



ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ  
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ  
ΤΟΜΕΑΣ ΣΗΜΑΤΩΝ, ΕΛΕΓΧΟΥ ΚΑΙ ΡΟΜΠΟΤΙΚΗΣ

# Semantic Segmentation with Deep Convolutional Neural Networks

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Σταμάτη Αλεξανδρόπουλου

**Επιβλέπων:** Πέτρος Μαραγκός  
Καθηγητής Ε.Μ.Π.

**Συνεπιβλέπων:** Χρήστος Σαχαρίδης  
Μεταδιδακτορικός Ερευνητής ΕΤΗ

ΕΡΓΑΣΤΗΡΙΟ ΟΡΑΣΗΣ ΥΠΟΛΟΓΙΣΤΩΝ, ΕΠΙΚΟΙΝΩΝΙΑΣ ΛΟΓΟΥ ΚΑΙ ΕΠΕΞΕΡΓΑΣΙΑΣ ΣΗΜΑΤΩΝ  
Αθήνα, Οκτώβριος 2022





Εθνικό Μετσόβιο Πολυτεχνείο  
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών  
Τομέας Σημάτων, Ελέγχου και Ρομποτικής  
Εργαστήριο Όρασης Υπολογιστών, Επικοινωνίας Λόγου και Επεξεργασίας  
Σημάτων

# Semantic Segmentation with Deep Convolutional Neural Networks

## ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΤΟΥ

Σταμάτη Αλεξανδρόπουλου

**Επιβλέπων:** Πέτρος Μαραγκός  
Καθηγητής Ε.Μ.Π.

**Συνεπιβλέπων:** Χρήστος Σακαρίδης  
Μεταδιδακτορικός Ερευνητής ΕΤΗ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 10<sup>η</sup> Οκτωβρίου, 2022.

.....  
Πέτρος Μαραγκός  
Καθηγητής Ε.Μ.Π.

.....  
Αθανάσιος Ροντογιάννης  
Αναπληρωτής Καθηγητής Ε.Μ.Π.

.....  
Γεράσιμος Ποταμιάνος  
Αναπληρωτής Καθηγητής Παν/μιο Θεσσαλίας

Αθήνα, Οκτώβριος 2022

.....  
**ΣΤΑΜΑΤΗΣ ΑΛΕΞΑΝΔΡΟΠΟΥΛΟΣ**  
Διπλωματούχος Ηλεκτρολόγος Μηχανικός  
και Μηχανικός Υπολογιστών Ε.Μ.Π.

Copyright © – All rights reserved Σταμάτης Αλεξανδρόπουλος, 2022.  
Με επιφύλαξη παντός δικαιώματος.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.



# Περίληψη

Η Σημασιολογική Κατάτμηση (Semantic Segmentation) είναι ένα από τα θεμελιώδη θέματα της Όρασης Υπολογιστών. Συγκεκριμένα, είναι η διαδικασία ανάθεσης μιας κατηγορίας σε κάθε pixel μιας εικόνας. Υπάρχει ευρύς αριθμός εφαρμογών της σε διάφορους τομείς, όπως η Αυτόνομη Οδήγηση, η Ρομποτική και η Επεξεργασία Ιατρικών Εικόνων, όπου η ετικετοποίηση (labeling) σε επίπεδο pixel θεωρείται μείζονος σημασίας. Τα Βαθιά Συνελικτικά Νευρωνικά Δίκτυα (Deep Convolutional Neural Networks) έχουν πετύχει πρόσφατα state-of-the-art επιδόσεις σε υψηλής κλίμακας προβλήματα αναγνώρισης. Αυτό έχει σαν αποτέλεσμα τέτοια μοντέλα να χρησιμοποιούνται πλέον στις προαναφερθείσες εφαρμογές αιχμής. Οι περισσότερες σχετικές εργασίες επικεντρώνονται σε αλλαγές στην αρχιτεκτονική στα χρησιμοποιούμενα δίκτυα προκειμένου να συνδυάσουν καλύτερα το περιεχόμενο ολόκληρης της εικόνας διατηρώντας παράλληλα τη λεπτομέρεια σε τοπικό επίπεδο και χρησιμοποιώντας ένα loss υπολογιζόμενο σε μεμονωμένα pixels. Ο σχεδιασμός πιο σύνθετων losses, που λαμβάνουν υπόψη τη δομή που περιέχεται στις σημασιολογικές ετικέτες, είναι αντικείμενο μεγάλης προσοχής. Ο στόχος αυτής της διπλωματικής είναι η μελέτη τέτοιων αρχών για το πρόβλημα της σημασιολογικής κατάτμησης και η χρησιμοποίησή τους στην επίβλεψη state-of-the-art δικτύων, προκειμένου να προκύψουν αποτελέσματα που αντικατοπτρίζουν καλύτερα την κανονικότητα των γνήσιων τμηματοποιήσεων.

Βασιζόμενοι στη γνώση σχετικά με την υψηλού επιπέδου κανονικότητα των πραγματικών σκηνών, προτείνουμε μια νέα μέθοδο για τη βελτίωση των προβλεπόμενων κλάσεων, μέσω της εκμάθησης της επιλεκτικής αξιοποίησης των πληροφοριών από τα συνεπίπεδα pixels. Συγκεκριμένα, εισάγουμε μια αρχή που προτείνει ότι για κάθε pixel υπάρχει ένα seed pixel, το οποίο ανήκει στην ίδια κλάση με το προηγούμενο. Σαν αποτέλεσμα, σχεδιάζουμε ένα νευρωνικό δίκτυο με δύο κεφαλές. Η πρώτη κεφαλή παράγει τις προβλεπόμενες κλάσεις για κάθε pixel, ενώ η δεύτερη παράγει ένα πυκνό πεδίο διανυσμάτων μετατόπισης (offset vectors) που προσδιορίζει τις θέσεις των seed pixels. Οι προβλέψεις των seed pixels χρησιμοποιούνται στη συνέχεια για να προβλέψουν την κλάση σε κάθε pixel. Προκειμένου να ληφθούν υπόψη πιθανές αποκλίσεις από την ακριβή τοπική ομαλότητα, η προκύπτουσα πρόβλεψη συγχωνεύεται προσαρμοστικά με την αρχική πρόβλεψη από την πρώτη κεφαλή χρησιμοποιώντας έναν χάρτη εμπιστοσύνης (confidence map), τον οποίο μαθαίνει το μοντέλο. Η συνολική αρχιτεκτονική έχει υλοποιηθεί στο μοντέλο HRNetV2, ένα state-of-the-art μοντέλο στο σύνολο δεδομένων Cityscapes. Το νέο HRNet μοντέλο που βασίζεται στα διανύσματα μετατόπισης εκπαιδεύεται τόσο στο Cityscapes όσο και στο ACDC σύνολο δεδομένων. Πραγματοποιούμε εκτενή ποιοτικά και ποσοτικά πειράματα και συγκρίνουμε τη μεθόδου μας με πρόσφατες μεθόδους. Τα πειράματα καταδεικνύουν την υπεροχή και τα πλεονεκτήματα της μεθόδου μας, η οποία επιτυγχάνει καλύτερα αποτελέσματα από το αρχικό μοντέλο. Ο κώδικάς μας είναι διαθέσιμος [εδώ](#).

**Λέξεις Κλειδιά** — Όραση Υπολογιστών, Σημασιολογική Κατάτμηση, Βαθιά Συνελικτικά Νευρωνικά Δίκτυα, Αυτόνομη Οδήγηση, HRNetV2, Seed Pixels



# Abstract

Semantic segmentation is one of the fundamental topics of computer vision. Specifically, it is the process of assigning a category to each pixel in an image. There are a number of applications in a variety of fields, such as Autonomous Driving, Robotics, and Medical Image Processing, where pixel-level labeling is critical. Deep Convolutional Neural Networks (DCNNs) have lately demonstrated state-of-the-art performance in high-level recognition tasks. As a result, such models may now be used in the above-mentioned cutting-edge applications. Most of the related works concentrate on architectural changes to the used networks in order to better combine global context aggregation with local detail preservation, and utilize a simple loss computed on individual pixels. Designing more complex losses that account for the structure contained in semantic labelings has gotten substantially less attention. The goal of this thesis is to investigate such priors for semantic segmentation and to use them in the supervision of state-of-the-art networks to get results that better reflect the regularity of genuine segmentations.

Based on knowledge about the high regularity of real scenes, we propose a method for improving class predictions by learning to selectively exploit information from coplanar pixels. In particular, we introduce a prior which claims that for each pixel, there is a seed pixel which shares the same prediction with the former. As a result of this, we design a network with two heads. The first head generates pixel-level classes, whereas the second generates a dense offset vector field that identifies seed pixel positions. Seed pixels' class predictions are then utilized to predict classes at each point. To account for possible deviations from precise local planarity, the resultant prediction is adaptively fused with the initial prediction from the first head using a learnt confidence map. The entire architecture is implemented on HRNetV2, a state-of-the-art model on Cityscapes dataset. The offset vector-based HRNetV2 was trained on both Cityscapes and ACDC datasets. We assess our method through extensive qualitative and quantitative experiments and ablation studies and compare it with recent state-of-the-art methods demonstrating its superiority and advantages. To sum up, we achieve better results than the initial model. Our source code can be found in our [project website](#).

**Keywords** — Computer Vision, Semantic Segmentation, Deep Convolutional Neural Networks, Autonomous Driving, HRNetV2, Seed Pixels



# Ευχαριστίες

Αρχικά με την ευκαιρία της παρούσης διπλωματικής μου, θα ήθελα να ευχαριστήσω θερμά τον καθηγητή κ. Πέτρο Μαραγκό, για την εμπιστοσύνη που έδειξε στο πρόσωπό μου και για την ευκαιρία που μου έδωσε προκειμένου να εκπονήσω την εν λόγω εργασία στο εργαστήριό του σε συνεργασία με το Computer Vision Lab (CVL) του ΕΤΗ. Αξίζει να αναφέρω ότι μέσα από τη διδασκαλία των προπτυχιακών του μαθημάτων στο Εθνικό Μετσόβιο Πολυτεχνείο με ενέπνευσε βαθύτατα και αποτέλεσε τον πιο καταλυτικό παράγοντα για να εμβαθύνω στους τομείς της Όρασης Υπολογιστών και της Μηχανικής Μάθησης.

Στην συνέχεια, θα ήθελα να ευχαριστήσω και τον κ. Χρήστο Σακαρίδη, μεταδιδακτορικό ερευνητή του ΕΤΗ, ο οποίος συνεπέβλεπε την εργασία μου. Άλλωστε, το πρόσφατο ερευνητικό έργο του αποτέλεσε οδηγό για εμένα, προκειμένου να ασχοληθώ με ένα τόσο ενδιαφέρον επιστημονικό πεδίο. Παρά τις πολλαπλές του υποχρεώσεις, στάθηκε αρωγός μου σε όλα τα επίπεδα, αφού από την πρώτη στιγμή ήταν διαθέσιμος να με καθοδηγήσει και να με προσανατολίσει σωστά με τις πολύτιμες συμβουλές του και την ένθερμη υποστήριξή του. Αισθάνομαι ιδιαίτερα τυχερός και ευγνώμων που συνεργάστηκα μαζί του στα πρώτα μου ερευνητικά βήματα.

Ένα μεγάλο ευχαριστώ οφείλω και στους γονείς μου, Νίκο και Κατερίνα για την κατανόηση και, φυσικά, την υποστήριξή τους καθόλη τη διάρκεια των σπουδών μου, αλλά και στους φίλους μου για τις ωραίες στιγμές που περάσαμε μαζί και για την επικοινωνιοδομητική ανταλλαγή απόψεων και προβληματισμών για επιστημονικά θέματα.

Σταμάτης Αλεξανδρόπουλος  
Οκτώβριος 2022



# Contents

<b>Contents</b>	<b>xi</b>
<b>List of Figures</b>	<b>xiii</b>
<b>List of Tables</b>	<b>xv</b>
<b>Acronyms</b>	<b>xix</b>
<b>Color Code</b>	<b>xxi</b>
<b>Εκτεταμένη Περίληψη στα Ελληνικά</b>	<b>1</b>
<b>1 Introduction</b>	<b>27</b>
1 Semantic Segmentation . . . . .	28
1.1 Definition and Applications . . . . .	28
1.2 Deep Neural Networks . . . . .	29
2 Outline . . . . .	29
<b>2 Theoretical Background</b>	<b>31</b>
1 Introduction to Deep Learning . . . . .	32
1.1 Convolutional Neural Networks (CNNs) . . . . .	32
1.2 Training the Neural Network . . . . .	40
1.3 Loss Function . . . . .	40
1.4 Performance Evaluation . . . . .	42
2 Introduction to Semantic Segmentation . . . . .	44
<b>3 Related Work</b>	<b>47</b>
1 Evolution of Semantic Segmentation . . . . .	48
1.1 Semantic Segmentation Before Deep Neural Networks . . . . .	48
1.2 Semantic Segmentation Using Deep Neural Networks . . . . .	48
2 Some popular state-of-the-art semantic segmentation models . . . . .	50
2.1 Based on Fully Convolutional Network . . . . .	50
2.2 Based on Dilation/Atrous convolution . . . . .	50
2.3 Based on Top-down/Bottom-up approach . . . . .	52
2.4 Based on receptive field enlargement and multi-scale context incorporation . . . . .	57
2.5 Based on Transformers . . . . .	60
2.6 Comparison . . . . .	64
<b>4 The Proposed Method: Offset Vector - Based Model</b>	<b>67</b>
1 Main Idea . . . . .	68
2 Seed Pixel Identification . . . . .	68
3 Confidence Loss . . . . .	71
4 Network Architecture . . . . .	71

<b>5</b>	<b>Experimental Results</b>	<b>73</b>
1	Dataset	74
1.1	Cityscapes	74
1.2	Adverse Conditions Dataset with Correspondences (ACDC)	74
2	Evaluation Metrics	75
3	Implementation Details	75
4	Qualitative Results	79
5	Baseline Experiments	84
6	Comparison with State of the Art	84
6.1	Cityscapes	84
6.2	ACDC	87
7	Ablation Study	89
8	Conclusions	89
<b>6</b>	<b>Conclusion and Future Work</b>	<b>91</b>
1	Conclusion	92
2	Future and Follow Up Works	92



# List of Figures

A .1	Παράδειγμα του αλγόριθμου σημασιολογικής κατάτμησης . . . . .	2
B .2	Η αρχιτεκτονική ενός Convolutional Neural Network (CNN) . . . . .	3
C .3	Παράδειγμα Σημασιολογικής Κατάτμησης. Από το [5] . . . . .	6
C .4	One-hot αναπαράσταση των labels. Από το [5] . . . . .	6
C .5	Τελικό Αποτέλεσμα. Από το [5] . . . . .	7
D .6	Απεικόνιση της δομής του High-Resolution Network (HRNet). Από το [6] . . . . .	8
D .7	Απεικονίζοντας πώς η διαδικασία συγχώνευσης συσσωρεύει την πληροφορία από υψηλές, μεσαίες και χαμηλές αναλύσεις από τα δεξιά προς τα αριστερά. Από το [6] . . . . .	9
D .8	Κεφαλές Αναπαράστασης . . . . .	10
E .9	HRNetV2 Βασισμένο στα Διανύσματα Μετατόπισης . . . . .	12
E .10	<b>Επισκόπηση της πλήρους μεθόδου.</b> Η έκδοση του HRNetV2 που βασίζεται σε διανύσματα μετατόπισης ( Σχ.Ε .9 ) , αποτελείται από δύο κεφαλές. Η πρώτη κεφαλή εξάγει logits ( $C$ ) σε επίπεδο pixel, ενώ η δεύτερη εξάγει ένα πυκνό πεδίο διανυσματών μετατόπισης ( $\mathbf{o}$ ) για τον καθορισμό των θέσεων των seed pixels μαζί με ένα χάρτη εμπιστοσύνης ( $F$ ). Στη συνέχεια, οι προβλέψεις των seed pixels χρησιμοποιούνται για να προβλέψουν κλάσεις σε κάθε pixel. Η πρόβλεψη που προκύπτει ( $S_s$ ) συγχωνεύεται προσαρμοστικά με την αρχική πρόβλεψη $S_i$ χρησιμοποιώντας τον χάρτη εμπιστοσύνης ( $F$ ), προκειμένου να υπολογιστεί η τελική πρόβλεψη $S_f$ . . . . .	13
F .11	Ποιοτικά Αποτελέσματα στο Cityscapes: Πρώτο Παράδειγμα . . . . .	16
F .12	Ποιοτικά Αποτελέσματα στο Cityscapes: Δεύτερο Παράδειγμα . . . . .	17
F .13	Ποιοτικά Αποτελέσματα στο ACDC: Πρώτο παράδειγμα . . . . .	18
F .14	Ποιοτικά Αποτελέσματα στο ACDC: Δεύτερο παράδειγμα . . . . .	18
F .15	<b>Ποιοτικά αποτελέσματα επιλεγμένων παραδειγμάτων στο Cityscapes.</b> Απο τα αριστερά προς τα δεξιά: εικόνα εισόδου, αρχικό HRNet και δικό μας μοντέλο . . . . .	21
F .16	<b>Ποιοτικά αποτελέσματα επιλεγμένων παραδειγμάτων στο ACDC.</b> Απο τα αριστερά προς τα δεξιά: εικόνα εισόδου, αρχικό HRNet και δικό μας μοντέλο . . . . .	23
1 .1	Example of a semantic image segmentation algorithm . . . . .	28
1 .2	An example of a CNN architecture . . . . .	29
1 .1	An example of a CNN architecture. From [36] . . . . .	32
1 .2	Kernel types . . . . .	33
1 .3	Illustration of the convolution operation . . . . .	33
1 .4	Types of pooling. From [38] . . . . .	34
1 .5	Example of a $3 \times 3$ transpose convolution. From [39] . . . . .	35
1 .6	Example of a $2 \times 2$ unpooling operation. From [40] . . . . .	35
1 .7	Batch Normalization. From [41] . . . . .	36
1 .8	Dropout Visualization . . . . .	37
1 .9	The four red dots depict the data points and the green dot is the point at which we want to interpolate. From [46]. . . . .	38
2 .1	Pixel-level Semantic Segmentation Segmentation. From [5] . . . . .	45
2 .2	One-hot representations of the class labels. From [5] . . . . .	45
2 .3	Final Result. From [5] . . . . .	45
1 .1	Example of thresholding using Otsu's method. From [61] . . . . .	49

1 .2	The structure of recovering high resolution from low resolution. (a) A low-resolution representation learning subnetwork (such as VGGNet [68], ResNet [69]), which is formed by connecting high-to-low convolutions in series. (b) A high-resolution representation recovering subnetwork, which is formed by connecting low-to-high convolutions in series. Representative examples include SegNet [63], DeconvNet [40], U-Net [64], encoder-decoder [67]. From [6]	49
2 .1	Illustration of different Fully Convolutional Network (FCN) architectures. (a) Illustration of FCN-32 architecture. (b) Illustration of FCN-16 architecture. (c) Illustration of FCN-8 architecture	51
2 .2	(a)1-DilatedNet with receptive field $3 \times 3$ , (b) 2-DilatedNet with receptive field $7 \times 7$ and (c)4-DilatedNet with receptive field $15 \times 15$ . From [78]	52
2 .3	Atrous convolution with kernel size $3 \times 3$ and different rates. Standard convolution corresponds to atrous convolution with rate = 1. Employing large value of atrous rate enlarges the model's field-of-view, enabling object encoding at multiple scales. From [80]	52
2 .4	Overall Deconvolutional Network (DeconvNet) architecture. From [81]	53
2 .5	U-Net architecture. From [84]	54
2 .6	SegNet architecture. From [85]	55
2 .7	Fully Convolutional DenseNet (FC-DenseNet) architecture. From [89]	56
2 .8	Parallel modules with Atrous Special Pooling Pyramid (ASPP), augmented with image-level features. From [80]	57
2 .9	Overview of proposed Pyramid Scene Parsing Network (PSPNet). From [12]	58
2 .10	Overview of proposed PSPNet. From [12]	58
2 .11	Overview of proposed SwiftNet. The proposed multi-scale architecture has shared encoders and pyramidal fusion. From [95]	59
2 .12	Illustrating the HRNet architecture. From [6]	59
2 .13	Illustrating how the fusion module aggregates the information for high, medium and low resolutions from left to right, respectively. From [6]	60
2 .14	Representation heads	60
2 .15	Illustrating the pipeline of OCR. From [9]	61
2 .16	Segmentation transformer. From [9]	62
2 .17	Illustrating the Vision Transformer (ViT) architecture. From [96]	63
2 .18	Illustrating the ViT architecture. From [96].	63
2 .19	Illustrating the Lawin Transformer architecture. . From [97].	63
2 .20	Overall architecture of Vision Transformer Adapter (ViT-Adapter). (a) The ViT, whose encoder layers are divided into $N$ equal blocks for feature interaction; (b) ViT-Adapter, which contains three key components; (c) The spatial prior module, which is used to model local spatial contexts from the input image; (d) The spatial feature injector for incorporating image prior into the ViT; (e) The multi-scale feature extractor for reconstructing fine-grained multi-scale features from the single-scale features of ViT. From [34].	64
1 .1	Offset vector-based HRNetV2	69
1 .2	<b>An overview of our full method.</b> Offset vector-based HRNetV2, whose architecture is shown in Fig. 1 .1, consists of two output heads. The first head outputs pixel-level Logits (C), while the second head outputs a dense offset vector field (o) identifying positions of seed pixels along with a confidence map (F). Then, the coefficients of seed pixels are used to predict classes at each position. The resulting prediction ( $S_s$ ) is adaptively fused with the initial prediction ( $S_i$ ) using the confidence map $F$ to compute the final prediction $S_f$	70
1 .1	Number of finely annotated pixels per class in ACDC. From [11]	74
4 .1	The optical flow field color-coding. Smaller vectors are lighter and color represents the direction. From [104]	79
4 .2	Viridis color map. The higher the confidence is, the lighter the map is. From [105]	79
4 .3	Qualitative results on Cityscapes: Example 1	81
4 .4	Qualitative results on Cityscapes: Example 2	81
4 .5	Qualitative results on Cityscapes: Example 3	82
4 .6	Qualitative results on ACDC: Example 1	82
4 .7	Qualitative results on ACDC: Example 2	83

---

4 .8	Qualitative results on ACDC: Example 3 . . . . .	83
6 .1	<b>Qualitative results of selected examples on Cityscapes.</b> From left to right: image, initial HRNet and ours. . . . .	86
6 .2	<b>Qualitative results of selected examples on ACDC.</b> From left to right: image, initial HRNet and ours. . . . .	88



# List of Tables

1	Αποτελέσματα αναπαραγωγής του μοντέλου μας στο Test Set . . . . .	19
2	Αποτελέσματα στο Cityscapes val test (multi-scale και flipping). Τα GFLOPs υπολογίζονται με βάση την εικόνα εισόδου ( $1024 \times 2048$ ). Μετρώνται μόνο Συνελικτικά και Γραμμικά επίπεδα μόνο. . . . .	19
3	Αποτελέσματα στο Cityscapes test set. Τα αποτελέσματα μας είναι ανώτερα όσον αφορά τις 4 μετρικές αξιολόγησης.D-ResNet-101 = Dilated-ResNet-101. Συγκρίνουμε τη μέθοδο μας με προηγούμενες SOTA μεθόδους, όπως στο [6] . . . . .	20
4	Ανα κλάση αποτελέσματα στο Cityscapes test set . . . . .	20
5	Αποτελέσματα στο ACDC val test (multi-scale και flipping). Τα GFLOPs υπολογίζονται με βάση την εικόνα εισόδου ( $1080 \times 1920$ ). Μετρώνται μόνο Συνελικτικά και Γραμμικά επίπεδα μόνο. . . . .	22
6	Αποτελέσματα στο ACDC test set. Τα αποτελέσματα μας είναι ανώτερα όσον αφορά το Mean Intersection over Union (mIoU). Συγκρίνουμε τη μέθοδο μας με προηγούμενες SOTA μεθόδους, όπως στο [11] . . . . .	22
7	Ανα κλάση αποτελέσματα στο ACDC test set . . . . .	23
3.1	Comparison results of different semantic segmentation models in terms of mIoU. . . . .	65
5.1	The architecture of the 1 <sup>st</sup> Version Offset Vector Based HRNet (main body). . . . .	76
5.2	The architecture of the 2 <sup>nd</sup> Version Offset Vector Based HRNet (main body). . . . .	77
5.3	The architecture of the 3 <sup>rd</sup> Version Offset Vector Based HRNet (main body). . . . .	78
5.4	Test Set Reproduction Results . . . . .	84
5.5	Semantic segmentation results on Cityscapes val test (multi-scale and flipping). The GFLOPs is calculated on the input size $1024 \times 2048$ . They are calculated for Convolution and Linear Layers only. . . . .	84
5.6	Semantic segmentation results on Cityscapes test set. Our results are superior in terms of the four evaluation metrics.D-ResNet-101 = Dilated-ResNet-101. We compare our method against SOTA methods as in [6] . . . . .	85
5.7	Per Class Results on Cityscapes test set . . . . .	85
5.8	Semantic segmentation results on ACDC val set (multi-scale and flipping). The GFLOPs is calculated on the input size $1080 \times 1920$ . They are calculated for Convolution and Linear Layers only. . . . .	87
5.9	Semantic segmentation results on ACDC test set. Our results are superior in terms of the mIoU metrics. We compare our method against SOTA methods as in [11] . . . . .	87
5.10	Per Class Results on ACDC test set . . . . .	88
5.11	<b>Ablation study of components of our method.</b> "Fr" : Freezed main body's and initial head's weights, "IHW": Initial Head Weights, "Ref": cascaded refinement of offsets, "OHW": Offset Head Weights, "V": Version, "OV": Offset Vector, "OHEM": Online Hard Example Mining (OHEM) Cross Entropy, I: Imagenet, C:Cityscapes , A: ACDC . . . . .	89



# Acronyms

**DCNN** Deep Convolutional Neural Network

**HRNet** High-Resolution Network

**AI** Artificial Intelligence

**ANN** Artificial Neural Network

**DNN** Deep Neural Network

**CNN** Convolutional Neural Network

**CRF** Conditional Random Field

**FCN** Fully Convolutional Network

**DeconvNet** Deconvolutional Network

**ReLU** Rectified Linear Unit

**VGG** Visual Geometry Group

**DenseNet** Densely Connected Convolutional Network

**ResNet** Residual Neural Network

**FC-DenseNet** Fully Convolutional DenseNet

**ASPP** Atrous Special Pooling Pyramid

**PSPNet** Pyramid Scene Parsing Network

**GSCNN** Gated-Shape CNN

**GCL** Gated Convolutional Layers

**GT** Ground-Truth

**mIoU** Mean Intersection over Union

**IoU** Intersection over Union

**ACDC** Adverse Conditions Dataset with Correspondences

**BN** Batch Normalization

**EMA** Exponential Moving Average

**MSE** Mean Square Error

**MAE** Mean Absolute Error

**SELU** Scaled Exponential Linear Unit

**FwIoU** Frequency-weighted IoU

**AUC** Area Under the Curve

**ROC** Receiver-Operator Characteristic curve

**PA** Pixel Accuracy

**PRC** Precision-Recall Curve

**mPA** Mean pixel accuracy

**NNI** Nearest Neighbour Interpolation

**iIoU** instance-level Intersection over Union

**OHEM** Online Hard Example Mining

**ViT** Vision Transformer

**ViT-Adapter** Vision Transformer Adapter

**OCR** Object-Contextual Representations



# Color Code

The "color code" that will be utilized is shown below.

## **Theorem 0 .1: title**

Theorem

## **Definition 0 .2: title**

Definition

## **Lemma 0 .3: title**

Lemma

## **Example 0 .4: title**

Example



# Εκτεταμένη Περίληψη στα Ελληνικά

## Σημασιολογική Κατάτμηση

### Περιγραφή του Προβλήματος

Η Σημασιολογική Κατάτμηση (Semantic Segmentation) είναι το πρόβλημα ανάθεσης μιας κλάσης σε κάθε pixel μιας εικόνας. Το πρόβλημα αυτό αναφέρεται συχνά και ως πυκνή πρόβλεψη, δεδομένου ότι η κλάση προβλέπεται για κάθε pixel της εικόνας.

### Ορισμός A .5: Σημασιολογική Κατάτμηση

Η Σημασιολογική Κατάτμηση απαιτεί την εκμάθηση μιας πυκνής απεικόνισης - συνάρτησης  $f_\theta : I(u, v) \rightarrow S(u, v)$  όπου

- $I$  είναι η εικόνα εισόδου με χωρικές διαστάσεις  $H \times W$
- $S$  είναι ο αντίστοιχος προβλεπόμενος χάρτης εξόδου, ο οποίος έχει τις ίδιες διαστάσεις με την εικόνα εισόδου
- $(u, v)$  είναι οι συντεταγμένες κάθε pixel της εικόνας
- $\theta$  είναι παράμετροι της συνάρτησης  $f$

Στην εκδοχή του προβλήματος αυτού με επίβλεψη, ένας σωστά (ground-truth) σημασιολογικά τμηματοποιημένος χάρτης  $H$  είναι διαθέσιμος την ώρα της εκπαίδευσης για κάθε εικόνα  $I$ . Στο διάστημα αυτό, οι παράμετροι  $\theta$  βελτιστοποιούνται, έτσι ώστε η συνάρτηση  $f_\theta$  να ελαχιστοποιήσει τη διαφορά ανάμεσα στην προβλεπόμενη εικόνα και την ground-truth εικόνα πάνω στο σύνολο δεδομένων προς εκπαίδευση  $T$ . Με άλλα λόγια, αυτό μπορεί να διατυπωθεί ως εξής:

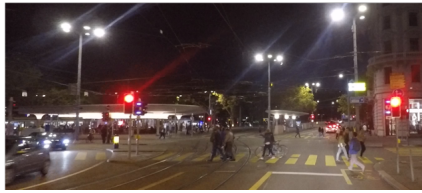
$$\min_{\theta} \sum_{(I, H) \in T} \mathcal{L}(f_\theta(I), H) \quad (\text{A .1})$$

όπου  $\mathcal{L}$  είναι μια συνάρτηση απωλειών που "τιμωρεί" τις αποκλίσεις μεταξύ της πρόβλεψης και της ground-truth εικόνας.

Πολλές εφαρμογές, όπως η ανάλυση ιατρικών εικόνων, η ρομποτική, η υλοποίηση συστημάτων παρακολούθησης, η αυτόνομη οδήγηση και άλλες πολλές, απαιτούν σημασιολογική πληροφορία από τις διάφορες εικόνες σε επίπεδο pixel. Σχετικά με την αυτόνομη οδήγηση, ένα παράδειγμα φαίνεται στο Σχήμα A .1. Στο σχήμα αυτό βλέπουμε ένα παράδειγμα μιας εικόνας εισόδου A .1a και της αντίστοιχης σημασιολογικά τμηματοποιημένης εικόνας A .1b. Επιπλέον, στο Σχήμα A .1c φαίνεται και η αντίστοιχη ground-truth εικόνα. Οι ετικέτες κάθε κλάσης μπορεί να είναι ο άνθρωπος, το αμάξι, ο δρόμος, ή οποιοδήποτε άλλο βασικό μοτίβο που βρίσκεται στην εικόνα. Ο αλγόριθμος αυτός είναι από τους λίγους που μας βοηθούν να αναλύσουμε το περιεχόμενο ενός περιβάλλοντος με το οποίο είμαστε εξοικειωμένοι. Σαν αποτέλεσμα, η σημασιολογική κατάτμηση χρησιμοποιείται ευρέως στα αυτόνομα οχήματα, όπου το περιεχόμενο του γύρω περιβάλλοντος είναι ζωτικής σημασίας.

## Προκλήσεις

Το πρόβλημα της Σημασιολογικής Κατάτμησης είναι πλέον από τα πιο φλέγοντα ερευνητικά θέματα. Αν και οι πρόσφατες τεχνικές έχουν παρουσιάσει πολλά υποσχόμενα αποτελέσματα, αδυνατούν να ξεπεράσουν την ανθρώπινη επίδοση. Ο άνθρωπος μπορεί εύκολα να ξεχωρίσει τα διάφορα αντικείμενα, παρόλο που αυτά μπορεί



(a) Εικόνα Εισόδου



(b) Ground Truth Εικόνα



(c) Αποτέλεσμα

Σχήμα Α .1: Παράδειγμα του αλγόριθμου σημασιολογικής κατάτμησης

να διαφέρουν σε όψεις, κλίμακα, φωτισμό ή διάταξη. Τέλος, ακόμα και αν τα αντικείμενα είναι μερικώς ευδιάκριτα, αυτά μπορούν να ταυτοποιηθούν.

Προκειμένου να ξεπεράσουμε αυτόματα αυτά τα εμπόδια με την χρήση υπολογιστών, οι state-of-the-art μέθοδοι σημασιολογικής κατάτμησης εξαρτώνται από τεχνικές βαθιάς μάθησης προκειμένου να κατανοήσουν τις πολλαπλές αναπαραστάσεις των αντικειμένων από τις παρεχόμενες εικόνες. Προκειμένου να επιτευχθεί η ακρίβεια των σύγχρονων προσεγγίσεων, είναι απαραίτητη η επισημείωση των εικόνων σε επίπεδο pixel. Δυστυχώς, οι εικόνες αυτές είναι πολλές φορές περιορισμένες για διάφορες εφαρμογές ή απλώς δεν είναι διαθέσιμες. Επιπλέον, αν γίνει η εκπαίδευση του μοντέλου πάνω σε ένα πεπερασμένο σύνολο επισημειωμένων εικόνων, αυτό το μοντέλο αφενός μεν μπορεί από τη μια να παράγει ικανοποιητικά αποτελέσματα σε εικόνες που μοιάζουν με αυτές του συνόλου δεδομένων εκπαίδευσης, αλλά από την άλλη, κανείς δεν εγγυάται τη γενίκευση του σε επιπρόθετες εικόνες. Αυτό το πρόβλημα είναι γνωστό ως *overfitting*.

## Βαθιά Νευρωνικά Δίκτυα

Η βαθιά μάθηση είναι μια τεχνική που χρησιμοποιείται για να χτίσει συστήματα Τεχνητής Νοημοσύνης. Βασίζεται στην ιδέα των Τεχνητών Νευρωνικών Δικτύων (Artificial Neural Networks), τα οποία επεξεργάζονται μεγάλους όγκους δεδομένων μέσω πολυάριθμων επιπέδων νευρώνων για την εκτέλεση σύνθετων αναλύσεων.

### Βαθιά Συνελικτικά Νευρωνικά Δίκτυα

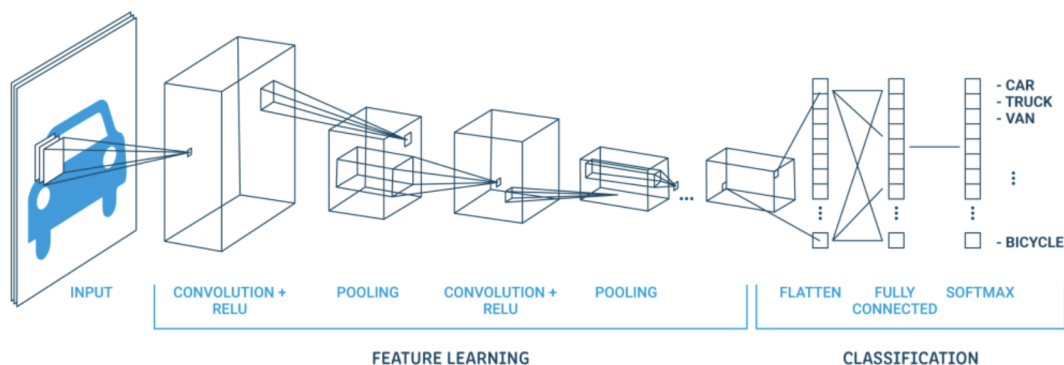
Υπάρχει μεγάλη ποικιλία Βαθιών Συνελικτικών Νευρωνικών Δικτύων (DCNNs). Αυτά χρησιμοποιούνται ευρέως σε συστήματα αναγνώρισης εικόνας και βίντεο. Τα παραδοσιακά τεχνητά νευρωνικά δίκτυα έχουν εξελιχθεί σε βαθιά συνελικτικά δίκτυα, τα οποία χρησιμοποιούν ένα τρισδιάστατο νευρωνικό πρότυπο εμπνευσμένο από τον οπτικό εγκέφαλο των ζώων. Τα τελευταία χρησιμοποιούνται σε προβλήματα αναγνώρισης αντικειμένων, ταξινόμησης εικόνων, σημασιολογικής κατάτμησης, αλλά και μερικές φορές σε προβλήματα επεξεργασίας φυσικής γλώσσας.

Η δύναμη των DCNNs έγκειται στη διαστρωμάτωσή τους. Ένα DCNN επεξεργάζεται το κόκκινο, το πράσινο και το μπλε κανάλι μιας εικόνας ταυτόχρονα χρησιμοποιώντας ένα τρισδιάστατο νευρωνικό δίκτυο. Σε σύγκριση με τα κλασικά feed forward νευρωνικά δίκτυα, αυτό μειώνει σημαντικά τον αριθμό των νευρώνων που απαιτούνται για την ανάλυση μιας εικόνας.

Τα DCNNs λαμβάνουν σαν είσοδο εικόνες και τις χρησιμοποιούν για να εκπαιδεύσουν ένα ταξινομητή. Αντί για πολλαπλασιασμό πινάκων, το δίκτυο χρησιμοποιεί μια εναλλακτική μαθηματική διαδικασία, γνωστή ως "συνέλιξη".

Η αρχιτεκτονική ενός συνελικτικού νευρωνικού δικτύου (CNN) γενικά αποτελείται από 4 επίπεδα: ένα συνελικτικό επίπεδο (convolutional layer), ένα επίπεδο συσσώρευσης (pooling layer), ένα επίπεδο ενεργοποίησης

(activation layer) και ένα πλήρως συνδεδεμένο επίπεδο (fully connected layer) [1]. Ένα παράδειγμα της αρχιτεκτονικής αυτής φαίνεται στο Σχήμα Β.2.



Σχήμα Β.2: Η αρχιτεκτονική ενός CNN

## Θεωρητικό Υπόβαθρο

### Εισαγωγή στη Βαθιά Μάθηση

#### Συνελικτικά Νευρωνικά Δίκτυα (CNNs)

Τα Συνελικτικά Νευρωνικά Δίκτυα (CNNs) χρησιμοποιούνται ευρέως στην Όραση Υπολογιστών. Κατασκευάζονται από νευρώνες, των οποίων τόσο τα βάρη όσο και το bias μαθαίνονται με παρόμοιο τρόπο με αυτό των κλασικών νευρωνικών δικτύων. Κάθε νευρώνας δέχεται ορισμένες εισόδους, εκτελεί ένα γινόμενο κάποιων όρων και, προαιρετικά, εισάγει μια μη γραμμικότητα στο αποτέλεσμα. Τελικά, το δίκτυο περιλαμβάνει μια συνάρτηση απωλειών (loss function) στο τελευταίο (πλήρως συνδεδεμένο) επίπεδο (π.χ. SVM/Softmax) [2].

Τα CNNs ξεπερνούν άλλες αρχιτεκτονικές νευρωνικών δικτύων σε προβλήματα επεξεργασίας εικόνας. Αυτό σχετίζεται με την κατάλληλη εκμετάλλευση των χωρικών συσχετίσεων στις εικόνες και το πλεονέκτημα του διαμοιρασμού των βαρών (weight sharing) κατά τη διάρκεια της εκπαίδευσης.

Τα CNNs συνήθως αποτελούνται από πολλά επίπεδα, καθένα από τα οποία έχει διαφορετικό σκοπό. Χαμηλού επιπέδου χαρακτηριστικά, όπως χρώματα, ακμές και σχήματα ανιχνεύονται από το πρώτα επίπεδα.

Τα επίπεδα αρχίζουν να μαθαίνουν σύνθετα χαρακτηριστικά, καθώς το μοντέλο εξελίσσεται και το τελευταίο επίπεδο παράγει τις προβλέψεις. Έτσι, το δίκτυο έχει μια πιο ολοκληρωμένη αντίληψη των εικόνων του συνόλου δεδομένων. Στην παρούσα ενότητα, θα ασχοληθούμε κυρίως με τις λειτουργίες ορισμένων επιπέδων των CNNs, που χρησιμοποιούνται κυρίως στην επεξεργασία εικόνας.

Μια κλασική αρχιτεκτονική ενός συνελικτικού δικτύου συνήθως αποτελείται από συνελικτικό επίπεδο, ένα επίπεδο συσσώρευσης (pooling layer), ένα επίπεδο ενεργοποίησης (activation layer) και ένα πλήρως συνδεδεμένο επίπεδο (fully connected layer). Την ίδια στιγμή συμβαίνουν διεργασίες, όπως το upsampling, η συγχώνευση (concatenation), το dropout και η διγραμμική παρεμβολή ή παρεμβολή κοντινότερου γείτονα.

#### Εκπαίδευση του Νευρωνικού Δικτύου

Με βάση τον ορισμό της αρχιτεκτονικής του νευρωνικού δικτύου, τα βάρη του νευρωνικού μαθαίνονται μέσω της διαδικασίας εκπαίδευσης. Αυτό γίνεται μέσω επίβλεψης για την πλειοψηφία των εφαρμογών βαθιάς μάθησης, όπως είναι η ανάλυση ιατρικών εικόνων ή η αυτόνομη οδήγηση. Σε αυτές τις εφαρμογές δίνονται σαν είσοδο στο σύστημα πολλά δείγματα με τα αντίστοιχα labels που μας ενδιαφέρουν. Για παράδειγμα, αν μια 2D ακτινογραφία στήθους δοθεί σαν είσοδος στο σύστημα, τότε το label μπορεί να είναι μια δυαδική ταξινόμηση (άρρωστο/υγιές άτομο). Για κάθε τέτοιο ζευγάρι εικόνας-label, η εικόνα εισέρχεται στο δίκτυο, το οποίο με τη σειρά του εξάγει μια πρόβλεψη που βασίζεται στους υπολογισμούς που έχει κάνει σε κάθε επίπεδο. Στην συνέχεια, χρησιμοποιείται μια συνάρτηση απωλειών  $L$ , η οποία μετρά πόσο παρόμοια είναι η πρόβλεψη με την ετικέτα (label). Στη συνέχεια,

αφού υπολογιστεί το σφάλμα που θα προκύψει από αυτή τη σύγκριση, ένας βελτιστοποιητής (optimizer) χρησιμοποιεί μια μέθοδο που καλείται backpropagation, προκειμένου να τροποποιήσει τα βάρη του δικτύου. Με αυτόν τον τρόπο, θα μειωθεί το σφάλμα την επόμενη φορά που το δίκτυο θα δει την ίδια περίπτωση. Τα βάρη πρέπει ιδανικά να έχουν προσαρμοστεί όταν το δίκτυο έχει εκπαιδευτεί για ένα batch, το οποίο είναι ένα γκρουπ από αρκετές εικόνες. Ο πιο συνηθισμένος τύπος βελτιστοποιητή είναι το Stochastic Gradient Descent. Εκτός από αυτόν υπάρχουν και άλλοι βελτιστοποιητές, όπως ο Adam ή ο RmsProp. Η διαδικασία της εκπαίδευσης επαναλαμβάνεται μέχρι η απώλεια (loss) σε ένα batch στα δεδομένα προς αξιολόγηση να ελαχιστοποιηθεί. Στην πράξη, αυτό μπορεί να πάρει λεπτά για μικρά βασικά dataset και μέρες ή ακόμα εβδομάδες για μεγαλύτερα δεδομένα. Αυτό εξαρτάται, βέβαια, και από τους υπολογιστικούς πόρους που είναι διαθέσιμοι στον χρήστη. Το overfitting μπορεί να συμβεί όταν ένα CNN εκπαιδεύεται σε ένα μικρό σύνολο δεδομένων για μεγάλο χρονικό διάστημα. Παρόλα αυτά, αυτό μπορεί να λυθεί με το να ελέγξουμε την απόδοση του μοντέλου σε άγνωστα δεδομένα που δεν έχει ξαναδεί.

## Ενίσχυση Δεδομένων (Data Augmentation)

Πολλές φορές λόγω της έλλειψης αρκετών δεδομένων κατά τη διαδικασία εκπαίδευσης, χρησιμοποιείται μια τεχνική που ονομάζεται Ενίσχυση Δεδομένων (Data Augmentation). Μέσω αυτής της διαδικασίας τροποποιούνται οι ήδη υπάρχουσες εικόνες με διάφορους τρόπους, όπως περιστροφή, αλλαγή του μεγέθους, της αντίθεσης (contrast) ή της έκθεσης στον ήλιο (exposure) κ.α. Έτσι εμπλουτίζεται το σύνολο δεδομένων και αποφεύγεται το overfitting.

## Συνάρτηση Απωλειών (Loss Function)

Η συνάρτηση απωλειών υπολογίζει τη διαφορά μεταξύ της σωστής εξόδου του αλγορίθμου και της προβλεπόμενης. Είναι μια μέθοδος για να αξιολογήσει πως ο αλγόριθμος μοντελοποιεί τα δεδομένα. Μπορεί να χωριστεί σε δύο κατηγορίες. Η πρώτη αφορά προβλήματα ταξινόμησης και η δεύτερη προβλήματα παλινδρόμησης (regression). Παρακάτω, θα αναφερθούμε ονομαστικά σε μερικά επιμέρους losses των κατηγοριών αυτών:

### Ταξινόμηση

- Cross-Entropy Loss
- Log-loss
- Exponential Loss
- Hinge Loss
- Kullback Leibler Divergence Loss

### Παλινδρόμηση

- Μέσο Τετραγωνικό Σφάλμα (MSE)
- Μέσο Απόλυτο Σφάλμα (MAE)

Από τα παραπάνω θα χρειαστεί να αναλύσουμε το Cross-Entropy Loss μιας και θα το χρησιμοποιήσουμε στην παρούσα διπλωματική. Συγκεκριμένα, το Loss αυτό μετρά την διαφορά ανάμεσα σε δύο συναρτήσεις κατανομής πιθανότητας. Το cross-entropy της κατανομής  $q$  σε σχέση με την κατανομή  $p$  πάνω σε ένα δεδομένο σύνολο ορίζεται ως εξής:

$$H(p, q) = -E_p[\log q] \quad (\text{C.1})$$

όπου το  $E_p[\cdot]$  είναι η αναμενόμενη τιμή με βάση την κατανομή  $p$ . Για διακριτές κατανομές  $p$  και  $q$  πάνω στο ίδιο σύνολο  $X$ , η εξίσωση C.1 συνεπάγεται ότι:

$$H(p, q) = - \sum_{x \in X} p(x) \log q(x) \quad (\text{C.2})$$

## Αξιολόγηση της Επίδοσης

Όταν αναλύουμε την απόδοση του προβλήματος της σημασιολογικής κατάτμησης, χρειάζεται να λάβουμε υπ' όψην μας δύο κριτήρια:

1. **την ορθότητα (accuracy):** μετρά πόσο επιτυχής είναι μια μέθοδος. Στη σημασιολογική κατάτμηση πρέπει, αφενός, να ελεγχθεί η απόδοση της ταξινόμησης κάθε pixel σε κάθε κατηγορία με βάση τα αντίστοιχα labels και, αφετέρου, η απόδοση της τοπικότητας, δηλαδή το κατά πόσο σωστά περικλείει το σωστό σέτ από pixels το αντίστοιχο αντικείμενο. Μια από τις πιο κοινές μετρικές που χρησιμοποιούνται στο πρόβλημα αυτό είναι το Intersection over Union (IoU). Αυτή τη μετρική χρησιμοποιούμε και εμείς στην διπλωματική αυτή.

