FEDERAL UNIVERSITY OF SANTA CATARINA
AUTOMATION AND SYSTEMS ENGINEERING GRADUATE PROGRAM

Rafael de Souza Toledo

A Performance Increment Strategy for Semantic Segmentation of Low-Resolution Images from Damaged Roads

Florianópolis
2022

Rafael de Souza Toledo

A Performance Increment Strategy for Semantic Segmentation of Low-Resolution Images from Damaged Roads

Master thesis submitted to the Automation and Systems Engineering Graduate Program of the Federal University of Santa Catarina in partial fulfillment of the requirements for the degree of Master of Science.
Advisor: Prof. Eric Aislan Antonelo, Ph.D.

Florianópolis
2022

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Toledo, Rafael de Souza
   A Performance Increment Strategy for Semantic Segmentation of Low-Resolution Images from Damaged Roads / Rafael de Souza Toledo ; orientador, Eric Aislan Antonelo, 2022.
   118 p.

   Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Engenharia de Automação e Sistemas, Florianópolis, 2022.

   Inclui referências.

   1. Engenharia de Automação e Sistemas. 2. Aprendizado Profundo. 3. Visão Computacional. 4. Segmentação Semântica. 5. Estradas danificadas. I. Antonelo, Eric Aislan. II. Universidade Federal de Santa Catarina. Programa de Pós Graduação em Engenharia de Automação e Sistemas. III. Título.

Rafael de Souza Toledo

A Performance Increment Strategy for Semantic Segmentation of Low-Resolution Images from Damaged Roads

This master thesis is recommended in partial fulfillment of the requirements for the degree of Master of Science which has been approved in its present form by the Graduate Program in Automation and System Engineering.

Prof. Aldo von Wangenheim, Dr.
INE/UFSC

Prof. Marcelo Ricardo Stemmer, Dr.
DAS/UFSC

Prof. Maurício Edgar Stivanello, Dr.
DAMM/IFSC

This is the **original and final** work properly evaluated to acquire the tittle of Master's degree in the Automation and Systems Engineering Graduate Program.

ICP-Edu
Documento assinado digitalmente
Julio Elias Normey Rico
Data: 21/10/2022 17:34:57-0300
CPF: ***.840.859-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Júlio Elias Normey Rico, Dr.
Graduate Program Coordinator

ICP-Edu
Documento assinado digitalmente
Eric Aislan Antonelo
Data: 21/10/2022 15:24:03-0300
CPF: ***.163.409-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Eric Aislan Antonelo, Ph.D.
Advisor

Florianópolis, 2022.

I dedicate this work to my beloved ones,
especially my family for their endless
support and friendship.

ACKNOWLEDGEMENTS

I want to thank the PPGEAS program for fostering this collaborative and scientific environment and helping me pursue a research career.

“Just as electricity transformed
almost everything 100 years ago,
today I actually have a hard time
thinking of an industry that
I don’t think AI will transform
in the next several years.”
- Andrew Ng

RESUMO

Segmentação semântica é essencial para o entendimento de cenário de estradas e, consequentemente, para a realização de navegação autônoma. Entretanto, novos desafios surgem quando essas tarefas são postas em países emergentes dado a falta de uma infra-estrutura de qualidade ou a restrição a recursos computacionais. Recentemente, a Confederação Nacional de Transportes (CNT) reportou que 85% das estradas brasileiras apresentam algum dano como rachaduras, buracos, e remendos; normalmente, esses danos não são levados em conta pelos modelos de aprendizado profundo do estado da arte, os quais são treinados para atender a infra-estrutura de países desenvolvidos em conjunto de dados de alta-resolução como Cityscapes (2048x1024) e CamVid (920x720). Em 2019, o dataset *Road Transverse Knowledge* (RTK) foi projetado especialmente para atender a realidade de países emergentes; ele consiste de 701 imagens densamente anotadas de baixa-resolução (352x288) e 12 classes com diferentes categorias de estradas e danos como buracos, poças d'águas, e rachaduras. Baseado no dataset RTK, esse trabalho indica os principais desafios para estradas de países emergentes: 1) detecção de objetos pequenos dado a baixa-resolução da imagem, 2) objetos de múltiplas escalas dado a irregularidade da forma dos objetos, e 3) classes altamente desbalanceadas dado que as classes de danos são de tamanho pequeno. Em seguida, esse trabalho propõe a estratégia de incremento de performance para melhorar os resultados em conjuntos de dados de países emergentes; a estratégia consiste em uma série de 15 experimentos a fim de escolher a melhor opção para cada configuração de treinamento, como ampliação de dados, função perda e otimizador. Além desses, a estratégia sugere modificações na arquitetura como a remoção da camada *max-pooling* da ResNet e taxas de dilatação híbrida e digressiva. Ao final do trabalho, a estratégia alavancou o *benchmark* do RTK de 0.547 para 0.798 mIoU no conjunto de validação, e atingiu 0.688 mIoU no conjunto de teste do TAS500; os melhores resultados publicados até o momento.

Palavras-chave: navegação autônoma; segmentação semântica; estradas em países emergentes; estradas danificadas; imagens de baixa-resolução.

RESUMO EXPANDIDO

Introdução

Visão computacional é um campo de estudo com soluções que permeiam o nosso dia-a-dia cada vez mais; desde aplicações em aparelhos celulares como reconhecimento facial para desbloqueio do próprio aparelho ou para validação de transações bancárias até em setores mais primários da economia como inspeção de placas em indústrias ou monitoramento de animais e colheitas no setor agropecuário.

Atualmente, muitas dessas soluções são chamadas basicamente de Inteligência Artificial (IA). Isso é devido suas soluções serem dominantemente baseadas em aprendizado profundo, um subcampo de IA. Aprendizado profundo é um paradigma que ganhou muita força a partir de 2010 dado o grande poder computacional e a farta disponibilidade de dados digitais dos nossos tempos. Hoje em dia, a grande difusão de soluções de IA na sociedade é vista analogamente ao que foi a eletricidade 100 anos atrás; isso é, essa tecnologia permeará todos os meios de produção, economias, e espaços da sociedade; portanto, uma indústria que não poderia ficar de fora dessa onda, obviamente, é a indústria automotiva.

A aplicação de IA aos veículos busca em um primeiro momento desonerar o motorista de estar totalmente empenhado nas faculdades motoras e mentais que exigem o ato de dirigir, o que traria mais conforto e segurança a ele. Posteriormente, como objetivo último, a indústria automotiva usaria IA para a completa substituição do motorista humano - isto é, navegação autônoma ao veículo -, reduzindo custos ao transporte de cargas e pessoas. A navegação autônoma requer que o veículo mapeie o que está acontecendo ao seu redor; o que é usualmente feito por vários sensores emissores de sinais como LIDAR. Entretanto, hoje o senso comum é que apenas a visão passiva, isto é, sensores que apenas recebem sinais do exterior (no nosso caso, câmeras) atendem melhor essa função.

Câmeras projetam em suas saídas uma imagem 2D, o que torna segmentação semântica chave para o entendimento da cena ao redor do veículo. Segmentação semântica é a tarefa que assinala uma classe a cada pixel da imagem. As classes usuais são a própria definição da coisa pela perspectiva de direção, como pedestres, carros, calçadas, faixas, e etc.

A possibilidade de navegação autônoma é muito excitante, porém, essas tecnologias são normalmente desenvolvidas em países desenvolvidos e primeiramente atenderão a sua realidade, que pode diferir bastante da realidade de países emergentes. Por exemplo, um relatório de 2021 da Confederação Nacional de Transportes (CNT) reportou que 85% das estradas brasileiras apresentam algum dano como rachaduras, remendos ou buracos. Além disso, outra restrição bastante comum aos países emergentes é a restrição ao poder computacional, o que impossibilita o uso do algoritmo mais potente; por exemplo, capacidade de processar a imagem em alta resolução.

Algum dos datasets mais comum para o desenvolvimento de algoritmos do estado da arte para segmentação semântica são datasets de alta resolução como Cityscapes (2048x1024) e CamVid (920x720) que representam estradas urbanas europeias em excelentes estados de manutenção e não possuem nenhum classe assinalada a danos nas estradas. A fim de representar a realidade brasileira pra esse problema, em 2019, o dataset Road Transverse Knowledge (RTK) foi lançado, contando com 701 imagens densamente rotuladas e 12 classes que representam variações da superfície da pista, sinalizações e danos.

Pelo fato do RTK apresentar um cenário particular, logo ele também apresenta desafios particulares, como 1) a presença de objetos minúsculos em números de pixels, por exemplo, faixas que tem bordas de 1 pixel de comprimento; 2) classes em multi-escalas de formas geométricas, por exemplo, olhos de gatos retangulares e pequenos no meio de faixas finas e compridas; ou classes de formas totalmente irregulares como poças d'águas e rachaduras; e 3) alto desbalanceamento de classes dado que danos são classes de tamanho pequeno, sendo 98.5% representando plano-de-fundo e superfícies da pista. Ademais, esse trabalho também usou o modelo de aprendizado profundo desenvolvido originalmente pelos autores do dataset RTK como ponto de partida; o qual não adotou práticas recorrentes encontradas no estado-da-arte que foram trazidos por esse trabalho a fim de melhorar a performance de um modelo no dataset RTK.
A proposta desse trabalho é a **estratégia de incremento de performance** que consiste de uma série de 15 experimentos. Cada experimento busca escolher a melhor opção para cada configuração de treinamento do algoritmo de aprendizado profundo; um experimento é composto de várias hipóteses, que são as possíveis opções para essa configuração; a melhor hipótese é escolhida a partir de um estudo de ablação. Os experimentos são aditivos, isso é, o experimento seguinte roda a partir da melhor hipótese consagrada no experimento anterior.
Por mais, os 15 experimentos foram organizados em 4 categorias: base, predição, técnica, e arquitetura. *Base* conta com apenas um experimento que testa o número de iterações. *Predição* também conta com um único experimento que usa a média de um elenco de predições de várias versões da mesma imagem. *Técnica* conta com os experimentos das configurações marginais aos treinamento como otimizador, ampliação de dados, ou número de estágios de treinamento. Por último, *arquitetura* conta com experimentos de alterações diretamente na arquitetura como tipos de textitencoder, tipos de arquitetura, hiper-parâmetros da arquitetura como *output stide*, e modificações de arquitetura como a remoção da camada de textitmax-pooling.

Objetivos
Os objetivos deste trabalho estão listado abaixo:

- Analisar quantitativamente e qualitativamente as características do dataset RTK.
- Desenvolver uma estratégia para melhorar a performance de segmentação semântica em estradas de países emergentes; isso é, a estratégia de incremento de performance.
- Aplicar a estratégia de incremento de performance no dataset RTK.
- Analisar quantitativamente e qualitativamente os resultados da estratégia.
- Aplicar a estratégia de incremento de performance no dataset TAS500.
- Resumir os principais achados após a execução da estratégia em dois datasets.

Metodologia
Todos os experimentos rodaram na plataforma do Google Colab usando a biblioteca PyTorch 1.12. Primeiramente, rodou-se todos os experimentos no dataset RTK. Depois, a fim de validar a estratégia, usou-se um segundo dataset, o TAS500.
A métrica usada para mensurar a performance dos experimentos foi o *mean intersection over union* (mIoU); e dado a volatilidade da métrica após a convergência do

treinamento no conjunto de validação (em torno de 1%), optou-se por usar a média das 10 últimas medições para estabelecer o mIoU final de cada hipótese. Cada hipótese rodou 200k iterações, salvo poucas exceções que foram explicitamente ditas.

Resultados e Discussão

A estratégia trouxe ganhos significativos e atingiu resultados top-1 em ambos datasets. Entretanto, o conjunto de hipóteses que melhor funcionou para um foi totalmente diferente do que melhor funcionou para o outro. Por exemplo, enquanto *resizing* como ampliação de dado, *output stride* 16 para o encoder do DeepLabV3+, função perda CE + soft-Dice, e otimizador SGD melhor funcionaram para o TAS500; *cutmix*, *output stride* 4, função perda CE, e otimizador Adam melhor atenderam o RTK.
A lista abaixo sumariza os principais pontos encontrados nos experimentos.

- A segmentação semântica de objetos pequenos em imagens de baixa-resolução é um problema se as dimensões do mapa de característica for reduzida antes da extração de características de mais alto-nível ou se essa informação não for re-adicionada a rede posteriormente, por exemplo, com uma skip-connection. A rede neural tende a preditar um alto número de falso-positivos de objetos pequenos; o algoritmo tende a ter uma alta sensibilidade para essas classes mas uma baixa precisão.
- As classes mais difíceis para segmentação semântica do RTK são: rachaduras, poças d'águas e olhos de gatos.
- Fora confirmado o viés que existe da métrica mIoU em objetos grandes.
- CE otimiza a métrica mIoU melhor do que a própria função-substituta de mIoU ou de dice.
- A performance de cada hipótese do experimento depende da condição inicial ao qual ela é testada.
- Quando uma hipótese foi melhor que a outra, ela funcionou melhor para todos os grupos de classes; por exemplo: objetos pequenos, objetos grandes, objetos sub-representados.
- A alta discrepância entre a melhor configuração entre os dois datasets endossa a necessidade de uma solução ajustada para cada contexto.
- Configurações convencionais são normalmente mais robustas que configurações mais sofisticadas, e elas devem ser nossas primeiras tentativas.

Considerações Finais

A estratégia foi efetiva, e acreditamos que o trabalho avança na direção de tornar possível a navegação autônoma em países emergentes. Também pensamos que esse conjunto de experimentos da estratégia pode ser um guia para outros problemas de segmentação semântica, já que muitas dessas configurações são comuns a diversos problemas.

ABSTRACT

Semantic segmentation is vital for understanding a road scene and, consequently, achieving autonomous driving. However, new challenges arise when attempting these tasks in emerging countries, given the lack of high-quality infrastructure or limited computational resources. Recently, the Brazilian National Transport Confederation (CNT) reported that 85% of the Brazilian roads present some damage like cracks, holes, and patches; these damages are usually not regarded by the state-of-the-art deep learning models of road semantic segmentation, which are trained to meet the developed countries infra-structure in high-resolution datasets like Cityscapes (2048x1024) and CamVid (920x720). In 2019, the Road Transverse Knowledge (RTK) was specially designed to meet the emerging country reality; it consists of 701 fine-annotated images of low-resolution (352x288) and 12 classes with different road surfaces and damages like potholes, water puddles, and cracks. Based on the RTK dataset, this work points out the main challenges for emerging country roads: 1) small objects given low-resolution images, 2) multiscale objects given irregular-shaped objects, and 3) highly imbalanced classes given road-damages small size. Finally, this work proposes the performance increment strategy to enhance results in emerging country datasets; the strategy consists of a series of 15 experiments to choose the best option for each training setup like data augmentation, loss function, and optimizer. Furthermore, the strategy suggests architecture modifications such as the max-pooling layer removal from ResNet and hybrid and digressive dilation rates. In the end, the strategy raised the RTK benchmark from 0.547 to 0.798 mIoU on the validation set; and reached 0.688 mIoU in the TAS500 test set, the best results published so far.

Keywords: autonomous driving; semantic segmentation; emerging countries roads; damaged roads; low-resolution images.

LIST OF FIGURES

Figure 1 – The performance increment strategy experiments. 35
Figure 2 – A biological and an artificial neuron. From (KARPATHY et al., 2016). 40
Figure 3 – From (HAYKIN, 2009). 40
Figure 4 – A 3-layer feedforward neural network. From (KARPATHY et al., 2016). 41
Figure 5 – A typical CNN architecture with two feature stages. From (LECUN; KAVUKCUOGLU; FARABET, 2010) 42
Figure 6 – From (FUKUSHIMA; MIYAKE, 1982). 43
Figure 7 – (left) Input image; (center) the channel's weight parameters (kernel); its size determines the receptive field of a single neuron, which is 5x5 in this case; (right) the computed feature map. From (LECUN et al., 1989b) 44
Figure 8 – A convolutional layer arranges its neurons in three dimensions (width, height, and depth). The convolutional layer transforms the 3D input volume into a 3D output volume of neurons' activations. In this example, the red input layer holds the image, and its volume is transformed along the network. From (KARPATHY et al., 2016). 44
Figure 9 – The receptive field of each convolutional layer neuron is based on a 3×3 kernel. The green and yellow areas mark the receptive field of a single pixel in layers 2 and 3, respectively. From (LIN; SHI; ZOU, 2017). 46
Figure 10 – A typical convolutional neural network layer. From (LECUN; BENGIO; HINTON, G., 2015). 46
Figure 11 – Example of a convolution operation of kernel 3x3 sliding across every 4x4 input position (stride 1) and no padding, resulting in a 2x2 output. From (DUMOULIN; VISIN, 2016). 47
Figure 12 – Example of a convolution operation of kernel 3x3 sliding across every 5x5 input position (stride 1) and padding 1, resulting in an output of the same size as the input. From (DUMOULIN; VISIN, 2016). 47
Figure 13 – The ReLU activation function. From (LECUN; BENGIO; HINTON, G., 2015) 47
Figure 14 – Illustration of max pooling translation invariance. From (LECUN; BENGIO; HINTON, G., 2015). 48
Figure 15 – Strided convolution of 2. From (LECUN; BENGIO; HINTON, G., 2015). 49
Figure 16 – Dilated convolution of kernel 3x3 (gray square) over a 7x7 input (blue square) resulted in a 3x3 output (green square). A factor of *rate* – 1 controls the space between the kernel's weights. From (DUMOULIN; VISIN, 2016). 49

Figure 17 – Transposed convolution of kernel 3x3 (gray square) over a 3x3 input (blue squares) padded with a 1x1 border and stride 2x2 resulted in a 5x5 output (green square). The stride's backward operation creates holes (filled with zero) between the input neurons. From (DUMOULIN; VISIN, 2016). 50
Figure 18 – A building block of residual learning. From (HE, K. et al., 2016). . . . 50
Figure 19 – ResNet architectures. From (HE, K. et al., 2016). 51
Figure 20 – ResNet50 with the stem of kernel size 7x7 followed by four blocks. From (KHAN et al., 2020). 51
Figure 21 – FCN architecture. From (LONG; SHELHAMER; DARRELL, 2015). . 53
Figure 22 – U-Net architecture. From (RONNEBERGER; FISCHER; BROX, 2015). 54
Figure 23 – DeepLabV3+ architecture. From (CHEN et al., 2018). 54
Figure 24 – Atrous Spatial Pyramid Pooling (ASPP). ASPP employs parallel dilated convolutions with different rates to classify the center pixel (orange). From (CHEN et al., 2017). 55
Figure 25 – Cutmix augmentation on the RTK dataset. 57
Figure 26 – Examples from the RTK dataset. From (RATEKE; VON WANGENHEIM, 2021). 61
Figure 27 – Examples of object masks with a short edge. 64
Figure 28 – Multiscale examples on the RTK dataset. The white circles highlight in a) lane markings with less than or equal to 2 pixels per edge at the end of the field of view and two nested classes (water puddles and cracks) of irregular shapes; in b) three different geometries and scales between classes: a wide and long road, an irregular and small storm drain at the border, and a road marking that changes its width along the road; and in c) a tiny cat's eye object between the road markings. 65
Figure 29 – Examples of RTK images regarding the number of classes. a) Single class label, only the unpaved road class; b) Label with four classes, the three road surfaces, asphalt (red), unpaved (yellow), paved (orange); and the lane markings (yellow-greenish); c) Label with six classes, it contains a transition of road surfaces (asphalt to paved), lane markings, cat's eyes (between the road markings), water-puddles (purple) and potholes (pink). 66
Figure 30 – Dense Upsampling Convolution (DUC). From (WANG, P. et al., 2018). 67
Figure 31 – Comparison between a bottleneck block and a Res2Net module. From (GAO et al., 2019). 68

Figure 32 – The usage of skip-connections in U-Net to combine multiscale information. a) U-Net, b) U-Net++, and c) U-Net3+. From (HUANG et al., 2020). 69
Figure 33 – Oscillation on the validation evaluation. 71
Figure 34 – The red and green lines show the training and the validation set evaluations, respectively. Overfitting badly affect the results on the validation set. 72
Figure 35 – Predictions with models trained at different output strides. 77
Figure 36 – ResNet backbone. "s" within the block is the block's stride; just below it prints the feature map spatial resolution. 78
Figure 37 – Without max-pooling layer at multiple output stride. 79
Figure 38 – Without max-pooling layer at multiple output stride. 80
Figure 39 – The default conv 7x7 layer with pre-trained weights are at the left path and the new proposed modules at the right path in a) rich stem and b) dummy rich stem. 81
Figure 40 – Series of hybrid dilation rates. 83
Figure 41 – Relation between IoU and median edge size of the class. 87
Figure 42 – Relation between IoU and normalized inverse frequency of the class. 88
Figure 43 – Inspection of small object predictions. The titles follow the loss hypothesis and the mIoU result. 89
Figure 44 – Notes from Figure 43. Point 1 underlines the exceeding predictions of underrepresented classes on *WCE* loss; point 2 shows a repeated error of predicting an unreal patch from the hypotheses; point 3 indicates the wrong pothole prediction of *CE*. 90
Figure 45 – Inspection of small objects predictions. 90
Figure 46 – Notes from Figure 45. Point 1 underlines the pedestrian lane reflected on the car hood; point 2 shows *WCE* over predicting underrepresented classes; point 3 shows the reflection predicted as a navigable path. 91
Figure 47 – Inspection of small objects predictions. 91
Figure 48 – Notes from Figure 47. Point 1 shows the larger blob prediction made by the *WCE* hypothesis; point 2 indicates that the *CE + Dice* hypothesis tends to vanish the lane in contrast to *CE + mIoU*, this pattern repeated in the *Dice* and *mIoU* standalone cases, see the Figure 47. 91
Figure 49 – False positive of small objects predicted by the hypotheses with max-pooling layer. 92
Figure 50 – False positive of small objects predicted by the hypotheses with max-pooling layer. 93

Figure 51 – Point 1 indicates the false positive small objects, and point 2 indicates the missed gap between the two markings predicted by the hypotheses with max-pooling layer. 93
Figure 52 – Inspection of multiscale objects predictions. 94
Figure 53 – Notes from Figure 52. Point 1 underlines missed lane in the label; point 2 indicates a missing surfaces transition by the *mIoU* hypothesis; point 3 underlines a nice transition between the surfaces. 94
Figure 54 – Evaluation of multiscale object predictions. 95
Figure 55 – Inspection of multiscale object predictions. 95
Figure 56 – Notes from Figure 55. Point 1 underlines the difficulty of predicting the small unpaved blob; point 2 shows that *WCE* does not present sharp boundaries as the *CE* hypothesis. 96
Figure 57 – TAS500 samples. From (METZGER; MORTIMER; WUENSCHE, 2021). 97
Figure 58 – Class distribution in the TAS500 dataset. The number of fine-grained pixels (y-axis) per class and their associated category (x-axis). From (METZGER; MORTIMER; WUENSCHE, 2021). 98
Figure 59 – Learning curve. 60k iterations are enough to reach convergence in the validation performance. Blue and gray lines are the evaluations of the training and validation sets, respectively. 99
Figure 60 – Comparison between the same hypothesis running with batch size 2 (longer lines) and 4 (shorter lines). The blue and gray lines represent the training and validation sets, respectively. Even running twice iterations for batch size 2, it can not get close to the batch size of 4 performance. 99
Figure 61 – Examples of TAS prediction. 103
Figure 62 – Examples of TAS prediction. 103
Figure 63 – Examples of TAS prediction. 103

LIST OF TABLES

Table 1 – Baseline number of iterations. 73
Table 2 – Single stage training. 73
Table 3 – Augmentation operations. 74
Table 4 – Encoder depth. 74
Table 5 – Resnet Variants. 75
Table 6 – Architectures types. 75
Table 7 – Output stride. 76
Table 8 – Max-pooling removal. 79
Table 9 – Rich Stem. 81
Table 10 – Upsampling by Transposed Convolution. 82
Table 11 – Hybrid Local Feature Extractor. 83
Table 12 – Cutmix addition. 84
Table 13 – Surrogate loss functions. 84
Table 14 – Surrogate loss functions on initial setup. 84
Table 15 – Optimizers. 85
Table 16 – Predictions Voting Ensemble. 86
Table 17 – Predictions Voting Ensemble w/ Best Checkpoint. 86
Table 18 – Intersection-over-union by class. 87
Table 19 – mIoU by classes groups. 88
Table 20 – mIoU by groups. 89
Table 21 – Output stride. 100
Table 22 – Cropping size. 100
Table 23 – Output stride. 100
Table 24 – Rich Stem & HLFE. 101
Table 25 – Cutmix. 101
Table 26 – Losses. 101
Table 27 – Optimizers. 102
Table 28 – Predictions Voting Ensemble w/ Best Checkpoint. 102

LIST OF ABBREVIATIONS AND ACRONYMS

ASPP	Atrous Spatial Pyramid Pooling
BCE	Binary Cross Entropy
CE	Cross-entropy
CNN	Convolutional Neural Network
DFN	Deep Feedforward Networks
FCN	Fully Convolutional network
HLFE	Hybrid Local Feature Extraction
IoU	Intersection-over-Union
LFE	Local Feature Extraction
mIoU	mean Intersection-over-Union
MLP	MultiLayer Perceptrons
MP	max-pooling
OS	Output Stride
RTK	Road Transverse Knowledge
SGD	Stochastic Gradient Descent
WCE	Weighted Cross Entropy

CONTENTS

1 INTRODUCTION . . . 32
1.0.1 RTK Issues . . . 33
1.0.2 Contributions . . . 34
1.1 PROPOSAL . . . 35
1.2 OBJECTIVES . . . 38
1.3 THESIS ORGANIZATION . . . 38
2 DEEP LEARNING BACKGROUND . . . 39
2.1 DEEP LEARNING . . . 39
2.1.1 Deep Feedforward Networks . . . 41
2.2 CONVOLUTIONAL NEURAL NETWORKS . . . 41
2.2.1 A Brief History . . . 42
2.2.2 Properties . . . 45
2.2.3 Typical Components . . . 46
2.2.3.1 Convolution Operation . . . 46
2.2.3.2 Activation Function . . . 47
2.2.3.3 Pooling . . . 48
2.2.3.4 Stride . . . 48
2.2.3.5 Dilated Convolution . . . 49
2.2.3.6 Transposed Convolution . . . 49
2.2.4 ResNet . . . 50
2.2.4.1 Dimensionalities . . . 51
2.3 SEMANTIC SEGMENTATION . . . 52
2.3.1 Neural Network Architectures . . . 52
2.3.1.1 Fully Convolutional Networks . . . 52
2.3.1.2 U-Net . . . 53
2.3.1.3 DeepLabV3+ . . . 54
2.3.2 Data Augmentation . . . 55
2.3.2.1 Geometry . . . 56
2.3.2.2 Resampling . . . 56
2.3.2.3 Color . . . 56
2.3.2.4 Synthetic . . . 57
2.3.3 Metrics And Losses Functions . . . 57
2.3.3.1 Cross Entropy . . . 57
2.3.3.2 Dice . . . 58
2.3.3.3 Intersection over Union . . . 59
3 RELATED WORKS . . . 60
3.1 ROAD TRAVERSING KNOWLEDGE DATASET . . . 60

3.1.1 Baseline 60
3.1.2 Criticism 61
3.1.2.1 Accuracy 61
3.1.2.2 Classes Weights 62
3.1.2.3 Number of Epochs 62
3.1.3 Challenges 63
3.1.3.1 Small objects 63
3.1.3.2 Multiscale objects 63
3.1.3.3 Imbalanced dataset 64
3.2 SMALL OBJECTS SEGMENTATION 64
3.2.1 Removing max-pooling layer 67
3.3 MULTISCALE SEGMENTATION 68
3.4 TRAINING AND INFERENCE PIPELINE 69
4 PERFORMANCE INCREMENT STRATEGY ON RTK 71
4.1 DEVELOPMENT ENVIRONMENT 71
4.2 METHODOLOGY 71
4.3 ABLATION EXPERIMENTS 72
4.3.1 Baseline: Iterations 72
4.3.2 Technique: Single Stage Training 73
4.3.3 Technique: Data Augmentation 73
4.3.4 Architecture: Encoder Depths 74
4.3.5 Architecture: Resnet Variants 75
4.3.6 Architecture: Networks 75
4.3.7 Architecture: Output Stride 76
4.3.8 Architecture: Max-Pooling Removal 77
4.3.9 Architecture: Rich Stem 80
4.3.10 Architecture: Transposed convolutions 81
4.3.11 Architecture: Hybrid Local Feature Extractor 82
4.3.12 Technique: Cutmix 83
4.3.13 Technique: Loss Functions 84
4.3.14 Technique: Optimizers 85
4.3.15 Prediction: Voting Ensemble 85
4.4 INSPECTING PERFORMANCE BY CLASSES AND GROUPS 86
4.5 INSPECTING SMALL OBJECTS PREDICTION 89
4.5.1 False Positive of Small Objects 92
4.6 INSPECTING MULTISCALE PREDICTION 93
5 PERFORMANCE INCREMENT STRATEGY ON TAS500 97
5.1 TAS500 DATASET 97
5.2 ABLATION EXPERIMENTS 98

5.2.1 Technique: Data Augmentation . . . 99
5.2.2 Technique: Cropping Size . . . 100
5.2.3 Architecture: Output Stride . . . 100
5.2.4 Architecture: Rich Stem & HLFE . . . 101
5.2.5 Technique: Cutmix . . . 101
5.2.6 Technique: Loss functions . . . 101
5.2.7 Technique: Optimizers . . . 101
5.2.8 Prediction: Voting Ensemble . . . 102
5.3 STRATEGY ANALYSIS . . . 102
5.4 TEST EVALUATION . . . 102
6 DISCUSSION AND CONCLUSIONS . . . 105
6.1 SUMMARY OF FINDINGS . . . 107
6.2 CONCLUSIONS . . . 107
6.3 FUTURE WORKS . . . 108
References . . . 109

1 INTRODUCTION

Vision is a powerful sense; it is no wonder that since the early days of the digital computer, attempts were made to give machines a sense of vision (HORN, B.; KLAUS; HORN, P., 1986). These attempts initiated the field of computer vision in the early 1970s which was viewed, at the time, as the visual perception component of an ambitious agenda to mimic human intelligence and to endow robots with intelligent behavior (SZELISKI, 2010). A robot vision system would analyze images and produces descriptions of what is imaged; these descriptions would capture useful aspects of the objects being imaged for carrying out some task (HORN, B.; KLAUS; HORN, P., 1986); as underlined in (SZELISKI, 2010), these descriptions could be shape, illumination, color distributions, or scene understanding like the semantic understanding or 3D structural description.

Nowadays, computer vision applications are in our daily life, for example, facial recognition for authentication systems, augmented reality for car parking assistance, and object detection for automatic inspection. And today, these applications are simply known as artificial intelligence, mainly because their solutions are based on deep learning, an artificial intelligence branch. Deep learning had an explosive growth of publications and commercial applications in the last decade, leveraged by the rising of large datasets and the increase of computational power (LECUN; BENGIO; HINTON, G., 2015); its applications are spread all over engineering solutions and economies; and, consequently, they are the most popular and widely approach for computer vision tasks.

Another long human ambition involving computer vision and artificial intelligence is autonomous driving, which is continuously pushed by the automotive industry, motivated by increasing people's safety, comfort, and transportation costs reduction. Autonomous driving is only possible if we fully understand the vehicle's surrounding scene, answering questions such as, What is happening? Why is it happening? What will happen next? What should I do? For example, the vision system would recognize nearby people and vehicles, anticipate their motions, infer traffic patterns, and detect road conditions (HOIEM et al., 2015). Surprisingly, the plain use of passive vision has accomplished these tasks, i.e., without sending any signal outwards, such that the system inputs are only based on 2D images; the plain use of 2D images turns **semantic segmentation** a central subject to solve these tasks.

