University of Padova

Department of Mathematics "Tullio Levi-Civita"

Master Thesis in Data Science

Domain Generalization for Semantic Segmentation exploiting Vision-Language features

Supervisor

Prof. Pietro Zanuttigh

University of Padova

Co-supervisor

Dr. Francesco Barbato

University of Padova

Master Candidate

Luca Careddu

Academic Year

2024-2025

I dedicate this work to the community of the field as a contribution to the current research.

Abstract

Domain Generalization in Semantic Segmentation (DGSS) is an attractive open research field in Computer Vision. It tackles the semantic segmentation performance drop that arises when predicting images of target datasets whose distribution is highly different from those of the source dataset. Unlike Unsupervised Domain Adaptation (UDA), where the target images, even though without their labels, can be exploited during training to facilitate the domain shift, Domain Generalization solely relies on the source dataset at training time. In some common settings, since manually labeling images for semantic segmentation is very time-consuming, the source dataset is typically made of synthetic images coming from video games (*e.g.,* GTA5) or game engines (*e.g.,* SELMA), and only during inference target datasets with real images (*e.g.,* Cityscapes) are employed. Lately, Vision Language Models (VLMs) such as CLIP have shown their remarkable generalization capabilities across many image classification datasets. Indeed, the rich semantics learned from textual supervision allow them to handle much better the domain shift at test time. Even though some tried to use those models to improve on previous DGSS specialized models, only a few exploited the text representations to drive the task. In this work, we assess the direct contribution of language in solving the DGSS task on all the generalization scenarios (*i.e.,* synthetic-to-real, real-to-real, and synthetic-to-synthetic) by building a model that employs VLMs as encoders to operate on image-text data and two decoders, one for each modality, that fuse the heterogeneous representations and solve the segmentation task.

Contents

Listing of figures

Listing of tables

Listing of acronyms

AI ..... Artificial Intelligence

DL ..... Deep Learning

NLP ..... Natural Language Processing

CV ..... Computer Vision

SOTA ..... State-Of-The-Art

SS ..... Semantic Segmentation

DA ..... Domain Adaptation

UDA ..... Unsupervised Domain Adaptation

DG ..... Domain Generalization

DGSS ..... Domain Generalization Semantic Segmentation

SSL ..... Self-Supervised Learning

MIM ..... Masked Image Modeling

LN ..... Layer Normalization

MHSA ..... Multi-Head Self-Attention

FNN ..... Feed Forward Network

VLM ..... Vision-Language Model

VFM ..... Vision Foundation Model

CL ..... Contrastive Learning

CNN ..... Convolutional Neural Network

mIoU ..... mean Intersection over Union

RoI ..... Region of Interest

FPN . . . . . Feature Pyramid Network

MSE . . . . . Mean Square Error

TE . . . . . Text Encoder

TD . . . . . Text Decoder

PTM . . . . . Pixel-Text Matching

M2F Mask2Former

M2F-S Mask2Former-Small

VReg . . . . . Vision Regularization

1 Introduction

Semantic Segmentation (SS) is a Computer Vision (CV) dense prediction task requiring each pixel in the image to be assigned a label from a set of pre-defined labels. The applications of this task are several, from medicine diagnostics to self-driving cars exploiting the multitude of images from environments such as indoor and satellite. One of the biggest problems in this research area is tackling the domain shift that arises when the model, trained on a source dataset, has to predict over target images having a different distribution because, captured in a different context, position, or under different conditions specific of the application in question. In the research subfield regarding autonomous driving, domain shifts are the main challenge when trying to adapt the models, trained on a specific use case, to different cities, weather conditions, or in completely different domains, for example, when moving from synthetic images, namely those created from video-games or game-engines, to real images.

