Avaliação de Redes Neurais para Segmentação Semântica no Contexto de Pulverização Direcionada na Agricultura

Gabriel Bianchin
230217
g230217@dac.unicamp.br

Gustavo Eloi
230218
g230218@dac.unicamp.br

Gustavo Vasconcelos
207873
g207873@dac.unicamp.br

I. Introdução

Atualmente a pulverização agrícola é realizada principalmente de forma generalizada em toda área cultivada, e em muitos casos, aplica-se uma quantidade maior de agroquímicos do que realmente é necessário, tratando toda a plantação com a mesma dose de herbicida [7]. Com o aumento da quantidade de agroquímicos utilizados nas plantações, diversos impactos negativos podem ocorrer, como poluição do meio ambiente, resistência química em pragas e doenças e danos na saúde dos consumidores [5].

A utilização de pulverizadores equipados com sistemas de aplicação direcionada é uma das soluções para esse problema [6]. Uma variante desse sistema é baseado em visão computacional, o qual realiza a captura de imagens da região alvo, processa a informação realizando a segmentação semântica das imagens e retorna o comando para os bicos de pulverização para serem ativados apenas nas regiões de relevância.

Entretanto, devido a necessidade de um ágil tempo de resposta para ativar os bicos de pulverização, redes neurais leves e com alto *framerate* são essenciais para essa aplicação. Nesse sentido, esse projeto tem como objetivo a avaliação de diferentes arquiteturas de redes de segmentação semântica que oferecem velocidade próxima ou acima de tempo-real.

As próximas seções estão organizadas da seguinte forma: a Seção II apresenta os conceitos utilizados para a realização do trabalho, a Seção III apresenta a metodologia aplicada nos experimentos, os resultados são apresentados na Seção IV e a conclusão do trabalho é apresentada na Seção V.

II. Fundamentação Teórica

Nesta seção é apresentada a fundamentação teórica utilizada no trabalho, dividida em arquiteturas das redes e as funções de perda.

A. Arquiteturas das Redes

Nesta subseção são apresentadas as arquiteturas utilizadas no trabalho.

1) Bisenet: A Bisenet é uma rede de segmentação semântica para inferência acima de tempo real, que emprega duas principais abordagens, *spatial path* e *context path* [11]. *Spatial path* é um método proposto para preservar o tamanho espacial da imagem e codificar informações

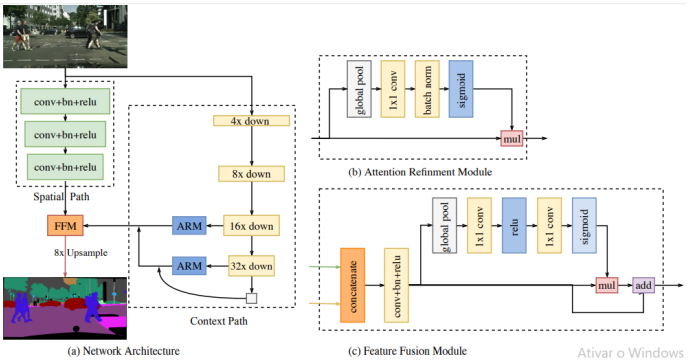

Figure 1. Arquitetura da Bisenet. Fonte: [11].

espaciais em alta resolução. O *context path* foi projetado para extrair informações através de um grande campo receptivo. Na segmentação, o campo receptivo é importante para a contextualização semântica. A arquitetura, devido uma estratégia rápida de *downsampling*, exige menos custo de computação. A arquitetura da Bisenet é apresentada na Figura 1.

2) U-net: A rede U-net é uma rede convolucional para marcar as classes que cada um dos pixels da imagem pertence. A arquitetura da rede pode ser vista na Figura 2. A rede possui duas partes: a parte de contração, que fica a esquerda, e a parte de expansão, que fica a direita da Figura 2. Na parte de contração, a rede faz etapas de convolução e *pooling*, aumentando a quantidade de características. Já na parte de expansão, são realizadas convoluções do tipo *up-convolution*, diminuindo a quantidade de características por canais.

Durante a expansão, as camadas recebem uma concatenação dos mapas de características das camadas da etapa de contração. Isto ocorre devido a perda de pixeis da borda das imagens durante todo o processo [9].

3) E-net: E-net é uma rede convolucional para a segmentação semântica de imagens. Esta rede possui um *encoder* para diminuir a dimensionalidade das imagens de entrada e um *decoder* para retornar as imagens ao tamanho original. No *encoder*, primeiramente é realizado o *pooling* seguido de convolução, fazendo com que seja possível aplicar maiores quantidades de filtros com custo computacional menor. Esta

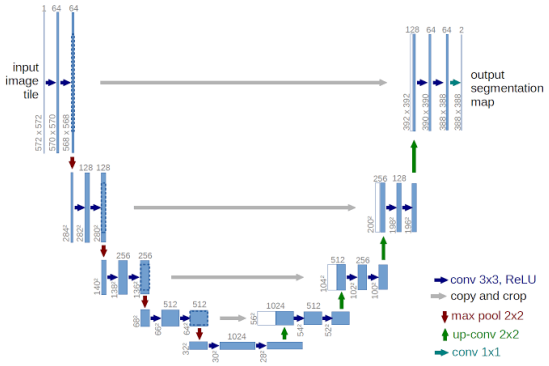

Figure 2. Arquitetura da U-net. Fonte: [9].

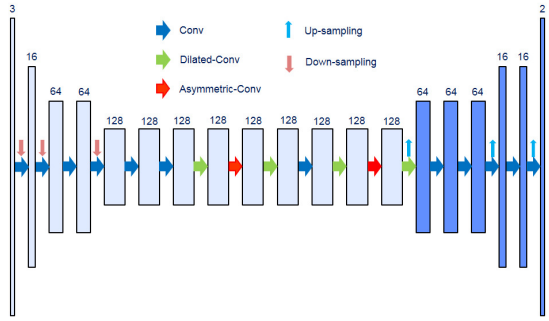

Figure 3. Arquitetura da E-net. Fonte: [4].

técnica permite que a rede seja eficiente e consiga analisar diversas imagens por segundo [8]. A arquitetura da E-net é apresentada na Figura 3.

4) PSPNet: A rede PSPNet[1] utiliza o contexto global para segmentar os pixels da imagem. Inicialmente, dada uma imagem de entrada, é realizada a extração de características da imagem com a rede pré-treinada ResNet [3]. Após a construção do mapa de características, a rede utiliza *pyramid pooling*, que realiza o *pooling* de diferentes tamanhos na imagem, seguido de convolução em cada um dos *poolings*. Esta técnica permite obter o contexto global da imagem [12]. Ao final, é feita a concatenação dos resultados da ramificação do *pyrimid pooling*. A arquitetura da PSPNet é mostrada na Figura 4.

5) Upernet: A Upernet[2] é uma arquitetura projetada objetivando reconhecer o maior número de conceitos visuais em uma imagem. A arquitetura é baseada na *Feature Pyramid Network* (FPN), o qual explora representações de recursos em vários níveis de hierarquia piramidal [10]. São usadas com conexões laterais para fundir informações semânticas de alto nível em níveis médios e baixos. A arquitetura da Upernet pode ser vista na Figura.