Moreover, autonomous driving is cutting-edge technology and, as usual, these technologies start in developed countries' industries meeting their infrastructure reality which can vary a lot from emerging countries that lack high-quality infrastructure. A typical example of emerging country roads is found in Brazil. The Brazilian National Transport Confederation (CNT) 2021 survey (CNT, 2021) published that only 15% of the Brazilian roads are in perfect condition; the other 85% of the roads, which sum 92,711 km, present some problems. The main problems are fatigue failure, cracks, holes, patches, and wavy asphalt surfaces. These difficulties force particular attention to developing autonomous driving systems for emerging countries.

Another common scenario in emerging countries is the lack of resources to spend on high-tech devices, which implies vehicles with less computing power, and, consequently, the processing of high-quality images becomes infeasible. Overall, an adapted autonomous vehicle in emerging countries would need to tackle a high number of damages on the road, different types of road surfaces, and low-resolution images. Nonetheless, most popular datasets used for training deep learning models designed to solve the semantic segmentation of a scene miss these hurdles.

Among the most popular semantic segmentation datasets for road and street scenes are Cityscapes (CORDTS et al., 2016), CamVid (BROSTOW; FAUQUEUR; CIPOLLA, 2009), and KITTI (GEIGER et al., 2013); all recorded in European cities in urban good-condition streets; all of them did not assign any label for damages. Alternatives dataset related to emerging countries are OffRoadScene (SHANG et al., 2013), RoadDamageDetector (MAEDA et al., 2018), and CaRINA (SHINZATO et al., 2016), however, either they do not have great variation in surfaces conditions and damages, or they are unavailable. Attempting to solve this need, (RATEKE; JUSTEN; VON WANGENHEIM, 2019) built the Road Transverse Knowledge (RTK) dataset.

The RTK dataset presents countryside roads of good and precarious qualities of asphalt, paved and unpaved surfaces. They captured more than 77 thousand low-resolution images in different light conditions during the daytime. 701 out of the 77k images are fine-annotated images with 12 classes for the semantic segmentation task. The assigned classes are the road surfaces (asphalt, paved, and unpaved), the road signs (markings, cat's eyes, and speed bumps), the damages and fixes (patches, water-puddles, potholes, and cracks), and storm-drains. As the RTK dataset represents a specific reality, it also raises specific issues presented next.

1.0.1 RTK Issues

The dataset has a highly imbalanced class condition in which the background and the road surface categories count for 98.5% of all pixels, remaining only 1.5% of the pixels for the other eight classes. The imbalanced scenario raises the risk of the model overlooking underrepresented classes or overrunning in false positives.

The dataset has an intentional low-resolution aspect of 352x288, in contrast with other datasets, e.g., Cityscapes use 2048x1024, and CamVid uses 960×720. The resolution disparity may require changes to usual architectures based on high-resolution datasets. Low-resolution images have tiny objects, given the low number of pixels; for example, RTK includes many objects with tiny edges less than or equal to 5 pixels, e.g., *cat's eyes* class has 70% of their objects in this situation; even a more extreme condition of a single-pixel edge occurs in 15% of the objects from the *road markings* class. These objects can vanish at the beginning of the neural network forward, given the stride from convolutional and pooling layers.

Another issue in the dataset is the presence of multiscale elements on the image, e.g., road surfaces are broad and have a well-defined shape, whereas patches do not have a regular shape and size. Multiple shapes with variations even within the same class happen in the same image. This issue becomes more complex when a high number of classes appear in the image, e.g., 17% of the images present five or more classes.

Furthermore, we noticed that the first solution (RATEKE; VON WANGENHEIM, 2021) proposed by the dataset's authors, which we set as the **baseline** for this work, missed some practices found in state-of-the-art projects in the field like cropping and resizing operations for data augmentation, Stochastic Gradient Descent (SGD) optimizer with polynomial learning rate, and a higher number of training iterations. Their first solution setup used U-Net (RONNEBERGER; FISCHER; BROX, 2015) backboned by Resnet34 (HE, K. et al., 2016) which was trained on a two-stage regime; in the first stage, the model was trained with cross-entropy loss by 100 epochs, and in the second stage, the model trained with weighted cross-entropy loss for another 100 epochs. Besides, their setup included horizontal rotations, perspective warping as data augmentation, and Adam optimizer with a 1-cycle learning policy.

In addition, their reported benchmark was 97% of accuracy over the validation set. Although it looks like a high number, accuracy is not the proper metric for a multiclass problem. The metrics commonly adopted on the semantic segmentation works are mean Intersection-over-Union (mIoU) (EVERINGHAM et al., 2015; CORDTS et al., 2016) that averages the IoU individually computed for each class. Thereby, an evaluation of mIoU is needed.

1.0.2 Contributions

This work proposed the **performance increment strategy** to check how the missing practices from state-of-the-art works and how other specific techniques from imbalanced datasets, small objects, and multiscale segmentation works would increment the RTK results, and, consequently, results in emerging country roads datasets.

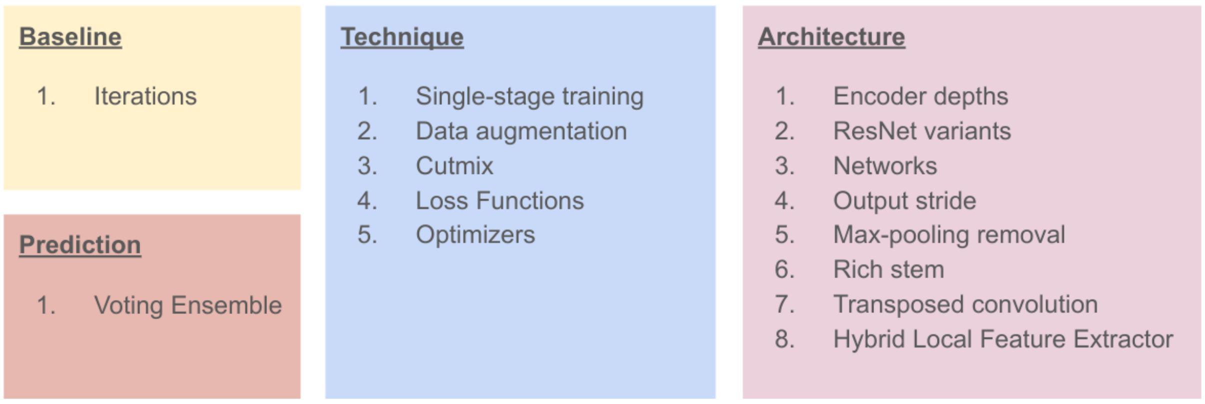
This work tested multiple state-of-the-art techniques in many parts of a deep learning training setup. For example, we tested the usual cropping and resizing along with cutmix as data augmentation, we tested the importance of depth on the architecture encoder, we tested ResNet variants as encoder backbone, and we tested region-based loss functions in comparison to the distribution-based ones; furthermore, we proposed modifications on the network architecture to meet the RTK issues needs like the removal of the max-pooling layer on the ResNet architecture which, to best of our knowledge, was the first time it was done to preserve small objects segmentation.

In the end, the proposed strategy raised the RTK benchmark from 0.549 to 0.798 mIoU. An additional dataset for unstructured environments, TAS500 (METZGER; MORTIMER; WUENSCHE, 2021), was also used to endorse the performance increment strategy, which reached the 0.688 mIoU on the test set evaluation, the best published result so far.

This work also presented an in-depth analysis of the RTK dataset, sharing statistics and visual analysis of small and multiscale objects. Besides, performance analysis of the trained models was presented by classes or groups of classes like *small objects*, *small edges*, and *underrepresented*.

1.1 PROPOSAL

The performance increment strategy consists of an additive series of ablation experiments. Each experiment checks the best performance between a set of hypotheses; for example, which augmentation operation works better for the RTK dataset? Geometry operation? Color operations? Both together? Each of these options is a hypothesis. The ablation experiments answer the questions of the best hypothesis for a part of the training setup. The next ablation experiment on the series is built over the best hypotheses acquired until that stage.


Baseline

1. Iterations

Prediction

1. Voting Ensemble

Technique

1. Single-stage training
2. Data augmentation
3. Cutmix
4. Loss Functions
5. Optimizers

Architecture

1. Encoder depths
2. ResNet variants
3. Networks
4. Output stride
5. Max-pooling removal
6. Rich stem
7. Transposed convolution
8. Hybrid Local Feature Extractor

Figure 1 – The performance increment strategy experiments.

The performance increment strategy consists of 15 ablation experiments organized into four categories: baseline, technique, architecture, and prediction; see Figure 1. Baseline and prediction categories include a single ablation experiment, each. The baseline category has the iteration experiment that keeps the exact baseline setup and tests only the number of iterations. Meanwhile, the prediction category has a voting ensemble experiment that tests an ensemble of predictions from the same image may help.

The technique category contains setups that surround the training like pre-processing and optimizer choices; the category includes five experiments: single-stage training, data augmentations, cutmix, loss functions, and optimizers. Finally, the architecture category, the largest one, says about any modification on the network and includes eight ablation experiments: encoder depths, ResNet variants, networks, output strides, max-pooling removal, rich stem, transposed convolutions, and local feature extractor.

The list below shows the series of ablation experiments in their application order; each item follows the category name, the experiment name, and its central question; the subitems outline the experiment's hypotheses. One note, the hypotheses options may be reduced or simplified to avoid overextending the list, mainly the "yes/no" answers; for complete visualization of the hypotheses, check out the experiments section 4.3.

1. Baseline: Iterations - Do more training iterations help?
    a) 14k iterations
    b) 100k iterations
    c) 200k iterations
2. Technique: Single-stage Training - Does single-stage work better?
    a) Single-stage with cross-entropy
    b) Single-stage with weighted cross-entropy
    c) Two-stages
3. Technique: Data Augmentations - Which augmentation operation works better?
    a) Crop
    b) Cropping & Color
    c) Cropping & Resizing
    d) Geometry
    e) Cutmix
    f) None
4. Architecture: Encoder Depths - Which backbone depth works better?
    a) Resnet-34
    b) Resnet-50
    c) Resnet-101
5. Architecture: ResNet Variants - Which variant works better?
    a) Resnet
    b) ResNeXt

    c) ResNeSt
    d) Res2Net

6. Architecture: Networks - Which network works better?
    a) U-Net
    b) DeepLabV3+
7. Architecture: Output strides - Which output stride works better?
    a) 4
    b) 8
    c) 16
8. Architecture: Max-Pooling Removal - Does removing the max-pooling layer help?
    a) Yes
    b) No
9. Architecture: Rich Stem - Does a richer ResNet stem help?
    a) Yes
    b) No
10. Architecture: Transposed Convolution - Does upsampling with learnable parameters help?
    a) Yes
    b) No
11. Architecture: Hybrid Local Feature Extractor - Do fancy approaches like degressive and hybrid dilation rates help?
    a) Yes
    b) No
12. Technique: Cutmix - Does cutmix along with another augmentation help?
    a) Yes
    b) No
13. Technique: Loss Functions - Which loss function works better?
    a) CE
    b) WCE
    c) Dice
    d) mIoU
    e) CE + Dice

  f) CE + mIoU

14. Technique: Optimizers - Which optimizer works better?
    a) SGD
    b) Adam
15. Prediction: Voting Ensemble - Does voting ensemble help?
    a) Default
    b) Default + Flipped versions
    c) Default + Flipped + Multiscale versions

1.2 OBJECTIVES

The main objectives are:

- Analyze quantitatively and qualitatively the characteristics of the RTK dataset.
- Develop a strategy to improve semantic segmentation in emerging country realities, i.e., the performance increment strategy.
- Carry out the performance increment strategy on the RTK dataset.
- Analyze quantitatively and qualitatively the strategy results.
- Carry out the performance increment strategy on the TAS500 dataset.
- Summarize the main findings from the strategy execution in two different datasets.

1.3 THESIS ORGANIZATION

Chapter 2 explains the major deep learning topics used during this work. Chapter 3 points out the critics of the RTK's baseline and suggests areas for improvement; moreover, the chapter presents related works that handled the segmentation of small and multiscale objects. Chapter 4 shows the application of the performance increment strategy on the RTK dataset; each experiment has a subsection that shows the results of each hypothesis; the chapter ends with quantitative and qualitative analyses between the hypotheses and groups of classes. Chapter 5 shows the application of a shortened performance increment strategy on the TAS500 dataset. Finally, the last chapter discusses the principal findings and observations of the strategy results, summarize the conclusions, and suggests possible future works.

2 DEEP LEARNING BACKGROUND

AlexNet's performance on the ImageNet Large Scale Visual Recognition Challenge in 2012 (KRIZHEVSKY; SUTSKEVER; HINTON, G. E., 2012) was a disruptive moment in deep learning and artificial intelligence communities. The authors' methods achieved a top-5 error of 15.3%, a considerable advantage of 10.8% from the second place, which got 26.1%. From then on, deep learning was a game-changer in computer vision tasks and other engineering fields, becoming the first choice when looking for a solution to most artificial intelligence problems in academia and industries.

2.1 DEEP LEARNING

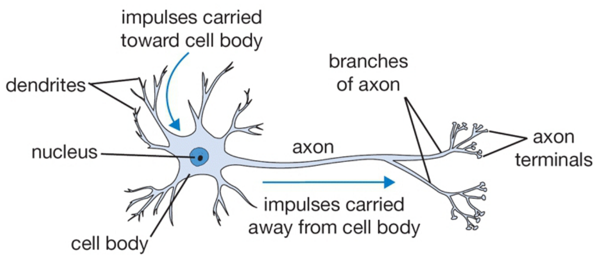
Deep learning is a subset of machine learning algorithms based on artificial neural networks (NN) that are deep enough to learn features from raw input data, i.e., without a pre-processing step of feature extraction. In addition, it commonly solves an optimization problem that minimizes a loss function by a gradient-based algorithm like SGD and the back-propagation algorithm to estimate such gradients.

As cited in (LECUN; BENGIO; HINTON, G., 2015), deep learning allows computers to learn from experience and understand the world in terms of a **hierarchy of concepts**, such complex concepts are built on top of the simpler ones. And if we draw a graph showing how these concepts are built on top of each other, the graph would be deep, with many layers.

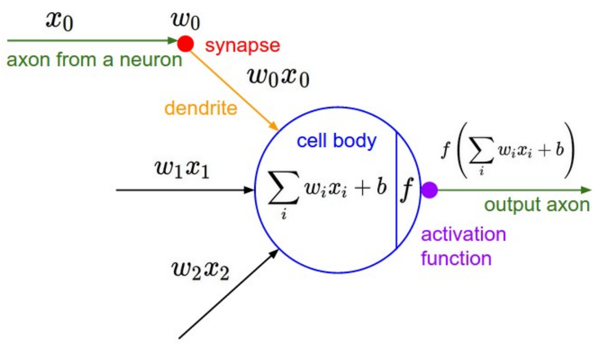
The ability to learn useful features from the raw input data, e.g., pixel values, audio frames, and text sentences, is called **representation learning**. Learned representations often result in much better performance than hand-designed representations (LECUN; BENGIO; HINTON, G., 2015). Representation learning is a fundamental ability of deep learning, and that is what makes it so remarkable. Meanwhile, most of the traditional machine learning algorithms use hand-designed features.

Although the massive popularization of deep learning happened after AlexNet's event, many of its core concepts can be traced to decades ago. The mathematical model of the neuron was first published in 1943 (MCCULLOCH; PITTS, 1943); and in 1959 appeared the Perceptron (ROSENBLATT, 1958), the first neural network learned by error correction (See Figure 2). The decade of the 1960s started with a lot of excitement (KURENKOV, 2020); however, it was soon declared that the neural networks could not learn more complicated logical functions like XOR (MINSKY; PAPERT, 1969). As there was no algorithm at the time that allowed them to update parameters for multilayered networks, they only trained a single-layer network, which explains the limitation found.

After a lengthy period, (RUMELHART; HINTON, G. E.; WILLIAMS, 1985) overcame this limitation by adapting the back-propagation algorithm, previously derived in the control theory in the 1960s, and the differentiable activation function, such as the sigmoid function in opposition to the perceptrons' threshold function (See Figure 3). These new ideas make it possible to train a multilayer network. Moreover, the same work presented the concept of **distributed representation** which a pattern is represented by multiple neurons instead of a single neuron, e.g., a red cat image should activate the "red" and the "cat" patterns instead of a single "red cat" pattern. This concept extends to the idea that many features should represent an input, and each feature should represent many inputs as possible.

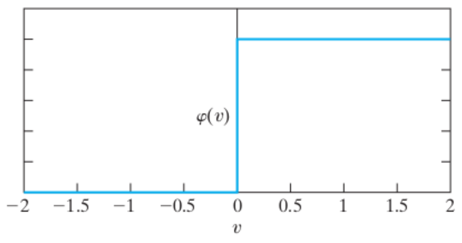

impulses carried toward cell body
dendrites
branches of axon
nucleus
axon
axon terminals
impulses carried away from cell body
cell body

(a) A biological neuron.

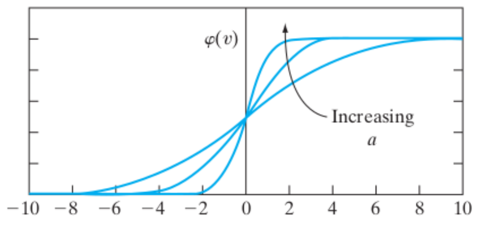


(b) A perceptron model.

Figure 2 – A biological and an artificial neuron. From (KARPATHY et al., 2016).



(a) The non-differentiable threshold function used on (ROSENBLATT, 1958).



(b) A differentiable activation function.

Figure 3 – From (HAYKIN, 2009).

Training neural networks with two or more hidden layers were computationally expensive until the 2010s because of the hardware available then; it did not allow much experimentation, causing the wane of DL many times. Nevertheless, since 2006, the third and last deep learning wave, the popularity and the achievements of deep learning only have grown (LECUN; BENGIO; HINTON, G., 2015). Much of this success is due to the access of powerful computers and larger datasets, which allowed much experimentation, exhausting training, and eased the burden of converging neural networks. In addition to the new resources, latter techniques also enabled more powerful model; for example, dropout (SRIVASTAVA, N. et al., 2014) and batch normalization (IOFFE; SZEGEDY, 2015) encourage better generalization, batch normalization also accelerates convergence time, and residual learning (HE, K. et al., 2016) enables deeper architectures.

The quintessential example of a deep learning model is a Deep Feedforward Networks (DFN) or MultiLayer Perceptrons (MLP).

2.1.1 Deep Feedforward Networks

DFNs are name feedforward because they only flow the input information through the hidden layers until the output layer. The network is called a recurrent neural network if there are also feedback connections. The name *network* refers to the composition of many different functions, typically in a chain fashion, e.g., $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(\mathrm{x})))$, where each $f^{(n)}$ is a layer. The chain length defines the network's depth, to which the terminology *deep* is related.

The final layer of the neural network is the *output layer*, and the previous ones are the *hidden layers*. The dimensionality of the hidden layers, i.e., the quantity of units (neurons), determines the *width* of the layer.

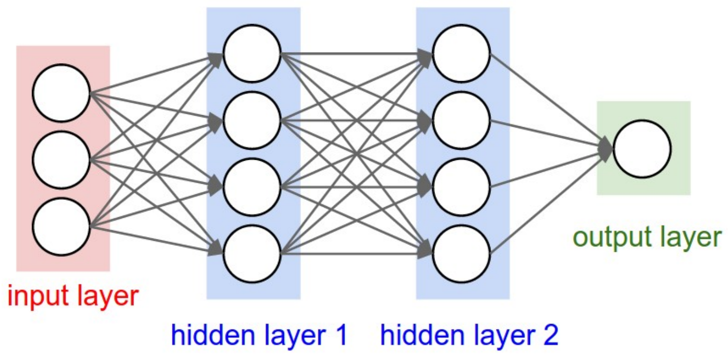


Figure 4 – A 3-layer feedforward neural network. From (KARPATHY et al., 2016).

2.2 CONVOLUTIONAL NEURAL NETWORKS

Convolutional Neural Network (CNN) are perhaps the greatest success story of artificial intelligence biologically inspired. Though convolutional networks have been guided by many other fields, some of the key design principles of neural networks were drawn from neuroscience (LECUN; BENGIO; HINTON, G., 2015).

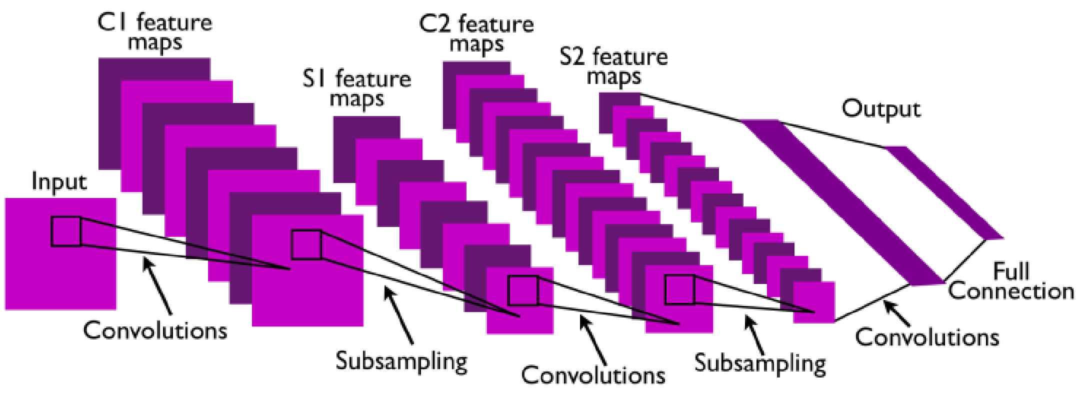


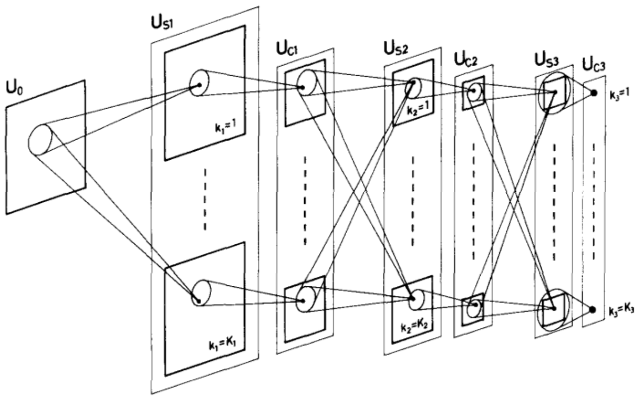
Figure 5 – A typical CNN architecture with two feature stages. From (LECUN; KAVUKCUOGLU; FARABET, 2010)

2.2.1 A Brief History

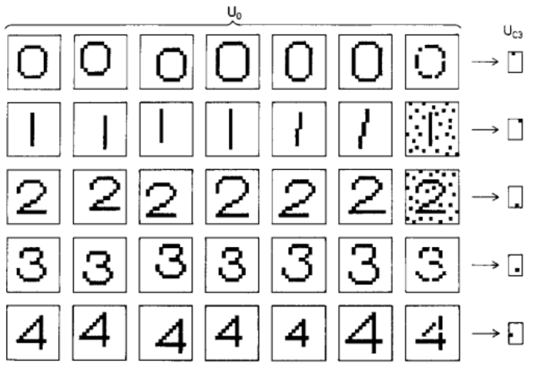
The core idea behind CNN came from the 1962 work on the cat's primary visual cortex (HUBEL; WIESEL, 1962) that identified the presence of simple and complex cells. Simple cells have a role similar to the first convolutional layers, where they detect simple features such as edges or lines in a particular orientation. Complex cells present a role similar to the pooling layer, which its activation is independent of small shifts around a neighborhood. The authors were awarded a Nobel prize for this and other mammalian vision system work. A brief disclaimer, despite some similarities, there are much more differences between CNNs and the mammalian vision system as listed on page 367 of the Deep Learning book (LECUN; BENGIO; HINTON, G., 2015).

The first computer model inspired by this cell's arrangement was the Neocognitron (FUKUSHIMA; MIYAKE, 1982), which presented a self-organized structure that alternated between simple cells (S-cells) and complex cells (C-cells). Neocognitron was trained in an unsupervised fashion; the S-cells made plastic connections with the previous layer, the connections were modifiable during the learning, some were strengthened, and others were inhibited; C-cells had fixed connections.

Neocognitron could recognize stimulus patterns based on the geometrical similarity of their shapes without being affected by their position or slight distortions. That was the first deep network conceived (SCHMIDHUBER, 2015), it counted five hidden layers. Before that, Fukushima was also one of the first researchers to develop a multilayer network (FUKUSHIMA, 1975).



(a) The Neocognitron architecture: an input layer followed by an alternate connection between S-cells U_S and C-cells U_C



(b) Examples of distorted input patterns correctly recognized in the last layer U_{C3} showed on the last columns, where it renders the activated cell of that layer.

Figure 6 – From (FUKUSHIMA; MIYAKE, 1982).

The subsequent Yann Lecun works (LECUN et al., 1989a, 1989c, 1989b) simplified and better organized Fukushima's architecture; besides, he adopted back-propagation to train the entire network in a supervised fashion, consequently, creating the modern version of CNN known these days. In the early 1990s, CNNs were already present in commercial applications as an operational bank check reading system running on ATMs in Europe and US; by the late 1990s, it had read over 10% of all the checks in the US (LECUN et al., 1998; LECUN; KAVUKCUOGLU; FARABET, 2010).

(LECUN et al., 1989b) expressed that a good generalization performance can be obtained if some prior knowledge about the task is built into the network. Therefore, as many image tasks used kernels to extract features from local patches and later combine them for object shape recognition, they designed the network to accomplish the same by constraining the connections to be local, i.e., the neurons would have a local receptive field from the previous layer, so each neuron was connected only to a small region like a 3x3 or a 5x5 patch.

In addition, Lecun's work adopted weight sharing, a technique first described in (RUMELHART; HINTON, G. E.; WILLIAMS, 1985) that constrains units from a feature map (a channel of a convolutional layer) to perform the same operation so that a plane of the feature map is the same feature detected at different input locations. Suppose that if a feature detector is useful in one part of an image, it is likely to be useful in other parts.

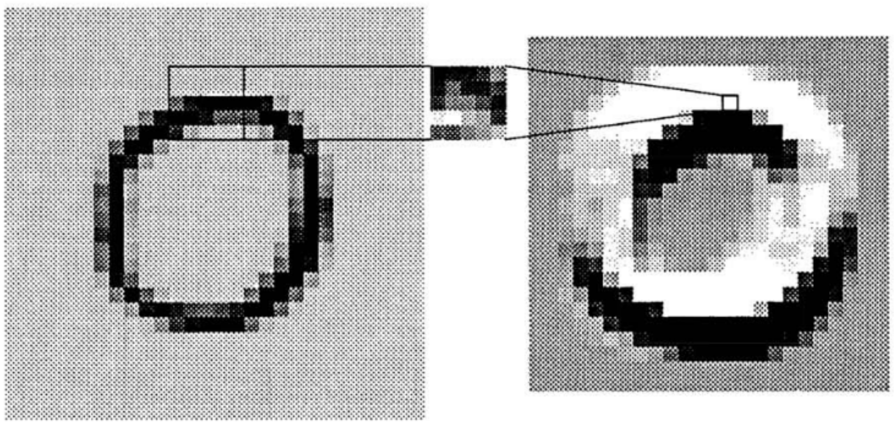


Figure 7 – (left) Input image; (center) the channel's weight parameters (kernel); its size determines the receptive field of a single neuron, which is 5x5 in this case; (right) the computed feature map. From (LECUN et al., 1989b)

Weight sharing can be interpreted as imposing an equal constraint on the strengths of the connections for all neurons of a layer. Weight sharing changed the learning paradigm from the space of connection strengths to the space of parameters; thus, the transformation parameters determine the connection's strength (LECUN et al., 1989a).

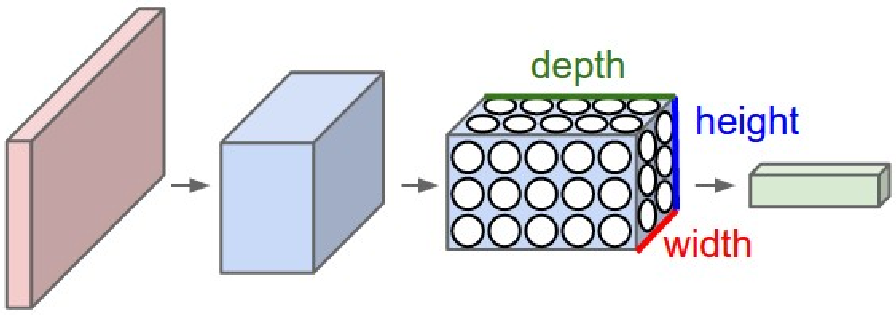


Figure 8 – A convolutional layer arranges its neurons in three dimensions (width, height, and depth). The convolutional layer transforms the 3D input volume into a 3D output volume of neurons' activations. In this example, the red input layer holds the image, and its volume is transformed along the network. From (KARPATHY et al., 2016).

After LeCun's work on check readings, CNNs leveraged other commercial applications. Microsoft deployed several OCR and handwriting recognition systems, including Arabic and Chinese characters. In the 2000s, Google deployed CNNs to detect face and license plates in Street View images for privacy concerns (LECUN; KAVUKCUOGLU; FARABET, 2010); and CNNs also won some competitions (SCHMIDHUBER, 2015). Nevertheless, the worldwide interest just boomed when AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, G. E., 2012) won the ImageNet object recognition challenge (LECUN; BENGIO; HINTON, G., 2015).

2.2.2 Properties

Three essential ideas support convolutional neural networks: sparse interactions, parameter sharing, and equivariant representations (LECUN; BENGIO; HINTON, G., 2015).

Sparse interactions happen when only a few of all possible interactions between the neuron's layers and their inputs are present, e.g., local connectivity. On CNNs, sparse interactions occur by making the kernel (the neuron's weights) size smaller than the inputs, so we have far fewer parameters than connections. For example, when processing an image, the input image might have thousands or millions of pixels, but we can detect small, meaningful features such as edges with kernels that occupy only tens or hundreds of pixels. It means that we need to store fewer parameters, which reduces the memory requirements of the model, speeds up the runtime, and improves the statistical efficiency (the model's parameters are closer to the true parameters of the data distribution).

Parameters sharing or weight sharing means that rather than learning a separate set of parameters for every location, we learn only one set. This property leads to equivariance to translation. A function is equivariant if the input changes so that the output changes in the same way; despite, convolutions are not equivariant transformations in scale or image rotation (LECUN; BENGIO; HINTON, G., 2015).

As said about sparse interactions, a convolutional layer uses local connectivity. The spatial area of connectivity is called the neuron's receptive field, which is determined by the kernel size attached. The standard size of 2D kernels is 3x3; however, sizes 1x1, 5x5, and 7x7 are also frequent. In addition to the 2D plane, a kernel usually connects throughout the input's depth; e.g., if an input has eight planes of the feature map, the kernel should have a dimension of 3x3x8.

The receptive fields affect the network's performance. Larger receptive fields get more context and represent more abstract meanings (LIN; SHI; ZOU, 2017), whereas small receptive fields may not be able to recognize large objects. As the information flows through the neural network, the receptive fields of deeper neurons increase, as shown in Figure 9. Pooling layers also increase the receptive field, but at the cost of lowering the resolution.

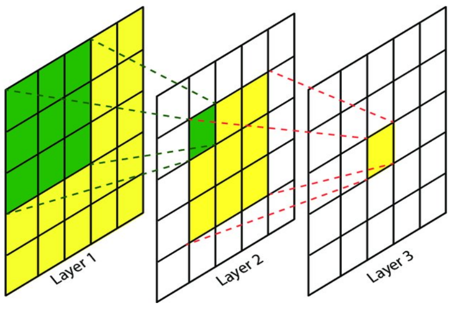


Figure 9 – The receptive field of each convolutional layer neuron is based on a 3×3 kernel. The green and yellow areas mark the receptive field of a single pixel in layers 2 and 3, respectively. From (LIN; SHI; ZOU, 2017).