- Intersection over Union (IoU)

Η μετρική αυτή είναι η τομή σε επίπεδο pixel των αποτελεσμάτων που θα εξάγει το μοντέλο με τις ground truth εικόνες προς την ένωση τους. Μετρά τόσο την ομοιότητα, όσο και την ποικιλομορφία του συνόλου δειγμάτων. Ορίζεται ως εξής:

$$IoU = \frac{\sum_{j=1}^k n_{jj}}{\sum_{j=1}^k (n_{ji} + n_{ij} + n_{jj})}, i \neq j \quad (C.3)$$

όπου  $n_{jj}$  είναι ο συνολικός αριθμός True Positives,  $n_{ij}$  είναι ο συνολικός αριθμός False Positives και  $n_{ji}$  ο συνολικός αριθμός False Negatives για την κλάση  $j$ .

Επίσης, αξίζει να αναφέρουμε ότι συχνά χρησιμοποιείται ο μέσος όρος του IoU (mIoU):

$$mIoU = \frac{1}{k} \sum_{j=1}^k \frac{n_{jj}}{n_{ji} + n_{ij} + n_{jj}}, i \neq j \quad (C.4)$$

2. **την υπολογιστική πολυπλοκότητα:** απαιτήσεις σε ταχύτητα και μνήμη.

## Εισαγωγή στη Σημασιολογική Κατάτμηση

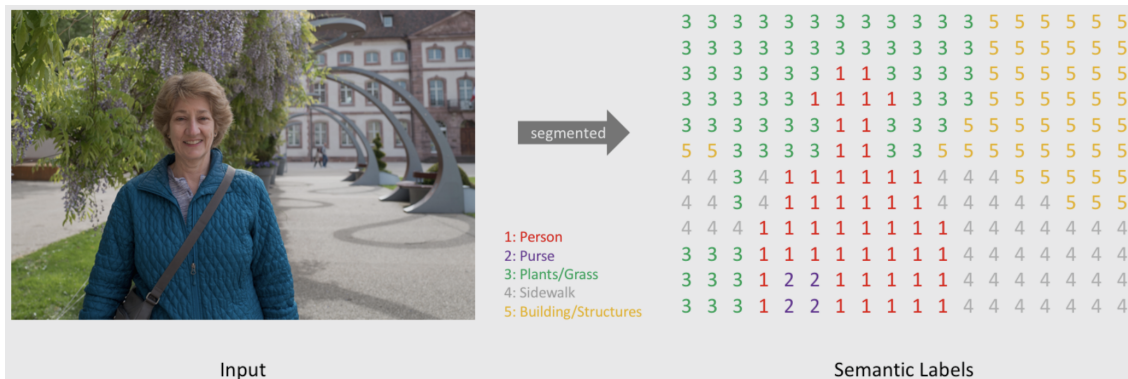
Υπάρχουν διάφορες μέθοδοι για να αναλύσει κανείς και να καταλάβει μια εικόνα. Αυτές μπορούν να χωριστούν σε τρεις κατηγορίες:

- **Ταξινόμηση Εικόνων:** Συνήθως έχει να ταξινομήσει ένα βασικό αντικείμενο σε μια εικόνα.
- **Ανίχνευση Αντικειμένων:** Εντοπίζει τα μπροστινά αντικείμενα που μας ενδιαφέρουν μέσα σε μια εικόνα. Αυτό συνεπάγεται την εύρεση ενός κατάλληλου πλαισίου οριοθέτησης (bounding box), το οποίο θα καλύπτει ολόκληρο το αντικείμενο και, παράλληλα, θα το κατηγοριοποιεί. Στην παρούσα διπλωματική δεν θα ασχοληθούμε με το πρόβλημα ανίχνευσης αντικειμένων.
- **Σημασιολογική Κατάτμηση:** Στόχος της είναι η ταξινόμηση μεμονομένων pixel σε κλάσεις και ο καθορισμός του σχήματος των κλάσεων στις οποίες ανήκουν όλα τα αντικείμενα. Η κατάτμηση συνήθως είναι πολύπλοκη, λόγω του ότι χρειάζεται να ταξινομήσουμε όλα τα pixels σε μια εικόνα, συμπεριλαμβανομένου τόσο του background όσο και του foreground. Το πλεονέκτημα της χρήσης τμημάτων έναντι των πλαισίων οριοθέτησης είναι ότι δύο τμήματα δεν επικαλύπτονται ποτέ, αλλά τα πλαίσια οριοθέτησης τείνουν να αλληλεπικαλύπτονται για την αναπαράσταση των ίδιων αντικειμένων, ειδικά όταν έχουμε πολλά πυκνά στοιβαγμένα ή ακανόνιστου σχήματος αντικείμενα.

Η κατάτμηση ή η ανίχνευση αντικειμένων είναι συχνά το αρχικό στάδιο, σε ένα ευφυές σύστημα που χρησιμοποιεί την όραση για τη λήψη κρίσιμων αποφάσεων. Για παράδειγμα, στην κατασκευή ενός αυτόνομου οχήματος, το σύστημα οδήγησης θα συλλέγει πρώτα φωτογραφίες του περιβάλλοντος και στη συνέχεια θα πρέπει να είναι σε θέση να κατανοήσει γρήγορα τη σκηνή και να τμηματοποιεί ή να αναγνωρίζει πεζούς και άλλα αντικείμενα. Η αποτελεσματικότητα της τμηματοποίησης αυτών των εικόνων καθορίζει τις επακόλουθες αποφάσεις του αυτόνομου συστήματος, όπως η οδήγηση, η στροφή ή η διενέργεια ενός ελιγμού για την αποφυγή εμποδίου.

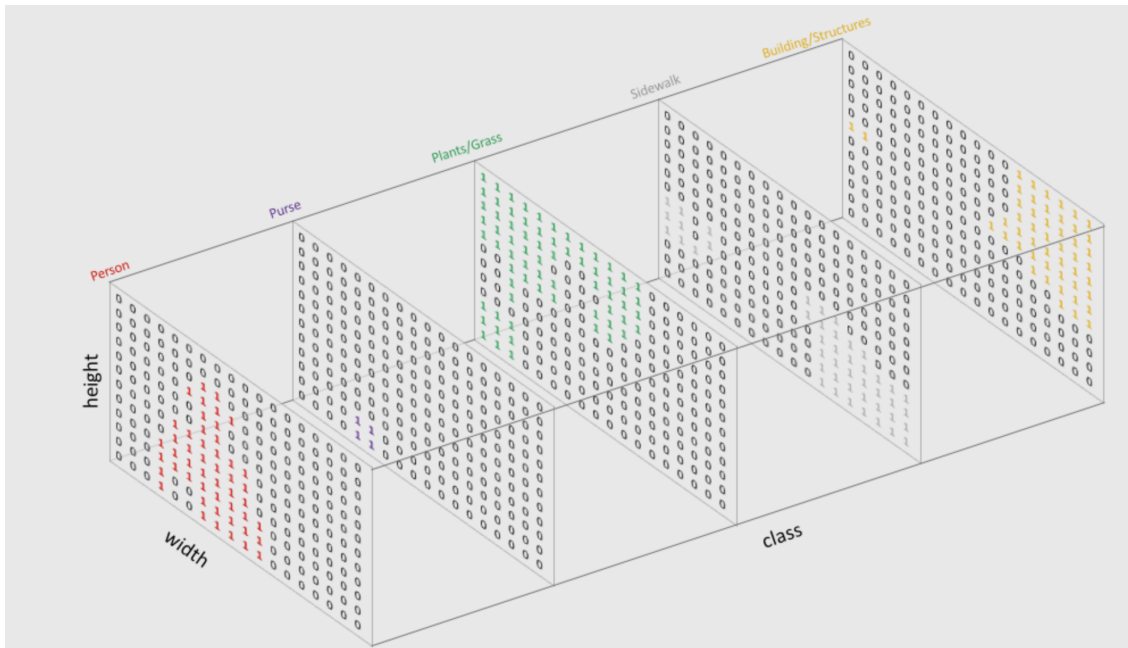
Άλλες εφαρμογές κατάτμησης στην Όραση Υπολογιστών περιλαμβάνουν ρομποτική ([3]), ανάλυση ιατρικών εικόνων ([4]) και συστήματα παρακολούθησης.

Σε αυτή τη διπλωματική εργασία, χρησιμοποιείται μια έγχρωμη RGB εικόνα ( $H \times W \times 3$ ) σαν είσοδο στο μοντέλο και το αποτέλεσμα είναι μια σημασιολογικά τμηματοποιημένη εικόνα, με κάθε pixel της να περιέχει μια ετικέτα κλάσης, υπο την μορφή μιας ακέραιας μεταβλητής. Αυτό φαίνεται στο σχήμα C.3.



Σχήμα C.3: Παράδειγμα Σημασιολογικής Κατάτμησης. Από το [5]

Κάθε μια απο τις πιθανές κλάσεις έχει το δικό της κανάλι, το οποίο περιέχει one-hot αναπαράστασεις των ετικετών. Μια τέτοια πρόβλεψη μπορεί να μετατραπεί σε ένα σημασιολογικά τμηματοποιημένο χάρτη εφαρμόζοντας απλά ένα argmax κατά μήκος του ίδιου pixel κάθε καναλιού. Αυτό απεικονίζεται στο σχήμα C.4.



Σχήμα C.4: One-hot αναπαράσταση των labels. Από το [5]

Στο σχήμα C.5 βλέπουμε την πρόβλεψη να είναι πάνω στην εικόνα εισόδου.

## Σχετική Βιβλιογραφία

### Εξέλιξη της Σημασιολογικής Κατάτμησης

Είναι αξιοσημείωτο ότι η σημασιολογική κατάτμηση είναι ένα απο τα θεμελιώδη προβλήματα της Όρασης Υπολογιστών. Από την έλευση των ψηφιακών εικόνων, οι ερευνητές μελετούν το θέμα αυτό. Οι αλγόριθμοι





Σχήμα C .5: Τελικό Αποτέλεσμα. Από το [5]

βελτιώνονται με την πάροδο των ετών, εξελίσσοντας από την απλή κατωφλιοποίηση της εικόνας, που ταξινομεί τα pixels σε δύο ομάδες, στα βαθιά νευρωνικά δίκτυα, που εκτελούσαν τμηματοποίηση πολλαπλών κατηγοριών με πολύ καλά αποτελέσματα.

### Σημασιολογική Κατάτμηση Πριν τη Χρήση Βαθιών Νευρωνικών Δικτύων

Πριν τη χρήση των Βαθιών Νευρωνικών Δικτύων χρησιμοποιούνταν μέθοδοι όπως ο Αλγόριθμος Πλημμύρας, η Συσταδοποίηση K-means, η Κατωφλιοποίηση της Εικόνας (για το διαχωρισμό του foreground από το background της εικόνας) , τα Conditional Random Fields (CRFs) κ.α.

### Σημασιολογική Κατάτμηση Μετά τη Χρήση Βαθιών Νευρωνικών Δικτύων

Η ενσωμάτωση της βαθιάς μάθησης στην Όραση Υπολογιστών αποτέλεσε γεγονός σταθμό, καθώς τα Νευρωνικά Δίκτυα πέτυχαν state-of-the-art επιδόσεις σε όλα τα προβλήματα επεξεργασίας εικόνας, συμπεριλαμβανομένου και της σημασιολογικής κατάτμησης. Τις διάφορες αρχιτεκτονικές των μοντέλων αυτών μπορούμε να τις ταξινομήσουμε στις εξής κατηγορίες με βάση το κυρίαρχο χαρακτηριστικό τους. Έτσι έχουμε Νευρωνικά που:

- **Βασίζονται σε Πλήρως Συνδεδεμένα Δίκτυα**

Στην κατηγορία αυτή ανήκουν τα Πλήρως Συνδεδεμένα Συνελικτικά Νευρωνικά Δίκτυα (Fully Convolutional Network). Αυτά ήταν από τα πρώτα μοντέλα που έκαναν fine tuning σε ήδη υπάρχοντα μοντέλα ταξινόμησης όπως το VGG/Alex Net κ.τ.λ για να χρησιμοποιηθούν για το πρόβλημα του semantic segmentation. Ουσιαστικά αφαίρεσαν από αυτά τα μοντέλα τα πλήρως συνδεδεμένα επίπεδα (fully connected layers) που αποτελούνταν από  $1 \times 1$  συνελίξεις και έβαλαν στη θέση τους  $1 \times 1$  συνελίξεις με διάσταση καναλιού 21 για να προβλέψουν σκορ για το Pascal Voc Dataset.

- **Βασίζονται σε συνελίξεις τύπου Dilation/Atrous**

Στην κατηγορία αυτή ανήκουν:

- **το DilatedNet:** συσσωρεύει πολυκλιματωτή πληροφορία για να βελτιώσει την τμηματοποίηση.
- **το μοντέλο της Deeplab:** χρησιμοποιεί την έννοια των «Atrous Convolutions» για να λύσει το πρόβλημα της μείωσης την ανάλυσης των feature maps, που παρουσιάστηκε στα FCN κατά ένα παραγοντα 32 λόγω των συνεχόμενων pooling και strides . Χάρη σε αυτά μπορούμε να προσαρμόσουμε το πεδίο όρασης με χαμηλό κόστος και λιγότερες παραμέτρους.

- **Ακολουθούν Top-down/Bottom-up προσέγγιση**

Στην κατηγορία αυτή ανήκουν το:

- **DeconvNet:** αποτελείται από ένα convolution δίκτυο και ένα deconvolution δίκτυο, το οποίο είναι καθρέφτης του αρχικού. Για να γίνει το unpooling, χρησιμοποιούνται τα max pooling indices που αποθηκεύονται κατά τη διάρκεια του πρώτου δικτύου.

- **U-Net**: μεταφέρει τους feature maps από τους encoders στους αντίστοιχους decoders και τους ενώνει με τους αντίστοιχους feature maps των decoders, οι οποίοι έχουν υποστεί υπερδευματοληψία.
- **SegNet**: παρόλο που μοιάζει με το DeconvNet, έχει μικρότερο κόστος από αυτό, γιατί στο μοντέλο αυτό έχουν διαγραφεί τα πλήρως συνδεδεμένα επίπεδα, ώστε να έχει υψηλότερης ανάλυσης feature maps.

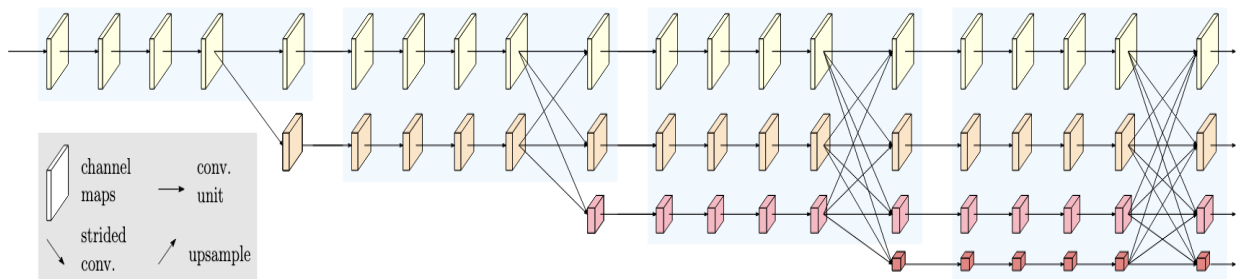
- **Βασίζονται σε πολυκλιμακωτή ενσωμάτωση της πληροφορίας**

Στην κατηγορία αυτή ανήκουν δύο βελτιωμένα απο την Deeplab:

- **Deeplabv2**: Στο μοντέλο αυτό χρησιμοποιήθηκε Χωρική Πυραμίδα από Atrous Convolutions, δηλαδή παράλληλες dilated συνελίξεις σε διαφορετικές κλίμακες και έπειτα αυτές έγιναν fuse.
- **Deeplabv3**: Στο μοντέλο αυτό χρησιμοποιήθηκε αυτή η Χωρική Πυραμίδα εμπλουτισμένη, όμως, με επιπλέον χαρακτηριστικά της εικόνας για πιο βελτιωμένα αποτελέσματα.

Επίσης, εδώ ανήκει το:

- **PSPNet**: μια πυραμίδα τεσσάρων επιπέδων από pooling πυρήνες.
- **SwiftNet**: χρήση διαμοιραζόμενων αναπαραστάσεων πυραμίδας και και σύντηξη ετερογενών χαρακτηριστικών την στιγμή της υπερδευματοληψίας. Το μοντέλο αυτό πετυχαίνει state-of-the-art αποτελέσματα (82.82%) στο Adverse Conditions Dataset with Correspondences (ACDC).
- **HRNet [6]**: Το μοντέλο αυτό θα μας απασχολήσει στην παρούσα διπλωματική. Χρησιμοποιείται ευρέως σε προβλήματα, όπως σημασιολογική κατάτμηση, ανίχνευση αντικειμένων και ταξινόμηση εικόνων. Πετυχαίνει χωρίς καμία επιπλέον προσθήκη άλλης δομής 81.6% <sup>1</sup> mIoU στο Cityscapes [7] και 54.0% mIoU στο Pascal-Context[8] που είναι και τα state-of-the-art αποτελέσματα για το πρόβλημα της σημασιολογικής κατάτμησης. Με την προσθήκη της δομής OCR [9] πετυχαίνει 84.5% και 56.2% στα αντίστοιχα σύνολα δεδομένων. Μπορεί να διατηρήσει αναπαραστάσεις υψηλής ανάλυσης καθ' όλο το μήκος του. Το μοντέλο αποτελείται αρχικά απο μια ροή υψηλή ανάλυσης (high resolution stream) και στη συνέχεια, πριν την παράλληλη σύνδεσή της με ροές πολυκλιμακωτών αναλύσεων, προστίθεται σε αυτή συνελίξεις που χαμηλώνουν την αρχική υψηλή ανάλυση. Το τελικό δίκτυο αποτελείται απο πολλά στάδια ( 4 στο [6]). Στο τελευταίο, γίνεται μια συνένωση (fusion) των διαφορετικών αναλύσεων. Θα αναλύσουμε τη δομή του HRNet σε βάθος. Αρχικά, δίνουμε σαν είσοδο στο δίκτυο μια εικόνα μέσω μια αρχικής δομής (stem network), που αποτελείται απο 2 3 × 3 συνελίξεις με stride 2. Αυτές μειώνουν την αρχική ανάλυση στο  $\frac{1}{4}$  της αρχικής. Έτσι, το κύριο σώμα του δικτύου επιστρέφει την τελική εικόνα με ανάλυση μειωμένη στο  $\frac{1}{4}$  της αρχικής. Η κύρια δομή του HRNet φαίνεται στο Σχήμα D .6.



Σχήμα D .6: Απεικόνιση της δομής του HRNet. Από το [6]

Αποτελείται απο τα εξής μέρη:

- \* **Παράλληλες συνελίξεις σε πολλαπλές κλίμακες**

Το μοντέλο αποτελείται αρχικά απο μια ροή υψηλή ανάλυσης (high resolution stream) και στη συνέχεια, πριν την παράλληλη σύνδεσή της με ροές πολυκλιμακωτών αναλύσεων, προστίθεται σε

<sup>1</sup>Όταν το μοντέλο εκπαιδεύεται με βάση το train set πετυχαίνει 80.4%, ενώ όταν εκπαιδεύεται με βάση το train+val set πετυχαίνει 81.6%

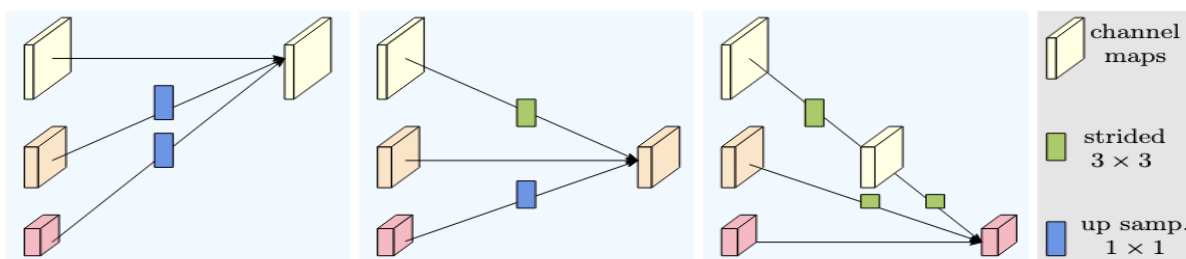
αυτή αλληπαλλήλες συνελίξεις που χαμηλώνουν την αρχική υψηλή ανάλυση. Σαν αποτέλεσμα, οι αναλύσεις των παράλληλων ροών ενός επόμενου σταδίου κατασκευάζονται ουσιαστικά από τις αναλύσεις προηγούμενων σταδίων συν ένα επιπλέον χαμηλότερης ανάλυσης.

#### \* Επαναλαμβανόμενα fusion σε πολλαπλές κλίμακες

Ο στόχος της συγχώνευσης των πολλαπλών αναλύσεων είναι ο διαμοιρασμός της πληροφορίας κατά μήκος όλων των αναπαραστάσεων των διαφόρων κλιμάκων. Αυτή η διαδικασία επαναλαμβάνεται μερικές φορές ( κάθε 4 residual units στο μοντέλο του [6]). Θα αναλύσουμε ένα παράδειγμα συγχώνευσης αναπαραστάσεων 3 διαφορετικών αναλύσεων. Αυτό απεικονίζεται στο σχήμα D .7. Η είσοδος αποτελείται από 3 αναπαραστάσεις  $R_r^i, r = 1, 2, 3$ , όπου το  $r$  είναι ο δείκτης της κάθε ανάλυσης, και οι αντίστοιχες αναπαραστάσεις εξόδου είναι  $R_r^o, r = 1, 2, 3$ . Η συνάρτηση μεταφοράς  $f_{xr}(\cdot)$  επιλέγεται με βάση τον δείκτη της ανάλυσης της αναπράστασης εισόδου  $x$  και της ανάλυσης της αναπράστασης εξόδου  $r$ . Έχουμε τις εξής περιπτώσεις:

- Αν  $x = r$ , τότε  $f_{xr}(R) = R$
- Αν  $x < r$ , τότε το  $f_{xr}(R)$  χρησιμοποιεί  $(r - s)$  stride-2  $3 \times 3$  συνελίξεις για να κάνει υποδειγματοληψία στην αναπράσταση εισόδου  $R$
- Αν  $x > r$ , τότε το  $f_{xr}(R)$  κάνει υπερδειγματοληψία στην αναπράσταση εισόδου  $R$  χρησιμοποιώντας διγραμμική υπερδειγματοληψία και μια  $1 \times 1$  συνέλιξη προκειμένου να προσαρμόσει τον αριθμό των καναλιών.

Κάθε αναπράσταση εξόδου καθορίζεται ως το άθροισμα των μετασχηματισμένων αναπαραστάσεων των 3 εισόδων:  $R_r^o = f_{1r}R_1^i + f_{2r}R_2^i + f_{3r}R_3^i$



Σχήμα D .7: Απεικονίζοντας πώς η διαδικασία συγχώνευσης συσσωρεύει την πληροφορία από υψηλές, μεσαίες και χαμηλές αναλύσεις από τα δεξιά προς τα αριστερά. Από το [6]

#### \* Κεφαλή Αναπράστασης

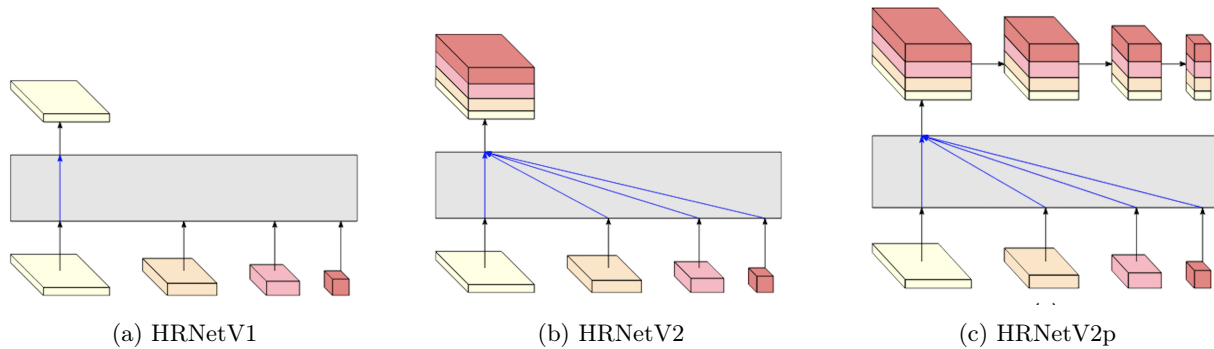
Οι κεφαλές αναπράστασης χωρίζονται σε 3 κατηγορίες, όπως φαίνεται και στο σχήμα D .8 :

- **HRNetV1:** Στην περίπτωση αυτή, μόνο η ροή της υψηλότερης ανάλυσης είναι η έξοδος του συστήματος, ενώ οι υπόλοιπες 3 που έχουν χαμηλότερη ανάλυση αγνοούνται.
- **HRNetV2:** Εδώ χρησιμοποιείται διγραμμική υπερδειγματοληψία ώστε οι χαμηλές αναλύσεις να αποκτήσουν ίδια ανάλυση με την υψηλότερη, χωρίς όμως να επηρεαστεί ο αριθμός των καναλιών του. Έπειτα, οι 4 αυτές αναπαραστάσεις, έχοντας πλέον την ίδια ανάλυση, συγχωνεύονται και έπειτα εφαρμόζεται σε αυτές μια  $1 \times 1$  συνέλιξη.
- **HRNetV2p:** Στην περίπτωση αυτή, πολυεπίπεδες αναπαραστάσεις κατασκευάζονται κάνοντας υποδειγματοληψία σε διάφορες κλίμακες της εξόδου του HRNetV2

Στην παρούσα διπλωματική θα χρησιμοποιήσουμε το HRNetV2.

#### • Βασίζονται σε Transformers

Στην κατηγορία αυτή ανήκει ο:



Σχήμα D .8: Κεφαλές Αναπαράστασης

- **Segmentation Transformer: Object-Contextual Representations (OCR)**: Το μοντέλο αυτό αρχικά μαθαίνει τις περιοχές των αντικειμένων (object areas), όπως αυτές υπολογίζονται από ένα βαθύ νευρωνικό δίκτυο (π.χ HRNet [6]). Έπειτα, υπολογίζει την αναπαράσταση των περιοχών αυτών συνδυάζοντας τις αναπαραστάσεις των pixel που συνθέτουν τις περιοχές αυτές. Τέλος, αφού υπολογίσει την σχέση μεταξύ κάθε pixel και κάθε περιοχής αντικειμένου (object region), προσθέτει στην αναπαράσταση του pixel την αναπαράσταση αντικειμένου (object-contextual representation), που είναι ουσιαστικά ένα βεβαρημένο άθροισμα όλων των αναπαραστάσεων των περιοχών των αντικειμένων.
- **Vision Transformer (ViT)**: Το μοντέλο αυτό ταξινομεί εικόνες, χρησιμοποιώντας πολλά τμήματα της αρχικής εικόνας και έχοντας παρόμοια δομή με αυτή του Transformer.

**Lawin Transformer**: Το μοντέλο αυτό, χρησιμοποιώντας μια μέθοδο προσοχής παραθύρου (window attention method), συμπεριλαμβάνει επιτυχώς πολυκλιμακωτές αναπαραστάσεις στον ViT, βελτιώνοντας σημαντικά την επίδοσή του.

**Vision Transformer Adapter (ViT-Adapter)**: Ο ViT έχει φτωχές επιδόσεις σε προβλήματα πυκνής πρόβλεψης, καθώς δεν έχει προηγούμενη γνώση των εικόνων, σε αντίθεση με τους πρόσφατους visual transformers που περιλαμβάνουν στα μοντέλα τους σχετικά με την όραση επαγωγικά (inductive) biases. Για την αντιμετώπιση αυτού του προβλήματος, προτάθηκε ο ViT-Adapter, ο οποίος, ενσωματώνοντας inductive biases μέσω ενός επιπλέον σχεδιασμού, μπορεί να διορθώσει τις αδυναμίες του ViT και να επιτύχει βελτιωμένες επιδόσεις. Το μοντέλο αυτό πετυχαίνει state-of-the-art αποτελέσματα στο Cityscapes.

## Η προτεινόμενη Μέθοδος: Μοντέλο Βασισμένο σε Διανύσματα Μετατόπισης ( Offset Vector - Based Model)

### Κεντρική Ιδέα

Σε αυτή την ενότητα, θα εισάγουμε μια νέα προσέγγιση για τη βελτίωση των σημασιολογικών προβλέψεων. Βασιζόμενοι στη γνώση σχετικά με την υψηλού επιπέδου κανονικότητα των πραγματικών σκηνών, προτείνουμε μια νέα μέθοδο για τη βελτίωση των προβλεπόμενων κλάσεων μέσω της εκμάθησης της επιλεκτικής αξιοποίησης των πληροφοριών των πληροφοριών από τα συνεπίπεδα pixels. Συγκεκριμένα, με βάση το [10], εισάγουμε ένα νέο Λήμμα, που υποστηρίζει ότι για κάθε pixel υπάρχει ένα seed pixel που έχει την ίδια πρόβλεψη με το αρχικό. Σαν αποτέλεσμα, σχεδιάζουμε ένα δίκτυο δύο κεφαλών. Μια παρόμοια προσέγγιση πραγματοποιήθηκε και για το πρόβλημα εκτίμησης βάθους στο [10].

Η νέα μέθοδος εφαρμόστηκε στο HRNetV2 [6]. Χρησιμοποιούμε το κύριο σώμα από το HRNet (όπως απεικονίζεται στο Σχήμα 2 .12), αλλά αποφεύγουμε να προβλέψουμε κατευθείαν κλάσεις. Αντίθετα, το χρησιμοποιούμε σαν έξοδο για τον καθορισμό σχέσεων μεταξύ γειτονικών pixel. Συγκεκριμένα, η πρώτη κεφαλή του δικτύου έχει σαν έξοδο 4 αναπαραστάσεις διαφορετικής ανάλυσης, οι οποίες στη συνέχεια ενώνονται, όπως φαίνεται στην Εικόνα 2 .14b. Έπειτα, αυτή η συνδυασμένη αναπαράσταση υψηλής ευκρίνειας χρησιμοποιείται σαν είσοδος στο

τελευταίο επίπεδο, το οποίο έχει σαν έξοδο logits σε επίπεδο pixel. Έπειτα, αυτά τα logits μετατρέπονται σε κλάσεις. Η βασική ιδέα της μεθόδου μας, αξιοποιεί το γεγονός ότι δυο pixels  $\mathbf{p}$  και  $\mathbf{q}$  που ανήκουν στην ίδια κλάση έχουν ιδανικά παρόμοιες αναπαραστάσεις logit. Αν τα pixels ανήκουν στην ίδια κλάση, αν εφαρμόσουμε στο  $\mathbf{p}$  την πρόβλεψη του  $\mathbf{q}$  για την εκτιμώμενη κλάση, τότε θα οδηγηθούμε σε έγκυρη πρόβλεψη.

Αξιοποιούμε αυτή την ιδιότητα μαθαίνοντας να εντοπίζουμε τα seed pixels που ανήκουν στην ίδια κλάση με το pixel που ελέγχεται, όταν αυτά υπάρχουν, ώστε να χρησιμοποιούμε επιλεκτικά τα logits αυτών των pixels για τη βελτίωση της προβλεπόμενης κλάσης. Αυτή η ιδέα βασίζεται στην αρχή που παρουσιάζεται στο [10], σύμφωνα με την οποία για κάθε pixel  $\mathbf{p}$  που σχετίζεται με ένα 3D επίπεδο, υπάρχει ένα seed pixel  $\mathbf{q}$  στη γειτονιά του  $\mathbf{p}$  που σχετίζεται με το ίδιο επίπεδο με το  $\mathbf{p}$ . Βασιζόμενοι σε αυτή την αρχή, είναι επόμενο ότι για κάθε pixel, υπάρχει ένα seed pixel που μοιράζεται την ίδια πρόβλεψη με το αρχικό. Όπως είναι λογικό, πρέπει πρώτα να οριστούν οι περιοχές στις οποίες ισχύει αυτή η αρχή. Στη συνέχεια, προκειμένου να προσδιοριστούν τα seed pixels σε αυτές τις περιοχές, πρέπει να προβλέψουμε τα διανύσματα μετατόπισης  $\mathbf{o}(\mathbf{p}) = \mathbf{q} - \mathbf{p}$  για κάθε pixel  $\mathbf{p}$ . Προκειμένου να ληφθούν υπόψη πιθανές αποκλίσεις από την ακριβή τοπική ομαλότητα, η πρόβλεψη που προκύπτει συγχωνεύεται προσαρμοστικά με την αρχική πρόβλεψη από την πρώτη κεφαλή χρησιμοποιώντας έναν χάρτη εμπιστοσύνης (confidence map), τον οποίο μαθαίνει το μοντέλο. Σαν αποτέλεσμα, σχεδιάζουμε μια δεύτερη κεφαλή η οποία παράγει ένα πυκνό πεδίο διανυσμάτων μετατόπισης και ένα χάρτη εμπιστοσύνης. Έτσι, έχουμε μια νέα έκδοση του HRNetV2 η οποία βασίζεται σε διανύσματα μετατόπισης. Αυτή η ιδέα υλοποιείται σε διαφορετικά μέρη του HRNetV2, όπως φαίνεται στο Σχήμα E.9. Στο Σχήμα αυτό, παρατηρούμε το κύριο σώμα του δικτύου με τη διακλάδωση να συμβαίνει στο δεύτερο ( Σχήμα E.9a ), στο τρίτο ( Σχήμα E.9b ) ή στο τέτατο ( Σχήμα E.9c ) στάδιο του δικτύου αντίστοιχα. Είναι αξιοσημείωτο ότι οι δύο κεφαλές δεν μοιράζονται τα ίδια βάρη.

Το δίκτυό μας εκτιμά κλάσεις, με το συνδυάζει επιλεκτικά προβλέψεις από κάθε pixel και του αντίστοιχου seed pixel. Τα προβλεπόμενα διανύσματα μετατόπισης χρησιμοποιούν τις προβλέψεις της πρώτης κεφαλής και παράγουν μια δεύτερη πρόβλεψη στη θέση του seed pixel. Στη συνέχεια, οι προβλέψεις αυτές από κάθε κεφαλή συγχωνεύονται προσαρμοστικά χρησιμοποιώντας τον χάρτη εμπιστοσύνης. Μια επισκόπηση αυτής της αρχιτεκτονικής φαίνεται στο Σχήμα E.10

Θα αξιολογήσουμε τη μέθοδό μας σε 2 datasets για σημασιολογική κατάτμηση με επίβλεψη:

- Cityscapes [7]
- ACDC [11]

## Seed Pixels

Ας υποθέσουμε ότι έχουμε ένα pixel  $\mathbf{p}$  το οποίο ανήκει σε ένα τμήμα μιας σημασιολογικά τμηματοποιημένης εικόνας. Εξ ορισμού, κάθε άλλο pixel του τμήματος αυτού έχει την ίδια ετικέτα κλάσης. Έτσι, ιδανικά, προκειμένου να πάρουμε όλες τις ετικέτες κλάσης σωστά, το δίκτυο πρέπει να προβλέψει τη σωστή κλάση σε ένα από τα pixels,  $\mathbf{q}$ . Αυτό το pixel μπορεί να ερμηνευτεί σαν seed pixel που περιγράφει το αντίστοιχο τμήμα-κλάση. Τελικά, αφήνουμε το δίκτυό μας να βρει αυτό το pixel και την αντίστοιχη περιοχή.

Η ιδέα αυτή βασίζεται στην αρχή Piecewise Planarity Prior από το [10]. Ορίζουμε το ακόλουθο Λήμμα

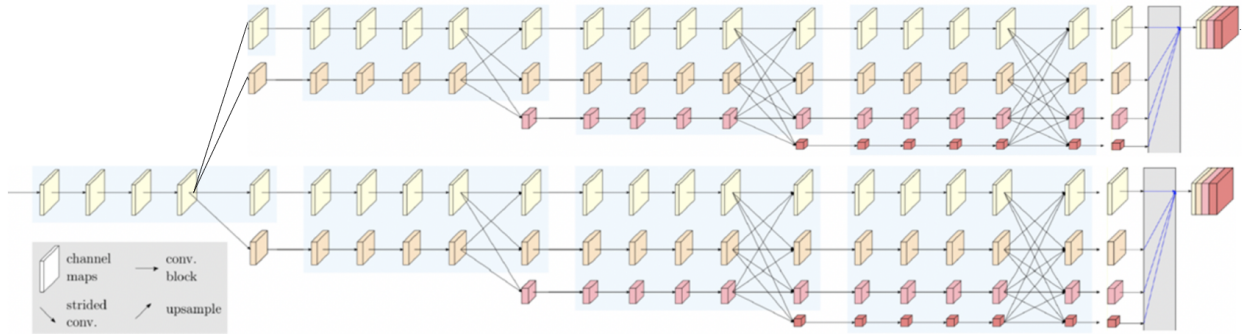
### Λήμμα E.6

Για κάθε pixel  $\mathbf{p}$  που σχετίζεται με μια 2D σημασιολογικά τμηματοποιημένη εικόνα, υπάρχει ένα seed pixel  $\mathbf{q}$  στη γειτονιά του  $\mathbf{p}$  που μοιράζεται την ίδια πρόβλεψη με το πρώτο.

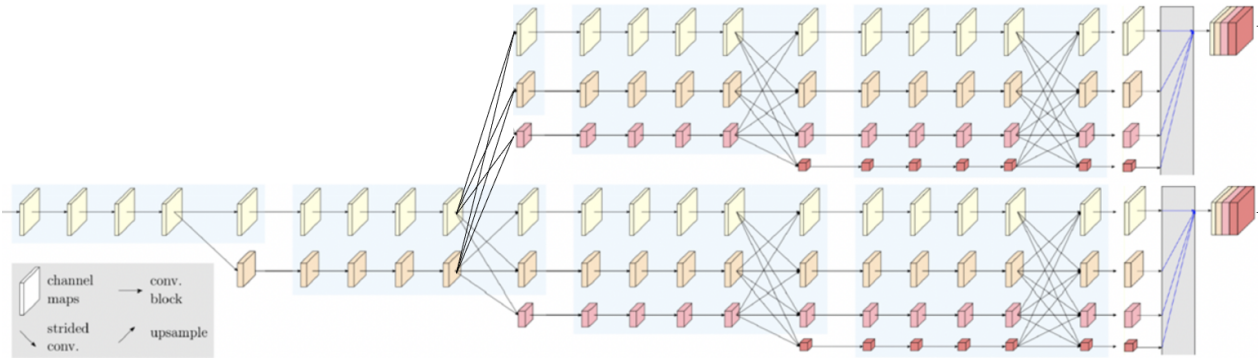
Γενικά, μπορούν να υπάρχουν πολλά seed pixels για το pixel  $\mathbf{p}$  ή κανένα. Δεδομένου ότι το παραπάνω Λήμμα ισχύει, το πρόβλημα της σημασιολογικής κατάτμησης για το  $\mathbf{p}$  μπορεί να λυθεί με το να καθορίσουμε το  $\mathbf{q}$ . Για αυτό το λόγο, αφήνουμε το δίκτυό μας να προβλέψει τα διανύσματα μετατόπισης (offset vectors)  $\mathbf{o}(\mathbf{p}) = \mathbf{q} - \mathbf{p}$ . Έτσι, σχεδιάζουμε το δίκτυό μας, έτσι ώστε να παράγει μια δεύτερη κεφαλή, η οποία προβλέπει ένα πυκνό πεδίο διανυσμάτων μετατόπισης  $\mathbf{o}(\mathbf{u}, \mathbf{v})$ . Σε όλες τις διαφορετικές εκδόσεις του μοντέλου, οι δύο κεφαλές μοιράζονται ένα κοινό σώμα και, όταν συμβαίνει η διακλάδωση, ακολουθούν διαφορετικά μονοπάτια. Έπειτα, βρίσκουμε τις τιμές των logits στις θέσεις των seed pixels χρησιμοποιώντας το πεδίο διανυσμάτων μετατόπισης:

$$C_s(\mathbf{p}) = C_i(\mathbf{p} + \mathbf{o}(\mathbf{p})) \quad (\text{E.1})$$

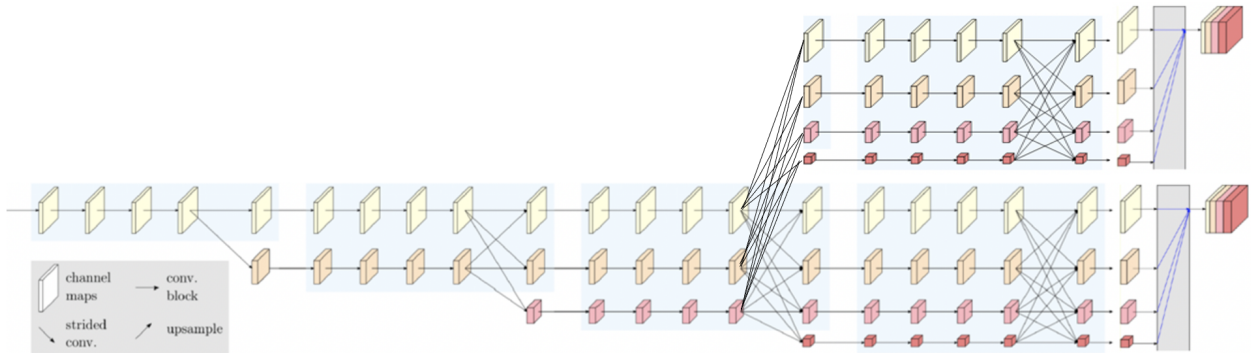




(a) Η διακλάδωση συμβαίνει στο 2<sup>ο</sup> στάδιο του HRNetV2

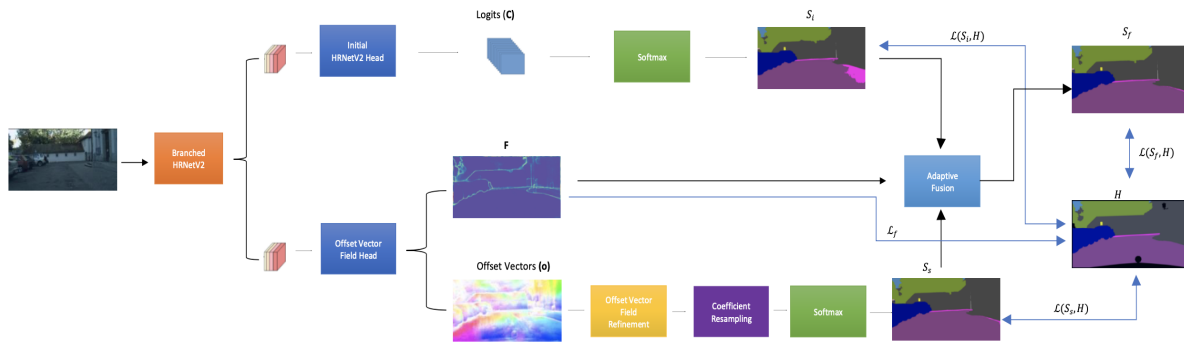


(b) Η διακλάδωση συμβαίνει στη μέση του HRNetV2



(c) Η διακλάδωση συμβαίνει στο 4<sup>ο</sup> στάδιο του HRNetV2

Σχήμα Ε .9: HRNetV2 Βασισμένο στα Διανύσματα Μετατόπισης



Σχήμα E .10: **Επισκόπηση της πλήρους μεθόδου.** Η έκδοση του HRNetV2 που βασίζεται σε διανύσματα μετατόπισης ( Σχ.Ε .9 ) , αποτελείται από δύο κεφαλές. Η πρώτη κεφαλή εξάγει logits ( $C$ ) σε επίπεδο pixel, ενώ η δεύτερη εξάγει ένα πυκνό πεδίο διανυσμάτων μετατόπισης ( $\mathbf{o}$ ) για τον καθορισμό των θέσεων των seed pixels μαζί με ένα χάρτη εμπιστοσύνης ( $F$ ). Στη συνέχεια, οι προβλέψεις των seed pixels χρησιμοποιούνται για να προβλέψουν κλάσεις σε κάθε pixel. Η πρόβλεψη που προκύπτει ( $S_s$ ) συγχωνεύεται προσαρμοστικά με την αρχική πρόβλεψη  $S_i$  χρησιμοποιώντας τον χάρτη εμπιστοσύνης ( $F$ ), προκειμένου να υπολογιστεί η τελική πρόβλεψη  $S_f$

Επειδή οι συντεταγμένες των διανυσμάτων μετατόπισης δεν είναι πάντα ακέραιες, χρησιμοποιούμε διγραμμική παρεμβολή. Τα logits στις θέσεις των seed pixels χρησιμοποιούνται για να υπολογίσουν μια δεύτερη πρόβλεψη σημασιολογικής κατάτμησης:

$$S_s(u, v) = h(C_s(u, v), u, v) \implies S_s(p) = S_i(p + o(p)) \quad (\text{E} .2)$$

Στο παράδειγμά μας,  $h = \text{softmax}$ .

Λόγω του ότι η παραπάνω πρόβλεψη δεν οδηγεί πάντα σε σωστό αποτέλεσμα, η αρχική σημασιολογικά πρόβλεψη  $S_i$  μπορεί μερικές φορές να προτιμάται περισσότερο από την πρόβλεψη  $S_s$  που βασίζεται στα seed pixels. Προκειμένου να αντιμετωπίσουμε τέτοιες περιπτώσεις, η δεύτερη κεφαλή προβλέπει επιπλέον ένα χάρτη εμπιστοσύνης  $F(u, v) \in [0, 1]$ , ο οποίος δείχνει την εμπιστοσύνη που δείχνει το μοντέλο στο να προτιμήσει την πρόβλεψη  $S_s$  έναντι της αρχικής  $S_i$ . Συνδυάζοντας προσαρμοστικά τα  $S_i$  και  $S_s$ , ο χάρτης εμπιστοσύνης χρησιμοποιείται για να υπολογίσει την τελική πρόβλεψη:

$$S_f(p) = (1 - F(p))S_i(p) + F(p)S_s(p) \quad (\text{E} .3)$$

Στο μοντέλο μας, κάνουμε επίβλεψη καθενός από τα  $S_f$ ,  $S_s$  και  $S_i$ , προσπαθώντας να βελτιστοποιήσουμε το ακόλουθο loss:

$$\mathcal{L}_{\text{semantic}}(p) = \mathcal{L}(S_f, H) + \kappa \mathcal{L}(S_s, H) + \lambda \mathcal{L}(S_i, H) \quad (\text{E} .4)$$

όπου  $\kappa$  and  $\lambda$  είναι υπερπαραμέτροι και το  $H$  συμβολίζει τα Ground-Truth (GT) train ids κάθε κλάσης για κάθε pixel. Με αυτόν τον τρόπο, ενδιαφερόμαστε

- την αρχική κεφαλή του HRNetV2 να παράγει μια ακριβή αναπαράσταση κατά μήκος όλων των pixel, ακόμα και όταν έχουν υψηλή τιμή εμπιστοσύνης
- την κεφαλή που προβλέπει τα διανύσματα μετατόπισης να μάθει υψηλές τιμές εμπιστοσύνης για τα pixel για τα οποία το Λήμμα ισχύει και χαμηλή τιμή εμπιστοσύνης για τα pixel για τα οποία το Λήμμα δεν ισχύει

Αυτή η ιδέα, όμως, έχει ένα μειονέκτημα. Συγκεκριμένα, το μοντέλο δεν κάνει επίβλεψη άμεσα στα διανύσματα μετατόπισης. Πράγματι, θα μπορούσε απλά να προβλέπει μηδενικά διανύσματα μετατόπισης παντού και ακόμα και έτσι να δίνει έγκυρες προβλέψεις  $S_s$  και  $S_f$  που είναι ίσες με το  $S_i$ . Οι αρχικές προβλέψεις  $S_i$ , όμως,

αλλάζουν ομαλά γύρω από τα σημασιολογικά σύνορα λόγω της κανονικότητας της συνάρτησης  $f_\theta$ , πράγμα που αποφεύγεται στην πράξη. Σαν αποτέλεσμα, η πρόβλεψη ενός μη μηδενικού διανύσματος μετατόπισης που δείχνει μακριά από τα όρια κάθε τμήματος δίνει μια μικρότερη τιμή για το  $\mathcal{L}_{semantic}$  για τα pixels εκατέρωθεν του ορίου. Αυτό συμβαίνει, επειδή ένα τέτοιο διάνυσμα μετατόπισης χρησιμοποιεί ένα seed pixel για το  $S_s$  το οποίο είναι πιο μακριά από το σύνορο και υποφέρει από μειωμένη ανακρίβεια λόγω εξομάλυνσης. Αυτά τα μη μηδενικά διανύσματα μετατόπισης μεταδίδονται, επίσης, από τα όρια στα εσωτερικά τμήματα των περιοχών με ομαλά τμήματα, βοηθώντας το δίκτυο στην πρόβλεψη μη τετριμμένων μετατοπίσεων λόγω της κανονικότητας της απεικόνισης που σχηματίζει το διανυσματικό πεδίο των μετατοπίσεων.

