Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
τομεας σηματων, ελεγχου και ρομποτικης

Multiple Resolutions in Semantic Image Segmentation

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

Σμαραγδής Μπενέτου

Επιβλέπων: Πέτρος Μαραγκός
Καθηγητής Ε.Μ.Π.

Συνεπιβλέπων: Χρήστος Σακαρίδης
Μεταδιδακτορικός ΕΤΗ

ΕΡΓΑΣΤΗΡΙΟ ΟΡΑΣΗΣ ΥΠΟΛΟΓΙΣΤΩΝ, ΕΠΙΚΟΙΝΩΝΙΑΣ ΛΟΓΟΥ ΚΑΙ ΕΠΕΞΕΡΓΑΣΙΑΣ ΣΗΜΑΤΩΝ

Αθήνα, Οκτώβριος 2023

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

Multiple Resolutions in Semantic Image Segmentation

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

Σμαραγδής Μπενέτου

Επιβλέπων: Πέτρος Μαραγκός
Καθηγητής Ε.Μ.Π.

Συνεπιβλέπων: Χρήστος Σακαρίδης
Μεταδιδακτορικός ΕΤΗ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 20η Οκτωβρίου, 2023.

.....	.....	.....
Πέτρος Μαραγκός	Αλέξανδρος Ποταμιάνος	Αθανάσιος Ροντογιάννης
Καθηγητής Ε.Μ.Π.	Αναπληρωτής Καθηγητής Ε.Μ.Π.	Αναπληρωτής Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2023

.....

Μπενετου Σμαραγδη
Διπλωματούχος Ηλεκτρολόγος Μηχανικός
και Μηχανικός Υπολογιστών Ε.Μ.Π.

στην οικογένειά μου

Περίληψη

Η Όραση Υπολογιστών είναι ένας τομέας στην επιστήμη των υπολογιστών που στοχεύει στην ενίσχυση της οπτικής αντίληψης των υπολογιστών. Με πολλές εφαρμογές σε διαφορετικούς τομείς όπως ο αυτόματος πιλότος, η ιατρική απεικόνιση, η ασφάλεια, η γεωργία κ.λπ., η πρόοδος της Όρασης Υπολογιστών βρίσκεται στο επίκεντρο της προσοχής. Οι εργασίες ή οι στόχοι της γίνονται συνεχώς πιο απαιτητικές. Αρχικά, η εργασία της "κατηγοριοποίησης" ταξινομεί μια εικόνα σε μια κλάση ενώ η "ανίχνευση αντικειμένων" προσδιορίζει όλα τα αντικείμενα σε μια εικόνα. Στη συνέχεια, εισάγεται η αυξανόμενης περιπλοκότητας εργασία της "σημασιολογικής κατάτμησης" που ζητά την κατηγοριοποίηση κάθε pixel σε μια εικόνα. Η σημασιολογική κατάτμηση είναι ζωτικής σημασίας σε εφαρμογές του πραγματικού κόσμου, καθώς επιτρέπει την πλήρη αντίληψη του περιβάλλοντος.

Οι πιο πρώιμες εργασίες προσεγγίζονται ικανοποιητικά με συνελικτικά μοντέλα. Με την εισαγωγή των μετασχηματιστών, η ανίχνευση αντικειμένων βελτιώθηκε περαιτέρω καθώς είναι πιο αποτελεσματικοί στην ανίχνευση αντικειμένων πολλαπλής κλίμακας. Η μονάδα "αυτο-προσοχής" των μετασχηματιστών κατάφερε να ικανοποιήσει αυτήν την απαίτηση ανιχνευσης πολλαπλών κλιμάκων και να εισάγει πληροφορίες συμφραζόμενων που δεν ήταν σε θέση να κάνουν τα συνελικτικά στοχεία. Η σημασιολογική κατάτμηση είναι μια ακόμη πιο περίπλοκη διαδικασία καθώς απαιτεί αναγνώριση του σχήματος ενός αντικειμένου σε πολλαπλές κλίμακες. Οι μετασχηματιστές ήταν σε θέση να εκτελέσουν αυτήν την εργασία, ωστόσο, η φιλοσοφία της αρχιτεκτονικής έπρεπε να αλλάξει προκειμένου να αυξηθεί η απόδοση σε αυτήν την πιο απαιτητική εργασία. Οι αρχιτεκτονικές κωδικοποιητή-αποκωδικοποιητή λειτουργούσαν αποτελεσματικά στην ταξινόμηση καθώς μετατρέπουν τις πληροφορίες εισόδου σε μια χαμηλότερη διάσταση εξάγοντας έτσι την μοναδική κλάση. Μεταγενέστερες προσεγγίσεις προσπάθησαν να εισάγουν πολλαπλές αναλύσεις χρησιμοποιώντας υπολειμματικές συνδέσεις από τον κωδικοποιητή στον αποκωδικοποιητή προκειμένου να αποτραπεί αυτή η απώλεια πληροφοριών. Αυτή η τεχνική, ωστόσο, εξακολουθεί να αντιμετωπίζει το πρόβλημα της επεξεργασίας πληροφοριών χωρίς απώλεια και εκεί είναι όπου οι πολλαπλές αναλύσεις εισάγουν μια λύση στο πρόβλημα. Μετά από εκτεταμένη έρευνα, οι πολλαπλές αναλύσεις παρατηρούμε ότι κυριαρχούν στα SOTA και βελτιώνουν τα αντίστοιχα μοντέλα μεμονωμένης ανάλυσης. Θεωρητικά, οι πολλαπλές αναλύσεις μπορούν μόνο να βελτιώσουν ένα μοντέλο, καθώς εισάγουν επιπλέον πληροφορίες από τις πληροφορίες που παράγονται στο αρχικό μοντέλο.

Το Mask2Former είναι ένα μοντέλο κατάτμησης πολλαπλών χρήσεων που μπορεί να εκπαιδευτεί χωρίς αλλαγή αρχιτεκτονικής σε: σημασιολογική κατάτμηση, κατάτμηση αντικειμένων και πανοπτική κατάτμηση. Αποτελείται από μονάδα επιπεδου pixel, έναν αποκωδικοποιητή μετασχηματιστή και μια κεφαλή κατάτμησης. Η μονάδα επιπέδου pixel μπορεί να αντικατασταθεί από οποιοδήποτε μοντέλο ταξινόμησης pixel, ωστόσο, μέχρι τώρα έχουν χρησιμοποιηθεί μόνο αρχιτεκτονικές μορφής κωδικοποιητή-αποκωδικοποιητή. Έτσι, σε αυτή τη διπλωματική εργασία ο στόχος είναι να χρησιμοποιηθούν αρχιτεκτονικές πολλαπλών αναλύσεων στην μονάδα επιπεδου pixel του Mask2Former με προοπτικές βελτίωσης της απόδοσής του στη σημασιολογική κατάτμηση. Ύστερα από πειραματισμούς επιτεύχθηκε η βελτίωση της απόδοσης της αρχικής αρχιτεκτονικής κατά 0.3mIoU στο Cityscapes και κατά 0.2mIoU στο ADE20k σύνολο δεδομένων.

Λέξεις Κλειδιά — Όραση υπολογιστών, Σημασιολογική κατάτμηση, Μηχανική μάθηση, Νευρωνικά δίκτυα, Μετασχηματιστές, Πολλαπλές αναλύσεις, Κωδικοποιητής-Αποκωδικοποιητής, Cityscapes, Mask2Former, HRNET

Abstract

Computer vision is a field in computer science aiming to enhance visual perception of computers. With numerous applications in different areas such as autopilot, medical imaging, security, agriculture etc., computer vision advancement is at the center of attention. Its tasks or goals are constantly getting more demanding. It started from classification which classifies an image. Then object detection was tackled which identifies all the objects in an image. Finally, semantic segmentation was introduced which requests for classification of every pixel in an image. Semantic segmentation is crucial in real-world applications as it would allow for complete environment perception.

The previous tasks were satisfactorily approached with convolutional models. Moving on to transformers, object detection was further improved as it is more effective on detecting multi-scale objects. The self-attention module of transformers was able to implement that requirement and introduce contextual information that convolutions were not able to. However, unlike classification or even object detection, semantic segmentation requires multiple scale recognition of objects' shapes. Transformers were able to perform this task, however, the architecture philosophy needed to be changed in order to scale up performance in a more demanding task. Encoder-decoder architectures are remains from the classification task as they transform information to a lower dimension producing the single class label. Later approaches attempted to introduce multiple resolutions by using residual connections from encoder to decoder in order to prevent this loss of information. This technique, though, still faces the problem of processing information without loss and that is where multiple resolutions introduce a solution to the problem. After the extended background research, multiple resolutions are dominating SOTA and improve their respective single resolution models. Theoretically, multiple resolutions can only improve a model as they introduce extra information than the information produced in the original model.

Mask2Former is a multi-purpose segmentation model that can be trained without changing architecture in : semantic segmentation, instance segmentation, and panoptic segmentation. It is composed of a pixel-level module, a transformer decoder, and a segmentation head. The pixel-level module in this model can be any feature extraction model, however, up until now only encoder-decoder architectures have been used. Thus, in this diploma research the goal is to introduce high resolution to the Mask2Former pixel-level module in prospects of improving its performance in semantic segmentation. We achieved through the multi-resolutional architecture an improvement of 0.3mIoU to the original model's performance in Cityscapes and a 0.2mIoU improvement in ADE20k.

Keywords — Computer vision, Semantic Segmentation , Machine Learning, Neural Networks, Transformers, Multiple Resolutions, Encoder-Decoder, Cityscapes, Mask2Former, HRNET

Ευχαριστίες

Αυτή διπλωματική μου προσέφερε την ευκαιρία να μελετήσω βαθύτερα διάφορα θέματα από τον τομέα της Μηχανικής μάθησης και να λάβω μια γεύση από την ερευνητική διαδικάσία. Ευχαριστώ θερμά τον συνεπιβλέποντα μου κ.Χρήστο Σακαρίδη, μεταδιδακτορικό του ΕΤΗ, που με καθοδήγησε με μεγάλο ενδιαφέρον και πίστη για την πορεία μου. Η βοήθειά του ήταν πολύτιμη στην εκπόνηση αυτής της διπλωματικής εργασίας. Επίσης, θα επιθυμούσα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ.Πέτρο Μαραγκό, καθηγητή Ε.Μ.Π., που κατά τη διάρκεια της φοίτησης μου, μου καλλιέργησε το ενδιαφέρον στην Όραση Υπολογιστών και μου έδωσε την ευκαιρία να εκπονήσω αυτήν την διπλωματική υπό την επίβλεψή του.

Σημαντικότεροι στην πορεία μου σε αυτό το πενταετές ταξίδι αλλά και στην υπόλοιποι ζωή μου είναι η οικογένειά μου. Τους ευχαριστώ για την στήριξή τους σε κάθε μου βήμα και τις υπέροχες στιγμές. Ευχαριστώ και τους φίλους μου για τα αξέχαστα φοιτητικά χρόνια.

Μπενέτου Σμαραγδή
Οκτώβριος 2023

Contents

List of Figures

List of Tables

Εκτεταμένη περίληψη στα Ελληνικά

Εισαγωγή

Κατηγορίες Κατάτμησης Εικόνων

Η κατάτμηση εικόνας σύμφωνα με το [1] αναφέρεται στον διαχωρισμό μιας εικόνας σε ένα σύνολο μη-επικαλυπτόμενων περιοχών, η ένωση των οποίων αποτελεί τη συνολική εικόνα. Αυτές οι περιοχές πρέπει να:

- Είναι ομοιογενείς όσον αφορά κάποια χαρακτηριστικά
- Είναι απλές χωρίς καμιά οπή
- Έχουν σημαντικά διαφορετικές τιμές με τις γειτονικές περιοχές ως προς το χαρακτηριστικό που είναι ομοιογενείς.
- Να έχουν ομαλά όρια και να μην έχουν πολλά άνοιγματα

Αρχικά, αυτοί οι στόχοι επιδιώχθηκαν να επιτευχθούν με διάφορες τεχνικές επεξεργασίας εικόνας. Μία από αυτές είναι η συσταδοποίηση εικόνας [2] [3] [4] που χρησιμοποιείται με ανίχνευση ακμών και περιγραφών και περαιτέρω εξελίσσεται με τη μοντελοποίηση χρησιμοποιώντας τη διαδικασία Markov [5]. Άλλες προσεγγίσεις χρησιμοποιούν ιστογράμματα, όπως η εξαγωγή χαρακτηριστικών HOG [6] και SIFT [7], τα οποία είναι ιστογράμματα προσανατολισμού. Ωστόσο, μετά την εισαγωγή των συνελικτικών νευρωνικών δικτύων, η σημασιολογική κατάτμηση εικόνας πήρε στροφή προς την επιβλεπόμενη μάθηση. Τα σύνολα δεδομένων έχουν αυξηθεί σημαντικά - Cityscapes [8], ImageNet [9], COCO [10] - και οι απαιτήσεις γενίκευσης από τις προβλέψεις του μοντέλου καθιστούν λογικό να υιοθετηθεί μια πιο προσαρμόσιμη και πολύπλοκη προσέγγιση. Τα νευρωνικά δίκτυα είναι ικανά να μάθουν μοτίβα και τα βαθιά νευρωνικά δίκτυα έχουν πολλές παραμέτρους που μπορούν να μοντελοποιήσουν πολύπλοκες συναρτήσεις. Η κατάτμηση εικόνας είναι πιο απαιτητική από την ταξινόμηση εικόνας, καθώς πρέπει να ανιχνεύσει σχέσεις μεταξύ των pixel σε διάφορες κλίμακες, γι 'αυτό χρειάζεται πιο σύνθετες δομές που λαμβάνουν υπόψη τόσο τα σημασιολογικά όσο και τα χωρικά χαρακτηριστικά.

Σημασιολογική Κατάτμηση Εικόνας

Ο στόχος αυτής της κατηγορίας, που είναι και ο κύριος στόχος αυτής της εργασίας, είναι να διαχωρίσει την εικόνα εισόδου σύμφωνα με τις σημασιολογικές πληροφορίες και να προβλέψει τη σημασιολογική κατηγορία κάθε εικονοστοιχείου από ένα συγκεκριμένο σύνολο ετικετών. Δεν διακρίνει ανάμεσα σε αντικείμενα της ίδιας κατηγορίας αλλά τα ομαδοποιεί. Σύμφωνα με τον Adelson [11], η σημασιολογική κατάτμηση σχεδιάστηκε για να αναγνωρίζει stuff που είναι άμορφες περιοχές με παρόμοια υφή ή υλικό. Η σημασιολογική κατάτμηση εικόνας έχει πολλές εφαρμογές σε καθημερινά προβλήματα. Χρησιμοποιείται ευρέως στην ιατρική, όπως η ανίχνευση εγκεφάλου και όγκων [12] και στην παρακολούθηση ιατρικών συσκευών στη χειρουργική [13]. Άλλες εφαρμογές περιλαμβάνουν την αυτόνομη οδήγηση [14], όπου ένα αυτοκίνητο μπορεί να πλοηγηθεί στο περιβάλλον του χωρίς ανθρώπινη παρέμβαση. Η σημασιολογική κατάτμηση είναι κρίσιμη στο πλαίσιο εφαρμογών αυτόνομης οδήγησης, όπου η δεν έχει μόνο ως στόχο να ταξινομήσει το περιεχόμενο στην εικόνα, αλλά και να επισημάνει τη θέση και το περίγραμμα των αντικειμένων στην πραγματική σκηνή.

Ορισμένα σύνολα δεδομένων για αυτήν την κατηγορία είναι τα Cityscapes [8], PASCAL VOC [15] και ADE20K [16].

Instance segmentation

Το Instance Segmentation είναι ένα πρόβλημα της όρασης υπολογιστών που εμπλέκει την αναγνώριση και τον διαχωρισμό μεμονωμένων αντικειμένων(things) σε μια εικόνα, συμπεριλαμβανομένου του εντοπισμού των ορίων κάθε αντικειμένου και της ανάθεσης μοναδικής ετικέτας σε κάθε αντικείμενο. Ο στόχος του instance segmentation είναι να παράγει έναν χάρτη ανά pixel σηματοδότησης της εικόνας, όπου κάθε pixel αντιστοιχίζεται σε ένα συγκεκριμένο αντικείμενο μιας κλάσης. Συνεπώς, είναι πολύ παρόμοια με τη σημασιολογική κατάτμηση, ωστόσο σε αυτή την εργασία το κάθε αντικείμενο ανιχνεύεται ξεχωριστά.

Ορισμένα datasets για αυτήν την εργασία είναι τα Cityscapes [8], COCO [10], και ADE20K [16].

Πανοπτική κατάτμηση

Η πανοπτική σηματοδότηση [17] ενοποιεί τις δύο εργασίες - σημασιολογική κατάτμηση και instance segmentation - ανιχνεύοντας τα things και τα stuff. Έτσι, το αποτέλεσμα κάθε pixel i είναι μια ετικέτα κατηγορίας(l_i) και ένα αναγνωριστικό αντικειμένου (z_i) - $(l_i, z_i) \in L \times \mathbb{N}$, όπου $L = L^{th} \cup L^{st}$ και $L^{th} \cap L^{st} = \emptyset$. Όταν ένα pixel επισημαίνεται με $l_i \in L^{st}$, τότε το αντίστοιχο αναγνωριστικό αντικειμένου αγνοείται. Ορισμένα pixel μπορεί να έχουν μια ειδική ετικέτα "κενό" (void label).

Ορισμένα παραδείγματα σηματοδότησης πανοπτικής για αυτήν την εργασία περιλαμβάνουν τα Cityscapes [8], COCO [10], Mapillary Vistas [18], και ADE20K [16].

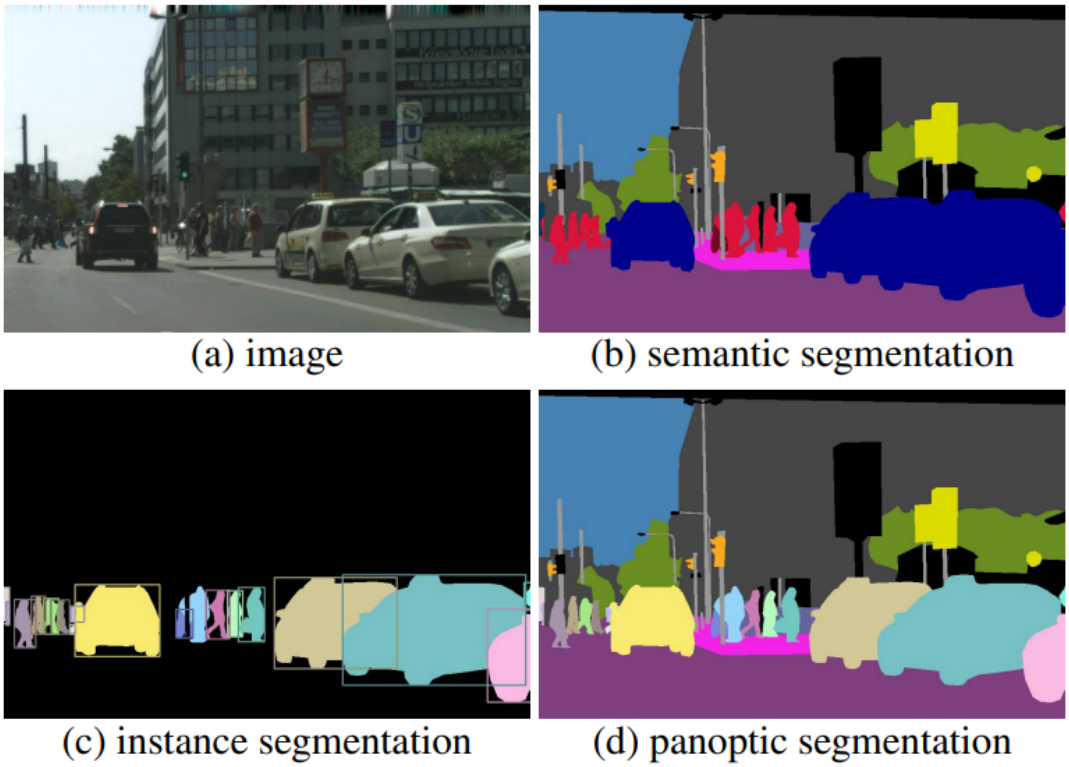


Figure 0.0.1: Κατηγορίες κατάτμησης εικόνας: α) σημασιολογική κατάτμηση: τα stuff δεν χωρίζονται σε αντικείμενα, β) instance segmentation: ανιχνεύονται μόνο τα αντικείμενα(things) ενώ τα stuff όπως ο ουρανός, ο δρόμος, κ.λπ. παραβλέπονται, γ) πανοπτική κατάτμηση: ανιχνεύονται τα stuff και τα things, με τα stuff να παραμένουν αναπόσπαστα.

Σύνολα Δεδομένων Σημασιολογικής Κατάτμησης

Σύνολο Δεδομένων Cityscapes

Το σύνολο δεδομένων Cityscapes [8] είναι ένα σύνολο δεδομένων που δημιουργήθηκε ειδικά για εφαρμογές αυτόνομης οδήγησης σε αστικά περιβάλλοντα. Περιλαμβάνει εικόνες τοπίων δρόμου από 50 διαφορετικά κέντρα πόλεων κατά τη διάρκεια όλων των 4 εποχών. Αυτές οι εικόνες είναι διαθέσιμες σε χαμηλή ανάλυση (8 bit) και υψηλή ανάλυση (16 bit). Το σύνολο δεδομένων περιλαμβάνει τόσο αναλυτικές επιπέδου pixel όσο και αναλυτικές επιπέδου περίπτωσης (instance-level) επισημάνσεις που παρέχονται σε δύο ομάδες - χονδροειδείς και λεπτομερείς επισημάνσεις. Οι λεπτομερείς επισημάνσεις παρέχονται σε 5000 εικόνες από 27 πόλεις, όπου όλα τα pixels έχουν επισημανθεί με τη δημιουργία πολυγώνων. Οι υπόλοιπες εικόνες προέρχονται από 23 πόλεις και χρησιμοποιούνται για τις χονδροειδείς επισημάνσεις, όπου τα πολύγωνα επιλέγονται πολύ πιο γρήγορα, μειώνοντας την ακρίβεια της επισήμανσης. Συνολικά περιλαμβάνονται 30 κατηγορίες, και 19 από αυτές χρησιμοποιούνται στην αξιολόγηση, όπως φαίνεται στην εικόνα 0.0.2.

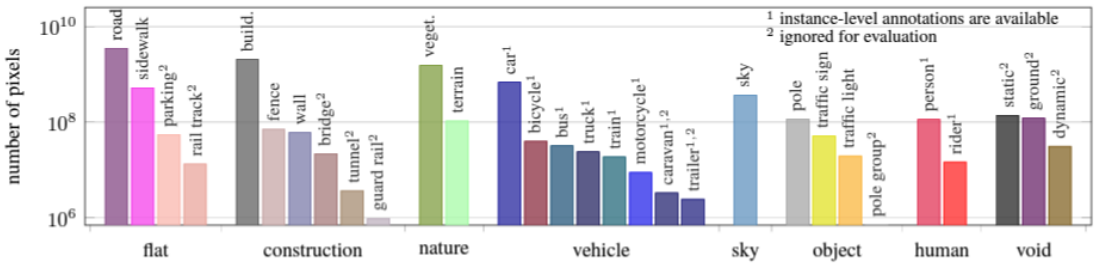


Figure 0.0.2: Κατηγορίες του Cityscapes, εικόνα από το [19]

	Cityscapes
Model	*mIoU*
InternImage-H [20]	87
HRNetV2-OCR_PSA [21]	86.93
InternImage-XL [20]	86.4
HRNet-OCR [22]	86.3
Vit-Adapter-L(Mask2Former, BEiT pretrain, Mapillary) [23]	85.8
OneFormer(ConvNetXt-XL, Mapillary, multiscale) [24]	85.8
SeMask(SeMask Swin-L Mask2Former) [25]	84.98
Sequential Ensemble (MiT-B5 + HRNet) [26]	84.8
OneFormer(ConvNetXt-XL, multi-scale) [24]	84.6
DiNAT-L(Mask2Former) [27]	84.5

Table 1: Πίνακας κατάταξης SOTA σημασιολογικής κατάτμησης στο cityscapes val [28]

Σύνολο Δεδομένων ADE20K

Το σύνολο δεδομένων ADE20K δημιουργήθηκε μετά την αναγνώριση της ανάγκης για ένα γενικό σύνολο δεδομένων που καλύπτει μια ποικιλία σκηνών και κοινών αντικειμένων. Τα σύνολα δεδομένων πριν από το ADE20K είχαν ένα περιορισμένο σύνολο σκηνών, όπως το Cityscapes [8], ή κάλυπταν λίγα ή ασήμαντα αντικείμενα, όπως το COCO[10] και το Pascal [15]. Στο ADE20K υπάρχουν 20,210 εικόνες στο σύνολο εκπαίδευσης, 2,000 εικόνες στο σύνολο επικύρωσης και 3,000 εικόνες στο σύνολο ελέγχου. Όλες οι εικόνες έχουν αναλυτικές επισημάνσεις αντικειμένων. Πολλά αντικείμενα έχουν επίσης αναλυτικές επισημάνσεις των μερών τους, όπως φαίνεται στην εικόνα 0.0.3. Για κάθε αντικείμενο υπάρχει επιπρόσθετη πληροφορία σχετικά με το αν είναι εμποδισμένο ή κομμένο και άλλα χαρακτηριστικά. Οι εικόνες στο σύνολο επικύρωσης έχουν αναλυτικές επισημάνσεις των μερών, ενώ οι επισημάνσεις των μερών δεν είναι αναλυτικές στις εικόνες του συνόλου εκπαίδευσης. Επιπλέον, τα μέρη μπορεί να έχουν υπομέρη που επίσεις επισημάνονται.

Figure 0.0.3: Επισημάνσεις στο ADE20K. Η δεύτερη σειρά έχει επισημάνσεις αντικειμένων και η τρίτη σειρά έχει επισημάνσεις μερών αντικειμένων.

Model	ADE20k *mIoU*
BEiT-3 [29]	62.8
EVA [30]	61.5
FD-SwinV2-G [31]	61.4
MaskDINO-SwinL[32]	60.8
OneFormer [24]	60.8
ViT-Adapter-L [23]	60.5
OneFormer [24]	58.6
ViT-Adapter-L [23]	58.4
OneFormer [24]	58.4
RSSeg-ViT-L [33]	58.4

Table 2: Πίνακας κατάταξης SOTA σημασιολογικής κατάτμησης στο ADE20k val [34]

Μετρικές σημασιολογικής κατάτμησης

Όπως αναφέρεται στο [35], η σημασιολογική κατάτμηση είναι μια πολύπλοκη διαδικασία που λαμβάνει υπόψη τις σχέσεις μεταξύ των ταξινομημένων pixel. Παρακάτω η αναφορά σε κλάσεις συμπεριλαμβάνει και τις κατηγορίες. Η ακρίβεια σε επίπεδο pixel(pACC) είναι μία αρχική μετρική για ποσοτικοποίηση της επίδοσης:

$$\text{acc} = \frac{\sum_{i=1}^{k} n_{ii}}{\sum_{i=1}^{k} t_i} \tag{0.0.1}$$

Οπού n_{ij} είναι ο αριθμός των pixel που ανήκουν στην κλάση i και επισημάνθηκαν ως κλάση j, k είναι ο συνολικός αριθμός κλάσεων, και $t_i = \sum_{j=1}^{k} n_{ij}$ είναι ο συνολικός αριθμός pixel της κλάσης i.

Ωστόσο, αυτή η μετρική είναι παραπλανητική καθώς επιτρέπει υψηλές αρχικές τιμές ακρίβειας σε σύνολα δεδομένων όπου μεγάλες περιοχές έχουν μια κλάση. Σε αυτές τις περιπτώσεις, το μοντέλο έχει μάθει μόνο τις

συχνές εμφανίσεις κλάσεων σε συγκεκριμένες θέσεις της εικόνας. Αυτό το πρόβλημα μπορεί να λυθεί με τις ακόλουθες μετρικές:

mACC Η μέση ακρίβεια είναι η μέση τιμή της ακρίβειας ανά κλάση: $\frac{1}{k}\sum_{i=1}^{k}\frac{n_{ii}}{t_i}$

IoU Ο λόγος της τομής προς την ένωση (IoU) [36] είναι μια αξιολόγηση ανά κλάση στην επικάλυψη της προβλεπόμενης σημασιολογικής επισήμανσης και της αληθινής επισήμανσης, διαιρεμένη με την ένωση (γνωστή και ως μετρική "intersection over union") εξαιρουμένων των pixel που υποσημειώνονται ως "κενά":

$$\text{IoU} = \frac{\text{true pos}}{\text{true pos} + \text{false pos} + \text{false neg}} \tag{0.0.2}$$

Όπου $truepos$, $falsepos$ και $falseneg$ είναι τα πλήθη των πραγματικά θετικών, ψευδών θετικών και ψευδών αρνητικών pixel για μια συγκεκριμένη κλάση αντίστοιχα.

mIoU Η μετρική αυτή είναι ο μέσος όρος ως προς όλες τις κλάσεις του λόγου της τομής προς την ένωση ανά κλάση(mIoU).

nIoU ή fwIoU Είναι γνωστό ότι η γενική μετρική IoU είναι προκατειλημμένη προς τις κλάσεις που καλύπτουν μεγάλη περιοχή της εικόνας. Σε περιβάλλοντα με έντονη μεταβολή της κλίμακας όπως στις αστικές σκηνές, αυτό μπορεί να αποτελέσει πρόβλημα. Ειδικά για τους πεζούς, που αποτελούν τις κύριες κατηγορίες στη συγκεκριμένη περίπτωση, στοχεύουμε στο να αξιολογήσουμε πόσο καλά αντιπροσωπεύονται οι ατομικές περιπτώσεις στη σκηνογράφηση. Για να αντιμετωπίσουμε αυτό το πρόβλημα, αξιολογούμε επιπλέον τη σημασιολογική κατάτμηση χρησιμοποιώντας μια κανονικοποιημένη μετρική IoU, την nIoU.

$$nIoU = \frac{truepos}{truepos + falsepos + falseneg} \tag{0.0.3}$$

Πάλι, $truepos$, $falsepos$ και $falseneg$ είναι τα πλήθη των πραγματικά θετικών, ψευδών θετικών και ψευδών αρνητικών pixel για μια συγκεκριμένη κλάση αντίστοιχα. Ωστόσο, σε αντίθεση με την τυπική μετρική IoU, αυτή η μετρική πολλαπλασιάζεται με το λ, το οποίο είναι ένας παράγοντας της συχνότητας αυτής της κλάσης. Αυτός ο παράγοντας είτε προκαθορίζεται από το benchmark είτε υπολογίζεται ως εξής:

$$\begin{aligned} \lambda &= (\sum_{i=1}^{k} t_i)^{-1} \\ nIoU \text{ or } fwIoU &= (\sum_{i=1}^{k} t_i)^{-1} \sum_{i=1}^{k} \frac{n_{ii}}{t_i - n_{ii} + \sum_{j=1}^{k} n_{ji}} \in [0,1] \end{aligned} \tag{0.0.4}$$

iIoU Αυτή η μετρική είναι η μέση τιμή της nIoU σε όλες τις κλάσεις.

Συναρτήσεις απώλειας στη σημασιολογική κατάτμηση

Στη σημασιολογική κατάτμηση, χρησιμοποιούνται διάφορες συναρτήσεις απώλειας [37] ανάλογα με τα χαρακτηριστικά του συνόλου δεδομένων, και γι' αυτό καμία συνάρτηση απώλειας δεν είναι ανώτερη από την άλλη. Ορισμένα παραδείγματα:

1. Δυαδική Συνάρτηση Cross Entropy

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

   όπου το y αντιπροσωπεύει την πραγματική τιμή και το $\hat{y}$ αντιπροσωπεύει την προβλεπόμενη τιμή.

2. Δυαδική Συνάρτηση Cross Entropy με Βάρη

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

   όπου το y αντιπροσωπεύει την πραγματική τιμή και το $\hat{y}$ αντιπροσωπεύει την προβλεπόμενη τιμή. Σε αυτή τη μετρική, οι θετικοί όροι έχουν βάρη με ένα συντελεστή β, ο οποίος μπορεί να επιλεγεί είτε για τη μείωση των ψευδών αρνήσεων με έναν παράγοντα $\beta > 1$, είτε για τη μείωση των ψευδών θετικών με έναν παράγοντα $\beta < 1$.

3. Ισορροπημένη Δυαδική Συνάρτηση Cross Entropy

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

   ίδια με την προηγούμενη αλλά εφαρμόζει επίσης ένα βάρος στα αρνητικά.

4. Συνάρτηση απώλειας Dice

   $DL(y, \hat{p}) = 1 - \frac{2y\hat{p}+1}{y+\hat{p}+1}$

5. Συνάρτηση απώλειας Focal

   $FL(p_t) = -a_t(1 - p_t)^{\gamma} \log(p_t)$
   όπου

$$p_t = \begin{cases} p, & \text{αν } y = 1 \\ 1 - p, & \text{ανάλογα} \end{cases}$$

   όπου για $\gamma = 1$ λειτουργεί όπως η συνάρτηση απώλειας Cross Entropy.

Σε μη ισορροπημένα σύνολα δεδομένων, οι συναρτήσεις απώλειας βασισμένες στην εστίαση λειτουργούν καλύτερα, ενώ σε ισορροπημένα σύνολα δεδομένων, η δυαδική συνάρτηση της διασταύρωσης είναι πιο κατάλληλη. Σύνολα δεδομένων με χαρακτηριστικά μεταξύ των δύο τύπων που αναφέρθηκαν παραπάνω μπορεί να επωφεληθούν περισσότερο από τον ομαλύνομενο ή γενικευμένο συντελεστή Dice.

*

Θεωρητικό Υπόβαθρο

Συνελικτικά Νευρωνικά Δίκτυα(CNN)

Το νευρώνιο

Τα νευρωνικά δίκτυα εμπνέονται από τη δομή του εγκεφάλου και από τη μονάδα κατασκευής του, το νευρώνιο. Συγκεκριμένα, ένα νευρωνικό δίκτυο αποτελείται από επίπεδα νευρώνων, η λειτουργία των οποίων μιμείται τη λειτουργία ενός βιολογικού νευρώνα. Συγκεκριμένα, ο νευρώνας ενός νευρωνικού δικτύου δέχεται εισόδους και παράγει μια έξοδο. Αυτές οι είσοδοι πολλαπλασιάζονται με βάρη για να ελέγξουν την ένταση της επίδρασής τους στο αποτέλεσμα. Η έξοδος παράγεται από αυτό το σταθμισμένο άθροισμα των εισόδων με την προσθήκη μιας παραμέτρου bias και την εφαρμογή μιας συνάρτησης ενεργοποίησης. Αυτή η συνάρτηση ενεργοποίησης ερμηνεύει το αποτέλεσμα και το μεταφράζει. Για παράδειγμα, μπορεί να παράγει πιθανότητες ή δυαδική έξοδο.

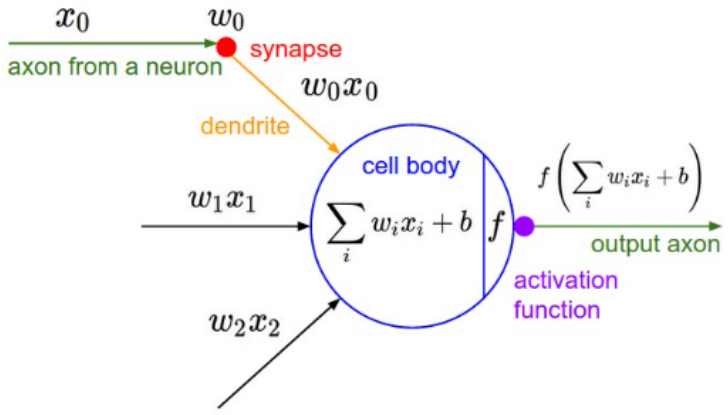


Figure 0.0.4: Το νευρώνιο

Συναρτήσεις ενεργοποίησης Οι συναρτήσεις ενεργοποίησης είναι συναρτήσεις που εφαρμόζονται στο εσωτερικό γινόμενο των εισόδων με τα βάρη του νευρώνα. Εισάγουν μη γραμμικότητα στον υπολογισμό του αποτελέσματος. Συνηθισμένες συναρτήσεις ενεργοποίησης περιλαμβάνουν:

1. Sigmoid Αυτή η συνάρτηση ενεργοποίησης μετατρέπει την είσοδο σε πιθανότητα να ανήκει σε μία από δύο κατηγορίες. Έτσι, το αποτέλεσμα ανήκει στο διάστημα $[0, 1]$, η πρώτη κατηγορία ανατίθεται στο 0 και η δεύτερη στο 1. Έχει τον ακόλουθο τύπο:

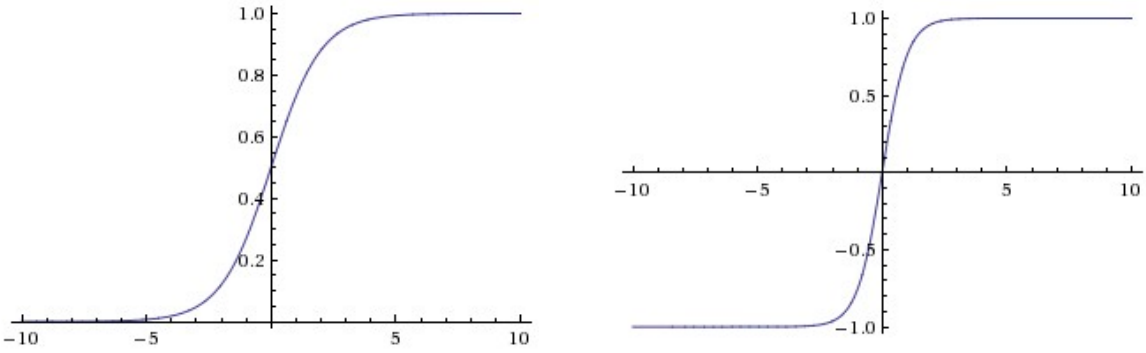
$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{0.0.1}$$

   Ωστόσο, έχει κάποια ανεπιθύμητα χαρακτηριστικά, όπως το γεγονός ότι προκαλεί κορεσμό της παραγώγου και δεν είναι κεντραρισμένη στο μηδέν.

2. Tanh Η γραφική αναπαράσταση της συνάρτησης tanh είναι πολύ παρόμοια με τη συνάρτηση σιγμοειδούς, αλλά τώρα είναι κεντραρισμένη στο μηδέν, όπως φαίνεται στο σχήμα 0.0.5. Η συνάρτηση tanh απεικονίζει την είσοδο στο διάστημα $[-1, 1]$ με τον ακόλουθο τύπο:

$$\tanh(x) = s\sigma(2x) - 1 \tag{0.0.2}$$

   Ωστόσο, εξακολουθεί να μπορεί να προκαλέσει την κλίση της απώλειας να κορεστεί.



(a) Συνάρτηση σιγμοειδούς (b) Συνάρτηση tanh

Figure 0.0.5: Η συνάρτηση tanh είναι παρόμοια με τη συνάρτηση σιγμοειδούς, αλλά είναι κεντραρισμένη στο μηδέν, πράγμα που καθιστά την παραγωγο της συνάρτησης πιο σταθερή.

ReLU Η συνάρτηση ReLU κατώφλιαζει την είσοδο στο μηδέν:

$$f(x) = \max(0, x) \tag{0.0.3}$$

Αυτή η συνάρτηση είναι εύκολα υλοποιήσιμη και εξαλείφει το πρόβλημα της κορεσμένης παραγώγου. Ωστόσο, μπορεί να οδηγήσει σε "θάνατο" νευρώνων κατά τη διάρκεια της εκπαίδευσης με το μηδενισμό των βαρών τους κάτι που είναι μη ανακάμψιμο για το υπόλοιπο της εκπαίδευσης.

3. Leaky-ReLU Αντί να μηδενίζει πλήρως τις αρνητικές εισόδους, η leaky-ReLU τις πολλαπλασιάζει με μια μικρή σταθερά ως εξής:

$$f(x) = \mathbb{1}(x < 0)(\alpha x) + \mathbb{1}(x \geq 0)(x) \tag{0.0.4}$$

η οποία έχει αμφιλεγόμενη απόδοση σε σχέση με την ReLU.