[1] *Pyramid Scene Parsing Network*
[2] *Unified Perceptual Parsing for Scene Understanding*

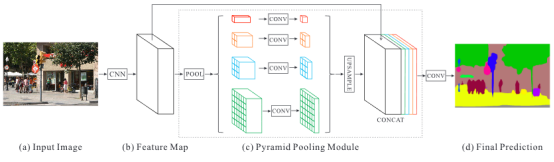

Figure 4. Arquitetura da PSPNet. Fonte: [3].

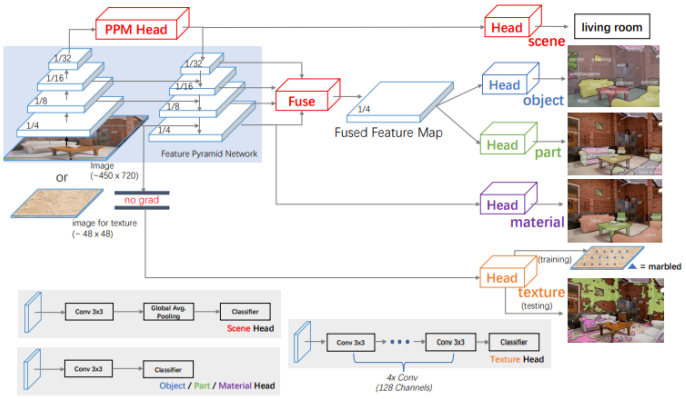

Figure 5. Arquitetura da Upernet. Fonte: [10].

B. Funções de perda

Nesta subseção são apresentadas as funções de perda utilizadas no trabalho.

1) Cross Entropy: A função de perda *cross entropy* indica a distância entre o que o modelo acredita que a distribuição de saída deve ser e qual é realmente a distribuição original. É usado quando as ativações das camadas de saída da rede podem ser entendidas como representações de probabilidade de que cada classe ser verdadeira, ou seja, quando a saída é uma distribuição de probabilidade. Assim, é normalmente usada como uma função de perda em redes neurais que possuem ativações *softmax* ou *sigmoid* na camada de saída.

2) Focal Loss: Essa função de custo é projetada para melhorar o desempenho quando há desequilíbrio na quantidade de instâncias para cada classe. Ela é uma reformulação da função de custo *cross entropy* padrão, sendo que o peso do custo para exemplos bem classificados é reduzido. A *Focal Loss* evita que um grande número de negativos fáceis sobrecarregue o detector durante o treinamento.

C. Métricas de Avaliação

Para a avaliação das redes, utilizamos as métricas acurácia, precisão, revocação, F1 e quadros por segundo (*frames per second* – FPS).

A acurácia é representada pela Equação 1, onde VP é o número de verdadeiros positivos, FP é o número de falsos positivos, P é o número de casos positivos e N é o número de casos negativos. Essa métrica obtém a porcentagem de decisões corretas atingidas pelo classificador.

$$\text{Acurácia} = \frac{\text{VP+VN}}{\text{P+N}} \quad (1)$$

A precisão é representada pela Equação 2, onde VP é o número de verdadeiros positivos e FP é o número de falsos

positivos. Esta métrica avalia a capacidade de classificar como positivo um dado que realmente é positivo.

$$\text{Precisão} = \frac{\text{VP}}{\text{VP+FP}} \tag{2}$$

O cálculo da revocação é dada pela Equação 3, onde VP é o número de verdadeiros positivos e FN é o número de falsos negativos. Esta medida verifica a capacidade do classificador de classificar corretamente dados positivos.

$$\text{Revocação} = \frac{\text{VP}}{\text{VP+FN}} \tag{3}$$

A taxa F1 é mostrada pela Equação 4. A taxa F1 corresponde à média harmônica da precisão e da revocação.

$$\text{F1} = 2 \times \frac{\text{Precisão} \times \text{Revocação}}{\text{Precisão} + \text{Revocação}} \tag{4}$$

O cálculo FPS é apresentado na Equação 5. O FPS é capaz de mostrar quantas imagens a rede consegue analisar por segundo.

$$\text{FPS} = \frac{\text{Quantidade de imagens analisadas}}{\text{Tempo em segundos}} \tag{5}$$

III. Materiais e Métodos

Nessa seção serão descritos a base de dados, a metodologia e as métricas de avaliação.

A. Base de Dados

A base de dados que utilizamos para realização de experimentos é o Agriculture Sugar Beet. Esta base de dados foi obtida a partir de dados recolhidos 3 vezes por dia durante 6 semanas, em uma fazenda de beterraba próxima a cidade de Bonn, Alemanha [1].

Robôs agrícolas foram utilizados para adquirir as imagens da plantação desde o início do nascimento das plantas até o momento em que a passagem do robô começou a atrapalhar o crescimento da plantação. Cada robô possuía câmeras com 4 canais espectrais com sensores RGB e infravermelho próximo, além de sensores para mensurar localização, navegação e mapeamento. A Figura 6 apresenta o robô utilizado para obter as imagens.

Ao todo, a base possui cerca de 5 TB de dados, entretanto, nesse projeto apenas as imagens em RGB serão utilizadas. Primeiramente iniciamos o projeto com uma análise detalhada do *dataset* Agriculture Sugar Beet[3]. Analisamos que existiam algumas imagens que não haviam anotações, sendo assim, 1506 imagens das 13870 foram retiradas do conjunto. Após o filtro das imagens, o *dataset* passou a conter 12364 imagens com suas respectivas anotações. A Figura 7 ilustra um exemplo de imagem encontrada na base.

Para que os experimentos não fossem enviesados pela diferença entre domínios, como o tamanho das plantas e iluminação, antes do início dos testes separamos aleatoriamente o *dataset* em três partes todos contendo todas os tamanhos e fases de crescimento das plantas. O conjunto de treino recebeu 50% das imagens, sendo assim, possui exatamente 6182 itens, enquanto a outra metade foi dividida entre validação e teste, com 3091 imagens cada. Para garantir uma análise justa entre as redes, o conjunto de dados foi separado igualmente para todos os experimentos, e nenhuma aumentação foi realizada.

[3]http://www.ipb.uni-bonn.de/datasets_IJRR2017/

Figure 6. Robô utilizado para captura das imagens em campo. Fonte: [2].

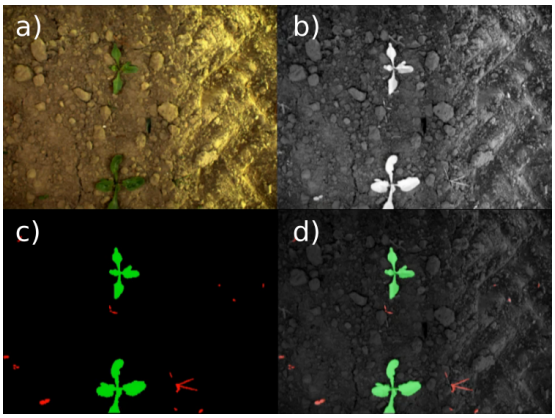

Figure 7. Exemplo de imagem encontrada na base de dados. a) Imagem em RGB. b) Imagem em infravermelho próximo. c) Anotação. d) Sobreposição.