Τελικά, πριν γίνει η επαναδειγματοληψία των logits, δοκιμάζουμε σε μερικές εκδόσεις του μοντέλου να κλιμακώσουμε τα διανύσματα μετατόπισης μερικές φορές. Η ιδέα πίσω από τη βελτίωση αυτή βασίζεται στο ότι τα seed pixels μέσα στο ίδιο τμήμα πρέπει να συγκλίνουν στο κέντρο του κάθε τμήματος. Αυτό βοηθά στη συσσώρευση της πληροφορίας από επιπρόσθετα pixel και βελτιώνει την πρόβλεψη της κλάσης του αντίστοιχου τμήματος. Η κλιμάκωση των διανυσμάτων μετατόπισης δεν βλάπτει την πρόβλεψη της σχετικής κλάσης, δεδομένου ότι στα pixels χωρίς αξιόπιστα seed pixels αποδίδεται ήδη χαμηλή τιμή εμπιστοσύνης.

## Loss Εμπιστοσύνης (Confidence Loss)

Το loss εμπιστοσύνης βασίζεται στην ιδέα ότι δεδομένου ενός pixel, τα περιβάλλοντα pixel του πρέπει να ανήκουν στο ίδιο τμήμα (κλάση). Για κάθε pixel  $p$ , το loss εμπιστοσύνης ορίζεται ως εξής:

$$\mathcal{L}_f(p) = -\mathbb{1}_{[H(p)=H(p)+o(p)]} \log F(p) - \mathbb{1}_{[H(p) \neq H(p)+o(p)]} \log(1 - F(p)) \quad (\text{E}.5)$$

Αυτή η ιδέα προέκυψε από το ότι η εμπιστοσύνη πρέπει να έχει μεγάλη τιμή για τα pixel των οποίων το διάνυσμα μετατόπισης δείχνει σε seed pixel που ανήκει στην ίδια κλάση. Ομοίως, η εμπιστοσύνη πρέπει να έχει μικρή τιμή για τα pixel των οποίων το διάνυσμα μετατόπισης δείχνει σε seed pixel που ανήκει σε διαφορετική κλάση. Όταν το αρχικό pixel  $\mathbf{p}$  και το seed pixel  $\mathbf{q}$  ανήκουν σε διαφορετικές κλάσεις, τότε ο πρώτος όρος απενεργοποιείται και ενεργοποιείται ο δεύτερος. Σε αυτή την περίπτωση, θέλουμε αυτό το seed pixel να έχει μικρή εμπιστοσύνη. Αυτή η ιδέα αντανακλάται από το  $\log(1 - F)$ , γιατί όταν  $F \rightarrow 0 \implies (1 - F) \rightarrow 1^- \implies -\log(1 - F) \rightarrow 0^+ \implies \mathcal{L}_f \rightarrow 0^+$ . Ομοίως, όταν  $F \rightarrow 1 \implies \mathcal{L}_f \rightarrow +\infty$ . Αντίστοιχα με την πρώτη περίπτωση, όταν τα  $\mathbf{p}$  και  $\mathbf{q}$  ανήκουν στην ίδια κλάση, τότε ο δεύτερος όρος απενεργοποιείται. Σε αυτή την περίπτωση, θέλουμε το seed pixel να έχει υψηλή εμπιστοσύνη. Όταν  $F \rightarrow 0 \implies \log(F) \rightarrow -\infty \implies \mathcal{L}_f \rightarrow +\infty$ . Ομοίως, όταν  $F \rightarrow 1 \implies \mathcal{L}_f \rightarrow 0^+$ .

Συνοψίζοντας, το συνολικό loss είναι:

$$\mathcal{L}_{final} = \mathcal{L}_{semantic} + \mathcal{L}_f \quad (\text{E}.6)$$

## Αρχιτεκτονική του Δικτύου

Το προτεινόμενο δίκτυο αποτελείται από τα εξής μέρη:

- **Κύριο σώμα πριν την διακλάδωση:** Η δομή του δικτύου μας πριν την διακλάδωση είναι ίδια με αυτή του HRNet [6]
- **Μετά τη διακλάδωση:** Το νέο HRNetV2 αποτελείται από δύο κεφαλές, η καθεμία από τις οποίες εξάγει διαφορετικά πράγματα. Κάθεμία κεφαλή έχει την ίδια δομή με το αρχικό HRNetV2. Η πρώτη κεφαλή εξάγει logits (C) για κάθε pixel, ενώ η δεύτερη εξάγει ένα πυκνό πεδίο διανυσμάτων μετατόπισης (o) που καθορίζουν τις θέσεις των seed pixels μαζί με ένα χάρτη εμπιστοσύνης (F). Έπειτα, οι συντελεστές των seed pixels χρησιμοποιούνται για να προβλέψουν τις κλάσεις σε κάθε θέση. Η πρόβλεψη που προκύπτει ( $S_s$ ) συνδυάζεται προσαρμοστικά με την αρχική πρόβλεψη ( $S_i$ ) χρησιμοποιώντας τον χάρτη εμπιστοσύνης προκειμένου να υπολογιστεί η τελική πρόβλεψη  $S_f$ .
- **Αρχική Κεφαλή του HRNetV2 :** Όπως και στο αρχικό HRNetV2 [6], το τελευταίο επίπεδο της κεφαλής εξάγει 19 κανάλια ( $19 \times H \times W$ ), δηλαδή 1 για κάθε κλάση. Αυτό γίνεται, γιατί οι κλάσεις τόσο στο Cityscapes όσο και στο ACDC είναι 19.



- **Κεφαλή διανυσμάτων μετατόπισης:** Το τελευταίο επίπεδο της κεφαλής αυτής έχει τροποποιηθεί προκειμένου να εξάγει 3 κανάλια ( $3 \times H \times W$ ): 2 για τα διανύσματα μετατόπισης και 1 για την εμπιστοσύνη. Στα διανύσματα μετατόπισης εφαρμόζεται στη συνέχεια ένα επίπεδο Tanh, με αποτέλεσμα τα διανύσματα πλέον να ανήκουν στο διάστημα  $[-1,1]$ . Από την άλλη, στο κανάλι της εμπιστοσύνης εφαρμόζεται μια σιγμοειδή, με αποτέλεσμα οι τιμές να ανήκουν στο διάστημα  $[0,1]$ .

## Αποτελέσματα και Συγκρίσεις

Για την αξιολόγηση της μεθόδου μας πραγματοποιήσαμε εκτενή ποιοτικά και ποσοτικά πειράματα και συγκρίσεις με άλλες μεθόδους, που καταδεικνύουν την αποτελεσματικότητα και την υπεροχή του μοντέλου μας έναντι άλλων state of the art μοντέλων.

### Σύνολα Δεδομένων

Σε αυτή την ενότητα θα παρουσιάσουμε τα σύνολα δεδομένων που χρησιμοποιήθηκαν για την εκπαίδευση και αξιολόγηση των πειραμάτων μας. Αυτά είναι:

- **Cityscapes [7]**

Το Cityscapes είναι ένα από τα πιο ανταγωνιστικά σύνολα δεδομένων και χρησιμοποιείται σε προβλήματα σημασιολογικής κατάτμησης. Περιλαμβάνει 30 κλάσεις από τις οποίες μόνο οι 19 θα χρησιμοποιηθούν για αξιολόγηση. Έχει επισημειωμένες 5000 εικόνες από τις οποίες οι 2.975 χρησιμοποιούνται για εκπαίδευση, οι 500 για το validation και οι 1.525 για το testing.

- **ACDC [11]**

Το ACDC είναι επίσης ένα απαιτητικό σύνολο δεδομένων, το οποίο περιέχει εικόνες που έχουν ληφθεί σε αντιζοες συνθήκες (ομίχλη, νύχτα, βροχή και χιόνι). Κληρονομεί τους ορισμούς των κλάσεων από το Cityscapes και, συνεπώς, αποτελείται από 19 κλάσεις. Συγκεκριμένα, αποτελείται από 1000 εικόνες σε συνθήκες ομίχλης, 1006 εικόνες τραβηγμένες νύχτα, 1000 σε συνθήκες βροχής και 1000 σε συνθήκες χιονιού. Κάθε σετ από αυτές τις 4 κατηγορίες χωρίζεται σε 400 εικόνες για την εκπαίδευση, 100 για το validation και 500 για το testing, με εξαίρεση ότι η συνθήκη της νύχτας περιλαμβάνει 106 εικόνες για το validation. Άρα, συνολικά έχουμε 1600 επισημειωμένες εικόνες για την εκπαίδευση, 406 επισημειωμένες εικόνες για το validation και 2000 εικόνες για το testing.

### Μετρικές Επίδοσης

Η κύρια μετρική με την οποία θα μετρήσουμε την επίδοση των πειραμάτων μας είναι ο μέσος όρος του Intersection over Union (IoU) (mIoU). Παράλληλα, θα παραθέσουμε και άλλες 3 μετρικές στο test set. Αυτές είναι οι: IoU category (cat.), instance-level Intersection over Union (iIoU) class και iIoU category (cat.).

### Υλοποίηση

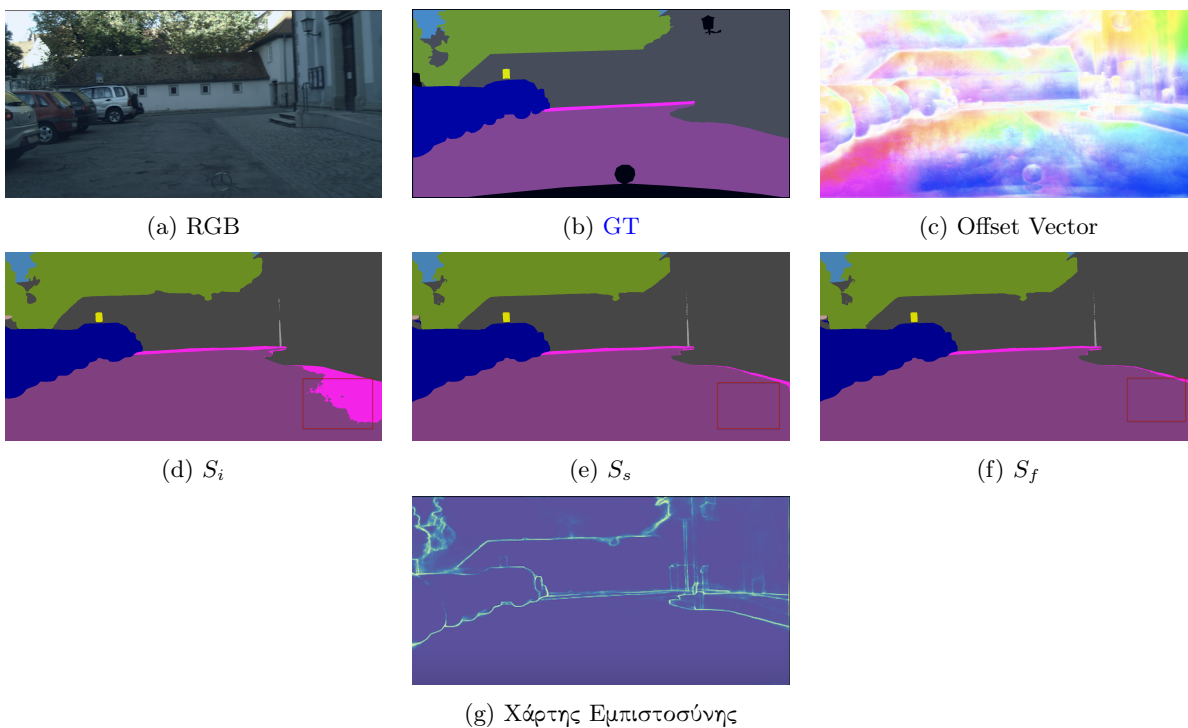
Όλες οι εκδοχές του δικτύου μας αποτελούνται από δύο κεφαλές. Ανάλογα με την εκδοχή η διακλάδωση συμβαίνει σε διαφορετικό σημείο του δικτύου μας. Η πρώτη κεφαλή παράγει 19 κανάλια (ένα για κάθε κλάση), ενώ η δεύτερη 3 κανάλια ( 2 για τις συντεταγμένες των διανυσμάτων μετατόπισης και 1 για τον χάρτη εμπιστοσύνης). Οι δύο κεφαλές ακολουθούν τη δομή του HRNetV2. Τόσο το αρχικό μοντέλο όσο και το μοντέλο μας είναι αρχικοποιημένα με βάρη που προέκυψαν από την προεκπαίδευση του μοντέλου μας στο Imagenet [6]. Η αρχικοποίηση αυτή είναι σημαντική προκειμένου να επιτύχουμε ανταγωνιστικά αποτελέσματα. Σε μερικά επιπλέον πειράματα που δοκιμάσαμε, ακολουθούμε μια διαφορετική προσέγγιση, παγώνοντας, καθόλη τη διάρκεια της εκπαίδευσης του μοντέλου μας, τόσο το κύριο σώμα όσο και την κεφαλή του αρχικού δικτύου. Ανάλογα με το ποιο σύνολο δεδομένων χρησιμοποιούμε, το παγωμένο μέρος του δικτύου μας θα αρχικοποιηθεί με τα αντίστοιχα βάρη του αρχικού μοντέλου, που είναι προεκπαιδευμένο στο αντίστοιχο σύνολο δεδομένων. Το μόνο μέρος που θα εκπαιδευτεί είναι η έξτρα κεφαλή που έχει αρχικοποιηθεί με τα βάρη από το Imagenet. Επιπλέον, χρησιμοποιούμε βελτιστοποιητή SGD με learning rate 0.01, momentum 0.9 και weight decay 0.0005. Στα διανύσματα μετατόπισης έχει εφαρμοσθεί ένα tanh επίπεδο, ώστε οι τιμές τους να ανήκουν στο διάστημα  $[-1,1]$ . Παράλληλα, εφαρμόζουμε και μια παράμετρο  $\tau$  η οποία ρυθμίζει το τελικό μήκος των διανυσμάτων.

Θέτουμε σαν default τιμή το  $\tau$  ίσο 0.5, τη διακλάδωση να συμβαίνει στο 4<sup>ο</sup> στάδιο και κανένα βήμα κλιμάκωσης των διανυσμάτων μετατόπισης. Στον χάρτη εμπιστοσύνης εφαρμόζουμε μια σιγμοειδή. Για τον υπολογισμό του loss στους όρους  $S_i$ ,  $S_s$  και  $S_f$  εφαρμόζεται η συνάρτηση του **OHEM** Cross Entropy. Ιδιαίτερη μνεία πρέπει να γίνει στο ότι αρχικά δοκιμάσαμε να εξάγουμε κατευθείαν τις προβλέψεις από το μοντέλο και όχι τα logits, πράγμα το οποίο οδήγησε σε πολύ χαμηλά αποτελέσματα. Τα βάρη  $\lambda$  και  $\mu$  τίθενται ίσα με 0.5. Το baseline μοντέλο έχει εκπαιδευτεί για 120K επαναλήψεις με batch size ίσο με 12 σε 4 GPUS. Αντιθέτως, το μοντέλο μας, εκπαιδεύτηκε σε 120K επαναλήψεις με batch size ίσο με 8 σε 4 GPUs, επειδή προέκυψαν προβλήματα με τη μνήμη.

## Ποιοτικά αποτελέσματα

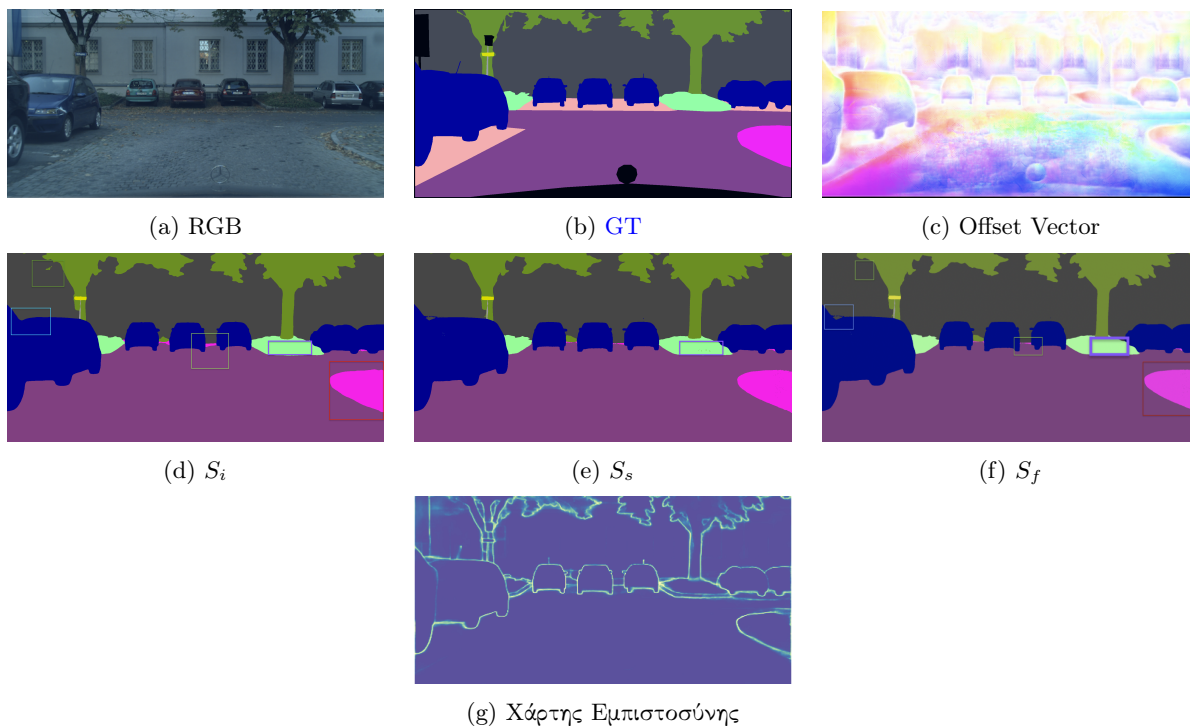
Στα σχήματα **F .11**, **F .12** και **F .13**, **F .14** δείχνουμε ορισμένα ποιοτικά αποτελέσματα της μεθόδου μας στο σύνολο δεδομένων Cityscapes και ACDC αντίστοιχα. Όπως φαίνεται, το μοντέλο μας δεν πετυχαίνει απλά πολύ ικανοποιητικά αποτελέσματα κοντά στις **GT** εικόνες, αλλά επίσης βελτιώνει τα αρχικά αποτελέσματα που παρήγαγε η αρχική κεφαλή του HRNetV2.

Συγκεκριμένα, παραθέτουμε μερικές προβλέψεις του μοντέλου μας σε κάποιες επιλεγμένες εικόνες από το Cityscapes, όπως φαίνεται στα σχήματα **F .11** και **F .12**. Παρατηρώντας το παράδειγμα **F .11**, βλέπουμε από την εικόνα του  $S_i$  ότι η πρόβλεψη που έχει γίνει στο κόκκινο πλαίσιο είναι λανθασμένη. Αυτό διορθώνεται μέσω της πρόβλεψης στα seed pixels ( $S_s$ ) και κατ' επέκταση λόγω μεγαλύτερης εμπιστοσύνης στο συγκεκριμένο κομμάτι στην τελική πρόβλεψη  $S_f$ . Επιπλέον, στο παράδειγμα **F .12**, βλέπουμε ότι λανθασμένες προβλέψεις τόσο του  $S_i$  (μπλε, κόκκινο και πράσινο πλαίσιο) όσο και του  $S_s$  (μόβλ πλαίσιο) διορθώνονται τελικά στην τελική έξοδο  $S_f$ . Παρ' όλα αυτά, αξίζει να αναφέρουμε ότι, αν και η πρόβλεψη του  $S_i$  που περιλαμβάνεται από το πράσινο πλαίσιο μειώνεται στο  $S_f$ , είναι τελικά λάθος, γιατί η **GT** εικόνα στο σημείο αυτό προβλέπει πεζοδρόμιο και όχι δρόμο. Αυτό, όμως, οφείλεται σε αδυναμία του αρχικού μοντέλου, μιας και στο περιβάλλον της αρχικής πρόβλεψης δεν υπάρχει πουθενά η σωστή κλάση και έτσι δεν μπορεί να την αξιοποιήσει το μοντέλο μας.



Σχήμα F .11: Ποιοτικά Αποτελέσματα στο Cityscapes: Πρώτο Παράδειγμα

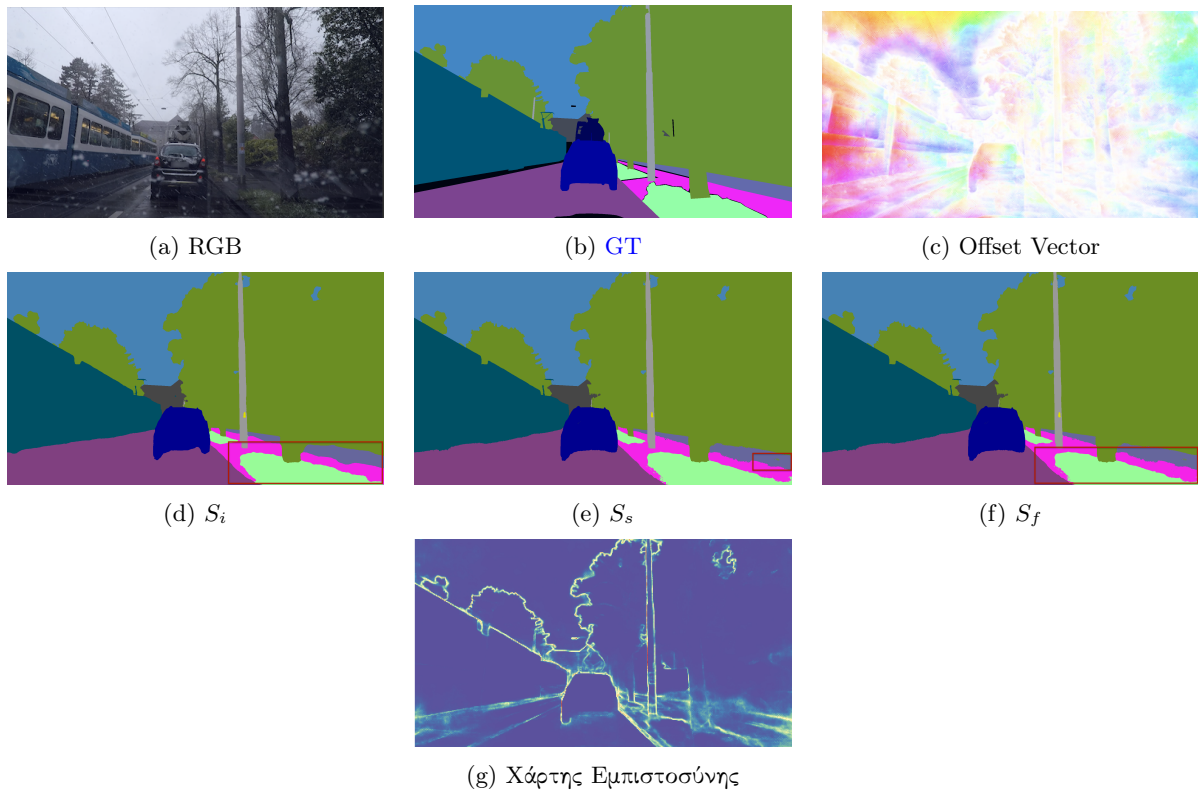
Ανάλογα αποτελέσματα παρατηρούμε και στις επιλεγμένες εικόνες (Σχήματα **F .13**, **F .14**) από το ACDC. Συγκεκριμένα, όπως φαίνεται στο Σχήμα **F .13**, η  $S_f$  πρόβλεψη περιορίζει το τμήμα του εδάφους και μεγαλώνει ορθά το τμήμα του πεζοδρομίου, πετυχαίνοντας έτσι καλύτερο τελικό αποτέλεσμα από την αρχική πρόβλεψη  $S_s$ . Επιπλέον, δεν λαμβάνει υπόψιν την λαθεμένη πρόβλεψη που έχει κάνει το  $S_s$ , όπως φαίνεται στο κόκκινο



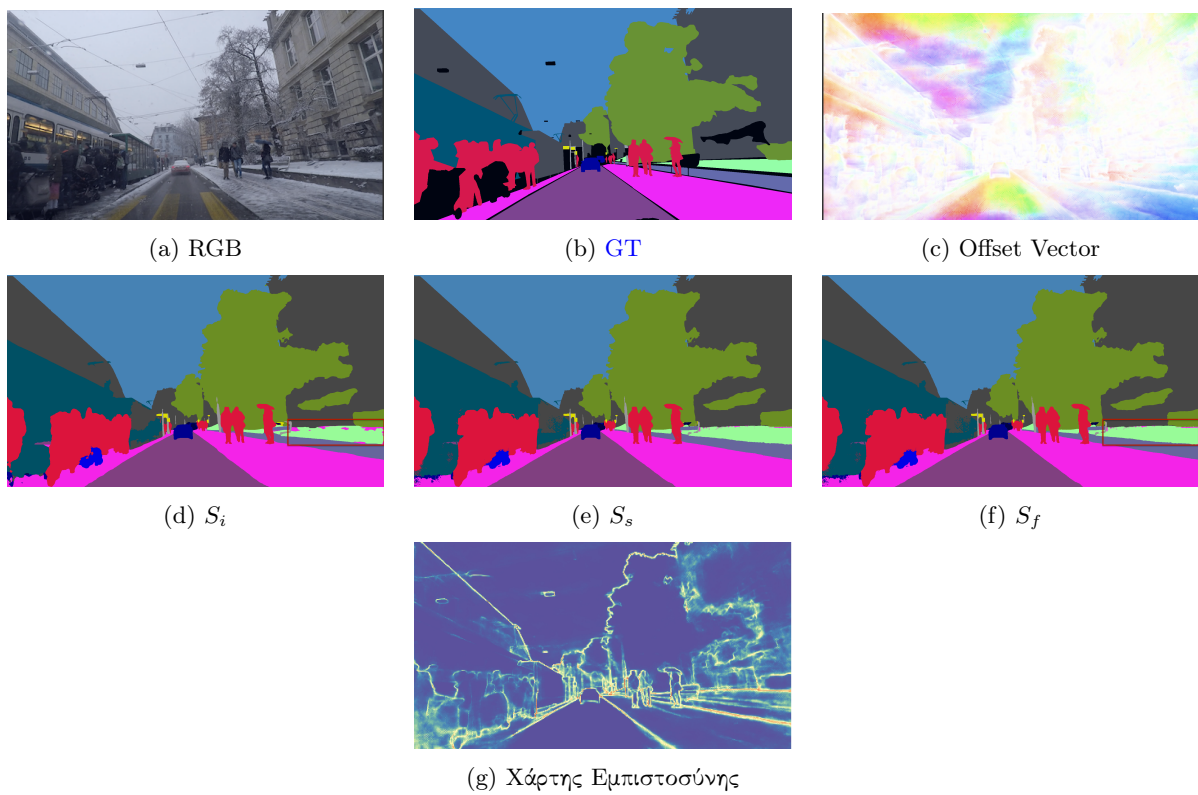
Σχήμα F .12: Ποιοτικά Αποτελέσματα στο Cityscapes: Δεύτερο Παράδειγμα

πλάισιο. Αυτό γίνεται, επειδή στο συγκεκριμένο σημείο το μοντέλο μας δείχνει μεγαλύτερη εμπιστοσύνη στη σωστή πρόβλεψη του  $S_i$ . Σχετικά με τις Εικόνες του Σχήματος F .14, παρατηρούμε μια ασυνέχεια στο τμήμα του εδάφους στο κόκκινο πλαίσιο της  $S_i$  πρόβλεψης, η οποία εξαλείφεται τελικά από την  $S_f$ .

Γενικά παρατηρούμε καλύτερη οπτικοποίηση των διανυσμάτων μετατόπισης στο Cityscapes σε αντίθεση με το ACDC. Στο τελευταίο παρατηρούμε σε πολλά σημεία μικρές τιμές των διανυσμάτων (άσπρο χρώμα). Αυτό είναι λογικό, γιατί στα σημεία αυτά έχουμε περιορισμένη ορατότητα λόγω των αντίξωων συνθηκών.



Σχήμα F .13: Ποιοτικά Αποτελέσματα στο ACDC: Πρώτο παράδειγμα



Σχήμα F .14: Ποιοτικά Αποτελέσματα στο ACDC: Δεύτερο παράδειγμα

## Πειράματα του Baseline Μοντέλου

Αρχικά, εκπαιδεύσαμε το αρχικό HRNetV2 μοντέλο από την αρχή και στα 2 σύνολα δεδομένων, Cityscapes και ACDC, προκειμένου να αναπαράγουμε τα αποτελέσματα που είναι δημοσιευμένα στο [6] και [11] αντίστοιχα. Χρησιμοποιήσαμε τις default παραμέτρους που όρισαν οι συγγραφείς. Το αρχικό μοντέλο εκπαιδεύτηκε σε 4 GPUs και η εκπαίδευση του ολοκληρώνεται μετά από 80 ώρες στο Cityscapes και 35 ώρες στο ACDC. Όπως βλέπουμε από τον Πίνακα 1, τα αποτελέσματα αναπαράγονται επιτυχώς στο Cityscapes. Σχετικά με το ACDC, το μοντέλο που περιγράφεται στο [11] έχει αρχικοποιηθεί με τα βάρη που έχουν προκύψει από την εκπαίδευση του μοντέλου στο Cityscapes. Αντιθέτως, το μοντέλο μας έχει αρχικοποιηθεί, όπως και στην περίπτωση του Cityscapes, με βάρη που έχουν προκύψει από την εκπαίδευση του μοντέλου στο Imagenet. Στα ακόλουθα πειράματα θα χρησιμοποιήσουμε την τελευταία αρχικοποίηση, προκειμένου να πετύχουμε ανταγωνιστικά αποτελέσματα.

Model	Backbone	Dataset	MeanIU
HRNetV2 [6]	HRNetV2-W48	Cityscapes [7]	80.4
HRNetV2 (Source Code)	HRNetV2-W48	Cityscapes [7]	80.5
HRNetV2 [6]	HRNetV2-W48	ACDC [11]	75.0
HRNetV2 (Source Code)	HRNetV2-W48	ACDC [11]	70.5

Πίνακας 1: Αποτελέσματα αναπαραγωγής του μοντέλου μας στο Test Set

## Σύγκριση με State of the Art μοντέλα

### Cityscapes

Τα αποτελέσματα του μοντέλου μας στο Cityscapes φαίνονται παρακάτω. Αξίζει να αναφέρουμε ότι πετυχαίνουμε σε παρόμοιο χρόνο εκπαίδευσης καλύτερα αποτελέσματα από το αρχικό HRNet, ξεπερνώντας προηγούμενες state-of-the-art αρχιτεκτονικές και στις 4 μετρικές τόσο στο val όσο και στο test set.

- **Αποτελέσματα στο val set:** Ο πίνακας 2 συγκρίνει το μοντέλο μας με το αρχικό HRNetV2 στο val test του Cityscapes, ως προς τον αριθμό των παραμέτρων, την υπολογιστική πολυπλοκότητα και το mIoU. Το μοντέλο μας πετυχαίνει καλύτερη απόδοση και, συγκεκριμένα, 0.6 μονάδες πάνω από το HRNetV2.

Model	Backbone	#param.	GFLOPs	mIoU
HRNetV2 [6]	HRNetV2-W48	65.9M	174	81.8
Ours	HRNetV2-W48	98.8M	234,7	<b>82.4</b>

Πίνακας 2: Αποτελέσματα στο Cityscapes val test (multi-scale και flipping). Τα GFLOPs υπολογίζονται με βάση την εικόνα εισόδου (1024 × 2048). Μετρίωνται μόνο Συνελικτικά και Γραμμικά επίπεδα μόνο.

- **Αποτελέσματα στο test set:** Ο πίνακας 3 συγκρίνει τη μέθοδο μας με προηγούμενες state-of-the-art μεθόδους στο Cityscapes test set. Έχουν αξιολογηθεί δύο περιπτώσεις: Η πρώτη αφορά τα μοντέλα που έχουν εκπαιδευτεί μόνο στο train set. Η δεύτερη αφορά τα μοντέλα που έχουν εκπαιδευτεί τόσο στο train όσο και στο val set. Και στις δύο περιπτώσεις το μοντέλο μας πετυχαίνει καλύτερη επίδοση, παρόλο που είναι εκπαιδευμένο **μόνο** στο train set. Συγκεκριμένα, πετυχαίνουμε καλύτερη επίδοση κατά 1.7% στο mIoU, 4% στο iIoU cla., 0.4 % στο IoU cat. και 1.7% στο iIoU cat.

Ο πίνακας 4 συγκρίνει αναλυτικά την προσέγγισή μας με αυτή του αρχικού HRNetV2 ανα κλάση. Όπως μπορούμε να δούμε, η μέθοδος μας πετυχαίνει καλύτερα αποτελέσματα στην πλειονότητα των κλάσεων. Το μοντέλο μας μαθαίνει μια αναπαράσταση των διαφόρων αντικειμένων, η οποία μπορεί να οφελήσει την ολική απόδοση του δικτύου.

Τα ποιοτικά αποτελέσματα στο Cityscapes υποστηρίζουν αυτά τα ευρήματα, όπως φαίνεται στο Σχήμα F.15. Πιο συγκεκριμένα, από τα αριστερά προς τα δεξιά, εικονίζονται η εικόνα εισόδου, η έξοδος του αρχικού HRNet και η έξοδος του μοντέλου μας. Όσον αφορά την πρώτη γραμμή, παρατηρούμε ότι το μοντέλο μας προσπαθεί να μεγεθύνει σωστά τα τμήμα του πεζοδρομίου, όπως φαίνεται στο κόκκινο πλαίσιο. Στη δεύτερη σειρά, το μοντέλο μας επιμηκύνει σωστά την ταμπέλα, την κολώνα και το πεζοδρόμιο (κόκκινο, πράσινο, μπλέ πλαίσιο αντίστοιχα). Δυστυχώς, όπως φαίνεται από το κίτρινο πλαίσιο, το αρχικό μοντέλο



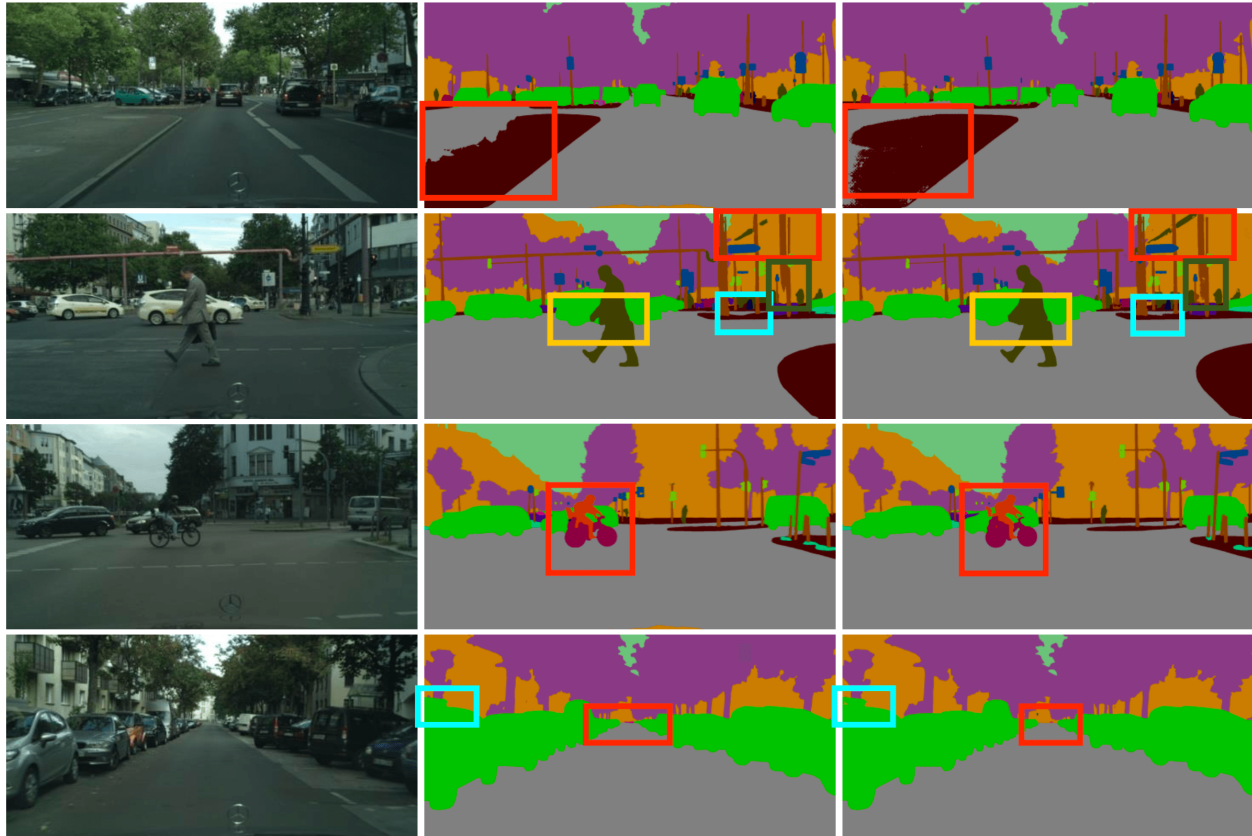
Model	Backbone	mIoU	iIoU cla.	IoU cat.	iIoU cat.
<i>Model trained on the train set</i>					
PSPNet [12]	D-ResNet-101	78.4	56.7	90.6	78.6
PSANet [13]	D-ResNet-101	78.6	-	-	-
PAN [14]	D-ResNet-101	78.6	-	-	-
AFF [15]	D-ResNet-101	79.1	-	-	-
HRNetV2 [6]	HRNetV2-W48	80.4	59.2	91.5	80.8
Ours	HRNetV2-W48	<b>81.8</b>	<b>61.6</b>	<b>91.9</b>	<b>82.2</b>
<i>Model trained on the train +val set</i>					
GridNet [16]	-	69.5	44.1	87.9	71.1
LRR-4x [17]	-	69.7	48.0	88.2	74.7
DeepLab [18]	D-ResNet-101	70.4	42.6	86.4	67.7
LC [19]	-	71.1	-	-	-
Piecewise [20]	VGG-16	71.6	51.7	87.3	74.1
FRRN [21]	-	71.8	45.5	88.9	75.1
RefineNet [22]	ResNet-101	73.6	47.2	87.9	70.6
PEARL [23]	D-ResNet-101	75.4	51.6	89.2	75.1
DSSPN [24]	D-ResNet-101	76.6	56.2	89.6	77.8
LKM [25]	ResNet-152	76.9	-	-	-
SAC [26]	D-ResNet-101	78.1	-	-	-
DepthSeg [27]	D-ResNet-101	78.2	-	-	-
ResNet38 [28]	WRResNet-38	78.4	59.1	90.9	78.1
BiSeNet [29]	ResNet-101	78.9	-	-	-
DFN [30]	ResNet-101	79.3	-	-	-
PSANet [13]	D-ResNet-101	80.1	-	-	-
PADNet [31]	D-ResNet-101	80.3	58.8	90.8	78.5
DenseASPP [12]	WRResNet-161	80.6	59.1	90.9	78.1
DANet [32]	D-ResNet-101	81.5	-	-	-
HRNetV2 [6]	HRNetV2-W48	81.6	<b>61.8</b>	<b>92.1</b>	82.2
Ours (only on train set)	HRNetV2-W48	<b>81.8</b>	61.6	91.9	<b>82.2</b>

Πίνακας 3: Αποτελέσματα στο Cityscapes test set. Τα αποτελέσματα μας είναι ανώτερα όσομ αφορά τις 4 μετρικές αξιολόγησης. D-ResNet-101 = Dilated-ResNet-101. Συγκρίνουμε τη μέθοδο μας με προηγούμενες SOTA μεθόδους, όπως στο [6]

Method	road	sidew.	build.	wall	fence	pole	light	sign	veget.	terrain	sky	person	rider	car	truck	bus	train	motorc.	bicycle	mIoU
HRNetV2 [6]	98.73	87.49	93.65	56.48	61.57	71.57	78.76	81.81	93.99	74.11	95.68	87.95	73.72	96.35	69.94	82.52	76.93	70.88	78.02	80.4
Ours	<b>98.74</b>	87.41	<b>93.79</b>	<b>61.65</b>	<b>64.00</b>	71.35	<b>78.98</b>	<b>81.65</b>	<b>94.00</b>	<b>73.42</b>	<b>95.81</b>	<b>87.99</b>	<b>74.36</b>	<b>96.42</b>	<b>74.76</b>	<b>87.70</b>	<b>82.83</b>	<b>71.77</b>	77.86	<b>81.8</b>

Πίνακας 4: Ανα κλάση αποτελέσματα στο Cityscapes test set

προβλέπει καλύτερα το σχήμα των χεριών του ανθρώπου. Συγκεκριμένα, ο άνδρας κρατά μια εφημερίδα και το μοντέλο μας λαθεμένα ταξινομεί τα pixel της εφημερίδας στην κλάση του ανθρώπου. Ωστόσο, συνολικά, το μοντέλο μας έχει πιο ακριβή πρόβλεψη. Σχετικά με την τρίτο set εικόνων, το μοντέλο μας εξαλείφει το πόδι του ποδηλάτη και διευρύνει το τμήμα του αμαξιού που βρίσκεται στο προσκήνιο. Έτσι, πετυχαίνει ένα καλύτερο αποτέλεσμα. Τελικά, στο τελευταίο set εικόνων, το μοντέλο μας διορθώνει μερικές λάθος προβλέψεις που έχουν γίνει από το αρχικό μοντέλο, συμπεριλαμβάνοντας την μείωση του τμήματος του αμαξιού ( βλέπε μπλέ πλαίσιο) και του τμήματος του χάρτη (κόκκινο πλαίσιο). Συνοψίζοντας, απο τα παραπάνω φαίνεται ότι το μοντέλο μας ξεπερνά την απόδοση του αρχικού HRNet.



Σχήμα F .15: Ποιοτικά αποτελέσματα επιλεγμένων παραδειγμάτων στο Cityscapes. Απο τα αριστερά προς τα δεξιά: εικόνα εισόδου, αρχικό HRNet και δικό μας μοντέλο

## ACDC

Τα αποτελέσματα του μοντέλου μας στο **ACDC** φαίνονται παρακάτω. Αξίζει να αναφέρουμε ότι και σε αυτό το dataset πετυχαίνουμε σε παρόμοιες ώρες εκπαίδευσης καλύτερα αποτελέσματα από το αρχικό HRNet, ξεπερνώντας προηγούμενες state-of-the-art αρχιτεκτονικές. Στα ακόλουθα πειράματα, σαν αρχικό μοντέλο χρησιμοποιείται αυτό που αρχικοποιήθηκε με τα βάρη που προέκυψαν από την εκπαίδευση του αρχικού μοντέλου στο Imagenet.

- **Αποτελέσματα στο val set:** Ο πίνακας 5 συγκρίνει το μοντέλο μας με το αρχικό HRNetV2 στο val test του **ACDC**, ως προς τον αριθμό των παραμέτρων, την υπολογιστική πολυπλοκότητα και το **mIoU**. Το μοντέλο μας πετυχαίνει καλύτερη απόδοση και, συγκεκριμένα, 0.41 μονάδες πάνω από το HRNetV2.

Model	Backbone	#param.	GFLOPs	mIoU
HRNetV2	HRNetV2-W48	65.9M	172.9	75.50
Ours	HRNetV2-W48	98.8M	233.1	<b>75.91</b>

Πίνακας 5: Αποτελέσματα στο ACDC val test (multi-scale και flipping). Τα GFLOPs υπολογίζονται με βάση την εικόνα εισόδου (1080 × 1920). Μετρίωνται μόνο Συνελικτικά και Γραμμικά επίπεδα μόνο.

- **Αποτελέσματα στο test set:** Ο πίνακας 6 συγκρίνει τη μέθοδο μας με προηγούμενες state-of-the-art μεθόδους στο **ACDC** test set και συγκεκριμένα σε όλες τις συνθήκες (All Condition), καθώς είναι εκπαιδευμένο σε όλες τις συνθήκες και όχι μόνο σε μία. Όπως παρατηρούμε, το μοντέλο μας πετυχαίνει καλύτερη επίδοση από τις άλλες μεθόδους. Συγκεκριμένα, πετυχαίνουμε καλύτερη επίδοση κατά 2.5% στο **mIoU**.