Νευρωνικά Δίκτυα

Τα Νευρωνικά Δίκτυα [38] αποτελούνται από νευρώνες που διαθέτουν εκπαιδεύσιμα βάρη και παραμέτρους. Κάθε νευρώνας λαμβάνει ορισμένες εισόδους, πραγματοποιεί ένα εσωτερικό γινόμενο (dot product) και προαιρετικά ακολουθεί την εφαρμογή ενός μη-γραμμικού στοιχείου. Τα Νευρωνικά Δίκτυα λαμβάνουν μια είσοδο και τη μετασχηματίζουν μέσω μιας σειράς κρυφών στρωμάτων. Κάθε κρυφό στρώμα αποτελείται από ένα σύνολο νευρώνων, όπου κάθε νευρώνας είναι πλήρως συνδεδεμένος με όλους τους νευρώνες στο προηγούμενο στρώμα, και όπου οι νευρώνες σε κάθε στρώμα λειτουργούν εντελώς ανεξάρτητα και δεν μοιράζονται συνδέσεις. Το τελευταίο πλήρως συνδεδεμένο στρώμα ονομάζεται "στρώμα εξόδου" και σε ρυθμίσεις ταξινόμησης αντιπροσωπεύει τις πιθανότητες κάθε κατηγορίας. Στη συνέχεια, η έξοδος συγκρίνεται με την επιθυμητή έξοδο - ground truth - ως απώλεια που υπολογίζεται με μια συνάρτηση απώλειας. Ο κλίση της απώλειας χρησιμοποιείται για να ενημερώσει τα βάρη των στρωμάτων του δικτύου σύμφωνα με τον αλγόριθμο πίσω διάδοσης (backpropagation).

Συνάρτηση απώλειας Η συνάρτηση απώλειας συγκρίνει την πραγματική έξοδο με την επιθυμητή έξοδο. Με βάση το αποτέλεσμά της, οι υπερπαράμετροι ενός μοντέλου προσαρμόζονται κατά την εκπαίδευση, προκειμένου να τη μειώσουν. Χρησιμοποιώντας μια συνάρτηση απώλειας L για τον υπολογισμό της απώλειας μεταξύ της πραγματικής εξόδου και της επιθυμητής, παίρνουμε τη συνολική απώλεια από τον μέσο όρο των ξεχωριστών απωλειών για κάθε δεδομένο του συνόλου εκπαίδευσης:

$$J(w^T, b) = \frac{1}{m} \sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) \tag{0.0.5}$$

όπου w είναι τα βάρη, b η πόλωση, m ο συνολικός αριθμός των σημείων δεδομένων του συνόλου εκπαίδευσης, $\hat{y}$ είναι η πρόβλεψη και y η επιθυμητή έξοδος.

Βελτιστοποίηση - Κλίση Κατάβασης Η συνάρτηση απώλειας μας επιτρέπει να ποσοτικοποιήσουμε την ποιότητα οποιασδήποτε συγκεκριμένης ομάδας βαρών W. Ο στόχος της βελτιστοποίησης είναι να βρεί τα W που ελαχιστοποιούν τη συνάρτηση απώλειας. Είναι δυνατόν να υπολογίσουμε την καλύτερη κατεύθυνση κατά την οποία το διάνυσμα των βαρών θα πρέπει να αλλάξει, η οποία μαθηματικά εγγυάται ότι είναι η κατεύθυνση της πιο απότομης κλίσης. Αυτή η κατεύθυνση θα σχετίζεται με την κλίση της συνάρτησης απώλειας ως εξής:

$$w^{l+1} = w^l - \eta \frac{\partial J(w^T, b)}{\partial w} \tag{0.0.6}$$

όπου η είναι ο ρυθμός μάθησης. Η κατάβαση κλίσης (Gradient Descent) είναι η διαδικασία εύρεσης της κλίσης και στη συνέχεια εκτέλεσης ενημέρωσης παραμέτρων προς αυτήν την κλίση.

Ρύθμιση (Regularization) Ο στόχος ενός νευρωνικού δικτύου είναι να μάθει μια αντιστοίχιση της εισόδου στην έξοδο από τα δεδομένα εκπαίδευσης και να την εφαρμόσει στα δεδομένα ελέγχου. Επομένως, είναι σημαντικό να μπορεί να γενικεύει τα βάρη του και όχι να μαθαίνει τα συγκεκριμένα παραδείγματα των δεδομένων εκπαίδευσης. Όταν ένα μοντέλο ταιριάζει τέλεια στα δεδομένα εκπαίδευσης, αυτό χαρακτηρίζεται ότι υπερ-προσαρμόζεται (overfitting). Η ρύθμιση (Regularization) είναι μια τεχνική ελέγχου της υπερ-προσαρμογής των νευρωνικών δικτύων. Συγκεκριμένα, προσθέτει έναν επιπλέον όρο στη συνάρτηση απώλειας που αποτρέπει τα βάρη από το να αυξάνουν υπερβολικά και, κατά συνέπεια, να ενημερώνονται με λιγότερη ευελιξία.

1. L1 ρύθμιση (L1 Regularization) Στην L1 ρύθμιση, για κάθε βάρος w προσθέτουμε τον όρο $\lambda_1|w|$ στη συνάρτηση απώλειας. Οι νευρώνες με L1 ρύθμιση καταλήγουν να χρησιμοποιούν μόνο ένα αραιό υποσύνολο των πιο σημαντικών εισόδων τους και να γίνονται σχεδόν ανεξάρτητοι από τις "θορυβώδεις" εισόδους. Επίσης κρατάει το μέτρο των διανυσμάτων μικρό.
2. L2 ρύθμιση (L2 Regularization) Η L2 ρύθμιση είναι ίσως η πιο συνηθισμένη μορφή ρύθμισης. Για κάθε βάρος ο όρος $\frac{1}{2}\lambda w^2$ προστίθεται στη συνάρτηση απώλειας όπου λ ιείναι η παράγοντας της ρύθμισης. Ο παράγοντας $\frac{1}{2}$ χρησιμοποιείται έτσι ώστε η κλίση του όρου να είναι πιο απλή, δηλαδή λw. Η L2 ρύθμιση ποινικοποιεί έντονα τους παράγοντες βάρους που έχουν κορυφές και προτιμά πιο ενιαίες τιμές στους παράγοντες βάρους. Αυτό έχει το πλεονέκτημα ότι ενθαρρύνει το δίκτυο να χρησιμοποιεί ισορροπημένα τις εισόδους του. Επιπλέον, κατά τη διάρκεια της ενημέρωσης των παραμέτρων με τη μέθοδο της κλίσης, η L2 ρύθμιση μετατοπίζει το κάθε βάρος γραμμικά προς το μηδέν, δηλαδή $w+ = -\lambda \times w$ προς το μηδέν. Είναι δυνατόν να συνδυαστεί η L1 ρύθμιση με την L2 ρύθμιση: $\lambda_1|w| + \lambda_2 w^2$.

Απόρριψη (Dropout) Η απόρριψη (Dropout) [39] είναι μια τεχνική εκπαίδευσης που αποτρέπει την υπερ-προσαρμογή. Η πιθανότητα απόρριψης υποδεικνύει την πιθανότητα με την οποία ένα νευρώνας παραμένει ενεργός ή τίθονται τα βάρη του σε μηδενικές τιμές. Μπορεί να θεωρηθεί και ως δειγματοληψία ενός νευρωνικού δικτύου εντός του πλήρους νευρωνικού δικτύου με ενημέρωση μόνο των παραμέτρων του δειγματοληπτημένου δικτύου.

Μέγεθος Δέσμης (Batch Size)

Το μέγεθος της δέσμης είναι μια υπερπαράμετρος που καθορίζει τον αριθμό των δειγμάτων που εξετάζονται πριν ενημερωθούν οι εσωτερικές παράμετροι του μοντέλου. Στο τέλος της δέσμης, οι προβλέψεις συγκρίνονται με τις αναμενόμενες εξόδους και υπολογίζεται ένα σφάλμα. Από αυτό το σφάλμα, χρησιμοποιείται ο αλγόριθμος ενημέρωσης για τη βελτίωση του μοντέλου, π.χ. κίνηση κάτω κατά μήκος της κλίσης σφάλματος. Ένα σύνολο εκπαίδευσης μπορεί να διαιρεθεί σε μία ή περισσότερες δέσμες.

Συνέλιξη (Convolution)

Η συνέλιξη [40] είναι μια μαθηματική πράξη που εφαρμόζεται σε δύο συναρτήσεις. Η διακριτή δυσδιάστατη συνέλιξη περιλαμβάνει δύο πίνακες και καθορίζεται από τον ακόλουθο τύπο:

$$(f * w)[x_1, x_2] = \sum_{m=-k}^{k} f(i,j) \cdot w(x_1 - i, x_2 - j) \tag{0.0.7}$$

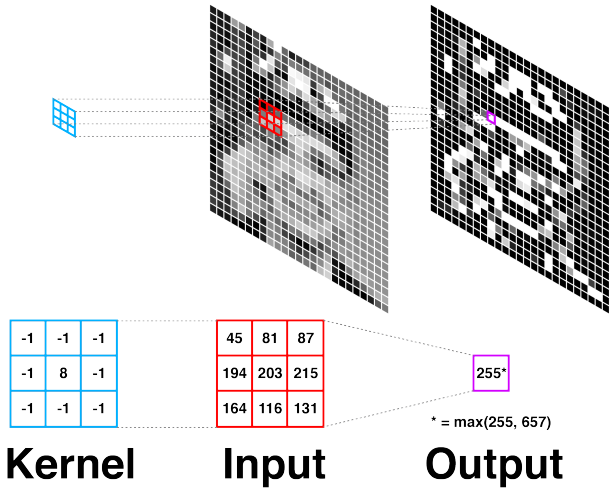


Figure 0.0.6: Οπτικοποίηση της συνέλιξης [41]

Γενικά, η συνέλιξη ορίζεται για πίνακες με το πρώτο όρισμα (f) να αναφέρεται συνήθως ως η είσοδος, ενώ το δεύτερο όρισμα (w) ως ο πυρήνας. Το αποτέλεσμα αναφέρεται ως χάρτης χαρακτηριστικών.

Αυτή η λειτουργία χρησιμοποιείται στα CNN για την επεξεργασία μιας εικόνας και την αναγνώριση σε αυτήν συγκεκριμένων χαρακτηριστικών. Αντίθετα από ένα κανονικό Νευρωνικό Δίκτυο, οι στρώσεις ενός CNN έχουν νευρώνες διατεταγμένους σε τρεις διαστάσεις: πλάτος, ύψος, βάθος. Οι νευρώνες σε μια στρώση συνδέονται μόνο με μια μικρή περιοχή της προηγούμενης στρώσης, αντί να είναι συνδεδεμένοι με όλους τους νευρώνες με πλήρως συνδεδεμένο τρόπο. Όπως περιγράφηκε παραπάνω, ένα απλό CNN αποτελείται από μια ακολουθία στρωσών, και κάθε στρώση ενός CNN μετατρέπει έναν όγκο ενεργοποιήσεων σε έναν άλλο μέσω μιας διαφορισιμής συνάρτησης. Οι τύποι στρωσών που χρησιμοποιούνται για να δημιουργηθεί η αρχιτεκτονική ενός CNN περιλαμβάνουν: Συνελικτική Στρώση, Στρώση Συγκέντρωσης (Pooling Layer), και Πλήρως Συνδεδεμένη Στρώση (Fully-Connected Layer).

Στρώση Συγκέντρωσης (Pooling Layer)

Οι στρώσεις συγκέντρωσης (pooling layers) είναι υπεύθυνες για την υποδειγματοληψία των χωρικών διαστάσεων της εισόδου. Το μέγεθος συγκέντρωσης (pooling size) καθορίζει το μέγεθος του πλαισίου με το οποίο χωρίζεται η είσοδος. Για παράδειγμα, η μέγιστη συγκέντρωση (max pooling) διατηρεί το μέγιστο αριθμό στο πλαίσιο αυτό και αντικαθιστά το πλαίσιο με αυτόν τον αριθμό. Έτσι, πραγματοποιείται υποδειγματοληψία κατά ένα παράγοντα ίσο με το μέγεθος του πλαισίου.

Κανονικοποίηση Δέσμης (Batch Normalization)

Η κανονικοποίηση δέσμης (BN) χρησιμοποιείται για την κανονικοποίηση των εισόδων σε μια συγκεκριμένη στρώση. Αυτό περιλαμβάνει τη διασφάλιση ότι η είσοδος έχει μέσο όρο μηδέν και διακύμανση μονάδα. Η στρώση BN μετασχηματίζει κάθε είσοδο στην τρέχουσα δέσμη αφαιρώντας τον μέσο όρο της εισόδου και διαιρώντας την

με την τυπική απόκλιση. Ωστόσο, δεν έχει αποδειχθεί ότι τα μοντέλα είναι καλύτερα με μηδενικό μέσο όρο και διακύμανση μονάδας. Ενδέχεται να αποδίδουν καλύτερα με άλλο μέσο όρο και διακύμανση. Επομένως, η στρώση BN εισάγει επίσης δύο παραμέτρους, το γάμα (γ) και το βήτα (β), που προσαρμόζονται κατά την εκπαίδευση.

Στρώση Πλήρως Συνδεδεμένη (Fully Connected Layer)

Οι νευρώνες σε μια πλήρως συνδεδεμένη στρώση (fully connected layer) έχουν πλήρεις συνδέσεις με όλες τις ενεργοποιήσεις στην προηγούμενη στρώση, όπως στα κανονικά νευρωνικά δίκτυα.

Μετασχηματιστές (Transformers)

Αρχιτεκτονική

Οι Μετασχηματιστές (Transformers) έχει αποδειχθεί ότι υπερτερεί των αλυσιδωτών και συνελικτικών μοντέλων στην επεξεργασία φυσικής γλώσσας, ενώ παρουσιάζει μικρότερη υπολογιστική πολυπλοκότητα και, συνεπώς, ταχύτερο χρόνο εκπαίδευσης. Ενώ στα CNNs απαιτούνται πολλά στοιβαγμένα επίπεδα για τον εντοπισμό εξαρτήσεων μεγάλης εμβέλειας, στον μηχανισμό αυτο-προσοχής των μετασχηματιστών, αυτές οι εξαρτήσεις ανιχνεύονται αποτελεσματικά για μια θέση εξόδου υπολογίζοντας πληροφορίες περιεχομένου από κάθε θέση εισόδου. Η αρχιτεκτονική των Μετασχηματιστών, όπως φαίνεται στο Σχήμα 0.0.7, αποτελείται από έναν κωδικοποιητή και έναν αποκωδικοποιητή. Ο κωδικοποιητής αποτελείται από 6 στοιβαγμένα επίπεδα, το καθένα αποτελούμενο από ένα πολυκεφαλικό μπλοκ αυτο-προσοχής και ένα δίκτυο προώθησης(feed-forward network). Ο αποκωδικοποιητής αποτελείται επίσης από τα 6 ίδια στοιβαγμένα επίπεδα, με ένα πρόσθετο πολυκεφαλικό μπλοκ αυτο-προσοχής που χρησιμοποιείται στην αρχή κάθε επιπέδου. Το πολυκεφαλικό μπλοκ αυτο-προσοχής αποτελείται από τα 8 προαναφερόμενα κεφάλια αυτο-προσοχής που εφαρμόζονται παράλληλα στην είσοδο και έχουν διαφορετικά εκπαιδεύσιμα βάρη. Ο σκοπός τους είναι να δημιουργήσουν διαφορετικές εκδοχές σχετικά με τις εξαρτήσεις σε μια ακολουθία και να τις συνδυάσουν σε έναν μόνο μέσο όρο εξόδου αυτο-προσοχής. Η αυτο-προσοχή μάσκας περιορίζει την εμβέλεια της αυτο-προσοχής, ώστε το υπολογισμός των εξόδων να μην επηρεάζεται από τα "μελλοντικά" αποτελέσματα ή ώστε να περιορίσει την υπλογιστική πολυπλοκότητα.

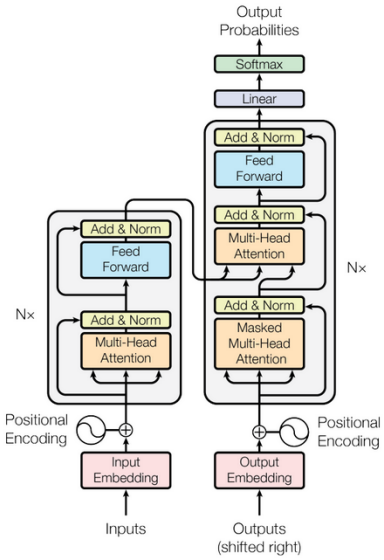


Figure 0.0.7: Transformer Architecture

Κάθε επίπεδο έχει έξοδο και είσοδο με τις ίδιες διαστάσεις d_{model}. Επίσης, διαθέτει υπολειμματική σύνδεση (residual connection) και μονάδα κανονικοποίησης δέσμης (batch normalization) μετά από κάθε υπο-επίπεδο. Τέλος, επιπλέον χαρακτηριστικά(όπως αποτέλεσματα εφαρμογής τριγωνομετρικών συναρτήσεων διαφορετικών συχνοτήτων στην είσοδο) ενσωματώνονται στο διάνυσμα εισόδου προκειμένου να μοντελοποιηθούν χωρικές ή χρονικές σχέσεις μεταξύ θέσεων και να έχει διαστάσεις d_{model}. Το δίκτυο προώθησης (Feed-Forward) υλοποιεί δύο γραμμικές μετασχηματίσεις με δύο ενεργοποιήσεις ReLU μεταξύ τους:

$$\mathrm{FF}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{0.0.8}$$

Αυτο-Προσοχή(Self-Attention)

Ξεκινώντας με τρία εκπαιδεύσιμα διανύσματα: το διάνυσμα ερώτησης (Q) και το ζεύγος κλειδιού (K) - τιμής (V), το υπο-επίπεδο αυτό-προσοχής στοχεύει να υπολογίσει την τιμή που προκύπτει από το κλειδί που είναι πιο κοντά στην ερώτηση. Παίρνοντας το εσωτερικό γινόμενο 0.0.9 μεταξύ της ερώτησης και του κλειδιού και μια συνάρτηση softmax, καταλήγουμε σε έναν πίνακα πιθανοτήτων ομοιότητας. Αυτός ο πίνακας χρησιμοποιείται ως βάρη για τον υπολογισμό της τιμής από το σταθμισμένο άθροισμα των τιμών του πίνακα τιμών. Το εσωτερικό γινόμενο μεταξύ της ερώτησης και του πίνακα κλειδιών κλιμακώνεται με έναν παράγοντα $\frac{1}{\sqrt{d_k}}$ (όπου d_k είναι η διάσταση του πίνακα κλειδιών) προκειμένου να αποτραπεί το πρόβλημα της εκρηγνυμένης κλίσης όταν έχουμε μεγάλες τιμές.

$$Attention(K, Q, V) = \mathrm{softmax}(\frac{QV^T}{\sqrt{d_k}})V \tag{0.0.9}$$

Στην πολυκεφαλική αυτο-προσοχή, υπάρχουν 8 κεφάλια αυτο-προσοχής, καθένα με τα δικά του εκπαιδεύσιμα βάρη που παρέχουν ένα διαφορετικό χώρο αναπαράστασης. Οι εξόδοι τους συνενώνονται ως εξής:

$$\begin{aligned} MH(Q, K, V) &= Concat(h_1, h_1, ..., h_h)W^o \\ h_i &= Attention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{0.0.10}$$

όπου οι πίνακες W είναι προβολές που φέρνουν τους πίνακες Q, K σε διάσταση d_{model}/h (όπου ο αριθμός των κεφαλών h=8) και φέρνουν τη συνενωμένη αναπαράσταση εξόδου σε διάσταση d_{model}.

Γενικά, η αυτο-προσοχή είναι πιο γρήγορη από τα συνελικτικά επίπεδα με το ίδιο πεδίο προσοχής, και στη χείριστη περίπτωση, η διαχωρίσιμη συνέλιξη(separable convolution) έχει την ίδια πολυπλοκότητα με την αυτο-προσοχή.

Σχετική βιβλιογραφία

Το πρόβλημα της σημασιολογικής κατάτμησης [42] προσεγγίστηκε αρχικά με αλγορίθμους συσταδοποίησης [43] που χρησιμοποιούν επιπρόσθετες πληροφορίες από περίγραμματα και άκρες. Αργότερα εξερευνήθηκε η μοντελοποίηση με διαδικασία Markov [5] καθώς και η συνδυασμένη ανίχνευση περιγράμματος σε μια ιεραρχική προσέγγιση. Ωστόσο, σήμερα τα κυρίαρχα μοντέλα κατευθύνονται σε μοντέλο που βασίζονται σε περιοχές, συνελικτικά νευρωνικά δίκτυα και μετασχηματιστές.

Μοντέλα που βασίζονται σε περιοχές

Τα μοντέλα που βασίζονται σε περιοχές χωρίζονται σε δύο ξεχωριστά στάδια: τον εντοπισμό των περιοχών σε μια εικόνα και την ταξινόμησή τους σύμφωνα με τις κατηγορίες που ορίζονται στο σύνολο δεδομένων. Στα εικονοστοιχεία ανατίθενται κλάσεις μετά από αυτά τα δύο στάδια, είτε με ολοκληρωτικά εκπαιδεύσιμο τρόπο είτε με εκπαιδεύσιμα μόνο τα δύο πρώτα στάδια.

R-CNN

Η άνοδος των CNNs στην κατηγοριοποίηση εικόνων οδήγησε στη δημιουργία του R-CNN [44], το οποίο είναι το πρώτο μοντέλο που έδειξε ότι τα CNN υπερτερούν στην ανίχνευση αντικειμένων. Αργότερα, τροποποιήθηκε για να χρησιμοποιηθεί στη σημασιολογική κατάτμηση. Στην ανίχνευση αντικειμένων, το R-CNN εξάγει περίπου 2000 προτάσεις περιοχών με χρήση της επιλεκτικής αναζήτησης. Αυτές οι περιοχές μετατρέπονται σε ένα καθορισμένου μεγέθους διάνυσμα χαρακτηριστικών χρησιμοποιώντας ένα μεγάλο CNN, το οποίο κατόπιν ταξινομείται

σε κλάσεις με γραμμικά SVMs. Εφαρμόζεται ένας αλγόριθμος μη μέγιστης κατάπνιξης (Non Maximum Suppression) για να απορρίψει μια περιοχή αν έχει χαμηλότερη βαθμολογία από μια περιοχή που τέμνεται μαζί της με διαφορά πάνω από ένα εκπαιδεύσιμο κατώφλι. Στη σημασιολογική κατάτμηση, R-CNN χρησιμοποιεί μια τροποποιημένη μορφή του O2P [45], το οποίο εξάγει 150 περιοχές με το CPMC [46] και τις αξιολογεί με υποστήριξη διανύσματος οπισθοδρόμησης (SVR). Ύστερα, το στάδιο εξαγωγής χαρακτηριστικών αντικαθίσταται με το CNN, που εξερευνά τρεις μεθόδους: η πρώτη υπολογίζει τα χαρακτηριστικά στο ορθογώνιο κουτί που περικλείει την εντοπισμένη περιοχή, η δεύτερη υπολογίζει τα χαρακτηριστικά μόνο στην προσκήνιο μάσκα της περιοχής και η τρίτη συγχωνεύει τα διανύσματα των δύο παραπάνω μεθόδων. Η καλύτερη μέθοδος απόδοσης ήταν η συγχώνευση των χαρακτηριστικών, δείχνοντας ότι ο περιβάλλων χώρος είναι κρίσιμος στη διαδικασία ταξινόμησης μιας περιοχής.

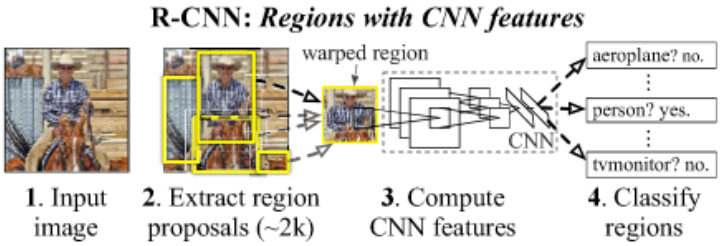


Figure 0.0.8: R-CNN [44] μοντέλο ανίχνευσης αντικειμένου

Fast and Faster R-CNN

Fast R-CNN [47] είναι μια βελτιωμένη έκδοση του R-CNN [44], η οποία χρησιμοποιείται στην ανίχνευση αντικειμένων και επηρεάζει αργότερα τις αρχιτεκτονικές στη σημασιολογική κατάτμηση. Συγκεκριμένα, εισάγει την Περιοχή Ενδιαφέροντος (ROI), η οποία θα εφαρμοστεί σε μοντέλα σημασιολογικής κατάτμησης που βασίζονται σε περιοχές και προσαρμόζονται σε περιοχές ελεύθερης μορφής. Είναι γρηγορότερο από το R-CNN, διότι περιλαμβάνει εκπαίδευση σε μία φάση με απώλεια που συγχωνεύει πολλαπλές εργασίες, ενώ η εκπαίδευση μπορεί να ενημερώσει όλα τα επίπεδα του δικτύου. Η Περιοχή Ενδιαφέροντος εξάγει ένα διάνυσμα χαρακτηριστικών από τα χαρακτηριστικά που παράγει το συνελικτικό δίκτυο ακι από τις συντεταγμένες της περιοχής. Χωρίζει τις περιοχές σε ισοπλήθη υποπεριοχές και κρατάει την μέγιστη τιμή τους όπως μία Στρώση Συγχώνευσης(Pooling layer) . Σε επόμενη επέκταση, το Faster R-CNN [48] επιταχύνει τον χρόνο εκπαίδευσης χρησιμοποιώντας το ίδιο συνελικτικό δίκτυο τόσο στην αναγνώριση περιοχών όσο και στην ταξινόμησή τους και επομένως το μοντέλο γίενται εκπαιδεύσιμο από άκρη σε άκρη.

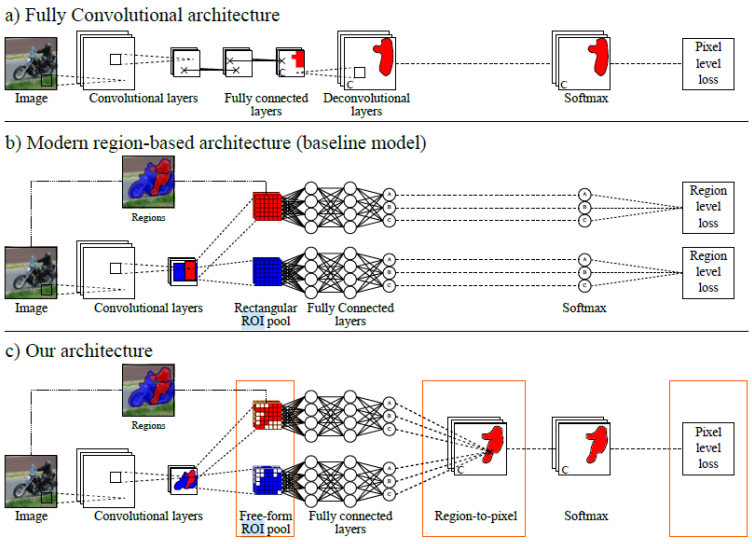


Figure 0.0.9: Σύγκριση των τριών αρχιτεκτονικών: a) Πλήρως Συνελικτά Δίκτυα στην σημασιολογική κατάτμηση είναι από άκρη σε άκρη εκπαιδεύσιμα αλλά αναγνωρίζουν περιοχές από τετράγωνα αποκόμματα και συνεπώς παράγουν ανακριβείς προβλέψεις για πολύπλοκες περιοχές. b) Το μοντέλο αναγνώρισης περιοχών με συνάρτηση απώλειας που υπολογίζεται στο στάδιο αναγνώρισης περιοχών c) Το μοντέλο αναγνώρισης περιοχών από άκρη σε άκρη εκπαιδεύσιμο αφού υπολογίζει την συνάρτηση απώλειας πάνω στα εικονοστοιχεία. Εικόνα από [49]

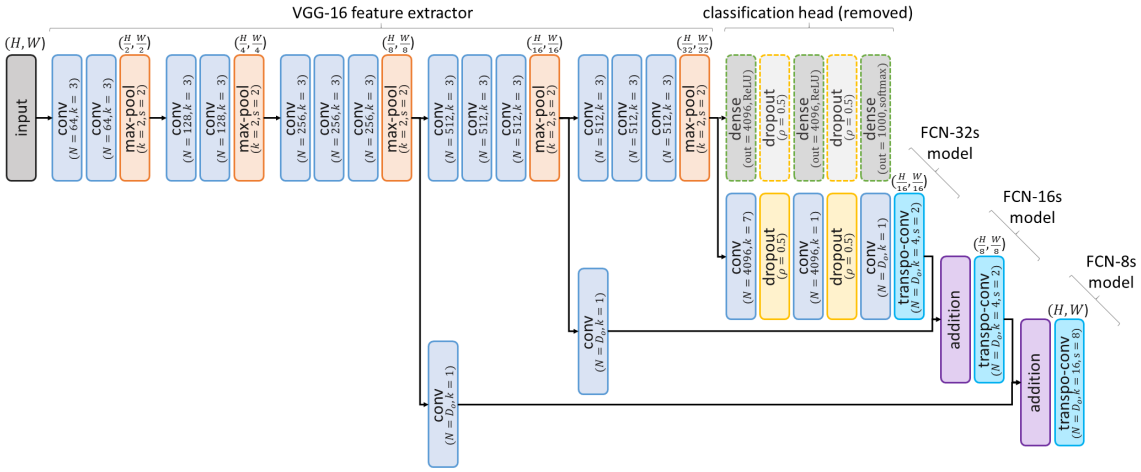


Figure 0.0.10: FCN αρχιτεκτονική βασισμένη στο VGG-16 από το Κεφ.6 στο [51]

Convolutional Neural Networks

Πλήρως συνελικτικά Νευρωνικά Δίκτυα(FCN)

Τα πλήρως συνελικτικά Νευρωνικά Δίκτυα (FCN), όπως περιγράφονται στο [50], είναι δίκτυα που δέχονται εισόδους οποιουδήποτε μεγέθους και παράγουν εξόδους ίδιου μεγέθους. Περιλαμβάνουν μόνο συνελικτικά επίπεδα και μηχανισμούς υπερ/υπό-δειγματοληψίας.Σκοπός ήταν η μετατροπή των Συνελικτικών Δικτύων ώστε να χρησιμοποιηθούν στην κατάτμηση εικόνων. Αυτό ήταν δυνατό με την αντικατάσταση των πλήρως συνδεδεμένων επιπέδων με συνελικτικά επίπεδα πυρήνα μεγέθους 1×1, παράγοντας έτσι ίσου μεγέθους εξόδου. Το αρχικό μέγεθος της εισόδου, το οποίο μειώνεται από τη υπο-δειγματοληψία στα αρχικά συνελικτικά επίπεδα, μπορεί να ανακτηθεί από μηχανισμούς υπερ-δειγματοληψίας. Ο μηχανισμός που χρησιμοποιήθηκε ήταν η αντίστροφη συνέλιξη, η οποία χρησιμοποιεί βήμα $\frac{1}{f}$ και βάρη που μπορούν να εκπαιδευτούν. Επιπλέον, για να

επιτευχθούν πιο ακριβείς προβλέψεις, λαμβάνοντας υπόψη τα τους περιορισμούς της υποδειγματοληψίας στην πρόβλεψη του αποτελέσματος, παρουσιάζεται ο μηχανισμός μεταπίδησης αρχιτεκτονικής (skip architecture) που προσθέτει μη γραμμικότητα στο μοντέλο και εκτελεί στοιχειώδη πρόσθεση των προβλέψεων από προηγούμενα επίπεδα στις υπερδειγματοληπτημένες προβλέψεις.

U-Net

U-Net Η αρχιτεκτονική U-Net [52] βασίζεται στην αρχιτεκτονική FCN χρησιμοποιώντας την αρχιτεκτονική μεταπίδησης και τις αντίστροφες συνελίξεις, αλλά εκπαιδεύεται πολύ γρηγορότερα και παράγει πιο ακριβή αποτελέσματα. Η αρχιτεκτονική μοιάζει με ένα U, καθώς κατασκευάζει συμμετρικά τον κωδικοποιητή και τον αποκωδικοποιητή, όπως φαίνεται στο 0.0.11. Η τροποποίηση που επιτρέπει καλύτερη ακρίβεια είναι το μεγάλο πλήθος χαρακτηριστικών στην πλευρά του αποκωδικοποιητή, ενώ στο FCN η πλευρά του "αποκωδικοποιητή", που αποτελείται από διαδοχικές αντίστροφες συνελίξεις και προσθέσεις, έχει σταθερό αριθμό χαρακτηριστικών ίσο με τον αριθμό των κατηγοριών. Όσον αφορά τον κωδικοποιητή, αποτελείται από 3 ομάδες 2 διαδοχικών στρωμάτων συνελίξεων ακολουθούμενων από ReLUs και μία Στρώση Συγχώνευσης. Αυτό επαναλαμβάνεται συμμετρικά στην πλευρά του αποκωδικοποιητή, με τις Στρώσεις Συγχώνευσης να αντικαθίστανται από αντίστροφες συνελίξεις και ένα επιπλέον επίπεδο προσθήκης να προηγείται κάθε επιπέδου υλοποιώντας την αρχιτεκτονική μεταβίβασης.

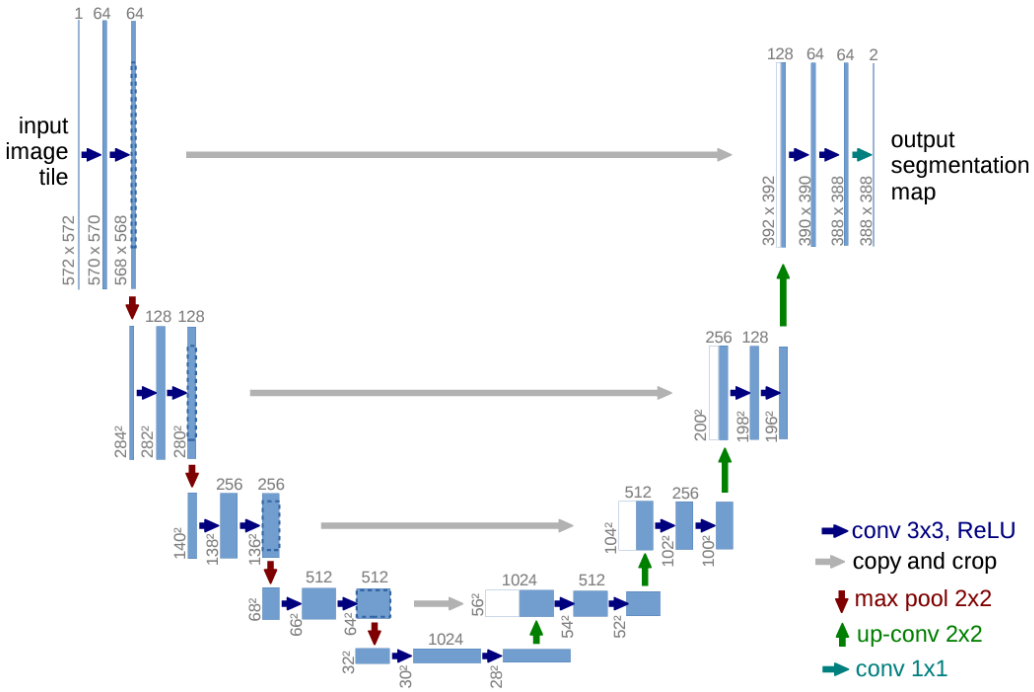


Figure 0.0.11: U-Net αρχιτεκτονική από [52]

DeepLab and Atrous Convolution

Το DeepLab [53] βασίζεται στις προηγούμενες αρχιτεκτονικές ενσωματώνοντας στις αρχιτεκτονικές του FCN την atrous συνέλιξη αντικαθιστώντας την αντίστροφη συνέλιξη. Η atrous συνέλιξη στη μία διάσταση παρουσιάζεται στην εξίσωση 0.0.1, και μπορεί να περιγραφεί από το γεγονός ότι κάνει συνέλιξη με ένα φίλτρο που έχει r-1 μηδενικά ανάμεσα στις τιμές του όταν ο ρυθμός διατάσεων είναι r. Αυτό επιτρέπει ένα μεγαλύτερο πεδίο θέασης, διατηρώντας τον ίδιο αριθμό παραμέτρων και διατηρώντας την ανάλυση όταν χρησιμοποιείται με γέμισμα(padding).

$$y[i] = \sum_k x[i + rk]w[k] \tag{0.0.1}$$

Βαθιά Συνελικτικά Δίκτυα Υψηλής Ανάλυσης (HRNet)

Τα Βαθιά Συνελικτικά Δίκτυα Υψηλής Ανάλυσης προέκυψαν από την παρατήρηση ότι κανένα από τα προηγούμενα μοντέλα δεν διατήρησε ροές υψηλής ανάλυσης σε όλο το δίκτυο, αλλά αντίθετα υπερδειγματολήπτησαν την ανάλυση από τη χαμηλή ανάλυση [50], συγχώνευσαν αρχικών επιπέδων υψηλή ανάλυση στα τέλη του δικτύου [52], δημιούργησαν ροές με μεσαία ανάλυση [53, 54], ή υλοποίησαν αρχιτεκτονικές κωδικοποιητή-αποκωδικοποιητή [55, 52]. Το HRNet [56] εισήγαγε την έννοια των παράλληλων ροών πολλαπλών αναλύσεων που συγχωνεύονται στο τέλος κάθε σταδίου, όπως φαίνεται στην 0.0.12. Το δίκτυο αποτελείται από 4 στάδια, και σε κάθε στάδιο προστίθεται παράλληλα μια ροή χαμηλότερης ανάλυσης. Σε κάθε στάδιο υπάρχουν 4 μονάδες σε κάθε ανάλυση, με κάθε μονάδα να αποτελείται από δύο συνελίξεις πυρήνα 3×3, κανονικοποίηση δέσμης και relu. Με εξαίρεση το πρώτο στάδιο όπου κάθε μονάδα αποτελείται από ένα bottleneck πλάτους 64 και μια συνέλιξη πυρήνα 3×3. Η πρώτη ροή είναι ανάλυσης $\frac{1}{4}$ και έχει C κανάλια και κάθε διαδοχική ροή κατεβάζει την ανάλυση και αυξάνει τα κανάλια κατά παράγοντα 2. Η υποδειγματοληψία γίνεται δυνατή με μια συνέλιξη 3×3 με βήμα 2, ενώ η υπερδειγματοληψία με διγραμμική παρεμβολή(bilinear interpolation) και συνέλιξη πυρήνα 1×1. Το αποτέλεσμα της σημασιολογικής κατάτμησης στο HRNetV2 προκύπτει από τη συγχώνευση των τεσσάρων ροών στην πρώτη ροή, μετά την υπερδειγματοληψία των τριών τελευταίων ροών και τη χρήση χαρτών εκτίμησης κατάτμησης (estimating segmentation maps).

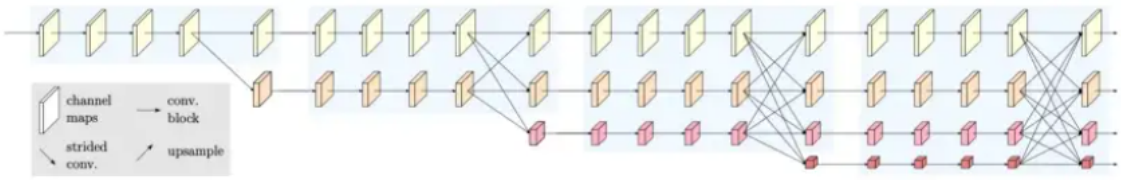


Figure 0.0.12: HRNet αρχιτεκτονική από [50]

Μετασχηματιστές

Μετασχηματιστής Μετατόπισης Παραθύρου (Swin Transformer)

Το Swin Transformer [57] είναι μια αρχιτεκτονική που στοχεύει στο να μετατρέψει τον Μετασχηματιστή από ένα εργαλείο επεξεργασίας φυσικής γλώσσας (NLP) σε ένα εργαλείο όρασης. Ωστόσο, για να μπορεί να εκμεταλλευτεί τα πλεονεκτήματα της αυτο-προσοχής (self-attention) στις εικόνες με αποτελεσματικότητα ενώ παράλληλα να ανιχνεύει εξαρτήσεις πολλών κλιμάκων, υλοποιεί την προσοχή μετατόπισης παραθύρου (shifted window attention) και τη συγχώνευση αποκομμάτων (patch merging). Σύμφωνα με αυτήν την αρχιτεκτονική, μια εικόνα διαιρείται σε κομμάτια μεγέθους 4×4 και αυτά τα κομμάτια ανήκουν σε παράθυρα μεγέθους $M \times M$. Το επίπεδο πολλαπλών κεφαλιών αυτό-προσοχής (multi-head self-attention) αντικαθίσταται από την πολλαπλών κεφαλιών παραθυρομένη αυτό-προσοχή (W-MSA) στον κωδικοποιητή και κάθε προσοχή περιορίζεται μόνο στα πλαίσια ενός παραθύρου. Τα παράθυρα, που αρχίζουν να χωρίζονται από την επάνω-αριστερή γωνία, μετατοπίζονται κάτω και δεξιά σε κάθε επίπεδο προσοχής στον αποκωδικοποιητή. Το Swin Transformer επιτρέπει επίσης

τον εντοπισμό εξαρτήσεων πολλαπλών αναλύσεων μέσω της συγχώνευσης αποκομμάτων σε κάθε στάδιο, όπως φαίνεται στο 0.0.13. Αυτή η αρχιτεκτονική χρησιμοποιεί τη GELU και ένα MLP 2 επιπέδων μετά τη μονάδα προσοχής. Επιπλέον, ένα επίπεδο κανονικοποίησης τοποθετείται μετά από κάθε μονάδα και χρησιμοποιούνται συνδέσεις υπολοίπου μεταξύ των μονάδων, όπως φαίνεται στο 0.0.13.