B. Experimentos

Todos os experimentos foram realizados em uma GPU Nvidia RTX 2070 de 8Gb de memória com dupla precisão no formato de ponto flutuante. As imagens de entrada têm apenas canais RGB e foram redimensionadas para a dimensão de 512x384. Essa resolução foi escolhida copiando a utilização em um outro trabalho que utiliza o mesmo *dataset*, permitindo assim uma comparação direta dos resultados com outros autores [6]. Foi escolhida a normalização dos *pixels* entre 0 e 1, dividindo-se os valores de 8-bits por 255. Além disso, cada rede descrita anteriormente foi trainada utilizando como tamanho de *batch* o maior número múltiplo de 2 de imagens que cabiam na GPU. Todas as arquiteturas utilizadas foram desenvolvidas utilizando o *framework* pytorch provenientes de

repositórios do github, a rede Bisenet do repositório BiSeNet[4], e as outras redes do repositório pytorch_segmentation[5]. A rede U-net, inicialmente havia sido extraída de um repositório em keras, como apresentado no baseline, entretanto, para uma maior igualdade nos experimentos, resolveu-se posteriormente utilizar a arquitetura do repositório em pytorch. Os tipos de experimentos realizados neste trabalho são descritos a seguir.

1) Experimento 1 - Modificação na função de custo: No primeiro experimento realizou-se a modificação da função de custo para todas as redes exceto a Bisenet, pois no repositório utilizado não havia essa opção. As funções de custo utilizadas foram a *cross entropy* e a *focal loss*. O objetivo desse experimento é verificar se há ganho nas métricas quando utilizamos a *focal loss*, devido ao desbalanceamento das classes no conjunto de dados.

2) Experimento 2 - Variação no número de imagens para treinamento: No segundo experimento, o objetivo é analisar a resposta da rede quando exposta a um dataset reduzido para treinamento. Portanto, todas as redes foram treinadas com o dataset completo, ou seja, 6182 instâncias de treinamento e posteriormente treinadas com praticamente a metade das imagens, com um conjunto de treinamento de 3000 instâncias. A justificativa para esse experimento se dá devido a dificuldade de se obter dados anotados para segmentação semântica, sendo assim, uma rede que consegue se adaptar melhor a um conjunto de dados reduzido é preferível. Nesse experimento, as redes Upernet e PSPNet foram treinadas com o *backbone* da resnet50, e a Bisenet com o *backbone* da resnet18, pois eram as menores arquiteturas disponíveis para treinamento.

3) Experimento 3 - Utilização de pesos pré-treinados no backbone: O objetivo do terceiro experimento é analisar se a utilização de pesos pré-treinados no dataset ImageNet resulta em algum ganho nas métricas quando comparado com as redes treinadas do zero. Nesse experimento foram utilizadas as redes Upernet e PSPNet utilizando o *backbone* resnet50 para comparação.

4) Experimento 4 - Variações na arquitetura do backbone: No quarto tipo de experimento foi modificado os *backbones* das redes Bisenet, Upernet e PSPNet para analisar a variação nas métricas quando utilizamos uma arquitetura de maior profundidade, e qual o impacto na velocidade de inferência. Os experimentos foram feitos com o *backbone* da resnet101, em comparação com os *backbones* resnet50 para Upernet e PSPNet e resnet18 para a Bisenet.

IV. Resultados

Nessa seção são apresentados os resultados dos experimentos do trabalho.

1) Experimento 1 - Modificação na função de custo: No primeiro experimento realizado, utilizando todo o conjunto de treinamento, todas as redes tiveram um resultado médio de F1 equivalente ou maior quando utilizada a função de custo *focal loss*. Os resultados da média das três classes (Solo, Planta, Erva Daninha) são mostrados na Figura 8. As redes Enet e PSPNet praticamente mantiveram o mesmo resultado, enquanto a Unet teve uma melhora de 2% e a Upernet de aproximadamente 3%. Agora comparando apenas a classe com menor número de instâncias *erva daninha*, apenas a rede E-net não obteve um ganho significativo, enquanto a PSPNet obteve cerca de 8% de ganho, a Unet 5% de ganho e a Upernet 7% de ganho.

Os resultados de todas as classes podem ser vistos no Apêndice A.

2) Experimento 2 - Variação no número de imagens para treinamento: Neste experimento, testamos as redes E-Net, U-Net, PSPNet e Upernet, com as funções de perda *cross entropy* e *focal loss* e com 6182 instâncias de treinamento e com um conjunto de treinamento de 3000 instâncias. Os resultados obtidos na média das três classes (Solo, Planta e Erva Daninha) são apresentados na Figura 9.

Observa-se que em todos os casos, quando utilizado um conjunto de dados reduzido para treinamento, houve uma grande redução no valor do F1 médio, seja utilizando a função de custo *cross entropy* ou *focal loss*. As reduções nos valores de F1 médio das redes treinadas com *Cross Entropy* e *Focal Loss* são respectivamente: para rede Enet uma redução de 22% e 25%, para a rede PSPNet uma redução de 12% e 22%, para a rede Unet uma redução de 25% e 26%, e para a rede Upernet redução de 13% e 15%. Em todos os casos, a classe mais prejudicada foi a classe com o menor número de instâncias *erva daninha*.

Os resultados de todas as classes podem ser vistos no Apêndice B.

3) Experimento 3 - Utilização de pesos pré-treinados no backbone: No terceiro experimento, testamos a utilização de pesos pré-treinados no dataset ImageNet nas redes Upernet e PSPNet e o backbone da resnet50. A média dos resultados obtidos das classes é apresentado na Figura 10.

Observou-se um ganho expressivo no valor médio de F1 para ambas as arquiteturas. No caso da PSPNet, o ganho médio de F1 foi de 10%, enquanto na Upernet o ganho foi de 14%. Em ambos os casos todas as classes tiveram ganhos significativos, sendo a classe *erva daninha* com o maior ganho absoluto, aumentando cerca de 30% no caso da Upernet e 16% na PSPNet.

Os resultados de todas as classes podem ser vistos no Apêndice C.

4) Experimento 4 - Variações na arquitetura do backbone: No último experimento, testamos variações de *backbones* nas redes Bisenet, PSPNet e Upernet. A Figura 11 mostra a média da acurácia, precisão, revocação e F1 das três classes.