Unsupervised Domain Adaptation (UDA) tries to bridge the gap between the different train and test domains by exploiting, during training, target domain images without involving their ground-truth labels. In general, UDA techniques take place at three levels of the model: input level, feature level, and output level. The input level shift is typically mitigated by aligning source images to the target image style copying a range of the spectrum of frequencies from the latter or by training Generative Adversarial Networks (GANs) to produce source images sharing similar visual appearances to target ones. Domain adaptation at the feature level instead aims to align the different source and target pre-logit feature space (the output of the last layer before the classifier) distributions. Common methods try to minimize distances in the

feature space by minimizing some divergence measures, while others directly exploit class centroids (prototypes), usually computed averaging the per-class feature spaces, and optimize w.r.t. those. Finally, at the output level, self-training is typically employed and consists of retraining the network on labels that are generated by itself (pseudo-labels) while filtering predictions having low confidence (based on the pixel-wise logits or softmax probabilities). Other works, at the output level, use adversarial learning to build a segmentation network that can fool the discriminator with similar outputs for both source and target images. UDA is an effective approach to tackle domain shifts and has recently shown its great potentiality when moving from the synthetic to the real domain ([1]), which is of importance since synthetic images can be labeled effortlessly compared to real images which can take hours just for a single one. Nevertheless, UDA techniques result in being too highly engineered and of poor flexibility when trying to adapt the model to multiple domains simultaneously. Additionally, one may need to stack them to achieve a significant improvement, ending up in a complex and over-fitted methodology.

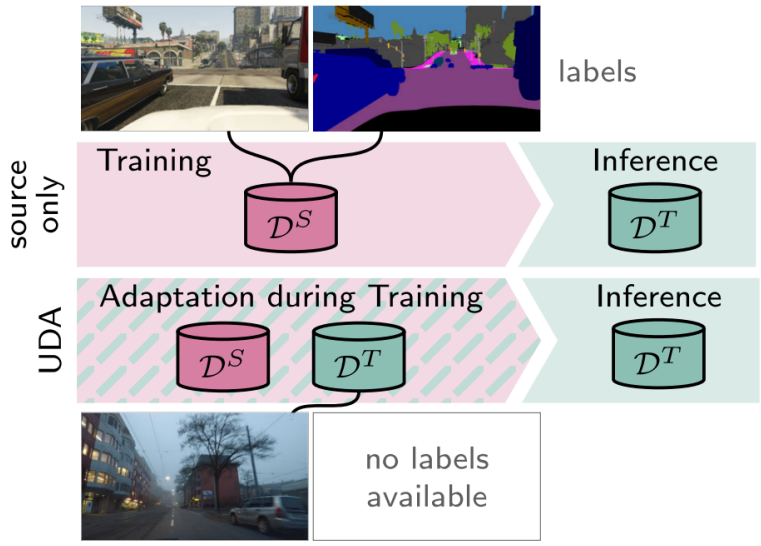

Figure 1.1: Difference between Unsupervised Domain Adaptation (UDA) and Domain Generalization (DG), corresponding to "source only". $\mathcal{D}^S$ and $\mathcal{D}^T$ represent the source and target datasets, respectively. Figure excerpt from "Survey on Unsupervised Domain Adaptation for Semantic Segmentation for Visual Perception in Automated Driving" [2].

Domain Generalization (DG) discards any attempt to adapt the model to a specific target dataset at any of the three levels and exploits models that, on their own, have highly general-

ization capabilities thanks to the way they have been pretrained. DG models only rely on the source dataset at training time and are then used to perform zero-shot evaluation on the target datasets (the difference between UDA and DG is illustrated in Figure 1.1). Modern DG models are typically large and have their backbones pretrained with multi-modal data such as text and image as in the case of Visual Language Models (VLMs) or uni-modal, and those are the Visual Foundation Models (VFMs) since they are a solid base from which to solve many vision tasks. VLMs are trained using Contrastive Learning (CL) strategies and learn to associate text and image pairs by learning a shared multi-modal embedding space, while VFMs typically rely on Self-Supervised Learning (SSL) and specifically Masked Image Modeling (MIM) to learn image representations by predicting missing (masked out) parts of the images. In practice, many of these models are based on the Transformer architecture, which has shown superior generalization capabilities compared to vanilla Convolutional Neural Networks (CNNs). Despite in DG a lot of generalization capability comes from the pretrained backbone encoder, recent works ([3, 4, 5]), discussed in Chapter 3, have shown that it is possible to significantly enhance the performance on the task by employing decoder architectures that work with both visual and language information and to surpass backbones fine-tuned solely on images with shallow decoders. Recently, the VLM CLIP [6] (described in Section 5.1.1) and its subsequent improved variants together with the transformer-based vision decoder Mask2Former [7] (described in Section 5.1.2), known for its generalization potential, are used in conjunction in DGSS to solve synthetic-to-real and real-to-real domain shift problems with excellent results.