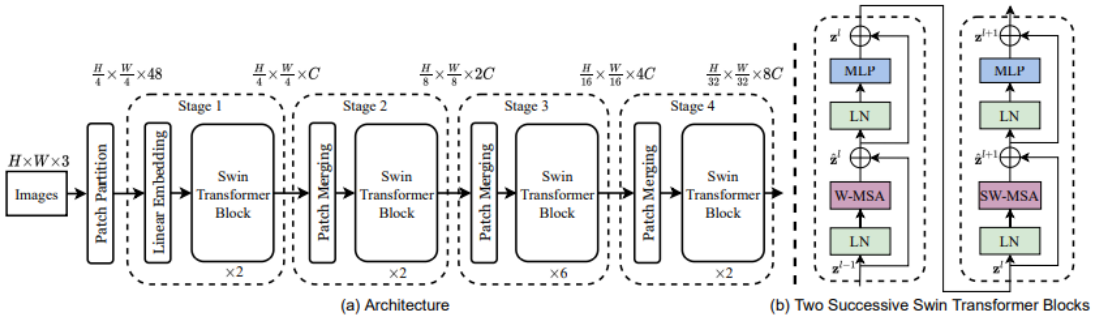


Figure 0.0.13: swin Transformer αρχιτεκτονική από [57]

Patch-merging Τα κομμάτια είναι μεγέθους 4×4 και οι διαστάσεις του παραθύρου ορίζονται ως $\frac{W}{4} \times \frac{H}{4}$. Κάθε κομμάτι έχει ένα token βάσει των $4 \times 4 \times 3 = 48$ χαρακτηριστικών τους. Ωστόσο, το token μετατρέπεται από ένα γραμμικό ενσωματωμένο στρώμα (linear embedding layer) σε διάσταση C και το κομμάτι κατευθύνεται στον Μετασχηματιστή ολοκληρώνοντας το πρώτο στάδιο. Στο δεύτερο στάδιο, ένα επίπεδο συγχώνευσης κομματιών συγχωνεύει $2 \times 2 = 4$ κομμάτια σε ένα κομμάτι με 4C συνενωμένα χαρακτηριστικά που και πάλι μετατρέπονται μέσω ενός γραμμικού ενσωματωμένου στρώματος σε 2C χαρακτηριστικά, ορίζοντας το μέγεθος του παραθύρου ως $\frac{W}{8} \times \frac{H}{8}$. Το δεύτερο στάδιο ολοκληρώνεται με ένα Swin Transofmer μπλοκ. Ομοίως επεξεργάζονται τα χαρακτηριστικά στα στάδια 3 και 4, όπου ξανά το μέγεθος του παραθύρου μειώνεται σε $\frac{W}{16} \times \frac{H}{16}$ και $\frac{W}{32} \times \frac{H}{32}$ αντίστοιχα και κάθε κομμάτι έχει διάσταση χαρακτηριστικών 4C και 8C αντίστοιχα.

Μετατόπιση παραθύρου (Window shifting) Η μετατόπιση παραθύρου εφαρμόζεται στον μηχανισμό πολλαπλών-κεφαλών αυτό-προσοχής στον αποκωδικοποιητή, επιτρέποντας τον εντοπισμό ευρύτερων εξαρτήσεων. Το αρχικό παράθυρο τοποθετείται στην αριστερή-επάνω γωνία, και τα υπόλοιπα τοποθετούνται διαδοχικά χωρίς επικάλυψη. Η επόμενη διαμόρφωση δημιουργείται με την διαγώνια μετατόπιση των παραθύρων. Ωστόσο, αυτό έχει ως αποτέλεσμα περισσότερα παράθυρα από τα αρχικά με κάποια να είναι μερικώς ολοκληρωμένα. Η λύση που δίνεται σε αυτό το πρόβλημα είναι η μέθοδος της κυκλικής μετατόπισης, όπου τα μερικώς ολοκληρωμένα παράθυρα από την επάνω-αριστερή πλευρά μετακινούνται στην κάτω-δεξιά γωνία και συνδέονται με τα μερικώς ολοκληρωμένα παράθυρα εκεί, προκειμένου να δημιουργηθούν κανονικά παράθυρα. Στη συνέχεια, η αυτό-προσοχή που εφαρμόζεται σε αυτά τα συγχωνευμένα παράθυρα λαμβάνει υπόψη αυτή τη μετατόπιση, εφορμόζοντας μια μάσκα στα χαρακτηριστικά του παραθύρου που δεν ήταν γείτονες πριν τη μετατόπιση.

Mask Transformer (MaskFormer, Mask2Former)

Η ιδέα πίσω από τον MaskFormer [19] βασίζεται στην ομοιότητα των διάφορων εργασιών σημασιολογικής κατάτμησης και στην αναγνώριση της ανάγκης για την κατασκευή μιας ενιαίας αρχιτεκτονικής που μπορεί να εκτελέσει και τις τρείς εργασίες. Ο στόχος του MaskFormer είναι να χρησιμοποιήσει την τεχνική της instance segmentation, την ταξινόμηση μασκών, στην σημασιολογική κατάτμηση εικόνας, αμφισβητώντας την έννοια της ταξινόμησης ανά pixel. Συγκεκριμένα, ο MaskFormer προβλέπει ένα σύνολο από N ζευγάρια κλάσης και μάσκας, $z = (c_i, m_i)_{i=1,\dots,N}$ εκτελώντας κατ' αρχάς ταξινόμηση μάσκας, και στη συνέχεια μετατρέπει τα αποτελέσματα σε έξοδο πανοπτικής, instance ή σημασιολογικής κατάτμησης. Έχει παρόμοια προσέγγιση με τις αρχιτεκτονικές R-CNN πριν μετατραπούν σε μοντέλα εκπαίδευσης από άκρη σε άκρη και συμφωνεί με την απουσία απώλειας σε επίπεδο pixel.

Architecture Η αρχιτεκτονική αυτού του μοντέλου ταξινόμησης μάσκας διαιρείται σε τρία μέρη όπως φαίνεται στο Σχήμα 0.0.14. Η "μονάδα σε επίπεδο pixel" χρησιμοποιεί μια αρχιτεκτονική ταξινόμησης επιπέδου pixel

που πρώτα υποδειγματοληπτεί την εικόνα και στη συνέχεια την ανακατασκευάζει με έναν αποκωδικοποιητή. Τα χαρακτηριστικά χαμηλής ανάλυσης που εξάγονται από το backbone προωθούνται στην "μονάδα μετασχηματισή αποκωδικοποίησης", ο οποίος εκτελεί ένα query πάνω σε αυτά και παράγει τα object queries. Ο πίνακας Q του μετασχηματιστή αυτού μαθαίνεται κατά την διάρκεια της εκπαίδευσης και περιέχει N αναπαραστάσεις που χρησιμοποιούνται για την αναγνώριση των N μασκών στην εικόνα. Η έξοδος της "μονάδας μετασχηματιστή αποκωδικοποίησης" είναι ένας πίνακας $C_Q \times N$ και περνά σε ένα MPL που τον μετατρέπει σε έναν πίνακα διάστασης $C_E \times N$. Η έξοδος περνά επίσης σε ένα γραμμικό ταξινομητή και ένα softmax για να δημιουργήσει τις προβλέψεις των κλάσεων των N μασκών.

Ο αποκωδικοποιητής επιπέδου pixel υπερδειγματοληπτεί τα χαρακτηριστικά χαμηλής ανάλυσης για να παράγει υψηλής ανάλυσης ανά pixel χαρακτηριστικά. Αυτά τα χαρακτηριστικά μεγέθους $W \times H \times C_E$ πολλαπλασιάζονται με την έξοδο του MPL, δημιουργώντας ένα αποτέλεσμα μεγέθους $W \times H \times N$, το οποίο περιέχει την πιθανότητα της μάσκας για κάθε pixel μετά την εφαρμογή ενός sigmoid.

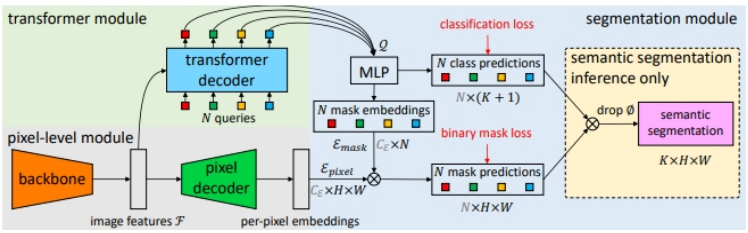


Figure 0.0.14: Αρχιτεκτονική MaskFormer από [19]

Συνεπώς, υπάρχουν δύο κλάδοι σημασιολογικής κατάτμησης: ένας που ταξινομεί μάσκες χρησιμοποιώντας μόνο το backbone και τον μετασχηματιστή αποκωδικοποίησης και έναν που εντοπίζει την περιοχή της μάσκας χρησιμοποιώντας το γινόμενο των χαρακτηριστικών της εικόνας από τον αποκωδικοποιητή σε επίπεδο pixel και τα χαρακτηριστικά των μασκών από τον μετασχηματιστή αποκωδικοποίησης. Η τελική έξοδος της εργασίας ταξινόμησης μάσκας αποτελείται από τις προβλέψεις κλάσης ανά μάσκα και τις προβλέψεις της μάσκας ανά pixel ($m_i[h,w]\epsilon[0,1]$, πιθανότητα ότι η i-οστή μάσκα περιέχει το pixel h, w). Οι δύο αυτές εξόδοι συνδυάζονται για να παράγουν την έξοδο της σημασιολογικής και της πανοπτικής κατάτμησης. Όσον αφορά τη σημασιολογική κατάτμηση, η ταξινόμηση των pixel υπολογίζεται με ανάθεση της κλάσης που παράγει το υψηλότερο γινόμενο της πιθανότητας της μάσκας και της πιθανότητας της κλάσης σε μια συγκεκριμένη μάσκα, δηλαδή $\arg\max_{c\in 1,\dots,K} \sum_{i=1}^{N} p_i(c) m_i[h,w]$. Στη συνέχεια συγχωνεύονται τα pixel με την ίδια κλάση. Ενώ, στην instance segmentation διατηρούμε τόσο την κλάση όσο και την μάσκα, δηλαδή $\arg\max_{i\in 1,\dots,N} p_i(c_i) m_i[h,w]$, όπου $c_i = \arg\max_c p_i(c)$, συγκεντρώνοντας τα pixel που ανήκουν στην ίδια μάσκα.

Υπολογισός απώλειας Ο σφάλμα κατηγοριοποίησης υπολογίζεται με συνδυασμό της απώλειας κατηγοριοποίησης μάσκας και την απώλεια εύρεσης μάσκας. Ωστόσο, για να υπολογιστεί η απώλεια, πρέπει να γίνει αντιστοίχιση μεταξύ των M αποτελεσμάτων των επιθυμητών αποτελεσμάτων (ground truth) και των N αποτελεσμάτων του MaskFormer. Το ισοζύγιο μεταξύ των δύο συνόλων είναι δυνατό με την εισαγωγή N-M αντικειμένων "null". Έτσι, με διμερή αντιστοίχιση (bipartite matching) είναι δυνατή η αντιστοίχιση των μασκών-κλάσεων 1 προς 1 και η υπολογισμός της απώλειας:

$$L_{mask-cls}(z, z^{gt}) = \sum_{j=1}^{N} [-\log p_{\sigma(j)}(c_j^{gt}) + \mathbb{1}_{c_j^{gt}\neq\emptyset} L_{mask}(m_{\sigma(j)}, m_j^{gt})\}] \tag{0.0.2}$$

όπου z είναι το σύνολο προβλέψεων, z^{gt} είναι το σύνολο των επιθυμητών τιμών (ground truth) και $N = |z|$. Το L_{mask} είναι η απώλεια της μάσκας με binary loss που λόγω της χρήσης της συνάρτησης δείκτη (indicator func-

tion) ως παράγοντα, υπολογίζεται μόνο όταν η μάσκα προβλέπει ένα συγκεκριμένο αντικείμενο μιας κατηγορίας. Ο πρώτος όρος είναι η απώλεια ταξινόμησης μάσκας με cross entropy.

Στοιχεία που χρησιμοποιήθηκαν Η "μονάδα σε επίπεδο pixel" αποτελείται από έναν πυρήνα (backbone) και έναν αποκωδικοποιητή σε επίπεδο pixel (pixel-decoder), που μπορούν να υλοποιηθούν από οποιοδήποτε υπάρχον μοντέλο που παράγειαποτελέσματα σημασιολογικής κατάτμησης. Αυτή η δυνατότητα σημαίνει ότι το MaskFormer μπορεί να μετατρέψει ένα οποιοδήποτε κυρίαρχο (SOTA) μοντέλο σε ένα μοντέλο κατηγοριοποίησης μάσκας (mask classification). Έχει αποδειχτεί [19] ότι βελτιώνει την απόδοση των μοντέλων αυτών.

Συνήθως, ως πυρήνας (backbone) χρησιμοποιείται το μοντέλο ResNet ή το μοντέλο Swin-Transformer, ενώ ο αποκωδικοποιητής σε επίπεδο pixel σχεδιάζεται βασιζόμενος σε προηγούμενα μοντέλα όπως το FPN.

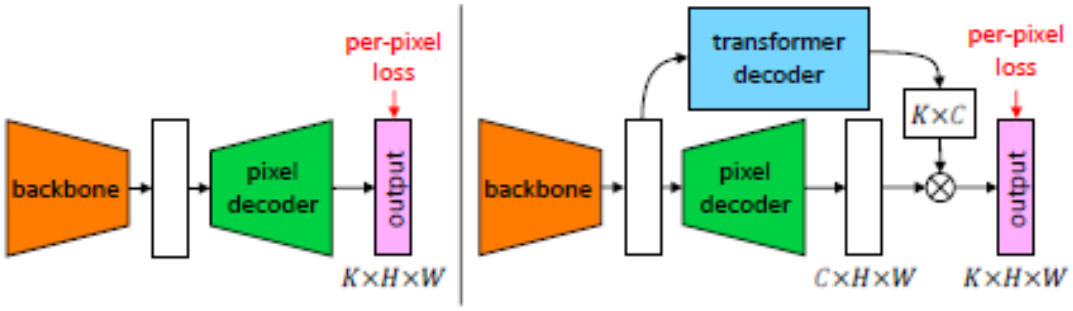


Figure 0.0.15: Μετατροπή ενός μοντέλου σημασιολογικήα κατάτμησης σε μοντέλο ταξινόμησης μάσκας, όπου με αυτόν τον τρόπο το άρθρο [19] δείχνει ότι βελτιώνεται η επίδοση.

Η "μονάδα μετασχηματιστή αποκωδικοποίησης" δανείζεται από το μοντέλο DETR [58]. Προεπιλεγμένα, εφαρμόζονται 6 επίπεδα "μετασχηματιστών αποκωδικοποίησης" με 100 ερωτήσεις και η ίδια απώλεια (loss) εφαρμόζεται μετά από κάθε αποκωδικοποιητή.

Mask2Former: Προσοχή μάσκας

Σε μια ενημερωμένη έκδοση του MaskFormer, το Mask2Former [59], εισάγεται η έννοια της προσοχής μάσκας, αντικαθιστώντας 0.0.16 τη διασταυρούμενη-προσοχή (cross-attention) στον μετασχηματιστή αποκωδικοποίησης και καθιστώντας την αρχιτεκτονική κορυφαία (state of the art) στη σημασιολογική, πανοπτική και instance κατάτμηση. Η αργή διαδικασία εκπαίδευσης και η υψηλή πολυπλοκότητα του MaskFormer οδήγησαν στον σχεδιασμό του μηχανισμού προσοχής μάσκας, όπου οι υπολογισμοί της διασταυρούμενης-προσοχής περιορίζονται στην προβλεπόμενη περιοχή της μάσκας από το προηγούμενο επίπεδο αποκωδικοποιητή. Οι πληροφορίες των συμφραζόμενων (context) εξακολουθούν να είναι χρήσιμες για την κατηγοριοποίηση των μασκών, ωστόσο, η διασταυρούμενη-προσοχή απαιτεί μεγάλο μέρος του χρόνου εκπαίδευσης για να λειτουργήσει όπως αναμένεται. Έτσι, προτιμάται να περιοριστεί η προσοχή για να εστιάσει στα προτεινόμενα χαρακτηριστικά της μάσκας και διατηρείται το επίπεδο αυτό-προσοχής για να εισάγει πληροφορίες από τα συμφραζόμενα. Η σειρά αυτο-προσοχής και προσοχής μάσκας εναλάσσεται σε κάθε επίπεδο.

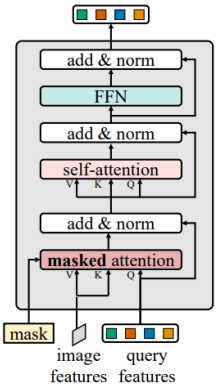


Figure 0.0.16: Mask2Former μετασχηματιστής αποκωδικοποίησης [59]

Στη διασταυρώμενη-προσοχή 0.0.3, χρησιμοποιούνται στους υπολογισμούς όλα τα χαρακτηριστικά των ερωτημάτων σε όλο το διάνυσμα εισόδου. Σε κάθε βήμα, παράγονται τα ενημερωμένα χαρακτηριστικά τους. Ο δεύτερος όρος αποτελεί την υπολειπόμενη σύνδεση των χαρακτηριστικών από το προηγούμενο επίπεδο στα νέα προβλεπόμενα χαρακτηριστικά. Ο πρώτος όρος αποτελεί το αποτέλεσμα softmax από την ενεργοποίηση των ερωτημάτων στα χαρακτηριστικά για τα δεδομένα κλειδιά. Η διαφορά της διασταυρώμενης προσοχής με την αυτό-προσοχή είναι οι είσοδοι στον μηχανισμό. Στην αυτο-προσοχή χρησιμοποιείται η ίδια είσοδος για τον υπολογισμό των Q,K,V αποτελεσμάτων ενώ στην διασταυρώμενη προσοχή χρησιμοποιείται μία είσοδος για τον υπολογισμό των K,V διανυσμάτων και μια διαφορετική για τον υπολογισμό του Q διανύσματος. Στην συγκεκριμένη περίπτωση η πρώτη είσοδος είναι τα χαρακτηριστικά που παράγονται από τον αποκωδικοποιητή ανά pixel σε διαφορετικά επίπεδα ανάλυσης και η δεύτερη τα εκπαιδεύσιμα query features.

$$\begin{aligned} X_l &= \text{softmax}(Q_l K_l^T) V_l + X_{l-1} \\ Q_l &= f_Q(X_{l-1}) \\ K_l &= f_K(I) \\ V_l &= f_V(I) \\ &\textit{where } I \textit{ are the image features.} \end{aligned} \tag{0.0.3}$$

Όπου l είναι το επίπεδο αποκωδικοποιητή, $X_l \in \mathbb{R}^{N \times C}$ τα N query features στο επίπεδο l, X_0 είναι τα αρχικά εκπαιδεύσιμα query features και $K_l, V_l \in \mathbb{R}^{H_l W_l \times C}$ όπου H_l, W_l η ανάλυση της εικόνας στο δεδομένο επίπεδο ανάλυσης.

Στη μάσκα-προσοχή (mask attention) 0.0.4, η διαφορά είναι ότι προστίθεται το M'_{l-1} στον όρο του softmax. Η προτεινόμενη μάσκα M_{l-1} είναι η ανακατασκευασμένη πρόβλεψη που έχει δυαδικοποιηθεί με κατώφλι 0.5. Το M'_{l-1} είναι $-\infty$ για τα εικοστοιχεία που δεν ανήκουν στην μάσκα και 0 για τα εικονοστοιχεία που ανήκουν στην μάσκα, όπου τα εικονοστοιχεία της μάσκας ανανεώνονται σε κάθε επίπεδο του μετασχηματιστή αποκωδικοποίησης. Επομένως, ουσιαστικά δεν χρησιμοποιούνται τα χαρακτηριστικά εκτός μάσκας στην προσοχή για τα οποία ισχύει το $X_l = X_{l-1}$. Σε αυτήν την έκδοση, τα χαρακτηριστικά ερωτήματος Q_0 είναι εκπαιδεύσιμα.

$$\begin{aligned} X_l &= \text{softmax}(\text{M'}_{l-1} + Q_l K_l^T) + X_{l-1} \\ \text{M'}_l(x,y) &= \begin{cases} 0 & \text{M}_l(x,y) = 1 \\ -\infty & else \end{cases} \end{aligned} \tag{0.0.4}$$

Ο Mask2Former εισάγει χαρακτηριστικά πολλαπλής ανάλυσης στον αποκωδικοποιητή μετασχηματισή όπως αναφέρθηκε παραπάνω. Συγκεκριμένα, παρουσιάζονται L συνεχόμενες ομάδες από 3 διαδοχικά επίπεδα του

αποκωδικοποιητή μετασχηματιστή, όπου κάθε επίπεδο λαμβάνει ως είσοδο ανάλυση $\frac{H}{32} \times \frac{W}{32}$, $\frac{H}{16} \times \frac{W}{16}$ και $\frac{H}{8} \times \frac{W}{8}$ αντίστοιχα. Αυτές οι αναλύσεις προέρχονται από διάφορα στάδια του αποκωδικοποιητή σε επίπεδο pixel. Ένα πολλαπλής κλίμακας επαναπροσαρμοζόμενο επίπεδο προσοχής(multi-scale deformable attention) [16] χρησιμοποιείται ως ανακτητής εικονοστοιχείων και προέρχεται από το DETR [58].

Άλλες τροποποιήσεις από τον MaskFormer περιλαμβάνουν: τον υπολογισμό της απώλειας μάσκας μόνο σε δείγματα της μάσκας που μειώνει σημαντικά τον χρόνο εκπαίδευσης, την αφαίρεση του dropout και την εισαγωγή εκπαιδεύσιμων query features.

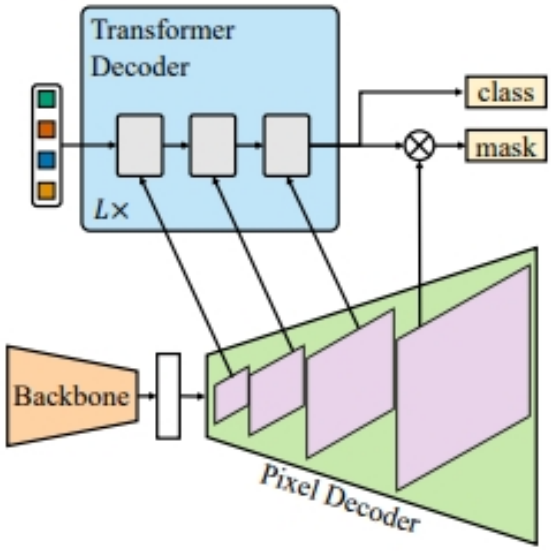


Figure 0.0.17: Mask2Former πολλαπλών αναλύσεων αρχιτεκτονική από αντίστοιχο άρθρο [59]

Πρόταση

Πολλαπλές αναλύσεις στην σημασιολογική κατάτμηση

Απλότερες εργασίες, όπως η ταξινόμηση αντικειμένων, εκτελούνταν με επιτυχία με αρχιτεκτονικές τύπου κωδικοποιητή-αποκωδικοποιητή. Αυτό ήταν δυνατό λόγω της φύσης τόσο της εργασίας όσο και της αρχιτεκτονικής που μετέτρεπε την είσοδο σε μικρότερη διάσταση. Ωστόσο, μετά την καθορισμένη των πιο πολύπλοκων εργασιών, όπως ο σημασιολογικός εντοπισμός, όπου η διάσταση της εισόδου είναι ίση με τη διάσταση της εξόδου, προέκυψε η ανάγκη για πιο πολύπλοκες αρχιτεκτονικές. Ειδικότερα, απαιτείται να επεξεργαστούμε δεδομένα σε πολλαπλές κλίμακες και να ανιχνεύσουμε κατηγορίες σε πολλαπλές κλίμακες. Αυτή η παρατήρηση οδήγησε τελικά στην εισαγωγή πολυ-αναλυτικών μοντέλων, όπως τα HRNet [56], HRFormer [60], HRSTFormer [61] τα οποία παρατηρήθηκε ότι αποδίδουν καλύτερα από τα αντίστοιχα μονο-αναλυτικά μοντέλα. Αυτά τα μοντέλα ακολουθούν τη βασική ιδέα που εισήχθη από το HRNet, όπως φαίνεται στο σχήμα 0.0.18. Υπάρχουν τέσσερες ροές ανάλυσης που προστίθενται στο μοντέλο σταδιακά μετά από κάθε στάδιο. Σε κάθε ροή, η ανάλυση διατηρείται σταθερή κατά τη διάρκεια των σταδίων επεξεργασίας της εισόδου. Κάθε στάδιο ακολουθείται από μια μονάδα σύγκλισης που ανταλλάσσει πληροφορίες μεταξύ των διαφορετικών αναλύσεων.

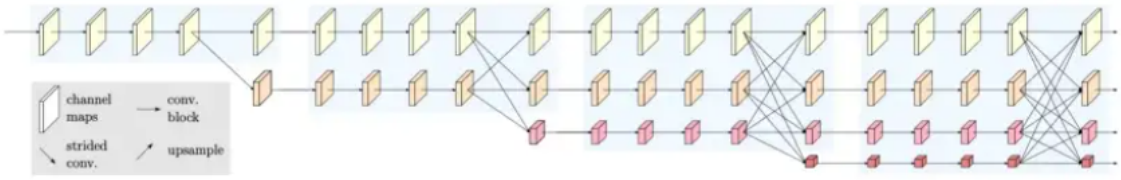


Figure 0.0.18: HRNet Architecture from [50]

Εισαγωγή πολλαπλών αναλύσεων στον Mask2Former

Το MaskFormer [19] προβλέπει ένα σύνολο N ζευγάρια ταξινόμησης και μάσκας, $z = (c_i, m_i)_{i=1,\dots,N}$ εκτελώντας καταρχάς ταξινόμηση μάσκας και στη συνέχεια μετατρέπει τα αποτελέσματα σε πανοπτική, περίπτωση ή σημασιολογική ταξινόμηση εξόδου. Όπως φαίνεται στο σχήμα 0.0.19, αποτελείται από ένα κύριο τμήμα, έναν αποκωδικοποιητή μετασχηματιστή και ένα κεφάλι σημασιολογικής επιμέρους επεξεργασίας. Ο κύριος τμήματος μπορεί να αντικατασταθεί από οποιαδήποτε αρχιτεκτονική, δεδομένου ότι ο βασικός στόχος του Mask2Former ήταν η μετατροπή διαφορετικών αρχιτεκτονικών τομής σε μια πολυσκοπική αρχιτεκτονική τομής βασισμένη σε μάσκες. Ωστόσο, το Mask2Former έχει χρησιμοποιηθεί προς το παρόν μόνο σε αρχιτεκτονικές μονο-ανάλυσης, όπως R-50, R101, Swin-T, Swin-S, SWin-B, Swin-L στο σύνολο δεδομένων Cityscapes, που είναι το σύνολο δεδομένων που εξετάζουμε.

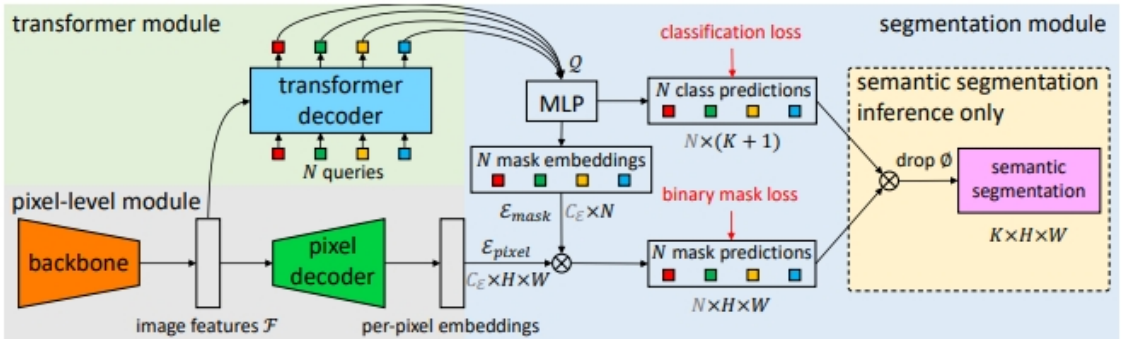


Figure 0.0.19: MaskFormer Architecture from corresponding paper [19]

Το Mask2Former με βάση το Swin Transformer (Swin-T) διαθέτει μια αρχιτεκτονική όπως αυτή που φαίνεται λεπτομερώς στο 0.0.20. Συνδυάζεται με το ενοποιημένο πολλαπλών κλιμάκων υποαλλοίωσης ενσωματωμένο ως αποκωδικοποιητή. Η προτεινόμενη αρχιτεκτονική βάσης, SwinHR, ακολουθεί τη δομή τεσσάρων επιπέδων της αρχικής αρχιτεκτονικής βάσης που περιγράφηκε στην subsection 3.3.2 αναπαράγοντας κάθε επίπεδο σε παράλληλες ροές. Για να διατηρηθεί σταθερή η ανάλυση, οι διακριτικοί μετασχηματισμοί εξαλείφονται στα επίπεδα. Επιπλέον, μετά από κάθε στάδιο, μια ροή εισάγεται με τη διακριτική μείωση της ανώτερης ροής ανάλυσης. Η μείωση ανάλυσης πραγματοποιείται με τη χρήση μονάδων συγχώνευσης πάτσων όπως εξηγείται στην section 3.3.1. Συνολικά, υπάρχουν τέσσερα στάδια, τα οποία είναι ισοδύναμα με τον αριθμό των ροών. Η τελική αρχιτεκτονική φαίνεται στο 0.0.21 και 0.0.22, τα οποία διαφέρουν στο σημείο από όπου λαμβάνεται το εισερχόμενο στοιχείο του αποκωδικοποιητή εικόνων.

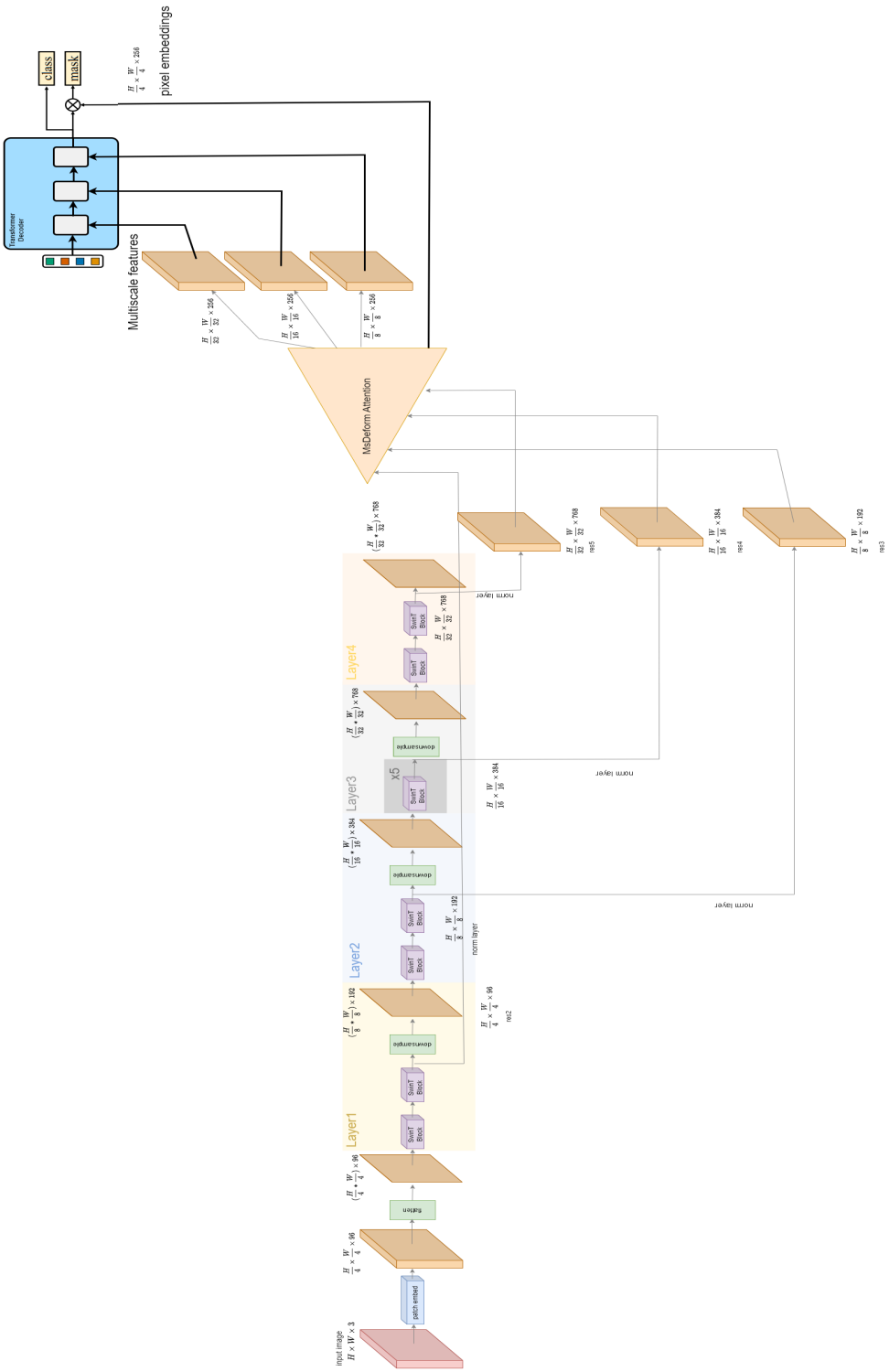


Figure 0.0.20: Mask2Former Architecture

Στην αρχική αρχιτεκτονική του Mask2former-Swin Transformer, το εισερχόμενο στοιχείο του αποκωδικοποιητή μετασχηματιστή προέρχεται από την έξοδο του αποκωδικοποιητή εικόνων. Ωστόσο, η ιδέα πίσω από τη μεταφορά αυτού του εισερχόμενου στο νέο αρχιτεκτονικό βασίζεται στην αυτόνομη απόδοση του πολυδιάστατου μοντέλου. Αυτό το νέο μοντέλο της βάσης δεν απαιτεί αποκωδικοποιητή, καθώς αντικαθιστά ολόκληρο

τον κωδικοποιητή-αποκωδικοποιητή. Ωστόσο, ο αποκωδικοποιητής διατηρείται αναλλοίωτος, καθώς θα αποκωδικοδικοποιήσει τα στοιχεία ανά pixel που θα συγχωνευτούν με την έξοδο του αποκωδικοποιητή εικόνων. Έτσι, το αναδιαμορφούμενο πολυδιάστατο επίπεδο προσαρμοστικής προσοχής διατηρείται στο νέο μοντέλο.