Model	mIoU
RefineNet [22]	65.3
DeepLabv2 [18]	55.3
DeepLabv3+ [33]	70.0
HRNetV2 [6]	70.5
Ours	<b>73</b>

Πίνακας 6: Αποτελέσματα στο ACDC test set. Τα αποτελέσματα μας είναι ανώτερα όσον αφορά το **mIoU**. Συγκρίνουμε τη μέθοδο μας με προηγούμενες SOTA μεθόδους, όπως στο [11]

Ο πίνακας 7 συγκρίνει αναλυτικά την προσέγγισή μας με άλλες αρχιτεκτονικές ανά κλάση. Όπως μπορούμε να δούμε, η μέθοδος μας πετυχαίνει καλύτερα αποτελέσματα σε όλες τις κλάσεις. Συγκεκριμένα, παρατηρούμε το εξής:

- Στο χιόνι, ο δρόμος και το πεζοδρόμιο έχουν χαμηλή τιμή, που μπορεί να αποδοθεί σε αδυναμία του αρχικού μοντέλου να ξεχωρίσει τις δυο κλάσεις λόγω παρόμοιας εμφάνισης. Παρ’ όλα αυτά, το μοντέλο μας βελτιώνει σημαντικά κατά 1.8 % την επίδοση στη κλάση του πεζοδρομίου.
- Είναι πιο δύσκολο για το μοντέλο να ξεχωρίσει τις κλάσεις κατά τη διάρκεια της νύχτας, που τα αντικείμενα είναι συνήθως σκοτεινά ή κακώς φωτισμένα. Τέτοια κλάσεις είναι για παράδειγμα τα κτήρια, τα δέντρα, ταμπέλες και ο ουρανός. Αυτή η συμπεριφορά παρατηρείται επίσης και στα διανύσματα μετατόπισης, καθώς έχουν μικρές τιμές όταν η ορατότητα είναι περιορισμένη.
- Σε συνθήκες ομίχλης, η επίδοση του μοντέλου σε κλάσεις που περιέχουν συνήθως μικρά αντικείμενα, όπως άνθρωπος, αναβάτης και ποδηλάτο, είναι χαμηλή. Αυτό γίνεται, λόγω του συνδυασμού της μείωσης της αντίθεσης και της χαμηλής ανάλυσης σε παραδείγματα αυτών των κλάσεων, που είναι μακριά από την κάμερα.

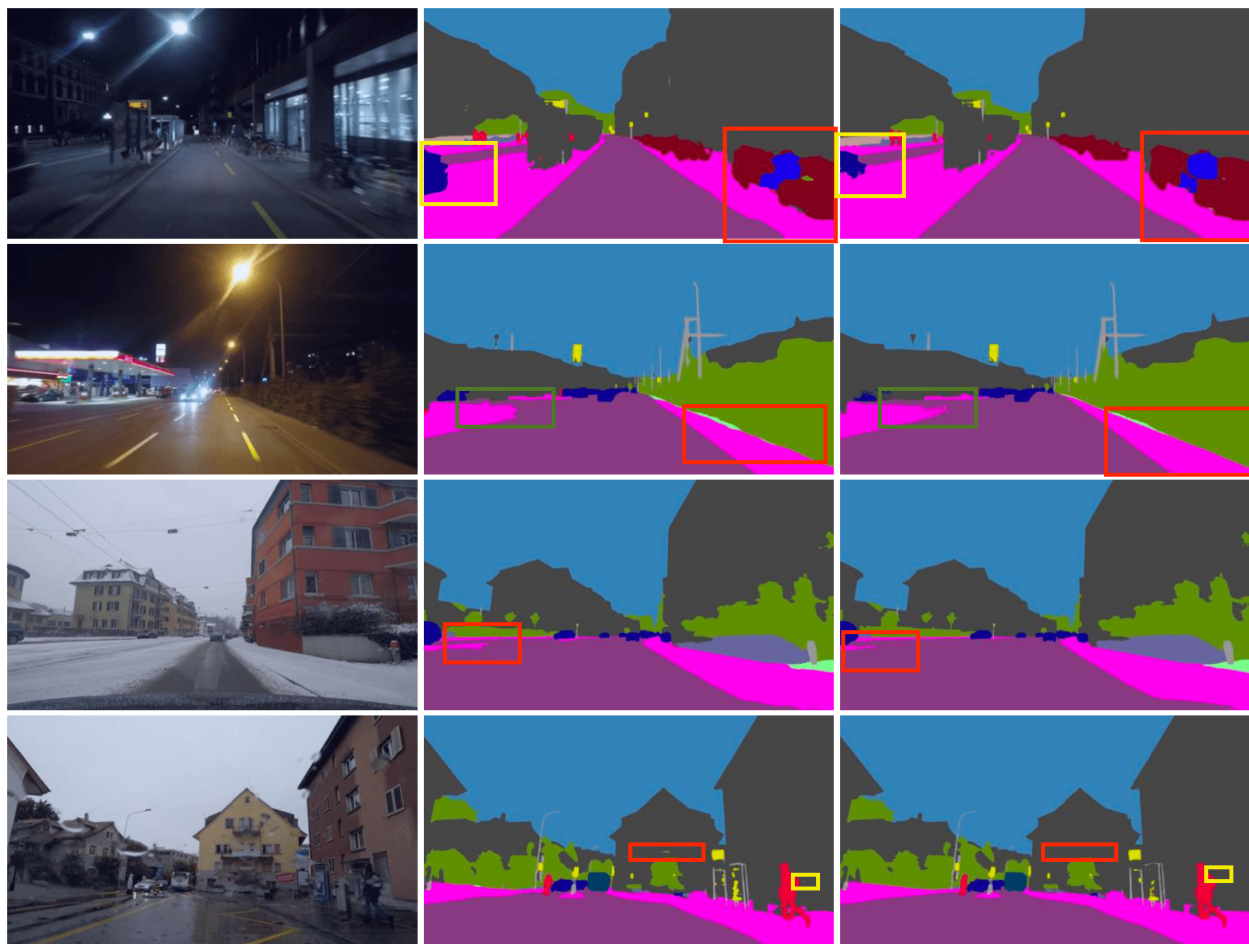
Τα ποιοτικά αποτελέσματα στο **ACDC** υποστηρίζουν αυτά τα ευρήματα, όπως φαίνεται στο σχήμα **F.16**. Πιο συγκεκριμένα, από τα αριστερά προς τα δεξιά, εικονίζονται η εικόνα εισόδου, η έξοδος του αρχικού HRNet και η έξοδος του μοντέλου μας. Όσον αφορά την πρώτη γραμμή, βλέπουμε ότι το μοντέλο μας μεγεθύνει το τμήμα του ποδηλάτου, διορθώνοντας κάποια λάθος pixels (βλέπε κόκκινο πλαίσιο). Δυστυχώς, όπως βλέπουμε από το κίτρινο πλαίσιο, το αρχικό μοντέλο πετυχαίνει καλύτερη πρόβλεψη, αφού στη συγκεκριμένη περιοχή εικονίζεται ένα αμάξι. Σχετικά με τη δεύτερη σειρά, παρατηρούμε ότι το μοντέλο μας προσπαθεί να μεγαλώσει ορθώς τη κλάση του πεζοδρομίου, τόσο στο κόκκινο όσο και στο



Method	road	sidew.	build.	wall	fence	pole	light	sign	veget.	terrain	sky	person	rider	car	truck	bus	train	motorc.	bicycle	mIoU
RefineNet [22]	92.5	71.2	86.2	39	44.0	53.2	68.8	66.0	85.1	59.3	94.9	65.2	38.5	85.8	53.8	59.7	76.2	47.5	54.5	65.3
DeepLabv2 [18]	88.0	62.3	80.8	37.0	35.1	33.9	49.8	49.5	80.1	50.7	92.5	51.1	26.5	79.9	49.0	41.1	72.2	26.5	44.2	55.3
DeepLabv3+ [33]	93.4	74.8	89.2	53.0	49.0	58.7	71.1	67.4	87.8	62.7	95.9	69.7	36.0	88.1	67.7	71.8	85.1	48.0	59.8	70
HRNetV2 [6]	95.3	80.3	90.5	52.0	53.1	65.1	78.2	74.2	89.2	68.4	96.7	70.6	36.1	88.2	55.9	54.3	88.0	43.8	58.9	70.5
Ours	<b>95.8</b>	<b>82.1</b>	<b>91.3</b>	<b>55.8</b>	<b>54.6</b>	<b>67.6</b>	<b>80.5</b>	<b>77.3</b>	<b>89.7</b>	<b>69.5</b>	<b>96.8</b>	<b>73.4</b>	<b>39.1</b>	<b>89.5</b>	<b>61.9</b>	<b>65</b>	<b>89.4</b>	<b>47.2</b>	<b>60.6</b>	<b>73</b>

Πίνακας 7: Ανα κλάση αποτελέσματα στο ACDC test set

πράσινο πλαίσιο, και μειώνει το τμήμα του εδάφους, που έχει προβλεφθεί λαθεμένα από το αρχικό μοντέλο. Σχετικά με το τρίτο σύνολο εικόνων, η προσέγγισή μας μειώνει ορθώς τη περιοχή του πεζοδρομίου (βλέπε κόκκινο πλαίσιο), καθώς στην αρχική εικόνα δεν υπάρχει τέτοια περιοχή. Τέλος, όσον αφορά το τελευταίο σύνολο εικόνων, το αρχικό μοντέλο ταξινομεί λαθεμένα την πινακίδα του σπιτιού στην κλάση των πινακίδων οδικής κυκλοφορίας (βλέπε κόκκινο πλαίσιο). Αντιθέτως, το μοντέλο μας διορθώνει όχι μόνο αυτό το λάθος, αλλά και μια ασυνέχεια που παρατηρείται στο κίτρινο πλαίσιο. Συνοψίζοντας, το μοντέλο μας ξεπερνά την επίδοση του αρχικού.



Σχήμα F .16: Ποιοτικά αποτελέσματα επιλεγμένων παραδειγμάτων στο ACDC. Απο τα αριστερά προς τα δεξιά: εικόνα εισόδου, αρχικό HRNet και δικό μας μοντέλο

## Συμπεράσματα

Συνοψίζοντας, μέσω αναλυτικών ποιοτικών και ποσοτικών συγκρίσεων καταφέραμε να δείξουμε τα εμφανή πλεονεκτήματα της μεθόδου μας έναντι προηγούμενων state-of-the-art μεθόδων στο task αυτό. Οι πίνακες 7 και 4 δείχνουν όχι μόνο ότι το μοντέλο μας επιτυγχάνει καλύτερη συνολική επίδοση, αλλά και καλύτερα

αποτελέσματα ανα κλάση στην πλειονότητα των κλάσεων σε παρόμοιο χρόνο εκπαίδευσης. Όπως αναφέραμε παραπάνω, η μέθοδος μας μοιράζεται πληροφορία από θέσεις των seed pixels και βελτιώνει τις προβλεπόμενες κλάσεις, αφού όχι μόνο πετυχαίνει ικανοποιητικά αποτελέσματα κοντά στις GT εικόνες, αλλά και βελτιώνει τις προβλέψεις που προέρχονται από το αρχικό μοντέλο του HRNetV2. Συγκεκριμένα, ταξινομεί μερικά λάθος προβλεπόμενα pixels στις σωστές κλάσεις. Έτσι, εξαλείφει ασυνέχειες και βελτιώνει το σχήμα καθώς και τη μορφή των αντίστοιχων τμημάτων, οδηγώντας σε πιο ρεαλιστικά αποτελέσματα.

## Συνεισφορές και μελλονικές προεκτάσεις

Στη παρούσα διπλωματική, μελέταμε το πρόβλημα της σημασιολογικής κατάτμησης, ενός από τα θεμελιώδη θέματα της Όρασης Υπολογιστών. Η αναγνώριση ενός αντικειμένου μέσα σε μια RGB εικόνα συνεπάγεται το καθορισμό μερικών χαρακτηριστικών που αφορούν το χρώμα και την υφή. Εφέ όπως η αλλαγή χρώματος, λόγω της απορρόφησης του φωτός, η παραμόρφωση των φακών και οι χρωματικές αλλαγές μπορούν να αλλάξουν την εμφάνιση των αντικειμένων και να επηρεάσουν σε μεγάλο βαθμό την αντίληψη τόσο του χρήστη όσο και των αλγορίθμων αναγνώρισης. Όλο και πιο σύνθετες μέθοδοι προσπαθούν να βελτιώσουν τις επιδόσεις των μοντέλων. Παρόλο που η Τεχνητή Νοημοσύνη έχει να διανύσει μεγάλο δρόμο ακόμα, προκειμένου να αναπτύξει μοντέλα συγκρίσιμα με τον άνθρωπο σε αυτά τα προβλήματα, μερικοί ερευνητές συνεχώς ανακαλύπτουν state-of-the-art μοντέλα, ανεβάζοντας το επίπεδο όλο και πιο ψηλά. Οι περισσότερες σχετικές εργασίες επικεντρώνονται σε αλλαγές στην αρχιτεκτονική στα χρησιμοποιούμενα δίκτυα προκειμένου να συνδυάσουν καλύτερα το περιεχόμενο ολόκληρης της εικόνας διατηρώντας παράλληλα τη λεπτομέρεια σε τοπικό επίπεδο και χρησιμοποιώντας ένα loss υπολογιζόμενο σε μεμονωμένα pixels. Ο σχεδιασμός πιο σύνθετων losses, που λαμβάνουν υπόψη τη δομή που περιέχεται στις σημασιολογικές ετικέτες, είναι αντικείμενο μεγάλης προσοχής. Προκειμένου να παράξουμε αποτελέσματα τα οποία αντικατοπτρίζουν καλύτερα την κανονικότητα των γνήσιων αναπαραστάσεων, μελέταμε προηγούμενες έρευνες στο κλάδο της σημασιολογικής κατάτμησης και προτείνουμε μια νέα προσέγγιση στο πρόβλημα.

Προηγουμένως, αναλύσαμε πλήρως την προτεινόμενη μέθοδο. Βασιζόμενοι στη γνώση σχετικά με την υψηλού επιπέδου κανονικότητα των πραγματικών σκηνών, προτείνουμε μια νέα μέθοδο για τη βελτίωση των προβλεπόμενων κλάσεων, μέσω της εκμάθησης της επιλεκτικής αξιοποίησης των πληροφοριών από τα συνεπίεδα pixels. Συγκεκριμένα, βασιζόμαστε στην ιδέα ότι για κάθε pixel υπάρχει ένα seed pixel, το οποίο ανείχει στην ίδια κλάση με το προηγούμενο. Σαν αποτέλεσμα, σχεδιάζουμε ένα νευρωνικό δίκτυο με δύο κεφαλές. Η πρώτη κεφαλή παράγει τις προβλεπόμενες κλάσεις για κάθε pixel, ενώ η δεύτερη παράγει ένα πυκνό πεδίο διανυσμάτων μετατόπισης (offset vectors) που προσδιορίζει τις θέσεις των seed pixels. Οι προβλέψεις των seed pixels χρησιμοποιούνται στη συνέχεια για να προβλέψουν την κλάση σε κάθε pixel. Προκειμένου να ληφθούν υπόψη πιθανές αποκλίσεις από την ακριβή τοπική ομαλότητα, η προκύπτουσα πρόβλεψη συγχωνεύεται προσαρμοστικά με την αρχική πρόβλεψη από την πρώτη κεφαλή χρησιμοποιώντας έναν χάρτη εμπιστοσύνης (confidence map), τον οποίο μαθαίνει το μοντέλο. Η συνολική αρχιτεκτονική έχει υλοποιηθεί στο μοντέλο HRNetV2, ένα state-of-the-art μοντέλο στο σύνολο δεδομένων Cityscapes. Η υπεροχή της μεθόδου μας έναντι άλλων προηγούμενων SOTA μεθόδων αποδείχθηκε μέσω αναλυτικής ποιοτικής και ποσοτικής πειραματικής αξιολόγησης τόσο στο Cityscapes όσο και στο ACDC σύνολο δεδομένων. Και τα δύο σύνολα δεδομένων είναι προκλητικά. Το πρώτο χρησιμοποιείται ευρέως σε προβλήματα σημασιολογικής κατάτμησης και το δεύτερο περιέχει εικόνες που έχουν ληφθεί σε αντίξοες συνθήκες (ομίχλη, νύχτα, βροχή και χιόνι).

Η μέθοδος μας βελτιώνει σε μεγάλο βαθμό τις προβλέψεις του αρχικού μοντέλου, αφού όχι μόνο πετυχαίνει καλύτερα συνολικά αποτελέσματα αλλά και καλύτερα αποτελέσματα ανά κλάση στην πλειονότητα των κλάσεων σε παρόμοιο χρόνο εκπαίδευσης. Ουσιαστικά, μαθαίνει μια αναπαράσταση των διαφόρων αντικειμένων, η οποία μπορεί να οφελήσει την ολική απόδοση του δικτύου. Έτσι, όχι μόνο πετυχαίνει ικανοποιητικά αποτελέσματα κοντά στις GT εικόνες, αλλά και βελτιώνει τις προβλέψεις που προέρχονται από το αρχικό μοντέλο του HRNetV2. Συγκεκριμένα, ταξινομεί μερικά λάθος προβλεπόμενα pixels στις σωστές κλάσεις. Έτσι, εξαλείφει ασυνέχειες και βελτιώνει το σχήμα καθώς και τη μορφή των αντίστοιχων τμημάτων, οδηγώντας σε πιο ρεαλιστικά αποτελέσματα. Αυτή είναι μια πολύ σημαντική συνεισφορά, που ανοίγει νέους δρόμους για πραγματικές καθημερινές εφαρμογές, όπως η εφαρμογή της ιδέας αυτή σε συστήματα Αυτόνομης Οδήγησης ή στην Ιατρική.

Τέλος, παρόλο επιτύχαμε πολύ υποσχόμενα αποτελέσματα σε μια ποικιλία από σύνολα δεδομένων, εξακολουθούν να υπάρχουν περιορισμοί, που μπορούν να οδηγήσουν σε μελλοντικές προεκτάσεις της μεθόδου. Παρακάτω αναφέρουμε συνοπτικά μερικούς από αυτούς:

- **Εφαρμογή της μεθόδου μας στους Visual Transformers.** Στην παρούσα χρονική στιγμή, οι ολοκαίνουργιοι Visual Transformers πετυχαίνουν state-of-the-art αποτελέσματα στο task της σημασιολογικής κατάτμησης. Σε επόμενο στάδιο, θα εφαρμόσουμε τη μέθοδο μας στον ViT-Adapter [34], αναμένοντας νέα state-of-the-art αποτελέσματα για το task αυτό.
- **Συνδυασμός της μεθόδου μας με άλλες τεχνικές.** Θα μπορούσαμε να μελετήσουμε τον συνδυασμό της μεθόδου μας σε άλλες τεχνικές σημασιολογικής κατάτμησης και instance segmentation σε μια πληθώρα από σύνολα δεδομένων. Για παράδειγμα, μπορεί να εφαρμοστεί στο συνδυασμό του HRNet με το Object-Contextual Representations (OCR) [9].
- **Μέθοδος βασισμένη στο ultimate erosion.** Όπως είδαμε στις οπτικοποιήσεις των διανυσμάτων μετατόπισης, μερικά διανύσματα αποτυγχάνουν να δείξουν στο κέντρο του αντίστοιχου αντικειμένου. Σαν αποτέλεσμα, η κατανομή τους στο συγκεκριμένο τμήμα δεν είναι κανονική. Προκειμένου να αντιμετωπίσουμε αυτό το πρόβλημα, θα μπορούσαμε να χρησιμοποιήσουμε την συνάρτηση της ultimate erosion. Συγκεκριμένα, μέσα από αυτή την συνάρτηση, θα μπορούσαμε να βρούμε σε κάθε GT εικόνα το κέντρο κάθε κλάσης και μετά να χρησιμοποιήσουμε αυτό το σύνολο συντεταγμένων κατά τη διάρκεια της εκπαίδευσης του μοντέλου. Έτσι, θα αναγκάσουμε τα διανύσματα μετατόπισης ενός συγκεκριμένου τμήματος να δείχνουν στο κέντρο του, οδηγώντας πιθανόν έτσι σε καλύτερες προβλέψεις.



# Chapter 1

## Introduction

---

1	Semantic Segmentation . . . . .	<b>28</b>
1.1	Definition and Applications . . . . .	28
1.2	Deep Neural Networks . . . . .	29
2	Outline . . . . .	<b>29</b>

---

# 1 Semantic Segmentation

## 1.1 Definition and Applications

Semantic segmentation is a problem of assigning a class label to each pixel. The task is also commonly referred as dense prediction, since the label is predicted for every pixel in the picture. It can be formulated as:

### Definition 1.1: Semantic Segmentation

Semantic Segmentation requires learning a dense mapping  $f_\theta : I(u, v) \rightarrow S(u, v)$  where

- $I$  is the input image with spatial dimensions  $H \times W$
- $S$  is the corresponding output prediction map of the same resolution
- $(u, v)$  are pixel coordinates in the image space
- $\theta$  are the parameters of the mapping  $f$

In the supervised version of the task, a ground-truth semantically segmented map  $H$  is available for each image  $I$  at training time. During training, the parameters  $\theta$  are optimized such that the function  $f_\theta$  minimizes the difference between the predicted image and the ground-truth image over the training set  $T$ . In other words, this can be formalized as

$$\min_{\theta} \sum_{(I, H) \in T} \mathcal{L}(f_\theta(I), H) \quad (1.1)$$

where  $\mathcal{L}$  is a loss function that penalizes deviations between the prediction and the ground truth.

Many applications, such as medical image analysis, robotics, surveillance, autonomous driving, and many more, require semantic information from images at the pixel level. As far as the autonomous driving is concerned, an example is shown in Figure 1.1. It illustrates an example image (Fig. 1.1a) and a semantic segmentation result (Fig. 1.1c). The objective is illustrated by the ground truth image (Fig. 1.1b). The class labels may be a person, a car, or a road, or any other basic pattern found in the image. It is one of a few algorithms that help us to analyze the context of an environment we are familiar with. As a result, semantic segmentation is commonly employed in autonomous cars, where the context of the environment is critical.

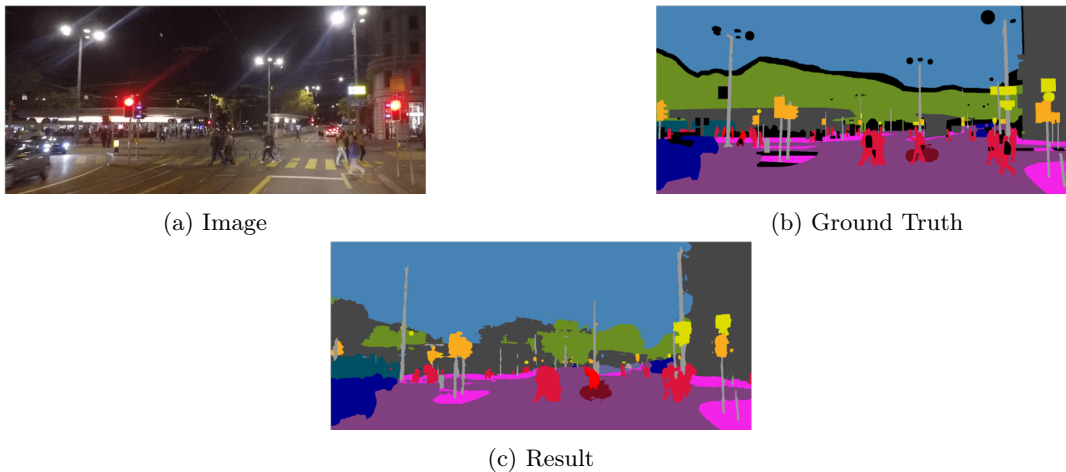


Figure 1.1: Example of a semantic image segmentation algorithm

### Challenges

Semantic Segmentation is one of the hottest topics in research. While modern techniques have shown promising improvements, they still fall short of human-level performance. Humans can recognize objects with little effort, even when they vary in views, scale, lighting, or they are translated or rotated. Even when objects are partially obscured from vision, they may be identified.

In order to automatically overtake these obstacles using computers, state-of-the-art semantic segmentation methods depend on machine learning techniques to understand the multiple representations of things from provided images. However, actual approaches have their own set of flaws as well. In order to attain the precision of state-of-the-art approaches, pixel-level annotated images are necessary. Unfortunately, these images are many times restricted for many applications or simply unavailable.

Furthermore, if we learn the object representation from a finite collection of labeled images, the model may obtain satisfactory results on samples that seem similar to those in the training set, but the algorithm's ability to generalize to additional images is not guaranteed. This problem is also known as overfitting.

## 1 .2 Deep Neural Networks

Deep learning is a machine learning technique used to build Artificial Intelligence (AI) systems. It is based on the concept of Artificial Neural Networks (ANNs) , which are meant to process large amounts of data through numerous layers of neurons to perform complicated analysis.

### Deep Convolutional Neural Networks (DCNNs)

There is a wide variety of Deep Neural Networks (DNNs). DCNNs are the most prevalent form of image and video pattern recognition system. Traditional artificial neural networks have been developed into DCNNs, which use a three-dimensional neural pattern inspired by animal visual brain.

DCNNs are mostly used for object recognition, image classification, and semantic segmentation, but they are also sometimes utilized for natural language processing.

The strength of DCNNs lies in their layering. A DCNN processes the Red, Green, and Blue parts of an image simultaneously using a three-dimensional neural network. When compared to standard feed forward neural networks, this significantly reduces the number of artificial neurons required to analyze an image.

DCNNs receive images as an input and use them to train a classifier. Instead of matrix multiplication, the network uses a particular mathematical process known as "convolution."

A convolutional network's architecture, generally, consists of four layers: convolutional, pooling, activation and fully connected one [1]. An example of this architecture is shown in Figure 1 .2 .

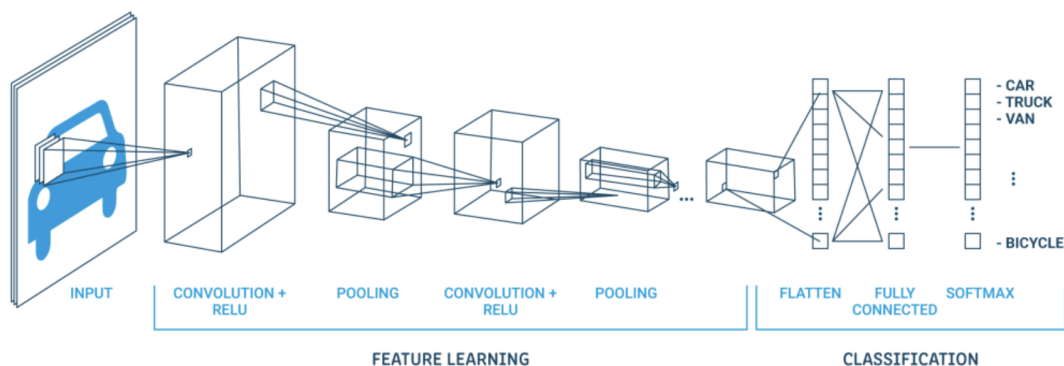


Figure 1 .2: An example of a CNN architecture

## 2 Outline

This thesis is divided in 5 chapters. The following is an overview of the contents of each chapter:

- In chapter 1, we gave an introduction to semantic image segmentation and discussed issues that may arise. In addition, it was given a brief introduction to deep learning .

- In chapter 2, we present a review of the background relevant to deep learning and the task of semantic segmentation.
- In chapter 3, we analyze related work on this task and address current state-of-the-art approaches.
- In chapter 4, we propose a new method for improving semantic segmentation predictions and introduce the offset vector - based HRNet model .
- In chapter 5, we discuss the experimental setup i.e., datasets, evaluation metrics and implementation details and report our models results.
- In chapter 6 we summarize our findings and give an outlook for possible future research.



# Chapter 2

## Theoretical Background

---

1	Introduction to Deep Learning . . . . .	<b>32</b>
1.1	Convolutional Neural Networks (CNNs) . . . . .	32
1.2	Training the Neural Network . . . . .	40
1.3	Loss Function . . . . .	40
1.4	Performance Evaluation . . . . .	42
2	Introduction to Semantic Segmentation . . . . .	<b>44</b>

---

# 1 Introduction to Deep Learning

## 1.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are widely used in computer vision. They are made up of neurons with learnable weights and biases, similar to ordinary Neural Networks. Each neuron takes some inputs, performs a dot product, and optionally adds a non-linearity to the result. From raw image pixels on one end to class scores on the other, the whole network still represents a single differentiable score function. Lastly, they still contain a loss function on the last (fully-connected) layer (e.g. SVM/Softmax) [2].

CNNs outperform other neural network architectures on image processing tasks. This is related to their intrinsic use of spatial correlations in images and the advantages of weight sharing in terms of training efficiency. CNNs usually consist of numerous layers, each with its own set of capabilities. Low-level characteristics such as colors, edges, and forms are detected by the first layers. The layers begin to learn complicated characteristics, as the model progresses, and the final layer produces predictions. As a consequence, the network has a more comprehensive grasp of the images in the dataset. This section, which is primarily based on the work of [35], briefly discusses the functioning of several layers of CNNs that are common to image processing.

A convolutional network's architecture usually consists of convolution, pooling, activation and fully connected layers. At the same time, processes such as upsampling, concatenation, dropout, nearest neighbour or bilinear interpolation occur. An example of a CNN architecture is illustrated in Fig. 1.1

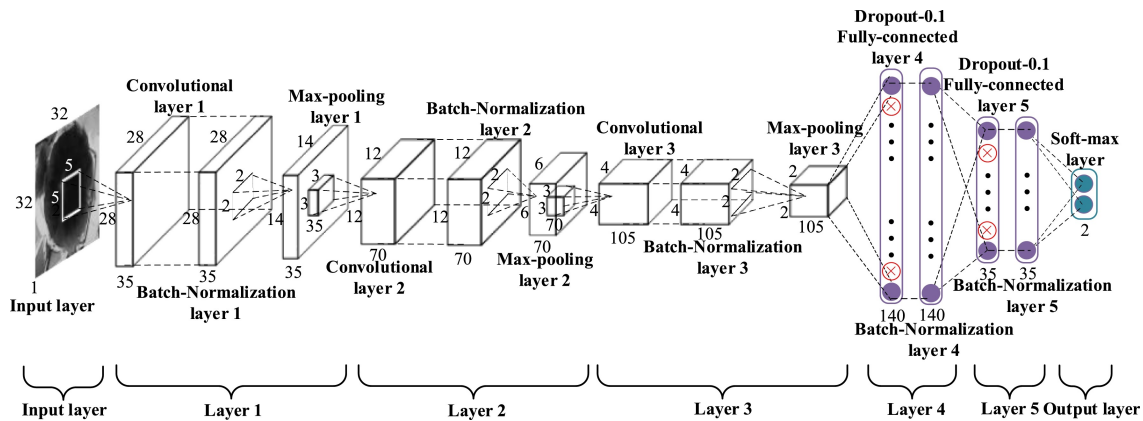


Figure 1.1: An example of a CNN architecture. From [36]

### Convolution

Images are represented by matrices containing pixel information. RGB color coding is the most widely used representation. As a result, an image has the dimensions  $(h \times w \times d)$ , where  $h$  denotes the height,  $w$  the width, and  $d$  the color channel depth. Convolutional layers are important layers in CNNs because they generate feature maps from input images or lower level feature maps. These layers consist of a kernel or a filter  $K$  with  $x$  rows,  $y$  columns, and a depth  $d$  that is smaller than the input image's height and breadth. Its content is based upon the operations to be performed. For example, in Fig. 1.2 an example of a kernel for applying Gaussian blur in order to smoothen the image before processing is shown, Sharpen image so as to enhance the depth of edges and edge detection. This kernel with the size  $(K_x \times K_y \times d)$  acts on the image's receptive field  $(K_x \times K_y)$ .

After that, the kernel moves across the picture, creating a feature map. Fig. 1.3 depicts an illustration of this procedure. In this example, the input image is  $3 \times 4$  and the convolution kernel size is  $2 \times 2$ . If we overlap the convolution kernel on top of the input picture, we can compute the product between the integers at the same place in the kernel and the input by adding these products together. For instance, if we overlap the kernel with the input's top left region, the convolution result at that spatial location is:  $1 \times 1 + 1 \times 4 + 1 \times 2 + 1 \times 5 = 12$ . Then we shift the kernel down one pixel and the upcoming convolution result is  $1 \times 4 + 1 \times 7 + 1 \times 5 + 1 \times 8 = 24$ . We keep moving the kernel down until it hits the input matrix's bottom border. The kernel is then returned to the top and moved to its right by one element (pixel).



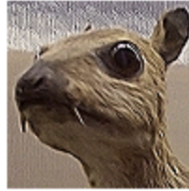

<i>Original</i>	<i>Gaussian Blur</i>	<i>Sharpen</i>	<i>Edge Detection</i>
$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$
			

Figure 1 .2: Kernel types

The convolution operation is defined identically for order 3 tensors. Assume the l-th layer’s input is an order 3 tensor of size  $H^l \times W^l \times D^l$ . A convolution kernel is also a size  $H \times W \times D^l$  tensor. When we overlay the kernel on top of the input tensor at  $(0, 0, 0)$ , we calculate the products of matching elements in all the  $D^l$  channels and sum the  $HWD^l$  products to produce the convolution result at this spatial location. To finish the convolution, we shift the kernel from top to bottom and from left to right.

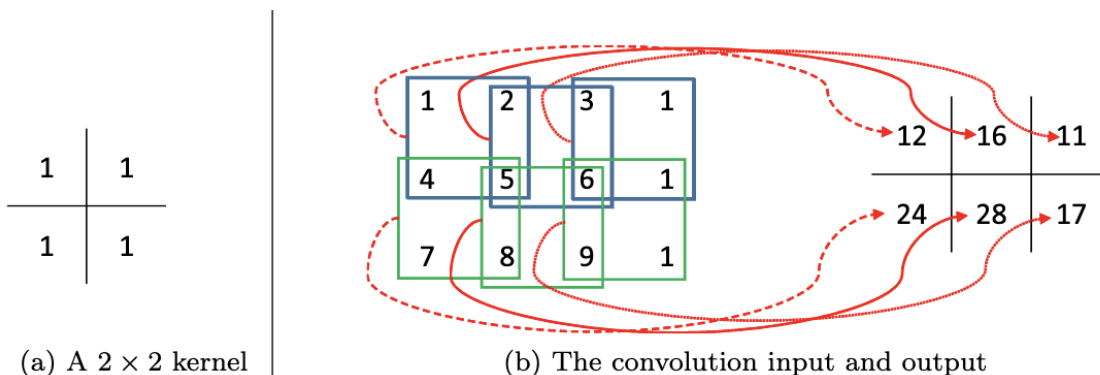


Figure 1 .3: Illustration of the convolution operation

In general, if the input is  $H^l \times W^l \times D^l$  and the kernel size is  $H \times W \times D^l \times D$ , the convolution result is  $(H^l - H + 1) \times (W^l - W + 1) \times D$  in size, where D denotes the number of kernels used. However, sometimes we need the input and output images to have the same height and width, and a simple padding trick can be used. For every channel of the input, if we pad ( i.e., insert)  $\lfloor \frac{H-1}{2} \rfloor$  rows above the first row and  $\lfloor \frac{H}{2} \rfloor$  rows below the last row, and pad  $\lfloor \frac{W-1}{2} \rfloor$  columns to the left of the first column and  $\lfloor \frac{W}{2} \rfloor$  columns to the right of the last column of the input, the convolution output will be  $H^l \times W^l \times D$  in size, i.e., having the same spatial extend as the input.

Another key notion in convolution is stride. Fig. 1 .3 depicts the kernel being convolved with the input at every possible spatial point, which corresponds to the stride  $s = 1$ . If  $s > 1$ , however, every kernel movement skips  $s - 1$  pixels. This means that the convolution is performed once every s pixels both horizontally and vertically [37].

To sum up, if the input is  $H^l \times W^l \times D^l$ , the kernel size is  $H \times W \times D^l \times D$ , and let’s denote the stride with S and the amount of zero padding with P then the convolution result is  $(\frac{H^l - H + 2P}{S} + 1) \times (\frac{W^l - W + 2P}{S} + 1) \times D$  in size, where D denotes the number of kernels used.

## Pooling

Pooling layers are used to reduce the dimensions of the feature maps. Thus, it reduces the number of parameters to learn and the amount of computation performed in the network. Another essential reason to use pooling layers is to avoid overfitting the training data. The most commonly used pooling methods are max pooling and average pooling as shown in Figure 1.4. The greatest input value or the average of all the values within the kernel is used to create the downsampled output, which results in a smaller output. The processed data is simplified by combining the convolutional and pooling layers, without losing crucial characteristics that are required to describe the picture [38].

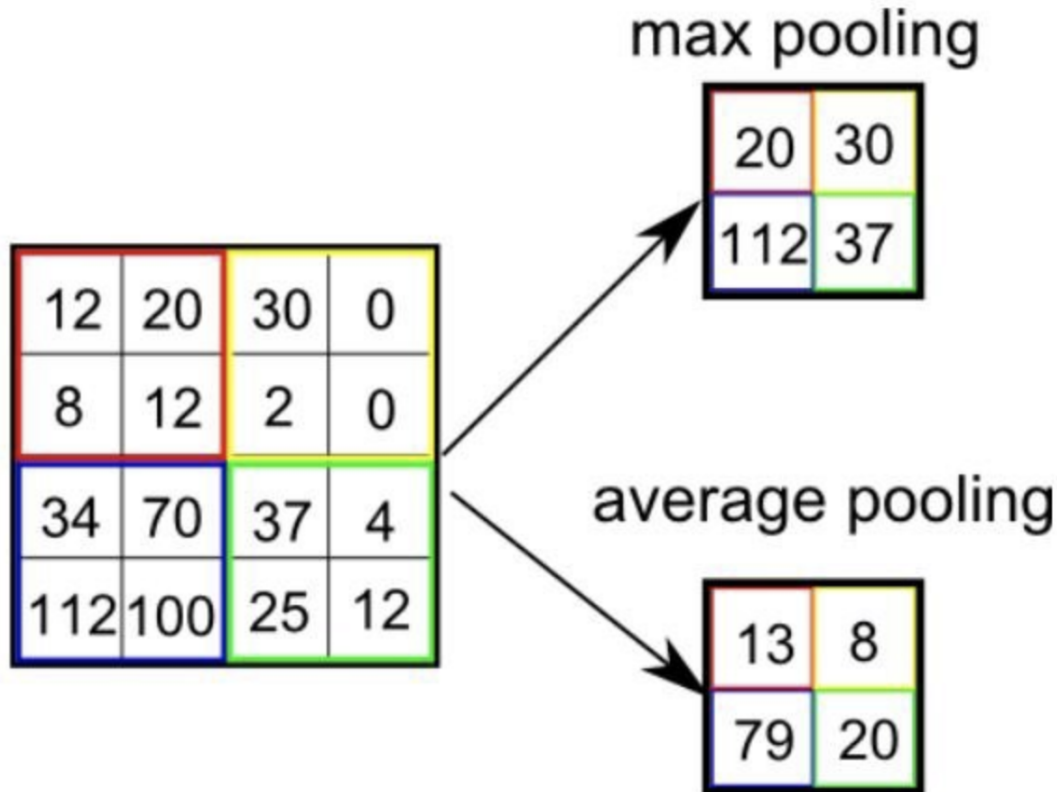


Figure 1.4: Types of pooling. From [38]

## Downsampling - Upsampling

The process of downsampling 2D images is used to lower the resolution of an input image. This is very useful for compressing image files while retaining as much information as possible. Upsampling is the inverse of downsampling, and it involves producing an output image with a greater resolution than the input. In reality, the goal is to produce a high-confidence image that is free of undesirable artifacts and retains a high level of detail. The transpose convolution and unpooling processes, which perform the opposite procedure of convolution and pooling, are two extensively used approaches [35]. The input image is smaller than the output image in transpose convolution. This is due to input dilation and padding to expand the matrix size, resulting in a convolution output that is larger than the original input. Transpose convolutional layers transfer one single activation to a field of multiple activations. An example of a 3x3 transpose convolution is depicted in Fig. 1.5.

Even though pooling cannot be reversed, the location of the pooling layer's maximum values is recorded in switch variables [40], which the unpooling process uses to place its matrix values at the appropriate positions, as shown in Figure 1.6. Since it is an expanded version of the input map, the output of such an unpooling layer is sparse.

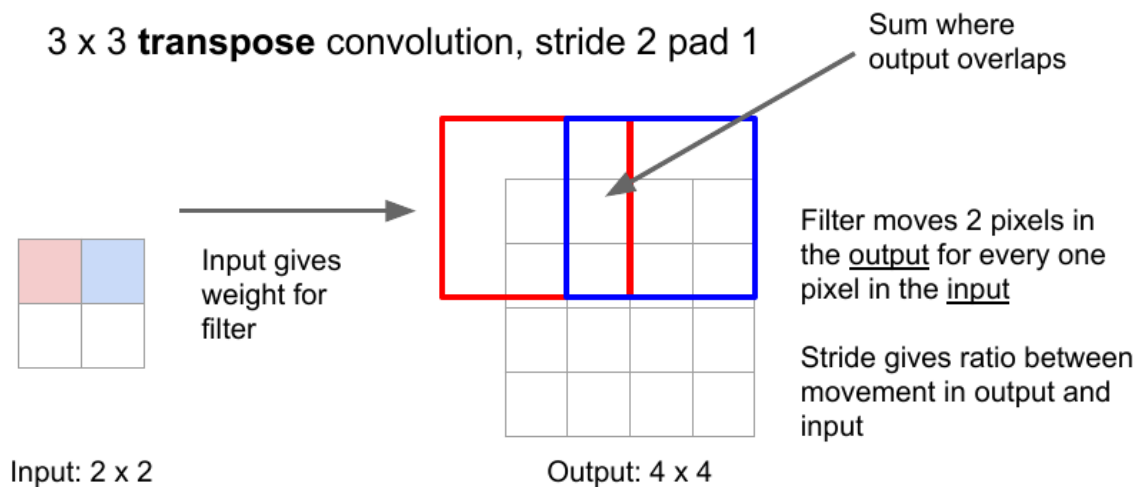


Figure 1 .5: Example of a 3x3 transpose convolution. From [39]

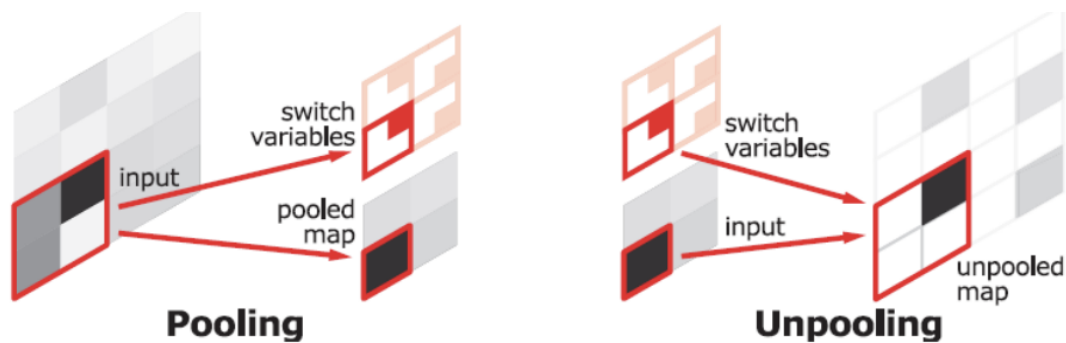


Figure 1 .6: Example of a 2x2 unpooling operation. From [40]

## Batch Normalization

Batch Normalization (BN) is a method that speeds up and stabilizes the training of DNN. It involves normalizing activation vectors from hidden layers using the current batch's first and second statistical moments (mean and variance). The nonlinear function is applied before (or after) this normalizing step. At each hidden layer, Batch Normalization transforms the signal as follow :

$$\mu = \frac{1}{n} \sum_i Z^{(i)} \quad (1.1)$$

$$\sigma^2 = \frac{1}{n} \sum_i (Z^{(i)} - \mu)^2 \quad (1.2)$$

$$Z_{norm}^{(i)} = \frac{Z^i - \mu}{\sqrt{\sigma^2 - \epsilon}} \quad (1.3)$$

$$\hat{Z} = \gamma * Z_{norm}^{(i)} + \beta \quad (1.4)$$

Using 1.1 and 1.2, the BN layer calculates the mean  $\mu$  and variance  $\sigma^2$  of the activation values across the batch. The activation vector  $Z^i$  is then normalized using 1.3. As a result, the output of each neuron follows a standard normal distribution across the batch.

Finally, it computes the output  $\hat{Z}$  of the layer using a linear transformation with two trainable parameters;  $\gamma$  and  $\beta$  (1.4). By modifying those two parameters, the model may find the best distribution for each of the hidden layers.

In general:

- $\gamma$  allows to adjust the standard deviation
- $\beta$  allows to adjust the bias, shifting the curve on the right or on the left side
- $\epsilon$  is a constant used for numerical stability

The network calculates the mean and standard deviation for the current batch after each iteration. Then it trains  $\gamma$  and  $\beta$  through gradient descent, using an Exponential Moving Average (EMA) to give more importance to the latest iterations [41]. An illustration of batch normalization is depicted below in Fig. 1.7

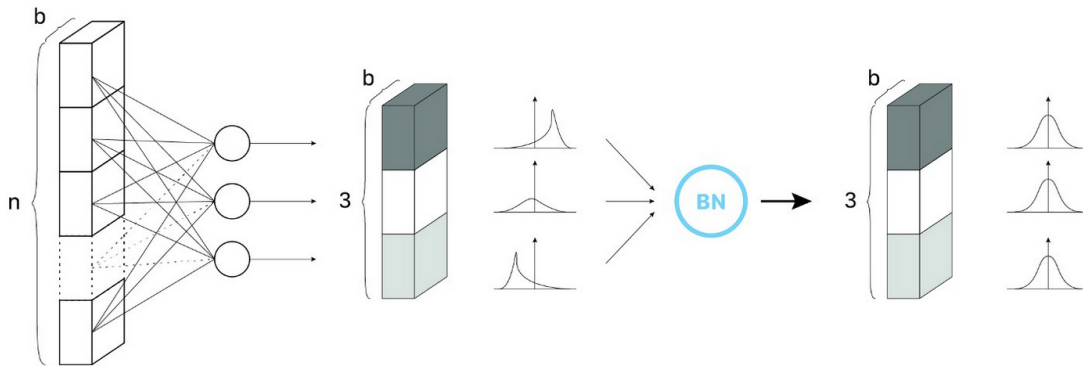


Figure 1.7: Batch Normalization. From [41]

## Dropout

Dropout is by far the most used regularization approach for deep neural networks. Even the most advanced models, which have a 95% accuracy rate, gain 2% accuracy by adding dropout, which is a significant improvement at that level. It is a basic technique for preventing overfitting. During training, a neuron is momentarily "dropped" or inhibited with probability  $p$  at each iteration. This signifies that, at this iteration, all of this neuron's inputs and outputs will be disabled. At each training step, the dropped-out neurons are resampled with probability  $p$ , so a dropped-out neuron at one step might become active at the next. The dropout-rate hyperparameter  $p$  is commonly a number around 0.5, which corresponds to 50 percent of the neurons being dropped out. In Fig. 1.8 a visualization of dropout is depicted.

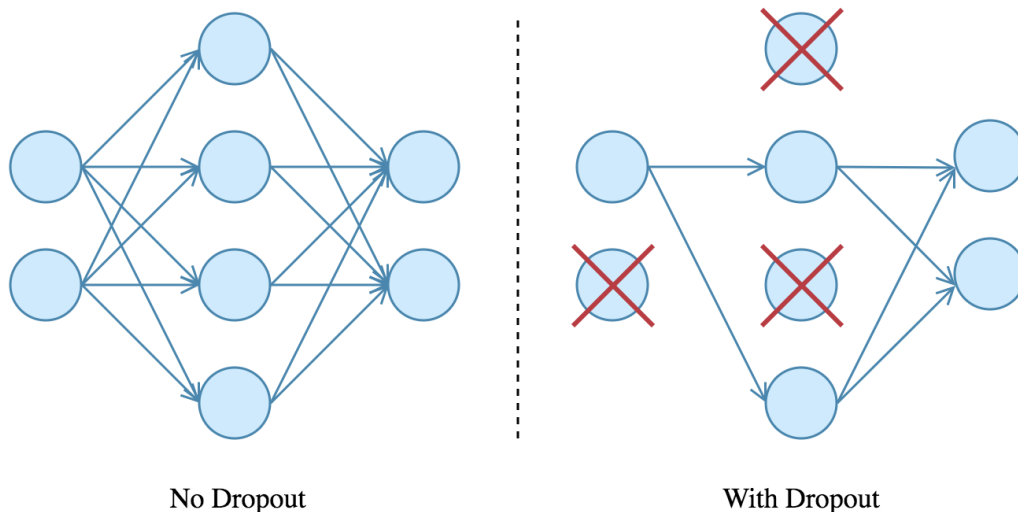


Figure 1.8: Dropout Visualization

Dropout is only applicable to input and hidden layer nodes, not output nodes. The edges in and out of the nodes that have been dropped out are deactivated. During each training phase, the nodes that were dropped out vary. We also don't use dropout during testing after the network has been trained, but we only use it during training [42].