2.2.3 Typical Components

Three stages compose a typical convolutional operation: a convolutional layer that produces a set of linear activations, a nonlinear activation function, e.g., the rectified linear, and a pooling operation. See Figure 10.

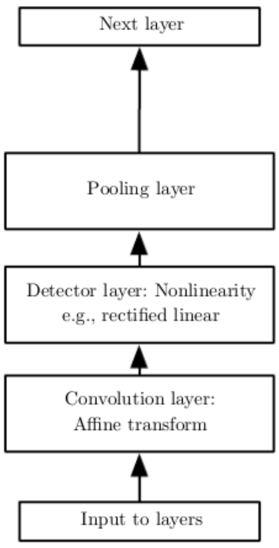

Next layer
Pooling layer
Detector layer: Nonlinearity e.g., rectified linear
Convolution layer: Affine transform
Input to layers

Figure 10 – A typical convolutional neural network layer. From (LECUN; BENGIO; HINTON, G., 2015).

2.2.3.1 Convolution Operation

The discrete convolution operation on a two-dimensional image is defined as:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n) \tag{1}$$

I is the input, i.e., the image pixels, K is the kernel, m and n are the dimensions of the kernel, i.e., the kernel size, and S is the feature map. Despite the convention in the machine learning community to call it convolution, the correct name of the operation is cross-correlation (LECUN; BENGIO; HINTON, G., 2015). See Figures 11 and 12.

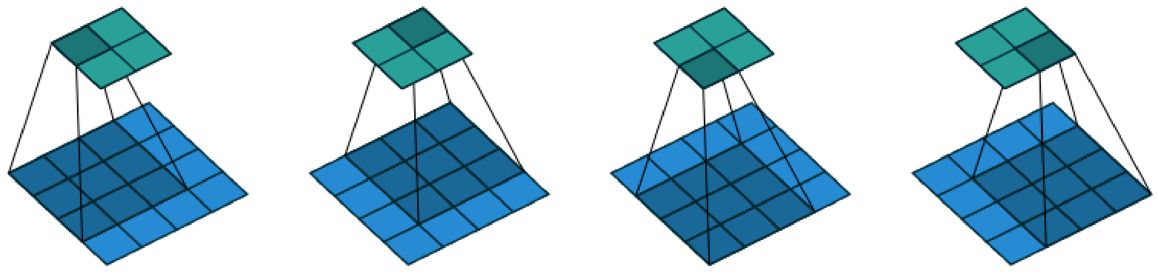


Figure 11 – Example of a convolution operation of kernel 3x3 sliding across every 4x4 input position (stride 1) and no padding, resulting in a 2x2 output. From (DUMOULIN; VISIN, 2016).

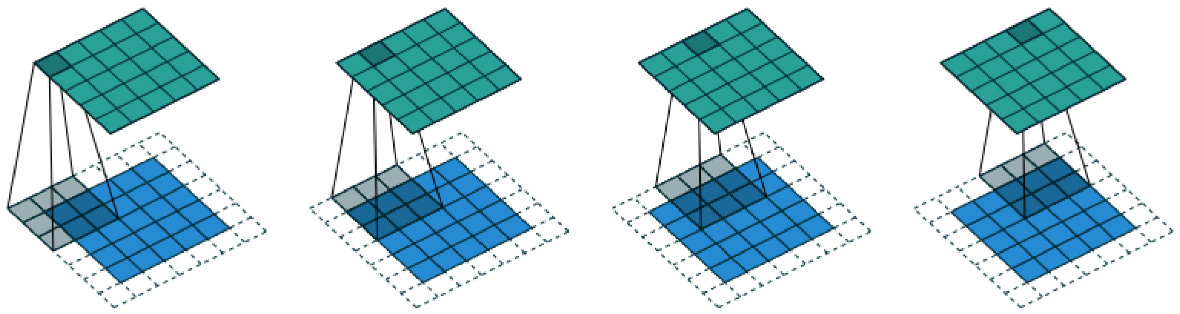


Figure 12 – Example of a convolution operation of kernel 3x3 sliding across every 5x5 input position (stride 1) and padding 1, resulting in an output of the same size as the input. From (DUMOULIN; VISIN, 2016).

2.2.3.2 Activation Function

The neuron's final value comes from a nonlinear function that enables mapping complex functions. The default choice on modern networks is ReLU (REctified Linear Unit). This function is almost linear because it joins two linear pieces, see Figure 13. Its adoption occurred after the 2010s to avoid the *gradient vanishing* problem, which typically happens on sigmoid and tahn activation functions.

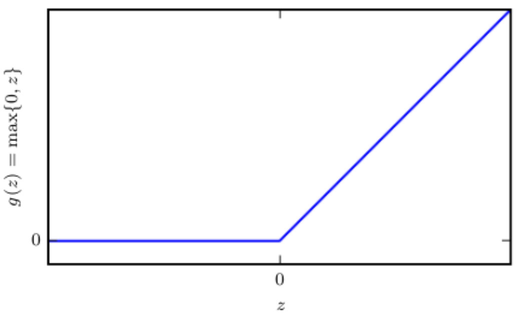
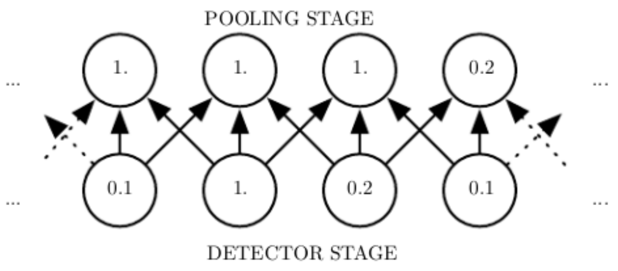


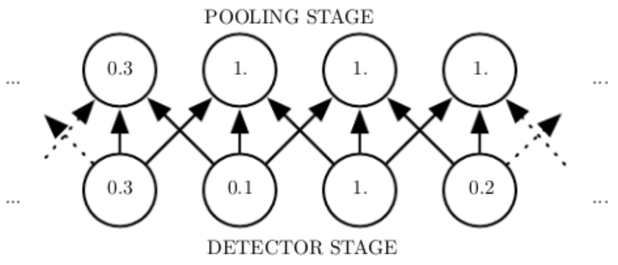
Figure 13 – The ReLU activation function. From (LECUN; BENGIO; HINTON, G., 2015)

2.2.3.3 Pooling

Like convolution operations, pooling operations are handled by a kernel that slides across the entire image. The most common pooling operation of CNNs is the maximum operation which yields the maximum value inside a neuron's neighborhood. Other pooling operations are average and L2-norm. The most important property of pooling is to make the layers' representations translation invariant, e.g., we do not want to know where a cat is in the image, such that we detect its presence. See Figure 14.



(a) Pooling operation of stride 1 and kernel size of 3.



(b) The same operation as (a) after the input is shifted to the right by one pixel. Every value in the lower neurons changed, but only half of the neurons changed in the upper part. The minor changes in the results happened because the max-pooling units are only sensitive to the maximum value in the neighborhood, not its exact location.

Figure 14 – Illustration of max pooling translation invariance. From (LECUN; BENGIO; HINTON, G., 2015).

2.2.3.4 Stride

Stride determines the kernel's pace over the input. Suppose stride 1; then, one convolution occurs centered at every pixel of an image. Meanwhile, in the case of stride 2, as in Figure 15, the input space is downsampled (also known as subsampling) by a factor of the same number, in this case, a factor of 2. Stride greater than 1 reduces the computational cost and increases the *receptive field* of the next layer.

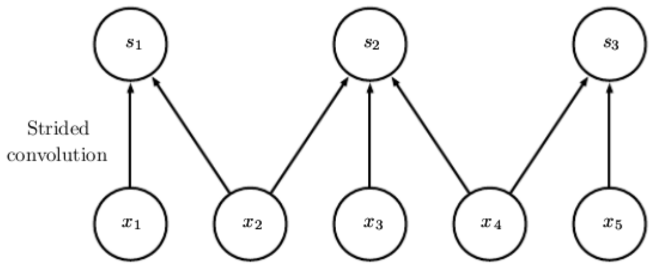


Figure 15 – Strided convolution of 2. From (LECUN; BENGIO; HINTON, G., 2015).

One of the essential hyperparameters for segmentation convolutional network is the Output Stride (OS). It defines the ratio between the sizes of the input image and the final feature map. The input size is typically downsampled through the network to produce higher-level features at the expense of resolution reduction.

2.2.3.5 Dilated Convolution

This technique was first developed in (HOLSCHNEIDER et al., 1990) and named atrous convolution; however, the latter work in deep learning (YU, F.; KOLTUN, 2015) named it dilated convolution. They are like regular convolutions with an expansion mechanism that enlarge the kernel's receptive field filling its length with zeros ("holes"). The expansion degree is controlled by a **dilation rate**.

Suppose a 3x3 kernel size; if the dilation rate is 1, it behaves just like the standard convolution with a receptive field of 3x3. However, if the dilation rate is 2, it enlarges the receptive field to 5x5. If the rate is 3, the receptive field goes to 7x7. See Figure 16.

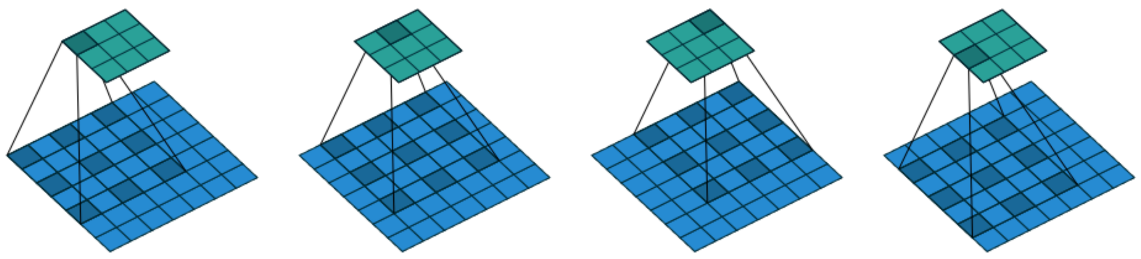


Figure 16 – Dilated convolution of kernel 3x3 (gray square) over a 7x7 input (blue square) resulted in a 3x3 output (green square). A factor of *rate*−1 controls the space between the kernel's weights. From (DUMOULIN; VISIN, 2016).

2.2.3.6 Transposed Convolution

This operation is a parametrized way to upsample the feature map. Sometimes, it is miscalled *deconvolution*. Transposed convolutions on feedforward mode apply the backward pass of a traditional convolutional layer. It upsamples a feature map with learnable parameters in opposition to interpolation methods. An important difference between conventional and transposed convolutional layers is that the stride reduces the input dimension for the first, whereas, for the second, it increases the dimension; inserting holes between the input neurons, see Figure 17.

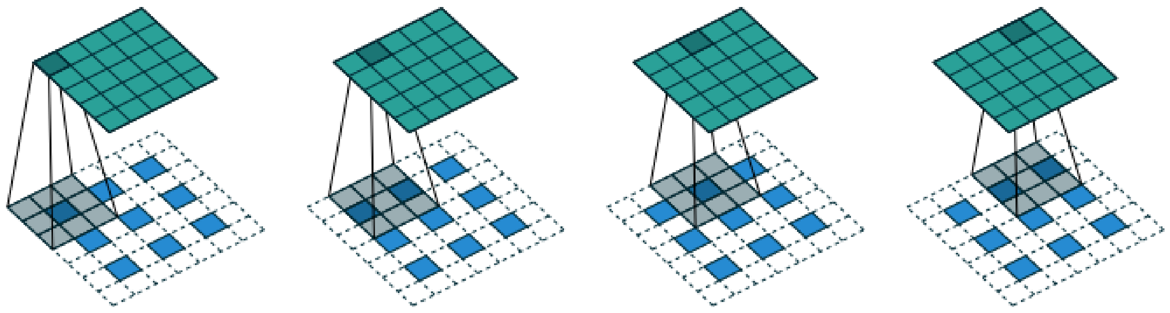


Figure 17 – Transposed convolution of kernel 3x3 (gray square) over a 3x3 input (blue squares) padded with a 1x1 border and stride 2x2 resulted in a 5x5 output (green square). The stride's backward operation creates holes (filled with zero) between the input neurons. From (DUMOULIN; VISIN, 2016).

2.2.4 ResNet

ResNet (HE, K. et al., 2016) is a classical CNN and usually the default choice for an encoder backbone of a semantic segmentation network. ResNet, inspired by the highway networks (SRIVASTAVA, R. K.; GREFF; SCHMIDHUBER, 2015), brought the residual learning concept, which enabled the training of very deep networks. Before this, training very deep networks had the problem of increasing training error after a period.

Their solution consisted of a *residual* mapping operation. Suppose deeper layers fail to fit at least an *identity* mapping from earlier layers, then it would be easier to learn the weights to zero. Therefore, it would avoid deteriorating information from shallow to deep layers. See Figure 18.

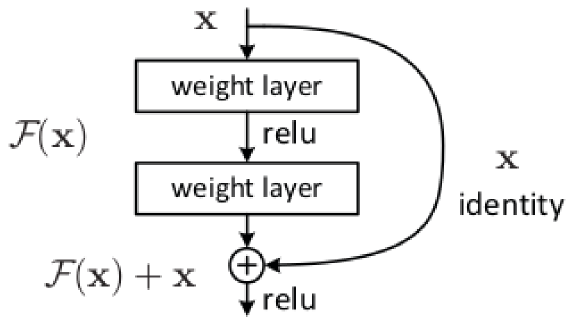
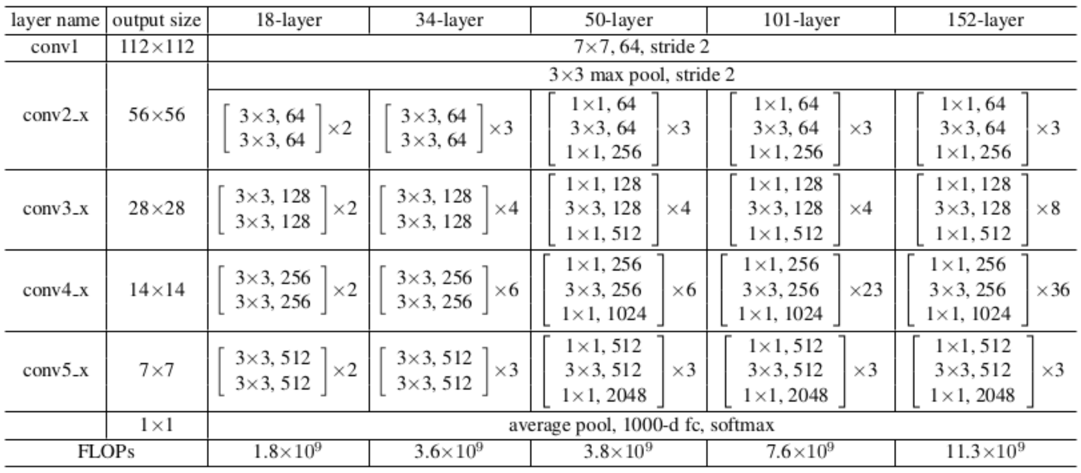

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 18 – A building block of residual learning. From (HE, K. et al., 2016).

The ResNet architecture consists of five parts: a stem composed of a single convolutional layer of kernel 7x7 and stride 2, followed by four convolutional blocks. The start of each convolutional block contains a convolutional layer of stride 2 followed by a repeated series of convolutional layers of stride 1. Deeper architectures keep the same four blocks arrangement but with more repetition of layers in blocks 2 and 3. The main branches of ResNet have depths of 34, 50, and 101; they are respectively called ResNet34, ResNet50, and ResNet101. See Figure 19.

20.



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 19 – ResNet architectures. From (HE, K. et al., 2016).

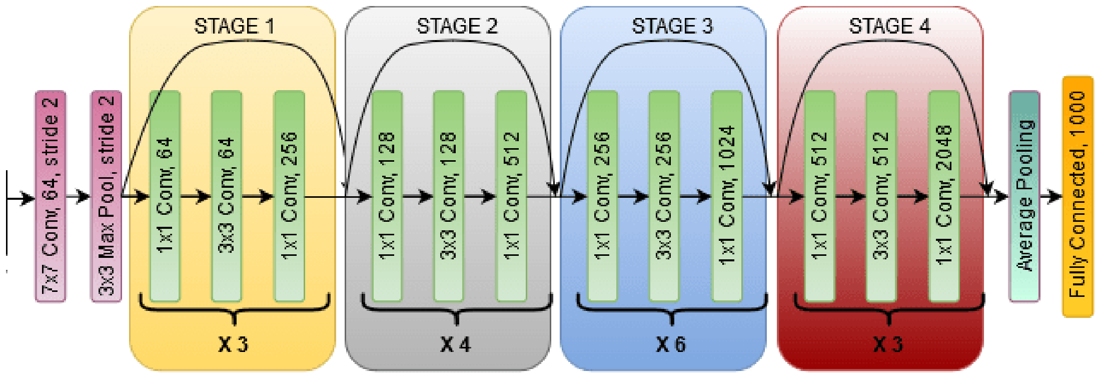


Figure 20 – ResNet50 with the stem of kernel size 7x7 followed by four blocks. From (KHAN et al., 2020).

2.2.4.1 Dimensionalities

The number of layers on the network defines the depth of the models. The number of units on a layer defines the layer's width. The widest width defines the network's width. Other ResNets variants proposed new dimensionalities; for example, cardinality (XIE et al., 2017) which splits a single convolutional layer path into parallel paths of distinct convolutional layers; another example is scale (GAO et al., 2019) that groups multiscale features inside of a single convolutional path.

2.3 SEMANTIC SEGMENTATION

Semantic segmentation stands for understanding the image at the pixel level, where each pixel has a class assigned. Semantic segmentation is a dense prediction task, i.e., its outcome has the same size as the input. Its applications are found in autonomous driving, medical imaging, image search engines, augmented reality, object detection, and video surveillance.

In the past, tradition computer vision algorithms used were Watershed (BEUCHER, 1979), Image thresholding (OTSU, 1979), and K-means clustering (LLOYD, 1982). Nowadays, after AlexNet's breakthrough, the dominant solution for semantic segmentation is deep learning.

This section will show the main parts of deep learning training for semantic segmentation: the neural network architectures, data augmentation techniques, loss functions, and evaluation metrics.

2.3.1 Neural Network Architectures

Most state-of-the-art semantic segmentation architectures are based on Fully Convolutional network (FCN) (LONG; SHELHAMER; DARRELL, 2015), which was the first network trained end-to-end for semantic segmentation. Currently, the term FCN is ambiguous; it may also refer to any CNN containing only convolutional, pooling, and activation layers. FCN consists of an encoder session in charge of extracting high-level features and increasing the receptive fields of the neurons; and a decoder session in charge of upsampling the feature map to the input dimensions.

During the encoder feedforward pass, max-pooling and stride convolutions downsample the feature map inevitably. The most common approaches to resize the feature map back are bilinear interpolation as in (CHEN et al., 2017), which is fast and requires fewer resources than transposed convolution (Sec. 2.2.3.6) as in (LONG; SHELHAMER; DARRELL, 2015; NOH; HONG; HAN, 2015)), which has learnable parameters.

Other architectures developed later based on FCN were U-Net (RONNEBERGER; FISCHER; BROX, 2015) and DeepLabV3+ (CHEN et al., 2018).

2.3.1.1 Fully Convolutional Networks

The FCN's core idea was to make the neural networks take an input of any size and produce a correspondingly-sized output. The typical constraint of a neural network accepting only fixed-sized inputs comes from the densely connected layers, which have fixed dimensions, and, thereby, discard spatial information. FCN overcame this hurdle by adopting only convolutional and pooling layers.

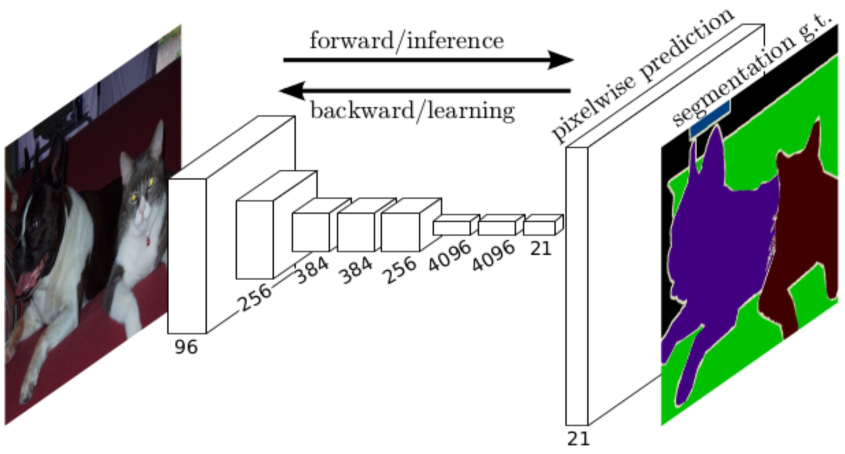


Figure 21 – FCN architecture. From (LONG; SHELHAMER; DARRELL, 2015).

In the original paper, it was proposed some FCN variants. There are the straight network FCN-32s with output stride 32 (see Figure 21), e.g., suppose if an input size is 256x256, then the most coarse feature map would be 8x8; later, the FCN-32s would recover with transposed convolutions the coarse map to a dense output, which has a prediction at each input pixel. The other variants are FCN-8s and FCN-16s that combine information from coarse feature maps at multiple output stride; for example, FCN-16s sums the feature map of OS 16 to the twice upsampled feature map of OS32.

2.3.1.2 U-Net

U-Net is a classical architecture for semantic segmentation. It was first proposed for medical image segmentation in 2015, and it turned out to be applicable in many other areas. The architecture consists of a contracting path, which captures the context, and a symmetric expanding path, which enables precise localization. The network is symmetrical, reminding a U-like shape. See Figure 22.

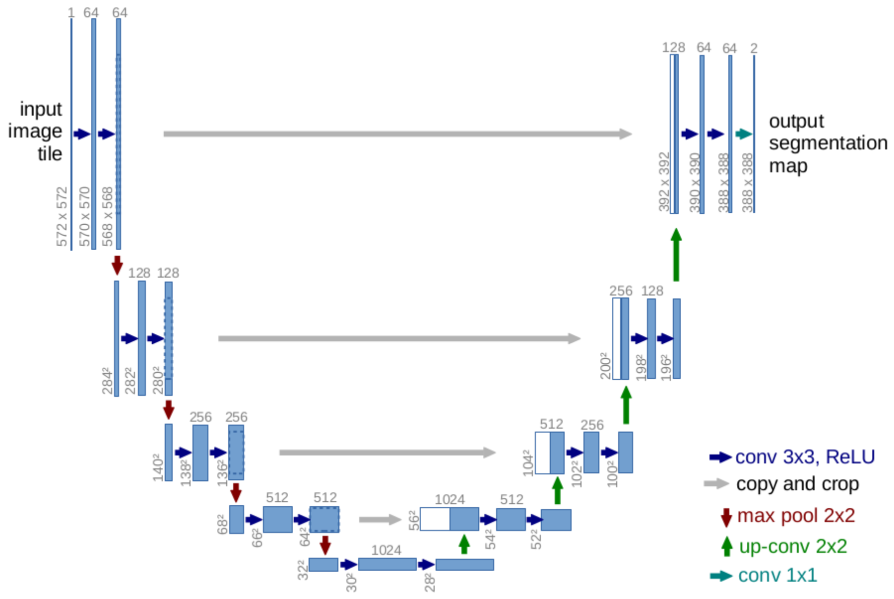


Figure 22 – U-Net architecture. From (RONNEBERGER; FISCHER; BROX, 2015).

2.3.1.3 DeepLabV3+

DeepLabV3+ results from a series of enhancements since the first DeepLab was introduced in (CHEN et al., 2017). The main concepts present in this architecture are the decoder module, the depthwise separable convolution, and the Atrous Spatial Pyramid Pooling (ASPP).

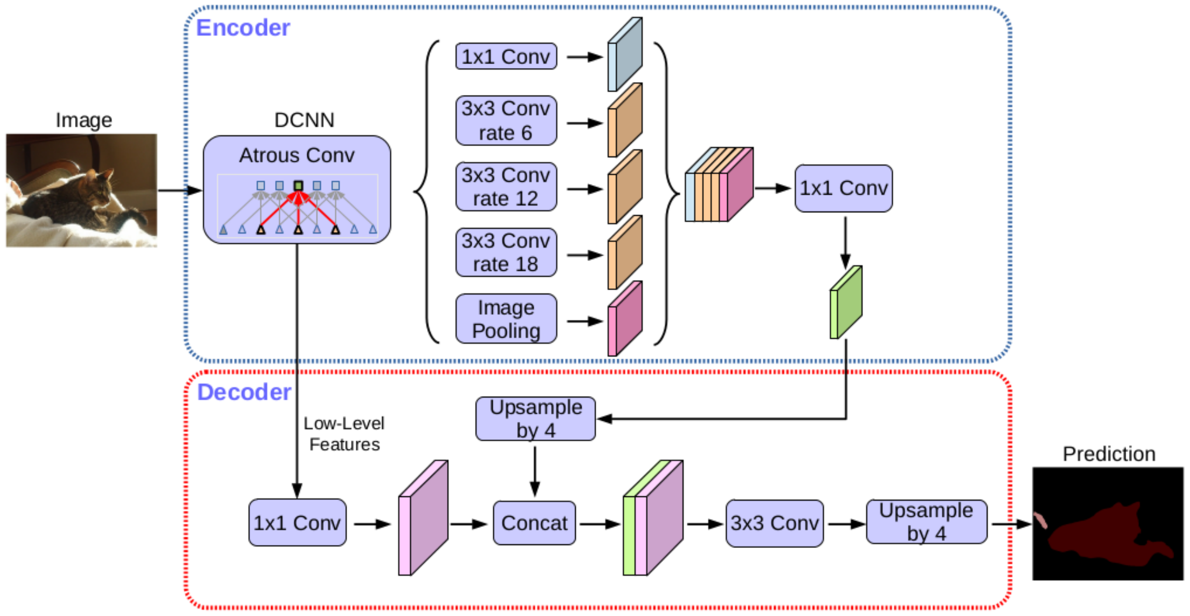


Figure 23 – DeepLabV3+ architecture. From (CHEN et al., 2018).

Depthwise separable convolutions are grouped convolutions followed by a point-wise convolution. On grouped convolutions, the kernels of one layer do not connect to all channels of the previous layer like a standard convolutional layer; each kernel connects only to a group of channels from the previous layer. Point-wise convolutions are convolutional layers of kernel 1x1 which combine the feature maps between the channels and control the final number of channels.

ASPP was inspired by Spatial Pyramid Pooling (LAZEBNIK; SCHMID; PONCE, 2006; HE, K. et al., 2015). ASPP helps to handle multiple scales objects (CHEN et al., 2017). ASPP simultaneously applies multiple dilated convolutions; each dilated convolution has a distinct rate that yields multiple receptive fields. ASPP helps to balance the trade-off between accurate localization (small receptive field) and context assimilation (large receptive field) (CHEN et al., 2017).

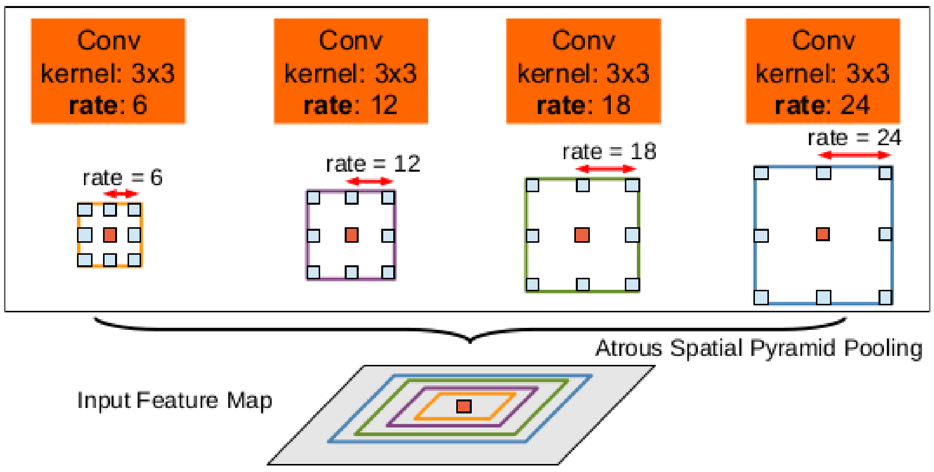


Figure 24 – Atrous Spatial Pyramid Pooling (ASPP). ASPP employs parallel dilated convolutions with different rates to classify the center pixel (orange). From (CHEN et al., 2017).

2.3.2 Data Augmentation

Collecting labeled data is expensive, especially fine-annotated labels like semantic segmentation. For example, an annotation using proper label tools in the Cityscapes dataset took around 1.5 hours for a single image (CORDTS et al., 2016). The high cost of collecting labeled data causes a shortage of data; one workaround to handle it is data augmentation.

Data augmentation expands training data by reusing training samples. Common modifications in images are brightness, rotation, and scale. The modified images become new samples on the training set. These new modified samples add variance to the training distribution that helps the model generalization for predicting unseen examples.

We roughly divided the augmentation transformations into four categories: geometric, resampling, color, and synthetic.

2.3.2.1 Geometry

Geometry transformations map images' pixels to new spatial positions. The labels must receive the same transformation. Examples of geometry transformations are horizontal flipping, random rotation, and random perspective.

Horizontal flipping just mirrors the image on the x-axis. Random rotation revolves the image around its center, typically inside a range of -30 to +30 degrees; these values are heuristically used in the community (TAYLOR; NITSCHKE, 2018). Random perspective, also known as homography, alters the plane of the image; a single parameter controls the distortion level.

2.3.2.2 Resampling

Resizing or cropping images are resampling examples. *Resize* allows working with objects from the same image at different scales. If we downscale the image, its objects can be detected with a lower receptive field; if we upscale the image, a larger receptive field is needed. Upscaling helps to deal with small objects.

Cropping cuts a patch of fixed size from an image, usually the default training size. Cropping is a way to sample many distinct images from a single image, breaking it into multiple patches as in (WANG, P. et al., 2018) or just randomly cropping the image on the fly. In addition, cropping can alleviate the burden of processing high-resolution images, as seen in (CORDTS et al., 2016; METZGER; MORTIMER; WUENSCHE, 2021). Another cropping function is correcting classes imbalance and mitigating the spatial correlation of dense patches (LONG; SHELHAMER; DARRELL, 2015).

Many state-of-the-art works join both techniques, as known as random resized cropping. A usual procedure is to cut from an image a random area between 8 and 100% with an aspect ratio between (3/4, 4/3), then resize the patch to comply with the training input size. The values mentioned are the default ones from the PyTorch library, which are based on ImageNet experiments (SZEGEDY et al., 2015).

2.3.2.3 Color

Lighting biases are a challenge on many image problems (SHORTEN; KHOSHGOFTAAR, 2019). Performing equally well on bright or dark images is critical. Usually, it is expected that the neural network to work with a certain degree of color-invariance on the objects we want to recognize and segment in the scene. Color transformation comes to this rescue.

Examples of transformations are color jitter and grayscale. Color jitter manipulates brightness, contrast, saturation, and hue. At the same time, grayscale transformation forces the network to pay attention only to structural and contrast information.

2.3.2.4 Synthetic

Synthetic transformation creates unnatural images that would not come from real-world data distributions. Two examples are filling images with blanks and pasting a part of one image on another. The second example is known as Cutmix, see Figure 25.

Cutmix is a very strong augmentation, i.e., it highly disturbs the original data. Cutmix randomly crops a patch from one image and to pastes it on another in a random position. The labels must follow the same procedure. The authors stand that cutmix improves the models against input corruptions and out-of-distribution samples (YUN et al., 2019).