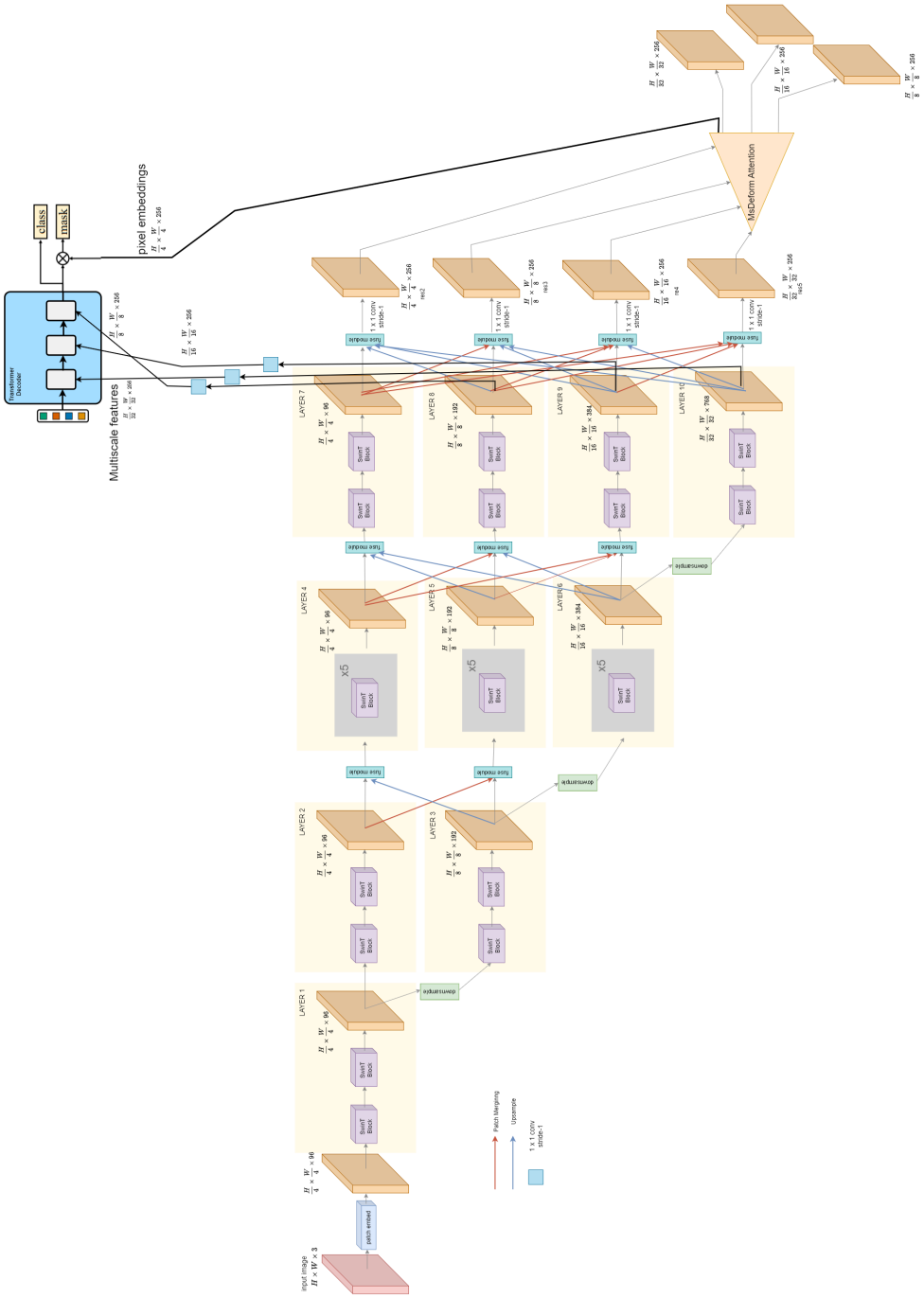


Figure 0.0.21: Mask2Former Multiple Resolutions Architecture

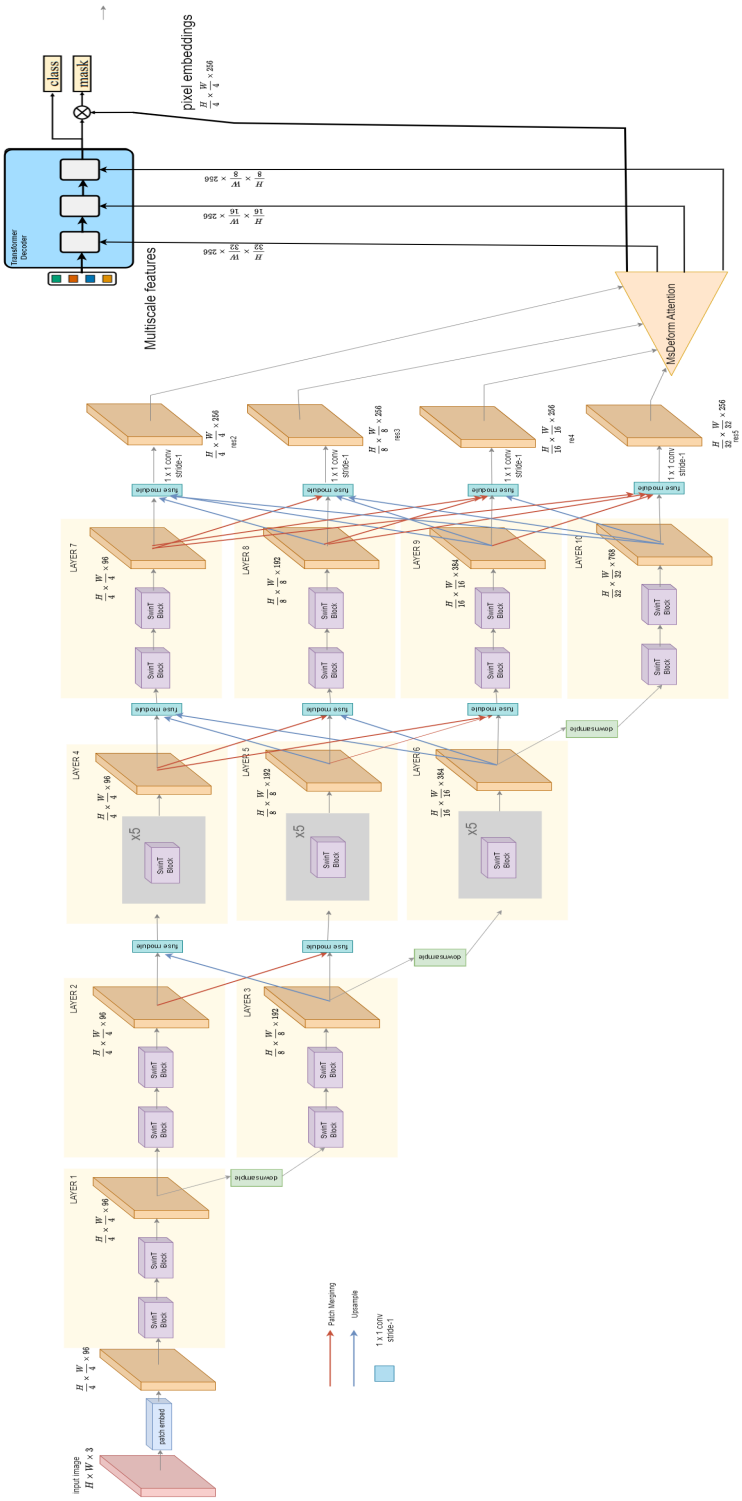


Figure 0.0.22: Mask2Former Multiple Resolutions Architecture alternate design : Transformer decoder input comes from the output of the pixel decoder

Swin Transformer

Το Swin Transformer προσπαθεί να μειώσει τον χρόνο επεξεργασίας ενός κανονικού μετασχηματιστή όταν υπολογίζει την αυτο-προσοχή σε μια εικόνα. Αυτό ήταν δυνατόν διαιρώντας τις εικόνες σε παράθυρα και υπ-

ολογίζοντας την αυτο-προσοχή σε κάθε παράθυρο ξεχωριστά, γνωστή ως αυτο-προσοχή σε παράθυρα. Ωστόσο, προκειμένου να διασχίσουμε τα όρια που καθορίζονται από αυτά τα παράθυρα στα γειτονικά pixel, σε κάθε διαδοχικό επίπεδο Swin Transformer, τα παράθυρα αυτά μετακινούνται και ανασχηματίζονται. Το αρχικό παράθυρο τοποθετείται στην αριστερή-επάνω γωνία και τα υπόλοιπα τοποθετούνται συνεχώς σε μια δίπλα, μη-επικαλυπτόμενη διάταξη. Η επόμενη διάταξη δημιουργείται με τη μετατόπιση των παραθύρων διαγώνια. Ωστόσο, αυτό έχει ως αποτέλεσμα τα αρχικά παράθυρα να αυξηθούν και να υπάρχουν αρκετά ανεξάρτητα παράθυρα. Η λύση που δίνεται σε αυτό το πρόβλημα είναι η μεθοδολογία της κυκλικής μετατόπισης, όπου τα ανεξάρτητα παράθυρα από την επάνω αριστερή πλευρά μετακινούνται προς την κάτω δεξιά γωνία και συνδέονται με τα ανεξάρτητα παράθυρα εκεί για να δημιουργηθούν κανονικά παράθυρα. Έπειτα, η αυτο-προσοχή που εφαρμόζεται σε αυτά τα συντεταγμένα παράθυρα λαμβάνει υπόψη αυτήν τη μετατόπιση με τον τρόπο που μάσκαρε τα χαρακτηριστικά στο παράθυρο που δεν ήταν γειτονικά πριν από τη μετατόπιση. Όπως φαίνεται στο 0.0.23, δύο διαδοχικά τμήματα του Swin Transformer συνδέονται μεταξύ τους και το δεύτερο εφαρμόζει τα μετακινημένα παράθυρα του πρώτου. Επίσης, σε κάθε τμήμα υπάρχει μια υπολειτουργία σύνδεσης (residual connection) και ένα πολυεπίπεδο perceptron (MLP) μετά τη μονάδα προσοχής. Πριν από κάθε τμήμα εφαρμόζεται προς τα εμπρός η κανονικοποίηση επιπέδου (layer normalization). Η διαδιαδικασία της πέρασης μιας εικόνας εισόδου z^{l-1} μέσω δύο διαδοχικών τμημάτων μετασχηματιστή Swin περιγράφεται ως εξής:

$$\begin{aligned} \hat{z}^l &= W - MSA(LN(z^{l-1})) + z^{l-1} \\ z^l &= MPL(LN(\hat{z^l})) + \hat{z^l} \\ z_{\hat{l+1}} &= SW - MSA(LN(z^l)) + z^l \\ z^{l+1} &= MLP(LN(z_{\hat{l+1}})) + z_{\hat{l+1}} \end{aligned} \tag{0.0.1}$$

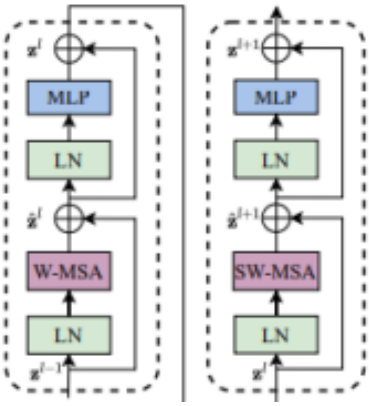


Figure 0.0.23: Two consecutive Swin Transformer modules

Αποτελέσματα

Σύνολα Δεδομένων

Σύνολο Δεδομένων Cityscapes

Το σύνολο δεδομένων Cityscapes [8] είναι ένα σύνολο δεδομένων που δημιουργήθηκε ειδικά για εφαρμογές αυτόνομης οδήγησης σε αστικά περιβάλλοντα. Περιλαμβάνει εικόνες τοπίων δρόμου από 50 διαφορετικά κέντρα πόλεων κατά τη διάρκεια όλων των 4 εποχών. Αυτές οι εικόνες είναι διαθέσιμες σε χαμηλή ανάλυση (8 bit) και υψηλή ανάλυση (16 bit). Το σύνολο δεδομένων περιλαμβάνει τόσο αναλυτικές επιπέδου pixel όσο και αναλυτικές επιπέδου περίπτωσης (instance-level) επισημάνσεις που παρέχονται σε δύο ομάδες - χονδροειδείς και

λεπτομερείς επισημάνσεις. Οι λεπτομερείς επισημάνσεις παρέχονται σε 5000 εικόνες από 27 πόλεις, όπου όλα τα pixels έχουν επισημανθεί με τη δημιουργία πολυγώνων. Οι υπόλοιπες εικόνες από 23 πόλεις χρησιμοποιούνται για χονδροειδείς επισημάνσεις, όπου τα πολύγωνα επιλέγονται πολύ γρηγορότερα, μειώνοντας την ακρίβεια της επισήμανσης. Συνολικά περιλαμβάνονται 30 κατηγορίες, και 19 από αυτές χρησιμοποιούνται για αξιολόγηση, όπως φαίνεται στην εικόνα 0.0.24.

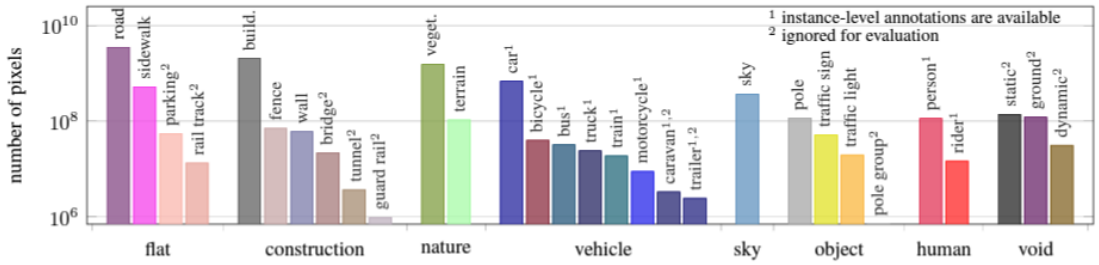


Figure 0.0.24: Κατηγορίες του Cityscapes, εικόνα από το [19]

Αυτό το σύνολο δεδομένων απεικονίζει μια πολύ ευρύτερη αντιπροσώπευση των εσωτερικών οδών της πόλης, λαμβάνοντας υπόψη την κυκλοφορία και τις διάφορες κλιματικές συνθήκες, με μια μεγάλη ποικιλία κατηγοριών, κάνοντας το μοναδικό στον τομέα της αυτόνομης οδήγησης. Άλλα σύνολα δεδομένων, όπως το Kitti [62], παρέχουν εικόνες από οδούς αλλά από προάστια, ενώ το CamVid [63] και το DUS[64] παρέχουν λιγότερες εικόνες από μόνο μια πόλη. Επιπλέον, όσον αφορά τις αναλυτικές επισημάνσεις επιπέδου αντικειμένου, το Cityscapes είναι το μόνο σύνολο δεδομένων ανάμεσα σε αυτά που παρέχει περιπτώσεις ανθρώπων και οχημάτων. Το μόνο σύνολο δεδομένων που είναι πιο πλήρες από το Cityscapes είναι το Mapillary Vistas Dataset [65], ωστόσο, είναι πολύ πιο πλούσιο από ό,τι χρειάζεται για αυτό το έργο, καθώς είναι πέντε φορές μεγαλύτερο από το σύνολο δεδομένων του Cityscapes, περιλαμβάνοντας 66 κατηγορίες με εικόνες που έχουν ληφθεί από όλο τον κόσμο με διάφορες μεθόδους λήψης.

Σύνολο Δεδομένων ADE20K

Το σύνολο δεδομένων ADE20K δημιουργήθηκε μετά την αναγνώριση της ανάγκης για ένα γενικό σύνολο δεδομένων που καλύπτει μια ποικιλία σκηνών και κοινών αντικειμένων. Τα σύνολα δεδομένων πριν από το ADE20K είχαν ένα περιορισμένο σύνολο σκηνών, όπως το Cityscapes [8], ή κάλυπταν λίγα ή ασήμαντα αντικείμενα, όπως το COCO[10] και το Pascal [15]. Στο ADE20K υπάρχουν 20,210 εικόνες στο σύνολο εκπαίδευσης, 2,000 εικόνες στο σύνολο επικύρωσης και 3,000 εικόνες στο σύνολο ελέγχου. Όλες οι εικόνες έχουν αναλυτικές επισημάνσεις αντικειμένων. Πολλά αντικείμενα έχουν επίσης αναλυτικές επισημάνσεις των μερών τους, όπως φαίνεται στην εικόνα 0.0.3. Για κάθε αντικείμενο υπάρχει επιπρόσθετη πληροφορία σχετικά με το αν είναι εμποδισμένο ή κομμένο και άλλα χαρακτηριστικά. Οι εικόνες στο σύνολο επικύρωσης έχουν αναλυτικές επισημάνσεις των μερών, ενώ οι επισημάνσεις των μερών δεν είναι αναλυτικές στις εικόνες του συνόλου εκπαίδευσης. Επιπλέον, τα μέρη μπορεί να έχουν επίσης μέρη, και αυτά επισημαίνονται επίσης με συσχετίσεις.

Μετρικές Επίδοσης

Για τη σημασιολογική εξαγωγή, η τυπική μετρική είναι η mIoU (μέση Τιμή Intersection-over-Union) [36], μια μετρική ανά pixel που αντιστοιχεί απευθείας στη διατύπωση ανά pixel ταξινόμησης. Χρησιμοποιούμε το υψηλότερο επιτευχθέν αποτέλεσμα mIoU ως τη μετρική απόδοσης ενός μοντέλου. Επιπρόσθετα, για το Cityscapes χρησιμοποιείται το iIoU (διαστασιακή επικάλυψη συνδιακύμανσης) επιπέδου παραδείγματος. Για το ADE20k χρησιμοποιούνται επίσης το fwIoU (συχνότητα ζυγίου επικάλυψης συνδιακύμανσης), το mACC (μέση ακρίβεια) και το pACC (πιθανοτική ακρίβεια). Αυτά καθορίζονται όλα στην .

Baseline

Η υλοποίησή μας βασίζεται στον πηγαίο κώδικα του Mask2former [66], ο οποίος χρησιμοποιεί το Detectron2 [67]. Οι πειραματισμοί που περιγράφονται παρακάτω επικεντρώνονται στα σύνολα δεδομένων Cityscapes [8] και

ADE20k [16].

Ρυθμίσεις εκπαίδευσης

Η επίσημη υλοποίηση χρησιμοποίησε 8 μονάδες GPU και μέγεθος παρτίδας 16. Ωστόσο, λόγω περιορισμένης ισχύος επεξεργασίας, στην υλοποίησή μας χρησιμοποιήσαμε 4 μονάδες GPU και μέγεθος παρτίδας 4. Η αρχική ρυθμίση της εκπαίδευσης ήταν με ρυθμό μάθησης 0.0001 και αποσύνθεση βάρους 0.05. Το υπο-μοντέλο (backbone) αρχικοποιήθηκε με βάρη από την ταξινόμηση εικόνας του Swin-T [57] στο ImageNet-22k [9] και έχει ρυθμιστεί πολλαπλασιαστικός ρυθμός μάθησης του 0.1. Ως βελτιστοποιητής χρησιμοποιήθηκε ο AdamW [68], και ως χρονοδιάγραμμα εκπαίδευσης χρησιμοποιήθηκε το πολυωνυμικό [69]. Λόγω της περιορισμένης ισχύος επεξεργασίας, σε αυτό το πείραμα εκπαιδεύσαμε το μοντέλο στο Cityscapes για 180.000 επαναλήψεις, ενώ στην επίσημη υλοποίηση εκπαιδεύτηκε για 90.000 επαναλήψεις. Για το σύνολο δεδομένων ADE20k, και οι δύο υλοποιήσεις εκπαιδεύτηκαν για 160.000 επαναλήψεις.

Σύγκριση αποτελεσμάτων

Εμφανίζονται στον πίνακα 3 και στον πίνακα 4 τα αποτελέσματα που επιτεύχθηκαν με την εκπαίδευση του μοντέλου Mask2Former με τους διαθέσιμους πόρους μας και τα επίσημα αποτελέσματα. Η διαφορά στην απόδοση βρίσκεται εντός λογικών ορίων για τη διαφορά στην ισχύ επεξεργασίας.

	Cityscapes			
Model	$Official(mIoU)$	$Official(iterations)$	$Ours(mIoU)$	$Ours(iterations)$
Mask2former (Swin-T)	82.3	90k	81.7	180k
Mask2former (Swin-B)	83.3	90k	83.18	180k

Table 3: Performance of Mask2Former on Cityscapes with official and replicated implementation

	ADE20k			
Model	$Official(mIoU)$	$Official(iterations)$	$Ours(mIoU)$	$Ours(iterations)$
Mask2former (Swin-T)	47.7	160k	47.2	160k

Table 4: Performance of Mask2Former on ADE20k with official and replicated implementation

Πειραματισμός με SwinHR backbone

Χωρίς αρχικοποίηση

Η αρχιτεκτονική του Mask2FormerHR, που περιγράφεται στο , χρησιμοποιώντας το SwinHR ως βάση, υλοποιήθηκε. Αρχικά, κατά την εκπαίδευση χωρίς αρχικοποίηση των βαρών, τα αποτελέσματα δεν ανταποκρινόταν στην αρχική αρχιτεκτονική που αρχικοποιήθηκε, όπως φαίνεται στο 0.0.25. Δεδομένου ότι η βάση δεν αρχικοποιήθηκε στην αρχιτεκτονική πολλαπλών αναλύσεων, ο πολλαπλασιαστής της βάσης αλλάζει σε 1, ώστε η βάση να εκπαιδεύεται με τον ίδιο ρυθμό με το υπόλοιπο μοντέλο.

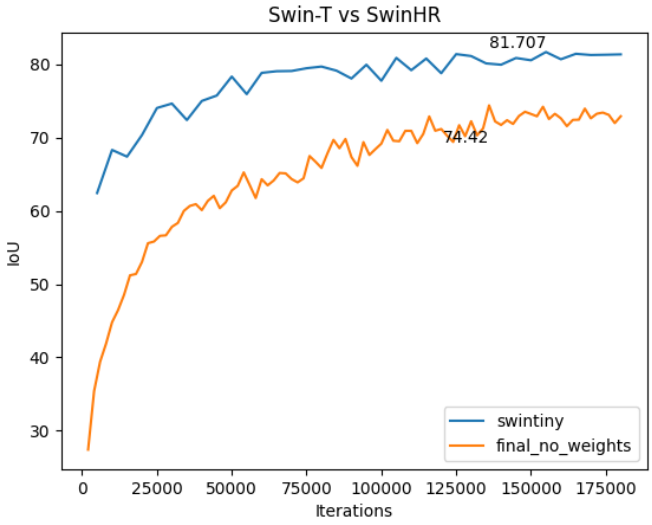


Figure 0.0.25: Mask2Former with Swin-T initialized and SwinHR uninitialized backbone

Τεχνικές αρχικοποίησης

Μεταφέροντας Swin-T βάρη ταξινόμησης Δεδομένου ότι οι δύο βάσεις, η Swin-T και η SwinHR, είναι πολύ παρόμοιες, προσπαθούμε να μεταφέρουμε τα εκπαιδευμένα βάρη της Swin-T από το ImageNet. Αυτό μπορεί να γίνει παρατηρώντας ότι τα διαγώνια μέρη των SwinHR, όπως φαίνεται στο 0.0.26, είναι παρόμοια με αυτά της Swin-T. Ένας προτεινόμενος τρόπος μεταφοράς είναι να αρχικοποιήσουμε μόνο τα διαγώνια μέρη του μοντέλου Swin-HR και να αφήσουμε τα υπόλοιπα μέρη της βάσης ανεκπαιδεύτα.

Πειραματιστήκαμε με διάφορους πολλαπλασιαστές του learning rate ώστε να παγώσουμε ή να κρατήσουμε την εκπαίδευση των αρχικοποιημένων μοντέλων. Αρχικά, στο 0.0.27, τα αρχικοποιημένα μέρη ανατέθηκαν έναν πολλαπλασιαστή εκμάθησης 0.1 και το εκπαιδευμένο μοντέλο ταίριαζε σχεδόν με την απόδοση του επίσημου μοντέλου. Στη συνέχεια, στο 0.0.28, εφαρμόστηκε ένας πολλαπλασιαστής ρυθμού μάθησης 0,001 στα αρχικοποιημένα μέρη και 0,1 στην υπόλοιπη βάση. Τέλος, στο 0.0.29, δεν εφαρμόστηκε πολλαπλασιαστής ρυθμού μάθησης σε κανένα από τα μέρη και αυτά παραμένουν να εκπαιδευτούν με τον ίδιο ρυθμό με το υπόλοιπο μοντέλο. Παρατηρούμε ότι εάν αφήσουμε το μοντέλο να εκπαιδευτεί ομοιόμορφα, τότε έχουμε ένα καλύτερο αποτέλεσμα που ταιριάζει με την επίσημη υλοποίηση. Ωστόσο, η απόδοση του μοντέλου στις πρώτες επαναλήψεις είναι πολύ χειρότερη από την αρχική απόδοση του επίσημου μοντέλου. Έτσι, καταλήγουμε στο συμπέρασμα ότι μια άλλη αρχικοποίηση μπορεί να είναι πιο κατάλληλη.

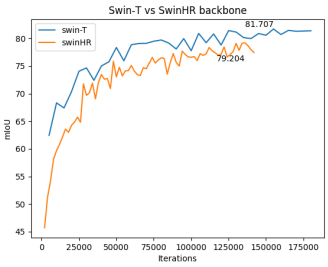


Figure 0.0.28: Weight transfer from Swin-T to SwinHR. No learning rate multiplier is applied.

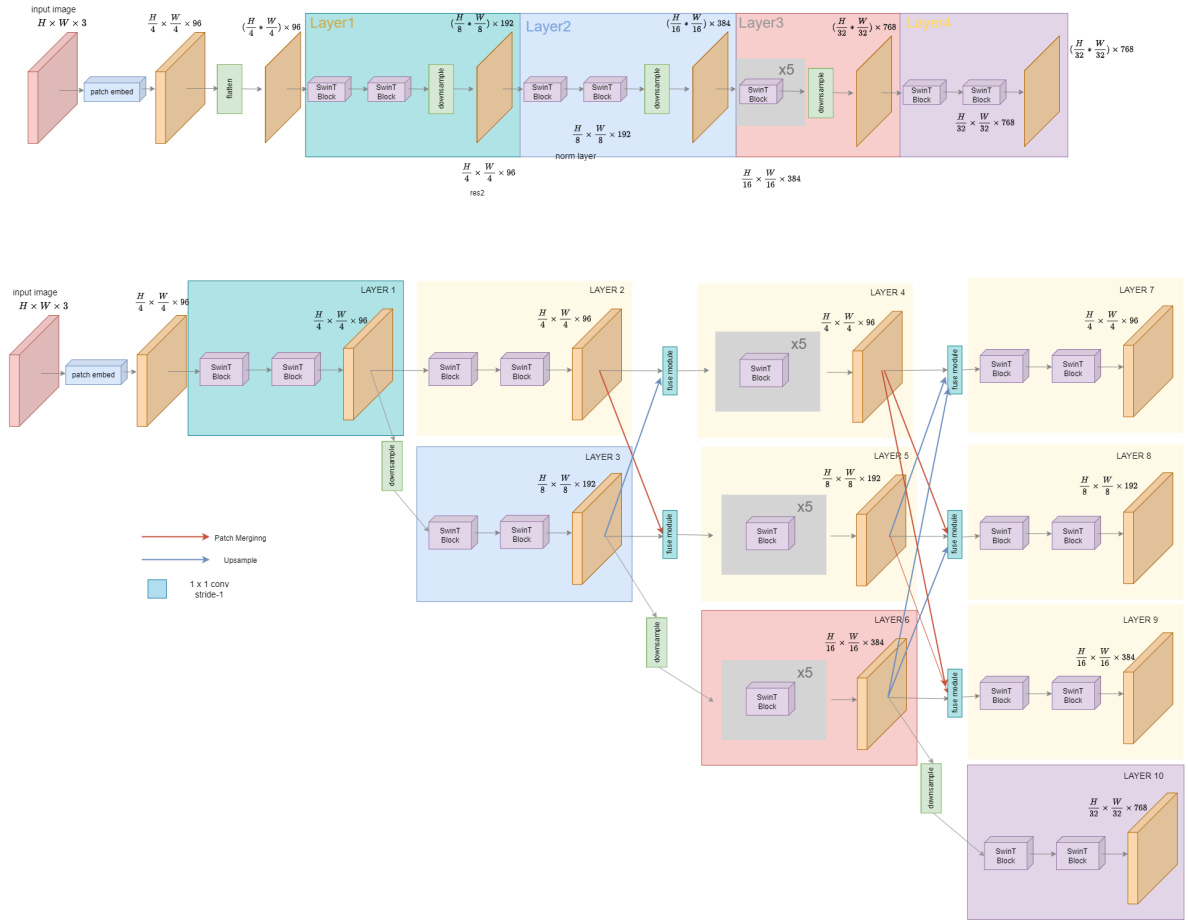


Figure 0.0.26: Weight transfer from Swin-T(top) to SwinHR(bottom). Module correspondence is shown with colored rectangles.

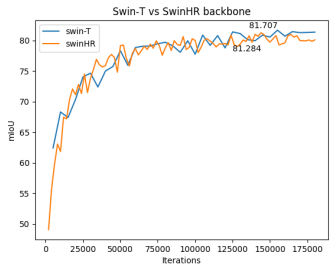


Figure 0.0.27: Weight transfer from Swin-T to SwinHR. Learning rate multiplier of 0.001 for initialized modules and 0.1 for the rest of the backbone.

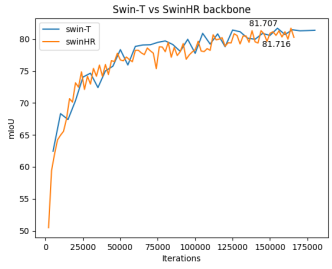


Figure 0.0.29: Weight transfer from Swin-T to SwinHR. No learning rate multiplier is applied.

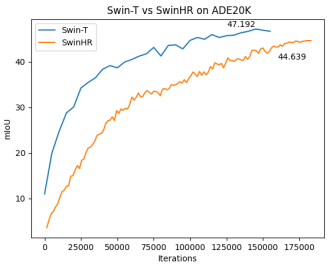


Figure 0.0.30: Weight transfer from Swin-T to SwinHR trained on ADE20k. No learning rate multiplier is applied.

Μεταφέροντας Swin-T βάρη κατάτμησης: Cityscapes σύνολο δεδομένων Παρατηρούμε ότι εάν μπορούσαμε να πάρουμε ως έξοδο από τη βάση τα ίδια αποτελέσματα που παράγει το Swin-T, τότε θα μπορούσαμε να ξεκινήσουμε με την ίδια απόδοση με αυτή του αρχικού μοντέλου. Αυτό θα επέτρεπε στο μοντέλο να επιτύχει τον μέγιστο μέσο όρο IoU (mIoU) του αρχικού μοντέλου και ενδεχομένως να τον υπερβεί. Δεδομένου ότι η βάση Swin-T παράγει ως έξοδο την έξοδο κάθε διαγώνιου ενότητας του μοντέλου SwinHR, επιχειρούμε να βρούμε μια αρχικοποίηση βαρών που θα μεταφέρει την έξοδο της πρώτης μονάδας κάθε ροής αναλλοίωτη στο τέλος της ροής. Αυτό είναι δυνατό εάν όλα τα βάρη των υπολοίπων μονάδων του μετασχηματιστή τίθενται σε μηδέν και, συνεπώς, η είσοδος είναι ίση με την έξοδο. Επίσης, οι στρώσεις συγχώνευσης (fuse layers) πρέπει να προσθέτουν μηδενισμένους χάρτες από άλλες ροές στο σημείο συγχώνευσης της ροής. Αυτό είναι δυνατό απλώς αναθέτοντας μηδενισμένα βάρη στις στρώσεις συγχώνευσης και στις μονάδες εκτός της διαγωνίου. Επίσης, ο αρχικός ρυθμός μάθησης πρέπει να ταιριάζει με τον ρυθμό μάθησης των τελευταίων επαναλήψεων κατά τη διάρκεια της εκπαίδευσης του επίσημου μοντέλου. Διαφορετικά, η απόδοση θα αποκλίνει πολύ εύκολα και δεν θα παράγει το επιθυμητό αποτέλεσμα. Οι αρχικοποιημένες στρώσεις της βάσης έχουν έναν πολλαπλασιαστή ρυθμού μάθησης 0,01, ενώ ο υπόλοιπος της βάσης και η κεφαλή σημασιολογικής σεγμεντάτιον έχουν πολλαπλασιαστή ρυθμού μάθησης 1.

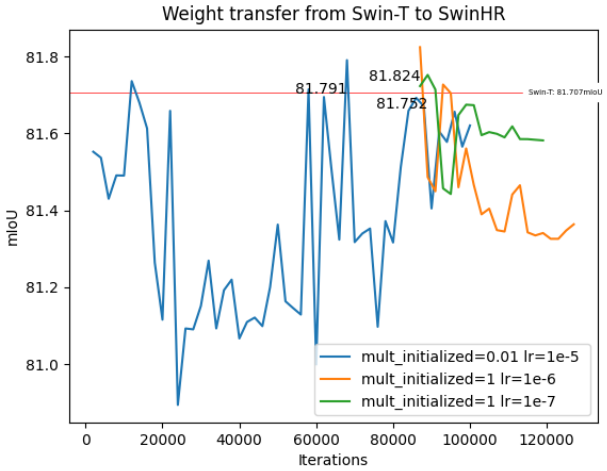


Figure 0.0.31: Μεταφορά βαρών από το Swin-T εκπαιδευμένο στο Cityscapes στο SwinHR με μηδενισμένα στοιχεία.

Αυτή η τεχνική επέτρεψε στο πολυδιάστατο μοντέλο να ξεκινήσει όπως υποψιαζόμασταν κοντά στη μέγιστη απόδοση του μοντέλου μονής ανάλυσης και να την υπερβεί κατά 0,1 mIoU.

Παρατηρούμε ότι τα μηδενισμένα βάρη δεν επιτρέπουν στο δίκτυο να μάθει, καθώς κολλάνε στο μηδέν, ειδικά στις στρώσεις συγχώνευσης που περιέχουν ReLUs. Για το λόγο αυτό, ανοίγουμε τις στρώσεις συγχώνευσης από την αρχή αναθέτοντας βάρη ίσα με 1, ώστε να υπάρχει συγχώνευση. Επίσης, με ενστικτώδη τρόπο αναθέτουμε βάρη ίσα με 1 στα αρχικοποιημένα βάρη με σκοπό τα ίδια αποτελέσματα αλλά με καλύτερες πιθανότητες μάθησης.

Τέλος, καταψύχουμε τόσο τα αρχικοποιημένα βάρη της βάσης όσο και τα αρχικοποιημένα βάρη της κεφαλής σημασιολογικής σεγμεντάτιον, ενώ η υπόλοιπη της βάσης έχει πολλαπλασιαστή ρυθμού μάθησης 1. Η εκπαίδευση γίνεται σε δύο φάσεις:

1. Χρησιμοποιώντας σταθερό ρυθμό μάθησης 10^{-4}
2. Χρησιμοποιώντας σταθερό ρυθμό μάθησης 10^{-5}

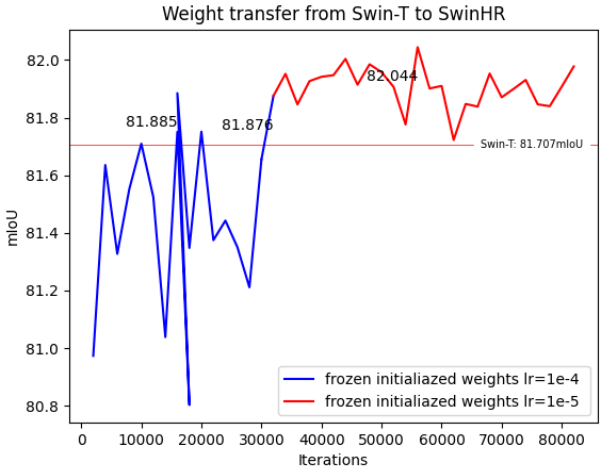


Figure 0.0.32: Μεταφορά βαρών από το Swin-T εκπαιδευμένο στο Cityscapes στο SwinHR με μοανδιαία στοιχεία.

Αυτή η τεχνική επέτρεψε στο πολυδιάστατο μοντέλο να ξεκινήσει όπως υποψιαζόμασταν γύρω από τη μέγιστη απόδοση του μοντέλου μονής ανάλυσης και να την υπερβεί κατά 0,3 mIoU.

Μεταφέροντας Swin-T βάρη κατάτμησης: ADE20k σύνολο δεδομένων

ADE20k Χρησιμοποιώντας τις παραπάνω τεχνικές, αυτό το σύνολο δεδομένων δείχνει να υπερχειλίζει. Επιδιώχθηκε να γίνει μια πιο προσεκτική προσέγγιση. Τα μη αρχικοποιημένα βάρη αρχικοποιούνται ξεχωριστά με βάση τη λειτουργικότητά τους. Αρχικά, τα επίπεδα συγχώνευσης περιέχουν επίπεδα συνέλιξης, κανονικοποίησης και ανάλογα με την περίπτωση υπερδειγματοληψίας. Τα βάρη της συνέλιξης μειώθηκαν στο 0.1 καθώς παρατηρήθηκε ότι προκαλούσαν υπερχείλιση μετά από κάποιες επαναλήψεις. Το επίπεδο κανονικοποίησης αρχικοποιήθηκε σε μηδενικό μέσο όρο και μηδενική διακύμανση.Στις μονάδες μετασχηματιστών υπάρχουν στοιχεία MLP που περιέχουν γραμμικά επίπεδα της μορφής:

$$y = xA^T + b \tag{0.0.1}$$

Οπου A είναι το διάνυσμα βάρους και b το διάνυσμα bias. Επομένως, αρχικοποιούμε το A ως μοναδιαίο πίνακα και b ως ένα πίνακα μηδενικών. Επιπρόσθετα, για τον μηχανισμό προσοχής αρχικοποιούμε τα Q,K,V ως μοναδιαίο πίνακα και μηδενικό bias. Ομοίως, για κάθε επίπεδο κανονικοποίησης αρχικοποιούμε τα βάρη σε μοναδιαίους πίνακες και το bias σε μηδενικούς πίνακες. Το επόμενο γράφημα δείχνει την εκπαίδευση του μοντέλου με παγωμένα όλα τα επίπεδα των οποίων τα βάρη προέρχονται από το Swin-T. Τέλος, χρησιμοποιείται σταθερός ρυθμός μάθησης τιμής 0.0001.

Figure 0.0.33: Μεταφορά βαρών από το Swin-T εκπαιδευμένο στο Ade20k στο SwinHR με παγωμένα στοιχεία.

Το SwinHR ξεπερνάει την επίδοση του Swin-T κατά 0.23mIoU.

Chapter 1

Introduction

1.1 Image Segmentation Categories

Image segmentation according to [1] is the partition of an image into a set of non-overlapping regions whose union is the entire image. These resulting regions should :

- Be homogeneous in regards to some characteristic
- Be simple without any small holes
- Have significantly different values in regards to the characteristic they are uniform with adjacent regions.
- Have boundaries that are smooth and not have many holes

The above goals initially were attempted to be met with various image processing techniques. One of them is clustering [2] [3] [4] used with detection of edges and contours and further evolved with modelling using the Markov process [5]. Other approaches use histograms such as HOG [6] and SIFT [7] feature extraction which are orientation histograms. However, after the introduction of convolutional neural networks image segmentation took a turn into supervised learning. Datasets have grown extensively - Cityscapes [8], ImageNet [9], COCO [10]- and the generalization requirements from the model predictions make it reasonable to adopt a more adaptable and complex approach. Neural Networks are capable of learning patterns and deep neural networks have numerous parameters that can imitate complex functions. Image segmentation is more demanding than image classification as it needs to detect relationships across various scales, thus, it requires more sophisticated structures that take into account both semantics and location.

1.1.1 Semantic Image Segmentation

The goal of this category which is also the focus of this work, is to segment the input image according to semantic information and predict the semantic category of each pixel from a given label set. It does not distinguish between objects of the same class and rather groups them together. According to Adelson [11], semantic segmentaton was designed to recognize stuff which are formless regions of similar texture or material. Semantic image segmentation has a variety of applications in real-world problems. It is widely used in medicine, such as brain and tumor detection [12] and discovering and tracking medical devices in surgery [13]. Other applications include autonomous driving [14] where a car is able to navigate in its environment. Semantic segmentation is crucial in the context of autonomous driving applications, where what semantic segmentation has to do is not only to classify the content in the image but also to mark the location of the target object in the actual scene.

Some example benchmarks for this task are Cityscapes [8], PASCAL VOC [15], and ADE20K [16].

1.1.2 Instance Image Segmentation

Instance Segmentation is a computer vision task that involves identifying and separating individual objects within an image, including detecting the boundaries of each object and assigning a unique label to each object. The goal of instance segmentation is to produce a pixel-wise segmentation map of the image, where each pixel is assigned to a specific object instance. Thus, it is very similar to semantic segmentation except in this task object instances are detected seperately. In contrast to semantic segmentation, instance segmentation studies things. While seemingly related, the datasets, details, and metrics for these two visual recognition tasks vary substantially.

Some example benchmarks for this task are Cityscapes [8], COCO[10], and ADE20K [16].

1.1.3 Panoptic Image Segmentation

Panoptic segmentation [17] unifies the two tasks - semantic and instance segmentation - detecting both thing and stuff. Therefore the output of every pixel i is a semantic label(l_i) and an instance id(z_i) - $(l_i, zi) \in L \times \mathbb{N}$ where $L = L^{th} \cup L^{st}$ and $L^{th} \cap L^{st} = \emptyset$. When a pixel is labeled with $l_i \in L^{st}$ then the corresponding id is irrelevant. However, some pixels may have a special void label. This task thus comes up with a panoptic quality metric that can quantify the performance of the model in this two-factor task.

Some example benchmarks for this task are Cityscapes [8], COCO[10] Mapillary Vistas [18], and ADE20K [16].

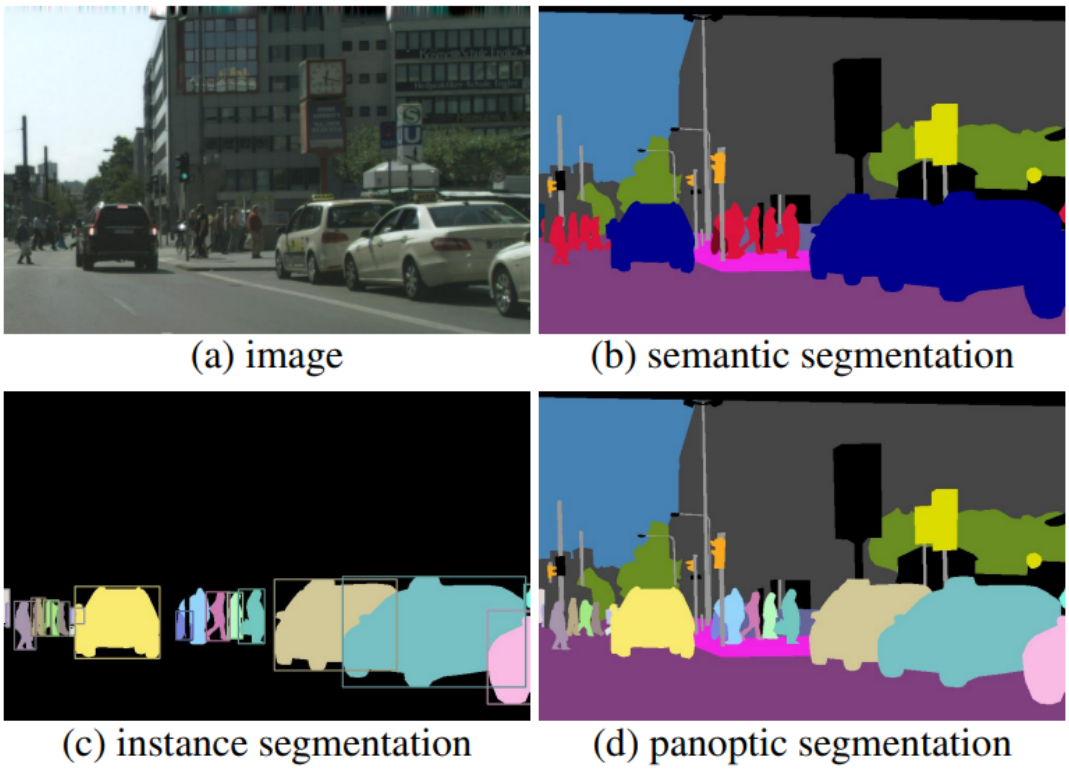


Figure 1.1.1: Image segmentation categories: a) semantic segmentation: stuff are not divided into objects, b)instance segmentation: only things are detected, stuff such as sky, road, etc. are ignored, c)panoptic segmentation: both stuff and things are detected with stuff remaining inseparable

1.2 Semantic Segmentation Datasets

1.2.1 Cityscapes Dataset

The Cityscapes Dataset [8] is a dataset specifically designed for autonomous driving in urban environments applications. It contains images of road scenery from 50 different city centers over all 4 seasons. These images are available in low(8 bit) and high(16 bit) resolution. The dataset includes both pixel-level and instance level annotations provided in two groups- coarse and fine annotations. Fine annotations are provided in 5000 images over 27 cities where all pixels have been labeled by creating polygons of instances. The rest of the images taken in 23 cities are used in coarse annotations where the polygons were chosen much faster causing lower accuracy labelling. In total 30 classes are included and 19 of them are used in evaluation as seen in 1.2.1.

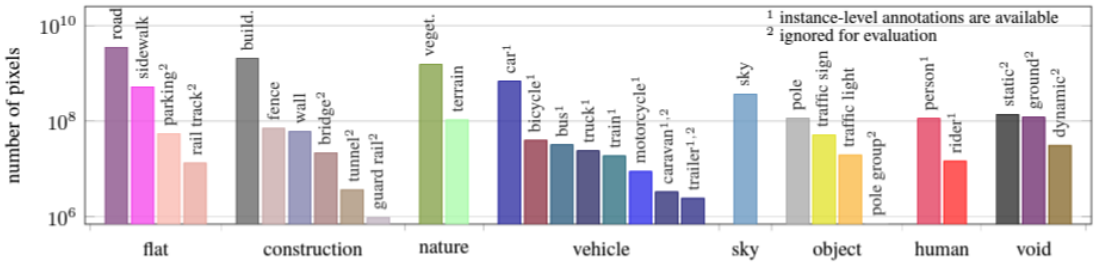


Figure 1.2.1: Cityscapes Classes, image from [19]

This dataset depicts much broader representation of inner city roads taking into account traffic and diverse climatic conditions with a wide variety of classes making it unique in the area of autonomous vehicle driving. Other datasets such as Kitti [62] provides images of roads but from suburban areas while CamVid [63] and DUS[64] provide less images from only one city. Also concerning instance-level annotations Cityscapes is the only dataset among these providing instances of people and vehicles. The only dataset more complete than the Cityscapes one is the Mapillary Vistas Dataset [65], however, it is too complex for the purpose of this project as it is five times the size of the Cityscapes dataset including 66 classes with images taken from all over the world with different capturing means.