Indo ao contrário da expectativa, o último experimento mostrou que aumentando a complexidade do *backbone* não necessariamente resulta em uma ganho nas métricas de avaliação. Na realidade, os testes mostraram que as arquiteturas com *backbones* mais profundas ou não modificaram os resultados ou reduziram os valores médios de F1. A rede PSPNet obteve uma redução de cerca de 8%, a rede

[4]https://github.com/ooooverflow/BiSeNet

[5]https://github.com/yassouali/pytorch_segmentation

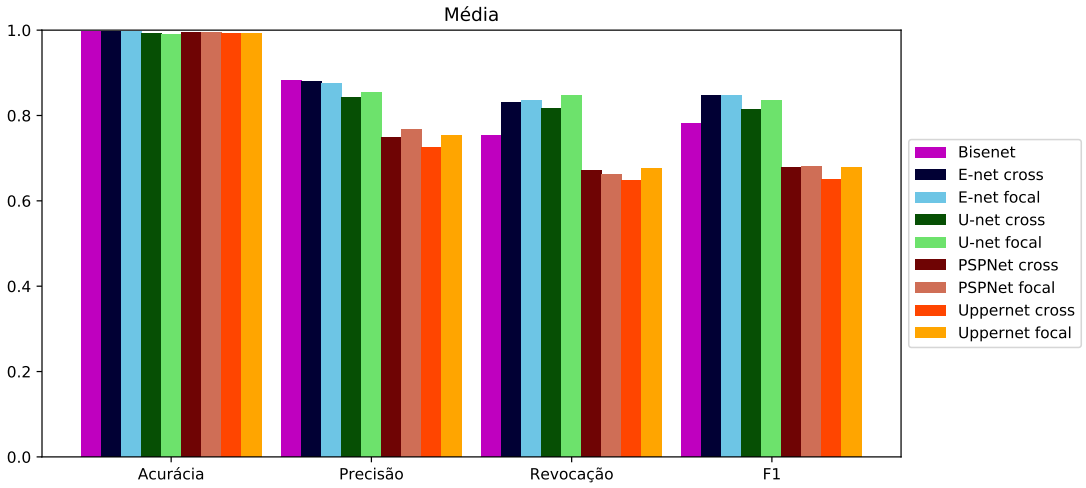

Figure 8. Média da acurácia, precisão, revocação e F1 das classes Solo, Planta e Erva Daninha no experimento 1.

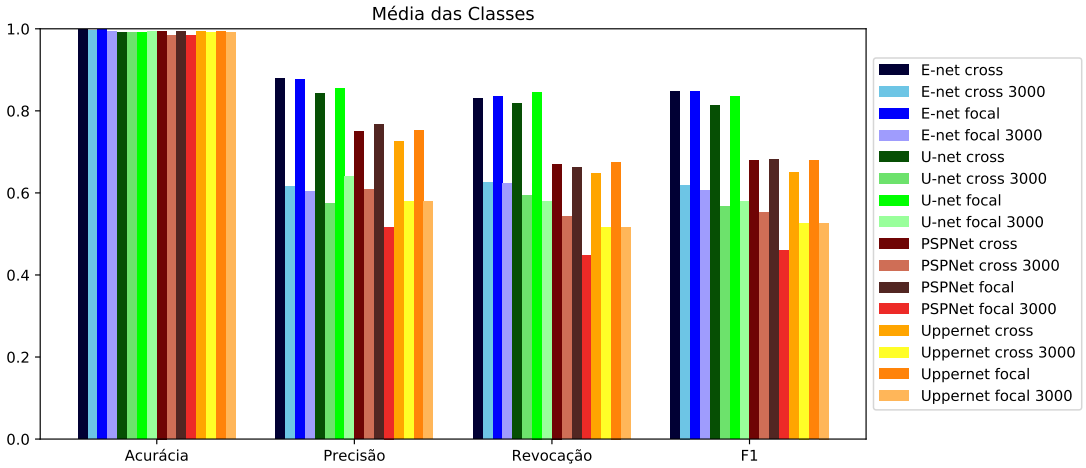

Figure 9. Média da acurácia, precisão, revocação e F1 das classes Solo, Planta e Erva Daninha no experimento 2.

Bisenet obteve uma redução de 16%, enquanto a Upernet não apresentou mudanças significativas nas métricas.

Os resultados de todas as classes podem ser vistos no Apêndice D.

5) FPS: Comparamos a quantidade de FPS entre as redes de modo geral. Como os experimentos anteriores não dependem da arquitetura, comparamos as redes com função de perda *cross entropy*. A Figura 12 apresenta os resultados obtidos.

Pelos resultados, percebemos que a Bisenet possui maior quantidade de quadros por segundo, com valores próximos a 80, seguida pela E-net, com valores próximos a 50. As redes U-net, PSPNet e Upernet possuem valores bem próximos, com cerca de 20 quadros por segundo.

V. Conclusão

A Utilização de agroquímicos modo igual em toda a área cultivada possui riscos tanto para a plantação quanto para a saúde dos consumidores. A utilização de pulverizadores equipados com bicos direcionados uma das soluções para este problema, desde que seja possível reconhecer as ervas daninhas que atrapalham a plantação.

Neste trabalho comparamos redes neurais convolucionais para a segmentação semântica em plantações. Realizamos experimentos com duas funções de perda, com menos instâncias no treinamento, com redes pré treinadas e com arquiteturas diferente no *backbone* das redes.

Pelos resultados obtidos, entendemos que de modo geral a função de perda não foi tão influente nos resultados obtidos e que a limitação de instâncias no treinamento piora os resultados. Observamos também que redes pré treinadas auxiliam na melhora dos resultados e que a utilização de *backbones* maiores e mais complexos não necessariamente melhoram os resultados.

De modo geral, se compararmos tanto o F1 médio quanto a quantidade de FPS, observamos que a melhor rede para ser aplicada em problemas reais é a Enet, já que possui um dos maiores F1 nas classes planta e erva daninha e possui cerca de 50 FPS, o segundo maior neste quesito.

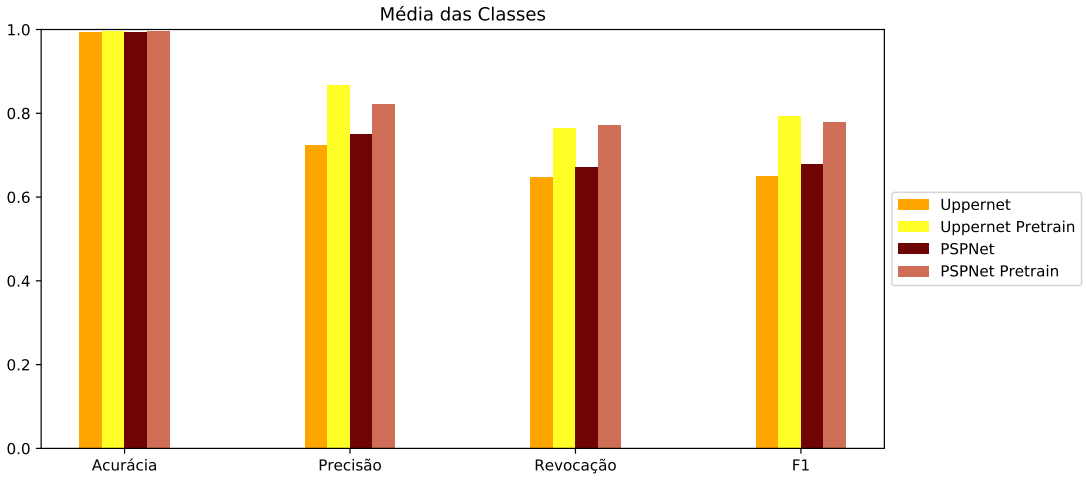

Figure 10. Média da acurácia, precisão, revocação e F1 das classes Solo, Planta e Erva Daninha no experimento 3.