## Nearest Neighbour Interpolation (NNI)

A relatively simple interpolation method is the Nearest Neighbour Interpolation . In general, interpolation aims to estimate the value of  $z$  at a new point  $x$  using a specified sample of data  $(z_1, z_2, \dots, z_n)$  at locations  $(x_1, x_2, \dots, x_n)$ . The NNI algorithm seeks to find  $i$  such that  $|x_i - x|$  is minimized, then the estimate of  $z$  is  $z_i$ .

A Thiessen polygon [43], sometimes called a Voronoi polygon, is formed by the collection of points that are closest to the value of  $x_i$ . Delauny triangulation [44] is the process used to calculate these polygons. A area made up of all points that are closer to  $x_i$  than any other  $x$ , corresponds to this specific polygon.

## Bilinear Interpolation

Bilinear interpolation is an extension of the one dimensional Linear Interpolation in two dimensions. It is performed using linear interpolation first in one direction, and then again in the other direction. In particular, the procedure computes values assigned to one new pixel as a lineal combination of the four closest pixels in the original image. Although each step's sampled values and position are linear, the interpolation as a whole is quadratic in the sample location [45].

Regarding the mathematical formulation of this procedure, suppose that we want to compute the value of the unknown function  $f$  at the point  $(x, y)$ , as the Fig.1.9 depicts. Let's assume that we know value of  $f$

at the four points  $Q_{11} = (x_1, y_1)$ ,  $Q_{12} = (x_1, y_2)$ ,  $Q_{21} = (x_2, y_1)$ , and  $Q_{22} = (x_2, y_2)$ . Initially, we do linear interpolation in the x-direction. This yields

$$f(x, y_1) = \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \quad (1.5)$$

$$f(x, y_2) = \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \quad (1.6)$$

The next step consists of interpolation in the y-direction to obtain the desired result:

$$\begin{aligned} f(x, y) &= \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} f(x, y_2) \\ &= \frac{y_2 - y}{y_2 - y_1} \left( \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \right) + \frac{y - y_1}{y_2 - y_1} \left( \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \right) \\ &= \frac{1}{(x_2 - x_1)(y_2 - y_1)} \begin{bmatrix} x_2 - x & x - x_1 \end{bmatrix} \begin{bmatrix} f(Q_{11}) & f(Q_{12}) \\ f(Q_{21}) & f(Q_{22}) \end{bmatrix} \begin{bmatrix} y_2 - y \\ y - y_1 \end{bmatrix} \end{aligned} \quad (1.7)$$

The same result will be achieved if the interpolation is done first along the y direction and then along the x direction [46].

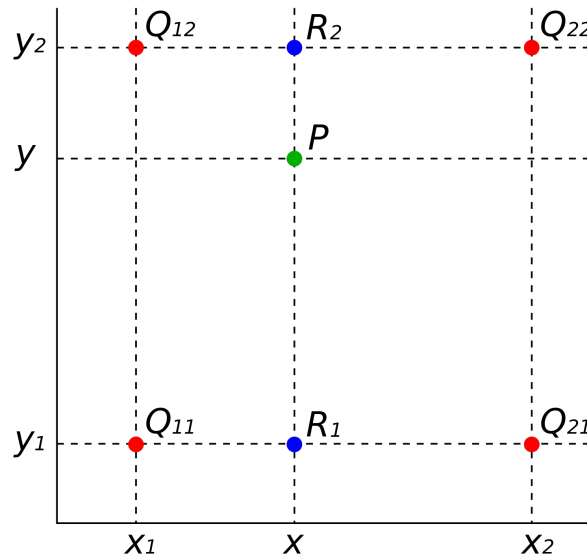


Figure 1.9: The four red dots depict the data points and the green dot is the point at which we want to interpolate. From [46].

The solution can also be written as a weighted mean of the  $f(Q)$ :

$$f(x, y) \approx w_{11}f(Q_{11}) + w_{12}f(Q_{12}) + w_{21}f(Q_{21}) + w_{22}f(Q_{22}) \quad (1.8)$$

where the weights sum to 1 and satisfy the transposed linear system

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_1 & x_2 & x_2 \\ y_1 & y_2 & y_1 & y_2 \\ x_1y_1 & x_1y_2 & x_2y_1 & x_2y_2 \end{bmatrix} \begin{bmatrix} w_{11} \\ w_{12} \\ w_{21} \\ w_{22} \end{bmatrix} = \begin{bmatrix} 1 \\ x \\ y \\ xy \end{bmatrix} \quad (1.9)$$



yielding the result

$$\begin{bmatrix} w_{11} \\ w_{12} \\ w_{21} \\ w_{22} \end{bmatrix} = \frac{1}{(x_2 - x_1)(y_2 - y_1)} \begin{bmatrix} x_2 y_2 & -y_2 & -x_2 & 1 \\ -x_2 y_1 & y_1 & x_2 & -1 \\ -x_1 y_2 & y_2 & x_1 & -1 \\ x_1 y_1 & -y_1 & -x_1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \\ xy \end{bmatrix} \quad (1.10)$$

which simplifies to

$$\begin{cases} w_{11} = \frac{(x_2 - x)(y_2 - y)}{(x_2 - x_1)(y_2 - y_1)} \\ w_{12} = \frac{(x_2 - x)(y - y_1)}{(x_2 - x_1)(y_2 - y_1)} \\ w_{21} = \frac{(x - x_1)(y_2 - y)}{(x_2 - x_1)(y_2 - y_1)} \\ w_{22} = \frac{(x - x_1)(y - y_1)}{(x_2 - x_1)(y_2 - y_1)} \end{cases} \quad (1.11)$$

in agreement outcome of repeated linear interpolation.

## Activation Functions

Activation functions are non-linear functions applied to the output of a transformation layer (convolution, batch normalization, etc.). Since this is an element-by-element action, every pixel from the preceding layer will be exposed to it. The network then moves on to another layer, such as a new convolutional layer or a pooling layer. There are a variety of activation functions for neural networks, however, only the most basic ones are discussed:

### 1. Rectified Linear Unit (ReLU)

A typical activation function for machine learning applications is [ReLU](#). Although [ReLU](#) is non-linear, it stays close to being linear and retains many of the useful properties for optimization and generalization of linear models [35]. The only difference between a [ReLU](#) and a linear unit is that a [ReLU](#)'s output is 0 for the first half of its area and discontinuous at the point  $x = 0$ . [ReLU](#) is applied element-wise to the input and is defined as:

$$ReLU(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \quad (1.12)$$

### 2. Scaled Exponential Linear Unit (SELU)

One of the novel activation functions is the Scaled Exponential Linear Unit. [SELU](#) is self-normalized, which means that the output always has a mean of 0 and a standard deviation of 1. When compared to external normalizing methods like batch normalization, this leads to fast convergence. The main idea is that each layer keeps the mean and variance from the preceding layer, allowing very robust learning and training multilayer networks.

Moreover, the gradients may be used to modify the variance. In order to enhance variance, the activation function requires an area with a gradient greater than 1. The parameter  $\lambda$  in Equation 1.13, which is the reason for the S(caled) in [SELU](#), accomplishes this. When  $\lambda$  is greater than 1, the gradient is greater than 1, and the activation function might cause the variance to grow. As a result, problems like vanishing and exploding gradient problems are difficult to solve with a [SELU](#) since the gradient is never close to 0. [SELU](#) is defined as:

$$SELU(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ a \exp^x - a & \text{if } x \leq 0 \end{cases} \quad (1.13)$$

This function has two predetermined values called  $\alpha \approx 1.6732$  and  $\lambda \approx 1.0507$ .

### 3. Softmax

The final outputs are normalized in the range [0,1] using the Softmax activation function, which represents the probability distribution of a random variable with  $n$  possible values [35]. This function is used to generate class probabilities in the final layer of a neural network and is defined as:

$$\text{Softmax}(x) = \frac{\exp(x)}{\sum_{n=0}^N \exp(x_n)} \quad (1.14)$$

## 1.2 Training the Neural Network

Following the definition of a network's architecture, the model's individual weights are learned through a training process. This is done in a supervised way for the majority of deep learning applications, such as medical analysis or autonomous driving, where multiple samples of input data are supplied with corresponding labels that are of interest. A 2D chest X-ray, for example, might be used as input data, and the label could be a binary diseased/non-diseased categorization. For each image-label pair, the image is sent into the network, which then generates a prediction based on the calculations produced by each layer. Then, using a loss function  $L$  that measures how similar they are, this prediction is compared to the label supplied. With this error measurement, an optimizer uses a method called backpropagation to modify the weights throughout the network, reducing the error the next time the network sees this case. The weights should ideally be adjusted when the network has been trained for a batch, which is a group of multiple samples. Stochastic Gradient Descent [35] is the most basic type of optimizer. Except for that, there are numerous others, such as Adam, RmsProp, and so on, making it yet another design decision before training [35]. This training procedure is repeated until the loss on a batch of validation data is minimized. In practice, this might take minutes for tiny basic datasets and days or weeks for bigger data, depending on the computational resources available to the user. Overfitting can occur when a CNN is trained on a small dataset for a long time, however this problem can be addressed by regularly testing against unknown validation data.

### Data Augmentation

Due to the lack of many examples to train on, overfitting occurs, resulting in a model with poor generalization performance. Obtaining new training data is difficult in most machine learning applications, particularly in image classification tasks. Data augmentation is a technique for generating extra training data from the data we already have. It "enriches" or "enlarges" the training data by producing new instances from existing ones through random manipulation. We may artificially increase the size of the training set in this way, preventing overfitting. As a result, data augmentation may be thought of as a regularization strategy.

Data augmentation is done dynamically during training time. Realistic pictures are required, and the modifications must be learnable. Rotation, shifting, resizing, exposure modification, contrast alteration, and other manipulations are common. This way we may produce a large number of new samples from a single training example. Furthermore, data augmentation is performed exclusively on the training data [42].

## 1.3 Loss Function

The loss function calculates the difference between the algorithm's actual output and the predicted output. It's a method to evaluate how the algorithm models the data. It may be divided into two categories. The first is for classification (discrete values, 0,1,2,...), while the second is for regression (continuous values).

### Classification

- **Cross-entropy Loss**

The cross-entropy as the Log Loss function (not the same but they measure the same thing) computes the difference between two probability distribution functions [47]. The cross-entropy of the distribution  $q$  relative to a distribution  $p$  over a given set is defined as follows:

$$H(p, q) = -E_p[\log q] \quad (1.15)$$

where  $E_p[\cdot]$  is the expected value operator with respect to the distribution  $p$ . For discrete probability distributions  $p$  and  $q$  with the same support  $X$ , equation 1 .15 implies that:

$$H(p, q) = - \sum_{x \in X} p(x) \log q(x) \quad (1 .16)$$

There are different types of Cross Entropy [48]. Some of them are the following:

- Binary Cross-Entropy:

$$L_{BCE}(y, \hat{y}) = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})) \quad (1 .17)$$

where  $y$  is the real value and  $\hat{y}$  is the predicted value by the prediction model.

- Weighted Binary Cross-Entropy:

$$L_{W-BCE}(y, \hat{y}) = -(\beta * y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})) \quad (1 .18)$$

where  $\beta$  value can be used to tune false negatives and false[49]. It is widely used in case of skewed data [50].

- Balanced Cross-Entropy:

$$L_{BCE}(y, \hat{y}) = -(\beta * y \log(\hat{y}) + (1 - \beta)(1 - y) \log(1 - \hat{y})) \quad (1 .19)$$

In terms of image processing,  $\beta$  is usually defined as  $1 - \frac{y}{H * W}$ . In this apart from just positive example [51] , we also weight also the negative examples.

- Ohem Cross-Entropy (OHEM):

Some object identification datasets include a disproportionately large number of simple instances and few challenging ones. Training may be more effective and efficient if these challenging instances are automatically chosen. The bootstrapping method known as OHEM, or Online Hard Example Mining, adapts SGD to sample from examples in a non-uniform manner based on the current loss of each example under consideration [52].

- **Log-loss**

The Log-loss is the Binary cross-entropy up to a factor  $\frac{1}{\log(2)}$ . For negative values, this loss function grows linearly. Logistic regression is a popular algorithm that uses the Log-loss.

- **Exponential Loss**

The exponential loss is convex and rises exponentially for negative values. As a result, it is more sensitive to outliers. The AdaBoost algorithm uses this loss. In the context of additive modeling, the main appeal of exponential loss is its computing efficiency. AdaBoost's additive expansion estimates onehalf of the log-odds of  $P(Y = 1|x)$ . This justifies using its sign as the classification rule [53].

The population minimizer is:

$$P(Y = 1|x) = \frac{1}{1 + \exp -2f(x)} \quad (1 .20)$$

- **Hinge Loss**

The Hinge loss function was created to fix the SVM algorithm's hyperplane in classification tasks. The purpose is to make different penalties at the point where the hyperplane is not precisely predicted or is too closed[54]. It's mathematical formula is the following:

$$Hinge = \max(0, 1 - y * f(x)) \quad (1 .21)$$

- **Kullback Leibler Divergence Loss**

The Kullback–Leibler divergence,  $D_{KL}(p||q)$ , is a measure of how one probability distribution  $q$  is different from a second, reference probability distribution  $p$ . For discrete probability distributions  $p$  and  $q$  defined on the same probability space,  $X$ , the KL divergence from  $q$  to  $p$  is defined [55] to be:

$$D_{KL}(p||q) = \sum_{x \in X} p(x) \log\left(\frac{p(x)}{q(x)}\right) \quad (1.22)$$

## Regression

- **Mean Square Error (MSE) — L2**

The Mean Squared Error is perhaps the simplest and most common loss function. It takes the difference between our model’s predictions and the ground truth, squares it, and averages it out across the whole dataset[56]. It’s mathematically defined as follows:

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \quad (1.23)$$

- **Mean Absolute Error (MAE) - L1**

The Mean Absolute Error differs just slightly from the **MSE** in terms of definition. It takes the difference between our model’s predictions and the ground truth, multiplies it by the absolute value, and averages it across the whole dataset[56]. It’s mathematically defined as follows:

$$MSE = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i| \quad (1.24)$$

## 1.4 Performance Evaluation

When analyzing the performance of semantic segmentation, there are two main criteria to consider:

1. accuracy: the success of a method
2. computation complexity: the speed and memory requirements

### Accuracy

Measuring segmentation performance might be difficult, due to the fact that there are two different parameters to track. The first is classification, which simply entails determining pixel-wise class labels. The second is localization, which entails calculating the correct set of pixels that enclose the object. Some of the most common principal measures used in evaluating semantic-segmentation performance are the following:

- **ROC-AUC**

Receiver-Operator Characteristic curve (**ROC**) summarizes the trade-off between true positive rate and false-positive rate for different probability thresholds in a predictive model. On the other hand, Area Under the Curve (**AUC**) is appropriate when observations are balanced between classes and is beneficial in evaluating binary classification issues. However, due to the fact that most semantic segmentation sets [57, 58] are unbalanced between the classes, this metric is no longer used .

- **Pixel Accuracy (PA)**

Pixel accuracy determines the ratio of correctly categorized pixels to the total number of pixels. It is mathematically formulated as :

$$PA = \frac{\sum_{j=1}^k n_{jj}}{\sum_{j=1}^k t_j} \quad (1.25)$$

where  $n_{jj}$  is the total number of True Positives for class  $j$  and  $t_j$  is the total number of pixels labelled as class  $j$ .

Mean pixel accuracy (**mPA**), is a version of **PA** which computes the ratio of correct pixels on a per-class basis. Its mathematical formula is the following:

$$mPA = \frac{1}{k} \sum_{j=1}^k \frac{n_{jj}}{t_j} \quad (1.26)$$

- **IoU**

**IoU** is the ratio of the intersection of the pixel-wise classification results with the ground truth to their union in semantics segmentation. It is a metric for comparing sample sets' similarity and diversity. It is formulated as:

$$IoU = \frac{\sum_{j=1}^k n_{jj}}{\sum_{j=1}^k (n_{ji} + n_{ij} + n_{jj})}, i \neq j \quad (1.27)$$

where  $n_{jj}$  is the total number of True Positives,  $n_{ij}$  is the total number of False Positives and  $n_{ji}$  is the total number of False Negatives for class  $j$ .

Two widely used extended versions of IoU are:

- **mIoU** : is the class-averaged IoU:

$$mIoU = \frac{1}{k} \sum_{j=1}^k \frac{n_{jj}}{n_{ji} + n_{ij} + n_{jj}}, i \neq j \quad (1.28)$$

- Frequency-weighted IoU (**FwIoU**): weighs each class importance depending on appearance frequency by using  $t_j$  as in Eq. 1.25. It is formulated as:

$$FwIoU = \frac{1}{\sum_{j=1}^k t_j} \sum_{j=1}^k t_j \frac{n_{jj}}{n_{ji} + n_{ij} + n_{jj}}, i \neq j \quad (1.29)$$

In general, **IoU** computes the ratio of true positives to the sum of false positives, false negatives and true positives. In comparison with **PA**, **IoU** is more informative because it takes into account the false positives, whereas **PA** does not. Unfortunately, **IoU** only counts the number of pixels that have been correctly labeled, not the accuracy of the segmentation boundaries. Lastly, **IoU** and its variants are the most commonly used accuracy evaluation metrics in the most popular semantic segmentation tasks [58].

- Precision-Recall Curve (**PRC**)-based metrics

Precision is the ratio of true positives over a summation of true positives and false positives. Recall is the ratio of true positives over a summation of true positives and false negatives. Precision and Recall are the two axes of the **PRC** used to depict the trade-off between precision and recall, under a varying threshold for the task of binary classification. Their mathematical formula for a given class  $j$  is:

$$Precision = \frac{n_{jj}}{n_{ij} + n_{jj}}, j \neq i \quad (1.30)$$

$$Recall = \frac{n_{jj}}{n_{ji} + n_{jj}}, j \neq i \quad (1.31)$$

Some of the main **PRC**-based metrics are the following:

1. F-score: the harmonic mean of the precision and recall for a given threshold. It is formulated as:

$$F_{score} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (1.32)$$

2. **PRC-AUC**: the area under the **PRC**.

### Computational Complexity

Two main criteria are used to evaluate the computation cost: how fast the algorithm completes and how much computational memory is required.

- **Execution time**: This is the total processing time, beginning with the introduction of a single picture to the system/algorithm and ending with the pixel-wise semantic segmentation results. This metric's performance is highly dependent on the hardware used. As a result, any execution time metric for an algorithm should be accompanied with a detailed description of the hardware employed.
- **Memory Usage**: Memory use is especially critical when using semantic segmentation in low-performance devices like cellphones and digital cameras, or when the system's requirements are extremely restrictive. Military systems or security-critical systems, such as self-driving automobiles, are prime examples. During the execution of a complex algorithm such as semantic segmentation, the amount of memory used may change dramatically. As a result, peak memory consumption, which is essentially the amount of memory required for the complete segmentation procedure for a single picture, is a commonly used metric for this purpose.

## 2 Introduction to Semantic Segmentation

There are various methods to understand or interpret the meaning of an image, and they can be divided into three categories:

- **Image-level classification**: has no localisation and usually just has one main object to categorize in an image.
- **Object detection**: locates foreground objects of interest inside a picture. This entails finding a bounding box for the entire item and categorizing it at the same time. Object detection is outside the scope of this thesis since it does not address regressing bounding boxes.
- **Semantic segmentation**: has more fine-grained localization, with the purpose of classifying individual pixels and defining the shape of all object classes. Segmentation can also be complicated by the need to categorize all pixels in an image, including foreground and background. The advantage of employing pixel-wise segments over bounding boxes is that two segments never overlap, but bounding boxes tend to overlap to represent the same items in the case of several densely packed or irregularly shaped objects.

Segmentation or detection is frequently the initial stage, in an intelligent system that employs vision to make critical decisions. For example, in the construction of an autonomous driving automobile, the driving system would first collect photos of the environment and must then be able to understand the scene quickly and segment or recognize pedestrians and other objects. The effectiveness of the segmentation of these images determines the autonomous system's subsequent decisions, such as driving, turning, or conducting a safety maneuver. Other applications of segmentation in computer vision include robotics [3], medical image analysis [4] and surveillance systems.

In this thesis, an RGB color image ( $H \times W \times 3$ ) is used as input, and the result is a segmentation map with each pixel containing a class label expressed as an integer ( $H \times W \times 1$ ) as shown in Fig. 2.1.

Each of the potential classes has its own output channel, which contains one-hot representations of the class labels. A prediction may be compressed into a segmentation map by obtaining the argmax of each depth-wise pixel vector, as depicted in Fig. 2.2.

When the prediction is overlaid onto the observation, a mask lighting the parts of an image belonging to a given class is generated [5] as illustrated in Fig 2.3.

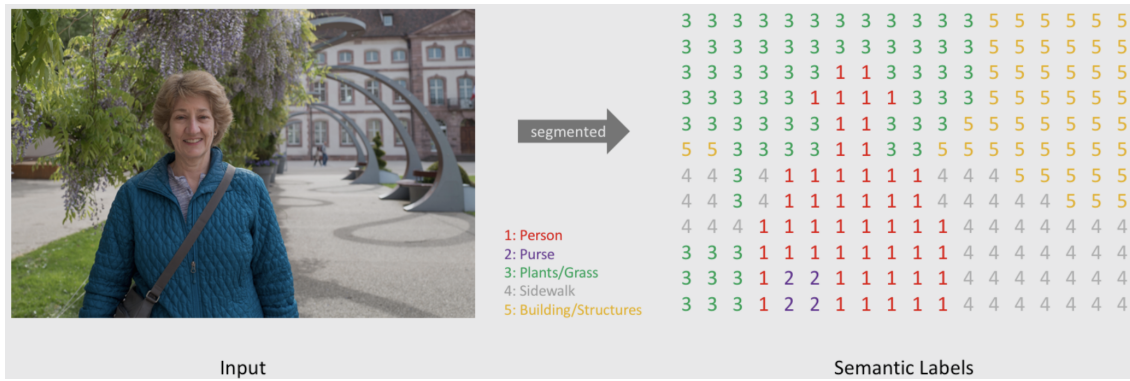


Figure 2 .1: Pixel-level Semantic Segmentation Segmentation. From [5]

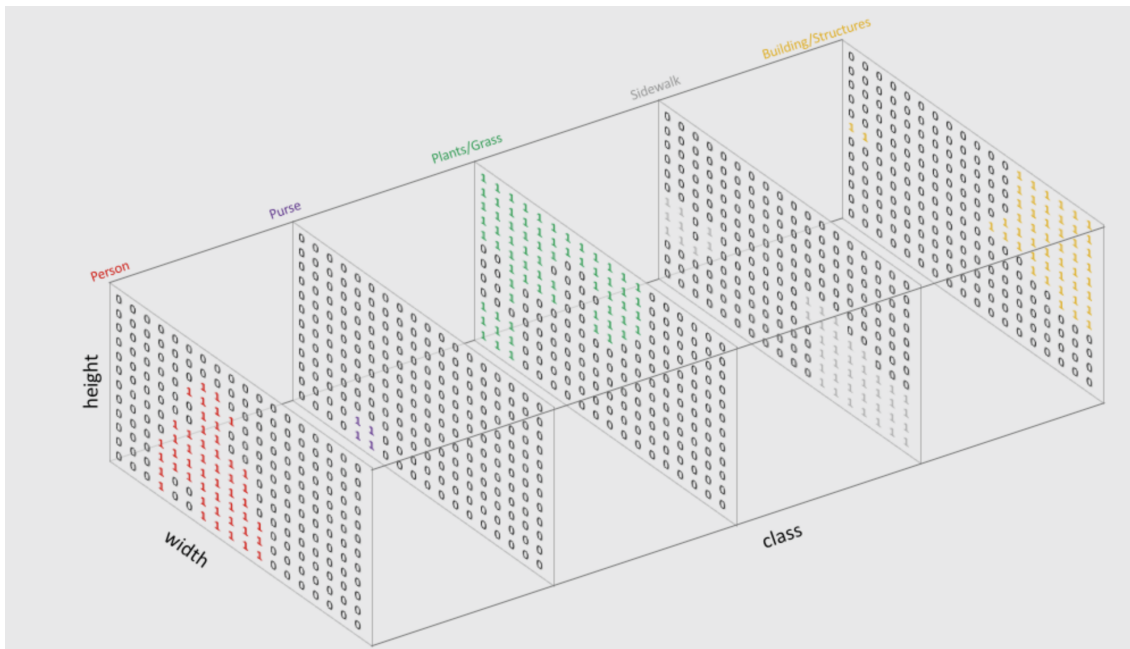


Figure 2 .2: One-hot representations of the class labels. From [5]



Figure 2 .3: Final Result. From [5]





# Chapter 3

## Related Work

---

1	Evolution of Semantic Segmentation . . . . .	<b>48</b>
1.1	Semantic Segmentation Before Deep Neural Networks . . . . .	48
1.2	Semantic Segmentation Using Deep Neural Networks . . . . .	48
2	Some popular state-of-the-art semantic segmentation models . . . . .	<b>50</b>
2.1	Based on Fully Convolutional Network . . . . .	50
2.2	Based on Dilation/Atrous convolution . . . . .	50
2.3	Based on Top-down/Bottom-up approach . . . . .	52
2.4	Based on receptive field enlargement and multi-scale context incorporation . . . . .	57
2.5	Based on Transformers . . . . .	60
2.6	Comparison . . . . .	64

---

# 1 Evolution of Semantic Segmentation

One of the most fundamental issues in computer vision is semantic segmentation. Since the advent of digital images, researchers have been studying this subject. The algorithms have been improving over the years, progressing from simple image thresholding, that classified pixels into two groups, to deep neural networks, that performed multi-class segmentation with excellent results.

## 1.1 Semantic Segmentation Before Deep Neural Networks

Semantic segmentation is the technique of segmenting image with understanding of image in pixel level. In other terms, it is the study and categorization of each pixel into many classes. Before DNNs, Watershed method, Image thresholding, K-means clustering, Conditional Random Fields, and more algorithms have been developed to address this difficult issue.

### Image Thresholding

Image thresholding is the most basic and perhaps the oldest technique for image segmentation. This method is a technique of splitting an image into two (or more) groups of pixels, namely foreground and background. In order to obtain a thresholded image, the original one is typically transformed to grayscale (as shown in Fig. 1.1a) and then the thresholding technique is applied (as shown in Fig. 1.1b). This approach is also known as Binarization, as the image is transformed to a binary format. In more detail, a pixel's intensity value is transformed to 1 (white) if it is less than the threshold value. These pixels are known as object points. On the other hand, if a pixel's value is larger than the threshold value, the pixel is set to 0 (black). These pixels are called background points. The threshold value can be either set by the designer of the method or automatically.

In 1979, Nobuyuki Otsu [59] devised an algorithm known as Otsu's technique, which has since become the most often used method for determining the threshold automatically. The threshold is set by reducing intra-class intensity variance, or, to put it another way, maximizing intra-class variation. The technique exhaustively searches for a threshold that minimizes intra-class variance, which is defined as the weighted sum of the two classes' variances:

$$\sigma_{\omega}^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t) \quad (1.1)$$

where  $\omega_0$  and  $\omega_1$  are the probabilities of two classes separated by the threshold  $t$ , and  $\sigma_0^2$  and  $\sigma_1^2$  are variances of these two classes [60]. The desired threshold  $T$  corresponds to the minimum intra-class variance:

$$T = \min_t(\sigma_{\omega}^2(t)) \quad (1.2)$$

### Conditional Random Field (CRF)

Random field approaches are a popular way of modeling spatial regularities in images. Their application range from low-level noise reduction to high-level object or category detection and semi-automatic object segmentation. The focus of early research was on generative modeling with Markov Random Fields. CRF models have grown in popularity as a result of their ability to predict segmentation (labeling) immediately from an observed image. Visual scene interpretation, which tries to divide images into their component semantic-level areas and give appropriate class labels to each zone, uses CRFs as a useful tool for a range of data segmentation and labeling tasks. It is vital to capture both the image's global context and local information for proper classification [60, 62].

## 1.2 Semantic Segmentation Using Deep Neural Networks

Deep learning approaches have been a major breakthrough in the field of computer vision, despite numerous standard image processing techniques. Neural networks set state-of-the-art in all image processing tasks including semantic segmentation. One of the very early deep convolutional neural networks used for semantic

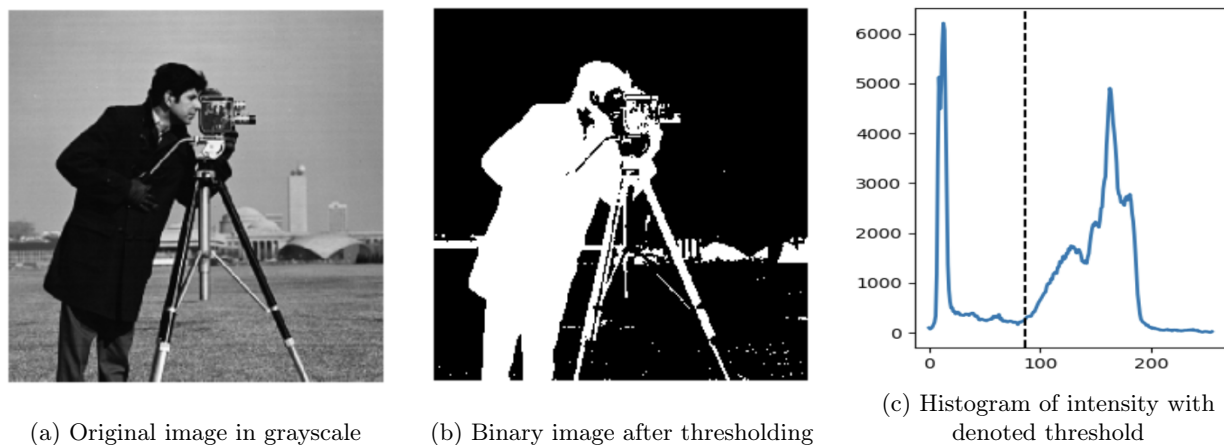


Figure 1 .1: Example of thresholding using Otsu's method. From [61]

segmentation is FCN. A variety of more advanced FCN-based approaches have been proposed to address this issue, including SegNet [63], DeconvNet [40], U-Net [64], and DeepLab [65, 66]. Afterwards, encoder-decoder architecture became one of the most successful deep learning semantic segmentation architectures [67]. A pre-trained classification network, such as VGG [68] or ResNet [69], is commonly used as the encoder. The decoder is responsible for semantically projecting the encoder's discriminative features onto the pixel space in order to get a dense classification of each pixel in the input image.

The previous state-of-the-art approaches used a high-resolution recovery procedure to raise the resolution of the representation from a low-resolution representation produced by a classification or classification-like network as shown in Figure 1 .2. J. Wang in [6] proposed HRNet, a novel architecture which is able to maintain high-resolution representations through the whole process. HRNet starts from a high-resolution convolution stream, gradually add high-to-low resolution convolution streams one by one, and connect the multi-resolution streams in parallel, as shown in Fig. 2 .12.

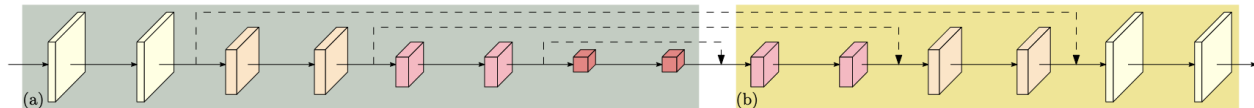


Figure 1 .2: The structure of recovering high resolution from low resolution. (a) A low-resolution representation learning subnetwork (such as VGGNet [68], ResNet [69]), which is formed by connecting high-to-low convolutions in series. (b) A high-resolution representation recovering subnetwork, which is formed by connecting low-to-high convolutions in series. Representative examples include SegNet [63], DeconvNet [40], U-Net [64], encoder-decoder [67]. From [6]

## 2 Some popular state-of-the-art semantic segmentation models

In this section, we'll go through the architectural details of some state-of-the-art CNN-based semantic segmentation models. The models are divided into categories based on the most important attribute. We briefly examined the advantages and disadvantages of each model category at the end of each categorization discussion.

### 2.1 Based on Fully Convolutional Network

#### Fully Convolutional Network (FCN)

Fully Convolutional Network for Semantic Segmentation [70] was the first model that re-architects and fine-tunes classification networks to direct dense prediction of semantic segmentation. Three deep convolutional neural networks were taken into consideration. AlexNet [71], VGGNet [68] and GoogleNet[72](all three pre-trained on ILSVRC [73] data) were used as base models. Long et al. transferred these models from classifiers to dense FCN by removing fully connected layers with  $1 \times 1$  convolutional layers and append a  $1 \times 1$  convolution with channel dimension 21 to predict scores for each of the 20 PASCAL VOC [74] classes and background class.

For each of the networks, fine-tuning from classification to segmentation resulted in realistic predictions. The authors have experienced among FCN-AlexNet, FCN-VGG16 and FCN-GoogLeNet, FCN-VGG16. The segmentation equipped VGG16 (FCN-VGG16) appeared to be state-of-the-art at 52.6 mean IoU performance on the PASCAL VOC 2011 test set. Training on extra data even raised the performance to 59.4 mean IoU.

While fully convolutional classifiers may be fine-tuned to segmentation with even high scores on standard metrics, their output is too coarse. The 32 pixel stride at final prediction layer restricts the scale of the detail in the output. In order to address this issue, the authors used bilinear interpolation to upsample the coarse output  $32 \times$  to make it pixel dense. However, fine-grained segmentation required more than just an upsampling. As a result, they have used skip connection to combine the last prediction layer with VGG16's feature-rich lower layers. Different combinations including FCN-16s and FCN-8s and FCN-32s as depicted in Figure 2.1 were used. Among them, FCN-8s improves the final performance of the net to 62.7% mean IoU on the VOC2011 test set and 62.2% on the VOC2012 test set [74] setting the state-of-the-art. Lastly, FCN-16s gave the best result on both NYUDv2 [75] & SIFT Flow [76] datasets.

The base model VGG16, bipolar interpolation technique for up-sampling the final feature map, skip connection for combining low layer and high layer features in the final layer for fine-grained semantic segmentation are all major changes in FCN that helped the model achieve state-of-the-art results.

FCN has used only local information for semantic segmentation. However, only local information makes semantic segmentation quite ambiguous as it loses global semantic context of the image. In order to reduce ambiguity, contextual information from the whole image is very helpful [77].

### 2.2 Based on Dilation/Atrous convolution

#### DilatedNet

Traditional CNN, which is used for classification tasks, loses resolution in its way and it is hence unsuitable for dense prediction. Yu and Koltun proposed dilated convolution or DilatedNet [78], a modified version of traditional CNN that systematically accumulates multi-scale contextual information for enhanced segmentation without sacrificing resolution. Unlike traditional pyramidal CNN, DilatedNet is a rectangular prism of convolutional layers. Without any loss of any spatial information, it can enable the exponential expansion of receptive fields as shown in figure 2.2

#### Deeplab

Deep Convolutional Neural Networks [79] have been proven to be useful for semantic segmentation when deployed in a fully convolutional manner. Chen et al. in [80] has brought together methods from DCNN and probabilistic graphical model. The repetitive use of max-pooling and striding at consecutive layers of these networks, diminishes the spatial resolution of the resultant feature maps by a factor of 32 in each direction

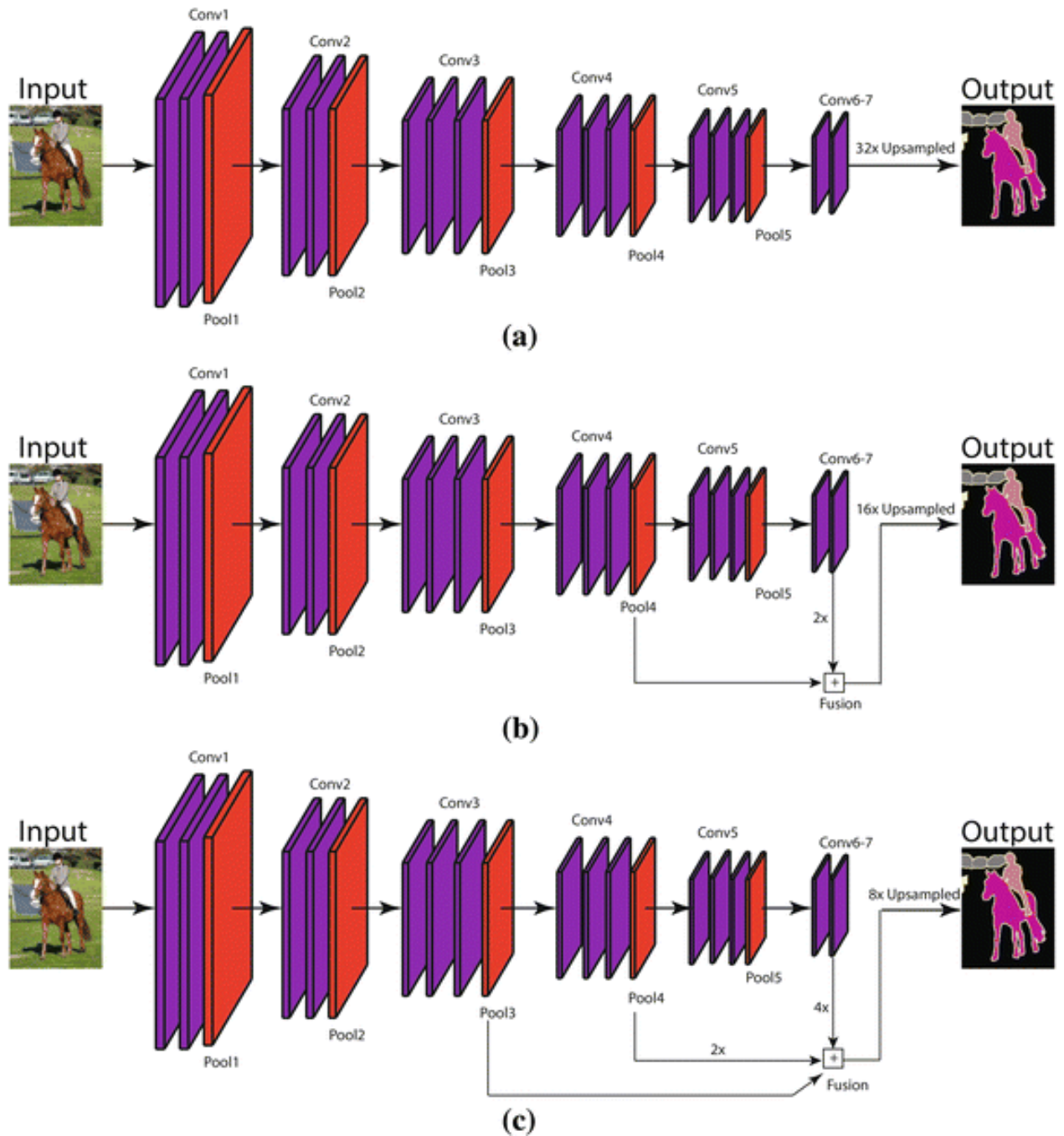


Figure 2 .1: Illustration of different FCN architectures. (a) Illustration of FCN-32 architecture. (b) Illustration of FCN-16 architecture. (c) Illustration of FCN-8 architecture

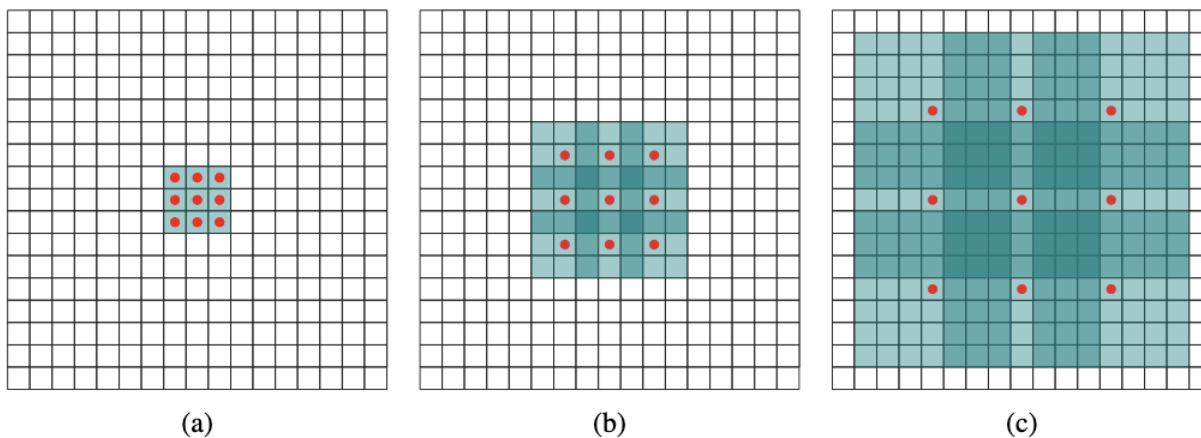


Figure 2.2: (a)1-DilatedNet with receptive field  $3 \times 3$ , (b) 2-DilatedNet with receptive field  $7 \times 7$  and (c)4-DilatedNet with receptive field  $15 \times 15$ . From [78]

in modern DCNNs [71]. In order to deal with this problem, they employed 'atrous' algorithm for efficient dense CNN computation. Atrous convolution is a strong tool for explicitly adjusting the filter's field-of-view as well as controlling the resolution of feature responses generated by DCNNs. To handle the problem of segmenting objects at multiple scales, designed modules, that use atrous convolution in cascade or in parallel to capture multi-scale context, used multiple atrous rates [80]. An example of atrous convolution with kernel size  $3 \times 3$  and different rates is depicted in Fig 2.3

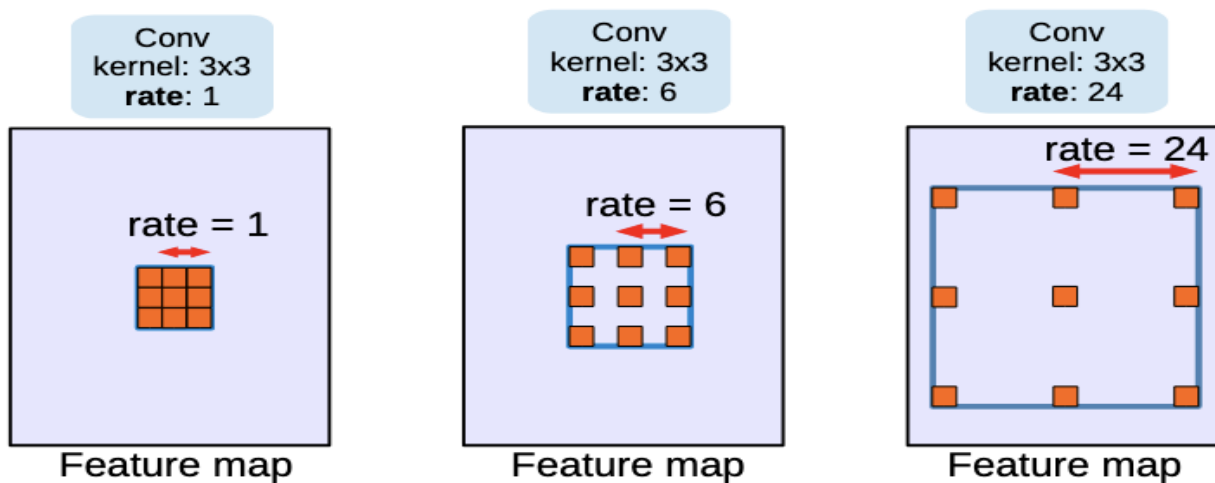


Figure 2.3: Atrous convolution with kernel size  $3 \times 3$  and different rates. Standard convolution corresponds to atrous convolution with rate = 1. Employing large value of atrous rate enlarges the model's field-of-view, enabling object encoding at multiple scales. From [80]

The main advantage of a dilation-based model is that it preserves the image's spatial resolution, allowing for dense prediction. However, dilation convolution separates pixels from their global context, making misclassification more possible to occur.

## 2.3 Based on Top-down/Bottom-up approach

### DeconvNet

H. Noh et al. [81] proposed a novel semantic segmentation algorithm by learning a deconvolution network. DeconvNet has a convolutional and deconvolutional network, as shown in Fig 2.4. The convolution network

is a feature extractor that converts the input image to multidimensional feature representation. Apart from the final classification layer, it is topologically identical to the first 13 convolution layers and two fully linked layers of VGG16 [68]. As far as the deconvolutional network is concerned, it is a shape generator that generates object segmentation from the convolution network's feature. It contains multiple series of unpooling, deconvolution, and rectification layers, and is a mirrored counterpart of the convolution network. Deconvolution networks, in contrast to convolution networks, which lower the size of activations by feed-forwarding, augment them through a mix of unpooling and deconvolution operations. Following a similar idea proposed in [82, 83], unpooling is done using max-pooling indices which are stored in the convolutional network during the convolution process. To densify enlarged but sparse un-pooled feature maps, convolution like operation is done using multiple learned filters by linking single input activation with multiple outputs. Unlike FCN, the authors used their network to predict pixel-by-pixel item suggestions taken from the input image. The results of all suggestions were then pooled and sent back to the original image space for segmentation of the whole image. This technique handles multi-scale objects in fine detail while simultaneously reducing training complexity and memory usage. The network's final output is a probability map with the same size as the input image, showing the likelihood of each pixel belonging to one of the predefined classes.

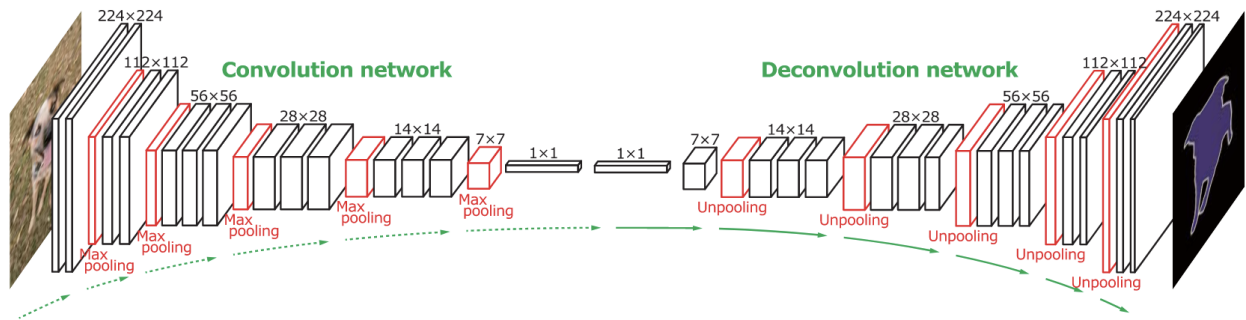


Figure 2 .4: Overall DeconvNet architecture. From [81]

One of the main drawbacks of DeconvNet is that it has large parameterization, needs more computational resources and is harder to train end-to-end, primarily due to the use of fully connected layers

## U-Net

U-Net [84] is a U-shaped semantic segmentation network which contains a contracting and expanded path as shown in Fig 2 .5. Two consecutive  $3 \times 3$  convolutions are followed by ReLU nonlinearity and max-pooling utilizing a  $2 \times 2$  window with stride 2 at each step of the contracting path. Feature information is enhanced while spatial information is diminished during contraction. On the contrary, every step of the expanding path consists of up-sampling of feature map followed by a  $2 \times 2$  up-convolution. This decreases the size of the feature map by a factor of two. The cropped feature map from the contracting path is then concatenated with the reduced feature map. Then ReLU nonlinearity is added after two consecutive  $3 \times 3$  convolution operations.

