado feito durante o treinamento não seria aplicável pelo modelo para o mundo real (data test).

Como visto nos gráficos, todas as curvas de treinamento apresentaram uma acurácia acima de 85%, o que é corroborado pela representação visual dos patch de treinamento no quadro 1.

Quadro 1: Patches de Treinamento

1 img s/ BN		
Patch	Reference	Prediction
		
1 img c/ BN		
Patch	Reference	Prediction
		
2 img s/ BN		
Patch	Reference	Prediction
		
2 img c/ BN		

Patch	Reference	Prediction
		
1 img + DSM s/ BN		
Patch	Reference	Prediction
		
1 img + DSM c/ BN		
Patch	Reference	Prediction
		
2 img +DSM s/ BN		
Patch	Reference	Prediction
		
2 img + DSM c/ BN		



Um dos grandes problemas enfrentados nesta atividade foi a representatividade da classe carro. O patch de treinamento com duas imagens s/ BN ilustra bem essa situação. No patch original da imagem aparecem dois carros, na referência do patch aparecem os mesmos dois carros mas no patch gerado pelo modelo pouquíssimos pixels são declarados como da classe carro. Uma surpresa, contudo, é o fato de que com a presença da banda de DSM nos treinamentos era esperado que as imagens geradas apresentassem melhores classificações, e isso não fica muito nítido nos patches de treinamento.

4.2. Acurácia Global e F1-Score

A Tabela 1 apresenta a acurácia global de todos os treinamentos realizados.

Todos os treinamentos do modelo tiveram uma acurácia próxima ou acima de 80% para o conjunto de dados de treinamento e validação quando o modelo foi aplicado s/ BN e apresentaram uma acurácia próxima ou superior a 90% quando o modelo foi aplicado c/ BN.

Contudo, a acurácia obtida a partir dos testes foi próxima a 70%, quando não menor, em ambos os casos, com uma performance levemente melhor para o modelo aplicando BN. Observando tudo o que foi levantado até o presente momento se deduz que essa diferença ocorre devido ao ajuste de hiperparâmetros como pesos das classes, épocas e taxa de aprendizado.

Tabela 1: Acurácia Global

Global Accuracy (%)	Train 1	Train 1 e 2	Train 1 c/ DSM	Train 1 e 2 c/ DSM
Sem Batch Normalization				
Training Data	84,22	87,29	88,92	90,66
Validation Data	79,92	83,04	81,77	86,00
Test Data (Patch)	69,45	70,36	61,88	68,49
Test Data (Mosaico)	71,56	72,48	63,14	70,28
Batch Normalization				
Training Data	92,34	93,95	95,11	95,48

Validation Data	88,66	89,15	89,17	90,49
Test Data (Patch)	70,69	72,25	63,92	71,25
Test Data (Mosaico)	72,09	74,31	65,78	73,50

No apêndice 7.2 encontram-se as tabelas com os resultados das métricas obtidas para *F1-Score*, *Precision* e *Recall*. Como visto em 3.4, a *F1-Score* é uma métrica que combina *Precision* e *Recall* e portanto os seus resultados discutidos a seguir são referentes ao F1-Score.

Reforçando os valores de acurácia apresentados na acurácia global é claramente perceptível uma maior percentagem para as classes nos conjuntos de treinamento e validação em comparação com os valores relacionados aos patches teste e mosaicos. A classe “carro” foi a classe que nitidamente teve pior percentagem, o que exemplifica como o desbalanceamento entre classes pode afetar os resultados finais em uma segmentação semântica.




Como visto na Tabela 1 os melhores resultados são vistos no modelo c/ BN e isso se replica na tabela de F1-Score (Tabela 2).




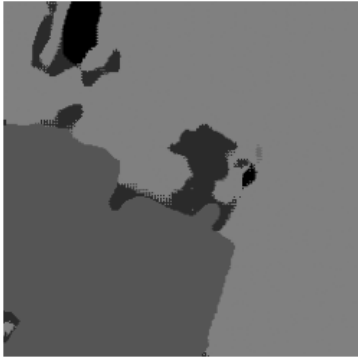
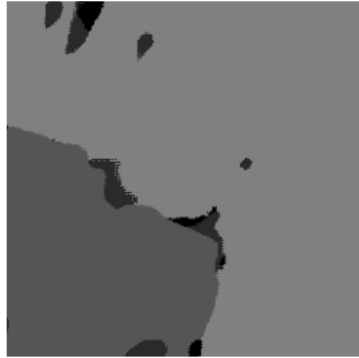

Por fim, os melhores resultados dentre os treinamentos apareceram, em geral, para os treinamentos utilizando duas imagens, o que explicita o fato da variabilidade do dataset de treinamento estar intimamente ligado ao desempenho do treinamento de um modelo.