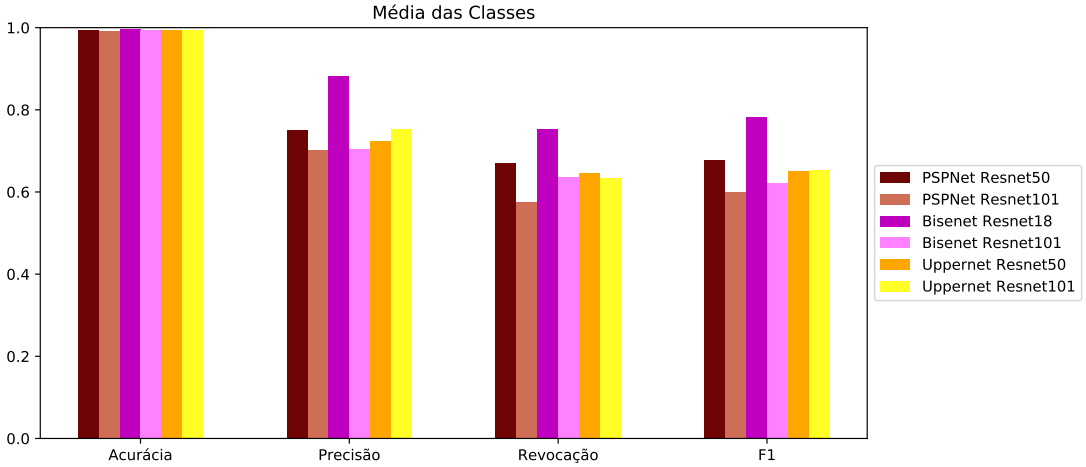

Figure 11. Média da acurácia, precisão, revocação e F1 das classes Solo, Planta e Erva Daninha no experimento 4.

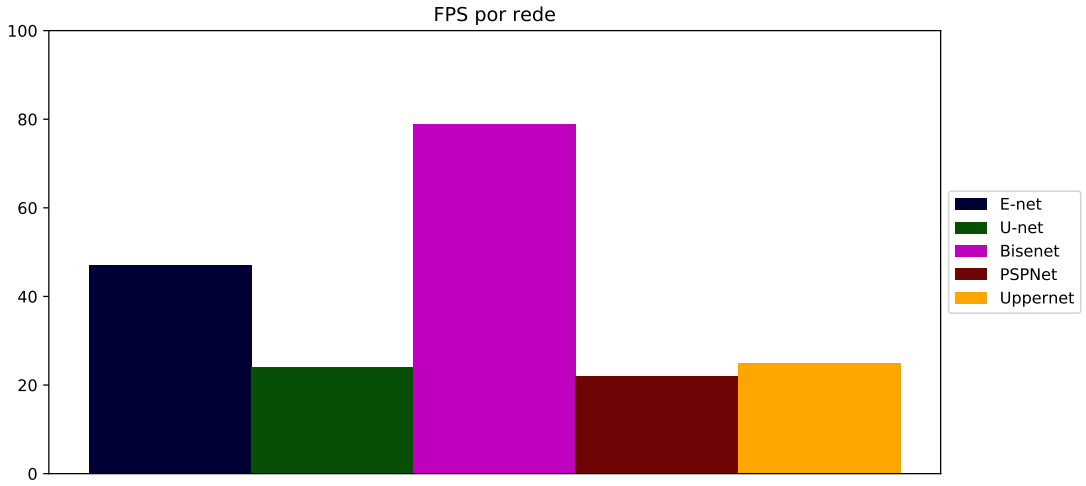

Figure 12. Comparação de FPS entre as diferentes arquiteturas.

References

[1] Nived Chebrolu et al. “Agricultural robot dataset for plant classification, localization and mapping on sugar beet fields”. In: *The International Journal of Robotics Research* (2017). DOI: 10.1177/0278364917720510.

[2] Nived Chebrolu et al. “Agricultural robot dataset for plant classification, localization and mapping on sugar beet fields”. In: *The International Journal of Robotics Research* 36.10 (2017), pp. 1045–1052. DOI: 10.1177/0278364917720510. eprint: https://doi.org/10.1177/0278364917720510. URL: https://doi.org/10.1177/0278364917720510.

[3] Kaiming He et al. “Deep residual learning for image recognition”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[4] Mina Khoshdeli and Bahram Parvin. “Deep learning models delineates multiple nuclear phenotypes in h&e stained histology sections”. In: *arXiv preprint arXiv:1802.04427* (2018).

[5] Philipp Lottes et al. “Fully convolutional networks with sequential information for robust crop and weed detection in precision farming”. In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 2870–2877.

[6] Philipp Lottes et al. “Joint stem detection and crop-weed classification for plant-specific treatment in precision farming”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 8233–8238.

[7] Andres Milioto, Philipp Lottes, and Cyrill Stachniss. “Real-time semantic segmentation of crop and weed for precision agriculture robots leveraging background knowledge in CNNs”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 2229–2235.

[8] Adam Paszke et al. “Enet: A deep neural network architecture for real-time semantic segmentation”. In: *arXiv preprint arXiv:1606.02147* (2016).

[9] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[10] Tete Xiao et al. “Unified perceptual parsing for scene understanding”. In: *European Conference on Computer Vision (ECCV)*. 2018, pp. 418–434.

[11] Changqian Yu et al. “Bisenet: Bilateral segmentation network for real-time semantic segmentation”. In: *European Conference on Computer Vision (ECCV)*. 2018, pp. 325–341.

[12] Hengshuang Zhao et al. “Pyramid scene parsing network”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2881–2890.

Appendix A
Resultados do Experimento 1

Nesta seção, apresentaremos os gráficos de acurácia, precisão, revocação e F1 das classes solo, planta e erva daninha, mostrados nas Figuras 13, 14 e 15.

Appendix B
Resultados do Experimento 2

Nesta seção, apresentaremos os gráficos de acurácia, precisão, revocação e F1 das classes solo, planta e erva daninha, mostrados nas Figuras 16, 17 e 18.

Appendix C
Resultados do Experimento 3

Nesta seção, apresentaremos os gráficos de acurácia, precisão, revocação e F1 das classes solo, planta e erva daninha, mostrados nas Figuras 19, 20 e 21.

Appendix D
Resultados do Experimento 4

Nesta seção, apresentaremos os gráficos de acurácia, precisão, revocação e F1 das classes solo, planta e erva daninha, mostrados nas Figuras 19, 20 e 21.