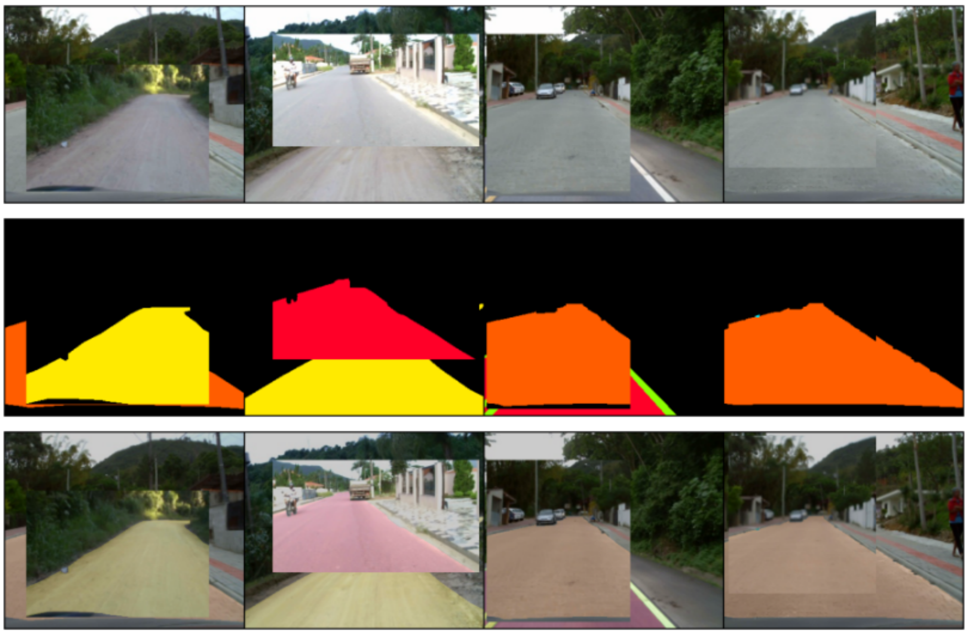


Figure 25 – Cutmix augmentation on the RTK dataset.

2.3.3 Metrics And Losses Functions

Choosing the proper loss function is fundamental for a good experiment result. The loss functions used for semantic segmentation tasks belong to four categories: Distribution-based, Region-based, Boundary-based, and Compounded (JADON, 2020)

2.3.3.1 Cross Entropy

Cross-entropy (CE) is a distribution-based loss and is widely used in the machine learning community for general tasks like detection, classification, and segmentation (MURPHY, 2012; LECUN; BENGIO; HINTON, G., 2015).

Cross-entropy, first proposed in the information theory field, measures the average amount of information that one distribution probability needs to transmit the information from another distribution; it is one way to calculate how close two distributions are; the lower the CE value is, the closer the distributions are. In the machine learning context, it is commonly used in supervised learning to measure the similarity between the ground truth distribution $y \sim P$ (y follow the probability distribution P) and the prediction distribution $\hat{y} \sim Q$. CE is given by $H(y, \hat{y})$.

For example, cross-entropy is defined below for the random variables labels $y \sim P_y$ and predictions $\hat{y} \sim Q_{\hat{y}}$. These random variables map pixels from the sample space Ω to a real number $\mathbb{R}$ that represents one of K possible pixel classes, such that, $y : \Omega \to \{1, ..., K\}$.

$$H(y, \hat{y}) = -\mathbb{E}_{k \sim P_y} \log Q_{\hat{y}}(k) \tag{2}$$

$$= -\sum_{k=1}^{K} P_y(k) \log Q_{\hat{y}}(k) \tag{3}$$

When CE is used as a loss function, the loss is a summation over all m training samples, see eq. (4).

$$Loss_{CE}(y, \hat{y}) = -\sum_{i=1}^{m} \sum_{k=1}^{K} P^{(i)}(k) \log Q^{(i)}(k) \tag{4}$$

For binary segmentation that separates the foreground from the background, we use Binary Cross Entropy (BCE), so the eq. (3) can be simplified to:

$$BCE(y, \hat{y}) = -y \log \hat{y} - (1 - y) \log (1 - \hat{y}) \tag{5}$$

Besides, we can set weight to classes on CE; this is known as Weighted Cross Entropy (WCE) (RONNEBERGER; FISCHER; BROX, 2015). Each class sample i is balanced by the weight α_i, which is usually inversely proportional to the class frequency.

$$WCE(y, \hat{y}) = -\sum_{k=1}^{K} \alpha_k P_y(k) \log Q_{\hat{y}}(k) \tag{6}$$

2.3.3.2 Dice

Dice is a region overlap-based metric introduced in (DICE, 1945), and it is known by many names like dice's coefficient, sørensen–dice index, and F1-score. It is a similarity measurement between two sets, e.g., X and Y, as shown below:

$$Dice = \frac{2|X \cap Y|}{|X| + |Y|} \tag{7}$$

Where $|X|$ and $|Y|$ are the cardinalities of the two sets, i.e., the number of elements. When dice is used for evaluation, the metric is first computed per class and then averaged. This metric is typically known as *mF1*.

Dice loss helps to handle imbalanced datasets. (SUDRE et al., 2017) noticed that when the level of imbalance increases, loss functions based on region-overlap are more robust than WCE.

However, as dice works on the discrete space, it is necessary to adopt surrogate functions that are differentiable regarding the neurons' activations as they are continuous values. (MILLETARI; NAVAB; AHMADI, 2016) proposed the surrogate function bellow:

$$Loss_{DICE} = 1 - \frac{2\sum_i^N y_i \hat{y}_i}{\sum_i^N (y_i^2 + \hat{y}_i^2)} \tag{8}$$

In eq. (8), the summations i to N regards all pixels of an image or a mini-batch of images.

(MILLETARI; NAVAB; AHMADI, 2016) affirms that the dice loss also called *soft dice*, avoids the need to assign class weights to balance the number of class samples as they would be more robust to class-imbalance than WCE. Nevertheless, region overlap-based losses often do not converge as well as cross-entropy (BERMAN; TRIKI; BLASCHKO, 2018).

2.3.3.3 Intersection over Union

Intersection-over-Union (IoU) is a region overlap-based metric, also known as the Jaccard index. It is defined as:

$$IoU = \frac{|X \cap Y|}{|X| \cup |Y|} \tag{9}$$

Suppose a naive algorithm predicts every pixel of an image as background; IoU would penalize it because the intersection area between the predictions and the ground truth would be zero, whereas cross-entropy and accuracy could still present good values.

IoU is commonly averaged across the classes for multiclass datasets, yielding the mean Intersection-over-Union (mIoU) (BERMAN; TRIKI; BLASCHKO, 2018).

As the dice loss (Sec. 2.3.3.2), IoU offers an alternative to handle class imbalance (NAGENDAR et al., 2018; EVERINGHAM et al., 2015). By the way, both metrics are highly correlated, but, IoU tends to penalize more bad mistakes than dice (HANG KWOK, n.d.). In addition, mIoU is also inherently non-differentiable. The works (NAGENDAR et al., 2018; BERMAN; TRIKI; BLASCHKO, 2018; RAHMAN; WANG, Y., 2016) propose surrogate losses functions. The equation (10) presents the (RAHMAN; WANG, Y., 2016) solution:

$$Loss_{IoU} = 1 - \frac{\sum_i^N y_i \hat{y}_i}{\sum_i^N (y_i^2 + \hat{y}_i^2 - y_i * \hat{y}_i)} \tag{10}$$

3 RELATED WORKS

This section concerns what challenges would be found in emerging country road datasets, especially the ones observed in the RTK dataset (RATEKE; JUSTEN; VON WANGENHEIM, 2019). We started with a short description of the RKT dataset, followed by a description of flaws and pitfalls found in the baseline training, i.e., the original dataset authors' training described in (RATEKE; VON WANGENHEIM, 2021; RATEKE, 2020)

3.1 ROAD TRAVERSING KNOWLEDGE DATASET

Road Transverse Knowledge (RTK) is an image dataset collected in Águas Mornas and Santo Amaro cities, a countryside region in the Santa Catarina state in Brazil. All images were captured during daytime in good weather conditions; the images present variations in the scene illuminations, and there is a presence of shadows. RTK is a dataset rich in characteristics representing many conditions on emerging countries' roads. As cited on the authors' review of 2021 (RATEKE; VON WANGENHEIM, 2021) and to the best of our knowledge, that is the unique road dataset to handle many road variations at once, e.g., damages, fixes, artifacts, three road surfaces, and surfaces transitions.

In total, the dataset consists of 77547 frames with a resolution of 352x288; only 701 out of them are pixel-wise labeled, i.e., available labels for semantic segmentation. The authors assign a validation set for the semantic segmentation task with 140 images; the training set has the 561 remaining ones. See samples in Figure 26.

The dataset consists of 12 classes. It covers three road surfaces: asphalt, paved, and unpaved; four road signs: markings, cat's eyes, and speed's bumps; four damages or fixes on the streets: patches, water puddles, potholes, cracks; and one road artifact, storm-drains. The classes are highly unbalanced. The background's pixels correspond to 65.9% of the total number of pixels. Each road surface corresponds to between 9 and 13% of the total. The other classes are way sparse, each corresponding to a value less than 0.78%, e.g., cats-eye and storm-drain classes have only 0.02% of all pixels.

3.1.1 Baseline

The dataset authors work (RATEKE; VON WANGENHEIM, 2021) presents a solution that we set as the **baseline** for our work. The solution consists of a U-Net architecture with an encoder ResNet34 pre-trained on ImageNet; they also adopted the Adam optimizer with a learning rate of 1e-4, a 1-cycle learning policy, and a batch size of 8. For data augmentation, they applied random perspective distortion and horizontal flipping, both with an occurrence probability of 50%. The neural network was trained

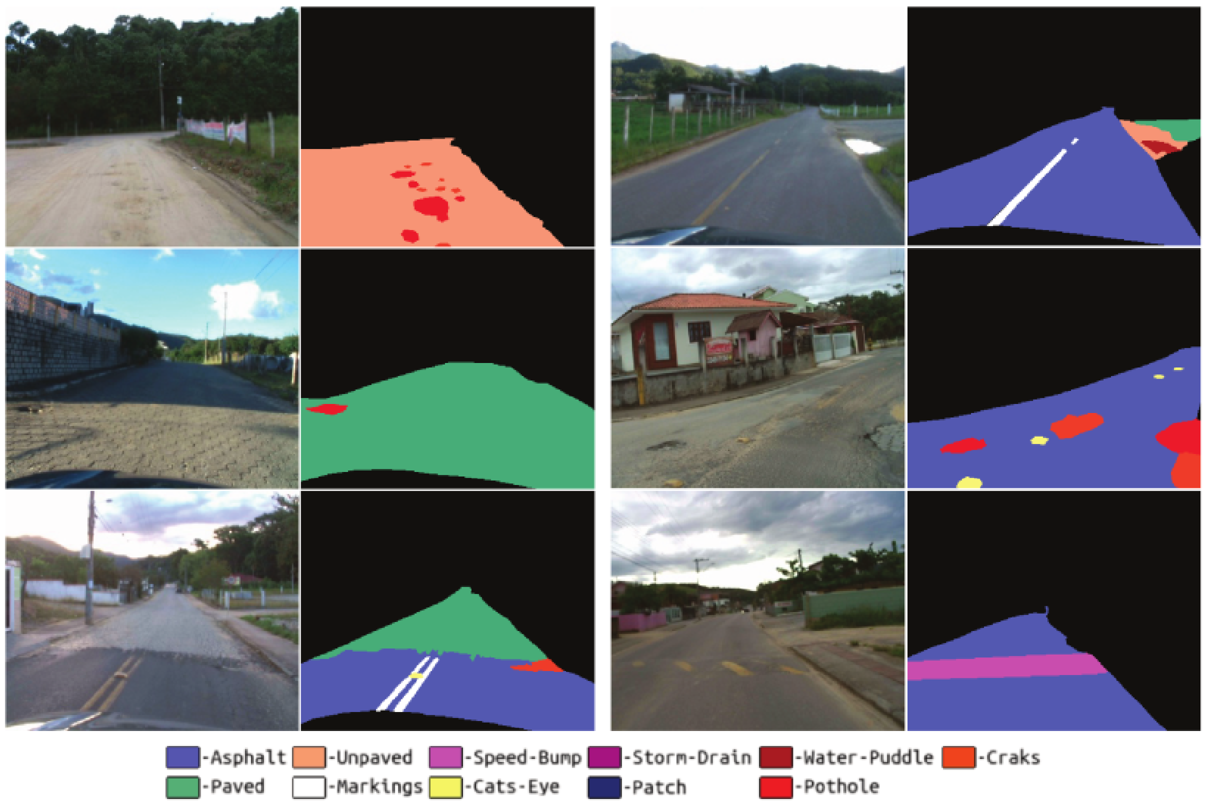


Figure 26 – Examples from the RTK dataset. From (RATEKE; VON WANGENHEIM, 2021).

on a two-stage regime. First, the authors trained the model with Cross-entropy (CE) for 100 epochs; afterward, they trained another 100 epochs with Weighted Cross Entropy (WCE). The setup details were annotated from the authors' medium post (RATEKE, 2020) and the *fastai*[1] library documentation.

The authors reported the system accuracy performance of 97%; when they broke the accuracy information by classes, the worst performance was 72% from *cracks*.

3.1.2 Criticism

This section raises flaws and pitfalls found in the baseline work during our research; these issues became areas for improvements for the RTK benchmark during our experiments.

3.1.2.1 Accuracy

Accuracy is not a proper metric for an imbalanced multiclass task, as seen in section 2.3.3; besides, accuracy is only a valid metric when used for overall system evaluation, not by class. In addition, the standard metric for semantic evaluation is the mean Intersection-over-Union (mIoU) (Sec. 2.3.3.3); it is the common choice on the

[1] https://docs.fast.ai/

deep learning community. There are multiple examples of dataset benchmarks using mIoU, for example, road datasets (CORDTS et al., 2016; ALHAIJA et al., 2018), scene datasets (EVERINGHAM et al., 2015; ZHOU, B. et al., 2017), and medical datasets (JHA et al., 2020; BERNAL; SÁNCHEZ; VILARINO, 2012).

Furthermore, the score presented in the medium post from a fastai builtin function - which is not explicit what it is -does not compute accuracy or mIoU; it does compute the *intersection of pixels given a true label* as seen in the fastai source code [2]. The fastai metric is known by many names as **recall**, sensitivity, or true positive rate. Although recall is a relevant statistic metric, it only measures the presence of true positives predicted by the model, not regarding false positives. A metric regarding both presences is essential because false positives tend to increase as true positives increase.

3.1.2.2 Classes Weights

The class weights should leverage the underrepresented classes such that all classes count equally on the loss computation. Class weights are typically set as the ratio between the number of pixels of the majority class and the respective classes, as shown in eq. (11). Besides, this ratio should count only the pixels that belonged to the training set.

$$w_{c_i} = \max_{c \in C} \frac{n(c)}{n(c_i)} \tag{11}$$

where C is the set of the classes, and $n(c_i)$ means number of pixels under a given class c_i.

WCE is used in the baseline in the second-stage training with the class weights of 1, 5, 6, 7, 75, 1000, 3100, 3300, 270, 2200, 1000, 180, respectively, to the class id. The weights computed in this work show that the correct values should be 1, 5, 6, 7, 87, 1736, 3590, 3793, 292, 2410, 1084, 219.

3.1.2.3 Number of Epochs

One epoch means to pass forward and backward once the entire training set through the network. In the case of the RTK dataset, it means to realize the passes throughout all the 561 training samples, which needs 71 iterations when using batch size 8. Hence, they trained two stages of 100 epochs that added up to 14200 iterations; this was not enough, as we have shown later in the section 4.3.1. Moreover, when a stronger data augmentation is applied during training, it requires a higher number of iterations for the training to converge.

[2] https://github.com/fastai/fastai1/blob/master/fastai/vision/interpret.py#L61

3.1.3 Challenges

Some characteristics of the RTK dataset that makes it specifically challenging are low-resolution images, small objects, multiscale objects, and imbalanced classes.

3.1.3.1 Small objects

The dataset images have a resolution of 352x288, which was intentionally done to represent the contexts of autonomous vehicles with low computational power that cannot handle high-quality images. These images are much lower than the ones found in other datasets from the field, e.g., Cityscapes, CamVid, and TAS500 have a resolution of 2048x1024, 960×720, and 2026x620, respectively. The difference between RTK resolution in the other datasets used for developing state-of-the-art models raises the need to adapt the model architecture for a low-resolution context.

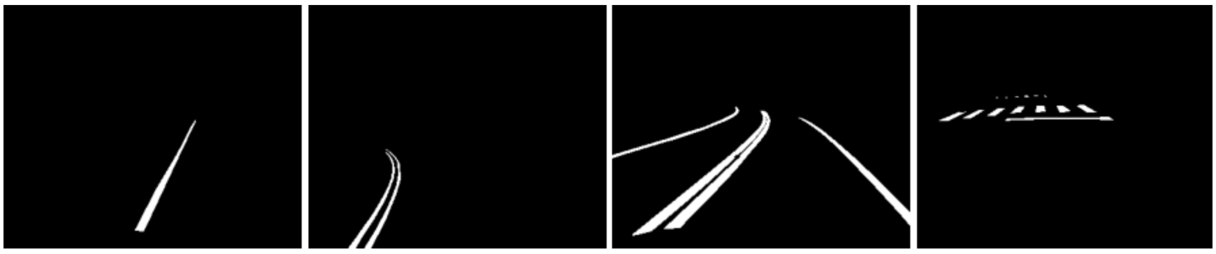
One consequence of low-resolution images is small or even tiny objects, given the low number of pixels in an image. For example, *road markings* (id = 4) have 20% of its objects with the shortest edge less than or equal to 3 pixels. *Cat's eyes* (id = 6) have more than 70% of the shortest edge object less than or equal to 5 pixels, this same condition also appears in 30% of *storm-drain* (id = 7) and *patches* (id = 12) objects. Furthermore, there are extreme cases when the shortest edge has **1 pixel**, happening on 15% of the *road markings* (id = 4) objects, and 2 pixels happening on 10% and 4% of *cat's eyes* and *cracks* objects (id 6 and 12, respectively). Figure 27 extracts exclusively a single class from the RTK images, showing only their shape in a binary image, i.e., a mask.

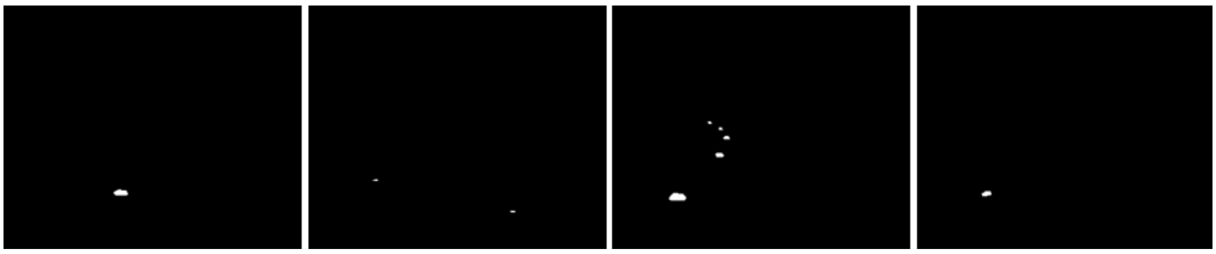
Very short edges can be bypassed by the first convolutional layer stride of a ResNet or in the following max-pooling layer. This situation can be concerning when trying to extract features for these objects either because the neural network simply did not compute a feature for that pixel location or it did not could build a high level of abstraction from that low-level features; besides, it can be significantly hard to recover these objects shape from a coarse feature map (a feature map of reduced dimensions).

3.1.3.2 Multiscale objects

Another challenge on the RTK dataset is the scale and geometry disparities between their many classes, e.g., objects from *patches*, *potholes*, *water puddles* do not have a clear shape pattern and size, whereas *road surfaces* have a broad well-defined shape and *road markings* are usually long and thin; see Figure 28.

In addition, it is also observed a high variance of scales inside the same class; for example, *cat's eyes* on a spaced series had bigger objects when they were close to the camera and smaller objects when they were far from the camera; it was measured that in a same spaced series the *cat-eyes* varied around 3 to 6 in the size of their object.


(a) Lane markings


(b) Cat's eyes

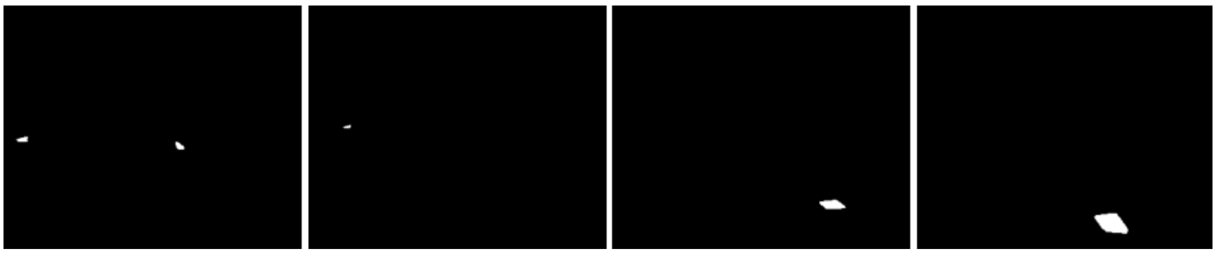

(c) Storm drain

Figure 27 – Examples of object masks with a short edge.

3.1.3.3 Imbalanced dataset

RTK is a highly imbalanced dataset that reaches a number of pixels ratio between the underrepresented and the overrepresented classes of 1 to 1000 up to 1 to 3000. The underrepresented classes can be overlooked during training; besides, many underrepresented classes are only present in a small subset of the images.

With respect to the entire dataset, including the background as a class, most of the ground truth (35.8%) has a single class, the road surface. Two classes happen 29.5% of the time, and three classes 17.1%. Images with 4 and 5 classes represent 8.7 and 8.3%, respectively. The maximum number of classes in a single image is 6 and occurs in 0.6% of the cases. See Figure 29.

3.2 SMALL OBJECTS SEGMENTATION

As one of the low-resolution datasets challenges is the segmentation of small objects, we point out some solutions found in the literature in this section. The works (HU; RAMANAN, 2017; HAMAGUCHI et al., 2018) have emphasized that context matters for detecting small objects. They have underlined the fact that even humans cannot recognize a small building in a satellite imagery patch without the context of roads, cars,

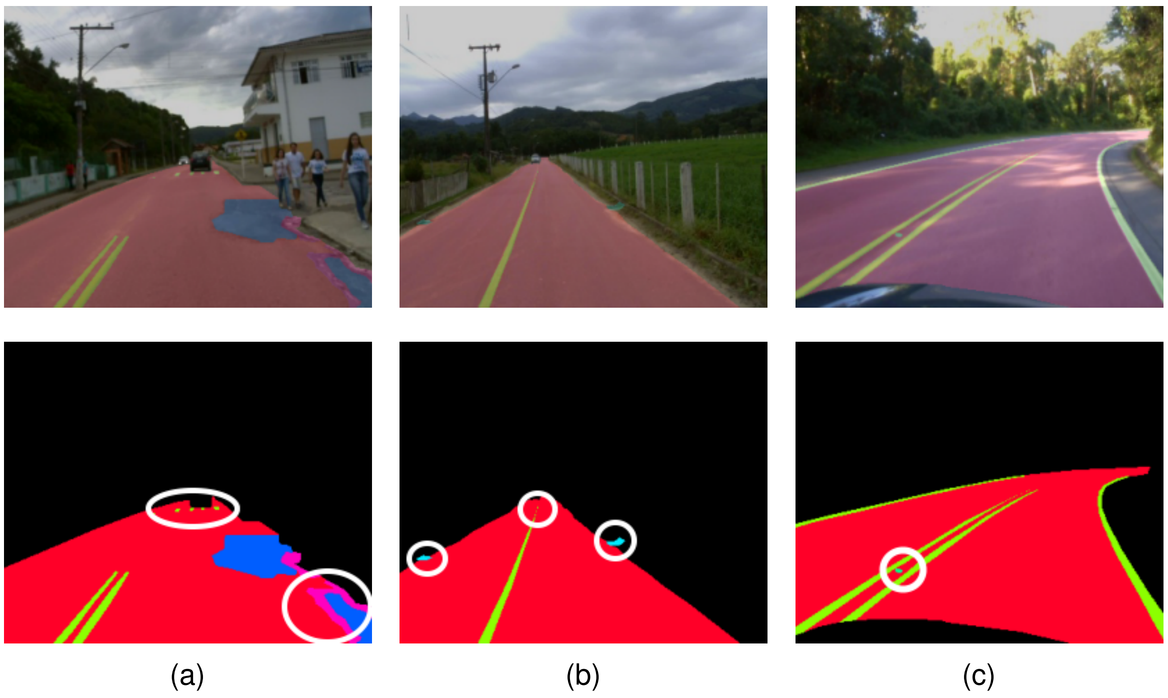


Figure 28 – Multiscale examples on the RTK dataset. The white circles highlight in a) lane markings with less than or equal to 2 pixels per edge at the end of the field of view and two nested classes (water puddles and cracks) of irregular shapes; in b) three different geometries and scales between classes: a wide and long road, an irregular and small storm drain at the border, and a road marking that changes its width along the road; and in c) a tiny cat's eye object between the road markings.

or other buildings. So they highlight the importance of images with a large field of view.

(SHEN et al., 2017) noted that the double sequence of stride 2 on ResNet loses feature information that impacts more small object predictions as their initial dimension is reduced by stride 4 at the beginning of the forward pass. It is especially concerning for us as many small objects in the RTK dataset have edges inferior to 5 pixels that could rapidly vanish at the first ResNet block. Their solution was to replace the first 7x7 convolutional layer of stride 2 with a series of three 3x3 convolutional layers, and stride 2 happens only at the first layer. Moreover, they replaced the 3x3 max-pooling with a 2x2 max pooling. The works (ZHOU, Peng et al., 2018; LI, Y. et al., 2019) also improved their performance by adopting this modification.

The (ZHOU, Peng et al., 2018) worked on object detection and pointed out that shallow feature maps are helpful to detect small objects as they have small receptive fields and that, on the other hand, deep feature maps are helpful for large objects because of the large receptive fields. Nevertheless, shallow features have the drawback of containing less semantic information, impairing the prediction quality of small objects against large objects. Thus, they proposed a scale-transfer module consisting of pooling and scale-transfer layers that integrate low and high-level features within a CNN. The pooling layer condenses feature map information, and the scale-transfer

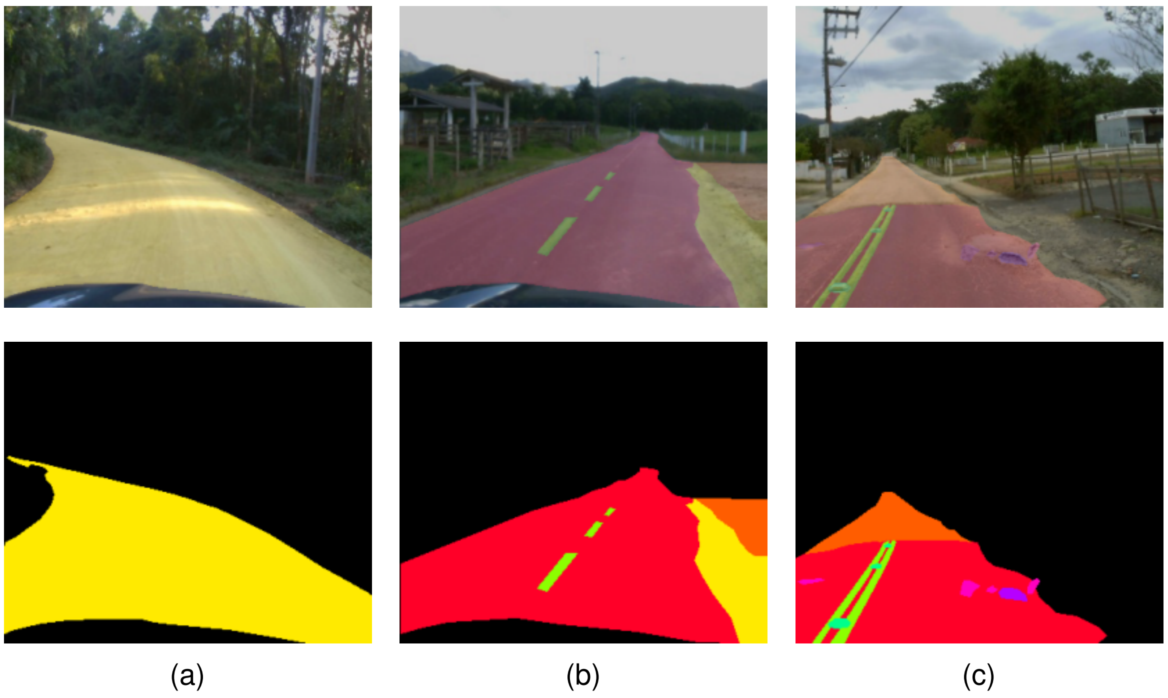


Figure 29 – Examples of RTK images regarding the number of classes. a) Single class label, only the unpaved road class; b) Label with four classes, the three road surfaces, asphalt (red), unpaved (yellow), paved (orange); and the lane markings (yellow-greenish); c) Label with six classes, it contains a transition of road surfaces (asphalt to paved), lane markings, cat's eyes (between the road markings), water-puddles (purple) and potholes (pink).

layer expands the feature map dimensions by decompressing the number of channels; e.g., it transforms a 1x1x16 tensor (width x height x channels) to a 4x4x1.

One work that significantly improved small object results in the Cityscapes (poles, traffic lights, and signs) was (WANG, P. et al., 2018). They projected the Dense Upsampling Convolution (DUC), which also decompresses the feature maps channels into a higher spatial resolution, just like the work cited above; see Figure 30. The authors said that DUC is able to recover detailed information missed by bilinear interpolations. Moreover, they also proposed a Hybrid Dilated Convolution (HDC) framework that would alleviate the *gridding* effect, which happens when convolution layers are applied in series with the same dilated rate. The HDC uses a **variation of dilation rates** to break the constant series; for example, the third ResNet50 block uses a series of kernel size 3 of length 6, if the dilation rate applied on this block is 2, it yields a series of kernel sizes of [6, 6, 6, 6, 6, 6] filled with holes (see. Sec. 2.2.3.5); now, if HDC applies a mix of dilation rates in a sequence [1, 2, 3, 3, 2, 1], it would yield kernel sizes of [3, 6, 9, 9, 6, 3], alleviating the gridding effect. The variation of dilation rates was tried later in the performance increment strategy (Sec. 4.3.11).

Regarding small object predictions, remote sensing faces even more complex difficulties given the crowd of objects. (HAMAGUCHI et al., 2018) found that when using

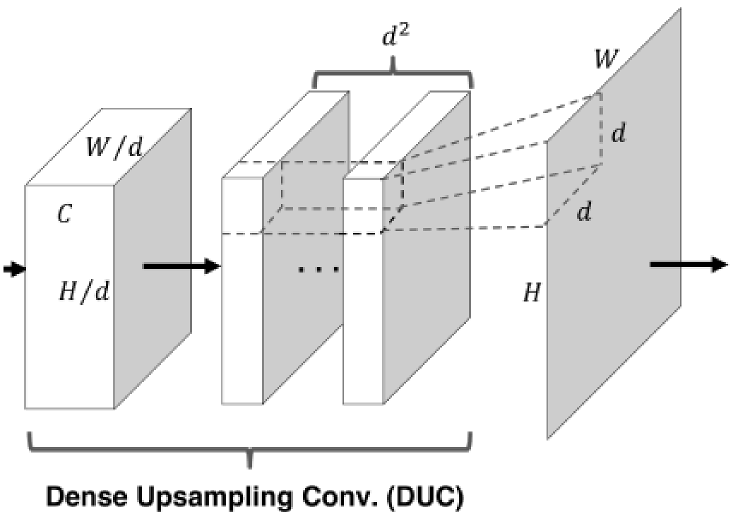


Figure 30 – Dense Upsampling Convolution (DUC). From (WANG, P. et al., 2018).

a progressive increase of dilation rates, the neighboring neurons stop interacting given the kernel sparsity (the holes between the weights), which causes two problems: spatial consistency between adjacent neurons becomes weak, and local structure cannot be extracted by deep layers; the latter specifically affects small objects as their local structure may not be recognized by the neighbor neurons; nevertheless, it does not affect large objects. Hence, they proposed a Local Feature Extraction (LFE) module that consists of dilated convolutional layers with a **decreasing dilation rates**, i.e., the dilation rate is reduced as the feature map dimension is reduced. For example, we would use degressive rates of [8, 4, 2] in opposition to the usual progressive rates of [2, 4, 8] from DeepLabV3+. The decreasing rates would recover consistency between neighbor neurons. This tactic was tried later in the performance increment strategy (Sec. 4.3.11).