4.3. Patches Teste e Mosaico

Todos os resultados visuais foram no geral ruins, piorando para os treinamentos com as imagens concatenadas à suas respectivas DSM. Este desempenho corrobora o que foi achado na acurácia global: mesmo que o treinamento tenha apresentado acurácias boas, a aplicação do que o modelo aprendeu nos dados de teste não foi bem sucedido, resultando assim numa segmentação semântica ruim.

Quadro 2: Patches de Teste


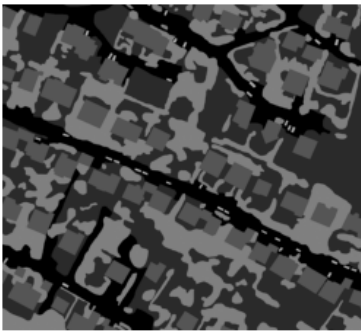
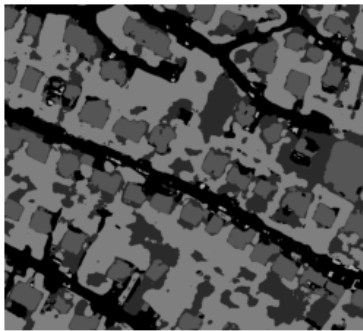
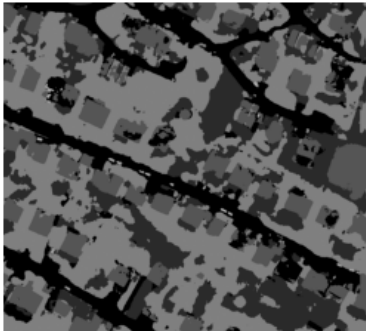
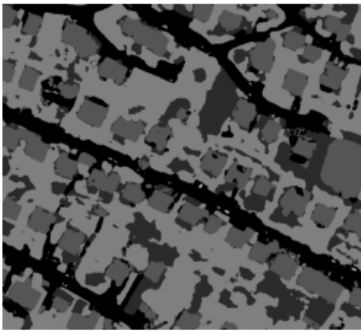
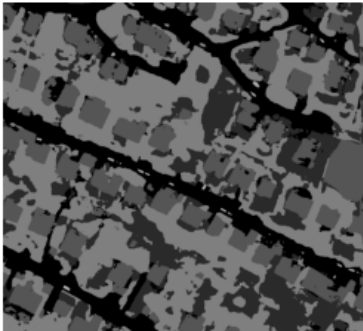
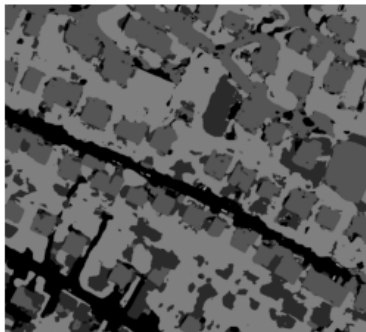
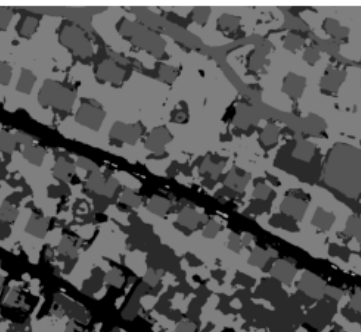
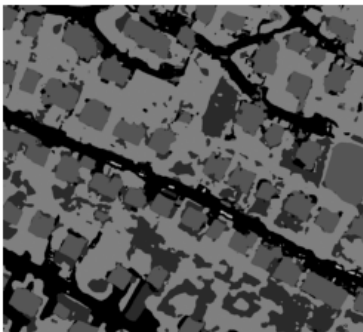
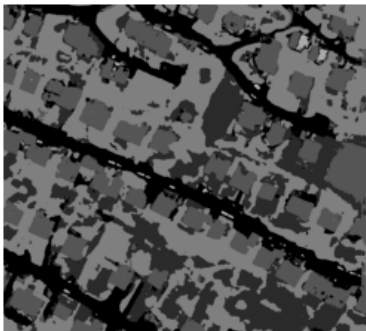
Patch	Referência	1 img s/ BN
		