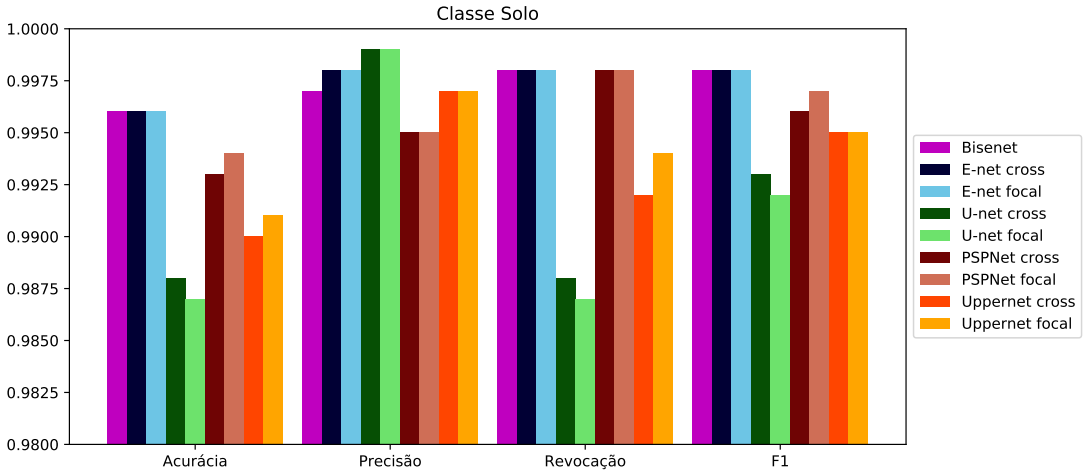

Figure 13. Acurácia, precisão, revocação e F1 da classe Solo no experimento 1.

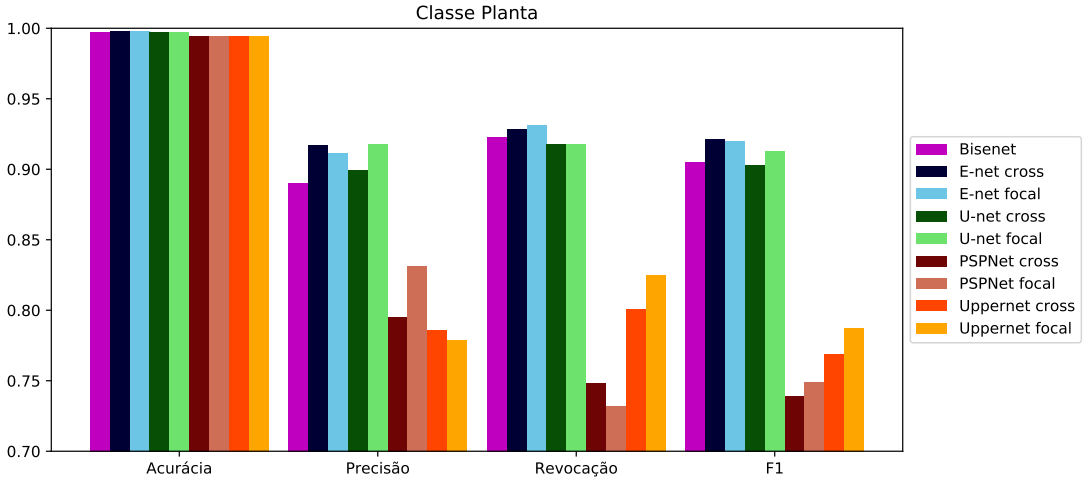

Figure 14. Acurácia, precisão, revocação e F1 da classe Planta no experimento 1.

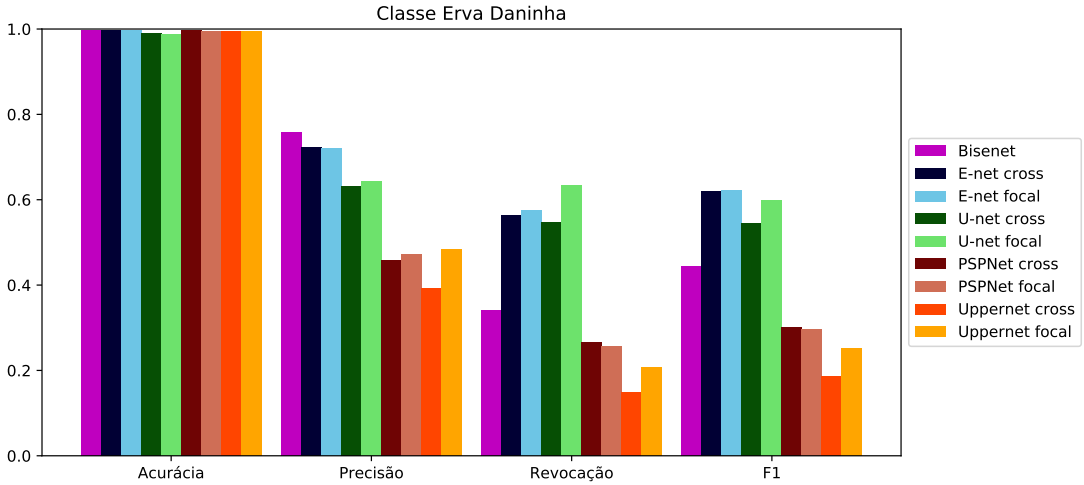

Figure 15. Acurácia, precisão, revocação e F1 da classe Erva Daninha no experimento 1.

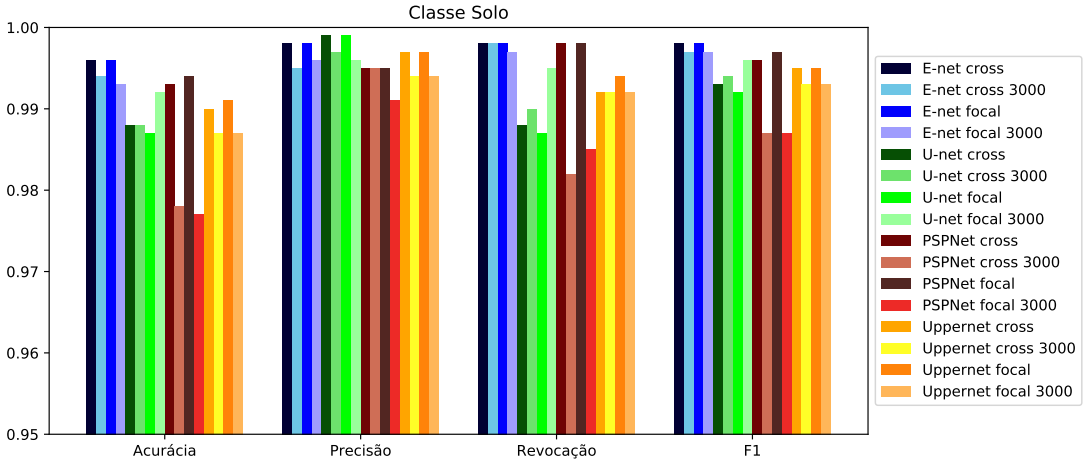

Figure 16. Acurácia, precisão, revocação e F1 da classe Solo no experimento 2.