3.2.1 Removing max-pooling layer

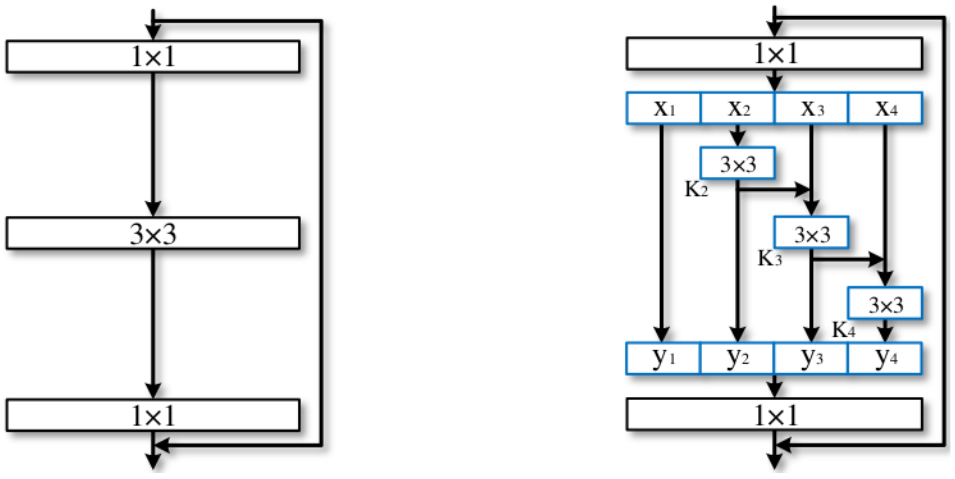
Removing the max-pooling layer is a substantial strategy to preserve small object information, as found out later in this work (Sec. 4.3.8), as this work. To the best of our knowledge, this is the first work to remove the first max-pooling layer of the ResNet architecture aiming to better segment small objects.

In the literature, very few times, the removal of the max-pooling layer was proposed in order to save small object details. When removing the pooling layer, most works look to reduce the network complexity, such that the pooling layer stride is replaced by a neighbor convolutional layer stride as seen in the works (DEVARAJ et al., 2021; XIA; ZHUGE; LI, H., 2018). Another purpose found was to alleviate the gridding effect (YU, F.; KOLTUN; FUNKHOUSER, 2017).

The few works found concerning max-pooling layer removal to preserve fine details are (CHRISTIANSEN et al., 2016; KENDRICK et al., 2018; AL-ANTARI et al., 2018). The anomaly detector (CHRISTIANSEN et al., 2016) increased the feature map resolution as they found out that a very low reduced map impaired anomalies detection in the agriculture plantations; they applied this modification over a Caffe reference model. The mobile neural network for facial landmarks (KENDRICK et al., 2018) emphasized that the data resolution should be preserved as much as possible because of the ease of losing key facial features, such as the eye corners and nose tip, when employing max-pooling layers; they measure the impact of the max-pooling layer over their custom architecture. The X-ray mammograms system tested in (AL-ANTARI et al., 2018) noticed that removing the max-pooling layer and keeping the high-resolution feature maps were especially important for the objects' edges.

3.3 MULTISCALE SEGMENTATION

This section raises the solutions found in the literature on another challenge of the RTK dataset, multiscale segmentation. (GAO et al., 2019) built an adapted version of the residual block from the ResNet architecture, the Res2Net module. Unlike most methods, Res2Net works at layer levels and extracts multiscale features at different resolutions inside a single bottleneck block, see Figure 31. A bottleneck block is a series of convolutional layers that intercalate one layer of kernel size 3 between two layers of kernel size 1.



(a) Bottleneck block. (b) Res2Net module.

Figure 31 – Comparison between a bottleneck block and a Res2Net module. From (GAO et al., 2019).

Another approach was the addition of *attention mechanism* to the different scales of feature maps. It combines the best predictions smartly from each scale and avoids scale pitfalls (TAO; SAPRA; CATANZARO, 2020; CHEN et al., 2016). As noted in (TAO; SAPRA; CATANZARO, 2020), fine details, such as edges of objects or thin structures, are often predicted better with large scales; on the other hand, large structure predictions require global context and are often predicted better with small scales.

One way to combine multiple scales was proposed in DeepLab, which attempted the multiscale problem using the ASAP module (Sec. 2.3.1.3), concatenating the feature map from multiple dilated convolutional layers. U-Net3+ architecture (HUANG et al., 2020) also combined multiscale information adapting U-Net to use skip connections from all scales; each decoder block received information of all feature maps of higher resolutions from the encoder blocks, and all features maps of lower resolution from the decoder blocks. See (c) in Figure 32.

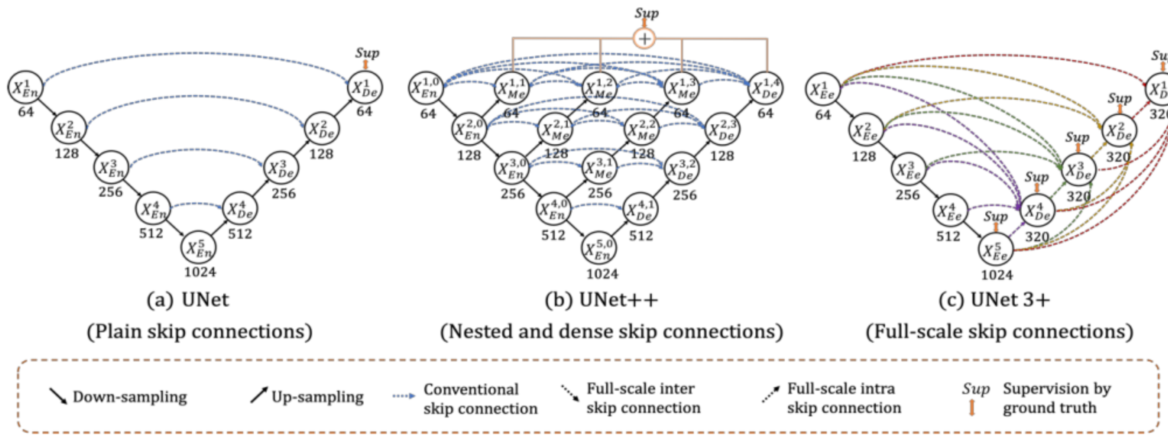

(a) UNet
(Plain skip connections)
(b) UNet++
(Nested and dense skip connections)
(c) UNet 3+
(Full-scale skip connections)
Down-sampling
Up-sampling
Conventional skip connection
Full-scale inter skip connection
Full-scale intra skip connection
Sup Supervision by ground truth

Figure 32 – The usage of skip-connections in U-Net to combine multiscale information. a) U-Net, b) U-Net++, and c) U-Net3+. From (HUANG et al., 2020).

3.4 TRAINING AND INFERENCE PIPELINE

This section points out the practices from state-of-the-art works missed by the RTK baseline and the alternatives to improve the RTK benchmark; many were tried on the performance increment strategy.

Some state-of-the-art works (CHEN et al., 2018; ZHAO et al., 2017; YUAN et al., 2019; KIRILLOV et al., 2020; YU, C. et al., 2021) used cropping, resizing, and horizontal flipping as data augmentation; the optimizer chosen was usually SGD instead of Adam; besides, SGD optimization follows a polynomial learning rate "poly" strategy with momentum of 0.9 and weight decay of 5e-4. The preference for SGD is explained by recent works showing that SGD reaches better generalization results (KESKAR; SOCHER, 2017; ZHOU, Pan et al., 2020). For the number of iterations, the neural networks usually train for more than 100k iterations in a single-stage regime. Many of these guidelines are also exposed in (HE, T. et al., 2019).

On the inference side, the ensemble of multiple predictions from the same image augmented in multiple scales and flipping direction is a common practice in competitions for an extra boost of the results. The ensemble of predictions is combined by averaging or max-pooling as seen in (TAO; SAPRA; CATANZARO, 2020; ZHAO et al., 2017; CHEN et al., 2016).

4 PERFORMANCE INCREMENT STRATEGY ON RTK

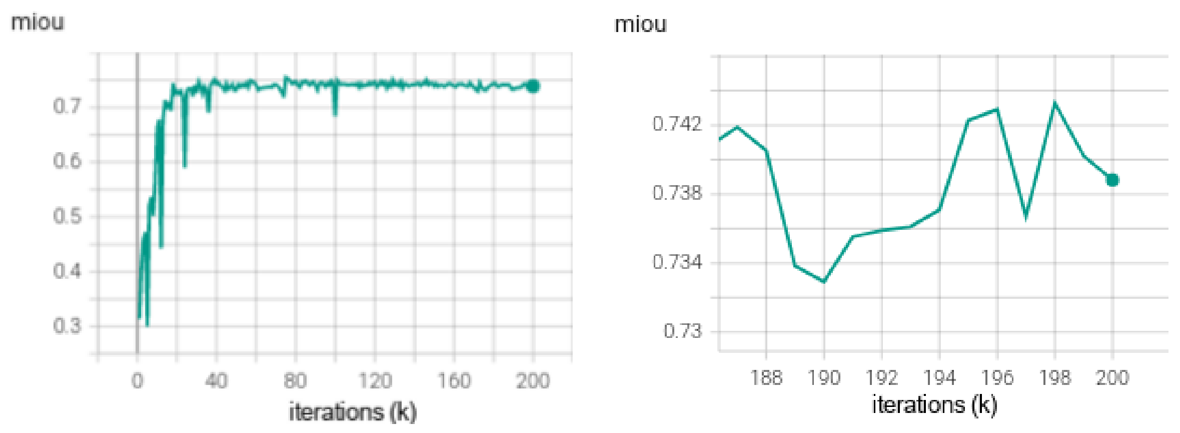
In this section, we go through the application of the performance boost strategy on the RTK dataset. It contains 15 ablation experiments; each experiment section has a table showing the hypotheses' mIoU results and a brief statement of the experiment cues or findings; furthermore, the sections may have further visual analysis if it was deemed necessary for patterns or pitfalls illustrations.

4.1 DEVELOPMENT ENVIRONMENT

All experiments ran on the Google Colab platform alternating between the Pro and Pro+ subscriptions that usually provide one of the GPUs: T4, P100, or V100; with 16 GB RAM. Besides, the code was developed with Python 3.7, and the deep learning library adopted was PyTorch of version 1.12.1. PyTorch was chosen given its high adoption in academia; consequently, this is the usual choice of state-of-the-art algorithms.

4.2 METHODOLOGY

Throughout the performance strategy, we kept tracking the training and validation sets evaluations of the hypotheses after each 1k iterations, which is regarded as a step. It was noticed that, after the convergence, there were small performance oscillations, as seen in Figure 33. In the figure 33 the performance alternates between 0.733 and 0.743, which represents 1% of the mIoU scale [0, 1]; this variation was enough to lead to an incorrect impression when comparing the hypotheses. Our workaround was averaging the validation evaluation from the last ten steps.

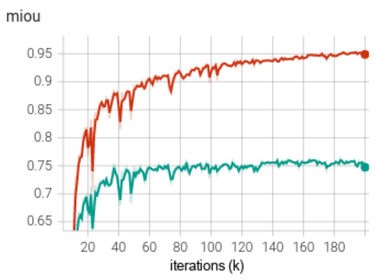


(a) Complete training (b) Last steps

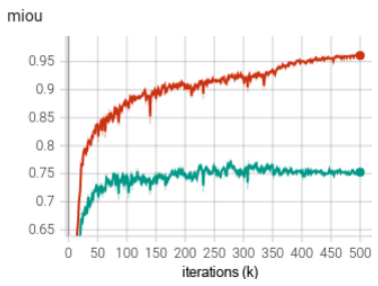
Figure 33 – Oscillation on the validation evaluation.

The default number of iterations was set to 200k for most of the hypotheses; it was far enough to reach the best performances on the validation set. However, when the hypotheses used a weak or no augmentation, or a shallow network, 100k iterations proved to be sufficient; it specifically occurred only in the hypotheses *None*, *Cutmix50*, *Cutmix80* from the *Data Augmentation* experiment (Sec. 4.3.3).

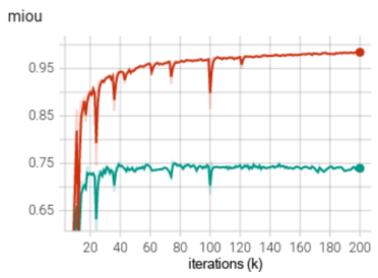
It is well known that a very long training may overfit the training data and, consequently, degrades generalization. Nevertheless, overfitting was not perceived in these experiments, see Figure 34. Likely because of the fewer quantity of parameters of the convolutional neural networks in comparison to densely connected networks, which occurs given the weight-sharing and sparse interaction properties.



(a) Cropping augmentation



(b) Cropping & Color augmentation on 500 steps



(c) GeomRTK augmentation

Figure 34 – The red and green lines show the training and the validation set evaluations, respectively. Overfitting badly affect the results on the validation set.

4.3 ABLATION EXPERIMENTS

During the experiments, all model configurations and training hyperparameters are kept from the best hypothesis of the previous ablation experiment; otherwise, the configuration is explicitly expressed. The ablation experiments only test one factor at a time.

4.3.1 Baseline: Iterations

- Do more training iterations help?

First off, the baseline work, i.e., the author's RTK benchmark, missed the mIoU evaluation (Sec. 3.1.2.1), so we needed to reproduce the code presented in (RATEKE, 2020) to establish a comparison value, which yielded 0.547 mIoU on the validation set. Moreover, we had noticed that 100 epochs, which add up to 14k iterations, were insufficient (Sec. 3.1.2.3); thus, we extended the training to 50k and 100k iterations each stage.

One disclaimer, these experiments implemented all the setup details from the baseline medium post (RATEKE, 2020) but the 1-cycle learning policy. Then, the results may slightly diverge from the original baseline; however, after later investigations, we checked the results with and without the 1-cycle learning policy are pretty close or even worse, as shown in Table 1.

No. Iterations	1-cycle policy	mIoU
200k		0.739
200k	✓	0.709
100k		0.693
14k		0.547

Table 1 – Baseline number of iterations.

Table 1 showed that the 200k iterations hypothesis substantially improved up to 0.739, a huge leap of 0.192 from the original baseline. Adjusting the number of iterations was crucial to compare the baseline setup fairly against the other setups. From now on, we call the longer trained baseline as **baseline200k**.

4.3.2 Technique: Single Stage Training

- Does single-stage work better?

For the sake of simplicity, we tried the single-stage regime for training, which is a usual setup (Sec. 3.4). The WCE loss used the corrected classes weights (Sec. 3.1.2.2).

Loss	mIoU
CE	0.739
baseline200k	0.739
WCE	0.732

Table 2 – Single stage training.

Single-stage training got equivalent results to two-stages, see Table 2; showing no need to add the complexity of dealing with two training steps. From now on, the hypotheses will follow with single-stage CE loss.

4.3.3 Technique: Data Augmentation

- Which augmentation operation works better?

This ablation tested different approaches for augmentations. The original work used horizontal flipping and geometry warping, which we named *geomRTK*, which we compared to cropping, resizing, color jitter, and cutmix. Moreover, we tried no augmentation just to measure the augmentation impact on the performance.

The cropping augmentation adopted a patch size of (224, 224), and resizing augmentation scaled the image edge randomly to a factor between (0.78, 2). Color augmentation used the ColorJitter function from PyTorch with jitter values of 0.27 to brightness, contrast, and saturation; and 0.07 to hue; furthermore, color augmentation involved a grayscale function that removes the color information. Each color function had a probability occurrence of 0.2.

Augmentation	mIoU
Cropping	0.754
Cropping & Color	0.751
GeomRTK	0.739
Resizing & Cropping	0.735
Cutmix50[1]	0.710
Cutmix80[1]	0.710
None	0.668

Table 3 – Augmentation operations.

Table 3 shows that *Crop* is the best result. The addition of resizing or color jitter worsened the results, especially resizing. Cutmix alone made surprisingly little impact, as it is regarded as a strong augmentation. No augmentation hypothesis had a huge decline in performance, showing us how augmentation was vital on this benchmark.

4.3.4 Architecture: Encoder Depths

- Which backbone depth works better?

The baseline backbone is ResNet34, so we tried deeper versions of ResNet. The experiment hypotheses from the table below were built on top of the best results so far: single-stage CE loss and cropping as augmentation.

Depth	mIoU
ResNet-101	0.765
ResNet-50	0.759
ResNet34	0.754

Table 4 – Encoder depth.

[1] The superscript number next to cutmix is the occurrence probability.

Table 4 clearly points a trend; deeper the backbone, better the performance. So we follow some next experiments with the depth of 101 layers.

4.3.5 Architecture: Resnet Variants

- Which variant works better?

We checked Resnet variants. Res2Net works at different scales within the ResNet module, ResNeSt adds an attention mechanism, and ResNeXt works with cardinalities, i.e., grouped convolutions. We evaluated them on the depth of 101 layers.

Variants	mIoU
ResNet-101	0.765
ResNeXt-101-32x8d	0.759
ResNeSt-101	0.754
Res2Net-101-26w4s	0.746

Table 5 – Resnet Variants.

On Table 5, any ResNet-101 variant did not lift the performance.

4.3.6 Architecture: Networks

- Which network works better?

The U-Net architecture was used on the baseline work. In this experiment, we tried DeepLabV3+ too.

Architecture	Encoder	mIoU
U-Net	ResNet-101	0.765
U-Net	ResNet-50	0.759
DeepLabV3+	ResNet-50	0.756
DeepLabV3+	ResNet-101	0.756

Table 6 – Architectures types.

U-Net outperformed DeepLabV3+ in two backbones, ResNet50 and ResNet101. However, we insisted on DeepLabV3+ but with further modifications. As we will see in the subsequent experiments, these modifications play an essential role in the DeepLabV3+ architecture that will outperform the U-Net's results. We kept ResNet-50 as DeepLabV3+ backbone as the depths 50 and 101 got the same results; we chose the simpler option which is the shallower one.

4.3.7 Architecture: Output Stride

- Which output stride works better

Although the last experiment had U-Net as the best hypothesis, on this and the next four ablations (until Hybrid Local Feature Extractor), we proposed a series of modifications to the DeepLabV3+ network. The first modification was the Output Stride (OS) (Sec. 2.2.3.4), which defines the ratio between the input dimension and the high-level feature map at the end of the encoder stage.

OS	mIoU
4	0.763
8	0.757
16	0.756

Table 7 – Output stride.

Table 7 presents a clear trend that diminishing the output stride increases the performance.

In these hypotheses, we captured a problem when using a higher output stride, especially 16, in which the model randomly predicts small background blobs over the road and small road blobs over the background. When OS is 8, there are random blobs of other road kinds over the main road. Although, The problem seems to gradually vanish until the output stride decreases to 4, see Figure 35. While this problem significantly impacts visually, it is barely noticed quantitatively, given the slight differences between the hypotheses' performance.

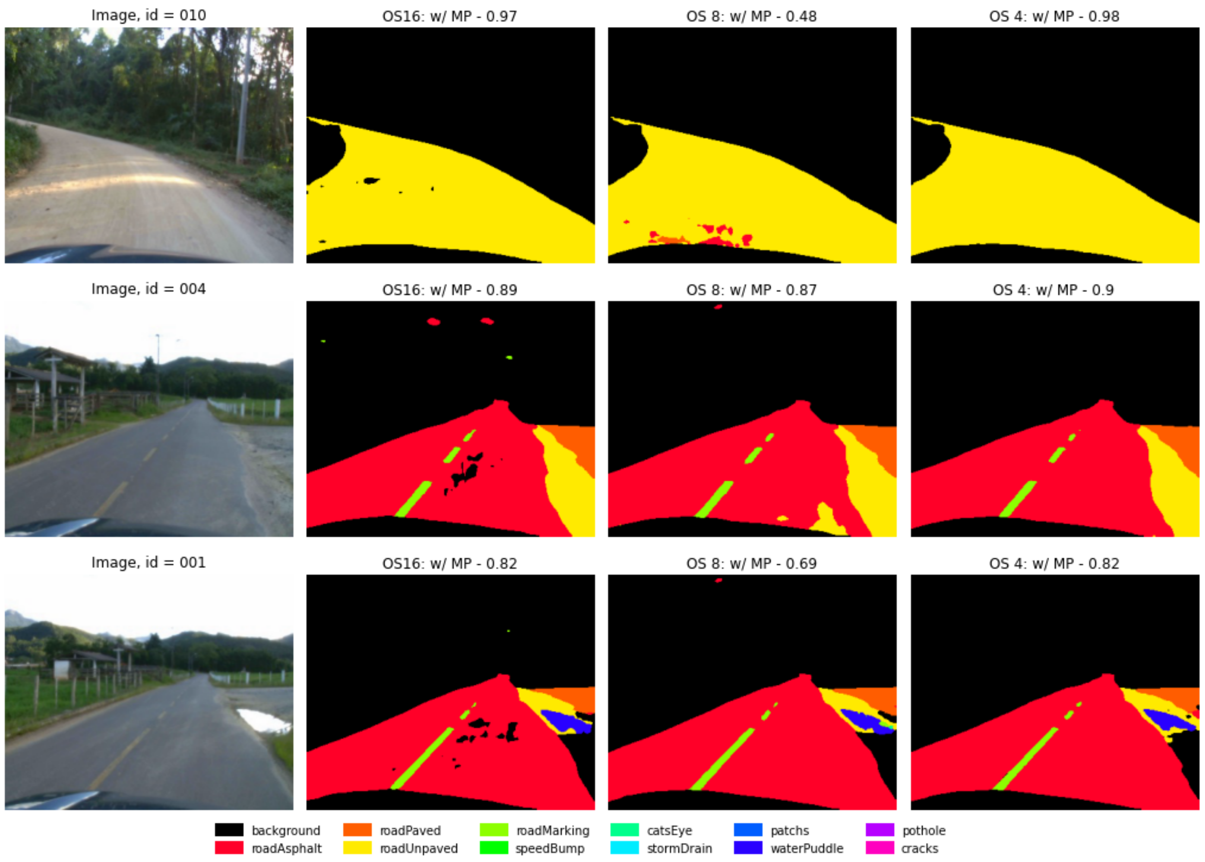


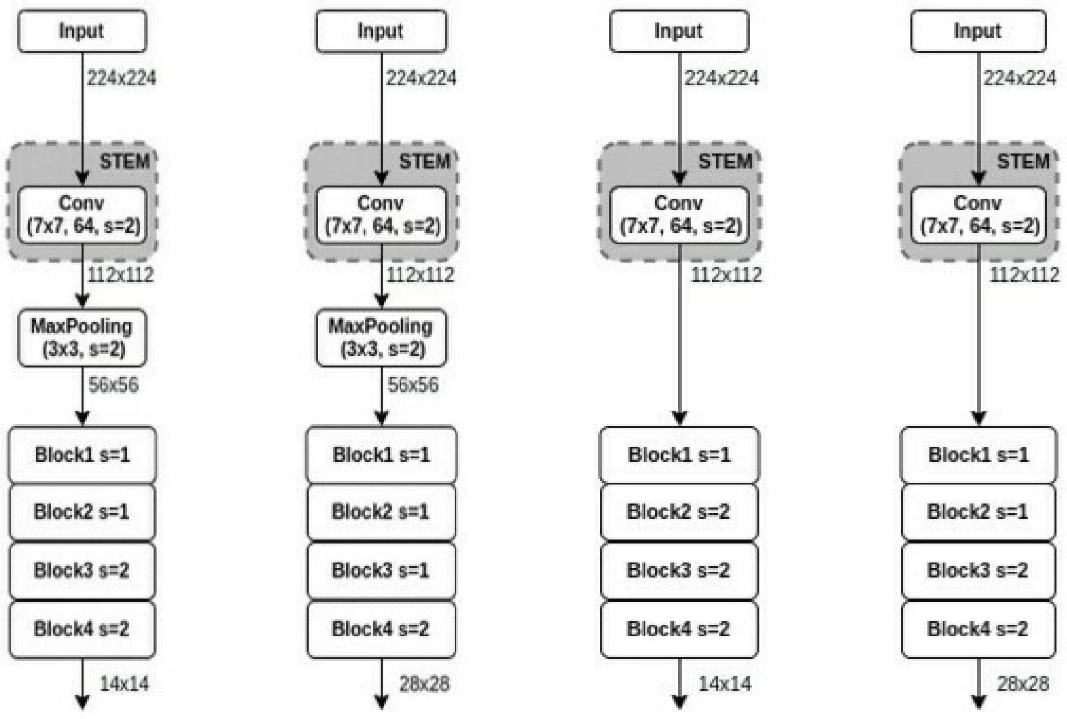
Figure 35 – Predictions with models trained at different output strides.

The random blob prediction was not perceived in the U-Net models.

4.3.8 Architecture: Max-Pooling Removal

- Does removing the max-pooling layer help?

The Figure 36 illustrates the blocks of DeepLabV3+ with and without (wo/) the max-pooling (MP) layer, showing how each block stride controls the feature map's dimension and, as a result, the final output stride. Removing the max-pooling layer at the beginning of the first block avoids early spatial information loss, which can be crucial for extracting small object features.



(a) OS=16; w/ MP (b) OS=8; w/ MP (c) OS=16; wo/ MP (d) OS=8; wo/ MP

Figure 36 – ResNet backbone. "s" within the block is the block's stride; just below it prints the feature map spatial resolution.

An important note, the DeepLabV3+ decoder concatenates high-level and low-level features, see Figure 23. The output stride referred in section 4.3.7 relates to the high-level features. Meanwhile, the resolution of the low-level features refers to the feature map just after the max-pooling layer; if the MP layer is absent, the low-level features are the ones just after the stem. The low-level output stride is 4 by default, whereas it is 2 when removing max-pooling.

Table 8 confirms a new best benchmark. Avoiding the MP layer positively impacted OS 8 and 16, though it did not matter for OS 4. It seems that a smaller low-level output stride is more crucial than the high-level output stride for DeepLabV3+; so when we concatenate a lower low-level output stride with a higher high-level output stride, it joins the best of both contexts, which would explain the Table 8 results.

Architecture	Encoder	OS	wo/ MP	mIoU
DeepLabV3+	ResNet-50	16	✓	0.769
DeepLabV3+	ResNet-50	8	✓	0.768
U-Net	ResNet-101	-		0.765
DeepLabV3+	ResNet-50	4		0.763
DeepLabV3+	ResNet-50	4	✓	0.762
DeepLabV3+	ResNet-50	8		0.757
DeepLabV3+	ResNet-50	16		0.756
U-Net	ResNet-101	-	✓	0.754

Table 8 – Max-pooling removal.

The max-pooling removal also solved the problem of random blobs predictions for OS 8 and 16 mentioned on the *Output Stride* experiment (Sec. 4.3.7), see Figures 37 and 38. By the way, the hypothesis *OS:16 - wo/ MP* presents the sharpest predictions.

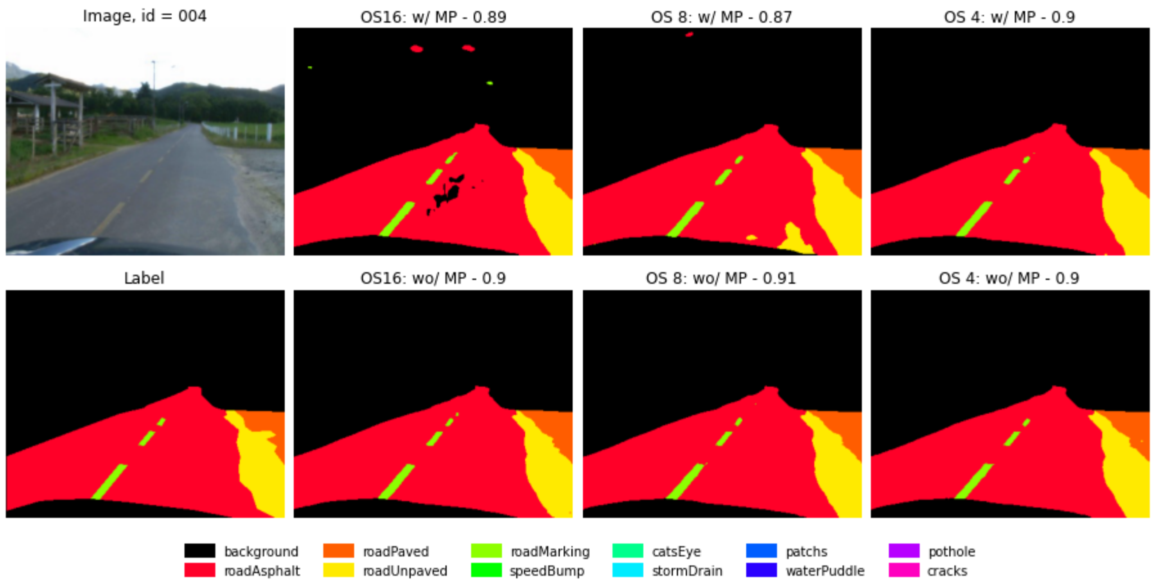


Figure 37 – Without max-pooling layer at multiple output stride.