	Cityscapes
Model	*mIoU*
InternImage-H [20]	87
HRNetV2-OCR_PSA [21]	86.93
InternImage-XL [20]	86.4
HRNet-OCR [22]	86.3
Vit-Adapter-L(Mask2Former, BEiT pretrain, Mapillary) [23]	85.8
OneFormer(ConvNetXt-XL, Mapillary, multiscale) [24]	85.8
SeMask(SeMask Swin-L Mask2Former) [25]	84.98
Sequential Ensemble (MiT-B5 + HRNet) [26]	84.8
OneFormer(ConvNetXt-XL, multi-scale) [24]	84.6
DiNAT-L(Mask2Former) [27]	84.5

Table 1.1: Cityscapes val Semantic Segmentation Leaderboard from the official website [28]

1.2.2 ADE20K Dataset

ADE20k was designed after recognizing the need for a general dataset covering a variety of scenes and common objects. Datasets before ADE20k had a limited set of scenes such as Cityscapes [8] or covered few or insignificant classes of objects such as COCO[10] and Pascal [15]. In ADE20k there are 20,210 images in the training set, 2,000 images in the validation set, and 3,000 images in the testing set. All the images are exhaustively annotated with objects. Many objects are also annotated with their parts as shown in 1.2.2. For each object there is additional information about whether it is occluded or cropped, and other attributes. The images in the validation set are exhaustively annotated with parts, while the part annotations are not exhaustive over the images in the training set. Additionally, parts can have parts too, and these are labeled with associations as well.

Figure 1.2.2: Annotations in ADE20k. Second row has object annotations and third row has object parts annotations.

Model	ADE20k *mIoU*
BEiT-3 [29]	62.8
EVA [30]	61.5
FD-SwinV2-G [31]	61.4
MaskDINO-SwinL[32]	60.8
OneFormer [24]	60.8
ViT-Adapter-L [23]	60.5
OneFormer [24]	58.6
ViT-Adapter-L [23]	58.4
OneFormer [24]	58.4
RSSeg-ViT-L [33]	58.4

Table 1.2: ADE20k val Semantic Segmentation Leaderboard from [34]

1.3 Metrics in semantic segmentation

As mentioned in [35] semantic segmentation is a complex task that takes into account relationships between classified pixels. In the following, every reference to class is applicable to category. The pixel-wise accuracy(pACC) is an initial metric to quantify performance :

$$acc = \frac{\sum_{i=1}^{k} n_{ii}}{\sum_{i=1}^{k} t_i} \tag{1.3.1}$$

Where n_{ij} is the number of pixels which belong to class i and were labeled as class j, k is the total number of classes/categoried, and $t_i = \sum_{j=1}^{k} n_{ij}$ is the total number of pixels of class/category i. However, this metric allows misleadingly for high initial accuracy rates in datasets where large regions have one class. In these

cases the model has only learnt frequent appearances of stuff in specific locations of the image. This problem can be solved with the following metrics:

mACC Mean accuracy is the mean accuracy across all classes: $\frac{1}{k}\sum_{i=1}^{k}\frac{n_{ii}}{t_i}$

IoU The metric IoU [36] is a per class assessment on the intersection of the inferred segmentation and the ground truth, divided by the union (commonly referred to as the 'intersection over union' metric) excluding pixels labelled as 'void':

$$IoU = \frac{truepos}{truepos + falsepos + falseneg} \tag{1.3.2}$$

mIoU Mean intersection over union(mIoU) is the mean across per class or per category IoU.

nIoU It is well-known that the global IoU measure is biased toward object instances that cover a large image area. In street scenes with their strong scale variation this can be problematic. Specifically for traffic participants, which are the key classes in our scenario, we aim to evaluate how well the individual instances in the scene are represented in the labeling. To address this, we additionally evaluate the semantic labeling using an instance-level intersection-over-union metric normalized IoU(nIoU) :

$$nIoU = \frac{truepos}{truepos + falsepos + falseneg} \tag{1.3.3}$$

Again $truepos$, $falsepos$, and $falseneg$ denote the numbers of true positive, false positive, and false negative pixels for a specific class, respectively. However, in contrast to the standard IoU measure, the metric is multiplied with λ which is a factor of frequency of this class. It is either predefined by the benchmark or calculated as such:

$$\begin{aligned} \lambda &= (\sum_{i=1}^{k} t_i)^{-1} \\ fwIoU &= (\sum_{i=1}^{k} t_i)^{-1} \sum_{i=1}^{k} \frac{n_{ii}}{t_i - n_{ii} + \sum_{j=1}^{k} n_{ji}} \in [0,1] \end{aligned} \tag{1.3.4}$$

This is why this metric is also called frequency weighted IoU.

iIoU This metric is the mean nIoU across all classes.

1.4 Loss functions in semantic segmentation

In semantic segmentation, various loss functions are used [37] for different dataset characteristics and therefore no loss function is superior to another. Some examples :

1. Binary Cross Entropy

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

   where y is the ground truth value and $\hat{y}$ is the predicted value

2. Weighted Binary Cross Entropy

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

   where y is the ground truth value and $\hat{y}$ is the predicted value. In this metric, the positive examples are weighted by some coefficient β which can be selected to either reduce the false negatives by setting $\beta > 1$ or to reduce the false positives by setting $\beta < 1$.

3. Balanced Cross Entropy

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

   same as the previous but also applies a weight on the negatives.

4. Dice Loss

   $DL(y, \hat{p}) = 1 - \frac{2y\hat{p}+1}{y+\hat{p}+1}$

5. Focal Loss

   $FL(p_t) = -a_t(1 - p_t)^{\gamma} \log(p_t)$
   where

   $$p_t = \begin{cases} p, & \text{if } y = 1 \\ 1 - p, & \text{if otherwise} \end{cases}$$

   where for $\gamma = 1$ it works just like Cross Entropy loss.

In imbalanced datasets focus based loss functions work better, while in balanced datasets binary-cross entropy is more applicable. Datasets with characteristics in between the two types mentioned above may benefit more from smoothed or generalized dice coefficient.

1.5 Transformers

There exist both CNN-based models and Transformer based models that approach the task of semantic segmentation. Initially, CNNs were used for this problem. When transformers became the new state-of-the-art in natural language processing (NLP) the Visual Transformer(ViT) [70] was developed. ViT showed remarkable performance in semantic segmentation [71] [57], beating the performance of state-of-the-art CNNs. A typical Transformer encoder consists of a multi-head self-attention (MSA) layer, a multi-layer perceptron (MLP), and a layer norm (LN). Transformers introduced the need for very large datasets. It has been proved [72] that they perform worse than CNNs in small datasets since their attention mechanisms need lots of information to learn local relations.

1.5.1 Attention

Attention is a mechanism in transformers that helps to draw connections between any parts of the sequence, so long-range dependencies are not a problem anymore. With transformers, long-range dependencies have the same likelihood of being taken into account as any other short-range dependencies.

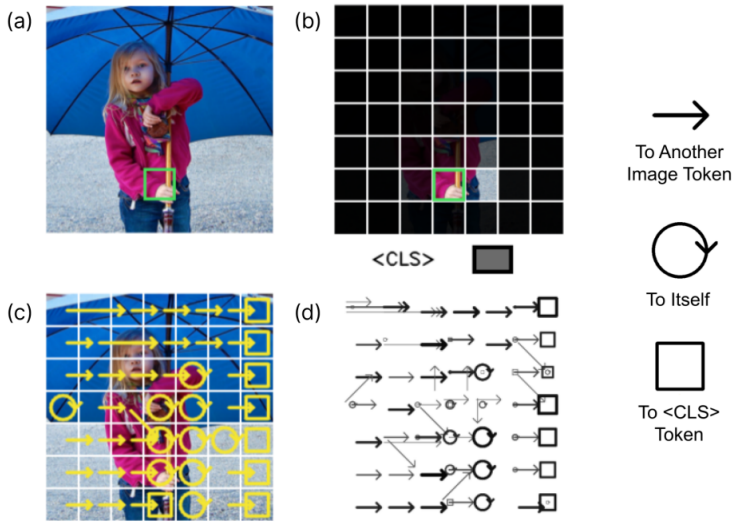


Figure 1.5.1: Attention in ViT: (a) Shows original image. (b) Transparency attention heatmap. The selected token (query) is highlighted with a green border. (c) Overlaid attention arrows. (d) Global attention flow. Taken from AttentionViz [73].

A visual example of how attention detects relationships between pixels in an image is shown in 1.5.1. The green box on 1.5.1.a is the query of the attention process. In 1.5.1.b transparency indicates the attention weight between the selected image patch and other regions of the image. This is also shown in 1.5.1.b where each arrow represents the strongest attention connection between a starting image patch and destination patch. Further, in 1.5.1.d all connections are shown on the image that are above a specific threshold and both opacity and line thickness are used to encode the strength of attention connections.

Chapter 2

Theoretical Background

2.1 Convolutional Neural Networks

2.1.1 The neuron

Neural networks are inspired from the brain's formation and from its building unit the neuron. Specifically a neural network is made of layers of neurons whose operation imitates a biological neuron's operation. Specifically, a neural network's neuron takes inputs and produces an output. These inputs are multiplied with learnable weights in order to control their magnitude of influence in the result. The output is produced by this weighted sum of inputs with an added learnable bias and an activation function applied on them. This activation function interprets the result and presents it in a meaningful way. For example, it can produce probabilities or binary output.

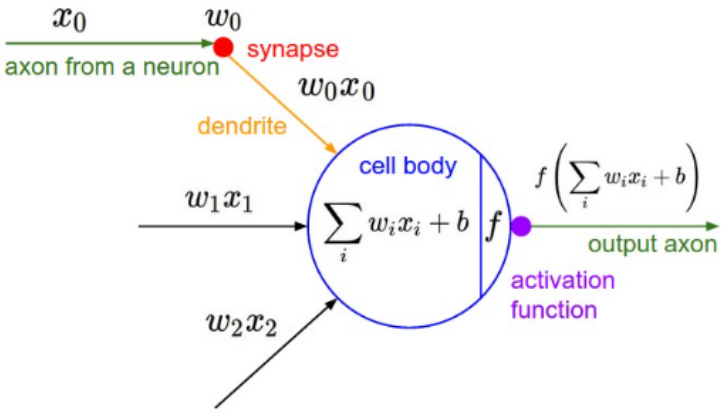


Figure 2.1.1: The neuron

Activation functions

Activation functions are functions that are applied on the dot product of the input signals with the weights of the neuron. They introduce a non-linearity to the computation of the result. Common activation functions are:

Sigmoid This activation function transforms the input to a probability of belonging to one of two classes. Thus the result belongs to $[0, 1]$ and the first class is assigned to 0 and the second to 1. It has the following formula:

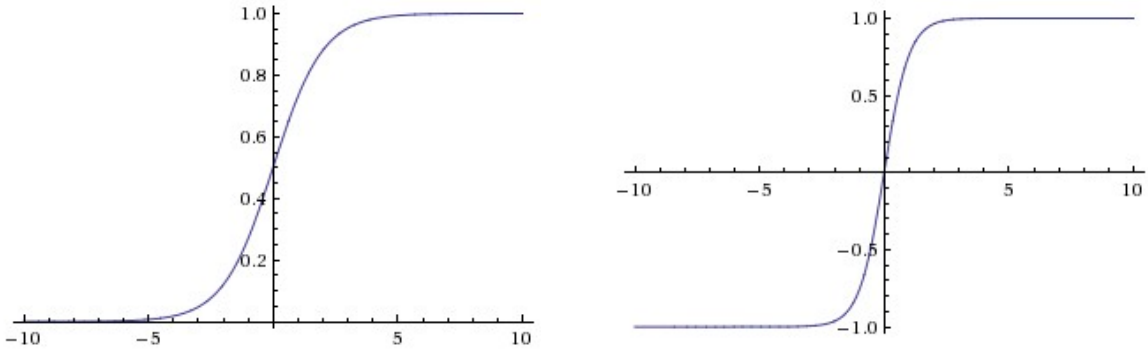
$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.1.1}$$

However, it has some undesirable characteristics such as causing the gradient to saturate and not being zero-centered.

Tanh The tanh graphical representation is very similar to the sigmoid but is now zero-centered as shown in 2.1.2. Tanh maps input to the range $[-1, 1]$ with this formula:

$$\tanh(x) = s\sigma(2x) - 1 \tag{2.1.2}$$

However, it still may cause the gradient to saturate.



(a) Sigmoid function

(b) Tanh function

Figure 2.1.2: Tanh function is similar to the sigmoid but it is also zero-centered which makes gradient descent more stable.

ReLU ReLu thresholds at zero by outputting:

$$f(x) = \max(0, x) \tag{2.1.3}$$

This function is easily implemented and eliminates the problem of the saturating gradient. However, they can cause neurons to die during training by zeroing their weights and this is irreversible for the rest of the training.

Leaky-ReLU Instead of completely zeroing negative inputs, leaky-ReLU multiplies them with a small constant as such:

$$f(x) = \mathbb{1}(x < 0)(\alpha x) + \mathbb{1}(x \geq 0)(x) \tag{2.1.4}$$

which has a controvesial performance compared to ReLU.

2.1.2 Neural Networks

Neural Networks [38] are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. Neural Networks receive an input (a single vector), and transform it through a series of hidden layers. Each hidden layer is made up of a set of neurons, where each neuron is fully connected to all neurons in the previous layer, and where neurons in a single layer function completely independently and do not share any connections. The last fully-connected layer is called the "output layer" and in classification settings it represents the class scores. The output then is compared to the desired output-ground truth- as loss which is calculated with a loss function. The gradient of the loss is used to update the weights of the network's layers according to the backpropagation algorithm.

Loss function

The loss function compares the output with the ground truth and based on that the hyperparameters of a model during training are adjusted in order to minimize it. Using any case specific loss function L to calculate the loss between one output and the ground truth, we obtain the total loss from the mean of the individual losses for every instance of the training set:

$$J(w^T, b) = \frac{1}{m} \sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) \tag{2.1.5}$$

where w are the weights, b the bias, m is the total number of training set data points, $\hat{y}$ is the prediction and y is the ground truth.

Optimization - Gradient Descent

The loss function lets us quantify the quality of any particular set of weights W. The goal of optimization is to find W that minimizes the loss function. It is possible to compute the best direction along which the weight vector should be changed that is mathematically guaranteed to be the direction of the steepest descent. This direction will be related to the gradient of the loss function as such:

$$w^{l+1} = w^l - \eta \frac{\partial J(w^T, b)}{\partial w} \tag{2.1.6}$$

where η is the learning rate. Gradient Descent is the procedure of repeatedly evaluating the gradient and then performing a parameter update.

Backpropagation Algorithm

However, in large neural networks the relationship of some weights and the loss function is very hard to find. The important contribution of the backpropagation technique [74] is in providing a computationally efficient method for evaluating the complex derivatives. The fraction in the calculation of the mean loss is not requires in the training process so starting from :

$$J(w) = \sum_{i=1}^{m} L_i(w) \tag{2.1.7}$$

For a Mean Squared Error function(MSE) J_i is defined as:

$$L_i = \frac{1}{2} \sum_{k} (\hat{y_{ik}} - y_{ik})^2 \tag{2.1.8}$$

Which calculates error when we have a multidimensional output. The input z_i in a unit is transformed into an output a_j of another unit from the dot product with the weight vector of the connection w_{ij}. The sum is then transformed by a non linear activation function h to give the activation z_j of unit j:

$$\begin{aligned} a_j &= \sum_i w_{ji} z_i \\ z_j &= h(a_j) \end{aligned} \tag{2.1.9}$$

Using the chain rule and 2.1.9 the derivative of J_i with respect to a weight w_{ij} can be obtained:

$$\frac{\partial L_i}{\partial w_{ij}} = \frac{\partial L_i}{\partial a_j}\frac{\partial a_j}{\partial w_{ij}} \tag{2.1.10}$$

where we define $\delta_j = \frac{\partial J_i}{\partial a_j}$ and from 2.1.9 $frac\partial a_j \partial w_{ij} = z_i$. For the final output units the gradient of the loss function produces $\delta_k = \hat{y_k} - y_k$ and for the hidden units, δ_k can be calculated from the output units using the chain rule as such:

$$\delta_j = \frac{\partial L_i}{\partial a_j} = \sum_k \frac{\partial L_i}{\partial a_k}\frac{\partial a_k}{\partial a_j}\delta_j = h'(a_j j)\sum_k w_{kj}\delta_k \tag{2.1.11}$$

The steps of the algorithm can thus be summarized as follows:

1. Apply an input vector x_m to the network and forward propagate through the network using 2.1.9 to find the activations of all the hidden and output units.
2. Evaluate the $\delta_k = \hat{y_k} - y_k$ for all the output units
3. Backpropagate the δ using 2.1.9 to obtain δ_j for each hidden unit in the network.
4. Use $\frac{\partial L_i}{\partial w_{ij}} = \delta_j z_i$ to evaluate the required derivatives.

The derivative of the total error J can then be obtained by repeating the above steps for each instance of the training set set and then summing over:

$$\frac{\partial J}{\partial w_{ij}} = \sum_m \frac{\partial L_m}{\partial w_{ij}} \tag{2.1.12}$$

Regularization

The goal of a neural network is to learn a correspondence of input to output from training data and apply it on test data. Thus, it is important to be able to generalize its weights and not learn specifically the examples from the training data. When a model fits perfectly the training data it is called overfitting. Regularization is a technique of controlling the overfitting of Neural Networks. Specifically, it adds an extra component to the loss function which prevents the weights from increasing excessively their magnitude and thus update in a less flexible way.

L1 regularization In L1 regularization, for each weight w we add the term $\lambda_1|w|$ to the loss function. It has the intriguing property that it leads the weight vectors to become sparse during optimization (i.e. very close to exactly zero). In other words, neurons with L1 regularization end up using only a sparse subset of their most important inputs and become nearly invariant to the "noisy" inputs. In practice, if you are not concerned with explicit feature selection, L2 regularization can be expected to give superior performance over L1.

L2 regularization L2 regularization is perhaps the most common form of regularization. It can be implemented by penalizing the squared magnitude of all parameters directly in the loss function. That is, for every weight w in the network, the term $\frac{1}{2}\lambda w^2$ is added to the loss function where λ is the regularization strength. The factor of $\frac{1}{2}$ is used so that the gradient of the term is simple λw. The L2 regularization has the intuitive interpretation of heavily penalizing peaky weight vectors and preferring diffuse weight vectors. This has the appealing property of encouraging the network to use all of its inputs a little rather than some of its inputs a lot. Additionally, during gradient descent parameter update, using the L2 regularization ultimately

means that every weight is decayed linearly:$w+ = -\lambda \times w$ towards zero. It is possible to combine the L1 regularization with the L2 regularization: $\lambda_1|w| + \lambda_2 w^2$.

Dropout Dropout [39] is a training technique that also prevents overfitting. The dropout probability indicates the probability with which a neuron is kept active or is set to zero in the model. It can be thought as sampling a Neural Network within the full Neural Network, and only updating the parameters of the sampled network based on the input data.

Batch size

The batch size is a hyperparameter that defines the number of samples to work through before updating the internal model parameters. A batch is the number of samples a for-loop iterates over making predictions. At the end of the batch, the predictions are compared to the expected output variables and an error is calculated. From this error, the update algorithm is used to improve the model, e.g. move down along the error gradient. A training dataset can be divided into one or more batches. When all training samples are used to create one batch, the learning algorithm is called batch gradient descent. When the batch is the size of one sample, the learning algorithm is called stochastic gradient descent. When the batch size is more than one sample and less than the size of the training dataset, the learning algorithm is called mini-batch gradient descent.

2.1.3 Convolution

Convolution [40] is a mathematical operation applied on two functions. Discrete 2-D convolution involves two matrices and is defined by the following formula:

$$(f * w)[x_1, x_2] = \sum_{m=-k}^{k} f(i,j) \cdot w(x_1 - i, x_2 - j) \tag{2.1.13}$$

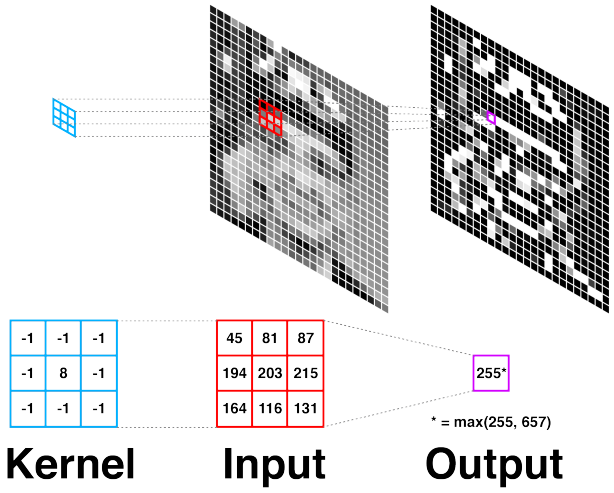


Figure 2.1.3: Convolution visualization from [41]

In general, convolution is defined for any matrices and the first argument (f) is often referred to as the input, and the second argument (w) as the kernel. The output is sometimes referred to as the feature map. This operation is used in CNNs to process an image and recognize on it specific characteristics. Unlike a regular Neural Network, the layers of a CNN have neurons arranged in 3 dimensions: width, height, depth. the neurons in a layer will only be connected to a small region of the layer before it, instead of all of the neurons in a fully-connected manner.As described above, a simple CNN is a sequence of layers, and every layer of a

CNN transforms one volume of activations to another through a differentiable function. The types of layers used to build a CNN architecture include: Convolutional Layer, Pooling Layer, and Fully-Connected Layer.

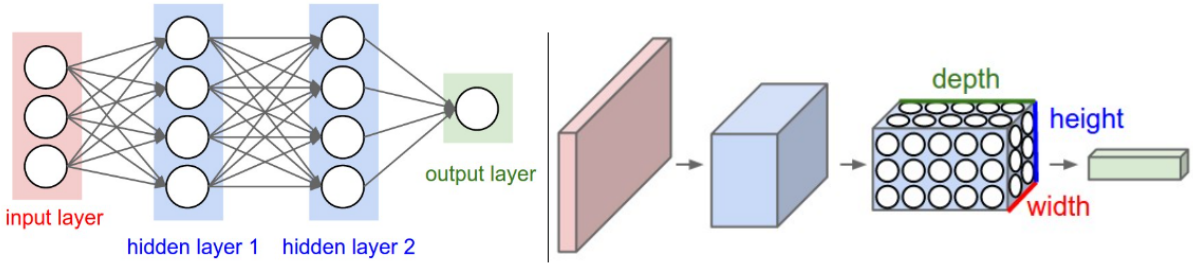


Figure 2.1.4: CNNs architecture from [38]

Due to the large number of pixels in an image, every neuron has a receptive field and is applied only to the corresponding layers. Therefore, it has $w \times h \times c$ number of weights where $w \times h$ is the receptive field and c is the number of channels of the input image. That's why convolutional neural networks do not detect as sufficiently as attention mechanisms contextual information. The depth of the filter as seen in 2.1.4 indicates that there are multiple neurons applied to the same patch of the input image. Therefore, the output image is also 3-dimensional with a depth equal to the depth of the filter. The purpose of multiple stacked filters is to detect multiple characteristics with one convolutional layer. Even though every neuron has a specific receptive field, there is not need for multiple neurons for every patch of the input image. The neurons can be simple shifted and compute the output for every patch using the same filter. The shifting is specified by the stride of the filter which indicates how many pixels the filter is slided across the image before it is reapplied. Lastly, zero-padding is also an important hyperparameter of a convolutional layer as it allows to control further the dimensions of the output. Assuming the input image is a square(ignoring the channels) and has one dimension W, the convolutional layer has one dimension of F, the stride is S, and the padding used is P then the result dimension of the square output is equal to $\frac{W-F+2P}{S} + 1$.

2.1.4 Pooling layer

The pool layers are in charge of downsampling the spatial dimensions of the input. The pooling size sets the patch size by which the input is divided. Then in this patch, max pooling, for example, keeps the max number and replaces the patch with that. Thus, it downsamples by a factor equal to the patch size.

2.1.5 Batch Normalization

BN is used to standardize the inputs to any particular layer. That entails the input to have zero mean and unit variance. BN layer transforms each input in the current mini-batch by subtracting the input mean in the current mini-batch and dividing it by the standard deviation. However, it is not proven that the models performs better with zero mean and unit variance. It might perform better with some other mean and variance. Hence the BN layer also introduces two learnable parameters γ and β which adjust those parameters.

2.1.6 Fully connected layer

Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset.

2.2 Transformers

2.2.1 Architecture

Transformers [75] is a state of the art architecture in sequence transduction that has been shown to outperform recurrent and convolutional models while depicting smaller computational complexity, thus faster training time. When in CNNs many stacked layers are needed to detected long-range dependencies, in self-attention, these long-range dependencies are detected by computing efficiently for an output position context information from every input position. Transformers' architecture depicted in Figure 2.2.1 is composed by an encoder and a decoder. An encoder is made of 6 stacked layers each one composed of a multi-head self-attention module and a position-wise feed-forward network. A decoder is also made of the same 6 stacked layers, with an additional Masked Multi-Head Attention module at the start of each layer. Multi-Head attention is stacked self-attention heads -8 mentioned in [75]- that are applied on the input in parallel and have different learnable weights. Their purpose is to bring different perspectives on dependencies in a sequence and combine into a single averaged attention output. Masked self-attention is the limitation of attention range in self-attention so as computation of outputs is not affected from "future" outputs.

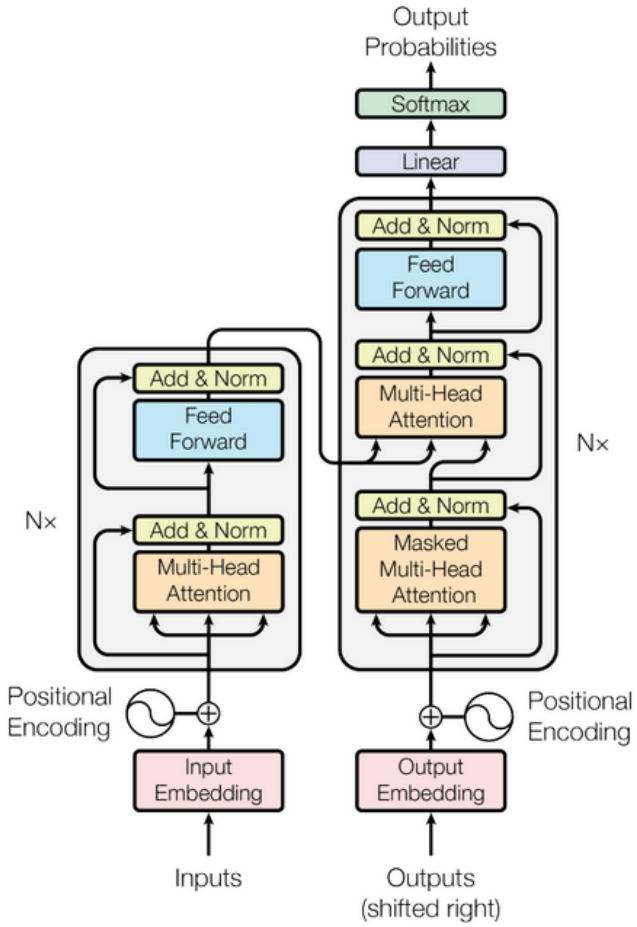


Figure 2.2.1: Transformer Architecture

Every layer has output and input of the same dimension of 512 transformed from original input using

learnt embeddings. It also has a residual connection and a batch normalization module after each sublayer. Lastly, positional embeddings(such as cosine functions of different frequencies) are fused into the input vector in order to model spatial or chronic relationships between positions. The Feed-Forward unit implements two linear transformations and two ReLU activations in between them on each position independently :

$$\mathrm{FF}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{2.2.1}$$

2.2.2 Self-Attention

Starting with three learnable vectors: query(Q) and key(K)-value(V) pair the attention sublayer intends to calculate the value that results from the key closest to the query. By taking the inner product 2.2.2 between the query and the key and a softmax function of it, we end up with a similarity probability matrix. This matrix is used as weights to produce the value result from the weighted value matrix. The dot product between the query and key matrix is scaled with a factor $\frac{1}{\sqrt{d_k}}$(d_k is the dimension of the key matrix) in order to prevent the problem of the exploding gradient when we have large values.

$$Attention(K, Q, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.2.2}$$

In multi-head self-attention, there are stacked 8 self-attention modules each one with its one learnable weights providing a different representation subspace. Their outputs are concatenated as such:

$$\begin{aligned} MH(Q, K, V) &= Concat(h_1, h_1, ..., h_h)W^o \\ h_i &= Attention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2.2.3}$$

where the W matrices are projections bringing the Q,K matrices to a dimension of d_{model}/h(number of heads h=8) and bringing the concatenated output representation to a dimension of d_{model}.

Self-attention computation time is faster than recurrent networks and constant considering the d dimension is larger than the size of the input n. In cases of long sequences, self-attention decreases the attention span to a neighbourhood of size r with complexity of $O(\frac{n}{r})$. In general self-attention is faster than convolutional layers with the same attention span and in the worst case, separable convolution has the same complexity as self-attention.

2.2.3 Interlaced Sparse Self-Attention

Modern applications of self-attention require computation in high resolution streams of data. Since in self-attention every position attends to every other position computational time is $O(n^2)$ which is very heavy for large n. Interlaced sparse self-attention is introduced as a much lighter and just as efficient mechanism. Specifically, it factorizes the dense affinity matrix A into two sparse matrices: one that represents long-range

dependencies(A^L) and one that represents the short-range ones(A^S). The input position are divided into Q equal subsets of P positions. The long-range dependency matrix is calculated by creating P subsets including only one position from every initial subset, known as interlacing. The short-range dependency matrix is created by applying self-attention in each one of these initial subsets separately. The affinity matrix, A, is the result of this equation:

$$\begin{aligned} A = \text{softmax}\,\frac{\theta(X)\phi(X)^T}{\sqrt{d}} \\ Z = Ag(X) \end{aligned} \tag{2.2.4}$$

The A^L computation as described is equivalent to permutating the originally input matrix and dividing the permutated version into P neighbouring subsets. The calculation of self-attention per p block is described by these equations:

$$\begin{aligned} A_p^L = \text{softmax}\,\frac{\theta(X_p^L)\phi(X_p^L)^T}{\sqrt{d}} \\ Z_p^L = A_p^L g(X_p^L) \\ A^L = \begin{pmatrix} A_1^L & 0 & ... & 0 \\ 0 & A_2^L & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & ... & A_P^L \end{pmatrix} \end{aligned} \tag{2.2.5}$$

Equivalently for the computation of the A^S matrix we used these equations:

$$\begin{aligned} A_p^S = \text{softmax}\,\frac{\theta(X_p^S)\phi(X_p^S)^T}{\sqrt{d}} \\ Z_p^S = A_p^S g(X_p^S) \\ A^S = \begin{pmatrix} A_1^S & 0 & ... & 0 \\ 0 & A_2^S & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & ... & A_P^S \end{pmatrix} \end{aligned} \tag{2.2.6}$$

This approach has been compared with various existing approaches and it has been shown that it achieves competitive performance on various semantic segmentation datasets while being much more efficient compared with the conventional self-attention mechanism.

Chapter 3

Related Work

The semantic segmentation problem [42] was initially approached with clustering algorithms [43] used with additional information from contours and edges. Later modelling with Markov process [5] was explored as long as combining contour detection in a hierarchical approach. However, nowadays the prominent models are directed to region-based models, Convolutional Neural Networks, and Transformers.

3.1 Region-based models

Region-based models are divided into to separate stages: identification of regions in an image and classification according to dataset defined classes. Pixels are assigned classes after these two stages in some models in an end-to-end trainable manner and in others training takes places only in the previous stages.

3.1.1 R-CNN

The rise of CNNs in image classification led to the creation of R-CNN [44] which is the first model to show CNNs outperforming in object detection. It was later altered to be used in semantic segmentation. In object detection, R-CNN extracts around 2000 bottom-up region proposals using selective search. These regions are translated into a standard length vector of features using a large CNN which is then classified with class-specific linear SVMs. A greedy non maximum suppression algorithm is applied to reject a region if it has a lower score than a region it is intersected with larger than a learnt threshold. In semantic segmentation, it alters O2P [45] which already extracts 150 regions with CPMC [46] and evaluates them with support vector regression(SVR). Specifically, the feature extraction stage is replaced with CNNs exploring three methods: the first computes features on the rectangular box that warps the detected region, the second computes features only on the region's foreground mask, and the third concatenates the two vectors of the methods above. The best performing method was the concatenation of features showing that context is crucial in the classification process of a region.

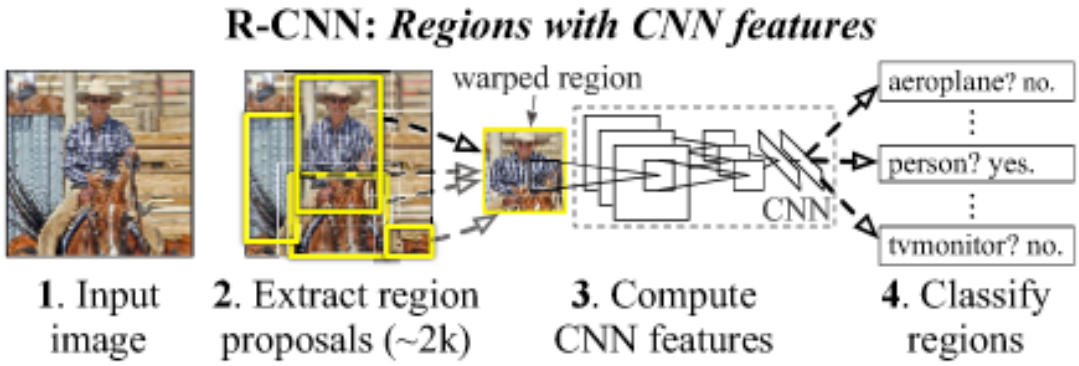


Figure 3.1.1: R-CNN [44] object detection model structure

3.1.2 Fast and Faster R-CNN

Fast R-CNN [47] is an optimization of R-CNN [44] used in object detection, later influencing architectures in semantic segmentation. Specifically, it introduces Region of Interest(ROI) which will be applied in semantic segmentation region-based models adapted for free form regions. It is faster than R-CNN since it includes single-stage training with multi-task loss while training can update all network layers, and no disk storage for feature caching is needed. The network first processes the whole image with several convolutional and max pooling layers to produce a conv feature map. Then, for each object proposal a region of interest (RoI) pooling layer extracts a fixed-length feature vector from the feature map. Each feature vector is fed into a sequence of fully connected (fc) layers that finally branch into two sibling output layers: one that produces softmax probability estimates over K object classes plus a catch-all "background" class and another layer that outputs four real-valued numbers for each of the K object classes. Each set of 4 values encodes refined bounding-box positions for one of the K classes.

Region of Interest(ROI)

The ROI pooling layer [76] extracts a feature vector from a feature map obtained from a Convolutional Neural Network and from the coordinates of the region. The ROI layer divides the region into equal-sized sections according to the dimension of the output and keeps the largest value in each region.

Further improvements

Faster R-CNN [48] further extends the concept of sharing the same CNN for accelerating training time by sharing the same CNN in the stage of identifying object regions and in the stage of classifying them 3.1.2. Additionally, it allows for an improved training procedure since it can be trained end-to-end.

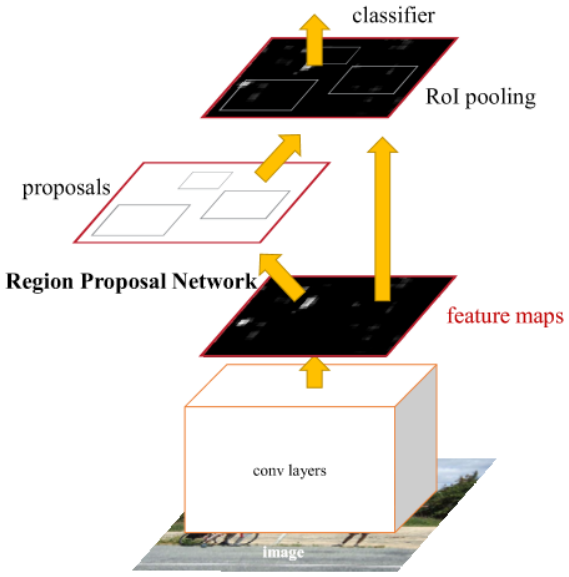


Figure 3.1.2: Faster R-CNN unified CNN model

3.1.3 Mask R-CNN

3.1.4 Region-based method applied in semantic segmentation

Up until this method, two were the prominent paradigms in semantic segmentation: region-based models and FCN models. The first one, produces pixel labelling separately from the main model not allowing for end-to-end training. On the other hand, FCNs allow for end to end training as they produce pixel labels immediately, however, the nature of convolutions which are performed with square patches doesn't recognize complex regions and doesn't produce sharp boundaries. The idea behind this model [49] was to fuse the pros of the previous models by converting the first paradigm into an end-to-end trainable model 3.1.3. The missing model is therefore added which converts regions to pixels and classifies pixels instead of regions allowing for calculation of per-pixel loss. It also adopts the ROI pooling layer as in Faster R-CNN [48] which produces features from free-form regions. ROI is not a CNN thus it doesn't need a bounding box to compute features, it computes them only from the pixels included in the region. However, region context is important, thus, ROI is also calculated on the region with its bounding box is also calculated and these two sets of features are passed forward to the FCN. According to two versions of the method, either they are both passed to the same FCN with the same tied weigths or to different FCNs with separate weights developing two different classifiers.

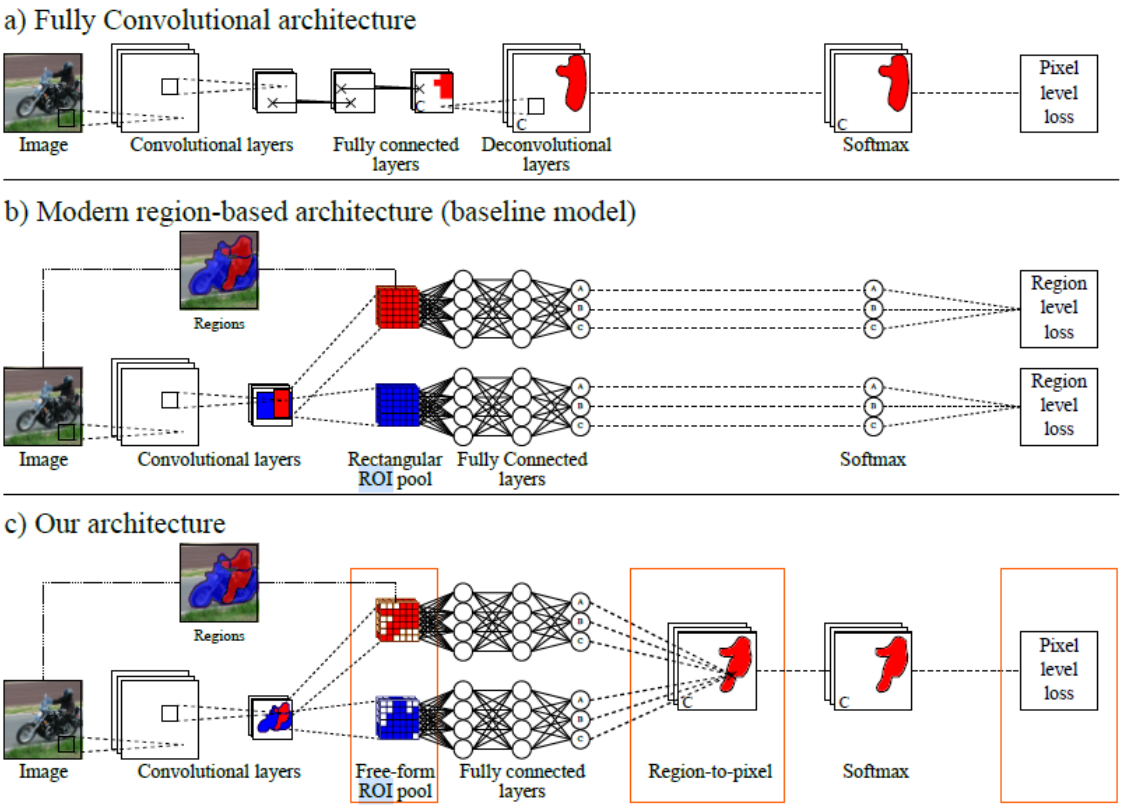


Figure 3.1.3: Summary of the three architectures: a) FCN in semantic segmentation are end-to-end trainable but they recognize regions from square patches, thus producing imprecise predictions of complex regions. b) The region-based model before pixel classification and therefore not end-to-end trainable. c) Newest region-based model which applies the loss criterion on the pixels producing an end-to-end trainable model. Insertions are highlights in orange. Image from [49]

Pixel labelling

Previously, at train time there was no pixel labelling and the mapping from region classes to pixel classes was done only at test time as such:

$$o_p = \arg\max_c \ max_{r \ni p} \ \text{softmax}_c \, S_{r,c} \tag{3.1.1}$$

Meaning that after obtaining the softmax of activations per region and class, the pixel is denoted with the class that has the highest softmax from the regions that include the pixel. Since softmax is applied before pixel classification, regions with low but highly varying activation scores are unsure about the class, but can still yield high probabilities due to the softmax. This leads to such non-distinctive regions wrongly affecting the final pixel prediction. Additionally, since $max_{r \ni p}$ occurs at test time, the pixel-wise evaluation criterion at test time is different from the region-level optimization criterion at training time. This leads to training with all regions and testing with most of them ignored.