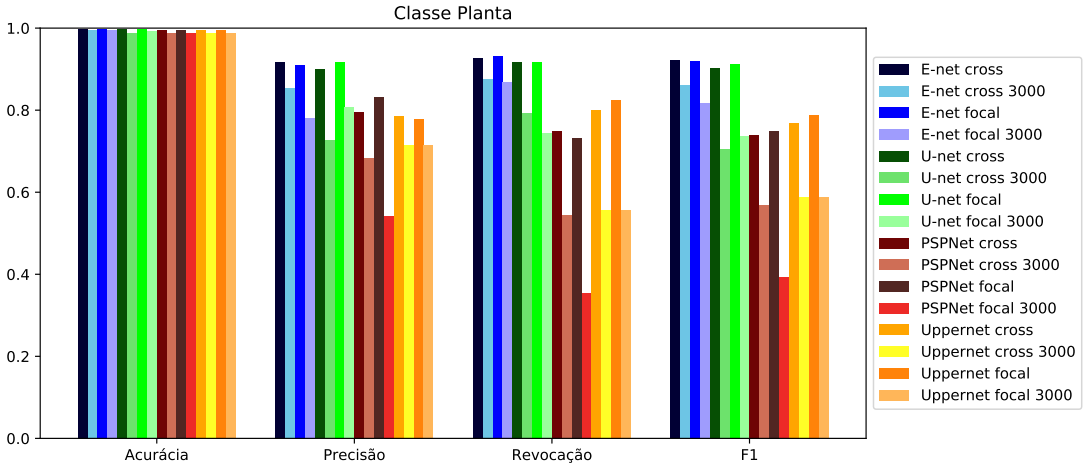

Figure 17. Acurácia, precisão, revocação e F1 da classe Planta no experimento 2.

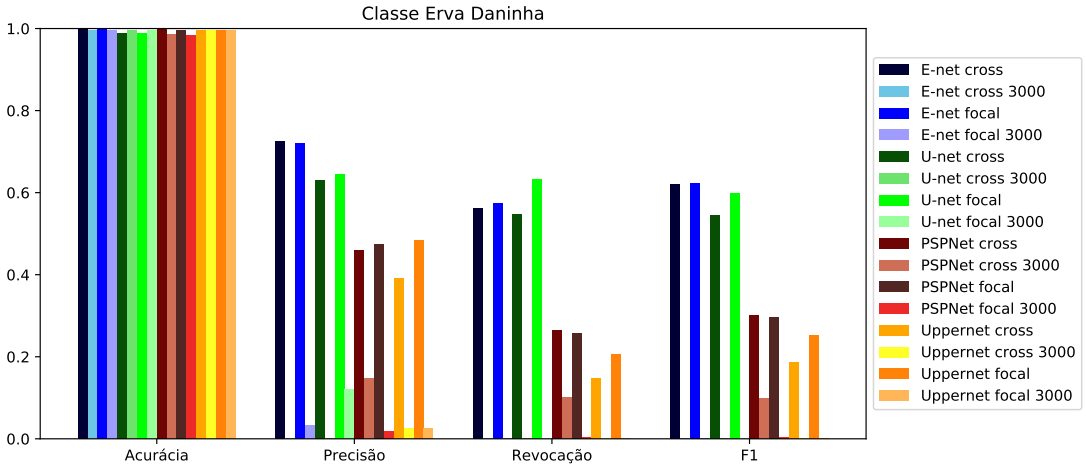

Figure 18. Acurácia, precisão, revocação e F1 da classe Erva Daninha no experimento 2.

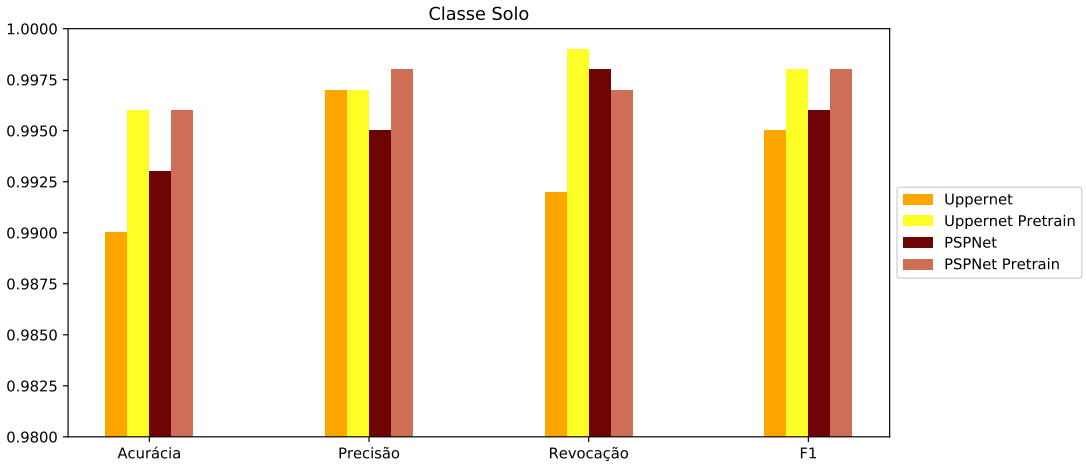

Figure 19. Acurácia, precisão, revocação e F1 da classe Solo no experimento 3.

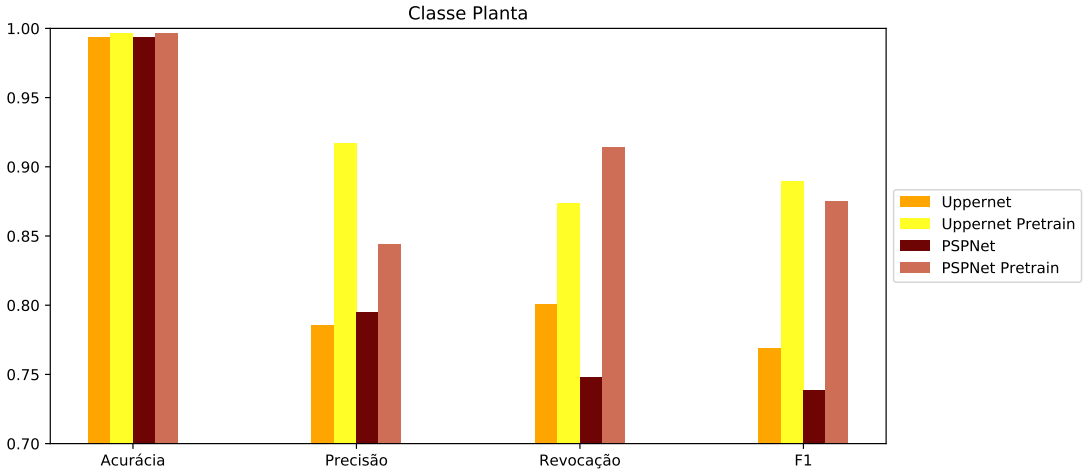

Figure 20. Acurácia, precisão, revocação e F1 da classe Planta no experimento 3.