2 Background

In 2017, the paper "Attention Is All You Need" [8] revolutionized the Deep Learning field by promoting a new way, solely based on attention mechanisms (the Transformer, Figure 2.1), to approach Natural Language Processing problems. The work is now a pillar in the AI community because, since then, the Transformer has been successfully adapted and applied to a variety of different data and tasks. Indeed, in 2021, "An Image Is Worth 16X16 Words" [9] introduces the Vision Transformer (ViT), a novel architecture based on the Transformer encoder that achieves state-of-the-art performance in many image classification datasets.
This chapter provides the information needed to understand the subsequent reading. In particular, Section 2.1 explains different types of attention mechanisms while Section 2.2 briefly describes the architecture of ViT [9].

2.1 Attention

This section presents three types of attention, namely, Self-Attention in Section 2.1.1, Cross-Attention in Section 2.1.2, and Deformable Self-Attention as in [10] in Section 2.1.3.

2.1.1 Self-Attention

Self-Attention, as in the Transformer encoder [8], is a mathematical operation that, given a sequence, compares elements pairwise and produces a response sequence related to the original

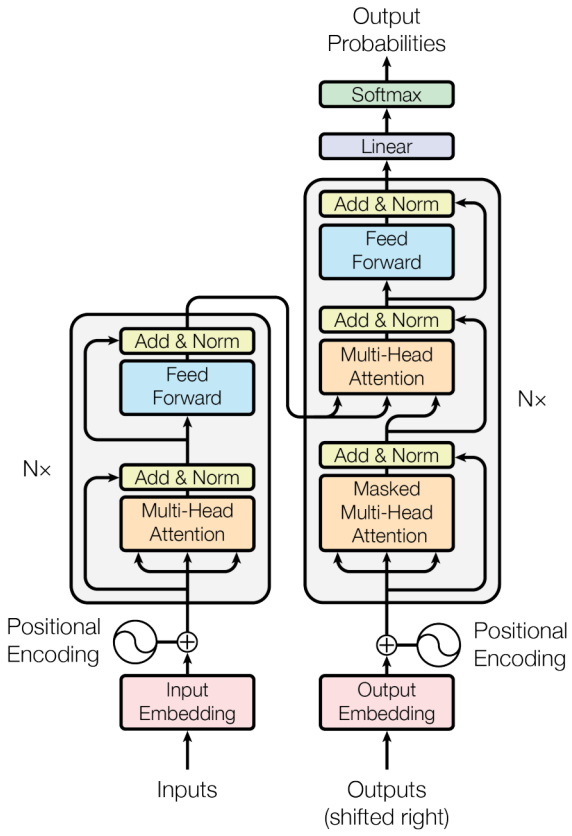

Figure 2.1: The architecture of the Transformer comprising Self-Attention in the encoder part (left) and, Masked Self-Attention and Cross-Attention in the decoder part (right). Figure excerpt from "Attention Is All You Need" [8].

one according to the results obtained from the comparison. In practice, a Query Q, Key K, and Value V matrices are obtained as linear transformations of the input sequence X, and the following operation, called "Scaled Dot-Product Attention" is performed:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^\intercal}{\sqrt{d_k}})V \tag{2.1}$$

where d_k is K dimensionality. The formula in Equation 2.1 queries a sequence Q into another one K to retrieve the values in V whose information is closest to what is requested in Q. The pairwise comparison is made via the softmax function that measures the strength of the connections between queries and keys and guarantees the correct assignment of the values. The product is scaled by the K dimensionality to avoid small-gradient problems as d_k increases, as explained in [8]. Note that Q, K, and V are input-dependent, differently from kernels in standard convolutions. Moreover, Attention can be seen as a generalization of convolutions that

attends to all sequence elements instead of just a constrained receptive field. Meanwhile, similarly to convolutions, the Scaled Dot-Product Attention operations can be split into multiple groups, called heads, to obtain "Multi-Head Self-Attention":

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O \tag{2.2}$$

where

$$\text{head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.3}$$

and $W_i^Q \in \mathbb{R}^{d \times d_k}$, $W_i^K \in \mathbb{R}^{d \times d_k}$, $W_i^V \in \mathbb{R}^{d \times d_v}$, and $W^O \in \mathbb{R}^{hd_v \times d}$ are linear projection matrices. Multi-Head Self-Attention (Equation 2.2) allows more parallelization and, unlike single-head attention, jointly attends to information from different subspaces to increase capability.
Self-Attention can be used to predict in a causal manner so as to avoid future elements in the sequence influencing earlier ones by masking out the formers. In the Transformer decoder [8], this is accomplished by setting future tokens (elements) to $-\infty$ so that Equation 2.1 becomes:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^\intercal + U}{\sqrt{d_k}})V \tag{2.4}$$

where U is an upper-triangular matrix of $-\infty$ values. Equation 2.4 with U as said is generally called "Masked Self-Attention" even though the term is rather general and, in practice, U can be in any form, even learnable.