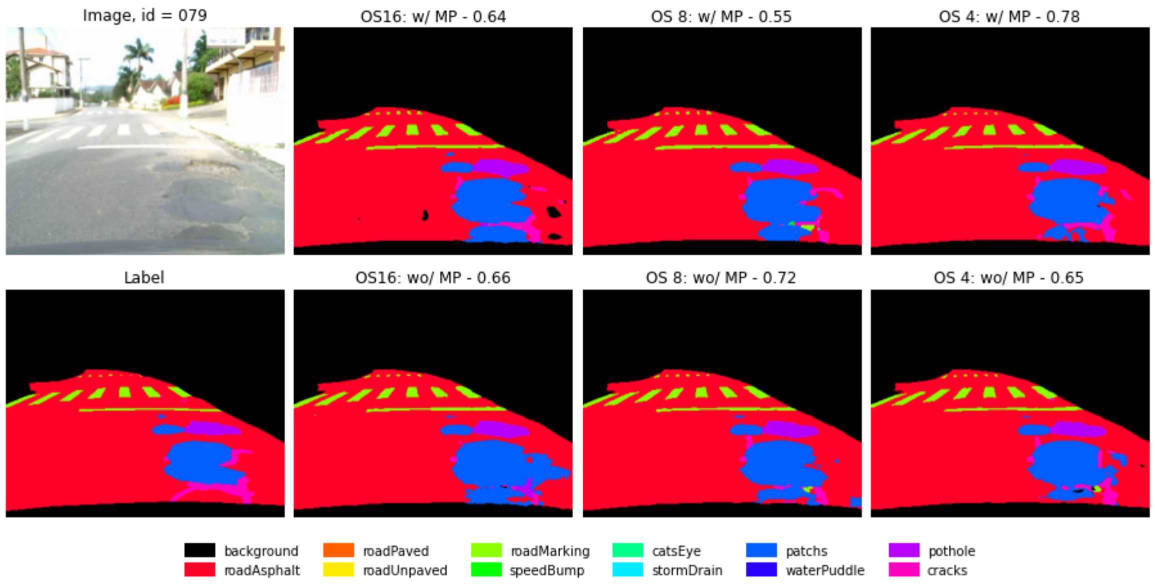


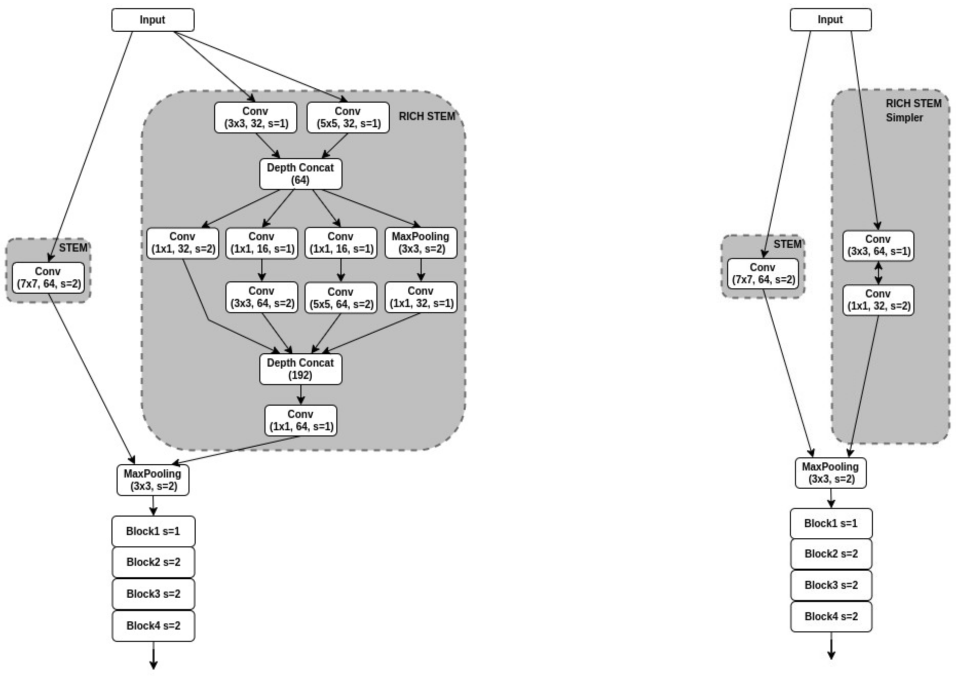
Figure 38 – Without max-pooling layer at multiple output stride.

4.3.9 Architecture: Rich Stem

- Does a richer ResNet stem help?

As ResNet stem has a single convolutional 7x7 layer of stride 2, we think the big kernel size and stride 2 would be terrible for extracting tiny features from the small objects. Therefore, we proposed a rich stem with a higher number of layers with small kernel sizes and stride 1; hence, it would enrich the feature maps from the stem block that receives the original input resolution.

We created a rich stem version that consists of a parallel path inspired by the inception module (SZEGEDY et al., 2015), involving multiple parallel convolutional layers with distinct kernel sizes. Moreover, we created a dummy rich version with a single parallel path with only one convolutional layer 3x3 followed by a point-wise convolution. The dummy rich version would be easier to train and to validate the idea of small kernel size and stride would help; see Figure 39,



(a) Rich stem (b) Simpler version

Figure 39 – The default conv 7x7 layer with pre-trained weights are at the left path and the new proposed modules at the right path in a) rich stem and b) dummy rich stem.

Parallel paths allow keeping the pre-trained weights of the original ResNet stem. The performance downgraded considerably when the experiment ran only with the rich module. The pre-trained weights seem crucial. The experiments setup were: DeepLabV3+, ResNet-50, output stride 16, without max-pooling, and cropping as augmentation.

Stem	mIoU
Default	0.769
Dummy rich	0.768
Rich	0.763

Table 9 – Rich Stem.

As shown in Table 9, the stem modifications did not present any enhancement. Maybe because of the lack of pre-trained weights.

4.3.10 Architecture: Transposed convolutions

- Does upsampling with learnable parameters help?

Transposed convolutions (convT) replace the upsampling operation. The DeepLabV3+ decoder has two upsampling blocks, see Figure 23. This change turned the upsampling operation parametrized, i.e., it has learnable weights now. We kept the *without max-pooling* setup.

ConvT	mIoU
	0.769
✓	0.760

Table 10 – Upsampling by Transposed Convolution.

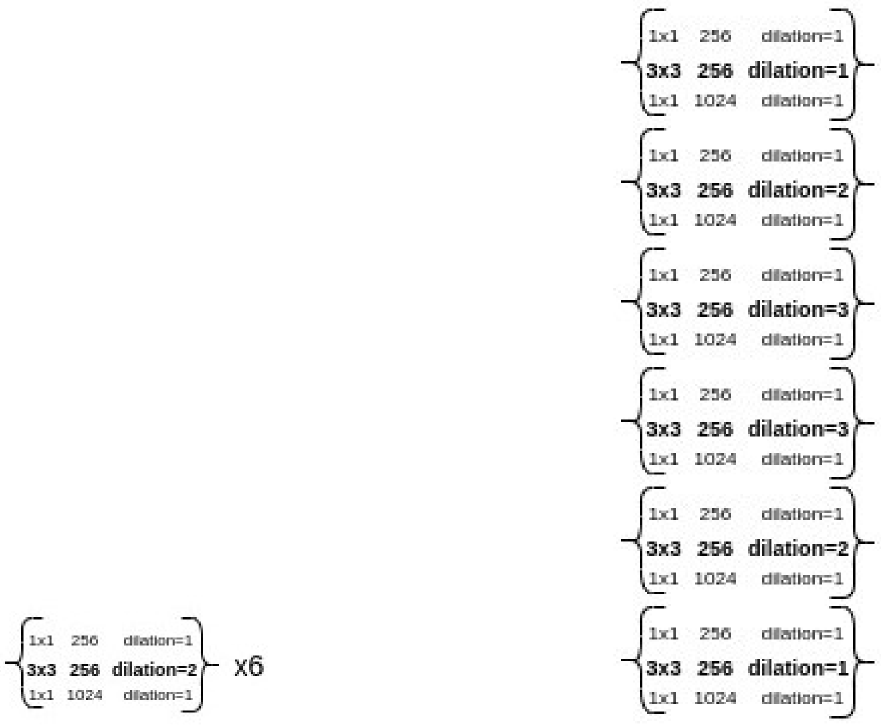
This modification did not show any enhancements, see Table 10.

4.3.11 Architecture: Hybrid Local Feature Extractor

- Do fancy approaches like degressive and hybrid dilation rates help?

Local Feature Extraction (LFE) (HAMAGUCHI et al., 2018) ensures local structure extraction that is usual lost when dilation rates are progressively enlarged. Inspired by LFE and Hybrid Dilated Convolutions (WANG, P. et al., 2018) (See Sec. 3.2), we adapted it to the ResNet architecture and called it Hybrid Local Feature Extraction (HLFE). While (HAMAGUCHI et al., 2018) created a custom architecture to implement LFE, we implemented it straight in blocks 3 and 4 of the ResNet using "hybrid" values.

The hybrid values chosen were a sequence of [1, 3, 5, 5, 3, 1] for block 3 and [1, 3, 1] for block 4, instead of the constant sequences [3, 3, 3, 3, 3, 3] for block 3 and [3, 3, 3] for block 4; see Figure 40.



(a) ResNet Block3. (b) Block 3 with a series of hybrid dilation rates.

Figure 40 – Series of hybrid dilation rates.

One caveat is that HLFE only applies to output stride of 8 or 4 when max-pooling is not used because dilated convolutional layers are not used for output stride 16.

OS	HLFE	mIoU
16		0.769
8		0.768
8	✓	0.767
4	✓	0.766
4		0.762

Table 11 – Hybrid Local Feature Extractor.

As seen in Table 11, HLFE did not bring any enhancement to the final results.

4.3.12 Technique: Cutmix

- Does cutmix along with another augmentation help?

Cutmix is a strong augmentation (Sec. 2.3.2.4). It corrupts a lot of the original image and changes its natural distribution. We left this augmentation last because the strong image corruption could affect the comparison between the hypotheses of the previous experiments. This experiment ran over DeepLabV3+, ResNet-50, OS 16, and without MP.

Aug	mIoU
Cropping & Cutmix80[1]	0.782
Cropping & Cutmix50[1]	0.771
Cropping	0.769

Table 12 – Cutmix addition.

Table 12 points a new best benchmark. The addition of cutmix as augmentation got a significant leap on mIoU, it raised from 0.769 to 0.782.

4.3.13 Technique: Loss Functions

- Which loss function works better?

We tried surrogate losses functions from dice and mIoU metrics, which are commonly as evaluation metrics (Sec. 2.3.3.2 and 2.3.3.3). This experiment setup ran over the cutmix addition setup.

Loss	mIoU
CE	0.782
CE + Dice	0.774
CE + mIoU	0.770
Dice	0.756
mIoU	0.751
WCE	0.689

Table 13 – Surrogate loss functions.

Table 13 shows the surrogate losses did not boost the performance of an already very calibrated model. Actually, their standalone usage even degraded the performances. We double-checked these loss functions impact running it on the baseline200k setup; see Table 14.

Loss	mIoU
CE + mIoU	0.748
CE + Dice	0.747
CE	0.739
WCE	0.732
Dice	0.720
mIoU	0.698

Table 14 – Surrogate loss functions on initial setup.

Table 14 shows an improvement when CE works along with mIoU or dice. On the other hand, the standalone surrogate losses considerably degraded the performance again! Although the mIoU measures the metrics, the optimizing function that excelled on intersection-over-union is cross-entropy; CE acts as a proxy for mIoU.

4.3.14 Technique: Optimizers

- Which optimizer works better?

Stochastic Gradient Descent (SGD) optimizer is known to have better generalization capability (KESKAR; SOCHER, 2017). We applied SGD with a learning rate warm-up policy of 5 steps which increases the rate gradually until its initial value of 1e-2; afterward, the learning rate decreases by a polynomial factor of $1 - \left(\frac{iter}{iter_{max}}\right)^{0.9}$ until the end of the training. A momentum of 0.9 is also applied.

Optimizer	mIoU
Adam	0.782
SGD	0.762

Table 15 – Optimizers.

In Table 15, Adam outperformed SGD by a meaningful difference, showing that SGD is not always the best option for generalization. By the way, there is no free lunch (WOLPERT, 1996).

4.3.15 Prediction: Voting Ensemble

- Does voting ensemble help?

The last strategy to boost performance was voting an ensemble of multiple predictions from many versions of the same image. Standard procedures are to flip and to multiscale (MS) the image, up and downscale.

RTK samples have a resolution of (352, 288); we adopted two more resolutions of (288, 224) and (352, 448) for multiscale prediction. All scaled images have an extra horizontally flipped version. Thus, the use of three resolutions sums up six versions in total. After getting the six predictions with the probabilities maps for all classes, the maps are resized to the original size (288, 224) using nearest neighbor interpolation. Then the maps are averaged to yield the final class probability map. Finally, the classes with the highest probability for each pixel are assigned as the prediction, creating the final prediction map.

In contrast with the methodology applied so far, this experiment counts with only the evaluation metrics referred to the saved model from the last training step, i.e., at

200k iterations. Thus, the default benchmark presented in Table 16 - representing the best benchmark so far, from the *cutmix* experiment -, is different from the reported value above. The same model is used to generate the prediction of all image versions.

Prediction	mIoU
Flipped	0.789
MS + Flipped	0.784
Default	0.774

Table 16 – Predictions Voting Ensemble.

Flipped and "multiscale + flipped" voting ensemble increased the performance as shown in Table 16. In addition to the saved model from the last step, we also got the results using the hypothesis model saved at the step of the best evaluation on the validation set, which does not necessarily take place at the last steps of the training. The results are in the Table 17.

Prediction	mIoU
Flipped	0.798
MS + Flipped	0.793
Default	0.793

Table 17 – Predictions Voting Ensemble w/ Best Checkpoint.

Once more, flipped ensemble got the best results, whereas the multiscale case did not always seem to help the final prediction. In the end, the most excellent performance of all hypotheses - following the original methodology -was 0.789 mIoU. Furthermore, regarding the case of the model saved at the best evaluation metric, the mIoU achieves 0.798.

4.4 INSPECTING PERFORMANCE BY CLASSES AND GROUPS

In this section, we offer a quantitative and qualitative analysis of the best hypothesis from the performance increment strategy not regarding the prediction voting ensemble, which was the best hypothesis from the *cutmix* experiment with the setup of DeepLabV3+, ResNet-50, output stride 16, without max-pooling, Adam optimizer, and cropping along with cutmix for data augmentation. We used the model saved at step 200.

Class	mIoU	Edge[1]	Inv Frequency[2]
background	0.99	215	1
roadAsphalt	0.94	132	5
roadPaved	0.96	152	6
roadUnpaved	0.95	134	7
roadMarking	0.82	15	87
speedBump	0.82	28	1736
catsEye	0.55	4	3590
stormDrain	0.68	6	3793
patchs	0.83	17	292
waterPuddle	0.48	27	2410
pothole	0.77	12	1084
cracks	0.49	10	219

Table 18 – Intersection-over-union by class.

The Table 18 shows the biggest class, *background*, had the greatest performance of 0.99. The other big object classes, the road surfaces, had great results also. The other classes with small objects, located between road marking and cracks in the table, had a performance gap compared to big object classes. Cracks, water puddles, and cat's eyes have the worst results by far, which are classes that involve tiny objects or irregular-shaped formats.

(CORDTS et al., 2016) pointed out that the mIoU metric is biased toward big objects; our results confirms this bias, see Figure 41. In addition, we also have a clear trend between the mIoU metric and normalized inverse frequency, which is directly related to the number of pixels, see Figure 42.

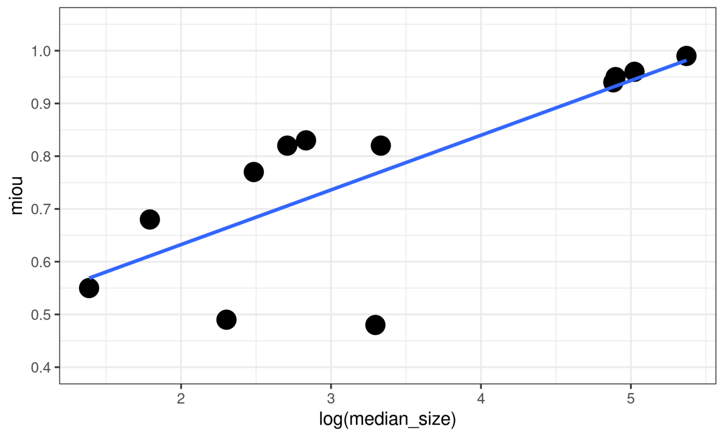


Figure 41 – Relation between IoU and median edge size of the class.

[1] the median size of the shortest edge of the class object.

[2] the normalized inverse frequency is the ratio of the number of pixels on the training set between the most popular class and the *i* class.

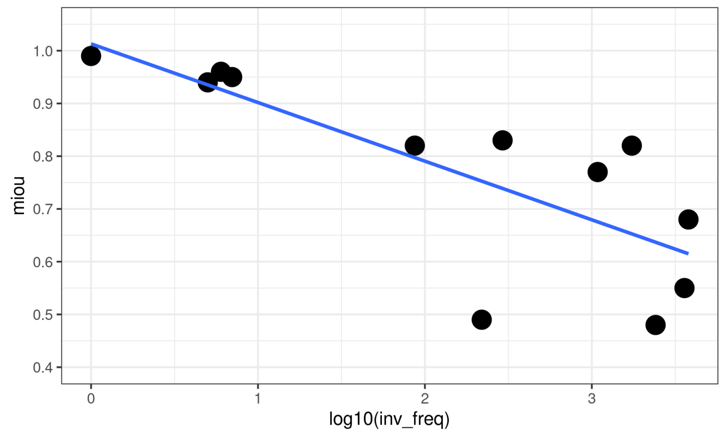


Figure 42 – Relation between IoU and normalized inverse frequency of the class.

For the subsequent analysis, we looked to evaluate the model between the different challenges present in the dataset, like small-sized objects, multiscale objects, and imbalanced dataset (Sec. 3.1.3). Hence, we grouped the classes into the categories of big objects, small objects, small edges, and underrepresented classes. The groups provide an outlook of how the models handle these challenges. The background is excluded from the analysis. The big objects are the road surfaces, and the small objects are all the others. Small edges are road markings, cat's eyes, and patches. The underrepresented classes are the ones with the fewest number of pixels for training, which are cat's eyes, storm drain, and water puddle.

Table 19 presents the evaluation by group for each hypothesis from the loss functions experiment (Sec. 4.3.13). We observed that the values between the hypotheses are close in each group, but *WCE*. By the way, we did not notice a trade-off between classes' performances, i.e., being better in one group at the expense of worse performance in another; it seemed that the better model is, the better it works for all groups.

Loss function	Big	Small	Small Edges	Underrepresented
CE	0.950	0.680	0.570	0.620
Dice	0.937	0.660	0.533	0.620
CE + Dice	0.947	0.688	0.573	0.630
mIoU	0.947	0.655	0.510	0.613
CE + mIoU	0.947	0.676	0.553	0.610
WCE	0.937	0.562	0.433	0.570

Table 19 – mIoU by classes groups.

We repeated the same evaluation checking the modifications applied on DeepLabV3+; see Table 20. Again there was not a clear advantage of any approach in any group. These hypotheses' setups were built over DeepLabV3+, Resnet50, OS 8, CE, and cropping augmentation.

Loss functions	Big	Small	Small Edges	Underrepresented
w/ MP	0.920	0.651	0.530	0.593
wo/ MP	0.953	0.661	0.530	0.610
wo/ MP + HLFE	0.957	0.666	0.550	0.613
wo/ MP + convT	0.957	0.665	0.543	0.613

Table 20 – mIoU by groups.

4.5 INSPECTING SMALL OBJECTS PREDICTION

This section shows the visual impact of different losses for small object predictions. The hypotheses used here are from the first table of the loss functions experiment (Sec. 4.3.13).

Figure 43 shows a scene of many small objects. The *CE + mIoU* hypothesis had the sharpest blobs, whereas *WCE* had a higher presence of false positives like bigger cat's eyes (green-light) and cracks (pink blobs). All hypotheses predicted an extra presence of cracks at the middle of the road, underlined at circle 3 in Figure 44.

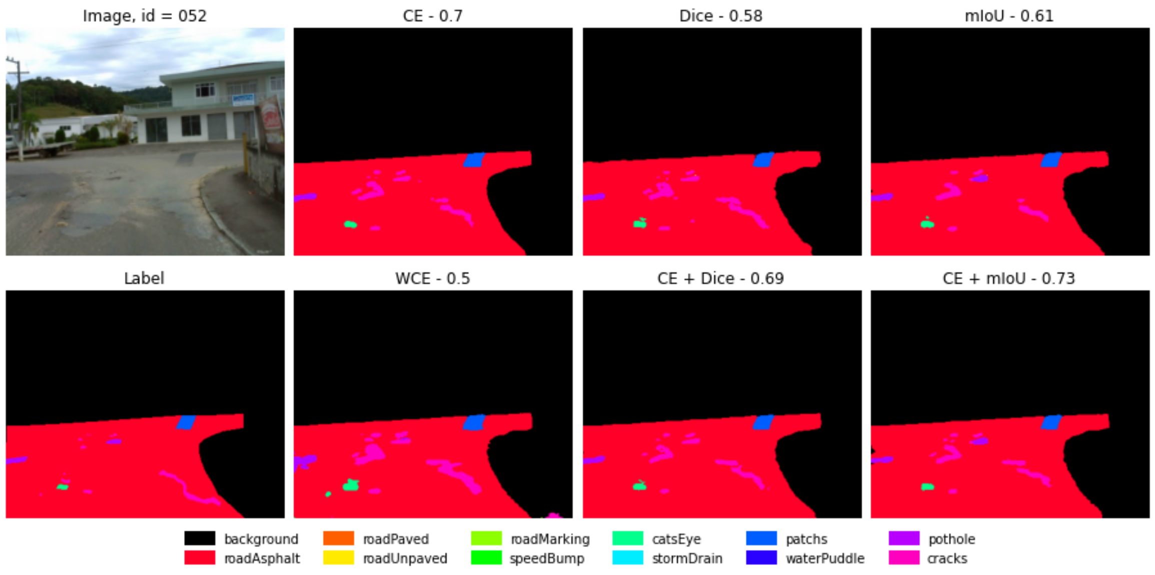


Figure 43 – Inspection of small object predictions. The titles follow the loss hypothesis and the mIoU result.

One caveat, the subfigures' titles assign the loss hypothesis and the image mIoU computed value. It is essential to warn that mIoU is not a reliable metric when evaluating a single image because if any class outside of the ground truth ones is predicted, its IoU results in zero, which pushes down the mean IoU (mIoU) value and explains why there are good visual predictions with low mIoU in the figures below. The mIoU is a stable metric when it works at least for a mini-batch of images.

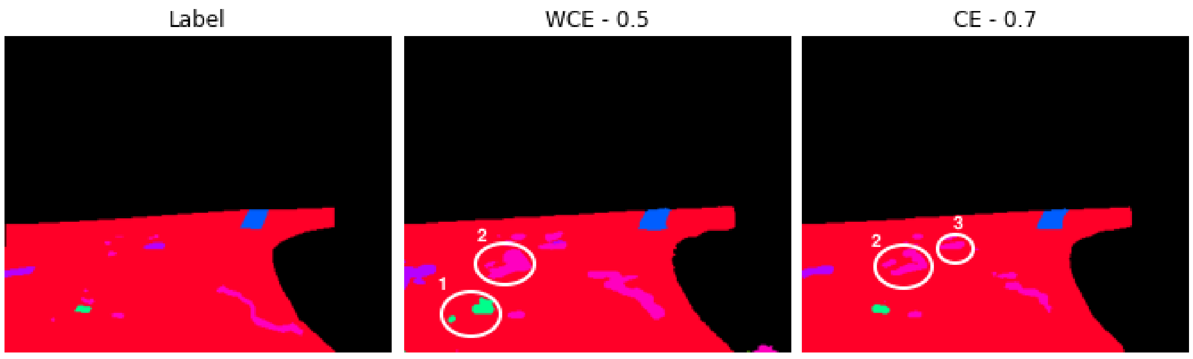


Figure 44 – Notes from Figure 43. Point 1 underlines the exceeding predictions of underrepresented classes on *WCE* loss; point 2 shows a repeated error of predicting an unreal patch from the hypotheses; point 3 indicates the wrong pothole prediction of *CE*.

In the Figure 45, *CE* got the sharpest results, almost in total agreement with the ground truth. Most of the hypotheses struggled to predict the straight white line in front of the camera; however, they did well to predict the farther small lanes of small at the end of the field of view. A peculiar event occurred in the image, underlined by a red circle 1 in Figure 46, the car hood reflected the pedestrian lane; the *Dice* even predicted the pedestrian lane as part of the navigable path.

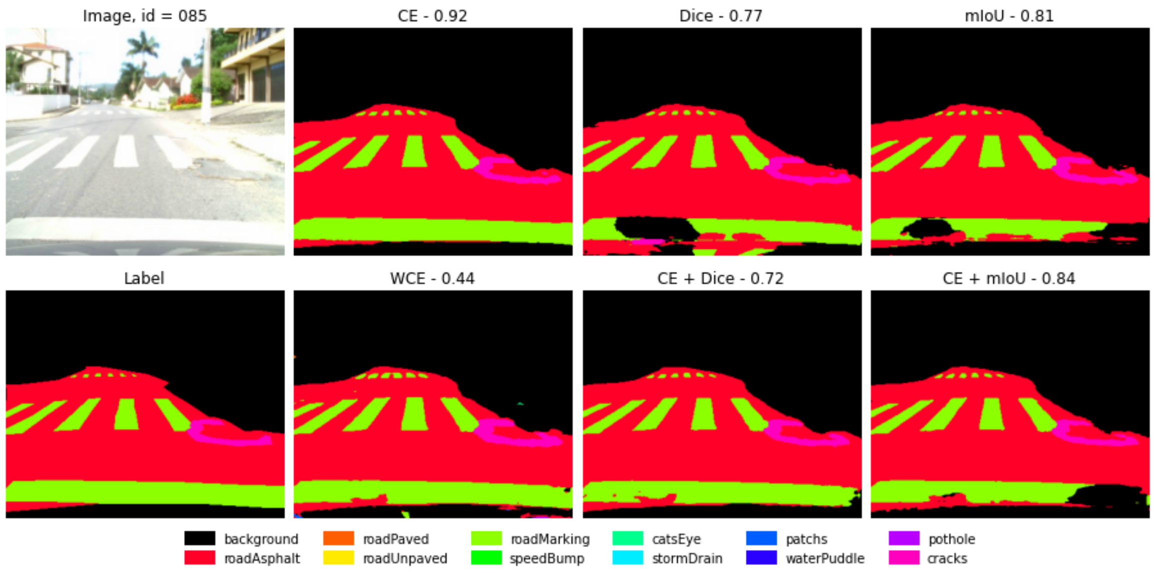


Figure 45 – Inspection of small objects predictions.

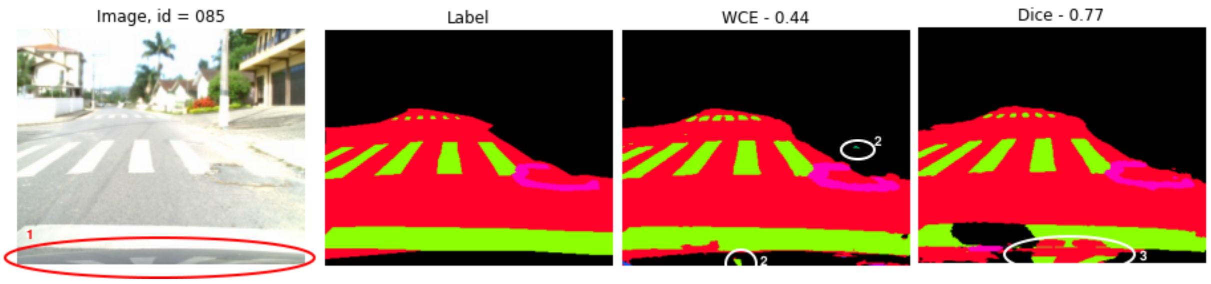


Figure 46 – Notes from Figure 45. Point 1 underlines the pedestrian lane reflected on the car hood; point 2 shows *WCE* over predicting underrepresented classes; point 3 shows the reflection predicted as a navigable path.

In Figure 47, we have a tiny storm drain at the right side of the scene; all hypotheses predicted it right. An interesting observation, the hypotheses had different resolutions for the farther road marking; see Figure 48.

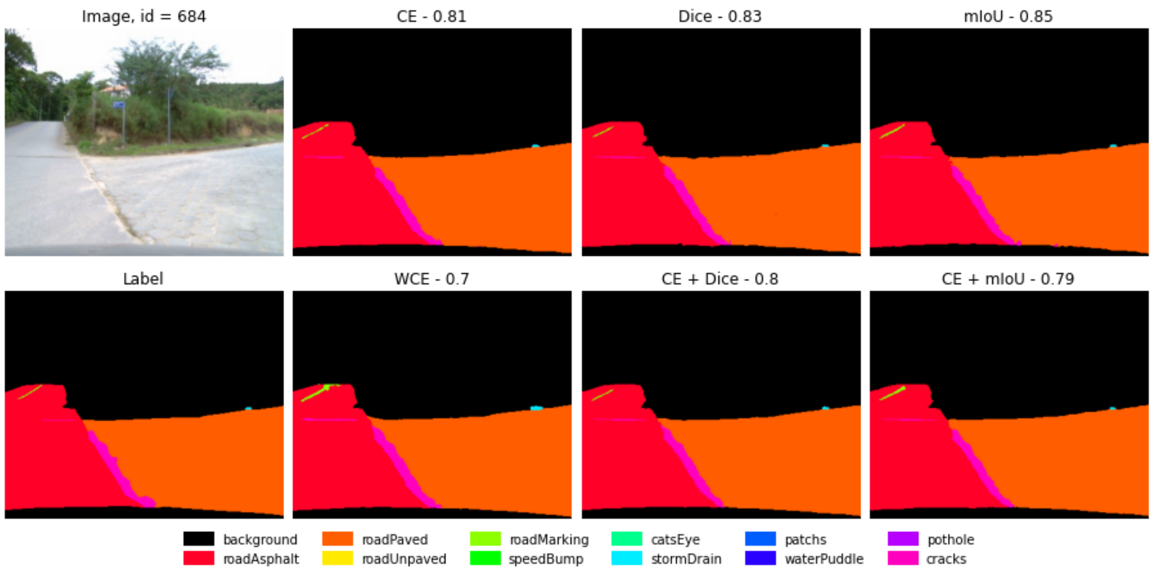


Figure 47 – Inspection of small objects predictions.

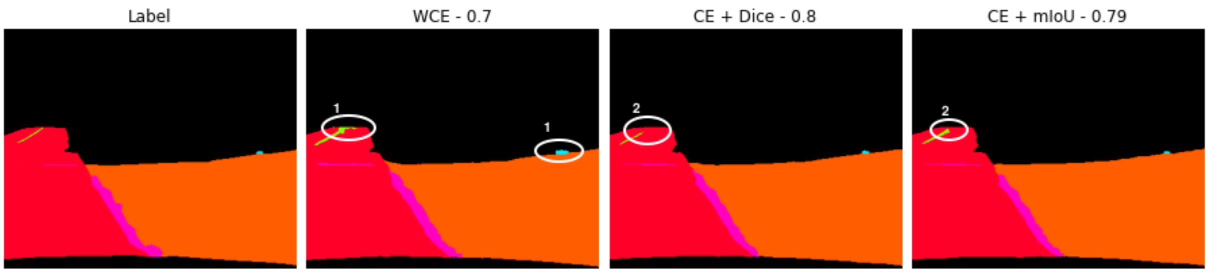


Figure 48 – Notes from Figure 47. Point 1 shows the larger blob prediction made by the *WCE* hypothesis; point 2 indicates that the *CE + Dice* hypothesis tends to vanish the lane in contrast to *CE + mIoU*, this pattern repeated in the *Dice* and *mIoU* standalone cases, see the Figure 47.

4.5.1 False Positive of Small Objects

Further investigations from the random blobs problem assigned on the sections 4.3.7 and 4.3.8 suggest that, in fact, the problem is about a high rate of false positives of small objects. The hypotheses with the max-pooling layer have a low-level output stride of 4, which early compress the spatial information; this seems to be the cause that confuses the neural network to judge correctly if a pixel belongs to a small or a large object. Consequently, the NN confusion leads to an excess of small objects as it is easier to yield many small objects than large objects, given their area. In the end, overrunning false positives of small objects seems a more serious concern than missing them. The Figures 49, 50, and 51 illustrate the cases.

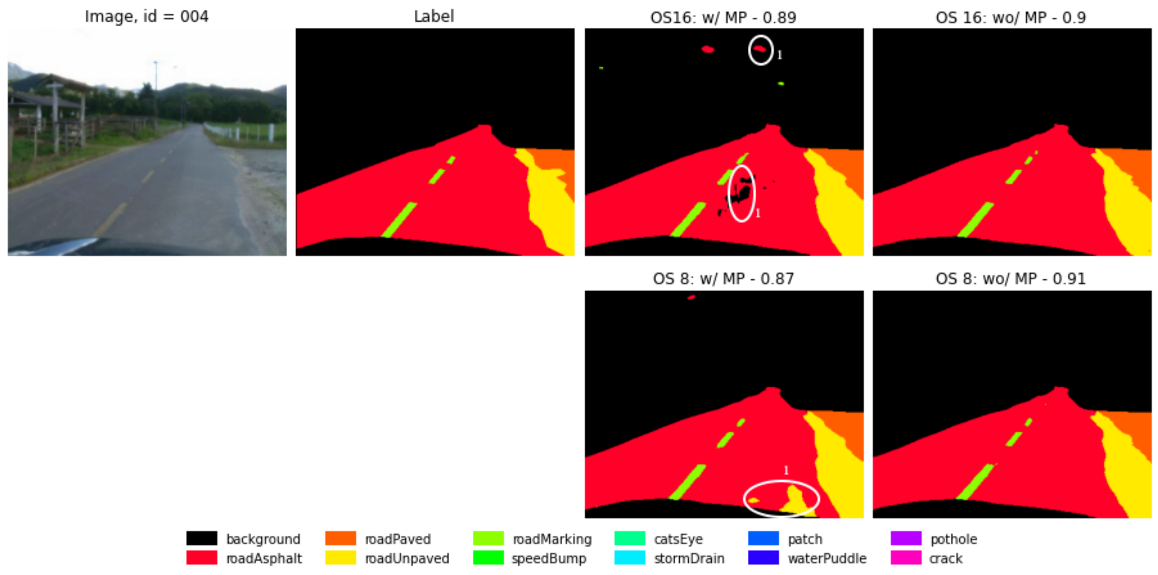


Figure 49 – False positive of small objects predicted by the hypotheses with max-pooling layer.