The loss calculation equation was:

$$L = -\sum_c \frac{1}{R} \sum_{r=1}^{R} y_{r,c} \ \log \text{softmax}_c \, S_{r,c} \tag{3.1.2}$$

It calculates the cross-entropy log-loss on the regions, where R is the number of regions, S_r is the region-level score for class c, and $y_{r,c}$ is the binary indicator of whether the ground truth of region r is class c. Therefore, pixels are ignored completely in the training procedure.

The alternate mapping from the classified region to the unclassified pixels proposed by this model is described in this equation:

$$o_p = \arg\max_c \ \text{softmax}_c \ max_{r \ni p} S_{r,c} \tag{3.1.3}$$

This approach classifies pixels before softmax and loss computation. Specifically it assigns a class to a pixel by choosing the max activation on this pixel for a certain pair of region and class. Therefore the loss is calculated on the assigned class of the pixel that is always the class with the highest activation on this pixel. The new loss calculation equation is:

$$L = -\sum_c \frac{1}{P} \sum_{p=1}^{P} y_{p,c} \ \log \text{softmax}_c \, S_{p,c} \tag{3.1.4}$$

It calculates the cross-entropy log-loss on the pixel, where P is the number of pixels, $S_{p,c}$ is the pixel-level score for class c, and $y_{p,c}$ is the binary indicator of whether the ground truth of pixel p is class c.

3.2 Convolutional Neural Networks

3.2.1 Fully Convolutional Neural Networks(FCN)

Fully Convolutional Networks(FCN) as described in [50] are networks that accept arbitrary size input and produce same size output. They include only convolutional layers and up-sampling/down-sampling mechanisms. In this paper, it is intended to transform state of the art classification CNNs so that they can be applied to segmentation tasks while using their pre-trained weights and trained pixelwise end-to-end. This was possible by replacing fully connected layers with 1×1 convolutional layers producing thus equal size of output. The original size of input which was decreased by down-sampling occurring in previous convolutional layers can be retained by up-sampling mechanisms. The one used in the paper is backwards convolution which uses a stride of $\frac{1}{f}$ and weights that can be learnt allowing for end-to-end learning in the model. Additionally, in order to achieve finer predictions after taking into account the limits of down-sampling in the output detail of prediction, a skip architecture is introduced. This architecture brings a non-linearity into the model by performing an element-wise addition of predictions from previous layers to up-sampled predictions. The FCN-8s, which performs best(FCN-32s and FCN-16s also exist), is composed as shown in Figure 3.2.1 by a main-line of convolutional layers followed by maxpool layers and secondary lines that are added to this main line after the output is upsampled with backwards convolution also known as transpose convolution. The second row takes the output from the 4th convolutional-maxpool unit, performs a stride 1 convolution, and adds it to the transpose convolution output of the first row. The third row takes the output from the second convolutional-maxpool unitk, performs a stride 1 convolution, and adds the output to the output of the second row performing again a transpose convolution to form an output of the desired size.

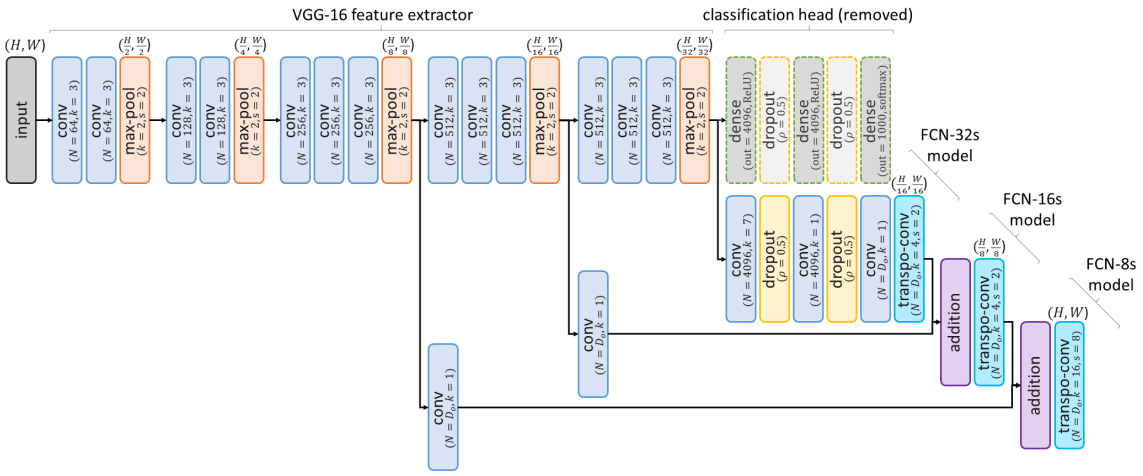


Figure 3.2.1: FCN Architecture based on VGG-16 from ch.6 in [51]

3.2.2 U-Net

The U-Net architecture [52] builds upon the FCN architecture using the skip architecture and the transpose convolutions but is trained much faster and produces more precise results. The architecture resembles a U as it symmetrically constructs the encoder and the decoder shown in 3.2.2. The modification that allows for better precision is the large number of features on the decoder side whilst in FCN the "decoder" side which is consisted by successive transpose convolutions and additions has a constant number of features equal with the number of classes. As for the encoder, it consists of 3 groups of 2 successive convolution layers

followed by ReLUs and a pooling layer. This is symmetrically repeated on the decoder side with the pooling layers replaced by transpose convolutions and an addition module precedes every layer implementing the skip architecture. An extension of the U-Net

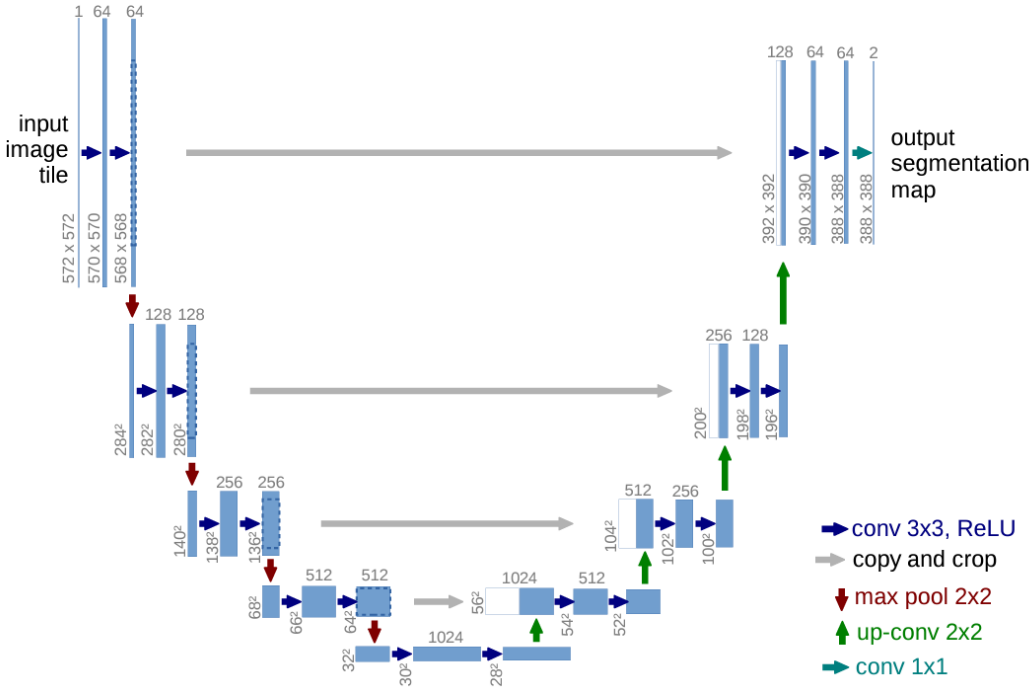


Figure 3.2.2: U-Net Architecture from [52]

3.2.3 DeepLab and Atrous Convolution

Deeplab [53] builds on the previous architectures incorporating in FCN architectures atrous convolution replacing deconvolution. Atrous convolution in 1D is displayed in 3.2.1, and can be described by convolving with a filter that has r-1 zeroes between its values when rate of dilation is r. It allows for a larger field of view while keeping the same number of parameters and keeps the resolution when used with padding.

$$y[i] = \sum_k x[i + rk]w[k] \tag{3.2.1}$$

Deeplab also implements bilinear interpolation with fully connected Conditional Random Field(CRF) [77] to get the the output stride from 8x to the original resolution and provide more detailed results. Lastly, Atrous Spacial Pyramid Pooling(ASPP) is proposed which adds atrous convolutions in multiple rates allowing for multi-resolutional processing. Extending this work Deeplabv3 [54] rejects CRFs and incorporates in ASPP another component which is a pooled version of the original image. Additionally, batch normalization is used after every atrous convolution. Deeplabv3+ [55] further improves this architecture by using the concept of the encoder-decoder as the previous version could only produce results up to 8 output stride which is not the expected resolution for semantic segmentation. Using the Deeplabv3 as the encoder with an output stride of 16 and 256 features forwarded as input to the decoder after it is convolved by a 1×1 filter to reduce the channels. Similarly the low-level layer of the FCN with the same resolution is convolved by a 1×1 filter to

reduce its channels and then it is concatenated with the previous result as shown in. After the concatenated vector is convolved by a 3×3 filter and produces the final result. Lastly, atrous convolution is replaced by depthwise seperable atrous convolution which is a much faster process. Comparing the three architectures we see that Deeplab processes the image in a higher resolutional level while U-Net[52] uses the encoder-decoder to reconstruct from low resolution and Deeplabv3+ combines both to produce a precise result from an already accurate result in the original resolution.

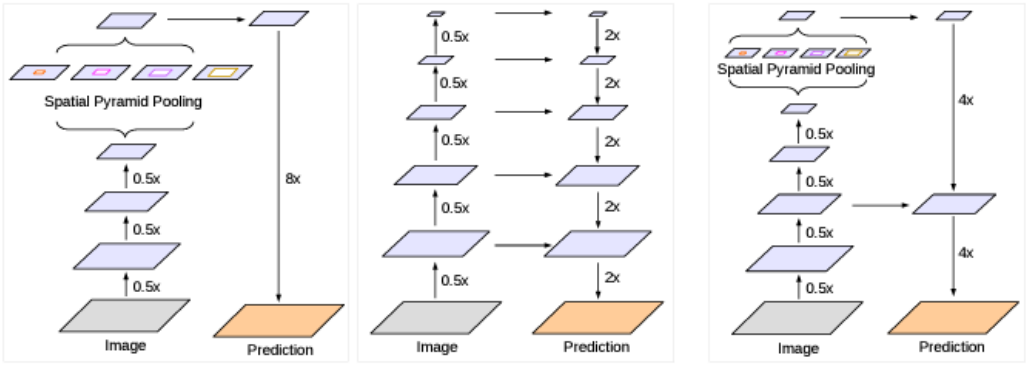


Figure 3.2.3: Comparison of Deeplab, U-Net, and Deeplabv3+ architectures from [55]

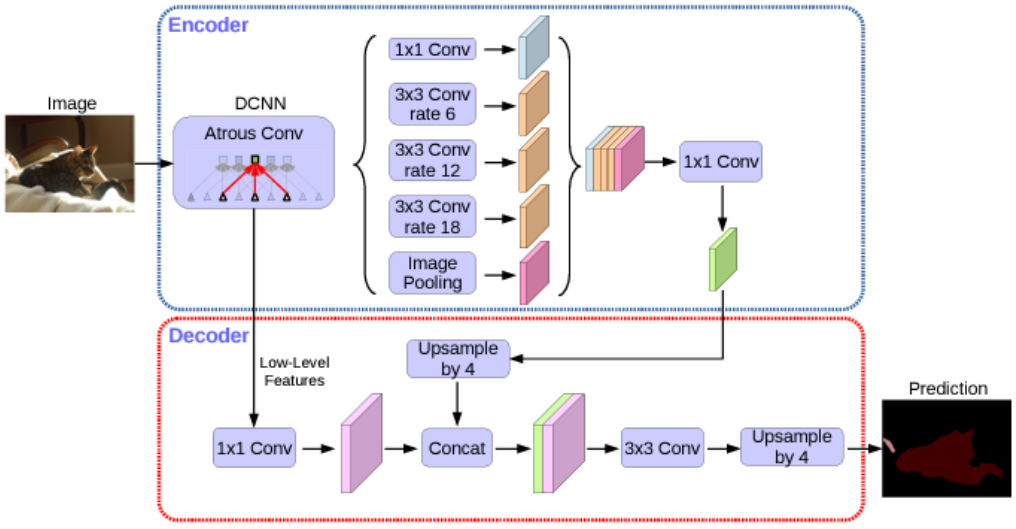


Figure 3.2.4: Deeplabv3+ Architecture from [55]

3.2.4 Deep High Resolution Convolutional Neural Networks (HRNet)

High resolution convolutional networks emerged from the observation that none of the architectures before them kept high resolution streams across the network but rather upsampled from low resolution[50], fused low level high resolution to the ending layers [52], created medium-resolution streams [53, 54], or implemented encoder-decoder architectures [55, 52]. HRNet [56] introduced the concept of parallel multi-resolution networks that fuse on the end of every stage shown in 3.2.5. The network is made of 4 stages and on every stage one lower resolution is added in parallel. On every stage there are 4 residual units at each resolution with every unit composed by two 3×3 convolution, batch normalization, and relu except for the first stage where every unit is made of a bottleneck with width 64 and a 3×3 convolution. The first stream is of resolution $\frac{1}{4}$ and has C channels and every consecutive one downsamples and increases the channels by a factor of 2. The downsampling is possible with a 3×3 convolution of stride 2 while the upsampling with bilinear interpolation followed by 1×1 convolution. The semantic segmentation result in HRNetV2 is taken by fusing the four streams to the first stream after upsampling the last three and using estimating segmentation maps.

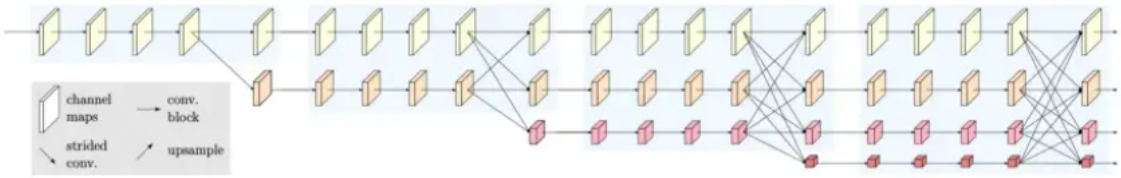


Figure 3.2.5: HRNet Architecture from [50]

3.3 Transformers

3.3.1 Shifted Window Transformer(Swin Transformer)

The Swin transformer [57] is an architecture that aims to convert the Transformer from an natural language processing tool (NLP) tool [78] [79] [80] to a vision one. However, in order to be able to exploit the advantages of self attention in images with efficiency while also detecting multi-scale dependencies it implements shifted window attention and patch merging. According to this architecture an image is split into patches of size 4×4 and these patches belong to windows of size $M \times M$. The multi-head self-attention layer which is now replaced by Window multi-head self-attention(W-MSA) on the encoder and each self-attention computation is restricted only within a window. The windows, originally split at the top-left corner, are shifted in every attention head on the Shifted-Window multi-head self-attention(SW-MSA) in the decoder. The Swin Transformer also allows for detection of multi-resolutional dependencies by sub-sampling input through patch merging on every stage as shown in 3.3.1. This architecture uses GELU and a 2-layer MLP after the attention unit. Additionally, a normalization layer is placed after every module and residual connections are used between modules as depicted in 3.3.1.

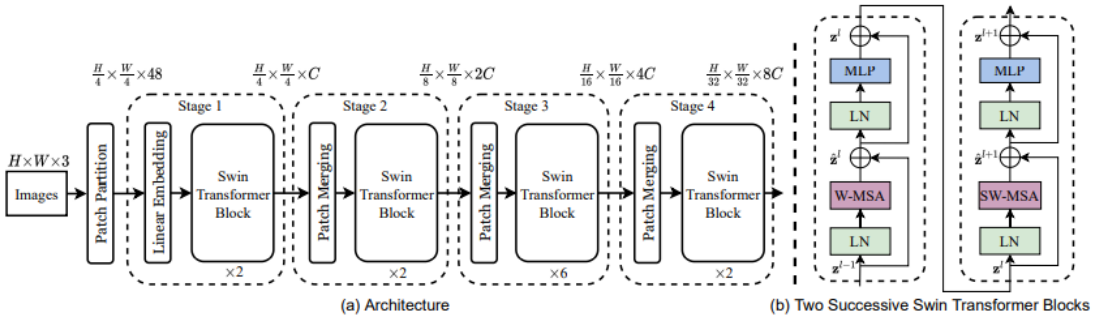


Figure 3.3.1: swin Transformer Architecture from [57]

Patch-merging

The patches are of size of 4×4 setting the window dimensions to $\frac{W}{4} \times \frac{H}{4}$. Every patch has a token set based on their $4 \times 4 \times 3 = 48$ concatenated features. However, the token converted by a linear embedding to a dimension C and the patch is directed to the Transformer completing there Stage 1. On Stage 2, a patch merging layer merges $2 \times 2 = 4$ patches into 1 patch with 4C concatenated features which are again converted by a linear embedding to 2C features setting window size to $\frac{W}{8} \times \frac{H}{8}$. Stage 2 is completed with a Swin Transformer Block. Further processing is applied on stage 3 and 4 in a similar way, where again window size is reduced to $\frac{W}{16} \times \frac{H}{16}$ and $\frac{W}{32} \times \frac{H}{32}$ and every patch has a feature dimension of 4C and 8C respectively.

Window shifting

Window shifting is implemented on the multi-head self-attention part of the decoder allowing for detection of broader dependencies. The initial window is placed on the left-top corner and the rest are placed consecutively in an adjacent non-overlapping way. The next formation is created by shifting diagonally the windows. However, this results in more windows than initially with some uncompleted. The solution given to this problem is the cyclic-shifting method where uncompleted windows from the top left are moved to the bottom right and connected with the uncompleted windows there to form regular windows. Afterwards, self-attention applied in these concatenated windows takes into account this shift by masking features in the window that weren't neighbouring before the shift.

High Resolution Swin Transformer (HRSTFormer)

The High Resolution Swin Transformer [61] borrows the logic of the HRNet [56] and applies it using Swin transformers. Similarly to the HRNet it uses 4 parallel resolution streams 1.2.1 and after every stage which is completed with one pass of every stream through a Swin Transformer Block it applies information exchange. This is possible by using patch merging blocks and thus creating for every stream the one resolution lower representation of the result. All of the results and their downsampled representations are forwarded to a multi-resolution feature fusion block except for the last stream's downsampled output which is used to create the next lower resolution stream stream. The multi-resolution feature fusion block fuses the results of equal resolution and outputs the same number of resolution streams as before. At the last stage, which is stage 4 for HRSTNet-4, every resolution stream is upsampled(with a patch expanding block) and downsampled to the rest of the resolutions so that we cam concatenate all the results in different resolutions. Every concatenate block is followed by a residual multi-resolution feature fusion(MRFF) block used to convert the results from 1D to 3D features and after upsampling to the main stream resolution all results are concatenated, upsamppled to the original resolution, and passed through a convolutional filter.

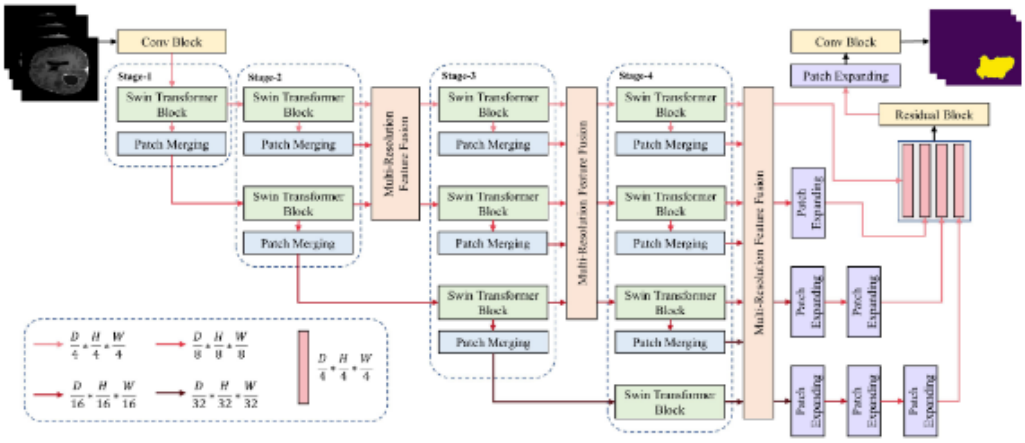


Figure 3.3.2: HRSTNet Architecture [61]

3.3.2 Mask Transformer (MaskFormer, Mask2Former)

The idea behind the MaskFormer [19] lies on the similarity of the different segmentation tasks and the acknowledgement of the need to construct a unified architecture that can perform these tasks simultaneously. The MaskFormer aims to use the instance segmentation technique, mask classification, in semantic image segmentation questioning the notion of per pixel classification. Specifically, the MaskFormer predicts a set of N pairs of class and mask, $z = \{(c_i, m_i)\}_{i=1,\dots,N}$ performing thus primarily mask classification and then converts the results to panoptic, instance, or semantic segmentation output. It has similar approach to R-CNN [48] [47] architectures before they were converted to end-to-end training models and agrees with their absence of pixel-level loss.

Architecture

The architecture of this mask classification model is divided in three modules as shown in 3.3.3. The pixel-level module uses a pixel-level classification architecture which first down-samples the image and then up-samples it with a decoder. The low-resolution image features extracted from the backbone of the down-sampled

image are forwarded to the transformer module which performs a query on them and produces the object queries. The query matrix of the transformer module is learnt during the training process and contains N representations used to identify the N masks on the image. The output of the transformer module is a matrix $C_Q \times N$ and is passed on to an MPL which converts it to a matrix of dimension $C_E \times N$. The output is also passed on to a linear classifier and a softmax to generate the N class predictions per mask. The pixel decoder up-samples the low-resolution features to produce high resolution per-pixel embeddings. These embeddings of size $W \times H \times C_E$ are multiplied with the output of the MLP thus generating a result of size $W \times H \times N$ which contains the mask probability for every pixel after a sigmoid is applied on it.

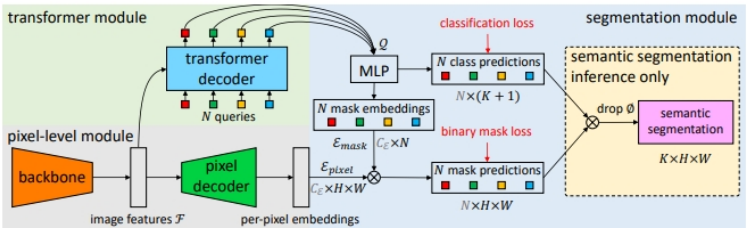


Figure 3.3.3: MaskFormer Architecture from corresponding paper [19]

Therefore, there are two segmentation branches: one that classifies masks using only the backbone and the transformer decoder and one that finds the mask region using product of image features from pixel decoder and mask embeddings from transformer decoder. The final output of the mask classification task is the class predictions per mask and the per pixel mask predictions($m_i[h,w]\epsilon[0,1]$ probability the ith mask contains the h,w pixel). The two outputs are combined to produce the output of the semantic segmentation and the panoptic segmentation task. As for the semantic segmentation, the pixel classification is calculated by assigning to each pixel the class that produces the highest product of mask probability and class probability at the specific mask,ie. $\arg\max_{c\epsilon\{1,\dots,K\}} \sum_{i=1}^{N} p_i(c)m_i[h,w]$. Then pixels with the same class are merged. Whereas, in instance segmentation we keep both class and mask, ie. $\arg\max_{i\epsilon\{1,\dots,N\}} p_i(c_i)m_i[h,w]$ where $c_i = \arg\max_c p_i(c)$ grouping pixels belonging to the same mask.

Loss Calculation

The classification error is a combination of mask classification loss calculated with cross entropy and binary mask loss. However, in order to calculate loss a matching between the M ground truth results and the N MaskFormer results must be made. The balance between the two sets is possible by introducing N-M null objects. Thus with bipartile matching it is possible to perform one-to-one matching and calculate the loss:

$$L_{mask-cls}(z, z^{gt}) = \sum_{j=1}^{N}[-\log p_{\sigma(j)}(c_j^{gt}) + \mathbb{1}_{c_j^{gt}\neq\emptyset} L_{mask}(m_{\sigma(j)}, m_j^{gt})\}] \tag{3.3.1}$$

Where z is the prediction set, z^{gt} is the ground truth set, and N is $|z|$. L_{mask} is mask loss which due to the indicator function used as a factor, it is calculated only when the mask predicts a certain object of a class. The first term is a cross entropy classification loss of the masks.

Modules used

The pixel-level module is consisted of a backbone and a pixel-decoder which together can be implemented from any current per-pixel segmentation model. That is MaskFormer's continuity, the fact that it can be used to transform any SOTA model 3.3.4 into a mask-classification model, which is proved in this paper [19] to improve performance of single-task oriented models. Usually used as backbone are ResNet [81] or Swin-Transformer [57] models and as pixel-decoder one designed based on FPN [82].

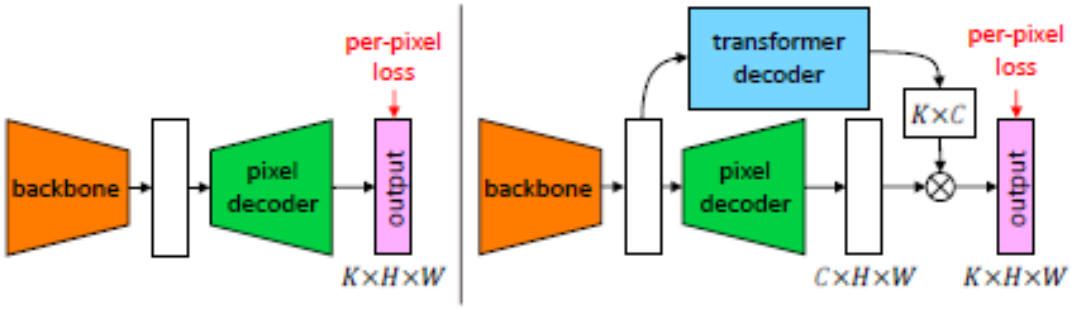


Figure 3.3.4: Conversion of a per-pixel segmentation model into a mask classification model shows in study of [19] that the later produces improved results.

The transformer decoder module is borrowed from DETR [58]. It has N query embeddings initialized to zero vectors, and every query is associated with a learnable positional encoding. By default, 6 Transformer decoder layers with 100 queries and the same loss is applied after each decoder.

The segmentation module is made of a multi-layer perceptron (MLP) which has 2 hidden layers of 256 channels to predict the mask embeddings. Both per-pixel and mask embeddings have 256 channels.

Mask2Former: Masked Attention

In an updated version of the MaskFormer, Mask2Former [59], the concept of masked attention is introduced replacing 3.3.5 cross-attention in the pixel decoder and making the architecture state of the art in semantic, panoptic, and instance segmentation. The slow training process and the high complexity of the MaskFormer led to the concept of mask attention where the calculation of cross attention in the transformer decoder limits in the predicted area of the mask by the previous decoder layer. Context information are still usefull to the classification of masks, however, cross-attention requires large part of training time to work as intended. Thus, it is preferred to limit mask-attention to attend on proposed mask features and use self-attention to introduce context information. The order of cross-attention and self-attention alternates in every layer.

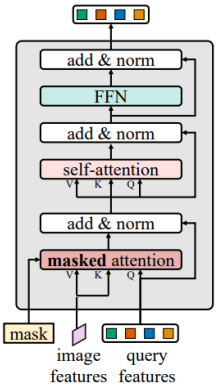


Figure 3.3.5: Mask2Former μετασχηματιστής αποκωδικοποίησης [59]

In cross attention 3.3.2, the whole input vector query features are used in calculations. At every step produces their updated version. The second term is the residual connection of the features of the previous stage to the newly predicted features. The first term is the soft max result of the activation of queries on features for the specified keys. Then it is multiplied with the value vector that corresponds to those keys. The key and value vectors are produced from the image features at every stage through the linear transformations fk, fv. The difference between self-attention and cross-attention is the input. In self-attention the same input is used in the calculation of the K,Q,V matrices but in cross-attention a different input is used for the calculation of the K,V matrices and another for the calculation of the Q matrix. In this instance, the first input is the image features produced from the different resolution levels of the pixel decoder and the second input is the second input is the trainable query features.

$$\begin{aligned} X_l &= \text{softmax}(Q_l K_l^T) V_l + X_{l-1} \\ Q_l &= f_Q(X_{l-1}) \\ K_l &= f_K(I) \\ V_l &= f_V(I) \\ &\textit{where } I \textit{ are the image features.} \end{aligned} \tag{3.3.2}$$

Where l is the decodel layer, $X_l \in \mathbb{R}^{N \times C}$ the N query features at layer l, X_0 the initial trainable query features, and $K_l, V_l \in mathbbR^{H_l W_l \times C}$ where H_l, W_l the resolution of the image in the specific resolution level. In mask attention 3.3.3, the difference is that M'$_{l-1}$ is added in the softmax arguments. The proposed mask M$_{l-1}$ is the resized prediction binarized on the threshold of 0.5. M'$_{l-1}$ is $-\infty$ for non-mask pixels and 0 for mask pixels where mask pixels are refreshed at every transformer decoder layer. Thus, it imitates not using non-mask features in attention and causes $X_l = X_{l-1}$ in pixels that don't belong in the mask. In this case, query features(X) are learnable.

$$\begin{aligned} X_l &= \text{softmax}(\text{M'}_{l-1} + Q_l K_l^T) + X_{l-1} \\ \text{M'}_l(x,y) &= \begin{cases} 0 & \text{M}_l(x,y) = 1 \\ -\infty & else \end{cases} \end{aligned} \tag{3.3.3}$$

As mentioned above, Mask2Former introduces multi-resolution features to the transformer decoder. Specifically, it includes L consecutive groups of 3 consecutive transformer layers where every layer takes as input $\frac{H}{32} \times \frac{W}{32}$, $\frac{H}{16} \times \frac{W}{16}$, and $\frac{H}{8} \times \frac{W}{8}$ respectively. These resolutions come from different stages of the pixel decoder with applied sinusoidal scale-level embedding. A multi-scale Deformable attention module [16] is used as a pixel decoder which derives from DETR [58].

Other modifications to the MaskFormer are: calculation of mask loss at sampled points of the mask reducing significantly training time, removing dropout, and introduction of learnable query features.

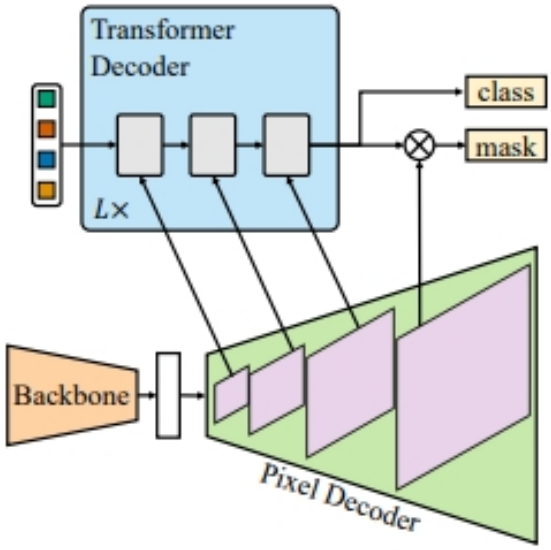


Figure 3.3.6: Mask2Former Multi-Resolutional Architecture from corresponding paper [59]

Detection Transformer (DETR) The DEtection TRansformer [58] is an architecture created for object detection. It was the first model designed to perform end-to-end object detection. Combining both CNNs and transformers it performs parallel decoding of input. As shown in 3.3.7, the DETR contains a CNN backbone which produces a lower resolution activation map from the input image. This map is then forwarded to an encoder-decoder architecture composed of transformer modules.The input is combined with positional embeddings since the transformer architecture is permutation invariant. The encoder transformer is made of a multi-head self-attention module and a feed-forward network(FFN). It produces N embeddings which are then processed by the transformer decoder which produces the output embeddings. The transformer decoder uses multi-headed self and encoder-decoder attention mechanisms. The output embeddings are then forwarded to the FFN layer which converts them to rectangle coordinates and class.

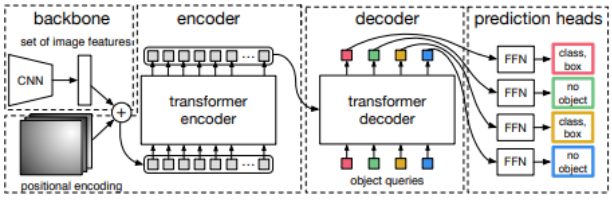


Figure 3.3.7: DETR Architecture from corresponding paper [58]

However, this architecture showed disadvantages in performance compared to other architectures. These included: it required much longer epochs than the existing object detection models and doesn't perform well in the detection of small models. Deformable DETR [16] was introduced as an improvement of DETR on the

previous downsides. It introduced deformable attention inspired by deformable convolution which replaces attention modules of the transformers. Deformable attention performs only on a small fixed set of sampling points predicted from the feature of query elements. Therefore, DETR's transformer encoder and decoder attention modules are replaced by multi-scale deformable attention modules. The multi-scale deformable attention module used in Mask2Former is very similar to the previous single-scale version, except that it samples LK points from multi-scale feature maps instead of K points from single-scale feature maps.

3.3.3 High Resolution Transformer (HRFormer)

The HRFormer [60] is an implementation of the HRNet [56] using Transformers. This architecture preserves the notion of the four parallel streams, their fusion after every stage, and the addition of a new stream after every stage. Staring with one high resolution stream every stage consists of a number of consecutive modules processing every stream independently and at the end of every stage every stream fuses its result with the upsampled and downsampled results of every other stream. Lastly, a new stream is added by downsampling of the lowest resolution stream at the end of every stage. Every module consists of a transformer which applies multi-head self-attention and specifically interlaced sparse self-attention which is computationally faster and equivalent to self-attention. However, to account for the locality of the results of the self-attention mechanism on independent blocks and to also exchange information across non-overlapping windows, a 3×3 depth-wise convolution is added on the FFN to introduce contextual information. The semantic segmentation result is extracted by concatenating all the representations from every stream upsampled to the highest resolution and then forwarded to a semantic segmentation head such as OCRNET [83].

Object Contextual Representation Network(OCRNET)

The OCRNET [83] is a method that refines the segmentation map from coarse to fine. It is designed based on the idea of the pixel classification depending on the contextual information of the image and the region the pixel belongs to. Thus, it aims to augment the initial features of a pixel with the weighted features of the regions surrounding it. Therefore, it requires a coarse prediction of the regions in an image used to describe every pixel separately. That is obtained from any semantic segmentation backbone such as HRFormer [60], HRNet [56] etc supervisely trained on a dataset. Then, the regions are described by a set of features extracted from the pixels they contain. The pixels are again described by their initial features augmented from the weighted addition of the new region features based on proximity. Similar previous methods have not taken advantage of regions and have not differentiated relationships of pixels based on their class. Instead, they weigh pixel relationships based on proximity. As mentioned, every pixel p_i is assigned to a class K from a backbone segmentor. Every region is then described from all the pixels with a weighted summation of their features based on the degree of belonging to this specific region as such:

$$f_k = \sum_{i \epsilon D} m_{ki} x_i \tag{3.3.4}$$

Where f_k is the representation of the region, x_i the representation of the pixel, and m_{ki} the spacial softmax degree for pixel p_i belonging to region k. The relationship between every pixel and every region is quantified as seen in 3.3.5 inspired from self-attention.

$$w_{ik} = \frac{e^{\kappa(x_i, f_k)}}{\sum_{j=1}^{K} e^{\kappa(x_i, f_j)}} \tag{3.3.5}$$

Where $\kappa(x_i, f_k) = \phi(x)^T \psi(f)$ is the unnormalized relation function with $\phi()$ and $\psi()$ two transformation functions implemented by 1x1 conv $\rightarrow$ BN $\rightarrow$ ReLU. The object contextual representation y_i of the pixel p_i is extracted from this equation:

$$y_i = \rho(\sum_{k=1}^{K} w_{ik} \delta(f_k)) \tag{3.3.6}$$

Finally, the representation of every pixel is the augmented representation of its initial features and its y_i features.

$$z_i = g([x_i^T \ y_i^T]) \tag{3.3.7}$$

Where g is a transform used to fuse the two representations made of 1x1 conv $\rightarrow$ BN $\rightarrow$ ReLU. The above procedure is summed in 3.3.8.

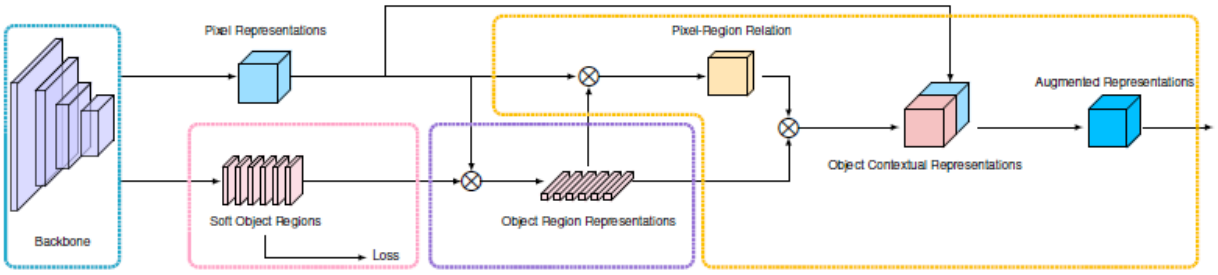


Figure 3.3.8: OCRNET architecture from corresponding paper [83]

3.3.4 Segmentation Transformer (SETR)

Segmentation Transformer or SETR [71] attempts to replace the encoder-decoder FCN architecture that has been used in semantic segmentation. The problem of information loss and dimension reduction is due to repeated convolutions is its main focus. Additionally. it has been observed that convolutions do not detect long range dependencies unless there is an extensive number of layers. After ViT [70], many architectures have approached semantic segmentation with transformers and self attention revealing the superior characteristics of the transformers architecture in image processing. SETR uses explicitly transformers and attempts to proces the image in a sequential manner on patches. Because the number of pixels in an image is too large to

be processed sequentially individually, it is divided into patches. These patches are mapped into a latent C-dimensional embedding space using a linear embedding projection function. This results into a 1D sequence of patch embeddings of size $L = \frac{HW}{256}$. To those embeddings e_i, p_i is added to encode positional enformation resulting in a final input sequence $E = \{e_1 + p_1, ..., e_L + p_L\}$. A pure transformer encoder is utilized to learn feature representations and has a global receptive field. It consists of L_e layers each containing a multi-head self-attention block and a multilayer perceptron module. Its input is a query, a key, and a value matrix calculated from the input of the previous layer and learnable query W_Q, key W_K, and value W_V matrices.

$$query = Z^{l-1}W_Q, key = Z^{l-1}W_K, value = Z^{l-1}W_V \tag{3.3.8}$$

Then self-attention is calculated as mentioned in 2.2.2 and uses residual connections. Then the decoder transforms the input to the final results of shape $H \times W$ before it first converts them to a 3D feature map of shape $\frac{H}{16} \times \frac{W}{16} \times C$. There are 3 possible decoders used: a) Naive: uses a simple bilinear interpolation to upsample b)Progressive UPsampling; alternates between upsampling layers and convolutional layers c)Multi-Level feature Aggregation: uses multi-level feature aggregation similar to the feature pyramid network [84] [82] to gradually upsample.

3.3.5 OneFormer

OneFormer [24] is a new multi-task universal image segmentation architecture that needs to be trained only on one panoptic dataset and it can be used to output results on all image segmentation categories. Given a sample image and a task input of the form "this task is a task", this model will output the result of semantic, instance, or panoptic segmentation. The unification of the three tasks was attempted before but never in this extend. Panoptic segmentation was introduced to unify the two tasks, however, introduced a new one and did not manage to separate them. Later, Mask2former [59], MaskFormer [19], and K-Net [85] made a step forward to unifying these tasks. They initiated the concept of a universal architecture, however, they still required individual training on the different tasks. Inspired from Mask2Former's architecture, OneFormer adopted the encoder-decoder architecture and the transformer module but added a few extra transformer modules to enhance processing with text description.