1 img c/ BN	2 img s/ BN	2 img c/ BN

		
1 img + DSM s/ BN	1 img + DSM c/ BN	2 img + DSM s/ BN
		
2 img + DSM c/ BN		
		

Os mosaicos apresentados no Quadro 3 não só confirmam o que é visto no Quadro 1 como também evidenciam ainda mais o desbalanceamento de classes presente na imagem. O que também corrobora as diferenças de percentagem encontradas no cálculo do F1-Score.

Quadro 3: Mosaicos

Mosaico	Referência	1 img s/ BN
---------	------------	-------------

		
1 img c/ BN	2 img s/ BN	2 img c/ BN
		
1 img + DSM s/ BN	1 img + DSM c/ BN	2 img + DSM s/ BN
		
2 img + DSM c/ BN		
		

5. Conclusões

No geral os treinamentos que apresentaram um resultado melhor no mosaico, valor mais altos de acurácia global e no F1-Score foram aqueles onde o modelo aplica a função Batch Normalization, o que era esperado devido ao seu efeito de regularização.

Ao investigar os patches e mosaico com suas respectivas referências, percebe-se que a própria identificação das classes não é perfeita e muitas vezes adiciona, por exemplo, partes de vegetação baixa a classe de árvores e vice-versa, isso implica em ruídos que dificultam o treinamento do modelo. Considerando o desbalanceamento de classes isso agrava a dificuldade em estabelecer os hiperparâmetros de pesos para cada classe.

Durante toda esta atividade observou-se que os treinamentos utilizando mais de uma imagem obtiveram no geral resultados melhores do que os obtidos com somente uma imagem.

O *data augmentation* por sua vez, provê uma maior diversidade nas amostras utilizadas para o treinamento, contudo, devido a dificuldade de identificação das classes, deve-se cogitar um ajuste nos parâmetros utilizados nesta função a fim de se obter melhores resultados, sabendo que estes ajustes implicam em novos reajustes para os hiperparâmetros caso se mantenha a função Batch Normalization no modelo.

Um dos resultados encontrados que não era esperado foi a perda na precisão nos treinamentos do modelo com presença da DSM no dataset de treinamento. Após várias verificações no código e nos hiperparâmetros não se conseguiu obter uma justificativa lógica para tal comportamento e portanto é necessário uma investigação mais profunda que possa responder essa incógnita.

6. Referências

- Aulas do curso de Deep Learning ministradas pelo Professor Gilson Costa, durante o semestre de 2021/2;
- “Convolutional Neural Network With Tensorflow and Keras”; Medium; 2021. Disponível em: <https://medium.com/geekculture/introduction-to-convolutional-neural-network-with-tensorflow-and-keras-cb52cdc66eaf>; acessado em: 10/02/2022.
- Shah, A.; “Breaking the ice with Batch Normalization”; Medium; 2018; Disponível em: https://medium.com/@anuj_shah/breaking-the-ice-with-batch-normalization-bc41dab8403; acessado em: 16/02/2022
- Brownlee, J.; “How to use Learning Curves to Diagnose Machine Learning Model Performance”; Machine Learning Mastery; 2019; Disponível em: <https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/>; acessado em: 17/02/2022.
- Ronneberger O., Fischer P., and Brox T.; “U-Net: Convolutional Networks for Biomedical Image Segmentation”, Computer Science Department and BIOS Centre for Biological Signalling Studies, University of Freiburg, Germany.

7. Apêndice

7.1. Código

O código utilizado na implementação deste trabalho encontra-se no link abaixo:

https://drive.google.com/file/d/1CU1d833_DzscsvkRe-uXusGgeooH6n86/view?usp=sharing

7.2. Tabelas Auxiliares

Todas as tabelas apresentadas neste relatório explicitam valores com duas casas decimais, onde os valores originais foram obtidos com até oito casas decimais, sendo assim, todos os valores sofreram aproximação e podem apresentar pequenos erros de aproximação caso sejam feitos cálculos manualmente a partir destes dados.