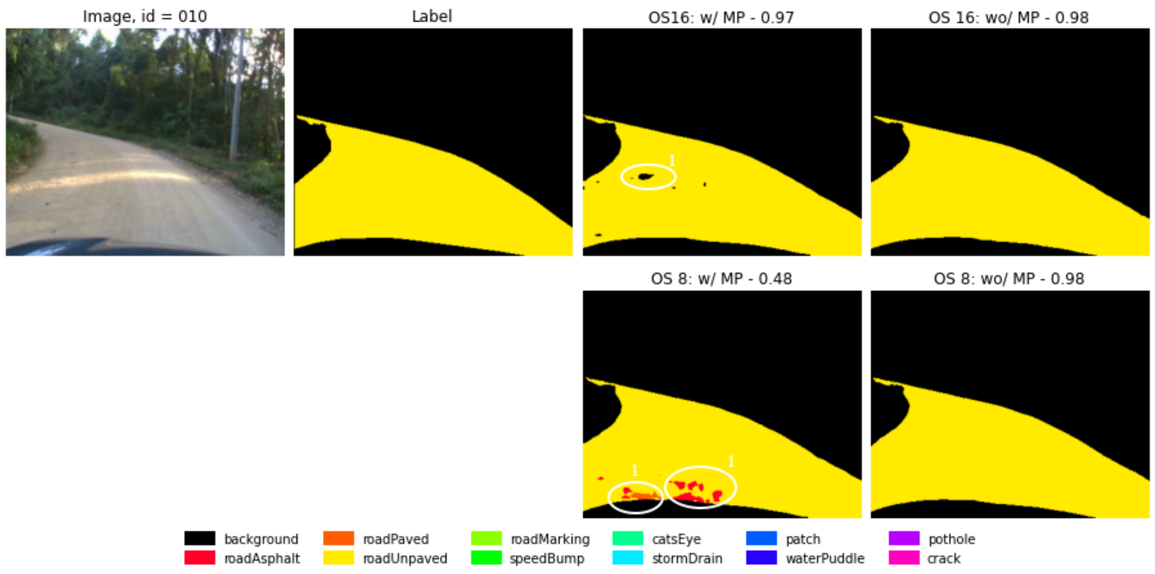


Figure 50 – False positive of small objects predicted by the hypotheses with max-pooling layer.

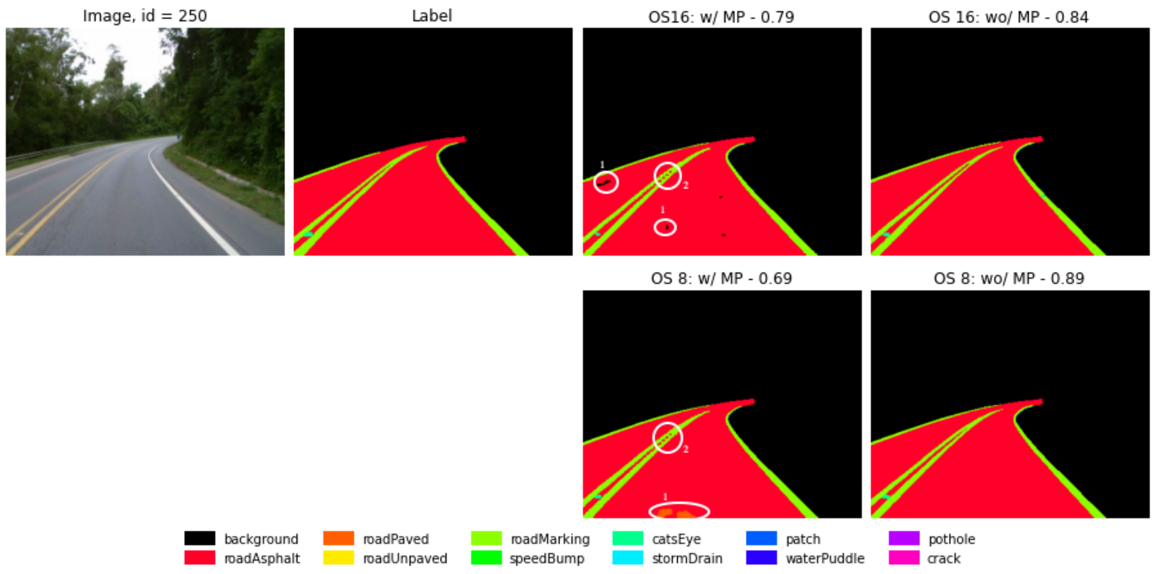


Figure 51 – Point 1 indicates the false positive small objects, and point 2 indicates the missed gap between the two markings predicted by the hypotheses with max-pooling layer.

4.6 INSPECTING MULTISCALE PREDICTION

The multiscale inspection presented scenes with many classes of different kinds of shapes and sizes. In the Figure 52, all hypotheses had fine predictions, however, *CE* shrank the water puddle; Figure 53 underlines at point 1 a missing lane not drawn in the label, but captured in the predictions.

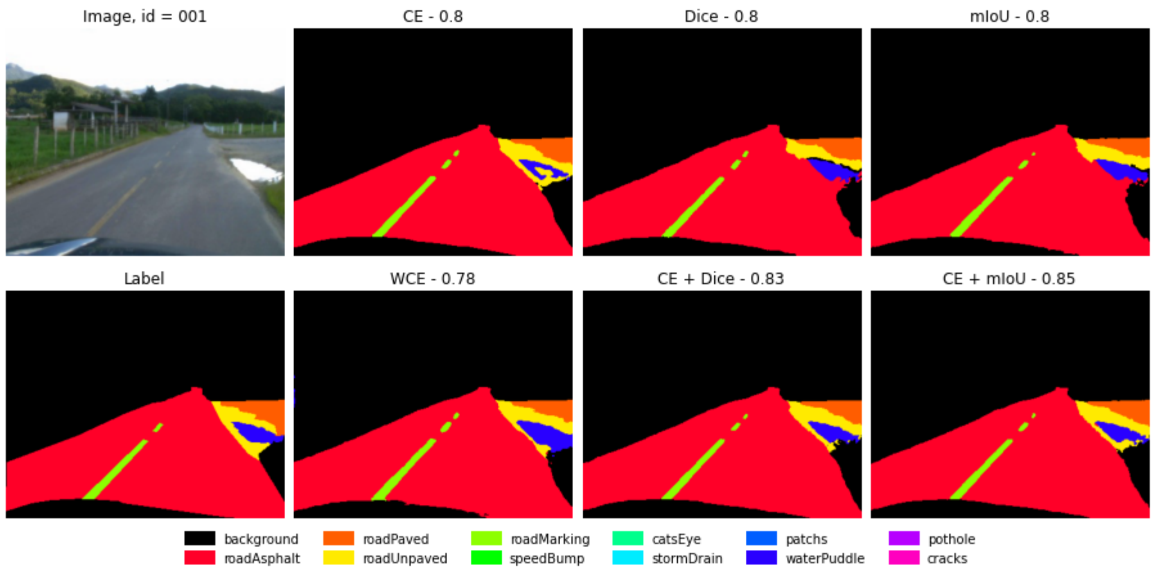


Figure 52 – Inspection of multiscale objects predictions.

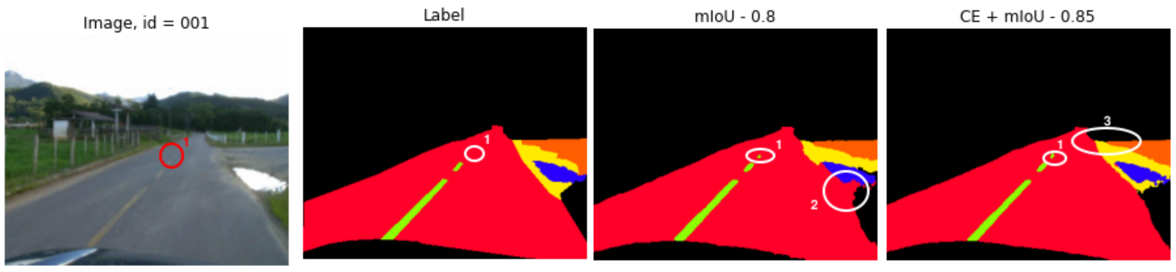


Figure 53 – Notes from Figure 52. Point 1 underlines missed lane in the label; point 2 indicates a missing surfaces transition by the *mIoU* hypothesis; point 3 underlines a nice transition between the surfaces.

In Figure 54, most of the hypotheses failed to predict all the three cat's eyes, missing one of them; except for *CE* and *WCE* that predicted all.

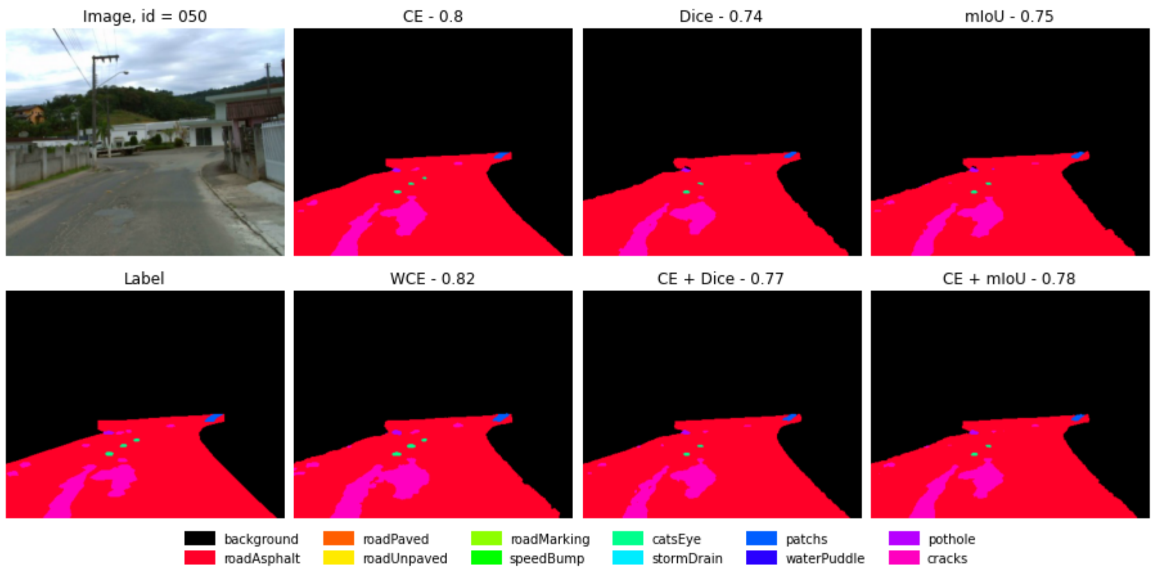


Figure 54 – Evaluation of multiscale object predictions.

Figure 55 presents a complex scene. All hypotheses performed well; however, most of them lost the paved blob outlined in circle 1 from Figure 56; only *WCE* overcame it.

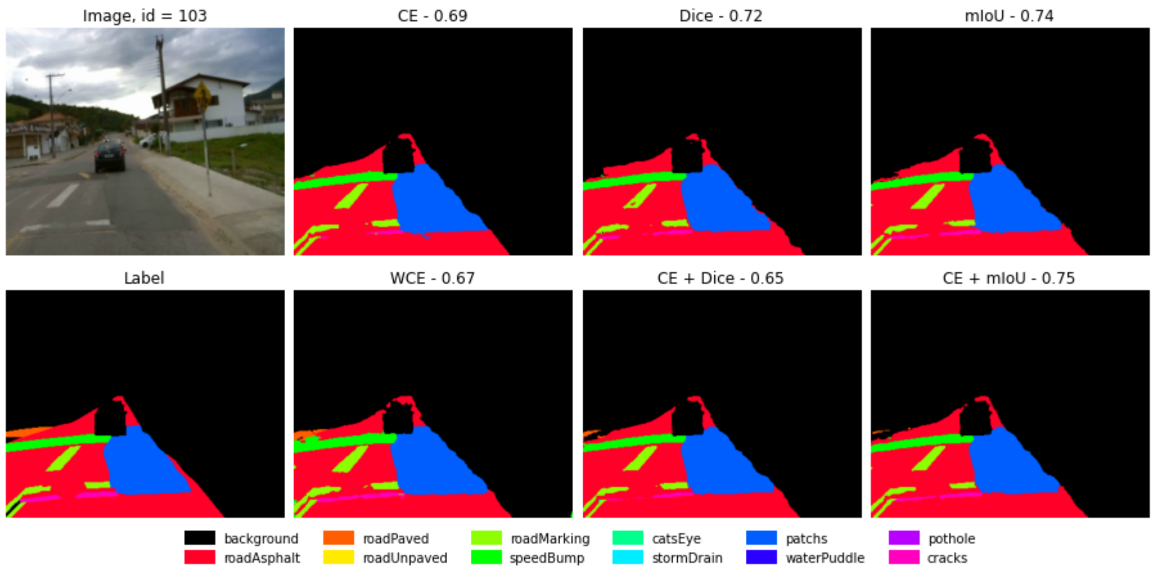


Figure 55 – Inspection of multiscale object predictions.

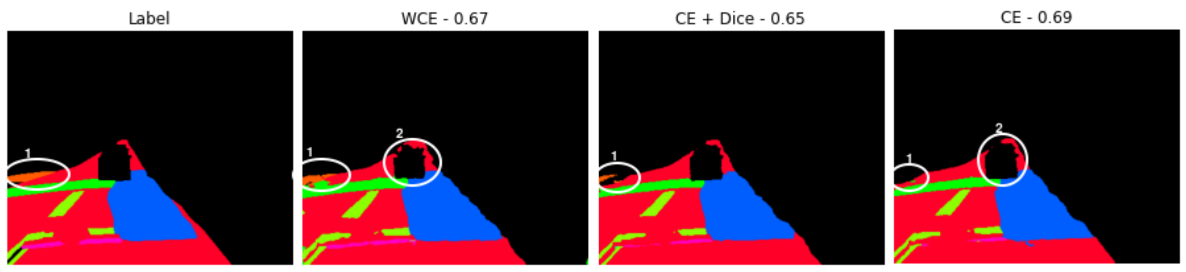


Figure 56 – Notes from Figure 55. Point 1 underlines the difficulty of predicting the small unpaved blob; point 2 shows that *WCE* does not present sharp boundaries as the *CE* hypothesis.

5 PERFORMANCE INCREMENT STRATEGY ON TAS500

Looking to endorse the performance increment strategy that showed a great enhancement for the RTK benchmark, we tested the strategy against another dataset, the TAS500. On this test, we want to validate our impressions first acquired in the RTK experiments or expand and better balance our conclusions.

5.1 TAS500 DATASET

The TAS500 dataset (METZGER; MORTIMER; WUENSCHE, 2021) is a novel dataset from 2021, which meets the complications found in autonomous driving in unstructured environments. TAS500 offers annotations of fine-grained vegetation and terrain classes to learn about drivable surfaces and natural obstacles in outdoor scenes. See samples of the dataset in the Figure 57.

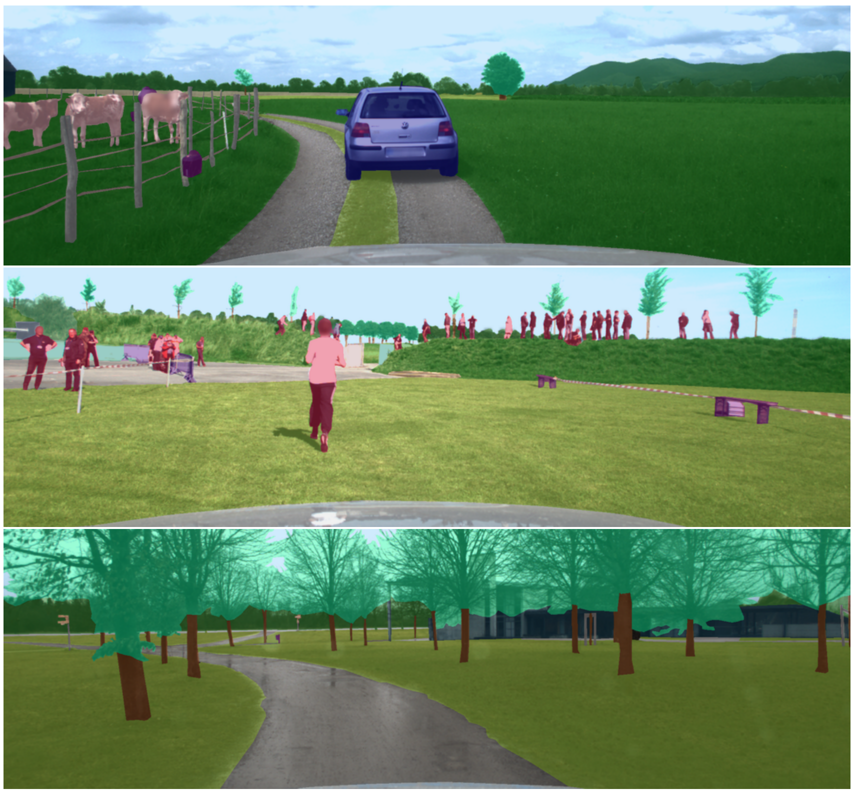


Figure 57 – TAS500 samples. From (METZGER; MORTIMER; WUENSCHE, 2021).

The dataset consists of 23 classes. They are categorized into **terrain**: asphalt, gravel, soil, and sand; **vegetation**: low grass, high grass, bush, forest, tree crown, tree trunk, and miscellaneous vegetation; **construction**: building, fence, and wall; **vehicles**: car and bus; objects: pole, traffic sign, and miscellaneous objects; and **others**: sky, human; animal, and void. Figure 58 shows the class distribution of the number of pixels.

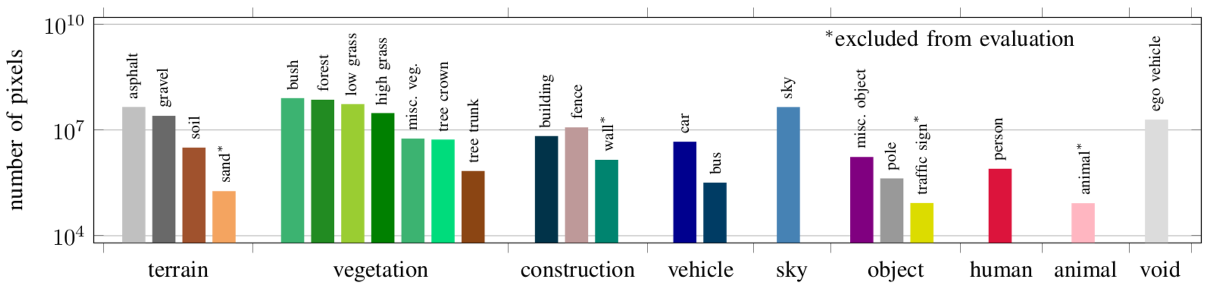


Figure 58 – Class distribution in the TAS500 dataset. The number of fine-grained pixels (y-axis) per class and their associated category (x-axis). From (METZGER; MORTIMER; WUENSCHE, 2021).

The dataset has 540 images with a resolution of 2026 x 620 pixels. The images were recorded in all seasons, but in winter, in sunny and rainy conditions. The division of images for training and validation sets is 440 and 100, respectively. The dataset has an official web server[1] that benchmarks the results for a hidden test set of 100 images.

In 2021, the German Association for Pattern Recognition (DAGM) host the Outdoor Semantic Segmentation Challenge[2], where the top-3 mIoU results on the test set were: 0.675, 0.663, and 0.659. By the way, the third place was obtained by the DeepUFSC team.

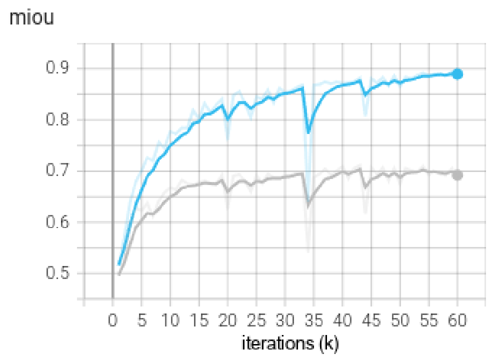
5.2 ABLATION EXPERIMENTS

As the TAS500 images are much larger than the RTK images, it takes much longer to complete a single iteration, so we ran only 60k iterations, much less than the previous 200k in the RTK experiments. However, 60k iterations proved to be enough to reach convergence on the performance, as seen in Figure 59. One more change from the previous setup was changing the batch size to 4 because of GPU memory restrictions, given the high-resolution pictures.

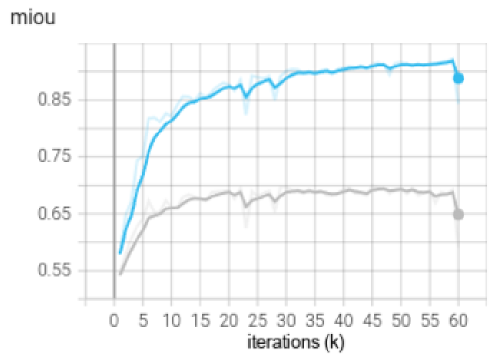
One caveat is that because of GPU memory constraints, some ablation experiments or hypotheses could not be accomplished, or they had their batch size reduced to 2. When it was tried batch size of 2, the performance did not go well, even reducing the learning rate by a factor of $\sqrt{2}$ as commonly recommended; see Figure 60. So, the discarded strategies were the *max-pooling removal* and *transposed convolutions* experiments, and many hypotheses from the *rich stem* and *loss functions* experiments.

Other constraining was given by the patch size, while we found that the widest patch was the best in *cropping size* experiment (Sec. 5.2.2), we could not use it for *output stride*, *rich stem*, *cutmix* and *loss functions* experiments given the GPU memory constraints.

[1] https://codalab.lisn.upsaclay.fr/competitions/5637
[2] https://unstructured-scene-understanding.com/challenge.html



(a) Cropping augmentation.



(b) Resizing & Cropping augmentation.

Figure 59 – Learning curve. 60k iterations are enough to reach convergence in the validation performance. Blue and gray lines are the evaluations of the training and validation sets, respectively.

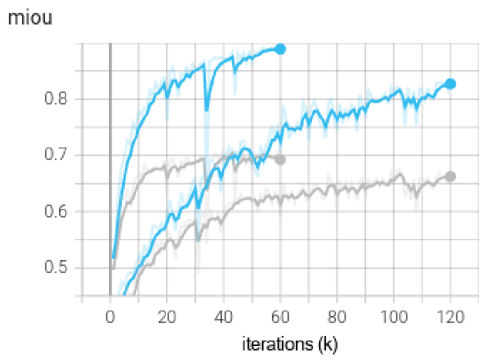


Figure 60 – Comparison between the same hypothesis running with batch size 2 (longer lines) and 4 (shorter lines). The blue and gray lines represent the training and validation sets, respectively. Even running twice iterations for batch size 2, it can not get close to the batch size of 4 performance.

Furthermore, we cut many hypotheses, the less relevant ones, from the performance increment strategy to reduce the application time; for example, we did not run the *networks* and *resnet variants* experiments.

The experiments setup started with DeepLabV3+ architecture, ResNet50 as a backbone, output stride 16, batch size 4, and Adam optimizer with a learning rate of 5e-5 (a reduced value from the 1e-4 in the RTK experiments given the reduced batch size). For validation evaluation, we followed the same methodology as the chapter before and averaged the performance of the last ten steps.

5.2.1 Technique: Data Augmentation

It is usual to train high-resolution images with cropped patches as it would take too long to do a single feedforward and backward pass and, consequently, to update the network's parameters, besides the GPU memory constraint. Thus, we used a fixed cropping operation with a patch size of 1024x512. The resizing works inside the scale of (0.83, 1.5).

Augmentation	mIoU
Resizing & Cropping	0.698
Resizing & Cropping & Color	0.694
Cropping	0.676
Cropping & Color	0.654

Table 21 – Output stride.

Table 21 shows that, in contrast to the RTK ablation, resizing mattered. Curiously, color augmentation also did not bring an improvement.

5.2.2 Technique: Cropping Size

We added a special experiment for TAS500, the patch size for cropping. The largest cropping size feasible for training with batch size 4 was 1248x512, given the GPU memory limit of 16GB. This experiment also used *resizing* found as best hypothesis in the *data augmentation* experiment.

Cropping Size	mIoU
1248x512	0.705
1024x512	0.698
900x512	0.693

Table 22 – Cropping size.

Table 22 presented a better performance for the wider patch.

5.2.3 Architecture: Output Stride

During the output stride experiment, we used the cropping size of 1024x512, given the GPU constraints. Table 23 shows that the best output stride of 16 plays an important role, having a considerable margin over the others.

OS	mIoU
16	0.698
8	0.665
32	0.652

Table 23 – Output stride.

5.2.4 Architecture: Rich Stem & HLFE

The rich stem experiment used the cropping size of 1024x512; meanwhile, the HLFE ablation used the cropping size of 1248x512. Both modifications did not improve the performance; see Table 24.

Cropping Size	Rich Stem	HLFE	mIoU
1024x512			0.698
1024x512	✓		0.693
1248x512			0.705
1248x512		✓	0.705

Table 24 – Rich Stem & HLFE.

5.2.5 Technique: Cutmix

This experiment used a resolution of 1024x512. Very different from the RTK results, we did not obtain an improvement using cutmix together with other augmentations.

Cutmix	mIoU
	0.698
✓	0.691

Table 25 – Cutmix.

5.2.6 Technique: Loss functions

We only tested the dice surrogate loss along with CE as that was the best alternative from the RTK experiment (Sec. 4.3.13). However, different from there, it did enhance the benchmark. See Table 26.

Loss	mIoU
CE + Dice	0.708
CE	0.698

Table 26 – Losses.

5.2.7 Technique: Optimizers

Another contrast to the RTK experiments was the considerable benchmark improvement when adopting SGD. It reached the new best benchmark of 0.725 mIoU, see Table 27.

Optimizer	mIoU
SGD	0.725
Adam	0.708

Table 27 – Optimizers.

5.2.8 Prediction: Voting Ensemble

Unlike the methodology of the previous experiments, the voting ensemble works with a single checkpoint model. In this case, the checkpoint model used was the one from the best validation evaluation. The extra scales had resolutions of 1013x310 and 2336x715. Table 28 shows that the voting ensemble, specially multiscale, significantly improved the TAS500 results. Again, it showed a different behavior than the RTK experiments that multiscale vaguely helped.

Prediction	mIoU
MS + Flipped	0.749
Flipped	0.736
Default	0.731

Table 28 – Predictions Voting Ensemble w/ Best Checkpoint.

5.3 STRATEGY ANALYSIS

One of the key points noticed during the performance increment strategy for the TAS500 benchmark was that a wider patch size made a significant difference, but it conflicted directly with hardware constraints. Other key points were resizing augmentation, CE + Dice loss function, and SGD were fundamental to raising the TAS500 benchmark. On the prediction side, the voting ensemble of multiple scales raised the performance substantially.

We started the experiments with a 0.654 mIoU and ended up with 0.747, which endorses the strategy experiments proposed. In the Figure 61 has some predictions done by our trained model.

5.4 TEST EVALUATION

Although the 2021 competition has ended, the hosts still make the competition server available for arbitrary submissions. So we checked how our best hypothesis - the one from the *optimizer* experiment - would perform on the Outdoor Semantic Segmentation Challenge. Using the default predictions, i.e., without the voting ensemble, we reached the benchmark of 0.676 mIoU on the test set. The drop in the mIoU number,

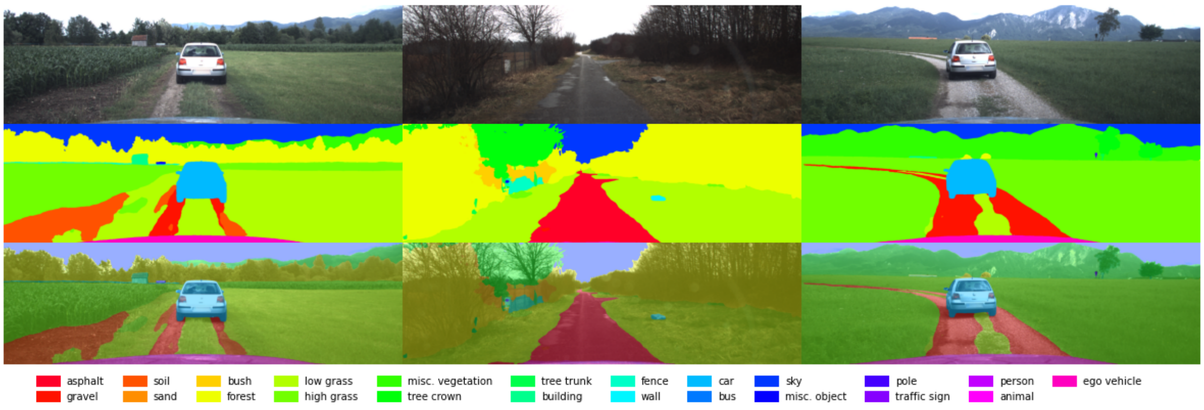


Figure 61 – Examples of TAS prediction.

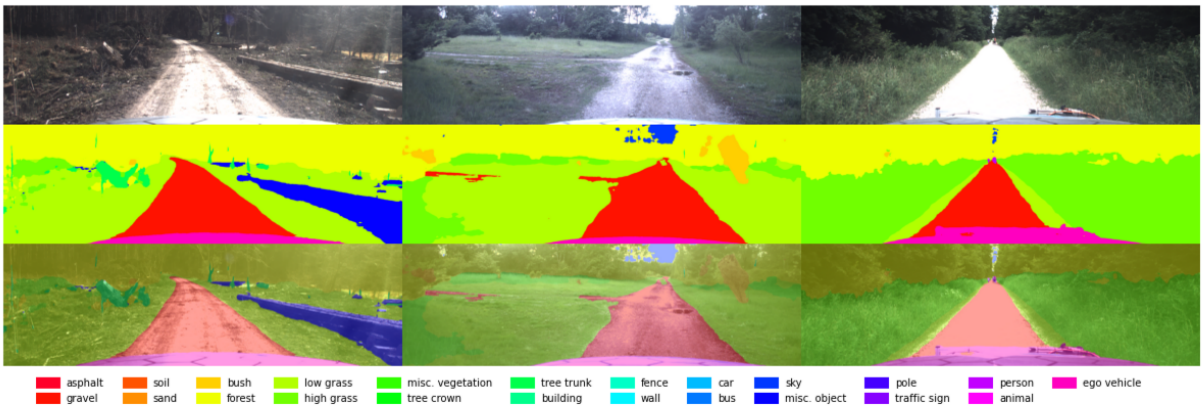


Figure 62 – Examples of TAS prediction.