Input

As seen in 3.3.9 the input consists of the image in 3.3.9.a and a text T_{task} of the form "the task is task" in 3.3.9.b. Then T_{task} is tokenized and Q_{task} is extracted to condition OneFormer to the task. During training, a text list T_{list} is created from GT annotations and inputted to the module in 3.3.9.b.

Text queries

The ground truth for every task is different, so for every sample the task is selected uniformly and the ground truth is extracted from the panoptic annotations. This text list (T_{list}) contains the ground truth whose every line corresponds to one binary mask. For every class(thing or stuff) a line is added to the text list of the form "a photo with a {CLS}" where CLS is the class of the binary mask. A line of the form "a/an {task} photo" is added for every null binary mask which is the padding text T_{pad}. T_{list} is inputted to a text mapper to produce the N_{text} embeddings which are concatenated with N_{ctx} context embeddings and together they form the text queries Q_{text}. Q_{text} is used to calculate the loss between that and the object queries Q.

Object queries

The image passes through the backbone and the pixel decoder in 3.3.9.a just like in Mask2Former. Multi-resolutional outputs are produced from which the $\frac{1}{4}$ resolution is forwarded to the 3.3.9.b. There it is processed from a 2-layer transformer and updates the N-1 object queries. The object queries are initialized with N-1 repetitions of the task token before being updated. After being updated, Q_{task} is concatenated to them creating N queries. They are then transferred to the transformer decoder along with the multi-resolutional outputs to produce the final result similarly to Mask2former.

Query-text constractive loss

A loss is calculated between Q_{text} and Q in 3.3.9.b. Pairs of object and text queries are formed and the similarity is measured with their dot product. Two losses are calculated and summed: a) object-to-text contrastive loss $L_{Q \to Q_{text}}$, b)text-to-object contrastive loss $L_{Q_{text} \to Q}$

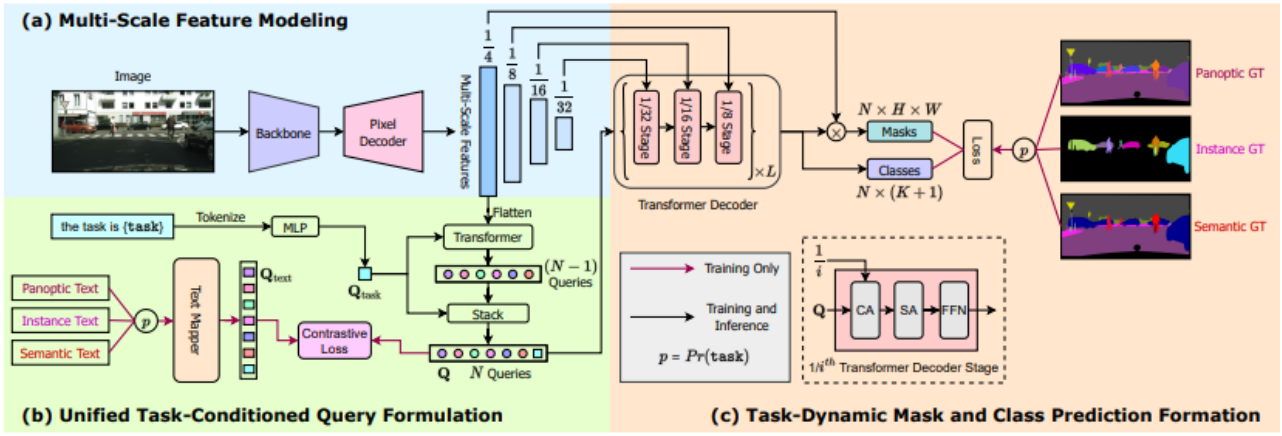


Figure 3.3.9: OneFormer architecture from corresponding paper [24]. a)Encoder-decore module just like in Mask2former[59], b)Task text(T_{task}) is inputted along with task specific GT annotations formulated in a text-list T_{list}. Q_{task} and Q_{test} are produced and loss is calculated between text representation queries Q_{test} and image representation queries Q. c)Output is produced from transformer decoder just like Mask2former.

Figure 3.3.10: OneFormer architecture text input from corresponding paper [24]. Text input depending on task is formulated in a text list T_{list} where every line corresponds to a binary mask. The line contains a phrase of the form: "a photo with a {CLS}" where CLS is the class of the binary mask or of "a/an {task} photo" if the binary mask is null where {task} is the name of the task.

Chapter 4

Proposal: Multiple resolution streams in Mask2Former

4.1 Multiple resolutions in semantic segmentation

Simpler tasks such as object classification were successfully performed with encoder-decoder type architectures. That was possible due to the nature of both the task and the architecture of transforming input into a smaller dimension. However, after the designation of more complex tasks, such as semantic segmentation, where the dimension of input is equal to the dimension of the output a need of more complex architectures surfaced. Specifically, it is needed to process data in multiple scales and detect classes in multiple scales. This observation eventually led to the introduction of multi-resolutional models such as HRNET [56], HRFormer [60], HRSTFormer [61] which were observed to perform better than their corresponding single dimensional models. These models follow the basic idea introduced by HRNET shown in 4.1.1. There are four resolution streams which are added gradually to the model after every stage. In every stream the resolution is kept constant throughout the processing stages of the input. Every stage is followed by a fusion module which exchanges information between the different resolutions. Every stage mimics the single-resolutional network by replicating its corresponding stage modules across all resolutions.

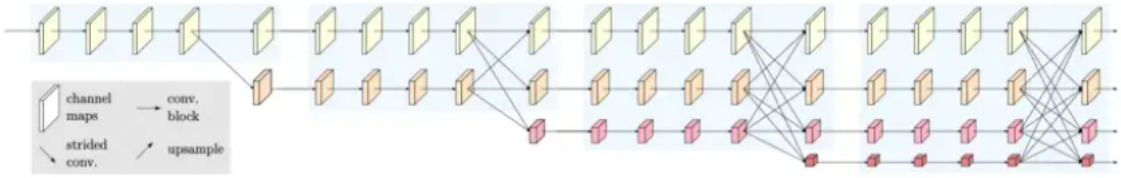


Figure 4.1.1: HRNet Architecture from [50]

4.2 Introducing multiple resolutions to Mask2Former

The MaskFormer [19] predicts a set of N pairs of class and mask, $z = \{(c_i, m_i)\}_{i=1,\dots,N}$ performing thus primarily mask classification and then converts the results to panoptic, instance, or semantic segmentation output. As shown in 4.2.1, it is composed of a backbone, a transformer decoder, and a segmentation head. The backbone can be replaced with any architecture as the Mask2Former's main goal was to transform different segmentation architectures into a multi-purpose segmentation architecture based on masks. The Mask2Former, however, has currently been used only on single-resolutional backbones such as R-50, R101, Swin-T, Swin-S, SWin-B, Swin-L in Cityscapes dataset which is our dataset of focus.

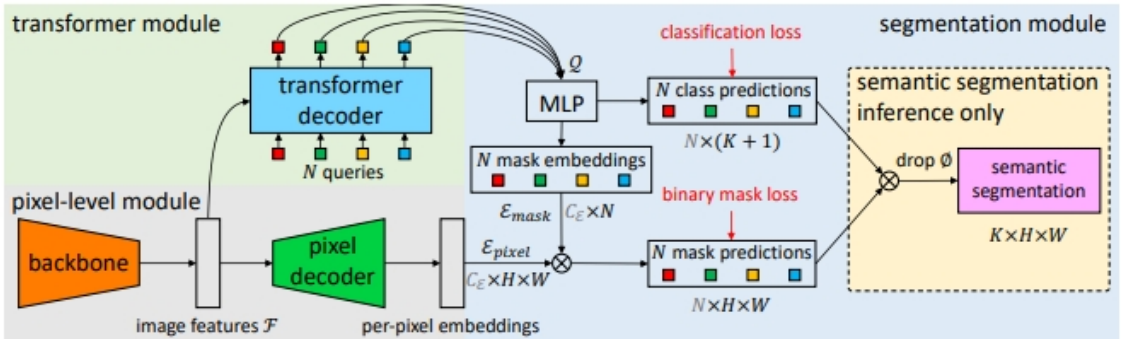


Figure 4.2.1: MaskFormer Architecture from corresponding paper [19]

Mask2Former with Swin trasformer(Swin-T) backbone has an architecture as the one displayed in detail in 4.2.2. It is paired with the Multi-Scale Deformable attention module [86] used as the decoder. The proposed backbone architecture -SwinHR- follows the four layer structure of the initial backbone architecture described in subsection 3.3.2 by replicating every layer in parallel streams. In order to keep the resolution constant, the downsampling modules are eliminated in the layers. Additionally, after every stage one stream is introduced by downsampling the lowest resolution stream. downsampling is done with patch merging

modules as explained in section 3.3.1 . The stages are in total four which is equivalent to the number of streams. The resulting architecture is shown in 4.2.3 and 4.2.4 which differ in the point where the input of the pixel decoder is taken from the backbone.

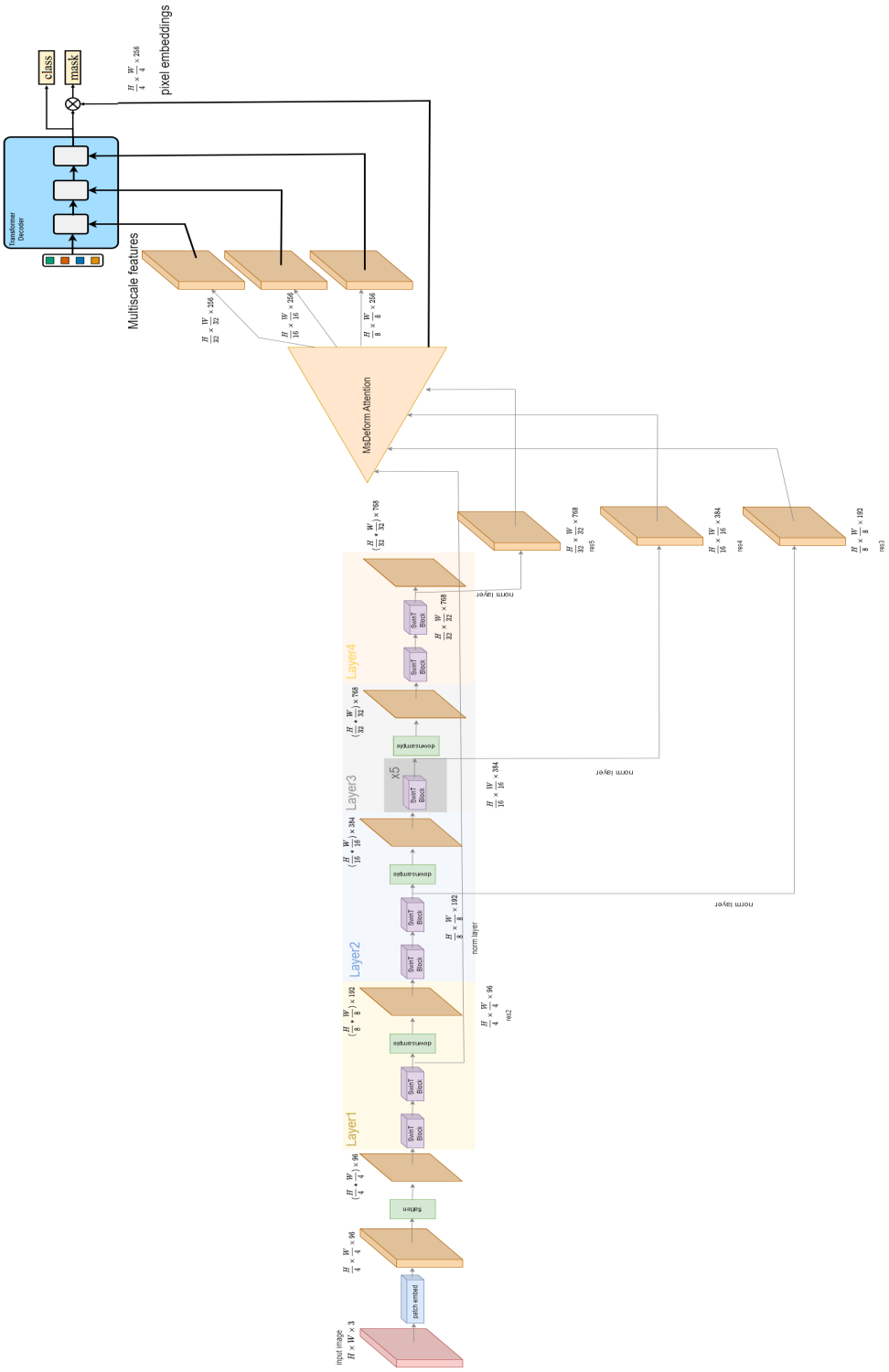


Figure 4.2.2: Mask2Former Architecture

In the original Mask2former-Swin Transformer architecture, input of the transformer decoder is taken from

the output of the pixel decoder. However, the notion behind moving that input further back in the new architecture is based on the stand-alone performance of the multi-dimensional model. This new backbone model does not require a decoder as it is replacing the encoder-decoder as a whole. However, the decoder is kept intact as it will decode the per pixel embeddings which will fuse with the pixel decoder output. Thus, the Multi-Scale Deformable attention module [86] module is present in the new model.

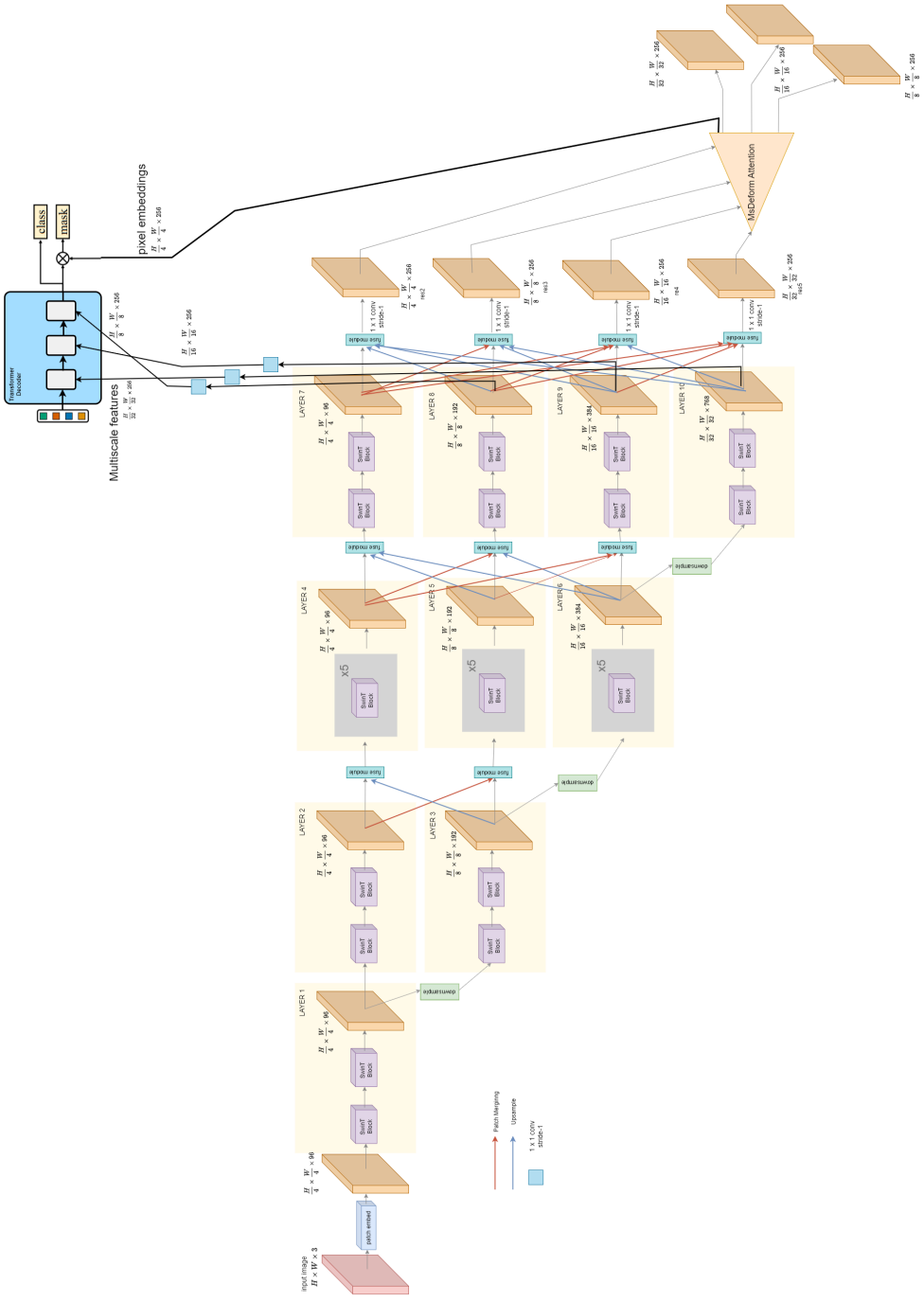


Figure 4.2.3: Mask2Former Multiple Resolutions Architecture

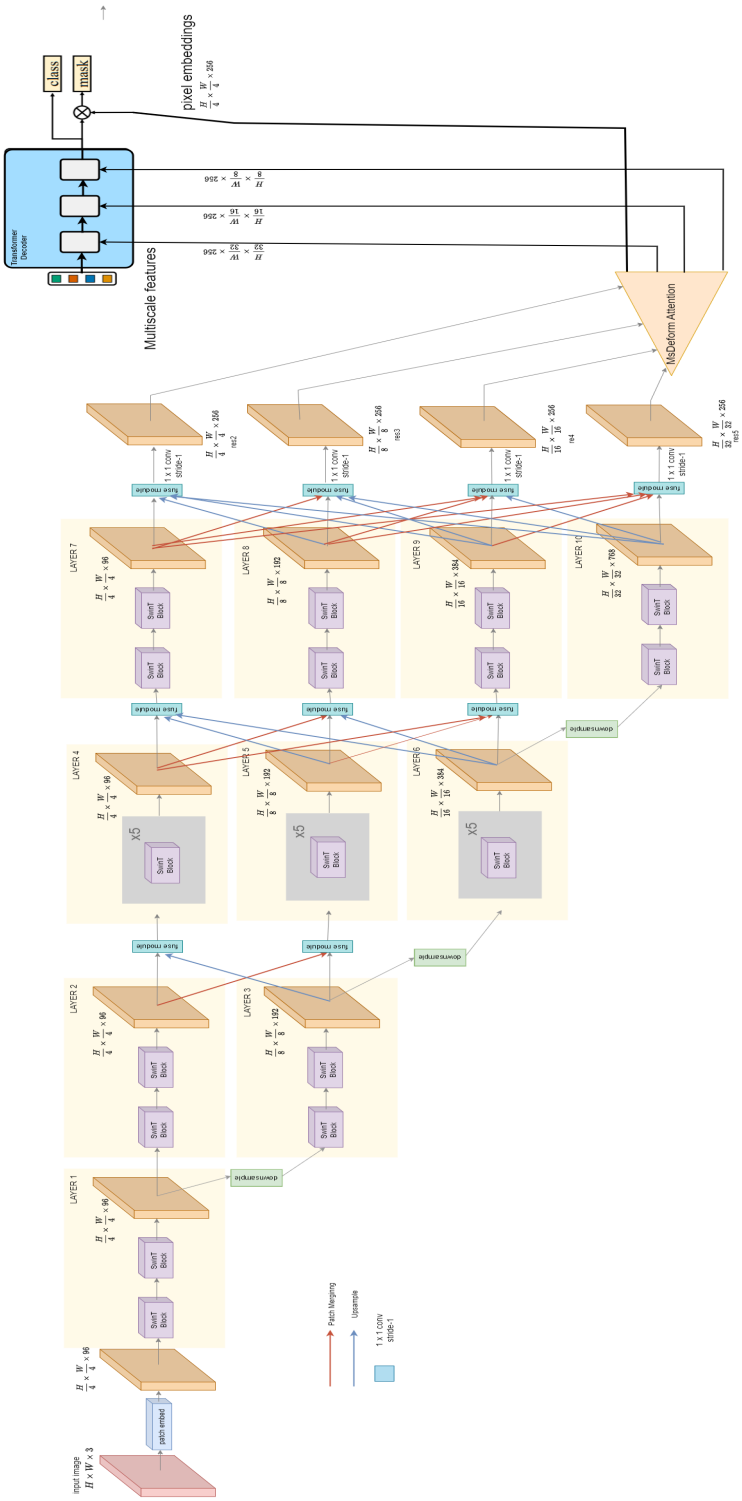


Figure 4.2.4: Mask2Former Multiple Resolutions Architecture alternate design : Transformer decoder input comes from the output of the pixel decoder

4.2.1 Swin Transformer

The Swin Transformer attempts to scale down the processing time of a regular transformer when calculating self-attention on an image. This was possible by dividing images in windows and calculating self-attention in

every window separately also known as windowed self-attention. However, in order to cross boundaries defined by those windows on neighbouring pixels, at every consecutive swin transformer layer those windows are shifted and reformed. The initial window is placed on the left-top corner and the rest are placed consecutively in an adjacent non-overlapping way. The next formation is created by shifting diagonally the windows. However, this results in more windows than initially with some uncompleted. The solution given to this problem is the cyclic-shifting method where uncompleted windows from the top left are moved to the bottom right and connected with the uncompleted windows there to form regular windows. Afterwards, self-attention applied in these concatenated windows takes into account this shift by masking features in the window that weren't neighbouring before the shift. As shown in 4.2.5, two consecutive swin transformer blocks are connected and the second one applies the shifted windows of the first. Also, in every block there is a residual connection and an MLP after the attention module. Before every module, layer normalization is applied. The passing of an input image z^{l-1} through two consecutive transformer blocks is described as such:

$$\begin{aligned} \hat{z}^l &= W - MSA(LN(z^{l-1})) + z^{l-1} \\ z^l &= MPL(LN(\hat{z}^l)) + \hat{z}^l \\ \hat{z_{l+1}} &= SW - MSA(LN(z^l)) + z^l \\ z^{l+1} &= MLP(LN(\hat{z_{l+1}})) + \hat{z_{l+1}} \end{aligned} \tag{4.2.1}$$

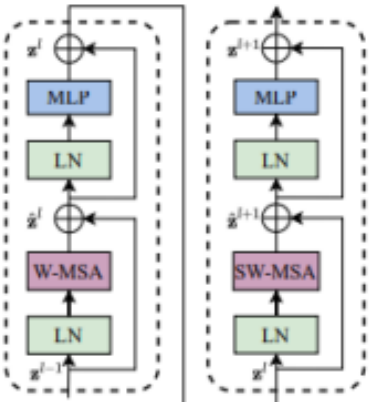


Figure 4.2.5: Two consecutive Swin Transformer modules

Attention

In transformers, the attention mechanism attempts to match a query to the nearest key and retrieve the corresponding value. The "nearest" quantity is calculated from a softmax layer after multiplying the query matrix with the key matrix. In order to find the corresponding value , the value matrix is multiplied with the previous probability matrix. Thus, the resulting value is a weighted sum of the values and the attention process is shown in .

$$Attention(Q, K, V) = Softmax(QK^T/\sqrt{d} + B)V \tag{4.2.2}$$

where $Q, K, V \in R^{M^2 \times d}$ are the query, key , and value matrices respectively, d is the query/key dimension, M^2 is the number of patches in a window and window is of dimension $M \times M$. The matrix B is the relative position bias

4.2.2 Fusion layers

After every stage the different streams fuse their outputs in order to exchange information. This fusion occurs at the end of every stream and every other stream transforms their output to match the resolution and channels of the target stream. The upsampling module depicted as a red arrow in 4.2.3 is composed of a convolution that changes the number of channels with a kernel size 1×1 and a stride 1×1 and an upsampling module of a scale factor 2 , 4 or 8 depending on the case as shown in 4.2.6. The downsampling module shown in 4.2.7 is composed of replication of a base module that downsamples with a factor of two. This base module is composed of two convolutions: one with stride 2×2 and kernel size 3×3 aiming to downsample with a factor of 2, and one with stride 1×1 and kernel size 1×1 aiming to convert the number of channels. If there are more than one replications of this base module, the second convolutional layer changes the number of channels only on the last instance of this module and between two modules there is a ReLU layer.

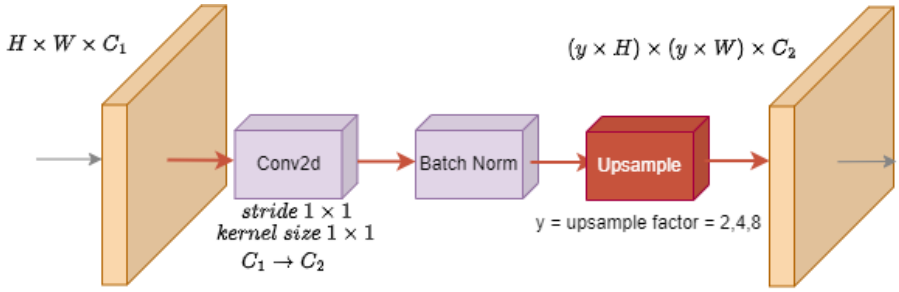


Figure 4.2.6: Upsample module

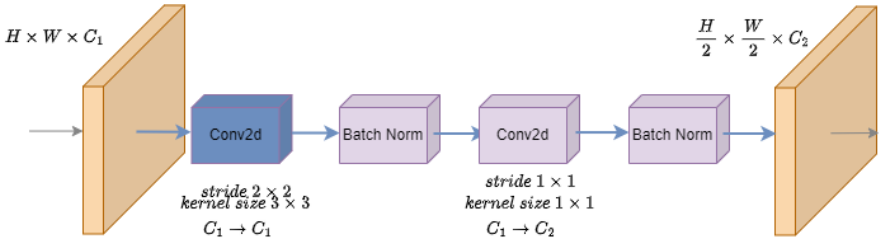


Figure 4.2.7: Downsample module: stacking of these modules achieves higher downsampling factor

Supposedly there are 3 streams, their fusion is an addition of their transformed outputs at a specific resolution. For O_s^r the ouput of stage s at resolution r after the fusion layers and for R_s^r the representation of output at stage s resolution r before the fusion layers we have:

$$O_s^r = \sum_{i=1}^{s} f_{rk}(R_s^k) for\ r \leq s \tag{4.2.3}$$

Where f_{rk} is the transform function from resolution k to resolution r. The transform function as explained above with modules is :

1. If $r = k$, $f_{rk}(R) = R$.
2. If $r < k$, $f_{rk}(R)$ downsamples the input representation R through $(r - s)$ stride 2 and 3×3 kernel size convolutions and a 1×1 convolution for aligning the number of channels.

3. If $r > k$, $f_{rk}(R)$ upsamples the input representation R through the bilinear upsampling followed by a 1×1 convolution for aligning the number of channels.

4.2.3 Semantic Segmentation Head

The output of the transformer module is a matrix $C_Q \times N$ and is passed on to an MPL which converts it to a matrix of dimension $C_E \times N$. The output is also passed on to a linear classifier and a softmax to generate the N class predictions per mask. The pixel decoder produces high resolution per-pixel embeddings. These embeddings of size $W \times H \times C_E$ are multiplied with the output of the MLP thus generating a result of size $W \times H \times N$ which contains the mask probability for every pixel after a sigmoid is applied on it. Therefore, there are two segmentation branches: one that classifies masks using only the backbone and the transformer decoder and one that finds the mask region using product of image features from pixel decoder and mask embeddings from transformer decoder. The final output of the mask classification task is the class predictions per mask and the per pixel mask predictions($m_i[h,w]\epsilon[0,1]$ probability the ith mask contains the h,w pixel). The two outputs are combined to produce the output of the semantic segmentation and the panoptic segmentation task. As for the semantic segmentation, the pixel classification is calculated by assigning to each pixel the class that produces the highest product of mask probability and class probability at the specific mask,ie. $\arg\max_{c\epsilon\{1,\dots,K\}} \sum_{i=1}^{N} p_i(c)m_i[h,w]$. Then pixels with the same class are merged. Whereas, in instance segmentation we keep both class and mask, ie. $\arg\max_{i\epsilon\{1,\dots,N\}} p_i(c_i)m_i[h,w]$ where $c_i = \arg\max_c p_i(c)$ grouping pixels belonging to the same mask.

Loss Calculation

The classification error is a combination of mask classification loss calculated with cross entropy and binary mask loss. However, in order to calculate loss a matching between the M ground truth results and the N MaskFormer results must be made. The balance between the two sets is possible by introducing N-M null objects. Thus with bipartile matching it is possible to perform one-to-one matching and calculate the loss:

$$L_{mask-cls}(z, z^{gt}) = \sum_{j=1}^{N} [-\log p_{\sigma(j)}(c_j^{gt}) + \mathbb{1}_{c_j^{gt} \neq \emptyset} L_{mask}(m_{\sigma(j)}, m_j^{gt})\}] \tag{4.2.4}$$

Where z is the prediction set, z^{gt} is the ground truth set, and N is $|z|$. L_{mask} is mask loss which due to the indicator function used as a factor, it is calculated only when the mask predicts a certain object of a class. The first term is a cross entropy classification loss of the masks.

4.2.4 Using OCRNet as Pixel Decoder

Looking at HRFormer [60], it is observed that it also converts the Swin Transformer into high resolution streams based on HRNet architecture. The HRFormer performs semantic segmentation using OCR [83] as a semantic segmentation head. Therefore, an alternate design of the previous proposed architectures that was tested is shown in 4.2.8 and it replaces MsDeform attention module with OCR.

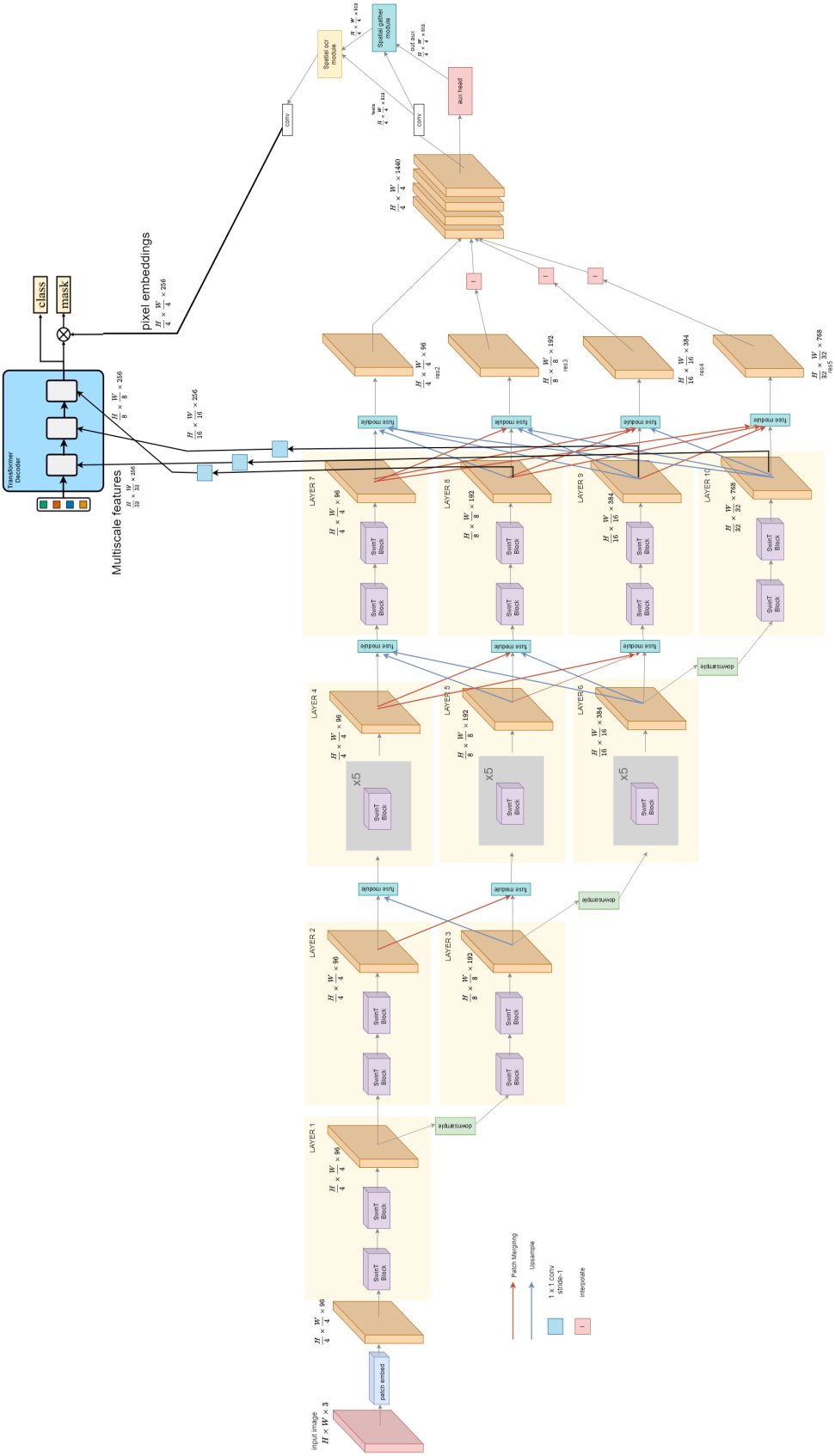


Figure 4.2.8: Mask2Former with OCR pixel decoder

Chapter 5

Experimental Results

5.1 Datasets

5.1.1 Cityscapes Dataset

The Cityscapes Dataset [8] is a dataset specifically designed for autonomous driving in urban environments applications. It contains images of road scenery from 50 different city centers over all 4 seasons. These images are available in low(8 bit) and high(16 bit) resolution. The dataset includes both pixel-level and instance level annotations provided in two groups- coarse and fine annotations. Fine annotations are provided in 5000 images over 27 cities where all pixels have been labeled by creating polygons of instances. Te rest of the images taken in 23 cities are used in coarse annotations where the polygons were chosen much faster causing lower accuracy labelling. In total 30 classes are included and 19 of them are used in evaluation as seen in 5.1.1.

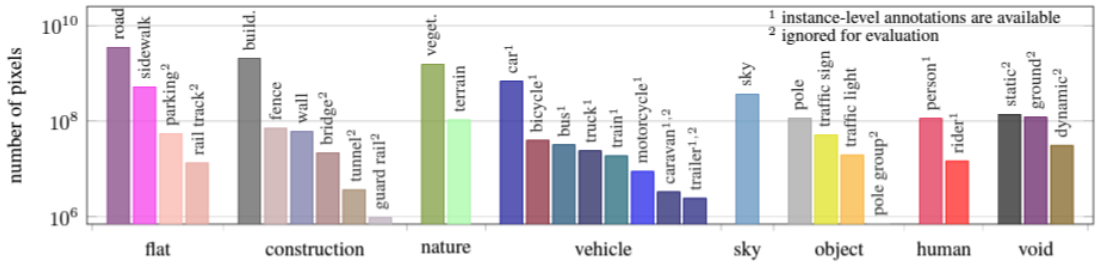


Figure 5.1.1: Cityscapes Classes, image from [19]

This dataset depicts much broader representation of inner city roads taking into account traffic and diverse climatic conditions with a wide variety of classes making it unique in the area of autonomous vehicle driving. Other datasets such as Kitti provides images of roads but from suburban areas while CamVid and DUS provide less images from only one city. Also concerning instance-level annotations Cityscapes is the only dataset among these providing instances of people and vehicles. The only dataset more complete than the Cityscapes one is the Mapillary Vistas Dataset [65], however, it is too complex for the purpose of this project as it is five times the size of the Cityscapes dataset including 66 classes with images taken from all over the world with different capturing means.

5.1.2 ADE20K Dataset

ADE20k was designed after recognizing the need for a general dataset covering a variety of scenes and common objects. Datasets before ADE20k had a limited set of scenes such as Cityscapes [8] or covered few or insignificant classes of objects such as COCO[10] and Pascal [15]. In ADE20k there are 20,210 images in the training set, 2,000 images in the validation set, and 3,000 images in the testing set. All the images are exhaustively annotated with objects. Many objects are also annotated with their parts. For each object there is additional information about whether it is occluded or cropped, and other attributes. The images in the validation set are exhaustively annotated with parts, while the part annotations are not exhaustive over the images in the training set. Additionally, parts can have parts too, and these are labeled with associations as well.

5.2 Evaluation Metrics

For semantic segmentation the standard metric is mIoU (mean Intersection over Union) [36], a per-pixel metric that directly corresponds to the per-pixel classification formulation. We use the highest achieved mIoU result as the metric of the performance of a model. Additionally, for Cityscapes iIoU(instance-level intersection over union) is used. For ADE20k fwIoU(frequency weighted intersection over union), mACC(mean accuracy) and pACC(probabilistic accuracy) are also used. These are all defined in 1.3.

5.3 Replicating baseline results

Our implementation is based on Mask2former source code [66] which uses Detectron2 [67]. The focus of the experiments below is on Cityscapes [8] and ADE20k [16] datasets.

5.3.1 Training settings

The official implementation used 8 GPUs and a batch size of 16. However, due to processing power limitations, in our implementation we used 4 GPUs and batch size of 4. Initial learning rate is 0.0001 with weight decay 0.05. The backbone was initialized with weights from image classification of Swin-T [57] on ImageNet-22k [9] and is assigned a learning rate multiplier of 0.1. AdamW [68] is used as an optimizer and poly [69] as a scheduler. Due to the inferior processing capabilities, in this experiment we trained the model on Cityscapes for 180k iterations while in the official implementation it was trained for 90k iterations. For ADE20k dataset both implementations were trained for 160k

5.3.2 Result Comparison

Displayed on 5.1 and 5.2 are the results achieved by training the Mask2Former model with our resources and the official results. The difference seen in performance is within reasonable limits for the difference in processing power.

	Cityscapes			
Model	$Official(mIoU)$	$Official(iterations)$	$Ours(mIoU)$	$Ours(iterations)$
Mask2former (Swin-T)	82.3	90k	81.70	180k
Mask2former (Swin-B)	83.3	90k	83.18	180k

Table 5.1: Performance of Mask2Former on Cityscapes with official and replicated implementation

	ADE20k			
Model	$Official(mIoU)$	$Official(iterations)$	$Ours(mIoU)$	$Ours(iterations)$
Mask2former (Swin-T)	47.7	160k	49.89	320k

Table 5.2: Performance of Mask2Former on ADE20k with official and replicated implementation

The following experiments results are only compared to our implementation performance as the processing power is identical.

5.4 Experimenting with SwinHR backbone

5.4.1 Without initialization

The Mask2FormerHR architecture described in 4.2 using SwinHR backbone was implemented. Initially trained without weight initialization the results were not matching the initialized original architecture shown in 5.4.1. Since the backbone was not initialized in the multiple resolutions architecture, the backbone multiplier is changed to 1 so that the backbone trains in the same rate as the rest of the model.

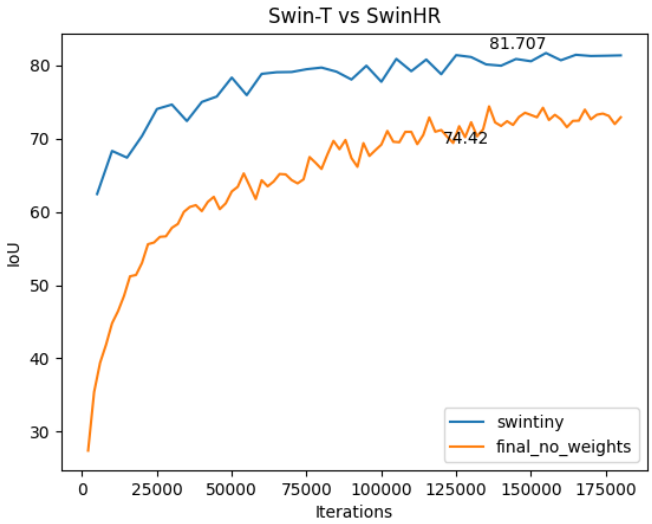


Figure 5.4.1: Mask2Former with Swin-T initialized and SwinHR uninitialized backbone

We observe that the model needs to be initialized for the results to compare to the official architecture. However, there are no weights from the trained version of the SwinHR in classification datasets.

5.4.2 Initialization techniques

Transferring Swin-T classification weights

Since the two backbones-Swin-T and SwinHR- are very similar, we attempt to transfer the trained weights of Swin-T on ImageNet. This can be done by observing that the diagonal modules of SwinHR as depicted in 5.4.2 are similar to Swin-T. Therefore, one proposed transformation is to initialize only the diagonal of the Swin-HR model and leave the rest of the backbone modules uninitialized.