## SegNet

SegNet [85] consists of an encoder network and a decoder network, followed by a pixelwise classification layer, as shown in Fig 2 .6. The encoder network has 13 convolutional layers, which correspond to the first 13 convolutional layers of the VGG16 object classification network [68]. At the deepest encoder output, authors reject the completely linked layers in favor of preserving higher resolution feature maps. This led to a decrease of the amount of parameters from 134M to 14.7M. In order to produce a set of feature maps, each encoder in the encoder network executes convolution with a filter bank. After that, batch normalization and an element-wise ReLU are applied. Batch normalization is used in order to reduce internal covariate shift. After max-pooling with a  $2 \times 2$  window and stride 2 (non-overlapping window) is conducted, the resulting output is then sub-sampled by a factor of two. Sub-sampling produces a large input image context (spatial window) for each pixel in the feature map. While more layers of max-pooling and sub-sampling can offer greater translation invariance for robust classification, the feature maps' spatial resolution suffers and there is lossy



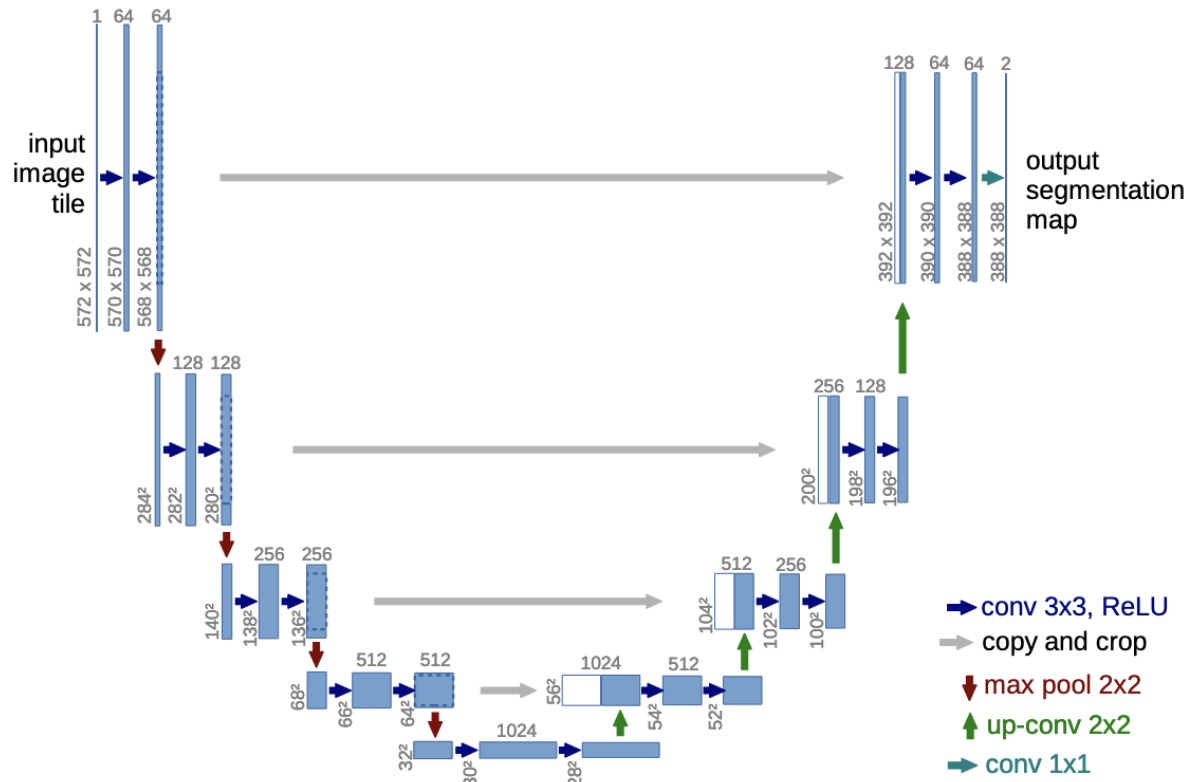


Figure 2.5: U-Net architecture. From [84]



image representation with blurred boundaries. As a result, before performing sub-sampling, it is important to capture and store boundary information in the encoder feature maps. Regarding the decoder network, since each encoder layer has a corresponding decoder layer, the decoder network consists of 13 layers. To retain the same output image resolution as the input image, SegNet up-samples the resultant high-resolution sparse feature map in its decoder, using the stored max-pooling indices from the corresponding encoder feature map. The output of the final decoder is sent into a multi-class soft-max classifier, which generates class probabilities for each pixel separately.

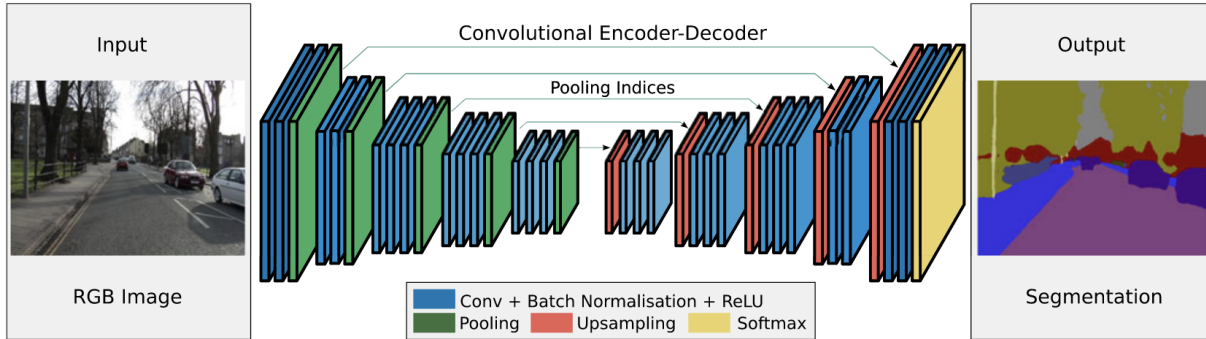


Figure 2 .6: SegNet architecture. From [85]

It is worth mentioning that [DeconvNet](#) [81] and [U-Net](#) [84] are two other models that have a similar architecture to SegNet, but with notable changes. [DeconvNet](#) has a big computational cost in comparison with SegNet. [U-Net](#) does not reuse pooling indices but instead transfers the entire feature map to the corresponding decoders and concatenates them to upsampled (via deconvolution) decoder feature maps. Unlike the Visual Geometry Group ([VGG](#)) net design, in [U-Net](#) there are no conv5 and max-pool five blocks. SegNet, on the other hand, makes use of all of the [VGG](#) net's pre-trained convolutional layer weights as pre-trained weights.

### Densely Connected Convolutional Network ([DenseNet](#))

[DenseNets](#) [86, 87] have demonstrated outstanding performance on image classification tasks. The main idea behind [DenseNets](#) is that if each layer is directly linked to every other layer in a feed-forward way, the network will be more accurate and easier to train. [DenseNets](#) are made up of dense blocks and pooling layers, with each dense block consisting of an iterative concatenation of previously created feature maps. Moreover, a *transition down* is introduced to reduce the spatial dimensionality of the feature maps. In this transformation, a  $1 \times 1$  convolution, which conserves the number of feature maps, is followed by a  $2 \times 2$  pooling operation. This architecture can be viewed as a development of Residual Neural Networks ([ResNets](#)) [88], which performs iterative summing of previously generated feature maps. However, this minor change has some interesting implications:

- parameter efficiency
- implicit deep supervision
- feature reuse

[DenseNets](#) are an excellent choice for semantic segmentation since they inherently induce skip connections and multi-scale supervision.

### [FC-DenseNet](#)

S. Jégou et al. [89] extended [DenseNets](#) to deal with the problem of semantic segmentation. They replaced the convolution technique with a dense block and upsampling operation, known as *transition up*. Transition up modules consist of a transposed convolution that upsamples the previous feature maps. The input of a new dense block is formed by concatenating the upsampled feature maps with the ones arriving from the skip connection. Skip connections are shown as yellow circles in Fig 2 .7 . The linear expansion in the number of

features would be excessively memory intensive, especially for the full resolution features in the pre-softmax layer, because the upsampling approach increases the feature maps spatial resolution.

In order to overcome this limitation, the input and output of a dense block are not concatenated. As a result, the transposed convolution is only applied to the feature maps produced by the final dense block, rather than all feature maps concatenated thus far. The last dense block at the same resolution summarizes the information included in all preceding dense blocks. Due to the pooling layers, some information from earlier dense blocks is lost in the transition down. However, this information is available in the network's downsampling path and may be passed via skip connections. As a result, all of the available feature maps at a given resolution are used to compute the dense blocks of the upsampling path. The architecture of FC-DenseNet is shown in Fig 2.7

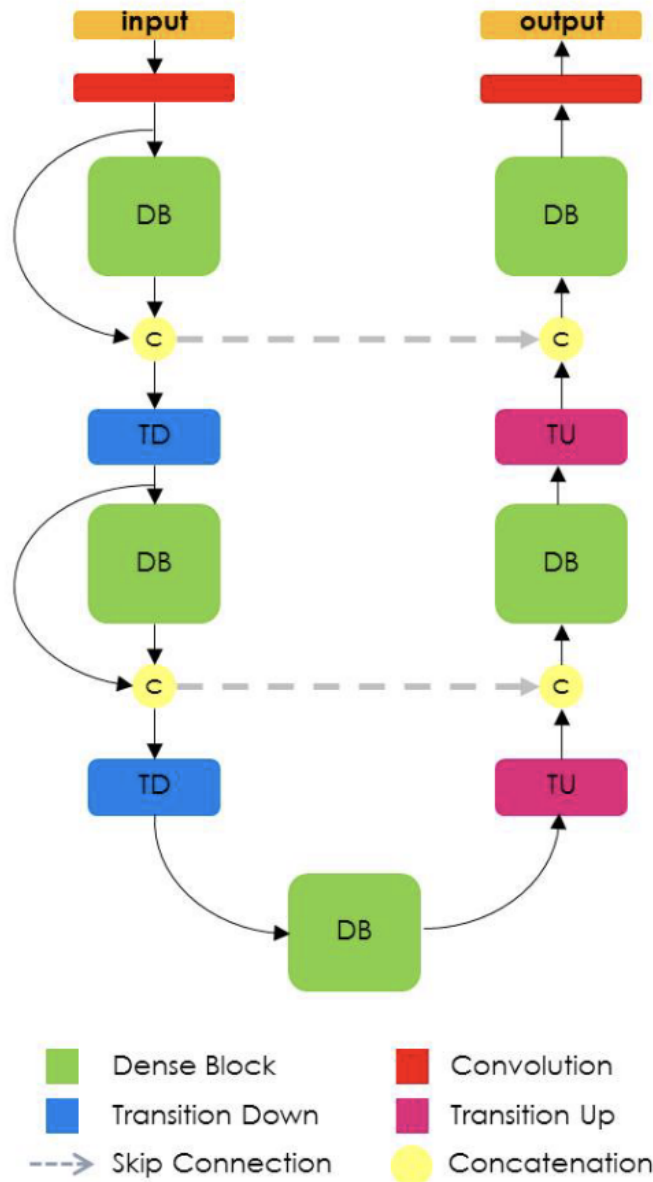


Figure 2.7: FC-DenseNet architecture. From [89]

Without using any additional post-processing modules or pretraining, FC-DenseNet achieved state-of-the-art performance on urban scene benchmark datasets like CamVid and Gatech. Furthermore, thanks to clever

model development, the proposed technique has many less parameters than the already reported best entry for these datasets.

## 2.4 Based on receptive field enlargement and multi-scale context incorporation

### DeepLabv2

DeepLabv2 [18] was suggested by the authors of DeepLab, who upgraded their network utilizing *ASPP* to capture multiscale objects and context . In this proposed technique we have parallel dilated convolutions with different rates applied in the input feature map, which are then fused together. *ASPP* helps to account for varying object sizes in the picture since objects of the same class might have varied sizes in the image. Object identification [90], [91], instance-level segmentation , visual question answering [92], and optical flow [93] are all examples of problems where the atrous convolution approach has been used. Both *ResNet* [69] and *VGG* Network [68] were used as base network in this architecture.

### DeepLabv3

DeepLabv3 [80] is a semantic segmentation architecture that has numerous improvements over DeepLabv2. Modules that apply atrous convolution in cascade or in parallel to capture multi-scale context by adopting several atrous rates are meant to solve the challenge of segmenting objects at multiple scales. In addition, DeepLabv2's *ASPP* module was enhanced with image-level features that encode global context and improve efficiency. The authors modify the *ASPP* module by applying global average pooling to the model's final feature map, feeding the image-level features to a  $1 \times 1$  convolution with 256 filters (with batch normalization), and then bilinearly upsampling the feature to the appropriate spatial dimension. The architecture of *ASPP* is depicted in Fig 2 .8 .

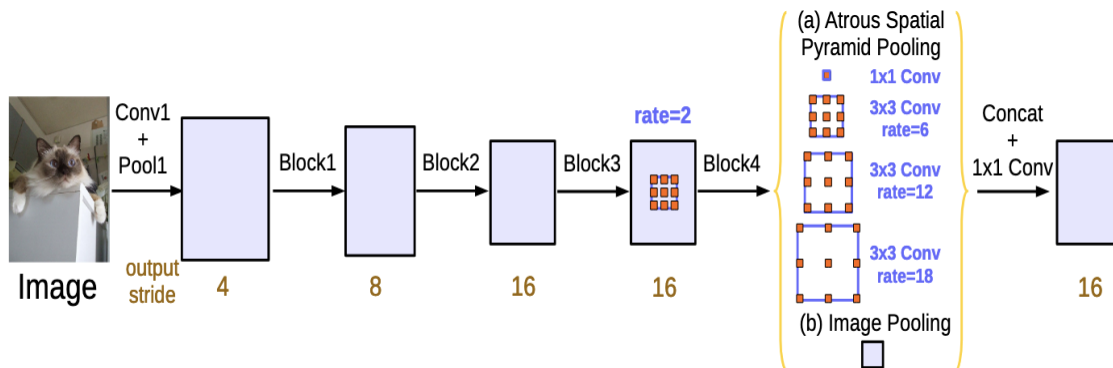


Figure 2 .8: Parallel modules with *ASPP*, augmented with image-level features. From [80]

### PSPNet

*PSPNet* [12] is a semantic segmentation model that employs a pyramid parsing module to exploit global context information through different-region based context aggregation. The combination of local and global cues improves the final prediction's accuracy. *PSPNet* extracts the feature map from an input image using a pretrained *CNN* with the dilated network technique. The input image size determines the final feature map size. The authors utilize the pyramid pooling module to aggregate context information on top of the map. The pooling kernels cover the entire, half of, and tiny sections of the image using the 4-level pyramid. They are fused as the global prior. Then, in the last part, there is a combination of the previous with the original feature map. After that, a convolution layer is used to create the final prediction map. The architecture of *PSPNet* is illustrated in Fig 2 .9.

### Gated-Shape CNN (GSCNN)

Takikawa et al. proposed *GSCNN* [94], a 2-stream *CNN* i.e. one stream is normal *CNN* (regular stream) while the other is a shape stream, which explicitly processes shape information in a separate stream. The

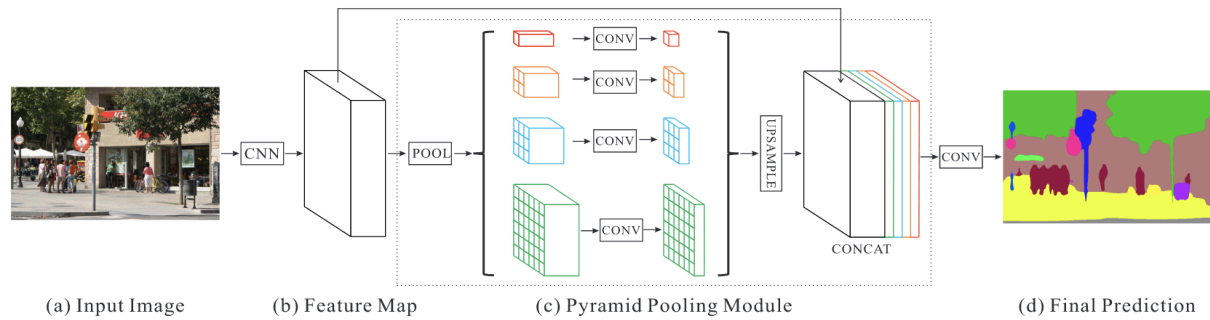


Figure 2.9: Overview of proposed PSPNet. From [12]

architecture of this model is shown in Fig 2.10. Regular stream maybe any feedforward fully convolutional network like ResNet [69] or VGG Network [68] based segmentation network. The Shape Stream takes image gradients as well as output of the first convolutional layer of the Regular Stream as input and outputs semantic boundaries. The network architecture is composed of a few residual blocks interleaved with Gated Convolutional Layers (GCLs). This GCL guarantees that the shape stream only processes boundary-relevant information. GT boundary edges (from GT segmentation masks) are used to supervise the shape stream using binary cross entropy loss on output boundaries. As far as the Fusion Module is concerned, it takes as input the dense feature representation from Regular Stream and fuses it with boundary map from Shape Stream in a way that multi-scale contextual information is preserved.

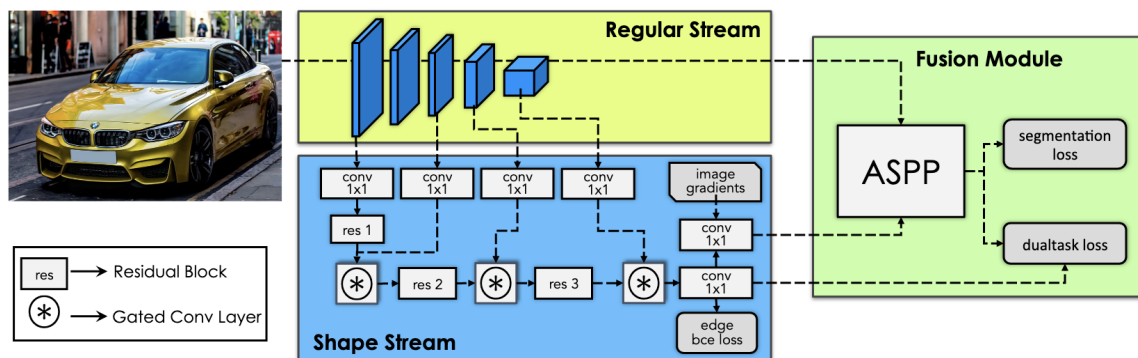


Figure 2.10: Overview of proposed PSPNet. From [12]

### SwiftNet

Oršić et. al. proposed SwiftNet [95], a brand-new method of semantic segmentation built on shared pyramidal representation and the fusion of several characteristics along the upsampling process. Due to the potent regularization effects induced by feature sharing throughout the resolution pyramid, the proposed pyramidal fusion strategy is particularly successful for dense inference in pictures with high scale variance. Interpretation of the decision process suggests that their approach succeeds by acting as a huge ensemble of relatively simple models, as well as due to large receptive range and strong gradient flow towards early layers. The architecture of this model is depicted in Fig. 2.11. This model achieves state-of-the-art results (82.82%) on ACDC dataset.

### HRNet

HRNet [6] is a general-purpose convolutional neural network that can be used to perform tasks such as semantic segmentation, object detection, and image classification. It can keep high-resolution representations throughout the whole process. Many of the more recent works on semantic segmentation use HRNet as the

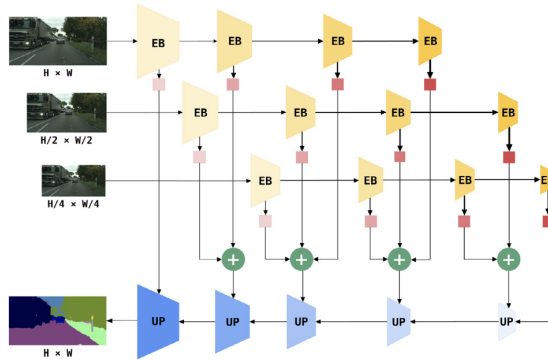


Figure 2.11: Overview of proposed SwiftNet. The proposed multi-scale architecture has shared encoders and pyramidal fusion. From [95]

backbone by exploiting contextual models, such as self-attention and its extensions. Starting with a high-resolution convolution stream, we add high-to-low resolution convolution streams one by one before connecting the multi-resolution streams in parallel. The resultant network has numerous (4 in [6]) phases, the last of which comprises streams that match to resolutions. The authors perform several multi-resolution fusions by repeatedly sharing data across parallel streams.

We will analyze the HRNet architecture in more detail. Initially, we feed the image through a stem, which consists of two stride 2  $3 \times 3$  convolutions that reduce the resolution to  $\frac{1}{4}$ , and then the main body, which returns the same resolution ( $\frac{1}{4}$ ) representation. The main body of HRNet is illustrated in Fig. 2.12.

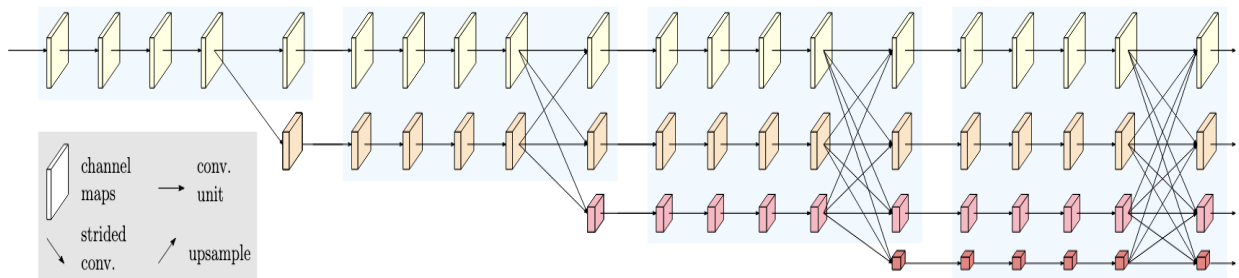


Figure 2.12: Illustrating the HRNet architecture. From [6]

It consists of the following components:

- **Parallel multi-resolution convolutions**

Regarding to parallel multi-resolution convolutions, we start with a high-resolution convolution stream, then add high-to-low resolution streams one by one, forming additional stages, and finally connecting the multi-resolution streams in parallel. As a result, the resolutions for parallel streams of a later stage are made up of the preceding stage's resolutions plus an extra lower one.

- **Repeated multi-resolution fusions**

The fusion module's goal is to share information amongst multi-resolution representations. It is repeated several times (e.g., every 4 residual units). We will analyze an example of fusing 3-resolution representations, as depicted in Fig. 2.13. The input consists of three representations:  $R_r^i$ ,  $r = 1, 2, 3$ , with  $r$  is the resolution index, and the associated output representations are  $R_r^o$ ,  $r = 1, 2, 3$ . The transform function  $f_{xr}(\cdot)$  is chosen in accordance with the input resolution index  $x$  and the output resolution index  $r$ . We have the following cases:

- If  $x = r$ ,  $f_{xr}(R) = R$

- If  $x < r$ ,  $f_{xr}(R)$  uses  $(r - s)$  stride-2  $3 \times 3$  convolutions to downsample the input representation  $R$
- If  $x > r$ ,  $f_{xr}(R)$  upsamples the input representation  $R$  using bilinear upsampling and a  $1 \times 1$  convolution to align the number of channels.

Each output representation is defined as the sum of the transformed representations of the three inputs:

$$R_r^o = f_{1r}R_1^i + f_{2r}R_2^i + f_{3r}R_3^i$$

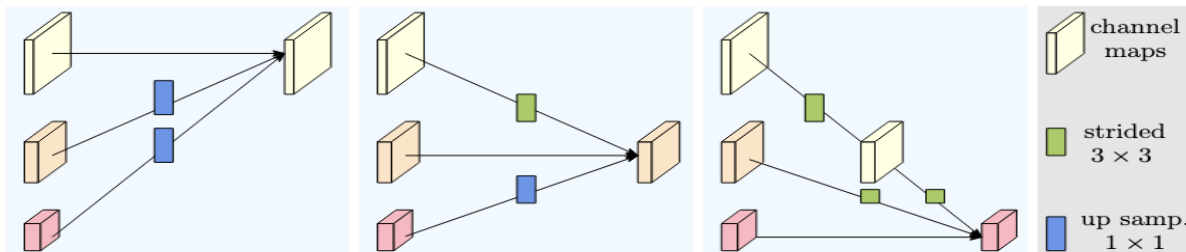


Figure 2.13: Illustrating how the fusion module aggregates the information for high, medium and low resolutions from left to right, respectively. From [6]

### • Representation head

As far as the representation head is concerned, there are three kinds illustrated in Fig. 2.14:

- **HRNetV1**. Only the high-resolution stream is represented in the output, whereas the other three low representations are ignored.
- **HRNetV2**. A bilinear upsampling to rescale the low-resolution representations to high resolution without affecting the number of channels is used. Then, there is a concatenation of the four representations before mixing them with a  $1 \times 1$  convolution.
- **HRNetV2p**. Multi-level representations are constructed by downsampling the high-resolution representation output from HRNetV2 to multiple levels.

For the task of semantic segmentation, HRNetV2 is used.

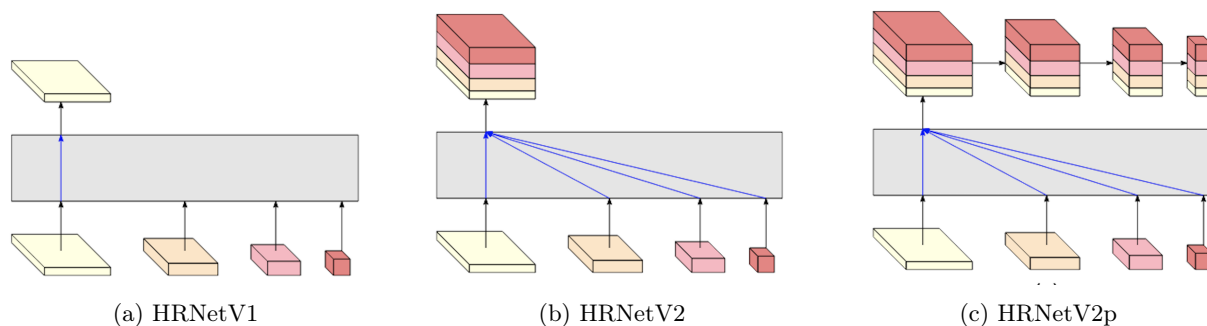


Figure 2.14: Representation heads

## 2.5 Based on Transformers

### Segmentation Transformer: Object-Contextual Representations (OCR) for Semantic Segmentation

Motivated by the fact that the label of a pixel is the category of the object to which the pixel belongs, Yuan et. al. in [9] provide an effective method for characterizing a pixel by utilizing the representation of the appropriate object class. This method is known as **OCR**. Under the guidance of the segmentation of the ground truth, they first learn object areas computed from a deep network such as HRNet [6]. Second, they

calculate the representation of the object region by combining the representations of the pixels that make up the object region. Then, after computing the relationship between each pixel and each object region, they add the object-contextual representation, which is a weighted aggregate of all the representations of the object regions, to each pixel's representation. The illustration of the OCR pipeline is shown in 2.15

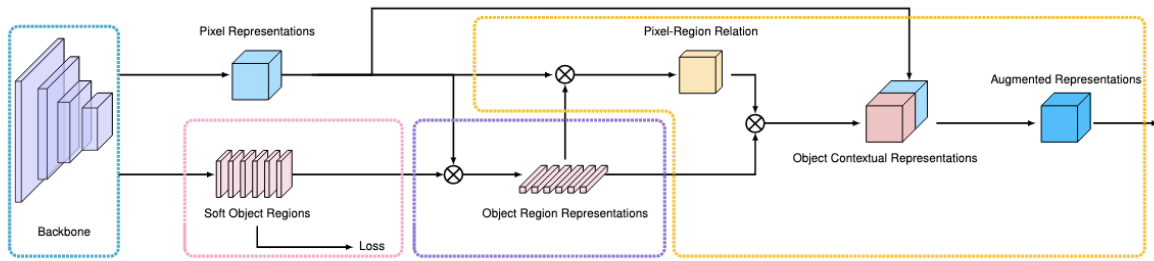


Figure 2.15: Illustrating the pipeline of OCR. From [9]

Rephrasing the OCR pipeline shown in Fig. 2.15 using the Transformer encoder-decoder architecture, we have the following architecture in Fig. 2.16. This architecture is known as Segmentation Transformer.

### Vision Transformer (ViT)

Dosovitskiy et. al. proposed ViT [96], a model for classifying images using patches of the picture, having a Transformer-like design. A sequence of vectors is created by dividing a picture into fixed-size patches, linearly embedding each one, adding position embeddings, and then feeding the assembled vectors to a conventional Transformer encoder. The traditional method of performing classification involves including an extra learnable "classification token" in the sequence. An illustration of ViT is shown below in Fig. 2.17

### Lawin Transformer

Yan et. al. introduced Lawin Transformer [97]. In particular, by using a window attention method, they successfully include multi-scale representations into the semantic segmentation ViT, substantially enhancing its effectiveness and performance. In order to do this, they provide big window attention, which enables the local window to query a wider region of the context window with a negligible processing overhead. Moreover, they enable the big window attention to catch the contextual information at various sizes, by controlling the ratio of the context area to the query area. Additionally, spatial pyramid pooling is used in conjunction with large window attention. Likewise, this resulted in a novel decoder for semantic segmentation called large window attention spatial pyramid pooling (LawinASPP). Fig. 2.18 depicts the difference between ASPP and LawinASPP. Atrous convolution in ASPP collects representations at several scales using various dilation rates. In contrast, LawinASPP substitutes the suggested big window attention for atrous convolution. The query area is represented by the red window. The context region is represented by the yellow, orange, and purple windows, which have various spatial sizes.

The final ViT, the Lawin Transformer, consists of a LawinASPP as the decoder and an effective hierarchical vision transformer (HVT) as the encoder. The image is pass into the encoder part, which is usually a MiT (encoder of SegFormer [98]). The LawinASPP-based decoder part is then fed with the features from the last three stages. Finally, the first-stage feature of the encoder enhances the output feature using low-level data. The term "MLP" stands for multi-layer perceptron, 'CAT' stands for combining the features. The word "Lawin" stands for wide-window focus, "R" stands for the context patch size to query patch size ratio. It achieves new state-of-the-art performance on Cityscapes (84.4 % mIoU), ADE20K [99] (56.2% mIoU) and COCO-Stuff datasets. Its architecture is shown in Fig. 2.19.

### Vision Transformer Adapter (ViT-Adapter)

ViT performs poorly on dense prediction problems because it lacks prior knowledge of the images, in contrast to recent visual transformers that include vision-specific inductive biases into their models. To address this



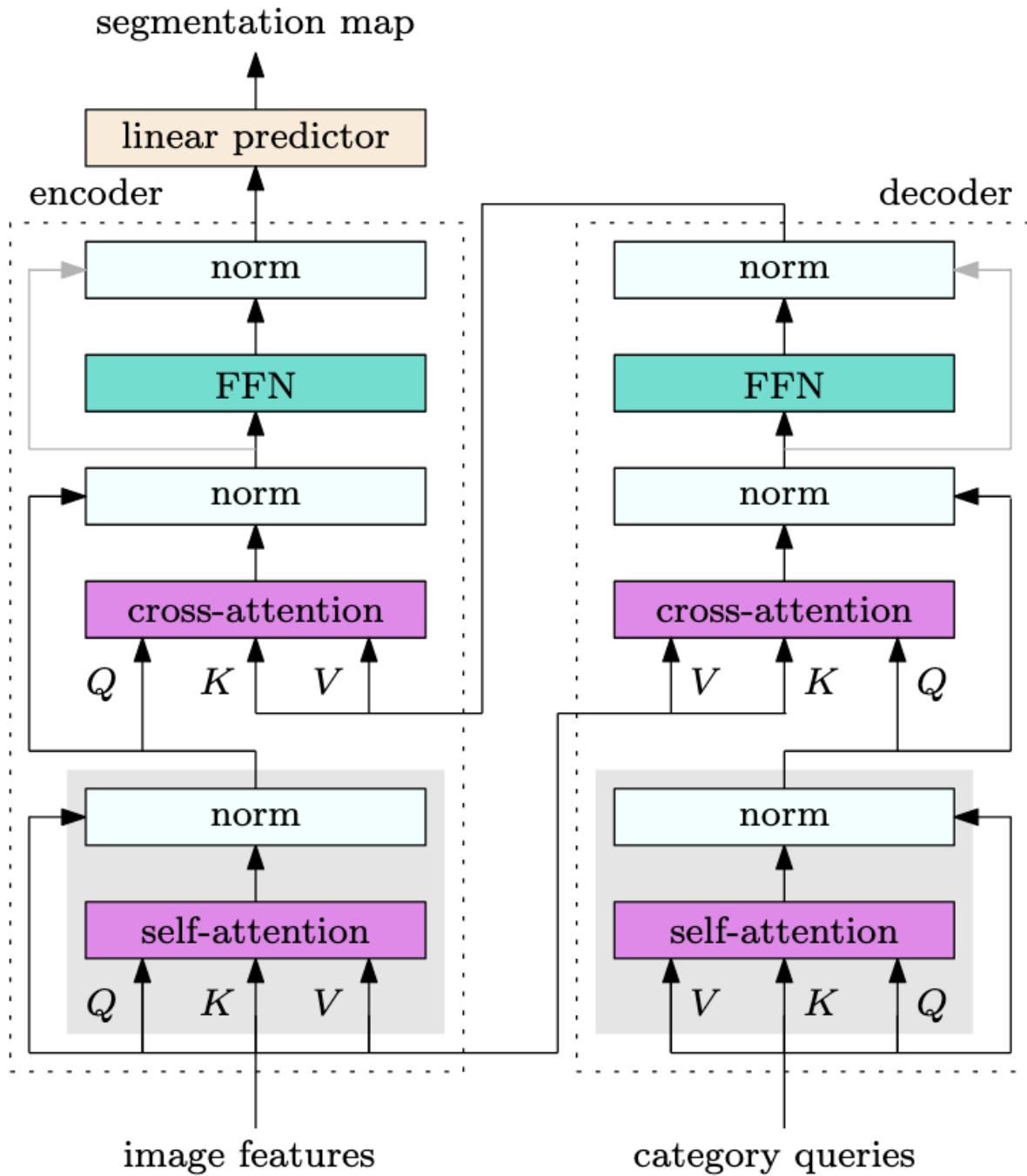


Figure 2.16: Segmentation transformer. From [9]



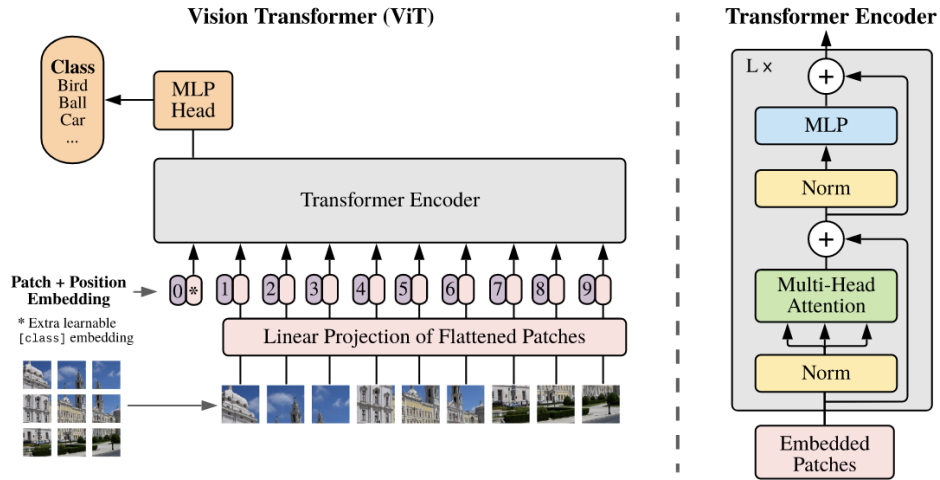


Figure 2 .17: Illustrating the ViT architecture. From [96]

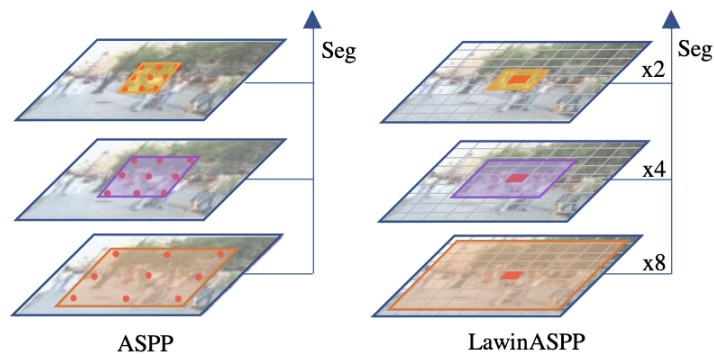


Figure 2 .18: Illustrating the ViT architecture. From [96].

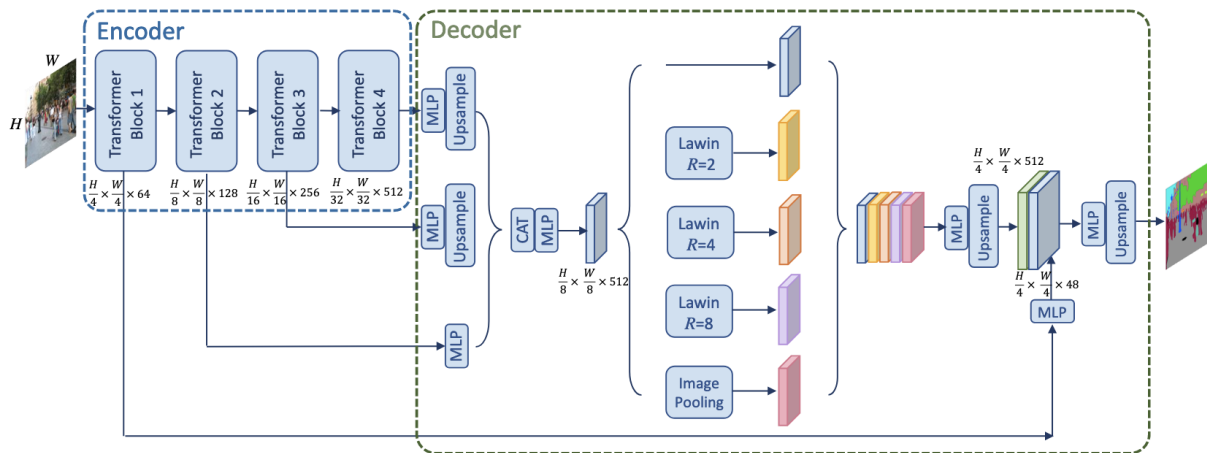


Figure 2 .19: Illustrating the Lawin Transformer architecture. . From [97].

problem, Chen et. al. suggest a Vision Transformer Adapter (ViT-Adapter) [34], which, by incorporating inductive biases via an extra design, may correct the shortcomings of ViT and attain performance equivalent to vision-specific models. Their framework’s main component is a simple transformer that can be pre-trained using multimodal data. A modality-specific adapter is used to transfer the data and previous knowledge of the tasks into the model during fine-tuning on downstream tasks, making the model appropriate for these activities. ViT-Adapter is evaluated on a variety of downstream tasks, such as semantic segmentation, instance segmentation, and object detection. Regarding the task of semantic segmentation, it achieves new state-of-the-art results on Cityscapes (85.2% mIoU) and on ADE20K val (60.5 % mIoU)

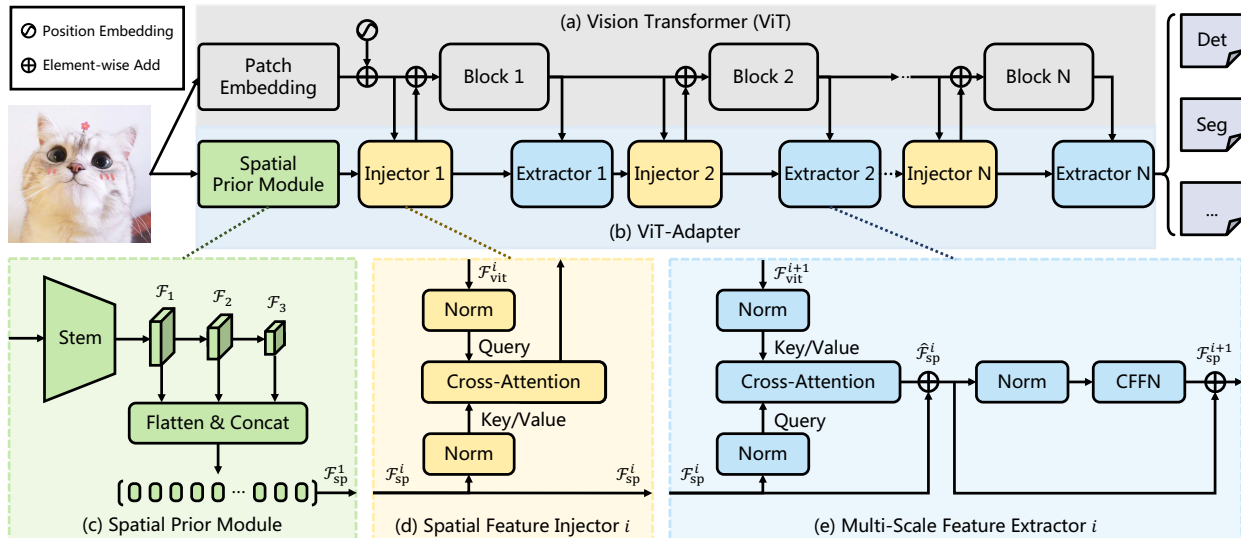


Figure 2.20: Overall architecture of ViT-Adapter. (a) The ViT, whose encoder layers are divided into  $N$  equal blocks for feature interaction; (b) ViT-Adapter, which contains three key components; (c) The spatial prior module, which is used to model local spatial contexts from the input image; (d) The spatial feature injector for incorporating image prior into the ViT; (e) The multi-scale feature extractor for reconstructing fine-grained multi-scale features from the single-scale features of ViT. From [34].

## 2.6 Comparison

To give a clear view on the performance of each model, Table 3.1 depicts the comparison results of several state-of-the-art semantic segmentation models on various datasets. The models are presented in chronological order. The performance metric is mIoU. We can observe that ViT-Adapter achieves state-of-the-art results on both Cityscapes and ADE20K dataset. In this thesis, we are going to use HRNetV2 for the implementation of our method.

<sup>1</sup>When model is learned on the train set it achieves 80.4%, whereas when it is learned on the train+val set it achieves 81.6%

Model	Year	Dataset	mIoU
FCN-VGG16 [70]	2014	Pascal VOC 2012 [100]	62.2%
DeepLab	2014	Pascal VOC 2012 [100]	71.6%
DeconvNet [40]	2015	Pascal VOC 2012 [100]	72.5%
U-Net [84]	2015	ISBI cell tracking challenge 2015 PhC-U373 DIC-HeLa	92% 77.5%
DilatedNet [78]	2016	Pascal VOC 2012 [100]	73.9%
SegNet [85]	2016	CamVid road scene segmentation [101] SUN RGB-D indoor scene segmentation [102]	60.1% 31.84%
PSPNet [12]	2017	PASCAL VOC 2012 [100] Cityscapes [7]	85.4% 80.2%
FC-DenseNet103 [89]	2017	CamVid road scene segmentation [101] Gatech [103]	66.9% 79.4%
GSCNN [94]	2019	Cityscapes [7]	82.8%
HRNetV2 [6]	2020	Cityscapes [7] PASCAL-Context [8] ACDC [11]	81.6% <sup>1</sup> 54.0% 75.0%
HRNetV2+OCR+ [9]	2020	Cityscapes [7] PASCAL-Context [8]	84.5% 56.2%
Lawin Transformer [97]	2022	Cityscapes [7] ADE20K [99]	84.4% 56.2%
ViT-Adapter [34]	2022	Cityscapes [7] ADE20K [99]	85.2% 60.5%
SwiftNet [95]	2022	ACDC[11]	82.82%

Table 3.1: Comparison results of different semantic segmentation models in terms of mIoU.



# Chapter 4

## The Proposed Method: Offset Vector - Based Model

---

1	Main Idea . . . . .	68
2	Seed Pixel Identification . . . . .	68
3	Confidence Loss . . . . .	71
4	Network Architecture . . . . .	71

---

## 1 Main Idea

In this section, we will introduce a new approach for improving semantic predictions. Based on knowledge about the high regularity of real scenes, we propose a method for improving class predictions by learning to selectively exploit information from coplanar pixels. In particular, based on the piece planarity prior from [10], we introduce a Lemma which claims that for each pixel, there is a seed pixel which shares the same prediction with the former. As a result, we design a network with two heads. A similar approach was used for the task of depth estimation on [10]

The new method is applied to HRNetV2 [6]. We employ the main body from HRNet (as depicted in Fig. 2.12), but we avoid explicitly predicting classes. Instead, we use it as a suitable output for defining interactions between pixels based on planarity priors. In particular, the first head of the network outputs four-resolution representations, which are then concatenated as shown in Fig 2.14b. Then this combined high-resolution representation is used as input to the last layer, which outputs pixel-level logits. Afterwards, these logits are converted to classes. Predicting logits is driven by the fact that two pixels  $\mathbf{p}$  and  $\mathbf{q}$  in the same class have ideally similar logit representations. If the pixels belong to the same class, applying the plane coefficient representation of  $\mathbf{q}$  for estimating class at the point of  $\mathbf{p}$  results in a valid prediction.

We leverage this property by learning to identify seed pixels which belong to the same class as the inspected pixel, whenever they exist, in order to selectively use the logits of these pixels for improving the predicted class. This idea is motivated by a piecewise planarity prior [10], which claims that for every pixel  $\mathbf{p}$  with an associated 3D plane, there exists a seed pixel  $\mathbf{q}$  in the neighborhood of  $\mathbf{p}$  which is also associated with the same plane as  $\mathbf{p}$ . Based on this prior, it is straightforward that for each pixel, there is a seed pixel which shares the same prediction with the former. In order to predict classes with this scheme, we need to find the regions where the prior is valid. Furthermore, in order to point out the seed pixels in these regions, we must predict the offset vector  $\mathbf{o}(\mathbf{p}) = \mathbf{q} - \mathbf{p}$  for each pixel  $\mathbf{p}$ . To account for possible deviations from precise local planarity, the resultant prediction is adaptively fused with the initial prediction from the first head using a learnt confidence map. As a result, we design a second head that generates a dense offset vector field and a confidence map. Thus, we have an offset vector-based HRNetV2. This idea will be implemented in different parts of HRNetV2 as shown in Fig. 1.1. We see the main body of the network in Fig. 1.1, where the branch occurs in the second (Fig. 1.1a), in the third (Fig. 1.1b) or in the fourth (Fig. 1.1c) stage of the network respectively. It is notable that the two heads don't share their weights.