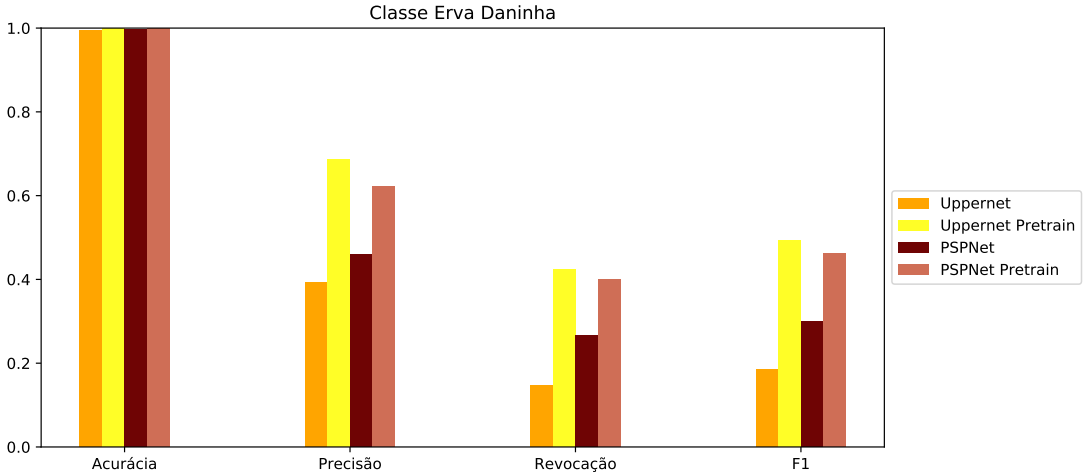

Figure 21. Acurácia, precisão, revocação e F1 da classe Erva Daninha no experimento 3.

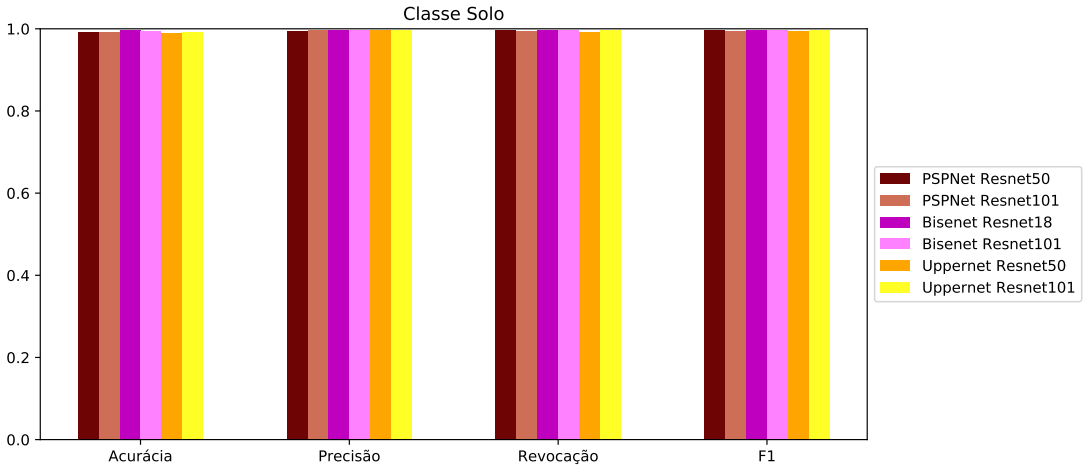

Figure 22. Acurácia, precisão, revocação e F1 da classe Solo no experimento 4.

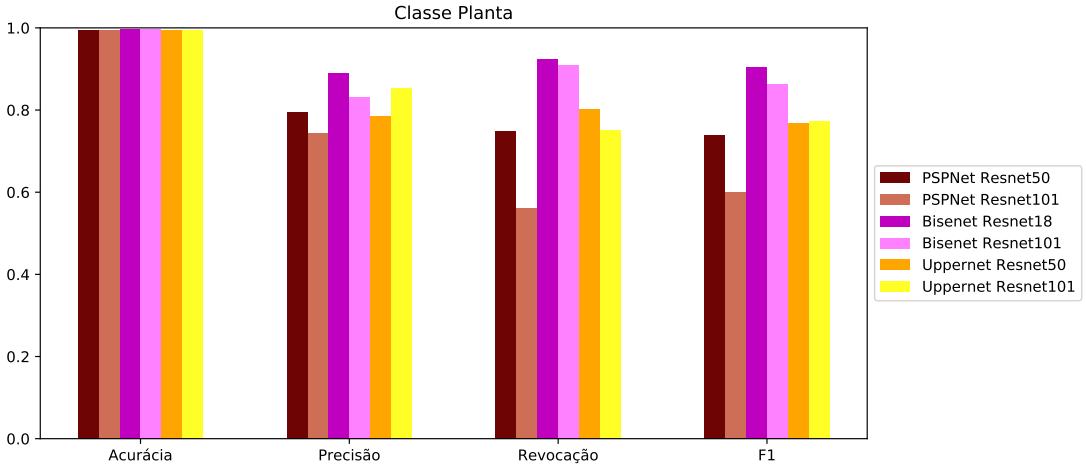

Figure 23. Acurácia, precisão, revocação e F1 da classe Planta no experimento 4.

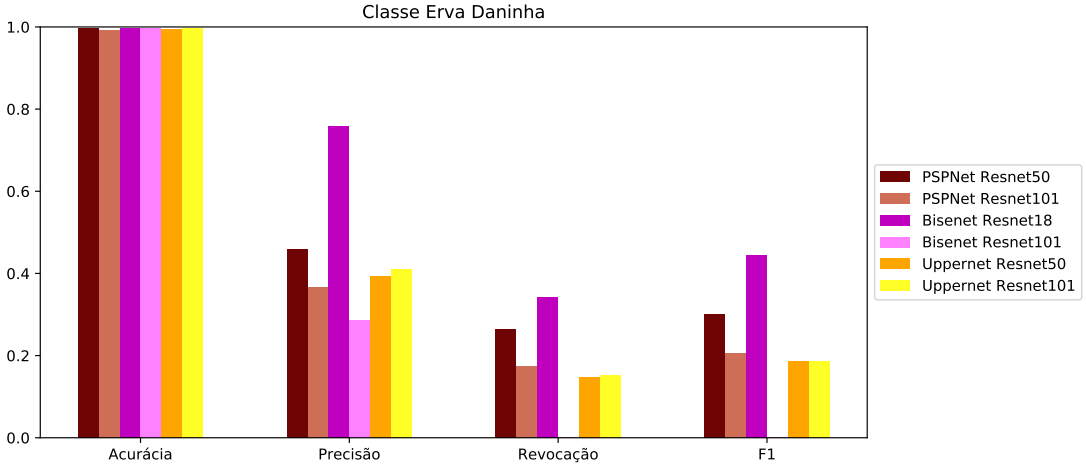

Figure 24. Acurácia, precisão, revocação e F1 da classe Erva Daninha no experimento 4.