The major drawback of Self-Attention is space complexity, which grows quadratically with the input sequence length, but several variants that mitigate this problem have been proposed.

2.1.2 Cross-Attention

Cross-Attention [8], relies on the Self-Attention mechanism with the only difference that the Query Q, Key K, and Value V matrices from Equation 2.1 can have different lengths - as they could correspond to different sequences - even though their dimensionalities still have to match. In general, Cross-Attention mixes two different sequences in input; for example, in the Transformer decoder [8], Q and K come from the Transformer encoder while V from the previous layer (see Figure 2.1).

2.1.3 Deformable Self-Attention

Apart from the quadratic complexity problem, in Self-Attention, with a proper parameter initialization, Query Q and Key K representations follow the standard distribution causing the attention weights (resulting from the softmax function) to be approximately $\frac{1}{N_k}$ as N_k (K dimensionality) increases hence producing ambiguous gradients and slowing down convergence. In the context of images, K represents image patches, and N_k can be very large. Motivated by these issues and by the Object Detection need to work on feature maps pyramids to recognize small objects, which implies having very large N_k values, Deformable DETR [10] introduces Deformable Self-Attention, based on keys and values point sampling to speed up convergence and highly reduce overall complexity. Formally, given an input feature map $x \in \mathbb{R}^{C \times H \times W}$, let z_q be a query representation and p_q its 2-d reference point, (Multi-Head) Deformable Attention is calculated as follows:

$$\text{DeformAttn}(z_q, p_q, x) = \sum_{m=1}^{M} W_m [\sum_{k=1}^{K} A_{mqk} \cdot W'_m x(p_q + \Delta p_{mqk})] \tag{2.5}$$

where M is the heads number, K such that $K \ll HW$ is the number of sampled points, W_m and W'_m are linear projection matrices and, Δp_{mqk} and A_{mqk} such that $\sum_{k=1}^{K} A_{mqk} = 1$ are the sampling offset and the attention weight of the k^{th} sampling point and m^{th} attention head, respectively. $\Delta p_{mqk} \in \mathbb{R}^2$ so bilinear interpolation is applied in computing $x(p_q + \Delta p_{mqk})$. In practice, Δp_{mqk} and A_{mqk} are obtained via a linear projection over the query feature z_q. Note that K does not depend on HW. Note also that when the sampling points traverse all locations, Equation 2.5 denotes the standard Self-Attention. Modern dense prediction task solvers benefit from feature pyramids to detect/segment small objects in the scenes, and, to this end, Deformable Attention can be naturally adapted to multi-scale features to obtain Multi-scale Deformable Attention. Equation 2.5 then becomes:

$$\text{MSDeformAttn}(z_q, \hat{p}_q, \{x^l\}_{l=1}^{L}) = \sum_{m=1}^{M} W_m [\sum_{l=1}^{L} \sum_{k=1}^{K} A_{mlqk} \cdot W'_m x^l(\phi_l(\hat{p}_q) + \Delta p_{mlqk})] \tag{2.6}$$

where M is the heads number, L is the feature levels number, K such that $LK \ll HW$ is the number of sampled points, W_m and W'_m are linear projection matrices, $\hat{p}_q \in [0, 1]^2$ is the normalized 2-d reference point and, Δp_{mlqk} and A_{mlqk} such that $\sum_{l=1}^{L} \sum_{k=1}^{K} A_{mlqk} = 1$

are the sampling offset and the attention weight of the k^{th} sampling point in the l^{th} feature level within the m^{th} attention head, respectively. $\phi_l(\hat{p}_q)$ function rescales $\hat{p}_q$ to the l^{th} feature pyramid level.