Our network estimates classes by selectively combining predictions from each pixel and its corresponding seed pixel. The predicted offsets are utilized to resample the predictions from the first head and generate a second class prediction at seed location. The class predictions from the two heads are then fused adaptively using the confidence map as fusion weights. Finally, it is worth mentioning that, by supervising the fused class prediction, supervision on the offsets and confidence map is applied implicitly. An overview of this architecture is depicted in Fig. 1.2

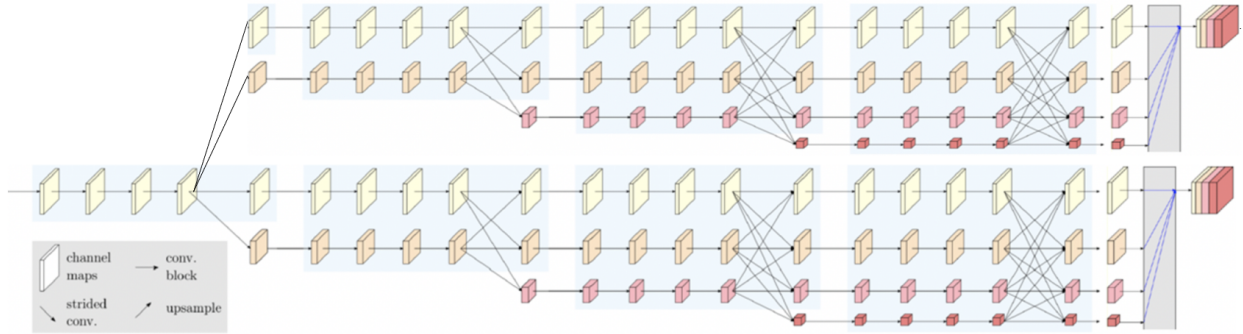
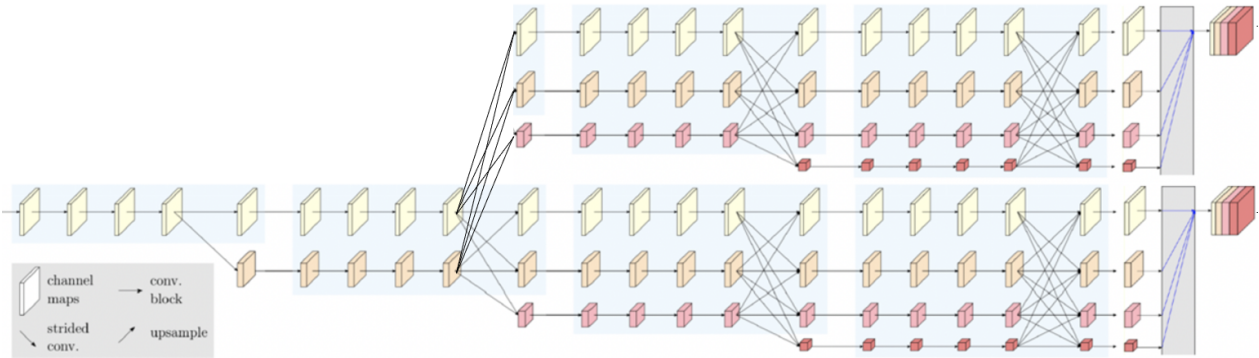
We will evaluate our method on 2 datasets for supervised semantic segmentation :

- Cityscapes [7]
- ACDC [11]

## 2 Seed Pixel Identification

Let us assume we have one pixel  $\mathbf{p}$  which belongs to segment of a semantically segmented image. By definition, every other pixel on this segment has the same class value. Thus, ideally, in order to get all of the class values accurate, the network only has to predict the class at one of these pixels,  $\mathbf{q}$ . This pixel can be interpreted as the seed pixel that describes the segment-class. Finally, we let the network find this seed pixel and the corresponding region.

This idea is based on Piecewise Planarity Prior from [10]. We define the following Lemma:

(a) Branch occurs in the 2<sup>nd</sup> stage of HRNetV2

(b) Branch occurs in the middle of HRNetV2

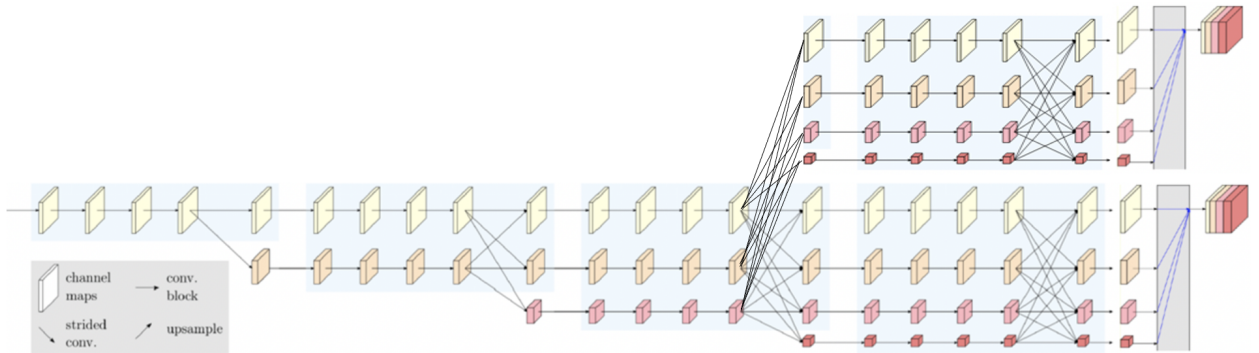
(c) Branch occurs in the 4<sup>th</sup> stage of HRNetV2

Figure 1 .1: Offset vector-based HRNetV2

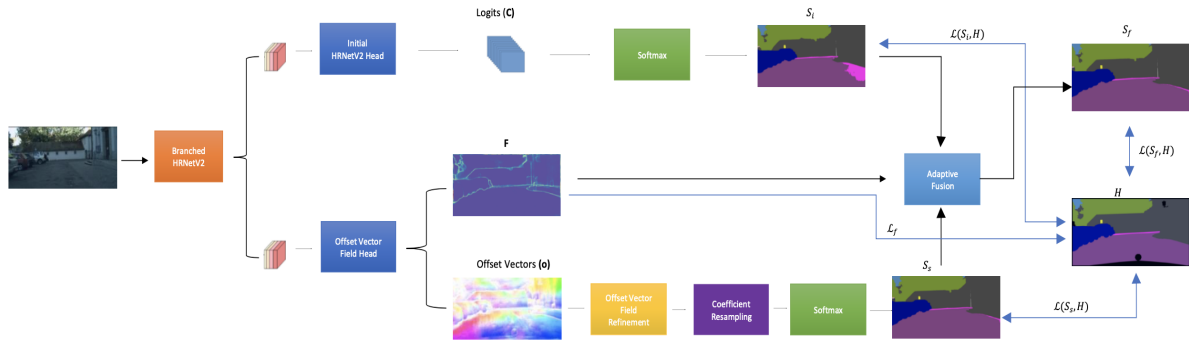


Figure 1 .2: **An overview of our full method.** Offset vector-based HRNetV2, whose architecture is shown in Fig. 1 .1, consists of two output heads. The first head outputs pixel-level Logits ( $C$ ), while the second head outputs a dense offset vector field ( $o$ ) identifying positions of seed pixels along with a confidence map ( $F$ ). Then, the coefficients of seed pixels are used to predict classes at each position. The resulting prediction ( $S_s$ ) is adaptively fused with the initial prediction ( $S_i$ ) using the confidence map  $F$  to compute the final prediction  $S_f$

### Lemma 2 .1

For every pixel  $\mathbf{p}$  with an associated 2D segmentally segmented image, there exists a seed pixel  $\mathbf{q}$  in the neighborhood of  $\mathbf{p}$  which shares the same prediction with the former.

In general, there may be numerous seed pixels for  $\mathbf{p}$  or none at all. Given that the Lemma holds, semantic segmentation task for  $\mathbf{p}$  can be solved by identifying  $\mathbf{q}$ . For this reason, we let our network predict the offset vector  $\mathbf{o}(\mathbf{p}) = \mathbf{q} - \mathbf{p}$ . Thus, we design our model so that it features a second, offset head and let this offset head predict a dense offset vector field  $\mathbf{o}(\mathbf{u}, \mathbf{v})$ . In all of the model versions, the two heads of the network share a common main body and then they follow different paths. We resample the initial logits  $C_i$ , being predicted by the first head, using the estimated offset vector field via:

$$C_s(p) = C_i(p + o(p)) \quad (2 .1)$$

To manage fractional offsets, bilinear interpolation is used. The resampled logits are then used to compute a second semantic segmentation prediction:

$$S_s(u, v) = h(C_s(u, v), u, v) \implies S_s(p) = S_i(p + o(p)) \quad (2 .2)$$

based on the seed locations. In our experiment,  $h = \text{softmax}$ .

Due to the fact that the prior is not always correct, the initial semantic prediction  $S_i$  may be preferred to the seed-based prediction  $S_s$ . To account for such cases, the second head additionally predicts a confidence map  $F(u, v) \in [0, 1]$ , which represents the model's confidence in adopting the predicted seed pixels for semantic segmentation via  $S_s$ . By adaptively merging  $S_i$  and  $S_s$ , the confidence map is used to compute the final prediction:

$$S_f(p) = (1 - F(p))S_i(p) + F(p)S_s(p) \quad (2 .3)$$

We apply supervision to each of  $S_f$ ,  $S_s$ , and  $S_i$  in our model, by optimizing the following loss:

$$\mathcal{L}_{\text{semantic}}(p) = \mathcal{L}(S_f, H) + \kappa \mathcal{L}(S_s, H) + \lambda \mathcal{L}(S_i, H) \quad (2 .4)$$

with  $\kappa$  and  $\lambda$  being hyperparameters and  $H$  denotes the GT train ids of each class for each pixel. In this way, we encourage the Initial HRNetV2 head to output an accurate representation across all pixels, even when



they have a high confidence value, and the offset vector head to learn high confidence values for pixels for which the Lemma 2 holds and low confidence values for pixels for which the Lemma 2 does not.

This formulation, however, comes with a drawback. The model is not supervised directly on the offsets. In fact, it could just predict zero offsets everywhere and yet give valid  $S_s$  and  $S_f$  predictions that are equal to  $S_i$ . Since the initial predictions  $S_i$  are erroneously smoothed around semantic boundaries due to the regularity of the mapping  $f_\theta$  in the case of neural networks, this undesirable behavior is avoided in practice. As a result, predicting a non-zero offset that points away from the boundary provides a lower value for  $\mathcal{L}_{semantic}$  for pixels on either side of the boundary. This happens due to the fact that such an offset utilizes a seed pixel for  $S_s$  that is further away from the border and suffers from reduced inaccuracy due to smoothing. These non-zero offsets are also transmitted from the boundaries to the inner sections of areas with smooth segments, helping the network in predicting non-trivial offsets due to the regularity of the mapping that forms the offset vector field.

Lastly, before resampling the logit maps, we cascade the offset vectors numerous times. The idea for this cascaded refinement is based on the fact that seed pixels within the same segment should converge to the segment’s center, which aids in the accumulation of information from additional pixels in predicting the segment’s class. Cascading the offsets does not harm the related class prediction since pixels without a trustworthy seed pixel are already assigned a low confidence value.

### 3 Confidence Loss

Our confidence loss is based on the concept that given a pixel coordinate, its surrounding pixels should be in the same segment. For each pixel  $p$ , we define the confidence loss as follows:

$$\mathcal{L}_f(p) = -\mathbb{1}_{[H(p)=H(p+o(p))]} \log F(p) - \mathbb{1}_{[H(p)\neq H(p+o(p))]} \log(1 - F(p)) \quad (3.1)$$

This idea is motivated by the fact that confidence should have large value ,for those pixels whose offset vector points to seed pixels with the same class. Similarly, confidence should have small value ,for those pixels whose offset vector points to seed pixels with different class. When the initial pixel  $\mathbf{p}$  and the seed pixel  $\mathbf{q}$  belong to different classes, then the first term is deactivated and the second one is activated. In this case, we want this seed pixel to have low confidence. This idea is reflected by  $\log(1 - F)$ , because when  $F \rightarrow 0 \implies (1 - F) \rightarrow 1^- \implies -\log(1 - F) \rightarrow 0^+ \implies \mathcal{L}_f \rightarrow 0^+$ . Similarly, when  $F \rightarrow 1 \implies \mathcal{L}_f \rightarrow +\infty$ . Accordingly to the first case, when  $\mathbf{p}$  and  $\mathbf{q}$  belong to the same class, then the second term deactivates. In this case, we want this seed pixel to have high confidence. When  $F \rightarrow 0 \implies \log(F) \rightarrow -\infty \implies \mathcal{L}_f \rightarrow +\infty$ . Similarly, when  $F \rightarrow 1 \implies \mathcal{L}_f \rightarrow 0^+$ .

To sum up, the complete loss is:

$$\mathcal{L}_{final} = \mathcal{L}_{semantic} + \mathcal{L}_f \quad (3.2)$$

## 4 Network Architecture

The proposed network consists of the following components:

- **Main Body before the branching:** Our networks structure before the two-way branching is identical to the HRNet [6]
- **After branching:** Offset Vector- based HRNetV2 consists of two output heads. Each head of our network follows the same structure with HRNetV2. The first head outputs pixel-level logits (C), while the second head outputs a dense offset vector field (o) identifying positions of seed pixels along with a confidence map (F). Then, the coefficients of seed pixels are used to predict classes at each position. The resulting prediction ( $S_s$ ) is adaptively fused with the initial prediction ( $S_i$ ) using the confidence map to compute the final prediction  $S_f$ .

- **Initial HRNetV2 Head** : As in HRNetV2 [6], the last layer of the first head output 19-channels ( $19 \times H \times W$ ), e.g one for each class. This is due to the fact that both Cityscapes and ACDC have 19 classes.
- **Offset Vector field Head**: The final layer of this head has been modified to output three channels ( $3 \times H \times W$ ), two for the offset vector field and one for the confidence. Tanh layers restrict the offset vector field, which means that offset vector values belong to  $[-1,1]$ . A sigmoid layer is applied to the confidence map.

# Chapter 5

## Experimental Results

---

1	Dataset . . . . .	<b>74</b>
1.1	Cityscapes . . . . .	74
1.2	Adverse Conditions Dataset with Correspondences (ACDC) . . . . .	74
2	Evaluation Metrics . . . . .	<b>75</b>
3	Implementation Details . . . . .	<b>75</b>
4	Qualitative Results . . . . .	<b>79</b>
5	Baseline Experiments . . . . .	<b>84</b>
6	Comparison with State of the Art . . . . .	<b>84</b>
6.1	Cityscapes . . . . .	84
6.2	ACDC . . . . .	87
7	Ablation Study . . . . .	<b>89</b>
8	Conclusions . . . . .	<b>89</b>

---

# 1 Dataset

In this section, we present the datasets used to evaluate our offset vector-based model. The main datasets used for training and testing are the following :

- Cityscapes [7]
- ACDC [11]

## 1.1 Cityscapes

The Cityscapes dataset [7] is a challenging dataset, tasked for urban scene understanding. It includes semantic, instance-wise, and dense pixel annotations for 30 classes divided into eight groups (flat surfaces, humans, vehicles, constructions, objects, nature, sky, and void). From these 30 classes only 19 classes are used for parsing evaluation. Around 5000 high quality pixel-level finely annotated images and 20000 coarsely annotated images make up the collection. During multiple months, daytimes, and ideal weather circumstances, data was collected in 50 cities. The finely annotated 5000 images are divided into 2.975, 500, 1.525 images for training, validation and testing respectively. It was initially shot as video, therefore the frames were hand-picked to include a high number of dynamic elements, a changing set arrangement, and a changing background.

## 1.2 Adverse Conditions Dataset with Correspondences (ACDC)

The ACDC dataset [11] is a demanding dataset, used for training and testing semantic segmentation methods on adverse visual conditions. It consists of a big collection of 4006 images divided evenly between four frequent unfavorable conditions: fog, dark, rain, and snow. A high-quality fine pixel-level semantic annotation, a corresponding image of the same scene obtained under normal conditions, and a binary mask that distinguishes between intra-image regions of clear and ambiguous semantic information are included with each adverse-condition image. As a result, ACDC can do both basic semantic segmentation and uncertainty-aware semantic segmentation. It directly inherits the class definitions from Cityscapes. It consists of 19 semantic classes, coinciding exactly with the evaluation classes of the Cityscapes dataset. Detailed annotation per class statistics are presented in Fig 1.1. Classes outside of this set of objects are represented by a fall-back label, and they are not utilized in training or testing . ACDC is manually split into four sets corresponding to the examined conditions: 1000 foggy, 1006 nighttime, 1000 rainy and 1000 snowy images. Each set of each adverse condition is split into 400 training, 100 validation and 500 test images, except the nighttime set with 106 validation images. This results in a total of 1600 training and 406 validation images with public annotations and 2000 test images with annotations withheld for benchmarking purposes [11]. There are two final annotation outputs are twofold:

- the final semantic annotation
- a binary invalid mask, which enables the new task of uncertainty-aware semantic segmentation

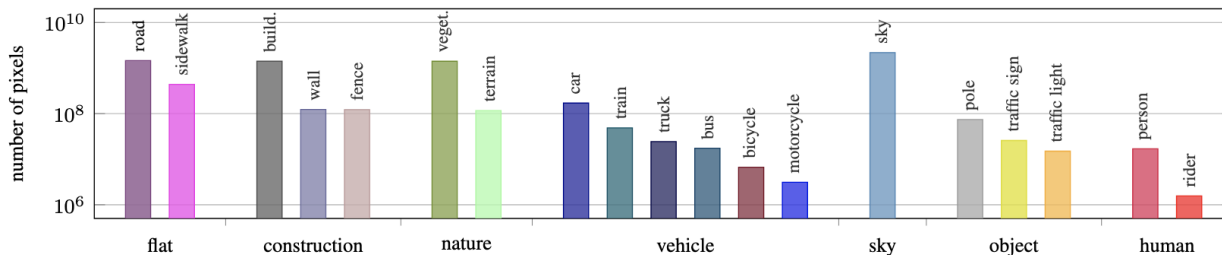


Figure 1.1: Number of finely annotated pixels per class in ACDC. From [11]

## 2 Evaluation Metrics

The mean of class-wise intersection over union (mIoU) is adopted as the evaluation metric. In addition to the mean of class-wise intersection over union (mIoU), we report other three scores on the test set: IoU category (cat.), iIoU class (cla.) and iIoU category (cat.)

## 3 Implementation Details

All versions of our network consist of two heads. The first head outputs 19 channels, one for each class. The second head outputs three channels: one for each coordinate of the offset vectors and one for confidence. These two heads follow the structure of HRNetV2. Both our Offset Vector - Based HRNetV2 and the Baseline HRNetV2 are initialized with pre-trained ImageNet [71] weights. This initialization is important to achieve competitive results as in [6]. In some extra experiments, we follow another approach by freezing, during the whole training process, both main body’s and initial head’s weights. Depending on which dataset we use, the freed part of our model is initialized with the corresponding Baseline’s final pre-trained weights. The only part that will be trained is the second head, which is initialized with pre-trained ImageNet weights. As indicated in Table 5.1, in Table 5.2, as well as in Table 5.3 our current design has four phases (excluding the conventional stem and head). Modularized blocks are repeated 1, 1, 4, and 3 times for each of the four levels of the Initial Head, accordingly. On the other hand, extra heads’ modularized blocks are repeated 1,3 and 2 times for each of the last three levels of the Offset Vector Head, since memory problems occurred. The extra head imitates the structure of the corresponding stages. For the first, second, third, and fourth phase, the modularized block has 1 (2, 3, and 4) branches. Each branch has a varied resolution and is made up of one multi-resolution fusion unit and four residual units. To be clear, Figure 2.13 might be used to understand the fusion unit (after each modularized block), which is not shown in the tables. Each cell in the table has three parts:

- the residual unit (||)
- the repetition times of the residual units (second number)
- the repetition times of the modularized blocks (final number)

The main body’s architecture of the 1<sup>st</sup> Version Offset Vector Based HRNet is depicted below in Fig. 5.1:

Version 1					
Heads	Resolution	Stage 1	Stage 2	Stage 3	Stage 4
HRNetV2 Head	4 ×	$\begin{bmatrix} 1 \times 1, 64, \\ 3 \times 3, 64, \\ 1 \times 1, 256 \end{bmatrix} \times 4 \times 1$	$\begin{bmatrix} 3 \times 3, C, \\ 3 \times 3, C \end{bmatrix} \times 4 \times 1$	$\begin{bmatrix} 3 \times 3, C, \\ 3 \times 3, C \end{bmatrix} \times 4 \times 4$	$\begin{bmatrix} 3 \times 3, C, \\ 3 \times 3, C \end{bmatrix} \times 4 \times 3$
	8 ×		$\begin{bmatrix} 3 \times 3, 2C, \\ 3 \times 3, 2C \end{bmatrix} \times 4 \times 1$	$\begin{bmatrix} 3 \times 3, 2C, \\ 3 \times 3, 2C \end{bmatrix} \times 4 \times 4$	$\begin{bmatrix} 3 \times 3, 2C, \\ 3 \times 3, 2C \end{bmatrix} \times 4 \times 3$
	16 ×			$\begin{bmatrix} 3 \times 3, 4C, \\ 3 \times 3, 4C \end{bmatrix} \times 4 \times 4$	$\begin{bmatrix} 3 \times 3, 4C, \\ 3 \times 3, 4C \end{bmatrix} \times 4 \times 3$
	32 ×				$\begin{bmatrix} 3 \times 3, 8C, \\ 3 \times 3, 8C \end{bmatrix} \times 4 \times 3$
Offset Vector Field Head	4 ×		$\begin{bmatrix} 3 \times 3, C, \\ 3 \times 3, C \end{bmatrix} \times 4 \times 1$	$\begin{bmatrix} 3 \times 3, C, \\ 3 \times 3, C \end{bmatrix} \times 4 \times 3$	$\begin{bmatrix} 3 \times 3, C, \\ 3 \times 3, C \end{bmatrix} \times 4 \times 2$
	8 ×		$\begin{bmatrix} 3 \times 3, 2C, \\ 3 \times 3, 2C \end{bmatrix} \times 4 \times 1$	$\begin{bmatrix} 3 \times 3, 2C, \\ 3 \times 3, 2C \end{bmatrix} \times 4 \times 3$	$\begin{bmatrix} 3 \times 3, 2C, \\ 3 \times 3, 2C \end{bmatrix} \times 4 \times 2$
	16 ×			$\begin{bmatrix} 3 \times 3, 4C, \\ 3 \times 3, 4C \end{bmatrix} \times 4 \times 3$	$\begin{bmatrix} 3 \times 3, 4C, \\ 3 \times 3, 4C \end{bmatrix} \times 4 \times 2$
	32 ×				$\begin{bmatrix} 3 \times 3, 8C, \\ 3 \times 3, 8C \end{bmatrix} \times 4 \times 2$

Table 5.1: The architecture of the 1<sup>st</sup> Version Offset Vector Based HRNet (main body).

The main body's architecture of the 2<sup>nd</sup> Version Offset Vector Based HRNet is depicted below in Fig. 5.2 :

Version 1					
Heads	Resolution	Stage 1	Stage 2	Stage 3	Stage 4
HRNetV2 Head	4 ×	$\begin{bmatrix} 1 \times 1, 64, \\ 3 \times 3, 64, \\ 1 \times 1, 256 \end{bmatrix} \times 4 \times 1$	$\begin{bmatrix} 3 \times 3, C, \\ 3 \times 3, C \end{bmatrix} \times 4 \times 1$	$\begin{bmatrix} 3 \times 3, C, \\ 3 \times 3, C \end{bmatrix} \times 4 \times 4$	$\begin{bmatrix} 3 \times 3, C, \\ 3 \times 3, C \end{bmatrix} \times 4 \times 3$
	8 ×		$\begin{bmatrix} 3 \times 3, 2C, \\ 3 \times 3, 2C \end{bmatrix} \times 4 \times 1$	$\begin{bmatrix} 3 \times 3, 2C, \\ 3 \times 3, 2C \end{bmatrix} \times 4 \times 4$	$\begin{bmatrix} 3 \times 3, 2C, \\ 3 \times 3, 2C \end{bmatrix} \times 4 \times 3$
	16 ×			$\begin{bmatrix} 3 \times 3, 4C, \\ 3 \times 3, 4C \end{bmatrix} \times 4 \times 4$	$\begin{bmatrix} 3 \times 3, 4C, \\ 3 \times 3, 4C \end{bmatrix} \times 4 \times 3$
	32 ×				$\begin{bmatrix} 3 \times 3, 8C, \\ 3 \times 3, 8C \end{bmatrix} \times 4 \times 3$
Offset Vector Field Head	4 ×			$\begin{bmatrix} 3 \times 3, C, \\ 3 \times 3, C \end{bmatrix} \times 4 \times 3$	$\begin{bmatrix} 3 \times 3, C, \\ 3 \times 3, C \end{bmatrix} \times 4 \times 2$
	8 ×			$\begin{bmatrix} 3 \times 3, 2C, \\ 3 \times 3, 2C \end{bmatrix} \times 4 \times 3$	$\begin{bmatrix} 3 \times 3, 2C, \\ 3 \times 3, 2C \end{bmatrix} \times 4 \times 2$
	16 ×			$\begin{bmatrix} 3 \times 3, 4C, \\ 3 \times 3, 4C \end{bmatrix} \times 4 \times 3$	$\begin{bmatrix} 3 \times 3, 4C, \\ 3 \times 3, 4C \end{bmatrix} \times 4 \times 2$
	32 ×				$\begin{bmatrix} 3 \times 3, 8C, \\ 3 \times 3, 8C \end{bmatrix} \times 4 \times 2$

Table 5.2: The architecture of the 2<sup>nd</sup> Version Offset Vector Based HRNet (main body).

The main body's architecture of the 3<sup>rd</sup> Version Offset Vector Based HRNet is depicted below in Fig. 5.3 :

Version 1					
Heads	Resolution	Stage 1	Stage 2	Stage 3	Stage 4
HRNetV2 Head	4 ×	$\begin{bmatrix} 1 \times 1, 64, \\ 3 \times 3, 64, \\ 1 \times 1, 256 \end{bmatrix} \times 4 \times 1$	$\begin{bmatrix} 3 \times 3, C, \\ 3 \times 3, C \end{bmatrix} \times 4 \times 1$	$\begin{bmatrix} 3 \times 3, C, \\ 3 \times 3, C \end{bmatrix} \times 4 \times 4$	$\begin{bmatrix} 3 \times 3, C, \\ 3 \times 3, C \end{bmatrix} \times 4 \times 3$
	8 ×		$\begin{bmatrix} 3 \times 3, 2C, \\ 3 \times 3, 2C \end{bmatrix} \times 4 \times 1$	$\begin{bmatrix} 3 \times 3, 2C, \\ 3 \times 3, 2C \end{bmatrix} \times 4 \times 4$	$\begin{bmatrix} 3 \times 3, 2C, \\ 3 \times 3, 2C \end{bmatrix} \times 4 \times 3$
	16 ×			$\begin{bmatrix} 3 \times 3, 4C, \\ 3 \times 3, 4C \end{bmatrix} \times 4 \times 4$	$\begin{bmatrix} 3 \times 3, 4C, \\ 3 \times 3, 4C \end{bmatrix} \times 4 \times 3$
	32 ×				$\begin{bmatrix} 3 \times 3, 8C, \\ 3 \times 3, 8C \end{bmatrix} \times 4 \times 3$
Offset Vector Field Head	4 ×				$\begin{bmatrix} 3 \times 3, C, \\ 3 \times 3, C \end{bmatrix} \times 4 \times 2$
	8 ×				$\begin{bmatrix} 3 \times 3, 2C, \\ 3 \times 3, 2C \end{bmatrix} \times 4 \times 2$
	16 ×				$\begin{bmatrix} 3 \times 3, 4C, \\ 3 \times 3, 4C \end{bmatrix} \times 4 \times 2$
	32 ×				$\begin{bmatrix} 3 \times 3, 8C, \\ 3 \times 3, 8C \end{bmatrix} \times 4 \times 2$

Table 5.3: The architecture of the 3<sup>rd</sup> Version Offset Vector Based HRNet (main body).



Following the same training protocol as in [6], the data are augmented by random cropping (from  $1024 \times 2048$  to  $512 \times 1024$  in Cityscapes and from  $1080 \times 1920$  to  $540 \times 960$  in ACDC), random scaling in the range of  $[0.5, 2]$ , and random horizontal flipping. We use the SGD optimizer with the base learning rate of 0.01, the momentum of 0.9 and the weight decay of 0.0005. The number of epochs used for training is 484. For lowering the learning rate, a poly learning rate policy with a power of 0.9 is applied. The offset vectors are restricted via a tanh layer to have a maximum length of  $\tau$  in normalized image coordinates. We set  $\tau$  to 0.5 by default, branch on 4<sup>th</sup> stage and zero steps of cascaded refinement to the offsets. The confidence map is predicted through a sigmoid layer. For  $S_i$ ,  $S_s$  and  $S_f$  predictions, Ohem Cross Entropy Loss is used. It is remarkable that, at first, the model computed the  $S_i$ ,  $S_s$ ,  $S_f$  predictions directly but that led to low mIoU results. On the other hand, the final model with the best results outputted the  $S_i$ ,  $S_s$ ,  $S_f$  logits. The loss weights  $\lambda$  and  $\mu$  are set to 0.5. In addition, the confidence based loss is applied using the final semantic prediction  $S_f$ . All models are trained on the corresponding Dataset’s train set. The Baseline models are trained for 120K iterations with the batch size of 12 on 4 GPUs and syncBN. On the other other hand, due to memory size problems, the Offset Vector Based models are trained for 120K iterations with the batch size of 8 on 4 GPUs and syncBN.

## 4 Qualitative Results

We provide qualitative results of our method for both Cityscapes (Fig. 4.3, 4.4 and 4.5) and ACDC (Fig. 4.6, 4.7 and 4.8). As we can see, our model not only achieves highly satisfactory results close to GT images, but also outperforms the  $S_i$  predictions outputted by the initial HRNetV2 head. For the vector field’s visualisation we used the color coding described in [104] and shown in 4.1. The color coding of the semantic classes matches Fig. 1.1. Regarding the confidence map’s visualization we use the viridis color maps described in [105] and shown in 4.2.



Figure 4.1: The optical flow field color-coding. Smaller vectors are lighter and color represents the direction. From [104]



Figure 4.2: Viridis color map. The higher the confidence is, the lighter the map is. From [105]

Specifically, we will analyze some predictions of our model in some selected RGB images of **Cityscapes**. At

first, we can observe from the Example shown in 4.3, that the final prediction  $S_f$  of our model is better than the  $S_i$ , as the pole enclosed by the red frame in the latter prediction should not exist. On the other hand,  $S_s$  prediction reduces this error with the help of seed pixels, and, in turn,  $S_f$ , by having higher confidence in the specific  $S_s$  pixel-level prediction, outputs an image far closer to the GT. Another point worth considering, is the Example depicted in 4.4. As we can see from the red frame in the  $S_i$  output, the predicted sidewalk is wrong. On the contrary,  $S_s$  and, as a result,  $S_f$  eliminates this error, by predicting that the pixels in this enclosed region belong to the 'road' class. Last but not least, in the last Cityscapes example (4.5), the sidewalk segment on the right side of the image (red frame) is smoother in  $S_f$  than in  $S_i$ . Moreover, the tree segment on the upper left side of the image (green frame) shown in  $S_i$  image, has been correctly deleted from the  $S_f$  prediction. By observing, also, the blue frame on the left, we can see from the GT image that the car segment is larger than the segment shown in the  $S_i$  one.  $S_f$  tries to enlarge this segment, as we see in the corresponding frame. On the other hand, in the  $S_s$ , we can see some false predicted dots that belong to the tree segment, which do not exist in the  $S_i$ . In turn,  $S_f$  shows higher confidence in the  $S_i$  and outputs the correct segment. Unfortunately, both  $S_i$  and  $S_f$  fail to predict the correct segment in the middle of the image (light green frame), with the former predicting a larger sidewalk segment and the latter trying to eliminate this segment, enlarging erroneously the road segment. Our prediction can not output the correct terrain segment, as the initial prediction  $S_i$  does not predict anywhere this class.

As far as some selected RGB images of **ACDC** are concerned, we can also confirm our previous findings. Particularly, regarding the Example shown in Fig. 4.6, we observe that  $S_i$  outputs some false predictions in the red frame. On the contrary,  $S_f$  predicts correctly the building segment. Moreover, it correctly shrinks the fence segment on the right side of the image. Similarly, in the Fig 4.7,  $S_f$  shrinks the terrain and enlarges the sidewalk segment. Moreover, it does not take into account the false prediction made in the road segment in the  $S_s$  prediction. Finally, the terrain segment wrapped by the red frame in Fig. 4.8 lacks of continuity in the  $S_s$ , as it contains erroneously some sidewalk segments. This discontinuity is fixed by the predictions in the seed pixels and, as a result,  $S_f$  outputs the correct segment.

Last but not least, comparing the offset vectors produced by the 2 datasets, the Cityscapes ones are far better than the ACDC ones, due to the fact that the former contains more clear images than the latter. **ACDC** targets semantic understanding of driving scenes in adverse visual conditions. Under these conditions, in the corresponding regions where the visibility is limited, our models predict small offset vector values.

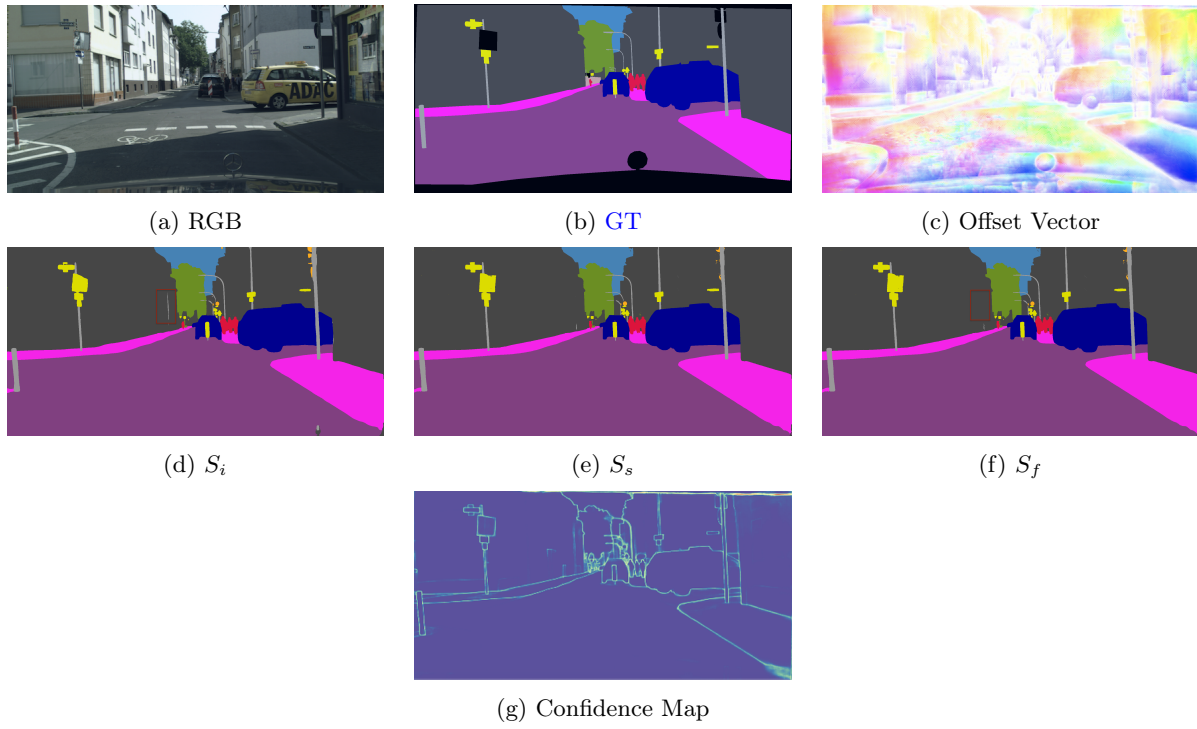


Figure 4 .3: Qualitative results on Cityscapes: Example 1

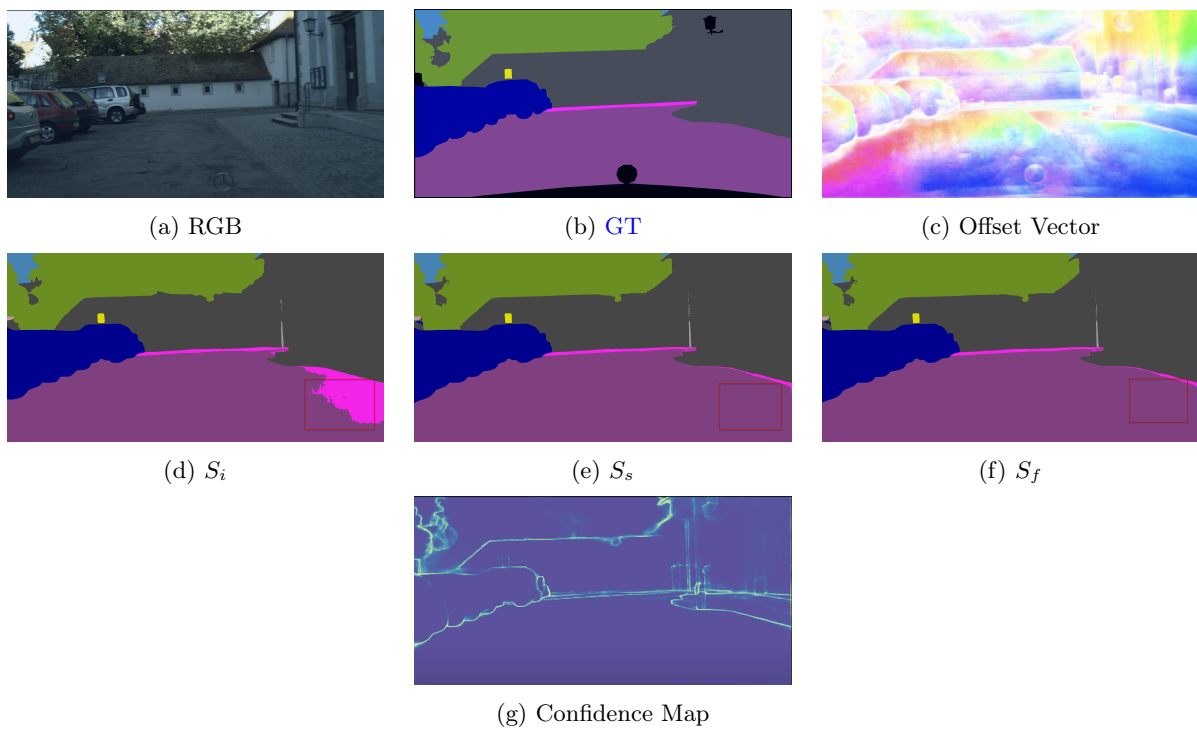


Figure 4 .4: Qualitative results on Cityscapes: Example 2

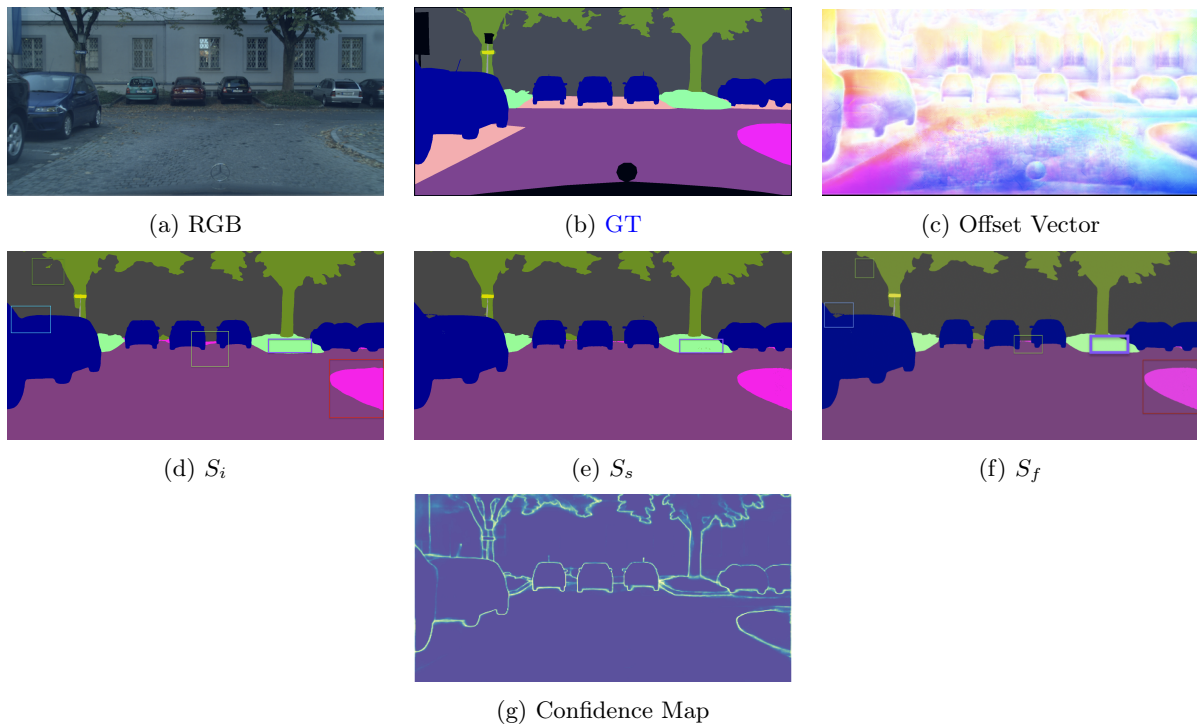


Figure 4.5: Qualitative results on Cityscapes: Example 3

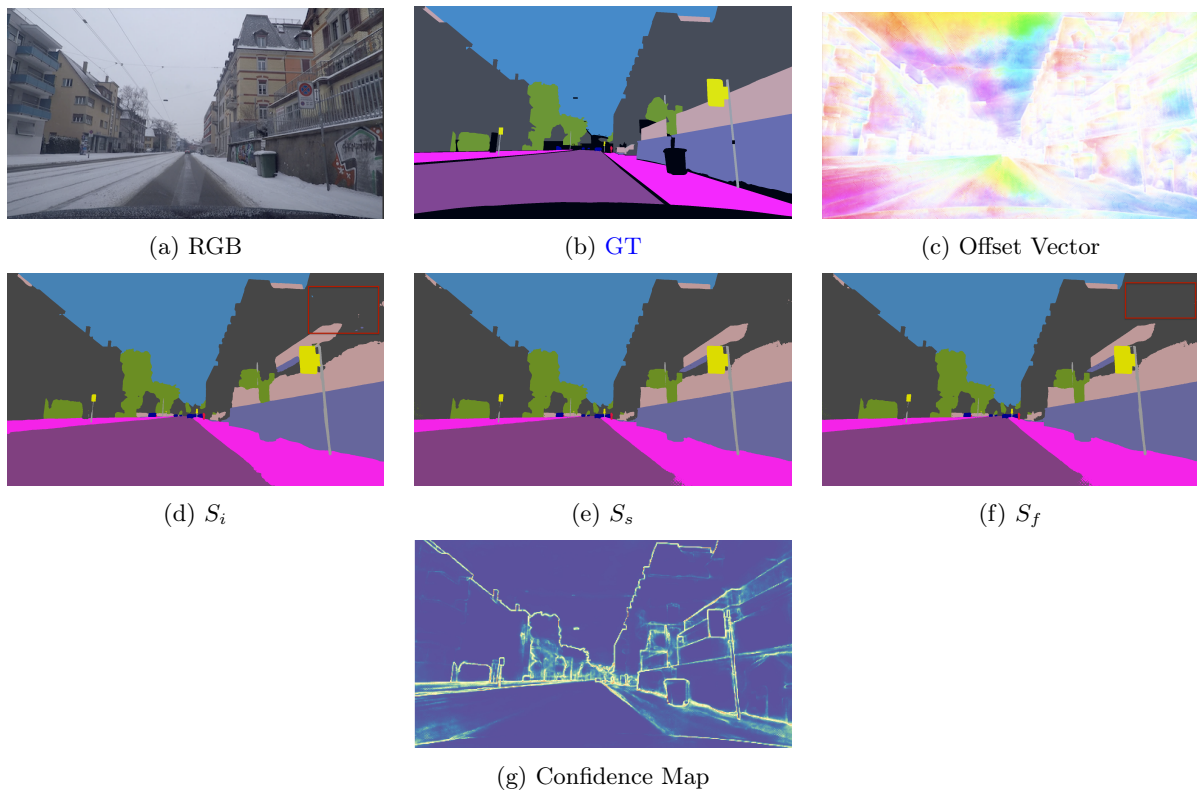


Figure 4.6: Qualitative results on ACDC: Example 1

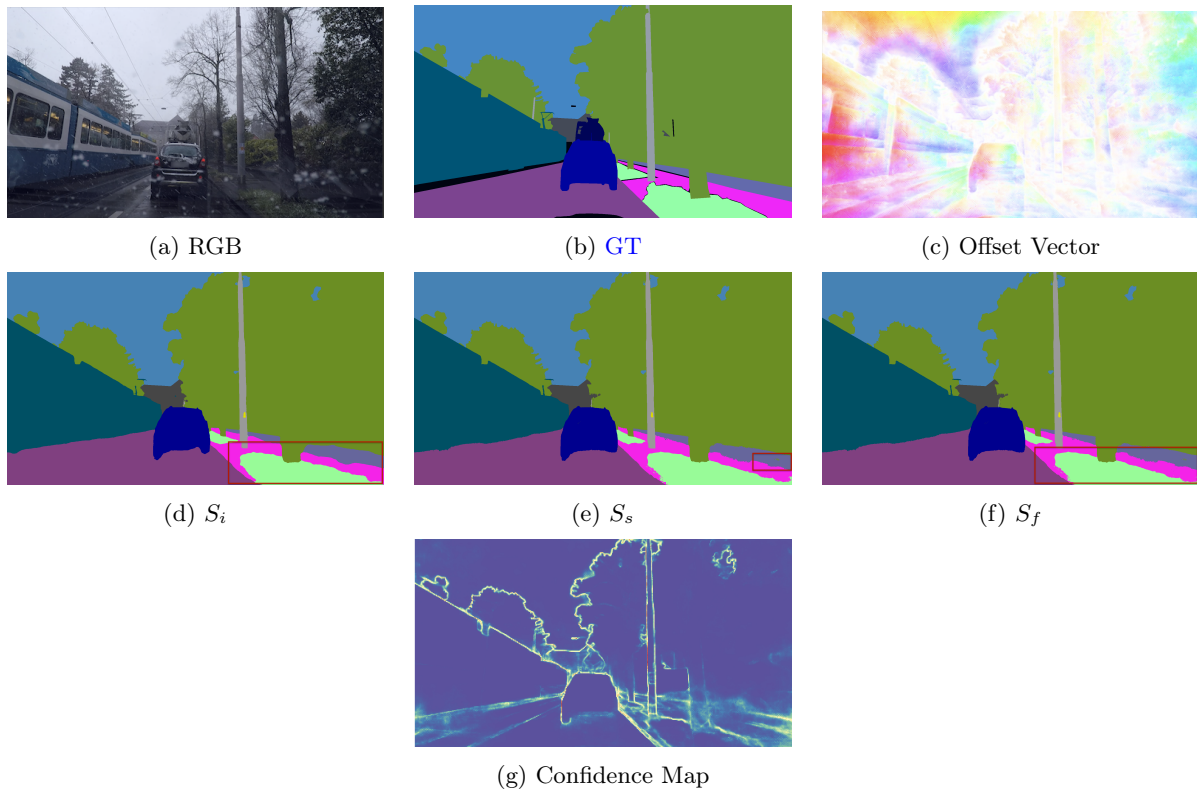


Figure 4 .7: Qualitative results on ACDC: Example 2

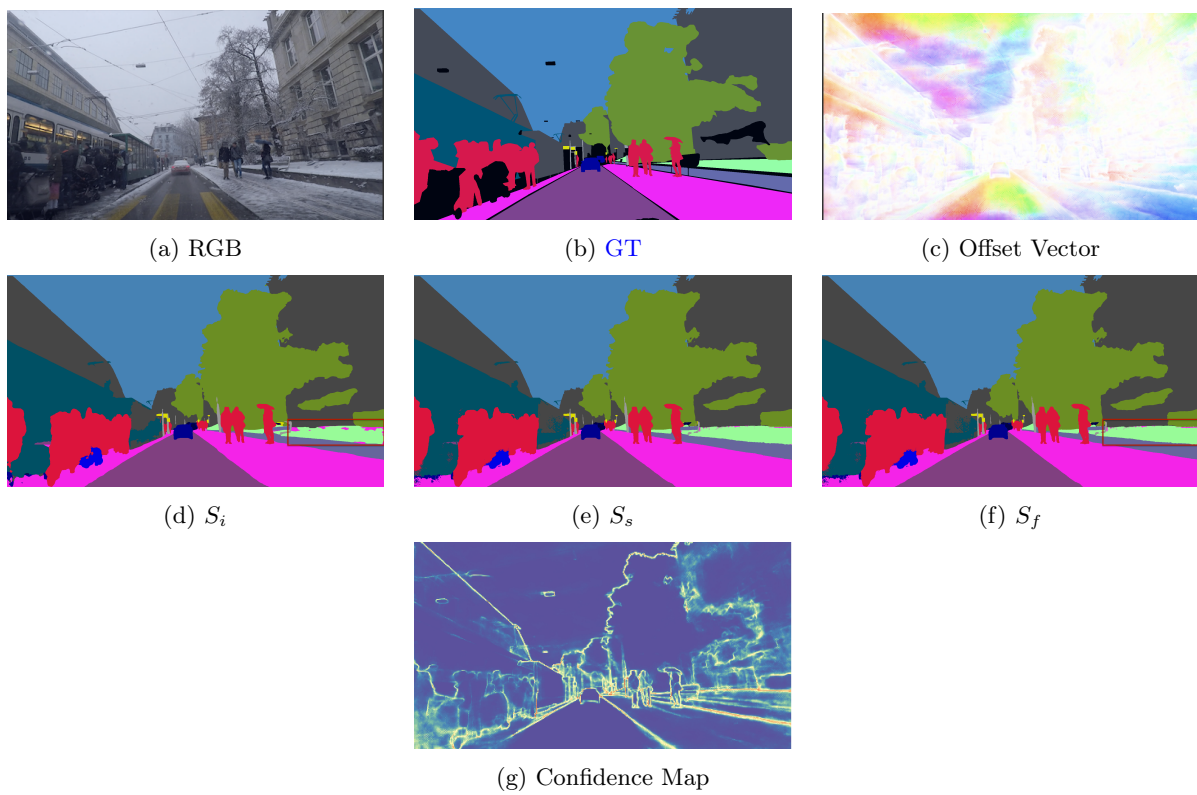


Figure 4 .8: Qualitative results on ACDC: Example 3

## 5 Baseline Experiments

At first, we trained the original HRNetV2 model from scratch on both Cityscapes and ACDC dataset, in order to reproduce the results submitted in [6] and in [11] correspondingly. We used the default parameters specified by the authors. In particular, for our experiments we used as backbone the HRNetV2-W48 model, where 48 means the width, as this model achieved the best performance. The HRNet-W48 is trained on 4 NVIDIA Titan X GPUs and it takes around 80 hours on Cityscapes and around 35 hours on ACDC. As we can see from Table 5.4 the results reproduced successfully on Cityscapes validation and test set. Regarding the ACDC Dataset, the model submitted in [11] was initialized with pre-trained Cityscapes weights. Instead, our model was initialized with pre-trained Imagenet weights. In the following experiments we are going to initialize our model with the latter initialization in order to achieve competitive results.