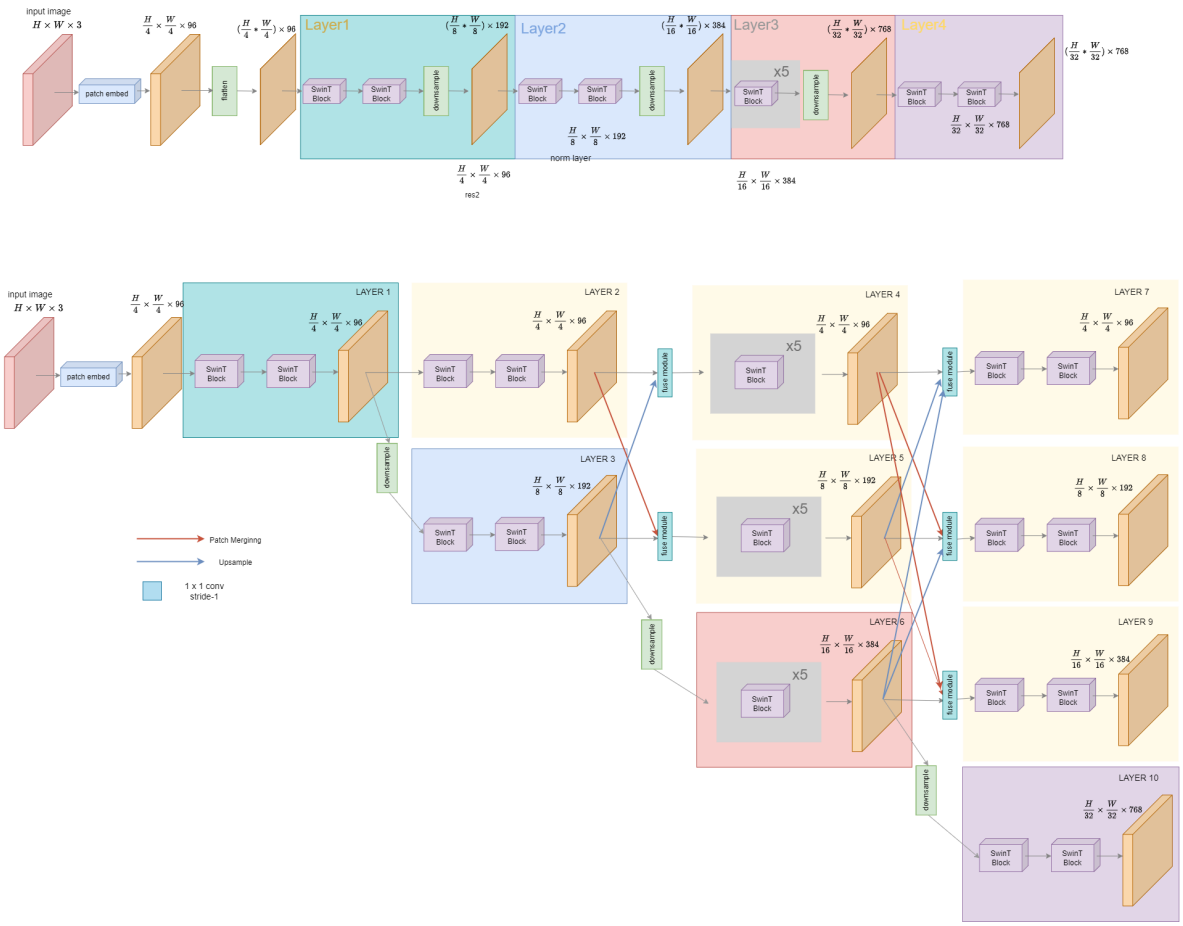


Figure 5.4.2: Weight transfer from Swin-T(top) to SwinHR(bottom). Module correspondence is shown with colored rectangles.

We experimented with different learning rate multipliers so that we freeze or hold back training of the initialized modules. First in 5.4.3, the initialized modules where assigned a 0.1 learning rate multiplier and the trained model almost matched the performance of the official model. Then in 5.4.4, a learning rate multiplier of 0.001 was applied to initialized modules and 0.1 to the rest of the backbone. Last in 5.4.5, no learning rate multiplier was applied to any of the modules and they are left to train in the same pace as the rest of the model. We observe, that leaving the model to train uniformly we obtain a better result that matches the official implementation. However, the performance of the model in the first iterations is much worse than the official model's initial performance. So we conclude that another initialization may be more fitting.

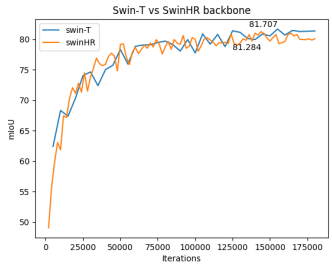


Figure 5.4.3: Weight transfer from Swin-T to SwinHR. Learning rate multiplier of 0.001 for initialized modules and 0.1 for the rest of the backbone.

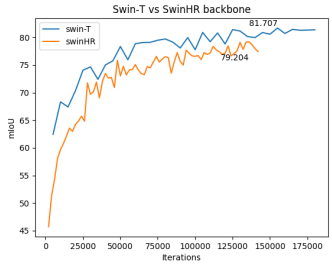


Figure 5.4.4: Weight transfer from Swin-T to SwinHR. A learning rate multiplier of 0.001 was applied to initialized modules and 0.1 to the rest of the backbone.

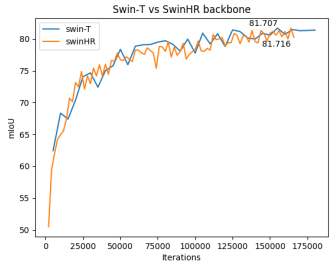


Figure 5.4.5: Weight transfer from Swin-T to SwinHR. No learning rate multiplier is applied.

Transferring Swin-T segmentation weights

Cityscapes We observe that if we were able to take as output from the backbone the same outputs that Swin-T produces, then we can start with the same performance of the original model. This would allow for the model to hit at minimum the maximum mIoU of the original model and possibly surpass it. Since the Swin-T backbone produces as output the output of every diagonal module of the SwinHR model, we attempt to find a weight initialization that would transfer the output of the first module of every stream intact to the end of the stream. That's possible if every weight of the rest of the transformer modules is zeroed and thus the input equals the output. Also, the fuse layers should add to a streams fusing point zeroed maps from other streams. This is possible by just assigning zeroed weights to the fuse layers and to the modules outside of the diagonal. Also, the initial learning rate should match the learning rate of the last iterations during the training of the official model. If not, the performance will diverge very easily and not produce the desired result. The initialized backbone layers have a learning rate multiplier of 0.01, while the rest of the backbone and the semantic segmentation head have a multiplier of 1.

Figure 5.4.6: Weight transfer from Swin-T trained on Cityscapes to SwinHR with zeroed modules.

This technique allowed for the multiple resolutions model to start as suspected around the maximum performance of the single-resolution model and surpass it by 0.1mIoU.

We observe that zeroed weights are not allowing the network to learn as they get stuck at zero specifically at the fuse layers which have ReLUs. Therefore, we open the fuse layers from the start by assigning weights equal to 1 so that we have fusion. Also, instictively we assign weights equal to 1 to the uninitialized weights aiming for the same results but with a better chance at learning. Lastly, we freeze both the initialized backbone weights and initialized semantic segmentation head weights, while the rest of the backbone has a learning rate multiplier of 1. Training occurs in two phases:

1. Using constant learning rate of 10^{-4}

2. Using constant learning rate of 10^{-5}

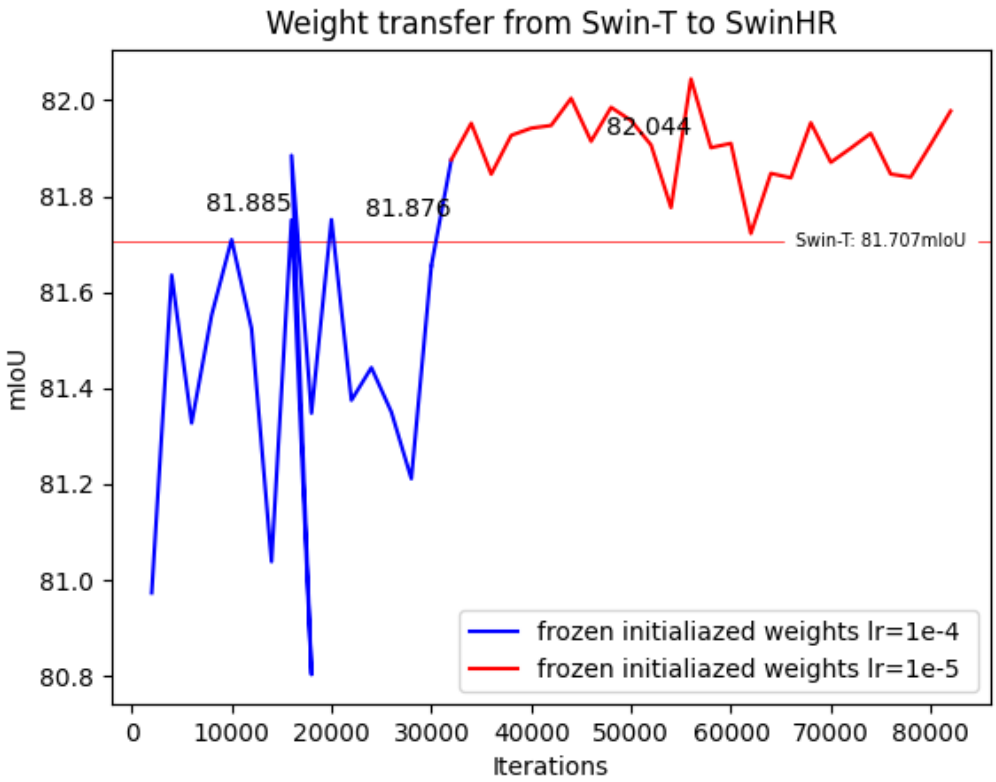


Figure 5.4.7: Weight transfer from Swin-T trained on Cityscapes to SwinHR with frozen modules.

This technique allowed for the multiple resolutions model to start as suspected around the maximum performance of the single-resolution model and surpass it by 0.3mIoU.

ADE20k This dataset seemed to overflow when training with the above mentioned techniques. A more attentive approach was taken. The uninitialized weights were initialized individually depending on their function. First, the fuse layers contain a Conv2D, a SyncBatchNorm, and depending on the case an Upsample module. The conv2D weights were taken down to 0.1 since they caused overflow after a few iterations. The SyncBatchNorm was initialized to 0 running mean and 0 running bias. For the Transformer modules we notice that the linear layers of MLPs are of the form:

$$y = xA^T + b \tag{5.4.1}$$

Where A is the weight vector and b the bias vector. Therefore, we initialize A as a unit matrix and b as a matrix of zeroes. Additionally, for the attention mechanism we assign a unit matrix for K, Q, and V weights and zero bias. Similarly, for all normalization layers weight was initialized to 1 and bias to 0. The following graph shows the training of the model with frozen all the layers whose weights have been transferred from Swin-T. Also, a constant learning rate of 0.0001 is used.

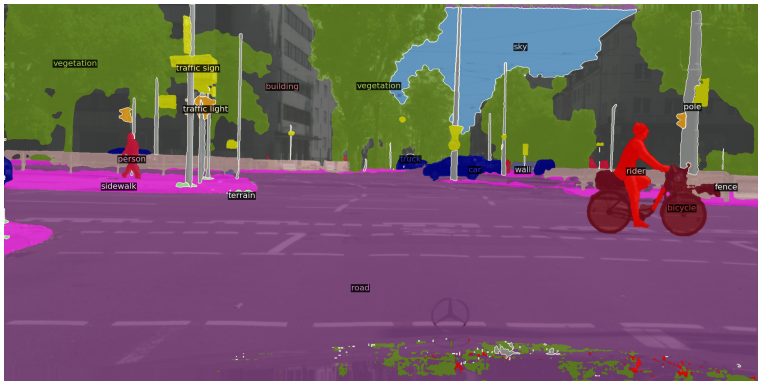
Figure 5.4.8: Weight transfer from Swin-T trained on Ade20k to SwinHR with frozen modules.

Where SwinHR surpasses Swin-T by 0.233mIoU.

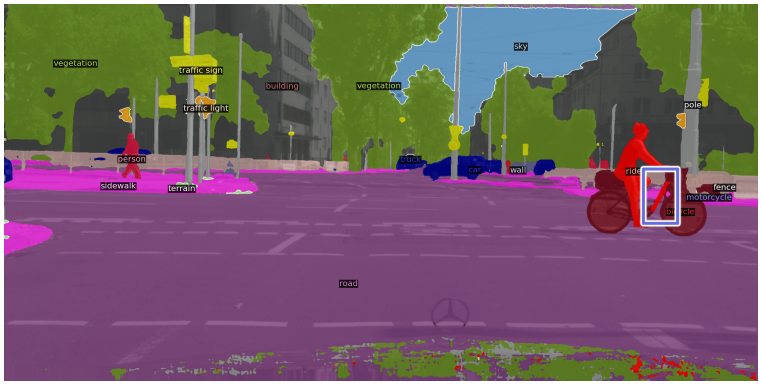
5.5 Qualitative Results

5.5.1 Cityscapes

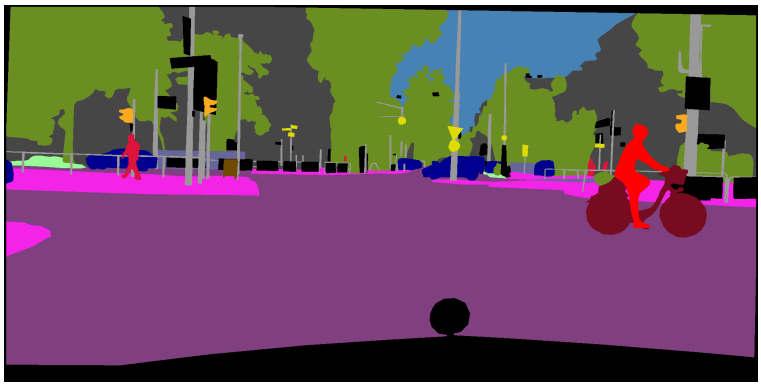
In the following results every region is color coded based on its category. Black pixels in ground truth image are unlabeled pixels and therefore ignored in evaluation. Observing 5.5.1 we see that in some areas SwinHR improves Swin-T results. These areas are marked with a blue rectangle on the Swin-T reference image. For example, SwinHR recognizes better a traffic swin at the back. Also, the part of the bicycle that is falsely assigned to the rider class, is smaller in SwinHR than in Swin-T. From 5.4 the fence class does indeed have a higher score in SwinHR than in Swin-T. Observing 5.5.2 we see only one difference between SwinHR ad Swin-T(excluding areas that are not used in evaluation). At the yellow rectangle we see that Swin-T falsely allocates a small region of the person class for the traffic sign class, something that SWinHR doesn't do.



(a) High resolution model: SwinHR backbone.



(b) Single resolution model: Swin-T backbone.

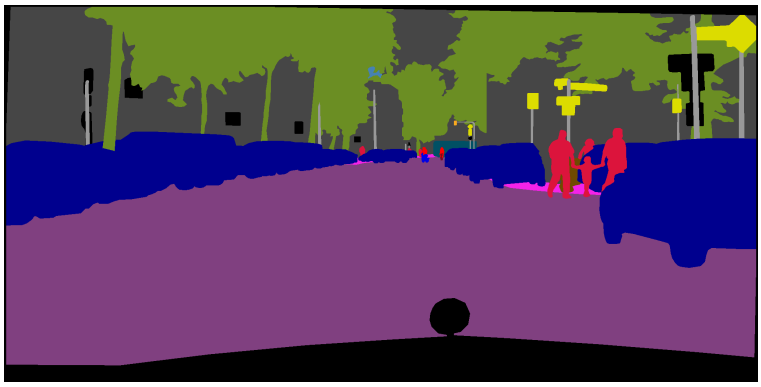


(c) Ground truth

Figure 5.5.1: Comparison of semantic segmentation results on val set image from Cityscapes

(a) High resolution model: SwinHR backbone.

(b) Single resolution model: Swin-T backbone.



(c) Ground truth

Figure 5.5.2: Comparison of semantic segmentation results on val set image from cityscapes

5.5.2 ADE20k

In the following results every region is color coded based on its category. Observing 5.5.3 we see that due to the complexity of ADE20k, it is not possible to compare the results since both performances are very small. The models seem to segment the regions satisfactorily but fail completely to classify them.

(a) High resolution model: SwinHR backbone.

(b) Single resolution model: Swin-T backbone.

Figure 5.5.3: Comparison of semantic segmentation results on val set image from ADE20K

5.6 Comparison with Mask2Former

5.6.1 Cityscapes

In this section we compare with our best model which is 5.4.2. We observe that compared to the baseline it improves by 0.1mIoU the performance on the task of semantic segmentation on Cityscapes val set.

		Cityscapes			
Model	*Backbobe*	$mIoU(class)$	$iIoU(class)$	$IoU(category)$	$iIoU(category)$
Mask2former	Swin-T	81.7069	64.9502	91.2996	80.9204
Mask2Former	SwinHR	**82.0443**	**65.7998**	**91.3846**	**81.6433**

Table 5.3: Performance of Mask2Former on Cityscapes with official and high resolution implementation

In the following tables, we compare per class and per category performance.

	IoU	
Class	$Swin-T$	$SwinHR$
road	98.3	98.4
sidewalk	86.6	87.1
building	93.7	93.8
wall	66.4	68.4
fence	68.8	69.3
pole	70.9	70.9
traffic light	76.1	76.2
traffic sign	82.4	83.5
vegetation	93.3	93.3
terrain	66.2	68.0
sky	95.4	95.3
person	85.6	85.8
rider	69.6	70.8
car	96.0	96.0
truck	87.3	85.9
bus	88.9	89.0
train	76.7	77.1
motorcycle	69.6	69.2
bicycle	80.6	80.8
Mean	81.7	**82.0**

Table 5.4: Per class performance of Mask2Former on Cityscapes with official and replicated implementation

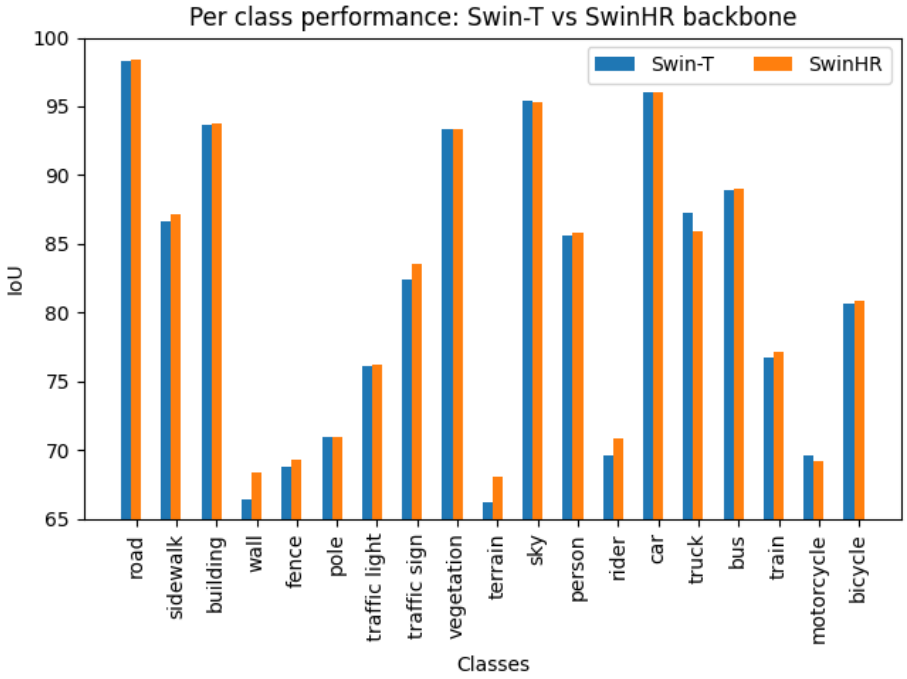


Figure 5.6.1: Per class performance comparison with Swin-T vs SwinHR backbone on Cityscapes.

	IoU	
Category	$Swin-T$	$SwinHR$
flat	98.9	98.9
construction	94.1	94.1
object	76.1	76.5
nature	93.6	93.6
sky	95.4	95.3
human	86.4	86.4
vehicle	94.8	94.8
Mean	91.3	**91.4**

Table 5.5: Per category performance of Mask2Former on Cityscapes with official and replicated implementation

We observe that out model performed slightly better in almost all of the categories and classes. From the graph we observe that it imrpoves mostly the sidewalk, the wall, the fence, the traffic sign, the terrain, the rider, and the train. It doesn't improve that much classes that already perform really well. Also, on the category performances table it is shown that our model only improves the object category. This could mean that the object category can be benefitted from multiple resolutions as it contains multi scale items.

5.6.2 ADE20k

In this section we compare with our best model which is 5.4.2. We observe that compared to the baseline it surpasses by 0.233mIoU on ADE20k.

		Cityscapes			
Model	*Backbobe*	*mIoU*	*fwIoU*	*mACC*	*pACC*
Mask2former	Swin-T	49.89	73.05	64.00	83.24
Mask2Former	SwinHR	50.13	73.14	63.65	83.36

Table 5.6: Performance of Mask2Former on ADE20k with official(replicated) and high resolution implementation

Additionally, we observe that it performs better according to all metrics except for pACC.

In the following graph per class mIoU is compared between the two models. These are a sample of the total of classes since there are in total 150 classes.

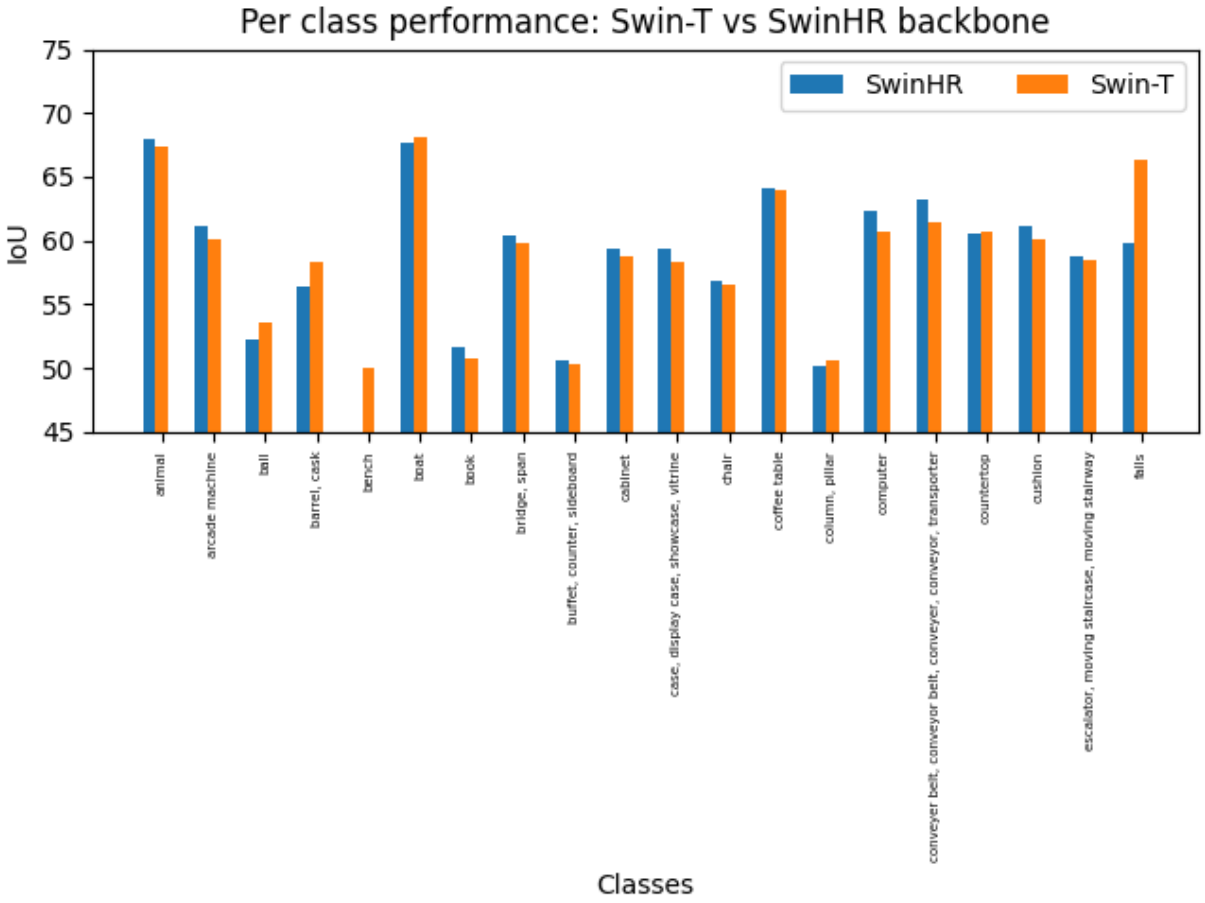


Figure 5.6.2: Per class performance comparison with Swin-T vs SwinHR backbone on ADE20k.

We see an improvement in most of the classes but this is still a sample of the total dataset.

5.7 Conclusions

This experimental application of multiple resolutions in the backbone of Mask2Former managed to match Mask2Former's performance proving that it can only provide more information rather than impede its training. Therefore, we conclude that multiple resolutions can be useful in a model's design and the choice of incorporating them depends on the performance against training time trade off. The experiments on different parts of the architecture couldn't be exhaustive in the context of this diploma research as the complexity of

this model introduces numerous parameters and the training time impedes chronically the collective tuning of them. However, from multiple experiments on different parameters such as learning rate, weight initialization, and pixel decoder model we managed to improve performance of the model on Cityscapes by 0.3mIoU and on ADE20k by 0.2mIoU.

Chapter 6

Conclusion

6.1 Conclusion

Semantic segmentation is an area in computer vision crucial in real-world applications. With the advancement of computer vision techniques we deviate from simple task problems such as classification or object detection. These tasks were satisfactorily approached with convolutional models. Moving on to transformers, object detection was further improved as it required multi-scale object detection. The self-attention module of transformers was able to implement that requirement and introduce contextual informations that convolutions were not able to. However, unlike classification or even object detection, semantic segmentation requires multiple scale recognition of object shape. Transformers were able to perform this task, however, the architecture philosophy needed to be changed in order to scale up performance in a more demanding task. Encoder-decoder architectures suit classification tasks as they transform information to a lower dimension and transform it to a class label. Later approaches attempted to introduce multiple resolutions by using residual connections from encoder to decoder in order to prevent this loss of information. This technique though still faces the problem of processing information without loss and that is where multiple resolutions introduce a solution to the problem. After the extended background research, multiple resolutions are dominating SOTA and improve their respective single resolution models. Theoretically, multiple resolutions can only improve a model as they introduce extra information than the information produced in the original model. However, the complexity of the model increases and that affects both training time and training curve.

In our implementation we observed that the Mask2Former backbone has an encoder-decoder style architecture. Following the notion of multiple resolutions we converted this backbone to preserve high resolution information. In this type of models, however, initialization is key. Due to the lack of a pre-existing SwinHR model, different weight transfer techniques had to be implemented in order to exploit single resolution available weights. Initially, we matched the weights with the diagonal modules of the model and left the rest of the modules uninitialized. Alternatively, we initialized the weights in such a way so that the model's starting performance is equal to the original model's best performance. This technique proved the most successfull. Different experiments were executed by changing learning rate values, and freezing layers. Lastly, we experimented with two dfferent decoders : OCR and MsDefform attention. The results of this experimentation proved that the multiple resolution model can match the performance of the original model and even improve it. Specifically, in cityscapes an improvement of 0.3mIoU was observed and in ADE20k an imrpovent of 0.2mIoU.

6.2 Future Work

In future work, there are a few possible alternatives that would maybe alter the outcome of this experimentation and prove more strongly our thesis. Some of those are:

1. Different initialization of weights that can be applied in all layers by properly transforming weights between the different resolutions
2. The parameters we experimented with in 5.4 can be more exhaustively searched maybe provided that we have in our disposal more processing resources and larger time frame.
3. Different pixel decoders can be tested. Also, the MsDefform attention decoder's hyperparameters such as number of transformer layers can be tuned to serve better the characteristics of the new backbone.
4. More frequent fuse layers can be added to the backbone so that information can be exchanged more frequently.
5. Experiment with other origin of transformer decoder inputs.
6. The model's functions could be further optimized to reduce training time and thus allow for faster experimentation.

Additionally, other ways of expanding our research are :

1. Testing this implementation on panoptic and instance segmentation tasks as Mask2former is a multi-purpose model. The performance of the Mask2former should be reflected equally successfully on all tasks in order to preserve its purpose.

2. Running multi-scale inference in order to compare with other architectures. Most architectures infer on augmented version of the input image in order to reach a multiple scale result.
3. Experiment with pixel decoder to test whether it could work in a more simplified version so as to reduce training time.
4. OneFormer is the new SOTA in all image segmentation categories and it is inspired by Mask2Former. It is also superior since it needs be trained only on panoptic segmentation and output results on all three categories. Similarly to Mask2former it contains the same pixel-level module with an encoder-decoder architecture which we can replace with a high resolution architecture like the one developed in this diploma thesis.

Appendix A

Bibliography

[1] Haralick, R. and Shapiro, L. *Computer and robot vision*. Computer and Robot Vision Bd. 2. Addison-Wesley Pub. Co., 1993.

[2] Weinland, D., Ronfard, R., and Boyer, E. "A survey of vision-based methods for action representation, segmentation and recognition". In: *Computer Vision and Image Understanding* 115.2 (2011).

[3] Sonka, M., Hlavac, V., and Boyle, R. *Image Processing: Analysis and Machine Vision*. 2nd ed. CL-Engineering.

[4] Ilea, D. E. and Whelan, P. F. "Image segmentation based on the integration of colour–texture descriptors—A review". In: *Pattern Recognition* 44.10 (2011), pp. 2479–2501.

[5] Geman, S. and Geman, D. "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-6.6 (1984), pp. 721–741.

[6] Dalal, N. and Triggs, B. "Histograms of oriented gradients for human detection". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. 2005, 886–893 vol. 1.

[7] Lowe, D. G. "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* (2004).

[8] Cordts, M. et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *CoRR* abs/1604.01685 (2016).

[9] Russakovsky, O. et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[10] Lin, T.-Y. et al. *Microsoft COCO: Common Objects in Context*. 2015. eprint:

[11] Adelson, E. H. "On seeing stuff: the perception of materials by humans and machines". In: *IS&T/SPIE Electronic Imaging*. 2001.

[12] Yuan, Y. *Automatic Brain Tumor Segmentation with Scale Attention Network*. 2020. DOI: 10.48550/arXiv.2011.03188.

[13] Wei, G.-Q., Arbter, K., and Hirzinger, G. "Automatic tracking of laparoscopic instruments by color coding". In: Springer Berlin Heidelberg, 1997, pp. 357–366. ISBN: 978-3-540-68499-2.

[14] Bartolomei, L., Teixeira, L., and Chli, M. "Perception-aware Path Planning for UAVs using Semantic Segmentation". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 5808–5815. DOI: 10.1109/IROS45743.2020.9341347.

[15] Everingham, M. et al. "The Pascal Visual Object Classes (VOC) Challenge." In: 88.2 (2010), pp. 303–338.

[16] Zhou, B. et al. "Scene Parsing through ADE20K Dataset". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[17] Kirillov, A. et al. *Panoptic Segmentation*. 2019. DOI: 10.48550/arXiv.1801.00868.

[18] Neuhold, G. et al. "The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 5000–5009. DOI: 10.1109/ICCV.2017.534.

[19] Cheng, B., Schwing, A. G., and Kirillov, A. "Per-Pixel Classification is Not All You Need for Semantic Segmentation". In: 2021.
[20] Wang, W. et al. "InternImage: Exploring Large-Scale Vision Foundation Models with Deformable Convolutions". In: *arXiv preprint arXiv:2211.05778* (2022).
[21] Liu, H. et al. *Polarized Self-Attention: Towards High-quality Pixel-wise Regression*. 2021. DOI: 10.48550/arXiv.2107.00782.
[22] Tao, A., Sapra, K., and Catanzaro, B. *Hierarchical Multi-Scale Attention for Semantic Segmentation*. 2020. DOI: 10.48550/arXiv.2005.10821.
[23] Chen, Z. et al. *Vision Transformer Adapter for Dense Predictions*. 2023. DOI: 10.48550/arXiv.2205.08534.
[24] Jain, J. et al. "OneFormer: One Transformer to Rule Universal Image Segmentation". In: 2023.
[25] Jain, J. et al. *SeMask: Semantically Masked Transformers for Semantic Segmentation*. 2022. DOI: 10.48550/arXiv.2112.12782.
[26] Khirodkar, R. et al. *Sequential Ensembling for Semantic Segmentation*. 2022. arXiv: 2210.05387 [cs.CV].
[27] Hassani, A. et al. *Neighborhood Attention Transformer*. 2023. DOI: 10.48550/arXiv.2204.07143.
[28] code, P. with. *Semantic Segmentation on Cityscapes val*. https://paperswithcode.com/sota/semantic-segmentation-on-cityscapes-val. Accessed: 2023-10-04. 2021.
[29] Wang, W. et al. "Image as a Foreign Language: BEIT Pretraining for Vision and Vision-Language Tasks". In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2023), pp. 19175–19186.
[30] Fang, Y. et al. *EVA: Exploring the Limits of Masked Visual Representation Learning at Scale*. 2022. DOI: 10.48550/arXiv.2211.07636.
[31] Wei, Y. et al. *Contrastive Learning Rivals Masked Image Modeling in Fine-tuning via Feature Distillation*. 2022. DOI: 10.48550/arXiv.2205.14141.
[32] Li, F. et al. *Mask DINO: Towards A Unified Transformer-based Framework for Object Detection and Segmentation*. 2022. DOI: 10.48550/arXiv.2206.02777.
[33] Hong, Y. et al. *Representation Separation for Semantic Segmentation with Vision Transformers*. 2022. DOI: 10.48550/arXiv.2212.13764.
[34] Paperswithcode. *Leaderboard Ade20k*. https://paperswithcode.com/sota/semantic-segmentation-on-ade20k-val. Accessed: 2023-10-04. 2022.
[35] Thoma, M. *A Survey of Semantic Segmentation*. 2016. DOI: 10.48550/arXiv.1602.06541.
[36] Everingham, M. et al. "The pascal visual object classes challenge: A retrospective". In: *International journal of computer vision* 111 (2015), pp. 98–136.
[37] Jadon, S. "A survey of loss functions for semantic segmentation". In: *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. IEEE, 2020.
[38] *CS231n: Convolutional Neural Networks for Visual Recognition*. https://cs231n.github.io/. Accessed: 2023-09-01.
[39] Srivastava, N. et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958.
[40] Goodfellow, I. J., Bengio, Y., and Courville, A. *Deep Learning*. http://www.deeplearningbook.org. Cambridge, MA, USA: MIT Press, 2016.
[41] Gundersen, G. *From Convolution to Neural Network*. https://gregorygundersen.com/blog/2017/02/24/cnns/. Accessed: 2023-10-17. 2017.
[42] Guo, Y. et al. "A review of semantic segmentation using deep neural networks". In: *International Journal of Multimedia Information Retrieval* 7 (2018), pp. 87–93.
[43] Weinland, D., Ronfard, R., and Boyer, E. "A survey of vision-based methods for action representation, segmentation and recognition". In: *Computer Vision and Image Understanding* 115.2 (2011), pp. 224–241. ISSN: 1077-3142. DOI: 10.1016/j.cviu.2010.10.002.
[44] Girshick, R. B. et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: (2013). DOI: 10.48550/arXiv.1311.2524.
[45] Carreira, J. et al. "Semantic Segmentation with Second-Order Pooling". In: 2012. ISBN: 978-3-642-33785-7. DOI: 10.1007/978-3-642-33786-4_32.

[46] Carreira, J. and Sminchisescu, C. "CPMC: Automatic Object Segmentation Using Constrained Parametric Min-Cuts". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.7 (2012), pp. 1312–1328. DOI: 10.1109/TPAMI.2011.231.
[47] Girshick, R. "Fast R-CNN". In: (2015). DOI: 10.48550/arXiv.1504.08083.
[48] Ren, S. et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: (2016). DOI: 10.48550/arXiv.1506.01497.
[49] Caesar, H., Uijlings, J., and Ferrari, V. "Region-based semantic segmentation with end-to-end training". In: (2016). DOI: 10.48550/arXiv.1607.07671.
[50] Long, J., Shelhamer, E., and Darrell, T. "Fully convolutional networks for semantic segmentation". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, 2015, pp. 3431–3440. DOI: 10.1109/CVPR.2015.7298965.
[51] Planche, B. and Andres, E. *Hands-On Computer Vision with TensorFlow 2*. Packt Publishing Ltd, 2019. ISBN: 978-1788839266.
[52] Ronneberger, O., Fischer, P., and Brox, T. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: (2015), pp. 234–241.
[53] Chen, L.-C. et al. "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs". In: (2017). DOI: 10.48550/arXiv.1606.00915.
[54] Chen, L. et al. "Rethinking Atrous Convolution for Semantic Image Segmentation". In: (2017). DOI: 10.48550/arXiv.1706.05587.
[55] Chen, L.-C. et al. "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation". In: *Computer Vision – ECCV 2018*. Cham: Springer International Publishing, 2018, pp. 833–851. ISBN: 978-3-030-01234-2.
[56] Wang, J. et al. "Deep High-Resolution Representation Learning for Visual Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.10 (2021), pp. 3349–3364. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2020.2983686.
[57] Liu, Z. et al. "Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 10012–10022.
[58] Carion, N. et al. *End-to-End Object Detection with Transformers*. 2020. DOI: 10.48550/arXiv.2005.12872.
[59] Cheng, B. et al. "Masked-attention Mask Transformer for Universal Image Segmentation". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 1280–1289. DOI: 10.1109/CVPR52688.2022.00135.
[60] Yuan, Y. et al. *HRFormer: High-Resolution Transformer for Dense Prediction*. 2021.
[61] Wei, C. et al. "High -Resolution Swin Transformer for Automatic Medical Image Segmentation". In: (2022). DOI: 10.48550/arXiv.2207.11553.
[62] Geiger, A., Lenz, P., and Urtasun, R. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.
[63] Garcia-Garcia, A. et al. *A Review on Deep Learning Techniques Applied to Semantic Segmentation*. 2017. DOI: 10.48550/arXiv.1704.06857.
[64] Scharwächter, T. et al. "Efficient Multi-cue Scene Segmentation". In: *German Conference on Pattern Recognition*. 2013.
[65] Neuhold, G. et al. "The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 5000–5009. DOI: 10.1109/ICCV.2017.534.
[66] FacebookResearch. *Leaderboard Ade20k*. `https://github.com/facebookresearch/Mask2Former`. Accessed: 2023-10-04. 2021.
[67] Wu, Y. et al. *Detectron2*. `https://github.com/facebookresearch/detectron`. Accessed: 2023-10-04. 2021.
[68] Loshchilov, I. and Hutter, F. *Decoupled Weight Decay Regularization*. 2019. DOI: 10.48550/arXiv.1711.05101.
[69] Chen, L.-C. et al. *DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs*. 2017. DOI: 10.48550/arXiv.1606.00915.
[70] Dosovitskiy, A. et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. DOI: 0.48550/arXiv.2010.11929.

[71] Zheng, S. et al. *Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers*. 2021. DOI: 10.48550/arXiv.2012.15840.
[72] Zhu, H., Chen, B., and Yang, C. *Understanding Why ViT Trains Badly on Small Datasets: An Intuitive Perspective*. 2023. DOI: 10.48550/arXiv.2302.03751.
[73] Yeh, C. et al. *AttentionViz: A Global View of Transformer Attention*. 2023. DOI: 10.48550/arXiv.2305.03210.
[74] Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer, 2006.
[75] Vaswani, A. et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017.
[76] K, S. *Region of Interest Pooling*. https://towardsdatascience.com/region-of-interest-pooling-f7c637f409af. Accessed: 2023-10-04. 2019.
[77] Krähenbühl, P. and Koltun, V. *Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials*. 2012. DOI: 10.48550/arXiv.1210.5644.
[78] Brown, T. B. et al. *Language Models are Few-Shot Learners*. 2020. DOI: 10.48550/arXiv.2005.14165.
[79] Devlin, J. et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. DOI: 10.48550/arXiv.1810.04805.
[80] Lewis, M. et al. *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*. 2019. DOI: 10.48550/arXiv.1910.13461.
[81] He, K. et al. *Deep Residual Learning for Image Recognition*. 2015. DOI: 10.48550/arXiv.1512.03385.
[82] Lin, T.-Y. et al. *Feature Pyramid Networks for Object Detection*. 2017. DOI: 10.48550/arXiv.1612.03144.
[83] Yuan, Y. et al. *Segmentation Transformer: Object-Contextual Representations for Semantic Segmentation*. 2021. DOI: 10.48550/arXiv.1909.11065.
[84] Kirillov, A. et al. *Panoptic Feature Pyramid Networks*. 2019. DOI: 10.48550/arXiv.1901.02446.
[85] Zhang, W. et al. *K-Net: Towards Unified Image Segmentation*. 2021. DOI: 10.48550/arXiv.2106.14855.
[86] Zhu, X. et al. "Deformable DETR: Deformable Transformers for End-to-End Object Detection". In: *arXiv preprint arXiv:2010.04159* (2020).