Compared to the standard Self-Attention (Equation 2.1), Deformable Self-Attention requires only linear time w.r.t. the input sequence length as, in practice, is common to have $K = 4$ and, in multi-scale scenarios, $L = 3$.

2.2 ViT

The Vision Transformer (ViT) [9] is the first effective work that leverages Transformers [8] for image classification with excellent results in comparison with the convolution-based models. The former issue with Transformers is that they work with 1-D sequences, so ViT, following the naïve approach, flattens images and adds a positional embedding to the sequence that encodes the pixels' spatial locations. Since using all pixel representations would imply having a very long sequence and given that Transformers suffer from the quadratic complexity problem, ViT proposes to reduce the computational burden by splitting images into patches of fixed dimension (*e.g.,* 16×16), linearly transforming, and flattening them to a 1-D sequence. In this way, after adding/concatenating 1-D learnable positional embeddings, the images can be treated as sequences to solve, with feasible computational complexity, Computer Vision tasks. The architecture of ViT is then simple: the first layer corresponds to the patchify stem, namely, a convolutional layer with kernel size and stride corresponding to the patch dimension, while the remaining layers are Transformer encoder layers [8], so each comprising Multi-Head Self-Attention followed by a Feed Forward Network as in Figure 2.1. In practice, the very last layer is fully connected - as typical in image classification - to predict the classes. ViT is trained in a supervised fashion using a `[CLASS]` token to predict the classes. ViT models family comprises ViT-B, ViT-L, and ViT-H corresponding to, respectively, "Base", "Large", and "Huge" model variants.

3
Related work

In this chapter are presented and explained the recent works that, from the perspective of the methodology and models used, are most similar to and on which ours is based. DenseCLIP [3], presented in Section 3.1, solves the general Semantic Segmentation task but is the foundation work that inspired the subsequent ones in the Domain Generalization field. Section 3.2 presents VLTSeg [4] while Section 3.3 tqdm [5], both state-of-the-art models in the Domain Generalization Semantic Segmentation field.

3.1 DenseCLIP

DenseCLIP [3] represents the foundation work in the line of research that exploits Visual Language Models (VLMs) to solve dense prediction tasks such as Semantic Segmentation. The authors of the work recognize the potential of VLMs and, in particular, of CLIP [6] that, trained on text-image pairs via contrastive learning, shows impressive generalization ability on dozens of classification datasets. The aim of their work was then to adapt CLIP to dense prediction tasks, where the model has to provide a label for each pixel in the image since, on its own, CLIP is not able to output dense predictions as trained at image-level instead of pixel-level. The authors, therefore, crafted a novel technique inspired by the original work that allows CLIP features to be used for dense prediction tasks. In practice, they convert the original image-text pair matching problem into a pixel-text pair one by adding an auxiliary segmentation loss, called Pixel-Text Matching (PTM) Loss, based on the score maps obtained from the pairwise cosine

similarity between the pixel and contextualized text embeddings. This is possible once noticed that the score maps themselves can be viewed as low-resolution segmentation maps. Before that, contextualized text embeddings are obtained by passing the visual and language representations to a Transformer Decoder-based module that computes the cross-attention between the two. In order to preserve the language priors, during training, they keep the CLIP text encoder frozen and learn an offset for the frozen text embedding residuals. Their simplest model has input text prompts fixed that summarize the associated classes, *i.e.,* in the form `"a photo of a {class_name}."`. Moreover, they exploit the context in the images to learn special embeddings that, prompted together with the original texts to the CLIP text encoder, further contextualize the language representations before computing the cross-attention and improve the overall performance.