Model	Backbone	Dataset	MeanIU
HRNetV2 [6]	HRNetV2-W48	Cityscapes [7]	80.4
HRNetV2 (Source Code)	HRNetV2-W48	Cityscapes [7]	80.5
HRNetV2 [6]	HRNetV2-W48	ACDC [11]	75.0
HRNetV2 (Source Code)	HRNetV2-W48	ACDC [11]	70.5

Table 5.4: Test Set Reproduction Results

## 6 Comparison with State of the Art

### 6.1 Cityscapes

The results on Cityscapes which is the major database, which focuses on semantic understanding of urban street scenes, are shown below. We achieve better results on Cityscapes than the initial HRNet under similar training time, outperforming prior state-of-the-art methods across all four standard metrics on both val and test set .

- **Results on the val set** Table 5.5 compares our model with the initial HRNetV2 on the Cityscapes val set in terms of parameter and computation complexity and mIoU class. Our model achieves better performance: 0.6 points over HRNetV2.

Model	Backbone	#param.	GFLOPs	mIoU
HRNetV2 [6]	HRNetV2-W48	65.9M	174	81.8
Ours	HRNetV2-W48	98.8M	234,7	<b>82.4</b>

Table 5.5: Semantic segmentation results on Cityscapes val test (multi-scale and flipping). The GFLOPs is calculated on the input size  $1024 \times 2048$ . They are calculated for Convolution and Linear Layers only.

- **Results on the test set** Table 5.6 compares our method with state-of-the-art methods on the Cityscapes test set. All the results are with six scales and flipping. Two cases w/o using coarse data are evaluated: One is about the model learned on the train set, and the other is about the model learned on the train+val set. In both cases, our offset vector - based HRNetV2-W48 achieves the superior performance, learned **only** on the train set. To become more specific, we achieve a superior relative performance gain of 1.7% in mIoU, 4% in iIoU cla., 0.4% in IoU cat. and 1.7% in iIoU cat.

Table 5.7 analytically compares our approach with HRNetV2’s per class results. As we can see our method achieves better results in the majority of classes. Our offset vector-based model learns an implicit representation of different objects which can benefit the overall semantic segmentation estimation capability of the network.

Qualitative results on Cityscapes support the above findings, as shown in Fig. 6.1. To be more specific, from left to right, we depict the input image, the initial HRNet’s output and our model’s output. Regarding the first row, we can observe that our model tries to enlarge correctly the sidewalk segment, as shown in the red segment. In the second row, our model broadens correctly the traffic sign, the pole and the sidewalk segment (red, green, blue framework correspondingly). Unfortunately,



Model	Backbone	mIoU	iIoU cla.	IoU cat.	iIoU cat.
<i>Model trained on the train set</i>					
PSPNet [12]	D-ResNet-101	78.4	56.7	90.6	78.6
PSANet [13]	D-ResNet-101	78.6	-	-	-
PAN [14]	D-ResNet-101	78.6	-	-	-
AFF [15]	D-ResNet-101	79.1	-	-	-
HRNetV2 [6]	HRNetV2-W48	80.4	59.2	91.5	80.8
Ours	HRNetV2-W48	<b>81.8</b>	<b>61.6</b>	<b>91.9</b>	<b>82.2</b>
<i>Model trained on the train +val set</i>					
GridNet [16]	-	69.5	44.1	87.9	71.1
LRR-4x [17]	-	69.7	48.0	88.2	74.7
DeepLab [18]	D-ResNet-101	70.4	42.6	86.4	67.7
LC [19]	-	71.1	-	-	-
Piecewise [20]	VGG-16	71.6	51.7	87.3	74.1
FRRN [21]	-	71.8	45.5	88.9	75.1
RefineNet [22]	ResNet-101	73.6	47.2	87.9	70.6
PEARL [23]	D-ResNet-101	75.4	51.6	89.2	75.1
DSSPN [24]	D-ResNet-101	76.6	56.2	89.6	77.8
LKM [25]	ResNet-152	76.9	-	-	-
SAC [26]	D-ResNet-101	78.1	-	-	-
DepthSeg [27]	D-ResNet-101	78.2	-	-	-
ResNet38 [28]	WRResNet-38	78.4	59.1	90.9	78.1
BiSeNet [29]	ResNet-101	78.9	-	-	-
DFN [30]	ResNet-101	79.3	-	-	-
PSANet [13]	D-ResNet-101	80.1	-	-	-
PADNet [31]	D-ResNet-101	80.3	58.8	90.8	78.5
DenseASPP [12]	WRResNet-161	80.6	59.1	90.9	78.1
DANet [32]	D-ResNet-101	81.5	-	-	-
HRNetV2 [6]	HRNetV2-W48	81.6	<b>61.8</b>	<b>92.1</b>	82.2
Ours (only on train set)	HRNetV2-W48	<b>81.8</b>	61.6	91.9	<b>82.2</b>

Table 5.6: Semantic segmentation results on Cityscapes test set. Our results are superior in terms of the four evaluation metrics. D-ResNet-101 = Dilated-ResNet-101. We compare our method against SOTA methods as in [6]

Method	road	sidew.	build.	wall	fence	pole	light	sign	veget.	terrain	sky	person	rider	car	truck	bus	train	motorc.	bicycle	mIoU
HRNetV2 [6]	98.73	87.49	93.65	56.48	61.57	71.57	78.76	81.81	93.99	74.11	95.68	87.95	73.72	96.35	69.94	82.52	76.93	70.88	78.02	80.4
Ours	<b>98.74</b>	87.41	<b>93.79</b>	<b>61.65</b>	<b>64.00</b>	71.35	<b>78.98</b>	<b>81.65</b>	<b>94.00</b>	<b>73.42</b>	<b>95.81</b>	<b>87.99</b>	<b>74.36</b>	<b>96.42</b>	<b>74.76</b>	<b>87.70</b>	<b>82.83</b>	<b>71.77</b>	77.86	<b>81.8</b>

Table 5.7: Per Class Results on Cityscapes test set

as we can see from the yellow framework, the initial model predicts better the shape of the man’s hands. In particular, the man holds a newspaper and our model erroneously classifies these pixels into the person class. However, in general, our model has a more accurate prediction. As far as the third set of images is concerned, our model condenses the cyclist’s leg and enlarges the background car’s segment, achieving a better final result. Last but not least, in the last set of images, our model corrects some false predictions made by the initial model, including the reduction of both the car segment (blue framework) and the fence segment (red framework). To sum up, our model surpasses the initial one’s performance.

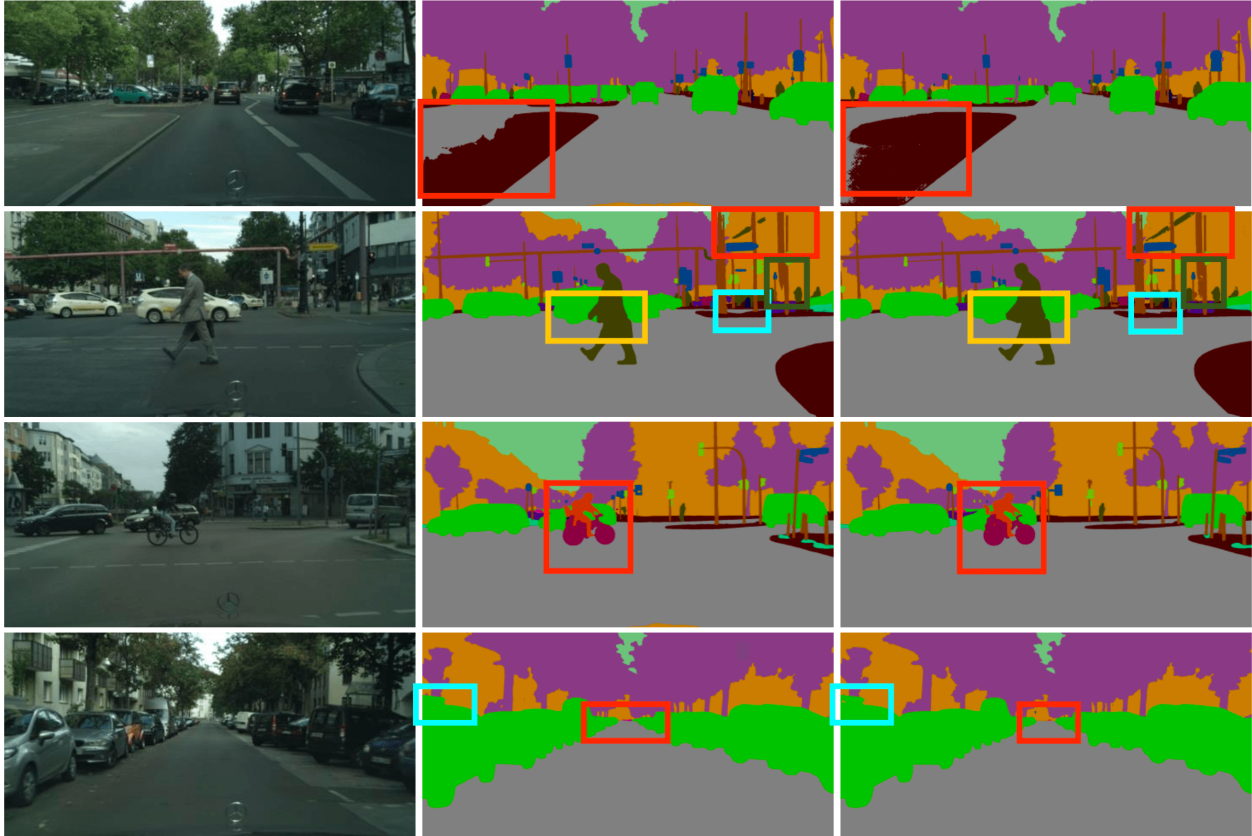


Figure 6 .1: **Qualitative results of selected examples on Cityscapes.** From left to right: image, initial HRNet and ours.



## 6.2 ACDC

The results on [ACDC](#), which targets semantic understanding of driving scenes in adverse visual conditions, are shown below. We also achieve far better results than the initial HRNet under similar training time, outperforming prior state-of-the-art methods on both val and test set. In the following experiments, the initial model is the one initialized with pre-trained Imagenet weights.

- **Results on the val test**

Table 5.8 compares our model with initial HRNet’s method on the ACDC val set in terms of parameter and computation and [mIoU](#) class. Our model achieves also in this dataset better performance 0.41 points over HRNetV2.

Model	Backbone	#param.	GFLOPs	<a href="#">mIoU</a>
HRNetV2	HRNetV2-W48	65.9M	172.9	75.50
Ours	HRNetV2-W48	98.8M	233.1	<b>75.91</b>

Table 5.8: Semantic segmentation results on ACDC val set (multi-scale and flipping). The GFLOPs is calculated on the input size  $1080 \times 1920$ . They are calculated for Convolution and Linear Layers only.

- **Results on the test set**

Table 5.9 compares our approach with state-of-the-art methods on the ACDC test set on All Conditions, as it is trained on all and not on a single condition. As it is observed, our model outperforms the other methods. To become more specific, we achieve a superior relative performance gain of 2.5% in [mIoU](#).

Model	<a href="#">mIoU</a>
RefineNet [22]	65.3
DeepLabv2 [18]	55.3
DeepLabv3+ [33]	70.0
HRNetV2 [6]	70.5
Ours	<b>73</b>

Table 5.9: Semantic segmentation results on ACDC test set. Our results are superior in terms of the [mIoU](#) metrics. We compare our method against SOTA methods as in [11]

Table 5.10 compares analytically our approach with other state of the art method’s per class results. As we can see our method achieves also in this dataset better results in all classes, improving the initial model. We observe the following:

- In snow, road and sidewalk performance is at its lowest, which can be attributed to misunderstanding between the two classes as a result of their similar look. On the other hand, our approach achieves a superior relative performance gain of 1.8 % [mIoU](#) in the sidewalk segment.
- It is more difficult to separate classes at night that are often dark or poorly lit, such as buildings, vegetation, traffic signs, and the sky. This behaviour is observed also in offset vector performance as they have small values when the visibility is limited.
- On foggy days, performance on classes containing small-instance instances, such person, rider, and bicycle, is at its lowest. This is likely because of the combined effect of contrast reduction and poor resolution for examples of these classes that are far from the camera.

Qualitative results on ACDC support the above findings, as shown in Fig. 6.2. To be more specific, from left to right, we depict the input image, the initial HRNet’s output and our model’s output. Regarding the first row, we notice that our model enlarges the bicycle segment, by correcting some false predicted pixels (red framework). Unfortunately, as we can see in the yellow framework, the initial model achieves a more accurate prediction. In particular, a car is depicted in the corresponding area and our model tries to eliminate it. In the second row, we can underline that our model tries to enlarge correctly the sidewalk segments in both red and green framework and reduces the erroneously terrain segment, predicted by the initial model. As far as the third set of images is concerned, our offset vector - based model eliminates correctly the sidewalk

Method	road	sidew.	build.	wall	fence	pole	light	sign	veget.	terrain	sky	person	rider	car	truck	bus	train	motorc.	bicycle	mIoU
RefineNet [22]	92.5	71.2	86.2	39	44.0	53.2	68.8	66.0	85.1	59.3	94.9	65.2	38.5	85.8	53.8	59.7	76.2	47.5	54.5	65.3
DeepLabv2 [18]	88.0	62.3	80.8	37.0	35.1	33.9	49.8	49.5	80.1	50.7	92.5	51.1	26.5	79.9	49.0	41.1	72.2	26.5	44.2	55.3
DeepLabv3+ [33]	93.4	74.8	89.2	53.0	49.0	58.7	71.1	67.4	87.8	62.7	95.9	69.7	36.0	88.1	67.7	71.8	85.1	48.0	59.8	70
HRNetV2 [6]	95.3	80.3	90.5	52.0	53.1	65.1	78.2	74.2	89.2	68.4	96.7	70.6	36.1	88.2	55.9	54.3	88.0	43.8	58.9	70.5
Ours	<b>95.8</b>	<b>82.1</b>	<b>91.3</b>	<b>55.8</b>	<b>54.6</b>	<b>67.6</b>	<b>80.5</b>	<b>77.3</b>	<b>89.7</b>	<b>69.5</b>	<b>96.8</b>	<b>73.4</b>	<b>39.1</b>	<b>89.5</b>	<b>61.9</b>	<b>65</b>	<b>89.4</b>	<b>47.2</b>	<b>60.6</b>	<b>73</b>

Table 5.10: Per Class Results on ACDC test set

area (red framework), as in the initial image there is no such area. Last but not least, regarding the last set of materials, the initial model classifies incorrectly the sign of the house into the traffic sign class( red framework). On the contrary, our model corrects not only this mistake, but also a discontinuity occurred in the yellow framework. To sum up, our model surpasses the initial one’s performance.

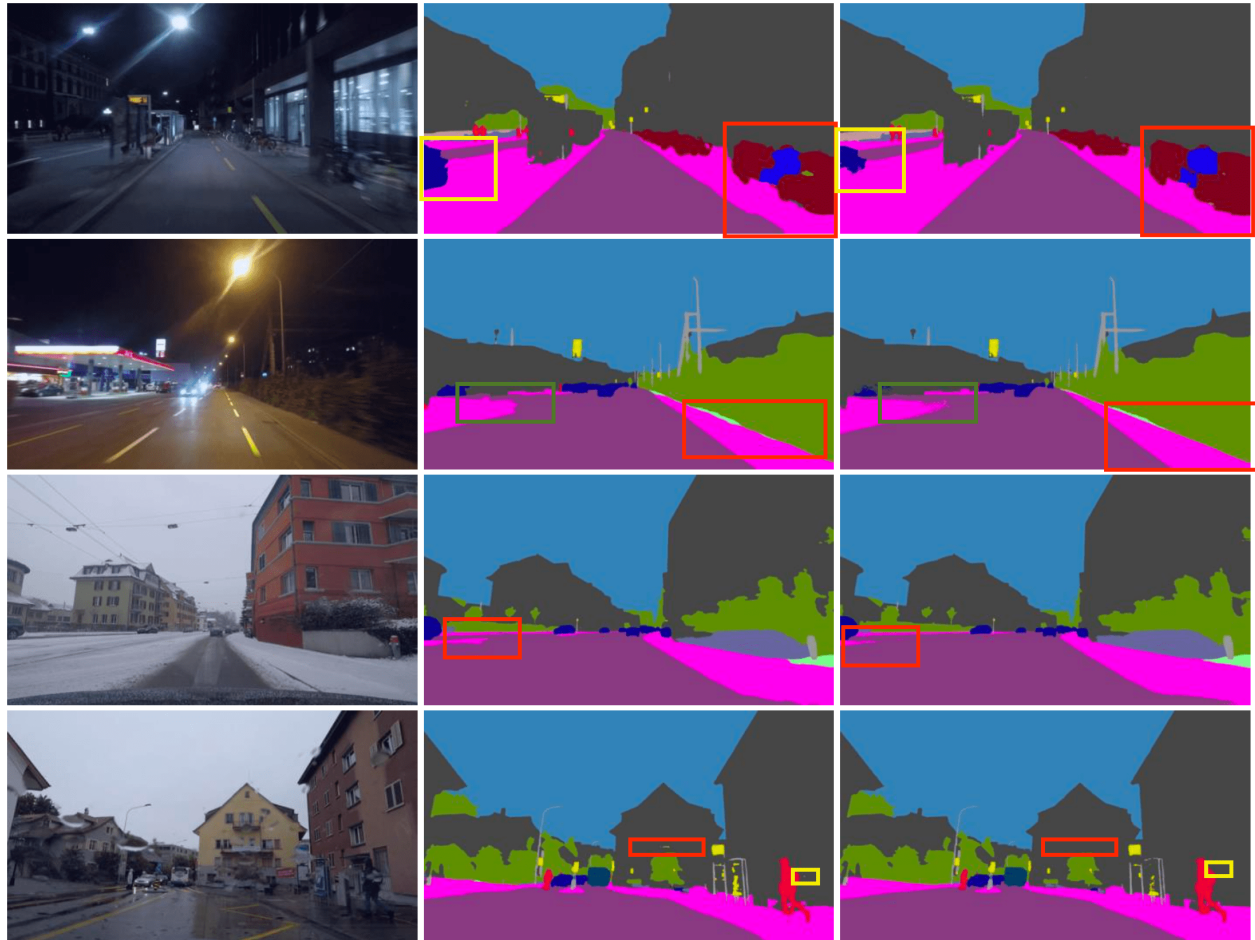


Figure 6.2: Qualitative results of selected examples on ACDC. From left to right: image, initial HRNet and ours.

## 7 Ablation Study

In order to experimentally confirm our design choices for the offset vector - based model, we performed an ablation study, as shown in Table 5.11. We trained and evaluated 7 different variations on Cityscapes. The performance of each model variation in relation to the ground truth images was calculated by means of the **mIoU** ( as defined above). Specifically, the following steps were taken in order to analyze our method;

1. At first, we initialized both heads of the network with the pre-trained Imagenet weights and set the offset vector length equal to 0.5.
2. Secondly, we froze both main body’s and initial head’s weights. The frozen part of our model was initialized with the corresponding Cityscapes final pre-trained weights. The only part trained was the second head, which was initialized with pre-trained ImageNet weights. As shown in Table 5.11, although the performance of our model is higher than the initial single head model’s one, it still remains lower than the case where both heads are trained simultaneously.
3. Then, we deactivated the "Freeze" feature and changed the offset vector length logarithmically, setting both the values 1 and 0.2. We observed that the more the length is the less **mIoU** achieves. This is due to the fact that larger offset vectors point to more distant objects that may affect erroneously the final prediction.
4. Furthermore, we deactivated the **OHEM** Cross Entropy Loss and enabled the simple Cross Entropy Loss. As expected, the performance of the model was lower. **OHEM** penalizes more high loss values and leads to a better training of the model.
5. Lastly, our model casts the utility of cascaded refinement of offsets aside.

Fr	IHW	OHW	V	OV	Ref	OHEM	mIoU
	I					✓	81.83
	I	I	3	0.5		✓	82.4
✓	C	I	3	0.5		✓	82.01
	I	I	3	1		✓	81.79
	I	I	3	0.2		✓	<b>82.80</b>
	I	I	3	0.2			81.96
	I	I	3	0.2	2	✓	81.92

Table 5.11: **Ablation study of components of our method.** "Fr" : Freezed main body’s and initial head’s weights, "IHW": Initial Head Weights, "Ref": cascaded refinement of offsets, "OHW": Offset Head Weights, "V": Version, "OV": Offset Vector, "OHEM": **OHEM** Cross Entropy, I: Imagenet, C:Cityscapes , A: ACDC

## 8 Conclusions

In summary, we conducted a thorough qualitative and quantitative comparison to show the clear advantages of our method over previous state-of-the-art methods in this field. Tables 5.6 and 5.9 verify the superiority of our method. Moreover, Tables 5.7 and 5.10 show that our approach achieves not only better total results but also better per class results in the majority of classes under similar training time. As we mentioned above, our method shares information from seed locations and improves the predicted segments. Also, it learns an implicit representation of different objects which can benefit the overall semantic segmentation estimation capability of the network. Our qualitative results further demonstrate that we achieved a better performance. In fact, not only our idea achieves highly satisfactory results close to GT images, but also outperforms the predictions outputted by the initial HRNetV2’s head. In particular, it classifies some false predicted pixels in the correct classes. Thus, it eliminates discontinuities ( see Fig. 4 .6) and improves the shape as well as the form of the corresponding segments, leading to more realistic results.



# Chapter 6

## Conclusion and Future Work

---

1	Conclusion . . . . .	92
2	Future and Follow Up Works . . . . .	92

---

## 1 Conclusion

In this thesis, we address the problem of semantic segmentation, one of the fundamental topics of computer vision. Recognizing an object within an RGB image means determining some of its distinctive color and texture features. Effects like the color shift due to light absorption, the lens distortion, and chromatic aberrations can alter objects' appearance and mess with users' perception and recognition algorithms. All the more sophisticated methods contribute towards improved performances. While AI still has a long way to go before it can develop models comparable to humans in these kinds of activities, several researchers are consistently outperforming state-of-the-art findings, setting the bar even higher. Most of the related works concentrate on architectural changes to the used networks in order to better combine global context aggregation with local detail preservation, and utilize a simple loss computed on individual pixels. Designing more complex losses that account for the structure contained in semantic labelings has gotten substantially less attention. In order to provide efficient results that better reflect the regularity of genuine segmentations, we investigate such priors for semantic segmentation (see Chapters 2 and 3) and propose a new approach to the problem.

We thoroughly analyzed the proposed method in Chapter 4. Based on knowledge about the high regularity of real scenes, we proposed a method for improving class predictions by learning to selectively exploit information from coplanar pixels. The key idea is based on our prior which claims that for each pixel, there is a seed pixel which shares the same prediction with the former. As a result of this, we design a network with two heads. The first head generates pixel-level classes, whereas the second generates a dense offset vector field that identifies seed pixel positions. Seed pixels' class predictions are then utilized to predict classes at each point. To account for possible deviations from precise local planarity, the resultant prediction is adaptively fused with the initial prediction from the first head using a learnt confidence map. The entire architecture is implemented on HRNetV2, a state-of-the-art model on Cityscapes dataset. The superior performance of our offset vector - based model against previous SOTA methods was demonstrated both qualitatively and quantitatively through extensive experimental evaluation on both Cityscapes and ACDC datasets (see Chapter 5). Both datasets are challenging. The former is tasked for urban scene understanding, while the latter is used for training and testing semantic segmentation methods on adverse visual conditions.

All in all, our method enhances by far the initial model's output predictions, since it achieves not only better total results but also better per class results in the majority of classes under similar training time. It learns an implicit representation of different objects which benefits the overall semantic segmentation estimation capability of the network. In fact, not only our idea achieves highly satisfactory results close to GT images, but also outperforms the predictions outputted by the initial model's head. In particular, it classifies some false predicted pixels in the correct classes. Thus, it eliminates discontinuities and improves the shape as well as the form of the corresponding segments, leading to more realistic results. This is a strong contribution which opens new pathways for real world applications, such as Self-Driving Cars or Medical Imaging and Diagnostics.

## 2 Future and Follow Up Works

Although we have shown promising results in various datasets, there are still limitations in our approach, which may open new doors for future research. We briefly pinpoint some of them bellow.

1. **Application of this method on Visual Transformers.** At this point in time, newly designed Vision Transformers achieve state-of-the-art results in the task of semantic segmentation . We will apply our method on the [ViT-Adapter \[34\]](#), expecting new state-of-the-art results for this task.
2. **Combination of our method with other techniques.** We could study the combination of our method with other techniques for semantic segmentation and instance segmentation on a plethora of datasets. For example, it can be applied on the combination of the HRNet with the object-contextual representation (OCR) scheme [9].
3. **Ultimate erosion - based method.** As we saw in the offset vector illustrations (see Fig. 4.3 - 4.8) some vectors fail to point to the center of the corresponding segment. As a result their distribution on the specific segment is not normal. In order to deal with this problem, we could use ultimate erosion.

In particular, through this function, we could find for each [GT](#) image each segment's center and then use this set of coordinates during the training process. Thus, we will force the offset vectors of a specific object to point to his center, leading to better predictions.





# Bibliography

- [1] *Deep Convolutional Neural Networks*. Accessed: 2022-04-07.
- [2] *CS231n Convolutional Neural Networks for Visual Recognition*. Accessed: 2022-04-23.
- [3] Vernon, D. *Machine vision: Automated visual inspection and robot vision*. Prentice-Hall, Inc., 1991.
- [4] Taghanaki, S. A. et al. *Deep Semantic Segmentation of Natural and Medical Images: A Review*. 2019. DOI: [10.48550/ARXIV.1910.07655](https://doi.org/10.48550/ARXIV.1910.07655).
- [5] *An overview of semantic image segmentation*. Accessed: 2022-04-13.
- [6] Wang, J. et al. *Deep High-Resolution Representation Learning for Visual Recognition*. 2019. DOI: [10.48550/ARXIV.1908.07919](https://doi.org/10.48550/ARXIV.1908.07919).
- [7] Cordts, M. et al. *The Cityscapes Dataset for Semantic Urban Scene Understanding*. 2016. DOI: [10.48550/ARXIV.1604.01685](https://doi.org/10.48550/ARXIV.1604.01685).
- [8] Mottaghi, R. et al. “The role of context for object detection and semantic segmentation in the wild”. English. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Publisher Copyright: © 2014 IEEE.; 27th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014 ; Conference date: 23-06-2014 Through 28-06-2014. IEEE Computer Society, Sept. 2014, pp. 891–898. DOI: [10.1109/CVPR.2014.119](https://doi.org/10.1109/CVPR.2014.119).
- [9] Yuan, Y. et al. “Segmentation Transformer: Object-Contextual Representations for Semantic Segmentation”. In: (2019). DOI: [10.48550/ARXIV.1909.11065](https://doi.org/10.48550/ARXIV.1909.11065).
- [10] Patil, V. et al. *P3Depth: Monocular Depth Estimation with a Piecewise Planarity Prior*. 2022. DOI: [10.48550/ARXIV.2204.02091](https://doi.org/10.48550/ARXIV.2204.02091).
- [11] Sakaridis, C., Dai, D., and Van Gool, L. “ACDC: The Adverse Conditions Dataset with Correspondences for Semantic Driving Scene Understanding”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021.
- [12] Zhao, H. et al. *Pyramid Scene Parsing Network*. 2016. DOI: [10.48550/ARXIV.1612.01105](https://doi.org/10.48550/ARXIV.1612.01105).
- [13] Zhao, H. et al. “Psanet: Point-wise spatial attention network for scene parsing”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 267–283.
- [14] Li, H. et al. “Pyramid attention network for semantic segmentation”. In: *arXiv preprint arXiv:1805.10180* (2018).
- [15] Ke, T.-W. et al. “Adaptive affinity fields for semantic segmentation”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 587–602.
- [16] Fourure, D. et al. “Residual conv-deconv grid network for semantic segmentation”. In: *arXiv preprint arXiv:1707.07958* (2017).
- [17] Ghiasi, G. and Fowlkes, C. C. “Laplacian pyramid reconstruction and refinement for semantic segmentation”. In: *European conference on computer vision*. Springer. 2016, pp. 519–534.
- [18] Chen, L.-C. et al. *DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs*. 2016. DOI: [10.48550/ARXIV.1606.00915](https://doi.org/10.48550/ARXIV.1606.00915).
- [19] Li, X. et al. “Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 3193–3202.
- [20] Lin, G. et al. “Efficient piecewise training of deep structured models for semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3194–3203.
- [21] Pohlen, T. et al. “Full-resolution residual networks for semantic segmentation in street scenes”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4151–4160.

- [22] Lin, G. et al. “Refinenet: Multi-path refinement networks for high-resolution semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1925–1934.
- [23] Jin, X. et al. “Video scene parsing with predictive feature learning”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 5580–5588.
- [24] Liang, X., Zhou, H., and Xing, E. “Dynamic-structured semantic propagation network”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 752–761.
- [25] Peng, C. et al. “Large kernel matters—improve semantic segmentation by global convolutional network”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4353–4361.
- [26] Wojna, Z. et al. “The devil is in the decoder”. In: *British Machine Vision Conference 2017, BMVC 2017*. BMVA Press. 2017, pp. 1–13.
- [27] Kong, S. and Fowlkes, C. C. “Recurrent scene parsing with perspective understanding in the loop”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 956–965.
- [28] Wu, Z., Shen, C., and Van Den Hengel, A. “Wider or deeper: Revisiting the resnet model for visual recognition”. In: *Pattern Recognition* 90 (2019), pp. 119–133.
- [29] Yu, C. et al. “Bisenet: Bilateral segmentation network for real-time semantic segmentation”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 325–341.
- [30] Yu, C. et al. “Learning a discriminative feature network for semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 1857–1866.
- [31] Xu, D. et al. “Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 675–684.
- [32] Fu, J. et al. “Dual attention network for scene segmentation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 3146–3154.
- [33] Chen, L.-C. et al. “Encoder-decoder with atrous separable convolution for semantic image segmentation”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 801–818.
- [34] Chen, Z. et al. *Vision Transformer Adapter for Dense Predictions*. 2022. DOI: [10.48550/ARXIV.2205.08534](https://doi.org/10.48550/ARXIV.2205.08534).
- [35] Goodfellow, I., Bengio, Y., and Courville, A. *Deep learning*. MIT press, 2016.
- [36] Liu, G. et al. “Passenger flow estimation based on convolutional neural network in public transportation system”. In: *Knowledge-Based Systems* 123 (2017), pp. 102–115. ISSN: 0950-7051. DOI: <https://doi.org/10.1016/j.knsys.2017.02.016>.
- [37] *Introduction to Convolutional Neural Networks*. Accessed: 2022-05-07.
- [38] *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. Accessed: 2022-05-04.
- [39] *Autoencoder: Downsampling and Upsampling*. Accessed: 2022-05-10.
- [40] Noh, H., Hong, S., and Han, B. “Learning deconvolution network for semantic segmentation”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1520–1528.
- [41] *Batch normalization in 3 levels of understanding*. Accessed: 2022-05-09.
- [42] *Applied Deep Learning - Part 4: Convolutional Neural Networks*. Accessed: 2022-05-08.
- [43] Wikipedia contributors. *Voronoi diagram* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 24-June-2022]. 2022.
- [44] Wikipedia contributors. *Delaunay triangulation* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 24-June-2022]. 2022.
- [45] Hurtik, P. and Madrid, N. “Bilinear Interpolation over fuzzified images: Enlargement”. In: Aug. 2015. DOI: [10.1109/FUZZ-IEEE.2015.7338082](https://doi.org/10.1109/FUZZ-IEEE.2015.7338082).
- [46] Wikipedia contributors. *Bilinear interpolation* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 23-June-2022]. 2022.
- [47] Yi-de, M., Qing, L., and Zhi-bai, Q. “Automated image segmentation using improved PCNN model based on cross-entropy”. In: *Proceedings of 2004 International Symposium on Intelligent Multimedia, Video and Speech Processing, 2004*. 2004, pp. 743–746. DOI: [10.1109/ISIMP.2004.1434171](https://doi.org/10.1109/ISIMP.2004.1434171).
- [48] Jadon, S. “A survey of loss functions for semantic segmentation”. In: *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. IEEE, Oct. 2020. DOI: [10.1109/cibcb48159.2020.9277638](https://doi.org/10.1109/cibcb48159.2020.9277638).

- 
- [49] Pihur, V., Datta, S., and Datta, S. “Weighted rank aggregation of cluster validation measures: a Monte Carlo cross-entropy approach”. In: *Bioinformatics* 23.13 (May 2007), pp. 1607–1615. ISSN: 1367-4803. DOI: [10.1093/bioinformatics/btm158](https://doi.org/10.1093/bioinformatics/btm158). eprint:
- [50] Ho, Y. and Wookey, S. “The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling”. In: *IEEE Access* 8 (2020), pp. 4806–4813. DOI: [10.1109/ACCESS.2019.2962617](https://doi.org/10.1109/ACCESS.2019.2962617).
- [51] Pan, S. et al. “Diagnostic Model of Coronary Microvascular Disease Combined With Full Convolution Deep Network With Balanced Cross-Entropy Cost Function”. In: *IEEE Access* 7 (2019), pp. 177997–178006. DOI: [10.1109/ACCESS.2019.2958825](https://doi.org/10.1109/ACCESS.2019.2958825).
- [52] Shrivastava, A., Gupta, A., and Girshick, R. *Training Region-based Object Detectors with Online Hard Example Mining*. 2016. DOI: [10.48550/ARXIV.1604.03540](https://doi.org/10.48550/ARXIV.1604.03540).
- [53] *Loss Functions in Machine Learning and LTR*. Accessed: 2022-05-11.
- [54] *What are Loss Functions?* Accessed: 2022-05-11.
- [55] MacKay, D. J., Mac Kay, D. J., et al. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.
- [56] *Understanding the 3 most common loss functions for Machine Learning Regression*. Accessed: 2022-04-11.
- [57] Wang, P. et al. *Joint Object and Part Segmentation using Deep Learned Potentials*. 2015. DOI: [10.48550/ARXIV.1505.00276](https://doi.org/10.48550/ARXIV.1505.00276).
- [58] Ulku, I. and Akagündüz, E. “A Survey on Deep Learning-based Architectures for Semantic Segmentation on 2D Images”. In: *Applied Artificial Intelligence* (Feb. 2022), pp. 1–45. DOI: [10.1080/08839514.2022.2032924](https://doi.org/10.1080/08839514.2022.2032924).
- [59] Otsu, N. “A Threshold Selection Method from Gray-Level Histograms”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 9.1 (1979), pp. 62–66. DOI: [10.1109/TSMC.1979.4310076](https://doi.org/10.1109/TSMC.1979.4310076).
- [60] Soukup, L. et al. “Sémantická segmentace obrazu pomocí hlubokých neuronových sítí”. In: (2020).
- [61] Cao, L. et al. “Otsu multilevel thresholding segmentation based on quantum particle swarm optimization algorithm”. In: *Int. J. Wirel. Mob. Comput.* 10 (2016), pp. 272–277.
- [62] Triggs, B. and Verbeek, J. “Scene segmentation with crfs learned from partially labeled images”. In: *Advances in neural information processing systems* 20 (2007).
- [63] Badrinarayanan, V., Kendall, A., and Cipolla, R. “Segnet: A deep convolutional encoder-decoder architecture for image segmentation”. In: *IEEE transactions on pattern analysis and machine intelligence* 39.12 (2017), pp. 2481–2495.
- [64] Ronneberger, O., Fischer, P., and Brox, T. “U-net: Convolutional networks for biomedical image segmentation”. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.
- [65] Chen, L.-C. et al. “Semantic image segmentation with deep convolutional nets and fully connected crfs”. In: *arXiv preprint arXiv:1412.7062* (2014).
- [66] Chen, L.-C. et al. “Rethinking atrous convolution for semantic image segmentation”. In: *arXiv preprint arXiv:1706.05587* (2017).
- [67] Peng, X. et al. “A recurrent encoder-decoder for sequential face alignment”. In: *Proc. European Conference on Computer Vision*, pp. 38–56.
- [68] Simonyan, K. and Zisserman, A. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. DOI: [10.48550/ARXIV.1409.1556](https://doi.org/10.48550/ARXIV.1409.1556).
- [69] He, K. et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.
- [70] Shelhamer, E., Long, J., and Darrell, T. “Fully Convolutional Networks for Semantic Segmentation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.4 (2017), pp. 640–651. DOI: [10.1109/TPAMI.2016.2572683](https://doi.org/10.1109/TPAMI.2016.2572683).
- [71] Krizhevsky, A., Sutskever, I., and Hinton, G. E. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012).
- [72] Szegedy, C. et al. “Going deeper with convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.
- [73] Russakovsky, O. et al. “Imagenet large scale visual recognition challenge”. In: *International journal of computer vision* 115.3 (2015), pp. 211–252.
- [74] Everingham, M. et al. “The pascal visual object classes (voc) challenge”. In: *International journal of computer vision* 88.2 (2010), pp. 303–338.
-

- [75] Silberman, N. et al. “Indoor segmentation and support inference from rgbd images”. In: *European conference on computer vision*. Springer. 2012, pp. 746–760.
- [76] Liu, C., Yuen, J., and Torralba, A. “Sift flow: Dense correspondence across scenes and its applications”. In: *IEEE transactions on pattern analysis and machine intelligence* 33.5 (2010), pp. 978–994.
- [77] Sultana, F., Sufian, A., and Dutta, P. “Evolution of Image Segmentation using Deep Convolutional Neural Network: A Survey”. In: *Knowledge-Based Systems* 201-202 (Aug. 2020), p. 106062. DOI: [10.1016/j.knosys.2020.106062](https://doi.org/10.1016/j.knosys.2020.106062).
- [78] Yu, F. and Koltun, V. “Multi-scale context aggregation by dilated convolutions”. In: *arXiv preprint arXiv:1511.07122* (2015).
- [79] LeCun, Y. et al. “Backpropagation applied to handwritten zip code recognition”. In: *Neural computation* 1.4 (1989), pp. 541–551.
- [80] Chen, L.-C. et al. *Rethinking Atrous Convolution for Semantic Image Segmentation*. 2017. DOI: [10.48550/ARXIV.1706.05587](https://doi.org/10.48550/ARXIV.1706.05587).
- [81] Noh, H., Hong, S., and Han, B. *Learning Deconvolution Network for Semantic Segmentation*. 2015. DOI: [10.48550/ARXIV.1505.04366](https://doi.org/10.48550/ARXIV.1505.04366).
- [82] Zeiler, M. D. and Fergus, R. “Visualizing and understanding convolutional networks”. In: *European conference on computer vision*. Springer. 2014, pp. 818–833.
- [83] Zeiler, M. D., Taylor, G. W., and Fergus, R. “Adaptive deconvolutional networks for mid and high level feature learning”. In: *2011 international conference on computer vision*. IEEE. 2011, pp. 2018–2025.
- [84] Ronneberger, O., Fischer, P., and Brox, T. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Ed. by N. Navab et al. Cham: Springer International Publishing, 2015, pp. 234–241. ISBN: 978-3-319-24574-4.
- [85] Badrinarayanan, V., Kendall, A., and Cipolla, R. “SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.12 (2017), pp. 2481–2495. DOI: [10.1109/TPAMI.2016.2644615](https://doi.org/10.1109/TPAMI.2016.2644615).
- [86] He, K. et al. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: [10.1109/CVPR.2016.90](https://doi.org/10.1109/CVPR.2016.90).
- [87] Huang, G. et al. *Densely Connected Convolutional Networks*. 2016. DOI: [10.48550/ARXIV.1608.06993](https://doi.org/10.48550/ARXIV.1608.06993).
- [88] He, K. et al. *Deep Residual Learning for Image Recognition*. 2015. DOI: [10.48550/ARXIV.1512.03385](https://doi.org/10.48550/ARXIV.1512.03385).
- [89] Jégou, S. et al. *The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation*. 2016. DOI: [10.48550/ARXIV.1611.09326](https://doi.org/10.48550/ARXIV.1611.09326).
- [90] Liu, W. et al. “SSD: Single Shot MultiBox Detector”. In: *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 21–37. DOI: [10.1007/978-3-319-46448-0\\_2](https://doi.org/10.1007/978-3-319-46448-0_2).
- [91] Dai, J. et al. *R-FCN: Object Detection via Region-based Fully Convolutional Networks*. 2016. DOI: [10.48550/ARXIV.1605.06409](https://doi.org/10.48550/ARXIV.1605.06409).
- [92] Dai, J. et al. *Instance-sensitive Fully Convolutional Networks*. 2016. DOI: [10.48550/ARXIV.1603.08678](https://doi.org/10.48550/ARXIV.1603.08678). URL:
- [93] Sevilla-Lara, L. et al. *Optical Flow with Semantic Segmentation and Localized Layers*. 2016. DOI: [10.48550/ARXIV.1603.03911](https://doi.org/10.48550/ARXIV.1603.03911).
- [94] Takikawa, T. et al. *Gated-SCNN: Gated Shape CNNs for Semantic Segmentation*. 2019. DOI: [10.48550/ARXIV.1907.05740](https://doi.org/10.48550/ARXIV.1907.05740).
- [95] Oršić, M. and Šegvić, S. “Efficient semantic segmentation with pyramidal fusion”. In: *Pattern Recognition* 110 (2021), p. 107611. ISSN: 0031-3203. DOI: <https://doi.org/10.1016/j.patcog.2020.107611>.
- [96] Dosovitskiy, A. et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2020. DOI: [10.48550/ARXIV.2010.11929](https://doi.org/10.48550/ARXIV.2010.11929).
- [97] Yan, H., Zhang, C., and Wu, M. *Lawin Transformer: Improving Semantic Segmentation Transformer with Multi-Scale Representations via Large Window Attention*. 2022. DOI: [10.48550/ARXIV.2201.01615](https://doi.org/10.48550/ARXIV.2201.01615).
- [98] Xie, E. et al. *SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers*. 2021. DOI: [10.48550/ARXIV.2105.15203](https://doi.org/10.48550/ARXIV.2105.15203).
- [99] Zhou, B. et al. “Scene Parsing Through ADE20K Dataset”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

- [100] Everingham, M. et al. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*. <http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html>.
- [101] Brostow, G. J. et al. "Segmentation and Recognition Using Structure from Motion Point Clouds". In: *Computer Vision – ECCV 2008*. Ed. by D. Forsyth, P. Torr, and A. Zisserman. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 44–57. ISBN: 978-3-540-88682-2.
- [102] Song, S., Lichtenberg, S. P., and Xiao, J. "SUN RGB-D: A RGB-D Scene Understanding Benchmark Suite". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015.
- [103] Hussain Raza, S., Grundmann, M., and Essa, I. "Geometric Context from Videos". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2013.
- [104] Ilg, E. et al. "FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks". In: (Dec. 2016).
- [105] Bob Rudis, Noam Ross and Simon Garnier. *Introduction to the viridis color maps*. [Online; accessed 30-August-2022].