Tabela 2: F1-Score

F1 Score (%)	Train 1	Train 1 e 2	Train 1 c/ dsm	Train 1 e 2 c/ dsm
Training Data				
Sem Batch Normalization				
Superfícies Impermeáveis	91,53	91,46	93,76	93,59
Vegetação baixa	74,52	74,44	81,50	83,34
Prédios/construções	91,81	93,21	94,27	95,44
Árvores	82,61	87,95	88,11	90,26
Carro	43,70	52,98	32,30	60,96
Batch Normalization				
Superfícies Impermeáveis	94,62	95,02	96,26	95,92
Vegetação baixa	89,71	90,17	93,21	93,37
Prédios/construções	95,38	97,11	97,22	97,58
Árvores	91,24	93,68	94,48	95,36
Carro	70,94	81,47	80,60	86,70
Validation Data				
Sem Batch Normalization				
Superfícies Impermeáveis	87,00	88,27	90,59	87,79
Vegetação baixa	66,27	69,04	69,30	74,49
Prédios/construções	88,52	90,01	90,73	91,01
Árvores	78,42	82,42	78,39	88,23
Carro	52,20	50,10	29,57	28,94

Batch Normalization				
Superfícies Impermeáveis	89,41	90,45	90,16	92,01
Vegetação baixa	81,25	76,53	84,47	80,97
Prédios/construções	94,73	95,58	94,61	94,68
Árvores	89,09	89,85	89,49	91,19
Carro	67,93	76,65	65,19	77,86
Test Data (Patch)				
Sem Batch Normalization				
Superfícies Impermeáveis	73,68	74,76	55,43	72,47
Vegetação baixa	55,53	54,83	44,13	50,33
Prédios/construções	81,47	84,11	71,42	82,13
Árvores	72,17	72,30	70,38	71,55
Carro	33,97	24,85	19,04	33,53
Batch Normalization				
Superfícies Impermeáveis	71,49	75,41	53,32	71,81
Vegetação baixa	63,19	60,84	54,43	60,35
Prédios/construções	80,32	82,93	73,77	83,26
Árvores	71,84	73,95	69,61	73,53
Carro	39,37	50,52	39,65	39,90
Test Data (Mosaico)				
Sem Batch Normalization				
Superfícies Impermeáveis	79,47	81,14	57,74	76,51
Vegetação baixa	56,22	55,02	45,80	51,67
Prédios/construções	82,37	84,99	71,15	82,92
Árvores	74,53	74,69	72,34	73,60
Carro	35,94	27,24	20,52	39,57
Batch Normalization				
Superfícies Impermeáveis	76,73	80,97	57,21	78,45
Vegetação baixa	62,10	60,88	56,01	61,65
Prédios/construções	80,56	83,54	73,78	84,19
Árvores	73,90	76,68	72,19	75,45
Carro	42,38	53,78	42,77	41,18

Tabela 3: Recall

Recall (%)	Train 1	Train 1 e 2	Train 1 c/ dsm	Train 1 e 2 c/ dsm
Training Data				
Sem Batch Normalization				
Superfícies Impermeáveis	93,62	92,58	93,89	93,28
Vegetação baixa	75,10	69,84	77,23	82,60
Prédios/construções	91,35	94,04	93,81	96,22
Árvores	81,05	90,04	92,03	90,74
Carro	42,00	40,33	20,19	49,75
Batch Normalization				
Superfícies Impermeáveis	94,61	96,46	96,62	97,32
Vegetação baixa	92,49	87,88	92,98	93,69
Prédios/construções	93,35	96,96	96,88	96,69
Árvores	90,09	94,25	94,64	94,66
Carro	62,43	74,42	76,05	85,61
Validation Data				
Sem Batch Normalization				
Superfícies Impermeáveis	89,00	90,59	90,91	86,52
Vegetação baixa	67,29	67,20	66,46	74,77
Prédios/construções	88,03	90,25	90,39	92,89
Árvores	76,07	82,10	81,83	88,37
Carro	49,21	36,71	18,15	21,08
Batch Normalization				
Superfícies Impermeáveis	89,02	91,46	90,01	93,57
Vegetação baixa	84,90	71,95	83,44	82,32
Prédios/construções	93,51	95,36	94,52	92,35
Árvores	87,67	91,82	90,55	90,81
Carro	68,82	69,90	59,31	81,52
Test Data (Patch)				
Sem Batch Normalization				
Superfícies Impermeáveis	73,56	72,58	45,26	68,75
Vegetação baixa	43,70	41,28	31,11	35,98