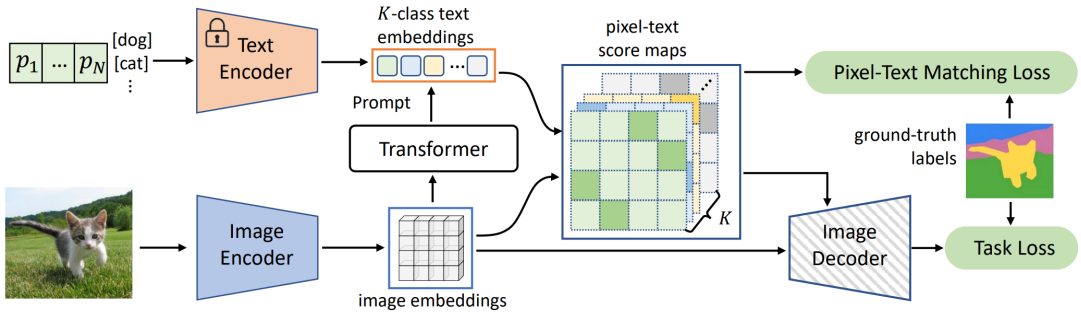

Figure 3.1: The DenseCLIP method involving language contextualization via an auxiliary Pixel-Text Matching (PTM) segmentation loss. Figure excerpt from "DenseCLIP: Language-Guided Dense Prediction with Context-Aware Prompting" [3].

The DenseCLIP method is illustrated in Figure 3.1. In their work, CLIP-Base with stride 16 (CLIP-B/16) and the lightweight Semantic Feature Pyramid Network (FPN) [11] are employed, respectively, as vision encoder and vision decoder, and the performance is assessed on the ADE20K dataset [12] - a scene-centric semantic segmentation dataset with 20k images and 150 classes - using as metric the mean Intersection over Union (mIoU). The results in the paper show that VLMs following the DenseCLIP method are capable of surpassing models, both convolution- and transformer-based, following the classical pre-training + fine-tuning paradigm and using only visual data. The DenseCLIP method, which then consists of vision-language pre-training + language-guided fine-tuning with context-aware prompting, is model-agnostic, in fact, the authors in the paper show that the same, employing a different vision encoder, such as ResNet101 [13] and Swin [14], is still superior to the classic supervised and even self-supervised training + fine-tuning approach. In the paper, results are reported also

for the method applied in the object detection and instance segmentation scenarios with similar improvements. This work pointed out important observations that paved the way for the subsequent works, among which the superiority of the VLM CLIP features to those of large self-supervised models trained just on images and the hidden potential of the CLIP language encoder which can truly play a key role in the fine-tuning stage.

3.2 VLTSeg

Visual-Language representation Transfer for domain generalized Segmentation (VLTSeg) [4] introduces a new approach to DGSS starting from the results obtained in DenseCLIP [3], and reaches the state-of-the-art in multiple tasks. The innovations the authors provide come totally from the architectural side that is upgraded to meet the latest standards. In particular, VLTSeg replaces the semantic FPN decoder [11] used in DenseCLIP with Mask2Former [7], a robust and versatile semantic model, known for its generalization abilities as designed for solving semantic, instance, and panoptic segmentation using the same architecture. Moreover, it initializes the backbone with EVA-CLIP [15] weights - a scaled version of CLIP [6] trained via a MIM (Masked Image Modeling) and CL (Contrastive Learning). As in the DenseCLIP method, VLTSeg uses the Pixel-Text Matching (PTM) auxiliary loss and keeps the backbone text encoder frozen.

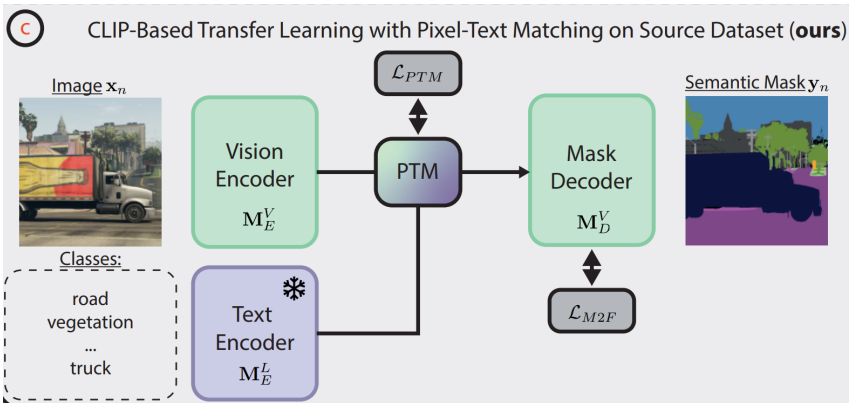

Figure 3.2: The VLTSeg architecture comprising a VLM backbone (CLIP, EVA-CLIP), the DenseCLIP [3] Pixel-Text Matching (PTM) module and the Mask2Former [7] vision decoder. Figure excerpt from "VLTSeg: Simple Transfer of CLIP-Based Vision-Language Representations for Domain Generalized Semantic Segmentation" [4].