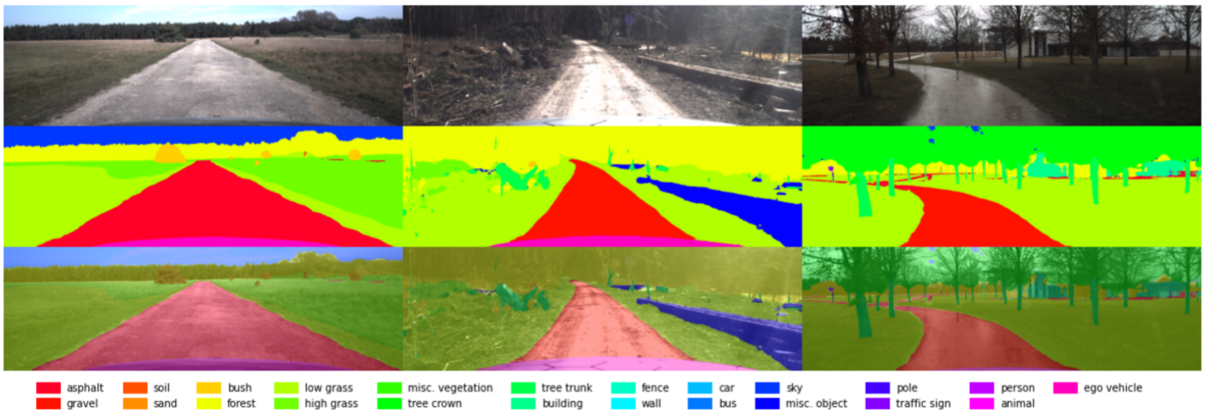


Figure 63 – Examples of TAS prediction.

when compared to the validation number, is due to a change in the classes distribution done in the test split; this is official information from the challenge hosts.

Afterward, we retrained the model by adding the validation data as training data; this new model reached 0.683 mIoU. Later, we applied the predictions voting ensemble that also raised the performance, reaching 0.688[3] mIoU. This last benchmark performance was the best reported so far on the web server, and all three evaluations made

[3] Results on https://codalab.lisn.upsaclay.fr/competitions/5637#results registered under the user *slow*.

in this section surpassed the best results of 0.675 mIoU from the 2021 challenge, endorsing the experiments from our performance increment strategy.

6 DISCUSSION AND CONCLUSIONS

As computer vision and artificial intelligence are increasingly involved in our daily routine, there is a need to adapt their solutions to local realities, as seen in emerging country roads, mainly because most of the cutting-edge technologies are first created in developed countries regarding their infrastructure. A correct road scene understanding for emerging countries should regard the issues of road damages and low-resolution images; for example, an autonomous navigation system that uses only low-resolution cameras must be aware that road damage may have a tiny size with few pixels.

These issues demand special attention when training a neural network model; for example, tiny objects are a problem if their locations vanish just after the first or second stride from the network, as seen in the RTK *max-pooling removal* experiment, which the hypotheses that avoided the max-pooling operation outperformed the alternatives that kept it (see Table 8); indicating the need to avoid compressing the data of the small object just at the beginning of the network.

Another RTK issue was the multiscale problem. First, we could confirm the mIoU bias to big objects; Table 18 and Figure 41 show the biggest objects, i.e., road surfaces and background, have the best metrics way far. In addition, Figure 42 shows a clear trend between the object edge size and mIoU, but it is not deterministic, e.g., the class *water puddle* has the worst performance of 0.48 mIoU with an intermediate edge size, whereas, *storm drain* has an intermediate performance of 0.68 mIoU with the second lowest edge size.

We noted that the hardest classes for predictions in damaged roads, based on the RTK dataset, are *cracks*, *water puddles*, and *cat's eyes*, which have very irregular shapes or present tiny objects.

It is known that deep learning models can require a long training time; hence, it is essential to keep track of training and validation metrics during the training. The baseline provided by the RTK authors missed the needed number of iterations, such that the simple act of increasing the number of iterations raised the mIoU metric from 0.547 to 0.739, a gain of 0.192 (Sec. 4.3.1); by far, it was the most increment of the series of experiments.

When trying to improve multiple setup parts, it was observed that the performance gain from a hypothesis might be overshadowed by a previous performance gain from another hypothesis. For example, on the RTK loss functions experiments when the training setup was not previously tuned, the compound losses, *CE + Dice* or *CE + mIoU*, worked better than bare CE, but, after the performance increment strategy started, the compound loss functions did not bring any improvement (Sec. 4.3.13). Thereby, we can say an experiment hypothesis performance gain depends on the setup condition that it is tested.

Another observation from the RTK loss functions experiment is that CE better optimizes the mIoU metric than even the surrogate mIoU loss function, the soft-mIoU, or the other region-based loss function, the soft-dice. It is well-known that some loss functions have a more proper loss landscape to find optimum solutions than others; it seemed to be the case with CE when compared to soft-mIoU or soft-Dice.

One problem raised when using the low-level output stride of 4 on the DeepLabV3+ to the RTK dataset, it was the prediction of random blobs from the big classes *road surfaces* (paved, unpaved, asphalt) and *background*. Later, we observed that this problem was actually a high rate of false positives of small objects; given the early compression of the spatial information, the NN was unsure if a pixel belonged to small or large objects. It visually damaged the predictions; the mIoU metric was not ruined by this problem, though (see Figure 35). We had two solutions that solved this problem: the max-pooling layer removal or the output stride diminishing; these solutions reveal the need for deeper features for features maps of larger spatial size when working with low-resolution images.

Among the attempts of DeepLabV3+ architecture modifications, we had unorthodox procedures, such as degressive rates, hybrid series of rates, and transposed convolutions. None of them emerged any rising on the results. The only effective architecture change was max-pooling layer removal, the most uncomplicated procedure, which mainly helped segment small objects.

The performance increment strategy worked for RTK and TAS500 datasets, although a very different subset of the hypotheses was better in each case. When cutmix augmentation, output stride of 4 if the max-pooling layer was present, CE loss, and Adam optimizer worked better for the RTK dataset; resizing augmentation, output stride of 16, CE + soft-Dice loss, and SGD optimizer worked better for the TAS500 dataset. The dissimilarity between these two hypotheses' subsets endorses the need for a custom solution in each context. Taking the freedom to expand the no free lunch statement (WOLPERT, 1996), that no machine learning algorithm is universally any better than any other, we could say that no training setup is universally any better than any other.

A curious fact observed in both datasets' strategies was that color augmentation did not bring any improvement and slightly degraded the performance. Maybe the color distributions between the training and the validation sets are the same, and color augmentation was unnecessary. Differently, cutmix augmentation brought a meaningful gain of 0.013% for the RTK benchmark, but it did not work for the TAS500. We suppose that the cutmix operation worked for the RTK dataset, given the tricky scenario of road surfaces filled with rough transitions between classes and the irregular shapes of many objects.

Another finding was that cropping augmentation, usually done to handle high-resolution images, also benefits low-resolution images like the RTK ones. We suppose it is because of the resampling power of cropping generating many versions of the same image.

The output strides showed to be an essential parameter to tune on the DeepLabV3+ architecture; see Tables 7 and 23. In the TAS500 dataset, the output stride 16 was crucial; it got a gain margin of 0.032% from the second best hypothesis of output stride 8. In the RTK dataset, the difference between the output strides was slight, only 0.006%, although still necessary to set it to 4 when removing the max-pooling layer.

The usual tactic to gain a few more points in competitions, predictions voting ensemble, achieved an extra performance in both benchmarks. Nevertheless, it is not practical for real-world applications that require fast responses, given the burden of computing multiple passes for the same image.

6.1 SUMMARY OF FINDINGS

- The segmentation of small objects is a problem if the feature map resolution is reduced before extracting deep features. The network tends specifically to predict an excess of false positives than miss small objects; it has high recall but low precision.
- The hardest RTK classes to segment are *cracks*, *water puddles*, and *cat's eyes*.
- We could confirm the mIoU towards big objects.
- CE optimizes the mIoU metric better than the surrogate mIoU and dice loss functions.
- An experiment hypothesis performance gain depends on the setup condition that it is tested.
- When an experiment hypothesis was better than another, it was better for all groups of classes.
- The high dissimilarity between the best setup for each dataset endorses the need for a custom solution in each context.
- Conventional setups usually bring better results than fancy tricks and should be the first attempt.

6.2 CONCLUSIONS

We conclude that the performance increment strategy worked for both datasets. It can be a recipe of suggestions to follow when working for emerging country roads or even expand it for a broad range of semantic segmentation problems. For the RTK dataset, the strategy raised the benchmark from 0.547 to 0.798 mIoU. For the TAS500 dataset, we reached a benchmark of 0.688 mIoU on the test set, which is superior to the 2021 outdoor challenge top-1 of 0.675 mIoU. We believe these enhancements are a step forward on the arduous goal of autonomous driving for emerging countries.

6.3 FUTURE WORKS

Other state-of-the-art techniques would be worth trying together with this work experiments are: Conditional Random Fields (CRF) (KRÄHENBÜHL; KOLTUN, 2011), a post-processing step after the model's prediction that sharpens the shape of the predicted object, and was also used in DeepLabV2 (CHEN et al., 2017); Tversky loss function (SALEHI; ERDOGMUS; GHOLIPOUR, 2017) which better balance the trade-off between precision and recall. Generalized Dice Loss (GDL) (SUDRE et al., 2017) which adjusts the contribution of each label by the inverse of its volume, such it reduces the dice score bias towards big objects.

On the evaluation side, we regard the adoption of the instance-level intersection-over-union metric (iIoU) presented in the Cityscapes work (CORDTS et al., 2016) which, in contrast to the traditional IoU, the contribution of each pixel is weighted by the ratio of the class' average instance size. Moreover, there is the Boundary Jaccard (FERNANDEZ-MORAL et al., 2018) metric, similar to the mIoU metric but penalizes the wrong borders predictions; it was adopted on the Outdoor Semantic Segmentation Challenge [1].

Furthermore, given the computational constraints on emerging countries, an additional study of the computational costs like memory footprint and inference time between the different architectures solutions would complement this work's results.

[1] https://unstructured-scene-understanding.com/challenge.html

REFERENCES

ALHAIJA, Hassan; MUSTIKOVELA, Siva; MESCHEDER, Lars; GEIGER, Andreas; ROTHER, Carsten. Augmented Reality Meets Computer Vision: Efficient Data Generation for Urban Driving Scenes. **International Journal of Computer Vision (IJCV)**, 2018.

AL-ANTARI, Mugahed A; AL-MASNI, Mohammed A; CHOI, Mun-Taek; HAN, Seung-Moo; KIM, Tae-Seong. A fully integrated computer-aided diagnosis system for digital X-ray mammograms via deep learning detection, segmentation, and classification. **International journal of medical informatics**, Elsevier, v. 117, p. 44–54, 2018.

BERMAN, Maxim; TRIKI, Amal Rannen; BLASCHKO, Matthew B. The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In: PROCEEDINGS of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2018. P. 4413–4421.

BERNAL, Jorge; SÁNCHEZ, Javier; VILARINO, Fernando. Towards automatic polyp detection with a polyp appearance model. **Pattern Recognition**, Elsevier, v. 45, n. 9, p. 3166–3182, 2012.

BEUCHER, Serge. Use of watersheds in contour detection. In: CCETT. PROCEEDINGS of the International Workshop on Image Processing. [S.l.: s.n.], 1979.

BROSTOW, Gabriel J; FAUQUEUR, Julien; CIPOLLA, Roberto. Semantic object classes in video: A high-definition ground truth database. **Pattern Recognition Letters**, Elsevier, v. 30, n. 2, p. 88–97, 2009.

CHEN, Liang-Chieh; PAPANDREOU, George; KOKKINOS, Iasonas; MURPHY, Kevin; YUILLE, Alan L. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 40, n. 4, p. 834–848, 2017.

CHEN, Liang-Chieh; YANG, Yi; WANG, Jiang; XU, Wei; YUILLE, Alan L. Attention to scale: Scale-aware semantic image segmentation. In: PROCEEDINGS of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2016. P. 3640–3649.

CHEN, Liang-Chieh; ZHU, Yukun; PAPANDREOU, George; SCHROFF, Florian; ADAM, Hartwig. Encoder-decoder with atrous separable convolution for semantic image segmentation. In: PROCEEDINGS of the European conference on computer vision (ECCV). [S.l.: s.n.], 2018. P. 801–818.

CHRISTIANSEN, Peter; NIELSEN, Lars N; STEEN, Kim A; JØRGENSEN, Rasmus N; KARSTOFT, Henrik. DeepAnomaly: Combining background subtraction and deep learning for detecting obstacles and anomalies in an agricultural field. **Sensors**, MDPI, v. 16, n. 11, p. 1904, 2016.

CNT. **Pesquisa CNT de rodovias 2021**. [S.l.]: SEST SENAT, 2021.

CORDTS, Marius; OMRAN, Mohamed; RAMOS, Sebastian; REHFELD, Timo; ENZWEILER, Markus; BENENSON, Rodrigo; FRANKE, Uwe; ROTH, Stefan; SCHIELE, Bernt. The cityscapes dataset for semantic urban scene understanding. In: PROCEEDINGS of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2016. P. 3213–3223.

DEVARAJ, Jayanthi; GANESAN, Sumathi; ELAVARASAN, Rajvikram Madurai; SUBRAMANIAM, Umashankar. A novel deep learning based model for tropical intensity estimation and post-disaster management of hurricanes. **Applied Sciences**, Multidisciplinary Digital Publishing Institute, v. 11, n. 9, p. 4129, 2021.

DICE, Lee R. Measures of the amount of ecologic association between species. **Ecology**, JSTOR, v. 26, n. 3, p. 297–302, 1945.

DUMOULIN, Vincent; VISIN, Francesco. A guide to convolution arithmetic for deep learning. **arXiv preprint arXiv:1603.07285**, 2016.

EVERINGHAM, Mark; ESLAMI, SM; VAN GOOL, Luc; WILLIAMS, Christopher KI; WINN, John; ZISSERMAN, Andrew. The pascal visual object classes challenge: A retrospective. **International journal of computer vision**, Springer, v. 111, n. 1, p. 98–136, 2015.

FERNANDEZ-MORAL, Eduardo; MARTINS, Renato; WOLF, Denis; RIVES, Patrick. A new metric for evaluating semantic segmentation: leveraging global and contour accuracy. In: IEEE. 2018 IEEE intelligent vehicles symposium (iv). [S.l.: s.n.], 2018. P. 1051–1056.

FUKUSHIMA, Kunihiko. Cognitron: A self-organizing multilayered neural network. **Biological cybernetics**, Springer, v. 20, n. 3, p. 121–136, 1975.

FUKUSHIMA, Kunihiko; MIYAKE, Sei. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In: COMPETITION and cooperation in neural nets. [S.l.]: Springer, 1982. P. 267–285.

GAO, Shang-Hua; CHENG, Ming-Ming; ZHAO, Kai; ZHANG, Xin-Yu; YANG, Ming-Hsuan; TORR, Philip. Res2net: A new multi-scale backbone architecture. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 43, n. 2, p. 652–662, 2019.

GEIGER, Andreas; LENZ, Philip; STILLER, Christoph; URTASUN, Raquel. Vision meets robotics: The kitti dataset. **The International Journal of Robotics Research**, Sage Publications Sage UK: London, England, v. 32, n. 11, p. 1231–1237, 2013.

HAMAGUCHI, Ryuhei; FUJITA, Aito; NEMOTO, Keisuke; IMAIZUMI, Tomoyuki; HIKOSAKA, Shuhei. Effective use of dilated convolutions for segmenting small object instances in remote sensing imagery. In: IEEE. 2018 IEEE winter conference on applications of computer vision (WACV). [S.l.: s.n.], 2018. P. 1442–1450.

HANG KWOK, Chi. **IoU vs F1 score**. [S.l.: s.n.]. Accessed: 2022-08-17. Available from: `https://tomkwok.com/posts/iou-vs-f1`.

HAYKIN, Simon. **Neural networks and learning machines, 3/E**. [S.l.]: Pearson Education India, 2009.

HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. Deep residual learning for image recognition. In: PROCEEDINGS of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2016. P. 770–778.

HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. Spatial pyramid pooling in deep convolutional networks for visual recognition. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 37, n. 9, p. 1904–1916, 2015.

HE, Tong; ZHANG, Zhi; ZHANG, Hang; ZHANG, Zhongyue; XIE, Junyuan; LI, Mu. Bag of tricks for image classification with convolutional neural networks. In: PROCEEDINGS of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. [S.l.: s.n.], 2019. P. 558–567.

HOIEM, Derek; HAYS, James; XIAO, Jianxiong; KHOSLA, Aditya. Guest editorial: Scene understanding. **International Journal of Computer Vision**, Springer, v. 112, n. 2, p. 131–132, 2015.

HOLSCHNEIDER, Matthias; KRONLAND-MARTINET, Richard; MORLET, Jean; TCHAMITCHIAN, Ph. A real-time algorithm for signal analysis with the help of the wavelet transform. In: WAVELETS. [S.l.]: Springer, 1990. P. 286–297.

HORN, Berthold; KLAUS, Berthold; HORN, Paul. **Robot vision**. [S.l.]: MIT press, 1986.

HU, Peiyun; RAMANAN, Deva. Finding tiny faces. In: PROCEEDINGS of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2017. P. 951–959.

HUANG, Huimin; LIN, Lanfen; TONG, Ruofeng; HU, Hongjie; ZHANG, Qiaowei; IWAMOTO, Yutaro; HAN, Xianhua; CHEN, Yen-Wei; WU, Jian. Unet 3+: A full-scale connected unet for medical image segmentation. In: IEEE. ICASSP 2020-2020 IEEE

International Conference on Acoustics, Speech and Signal Processing (ICASSP). [S.l.: s.n.], 2020. P. 1055–1059.

HUBEL, David H; WIESEL, Torsten N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. **The Journal of physiology**, Wiley-Blackwell, v. 160, n. 1, p. 106, 1962.

IOFFE, Sergey; SZEGEDY, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: PMLR. INTERNATIONAL conference on machine learning. [S.l.: s.n.], 2015. P. 448–456.

JADON, Shruti. A survey of loss functions for semantic segmentation. In: IEEE. 2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB). [S.l.: s.n.], 2020. P. 1–7.

JHA, Debesh; SMEDSRUD, Pia H; RIEGLER, Michael A; HALVORSEN, Pål; LANGE, Thomas de; JOHANSEN, Dag; JOHANSEN, Håvard D. Kvasir-seg: A segmented polyp dataset. In: SPRINGER. INTERNATIONAL Conference on Multimedia Modeling. [S.l.: s.n.], 2020. P. 451–462.

KARPATHY, Andrej et al. Cs231n convolutional neural networks for visual recognition. **Neural networks**, v. 1, n. 1, 2016.

KENDRICK, Connah; TAN, Kevin; WALKER, Kevin; YAP, Moi Hoon. The Application of Neural Networks for Facial Landmarking on Mobile Devices. In: VISIGRAPP (4: VISAPP). [S.l.: s.n.], 2018. P. 189–197.

KESKAR, Nitish Shirish; SOCHER, Richard. Improving generalization performance by switching from adam to sgd. **arXiv preprint arXiv:1712.07628**, 2017.

KHAN, Mohammad Mahmudur Rahman; SAKIB, Shadman; SIDDIQUE, Md Abu Bakr; CHOWDHURY, Madiha; HOSSAIN, Ziad; AZIZ, Anas; YASMIN, Nowrin. Automatic detection of covid-19 disease in chest x-ray images using deep neural networks. In: IEEE. 2020 IEEE 8th R10 Humanitarian Technology Conference (R10-HTC). [S.l.: s.n.], 2020. P. 1–6.

KIRILLOV, Alexander; WU, Yuxin; HE, Kaiming; GIRSHICK, Ross. Pointrend: Image segmentation as rendering. In: PROCEEDINGS of the IEEE/CVF conference on computer vision and pattern recognition. [S.l.: s.n.], 2020. P. 9799–9808.

KRÄHENBÜHL, Philipp; KOLTUN, Vladlen. Efficient inference in fully connected crfs with gaussian edge potentials. **Advances in neural information processing systems**, v. 24, 2011.

KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. Imagenet classification with deep convolutional neural networks. **Advances in neural information processing systems**, v. 25, 2012.

KURENKOV, Andrey. A Brief History of Neural Nets and Deep Learning. **Skynet Today**, 2020.

LAZEBNIK, Svetlana; SCHMID, Cordelia; PONCE, Jean. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: IEEE. 2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06). [S.l.: s.n.], 2006. v. 2, p. 2169–2178.

LECUN, Yann et al. Generalization and network design strategies. **Connectionism in perspective**, North Holland, v. 19, n. 143-155, p. 18, 1989.

LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep learning. **nature**, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015.

LECUN, Yann; BOSER, Bernhard; DENKER, John; HENDERSON, Donnie; HOWARD, Richard; HUBBARD, Wayne; JACKEL, Lawrence. Handwritten digit recognition with a back-propagation network. **Advances in neural information processing systems**, v. 2, 1989.

LECUN, Yann; BOSER, Bernhard; DENKER, John S; HENDERSON, Donnie; HOWARD, Richard E; HUBBARD, Wayne; JACKEL, Lawrence D. Backpropagation applied to handwritten zip code recognition. **Neural computation**, MIT Press, v. 1, n. 4, p. 541–551, 1989.

LECUN, Yann; BOTTOU, Léon; BENGIO, Yoshua; HAFFNER, Patrick. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, Ieee, v. 86, n. 11, p. 2278–2324, 1998.

LECUN, Yann; KAVUKCUOGLU, Koray; FARABET, Clément. Convolutional networks and applications in vision. In: IEEE. PROCEEDINGS of 2010 IEEE international symposium on circuits and systems. [S.l.: s.n.], 2010. P. 253–256.

LI, Yuxia; PENG, Bo; HE, Lei; FAN, Kunlong; LI, Zhenxu; TONG, Ling. Road extraction from unmanned aerial vehicle remote sensing images based on improved neural networks. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 19, n. 19, p. 4115, 2019.

LIN, Haoning; SHI, Zhenwei; ZOU, Zhengxia. Maritime semantic labeling of optical remote sensing images with multi-scale fully convolutional network. **Remote sensing**, Multidisciplinary Digital Publishing Institute, v. 9, n. 5, p. 480, 2017.

LLOYD, Stuart. Least squares quantization in PCM. **IEEE transactions on information theory**, IEEE, v. 28, n. 2, p. 129–137, 1982.

LONG, Jonathan; SHELHAMER, Evan; DARRELL, Trevor. Fully convolutional networks for semantic segmentation. In: PROCEEDINGS of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2015. P. 3431–3440.

MAEDA, Hiroya; SEKIMOTO, Yoshihide; SETO, Toshikazu; KASHIYAMA, Takehiro; OMATA, Hiroshi. Road damage detection and classification using deep neural networks with smartphone images. **Computer-Aided Civil and Infrastructure Engineering**, Wiley Online Library, v. 33, n. 12, p. 1127–1141, 2018.

MCCULLOCH, Warren S; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, Springer, v. 5, n. 4, p. 115–133, 1943.

METZGER, Kai A; MORTIMER, Peter; WUENSCHE, Hans-Joachim. A fine-grained dataset and its efficient semantic segmentation for unstructured driving scenarios. In: IEEE. 2020 25th International Conference on Pattern Recognition (ICPR). [S.l.: s.n.], 2021. P. 7892–7899.

MILLETARI, Fausto; NAVAB, Nassir; AHMADI, Seyed-Ahmad. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In: IEEE. 2016 fourth international conference on 3D vision (3DV). [S.l.: s.n.], 2016. P. 565–571.

MINSKY, Marvin; PAPERT, Seymour. Perceptrons. MIT press, 1969.

MURPHY, Kevin P. **Machine learning: a probabilistic perspective**. [S.l.]: MIT press, 2012.

NAGENDAR, Gattigorla; SINGH, Digvijay; BALASUBRAMANIAN, Vineeth N; JAWAHAR, CV. Neuro-IoU: Learning a Surrogate Loss for Semantic Segmentation. In: BMVC. [S.l.: s.n.], 2018. P. 278.

NOH, Hyeonwoo; HONG, Seunghoon; HAN, Bohyung. Learning deconvolution network for semantic segmentation. In: PROCEEDINGS of the IEEE international conference on computer vision. [S.l.: s.n.], 2015. P. 1520–1528.

OTSU, Nobuyuki. A threshold selection method from gray-level histograms. **IEEE transactions on systems, man, and cybernetics**, IEEE, v. 9, n. 1, p. 62–66, 1979.

RAHMAN, Md Atiqur; WANG, Yang. Optimizing intersection-over-union in deep neural networks for image segmentation. In: SPRINGER. INTERNATIONAL symposium on visual computing. [S.l.: s.n.], 2016. P. 234–244.

RATEKE, Thiago. [S.l.: s.n.], Aug. 2020. Available from: `https://towardsdatascience.com/road-surface-semantic-segmentation-4d65b045245`.

RATEKE, Thiago; JUSTEN, Karla Aparecida; VON WANGENHEIM, Aldo. Road surface classification with images captured from low-cost camera-road traversing knowledge (rtk) dataset. **Revista de Informática Teórica e Aplicada**, v. 26, n. 3, p. 50–64, 2019.

RATEKE, Thiago; VON WANGENHEIM, Aldo. Road surface detection and differentiation considering surface damages. **Autonomous Robots**, Springer, v. 45, n. 2, p. 299–312, 2021.

RONNEBERGER, Olaf; FISCHER, Philipp; BROX, Thomas. U-net: Convolutional networks for biomedical image segmentation. In: SPRINGER. INTERNATIONAL Conference on Medical image computing and computer-assisted intervention. [S.l.: s.n.], 2015. P. 234–241.

ROSENBLATT, Frank. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, American Psychological Association, v. 65, n. 6, p. 386, 1958.

RUMELHART, David E; HINTON, Geoffrey E; WILLIAMS, Ronald J. **Learning internal representations by error propagation**. [S.l.], 1985.

SALEHI, Seyed Sadegh Mohseni; ERDOGMUS, Deniz; GHOLIPOUR, Ali. Tversky loss function for image segmentation using 3D fully convolutional deep networks. In: SPRINGER. INTERNATIONAL workshop on machine learning in medical imaging. [S.l.: s.n.], 2017. P. 379–387.

SCHMIDHUBER, Jürgen. Deep learning in neural networks: An overview. **Neural networks**, Elsevier, v. 61, p. 85–117, 2015.

SHANG, Erke; ZHAO, Haiyang; LI, Jian; AN, Xiangjing; WU, Tao. OffRoadScene: An open database for unstructured road detection algorithms. In: IEEE. 2013 International Conference on Computer Sciences and Applications. [S.l.: s.n.], 2013. P. 779–783.

SHEN, Zhiqiang; LIU, Zhuang; LI, Jianguo; JIANG, Yu-Gang; CHEN, Yurong; XUE, Xiangyang. Dsod: Learning deeply supervised object detectors from scratch. In: PROCEEDINGS of the IEEE international conference on computer vision. [S.l.: s.n.], 2017. P. 1919–1927.

SHINZATO, Patrick Y et al. Carina dataset: An emerging-country urban scenario benchmark for road detection systems. In: IEEE. 2016 IEEE 19th international conference on intelligent transportation systems (ITSC). [S.l.: s.n.], 2016. P. 41–46.

SHORTEN, Connor; KHOSHGOFTAAR, Taghi M. A survey on image data augmentation for deep learning. **Journal of big data**, Springer, v. 6, n. 1, p. 1–48, 2019.

SRIVASTAVA, Nitish; HINTON, Geoffrey; KRIZHEVSKY, Alex; SUTSKEVER, Ilya; SALAKHUTDINOV, Ruslan. Dropout: a simple way to prevent neural networks from overfitting. **The journal of machine learning research**, JMLR. org, v. 15, n. 1, p. 1929–1958, 2014.

SRIVASTAVA, Rupesh Kumar; GREFF, Klaus; SCHMIDHUBER, Jürgen. Highway networks. **arXiv preprint arXiv:1505.00387**, 2015.

SUDRE, Carole H; LI, Wenqi; VERCAUTEREN, Tom; OURSELIN, Sebastien; JORGE CARDOSO, M. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In: DEEP learning in medical image analysis and multimodal learning for clinical decision support. [S.l.]: Springer, 2017. P. 240–248.

SZEGEDY, Christian; LIU, Wei; JIA, Yangqing; SERMANET, Pierre; REED, Scott; ANGUELOV, Dragomir; ERHAN, Dumitru; VANHOUCKE, Vincent; RABINOVICH, Andrew. Going deeper with convolutions. In: PROCEEDINGS of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2015. P. 1–9.

SZELISKI, Richard. **Computer vision: algorithms and applications**. [S.l.]: Springer Science & Business Media, 2010.

TAO, Andrew; SAPRA, Karan; CATANZARO, Bryan. Hierarchical multi-scale attention for semantic segmentation. **arXiv preprint arXiv:2005.10821**, 2020.

TAYLOR, Luke; NITSCHKE, Geoff. Improving deep learning with generic data augmentation. In: IEEE. 2018 IEEE Symposium Series on Computational Intelligence (SSCI). [S.l.: s.n.], 2018. P. 1542–1547.

WANG, Panqu; CHEN, Pengfei; YUAN, Ye; LIU, Ding; HUANG, Zehua; HOU, Xiaodi; COTTRELL, Garrison. Understanding convolution for semantic segmentation. In: IEEE. 2018 IEEE winter conference on applications of computer vision (WACV). [S.l.: s.n.], 2018. P. 1451–1460.

WOLPERT, David H. The lack of a priori distinctions between learning algorithms. **Neural computation**, MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . ., v. 8, n. 7, p. 1341–1390, 1996.

XIA, Haiying; ZHUGE, Ruibin; LI, Haisheng. Retinal vessel segmentation via a coarse-to-fine convolutional neural network. In: IEEE. 2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM). [S.l.: s.n.], 2018. P. 1036–1039.

XIE, Saining; GIRSHICK, Ross; DOLLÁR, Piotr; TU, Zhuowen; HE, Kaiming. Aggregated residual transformations for deep neural networks. In: PROCEEDINGS of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2017. P. 1492–1500.

YU, Changqian; GAO, Changxin; WANG, Jingbo; YU, Gang; SHEN, Chunhua; SANG, Nong. Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation. **International Journal of Computer Vision**, Springer, v. 129, n. 11, p. 3051–3068, 2021.

YU, Fisher; KOLTUN, Vladlen. Multi-scale context aggregation by dilated convolutions. **arXiv preprint arXiv:1511.07122**, 2015.

YU, Fisher; KOLTUN, Vladlen; FUNKHOUSER, Thomas. Dilated residual networks. In: PROCEEDINGS of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2017. P. 472–480.

YUAN, Yuhui; CHEN, Xiaokang; CHEN, Xilin; WANG, Jingdong. Segmentation transformer: Object-contextual representations for semantic segmentation. **arXiv preprint arXiv:1909.11065**, 2019.

YUN, Sangdoo; HAN, Dongyoon; OH, Seong Joon; CHUN, Sanghyuk; CHOE, Junsuk; YOO, Youngjoon. Cutmix: Regularization strategy to train strong classifiers with localizable features. In: PROCEEDINGS of the IEEE/CVF international conference on computer vision. [S.l.: s.n.], 2019. P. 6023–6032.

ZHAO, Hengshuang; SHI, Jianping; QI, Xiaojuan; WANG, Xiaogang; JIA, Jiaya. Pyramid scene parsing network. In: PROCEEDINGS of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2017. P. 2881–2890.

ZHOU, Bolei; ZHAO, Hang; PUIG, Xavier; FIDLER, Sanja; BARRIUSO, Adela; TORRALBA, Antonio. Scene parsing through ade20k dataset. In: PROCEEDINGS of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2017. P. 633–641.

ZHOU, Pan; FENG, Jiashi; MA, Chao; XIONG, Caiming; HOI, Steven Chu Hong, et al. Towards theoretically understanding why sgd generalizes better than adam in deep learning. **Advances in Neural Information Processing Systems**, v. 33, p. 21285–21296, 2020.

ZHOU, Peng; NI, Bingbing; GENG, Cong; HU, Jianguo; XU, Yi. Scale-transferrable object detection. In: PROCEEDINGS of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2018. P. 528–537.