Prédios/construções	82,89	91,57	91,04	92,78
Árvores	89,77	91,33	91,78	92,77
Carro	34,68	15,45	10,84	23,22
Batch Normalization				
Superfícies Impermeáveis	69,41	72,37	41,86	68,10
Vegetação baixa	52,40	48,84	43,12	49,58
Prédios/construções	79,41	90,87	87,42	90,65
Árvores	88,46	89,13	88,66	87,59
Carro	31,33	36,70	26,94	37,70
Test Data (Mosaico)				
Sem Batch Normalization				
Superfícies Impermeáveis	86,18	85,33	49,59	78,33
Vegetação baixa	44,32	41,40	32,97	37,30
Prédios/construções	82,68	92,07	90,66	92,67
Árvores	89,47	91,13	90,41	91,97
Carro	42,13	17,14	11,73	28,98
Batch Normalization				
Superfícies Impermeáveis	80,27	85,01	47,31	80,16
Vegetação baixa	50,41	47,92	45,33	50,71
Prédios/construções	79,30	91,50	86,47	89,84
Árvores	89,41	89,77	88,19	87,37
Carro	35,09	40,20	29,53	41,74

Tabela 4: Precision

Precision (%)	Train 1	Train 1 e 2	Train 1 c/ dsm	Train 1 e 2 c/ dsm
Training Data				
Sem Batch Normalization				
Superfícies Impermeáveis	89,53	90,37	93,63	93,90
Vegetação baixa	73,94	79,69	86,28	84,08
Prédios/construções	92,27	92,39	94,75	94,68
Árvores	84,23	85,96	84,51	89,78
Carro	45,54	77,18	80,70	78,69
Batch Normalization				

Superfícies Impermeáveis	94,62	93,63	95,90	94,56
Vegetação baixa	87,10	92,58	93,45	93,06
Prédios/construções	97,49	97,26	97,57	98,48
Árvores	92,41	93,12	94,33	96,07
Carro	82,14	90,00	85,73	87,81
Validation Data				
Sem Batch Normalization				
Superfícies Impermeáveis	85,09	86,06	90,27	89,09
Vegetação baixa	65,29	70,99	72,38	74,22
Prédios/construções	89,01	89,78	91,07	89,21
Árvores	80,93	82,74	75,23	88,08
Carro	55,59	78,87	79,81	46,11
Batch Normalization				
Superfícies Impermeáveis	89,80	89,46	90,30	90,50
Vegetação baixa	77,90	81,73	85,53	79,67
Prédios/construções	95,98	95,80	94,71	97,12
Árvores	90,57	87,96	88,46	91,57
Carro	67,06	84,85	72,36	74,52
Test Data (Patch)				
Sem Batch Normalization				
Superfícies Impermeáveis	73,79	77,07	71,49	76,62
Vegetação baixa	76,15	81,60	75,91	83,72
Prédios/construções	80,11	77,77	58,76	73,68
Árvores	60,34	59,84	57,06	58,23
Carro	33,28	63,60	78,21	60,29
Batch Normalization				
Superfícies Impermeáveis	73,69	78,72	73,43	75,94
Vegetação baixa	79,58	80,66	73,81	77,10
Prédios/construções	81,25	76,27	63,81	76,98
Árvores	60,48	63,19	57,30	63,36
Carro	52,94	81,06	75,07	42,37
Test Data (Mosaico)				
Sem Batch Normalization				

Superfícies Impermeáveis	73,72	77,35	69,10	74,78
Vegetação baixa	76,84	82,01	74,98	84,06
Prédios/construções	82,06	78,93	58,54	75,03
Árvores	63,87	63,28	60,29	61,35
Carro	31,33	66,37	81,72	62,36
Batch Normalization				
Superfícies Impermeáveis	73,49	77,29	72,35	76,81
Vegetação baixa	80,84	83,47	73,27	78,61
Prédios/construções	81,85	76,85	64,34	79,21
Árvores	62,98	66,91	61,10	66,39
Carro	53,50	81,22	77,55	40,63