The overall architecture can be visualized in Figure 3.2. The authors of the work also introduce a new robustness metric inspired by the relative Performance under Corruption (rPC)

called relative Performance under Domain shift (rPD) and defined as the ratio between the mean mIoU of the target datasets and the source dataset mIoU, and motivate it by pointing out that corruptions are technically one form of domain shift. In their experiments they use as synthetic datasets GTA5 [16] and SYNTHIA [17] while Cityscapes [18], BDD100k [19], and Mapillary Vistas [20] as real datasets, and solve both synthetic-to-real and real-to-real tasks evaluating, when possible, with their rPD metric. The results show that their method, which involves performing transfer learning from VLMs, drastically outperforms, on average over three challenging real-world datasets, the classic pre-train and fine-tune paradigm with SegFormer [21] as the backbone. They also surpass the previous SOTA on the GTA5 $\rightarrow$ Cityscapes benchmark and the previous unsupervised SOTA on Cityscapes $\rightarrow$ ACDC [22]. Finally, at the time of writing, the model still represents the state-of-the-art in the supervised setting on Cityscapes.

3.3 TQDM

Textual Query-Driven Mask transformer (tqdm) [5] represents the state-of-the-art in DGSS to date. Based on DenseCLIP [3] and resembling VLTSeg [4] in the model architecture choice, tqdm for the first time leverages textual object queries in the vision decoder and adds three regularization losses that preserve the vision-language alignment of the pre-trained VLMs during training to achieve state-of-the-art mIoU on GTA5 $\rightarrow$ Cityscapes [16, 18] improving on the prior vision SOTA Rein [23] and vision-language SOTA VLTSeg [4]. The textual object queries used in the method correspond to the contextualized text embeddings discussed in Sections 3.1 and 3.2 that are transformed and, this time, passed as queries to the vision decoder (Mask2Former [7]) to be used within the cross-attention layers. The original Mask2Former uses 100 randomly initialized object queries so that tqdm manages to perform better using only 19% of the original number of queries, corresponding to the 19 Cityscapes class text embeddings. Moreover, the method also inserts additional cross-attention layers within the vision decoder to obtain language-informed pixel embeddings using, under a different transformation, the contextualized text embeddings as key and value within the cross-attention mechanism. As in DenseCLIP [3] and VLTSeg [4], the method uses context-aware prompting to learn dynamic prompts based on the context of the images that are concatenated with the static text prompts summarizing the class labels, and employs three regularization losses: one for the image encoder alignment, another for the text encoder alignment, and a last one for image-text encoders alignment. The first two regularization losses keep the encoders aligned

with their original pretrained instance, while the third one keeps the two encoders aligned with each other w.r.t. their current instances and the original pretraining task. As in prior works, the CLIP [6] text encoder is frozen so that the language regularization is performed at prompt level when context-aware prompting is employed. Finally, the third regularization corresponds to the Pixel-Text Matching (PTM) auxiliary loss introduced in DenseCLIP to keep pixel and text embeddings aligned. As in VLTSeg, the highest mIoU are reached when the encoder is initialized with EVA-CLIP [15] weights resulting, on average, in a 10% gain w.r.t. the encoder initialized with CLIP weights. This pioneering work leaves room for new ideas on how to leverage language information within a vision decoder to generalize across different domains and possibly across different CV tasks. The overall method is depicted in Figure 3.3.

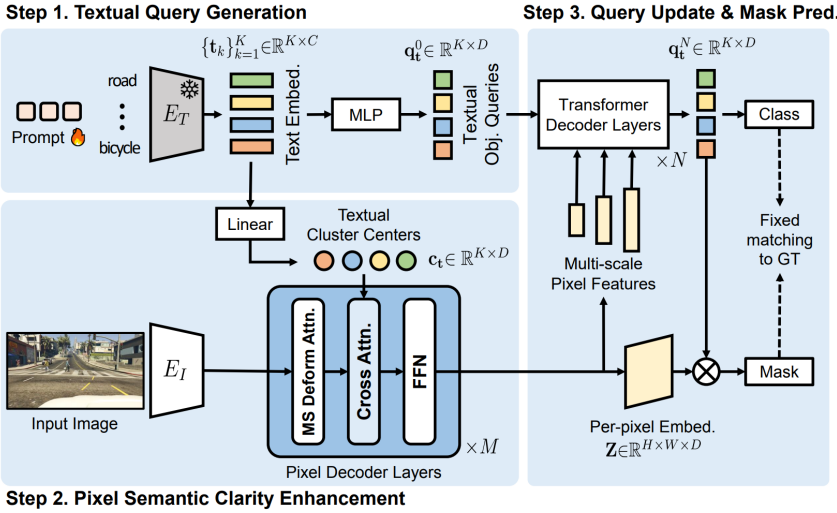

Figure 3.3: The tqdm method passing contextualized text embeddings [3] to the Mask2Former [7] vision decoder in the form of queries (transformer decoder layers) and keys (pixel decoder layers) within the transformer cross-attention mechanism. Figure excerpt from "Textual Query-Driven Mask Transformer for Domain Generalized Segmentation" [5].

4 Datasets

This chapter reports the main information needed to understand the data used in the experiments. In particular, Section 4.1 presents Cityscapes [18], a real-world dataset used in literature to evaluate models trained on both synthetic and real images while Section 4.2 presents GTA5 [16], a synthetic dataset commonly used as source dataset in both Domain Adaptation and Domain Generalization. Section 4.3 and Section 4.4 respectively present ACDC [22] and Mapillary Vistas [20], two real-world datasets used in practice as target datasets in DG. Finally, Section 4.5 presents SELMA [24], a synthetic dataset that can be used for both synthetic-to-real and synthetic-to-synthetic tasks.

4.1 Cityscapes

Cityscapes [18] is a real image dataset comprising 5000 finely annotated and 20000 coarsely annotated images with 2048×1024 resolution originally captured as stereo video sequences recorded in urban streets from 50 different cities. The annotations provided for the 30 classes are at both pixel and instance (for vehicles and people) levels, enabling the use of the data for semantic, instance, and panoptic segmentation. The 30 classes represent objects commonly present in urban street scenes and can be grouped into 8 categories: *flat surfaces*, *humans*, *vehicles*, *constructions*, *objects*, *nature*, *sky*, and *void*. In practice, out of 30 classes, only 19 are typically used as they are the only ones fully annotated across the dataset, and these are: *road*, *sidewalk*, *building*, *wall*, *fence*, *pole*, *traffic light*, *traffic sign*, *vegetation*, *terrain*, *sky*, *person*, *rider*,

car, *truck*, *bus*, *train*, *motorcycle*, and *bicycle*. On the other side, the 20000 additional coarsely annotated images can be used as a source of weakly-labeled data in weakly-supervised learning scenarios as greater in number compared to the finely annotated ones. The biggest problem researchers may encounter when training models using this dataset usually concerns the class distribution (see Figure 4.1); for example, the classes *road*, *building*, and *sky* are present in the majority of the images and cover big areas of the background view while rare classes such as *bicycle* and *train*, and classes representing smaller objects such as *traffic sign* and *traffic light*, are more difficult to learn and generally decrease the performance in the segmentation task. The dataset corresponding to the finely annotated images is already split into three subsets: training set with 2975 samples, validation set with 500 samples, and test set with 1525 samples. The test set labels are not publicly available.

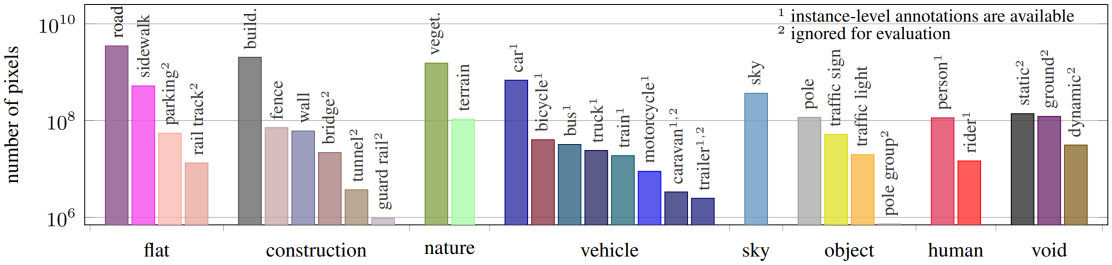

Figure 4.1: Cityscapes class frequencies. The plot highlights the rare classes of the dataset such as *bus*, *train*, and *motorcycle*. Figure excerpt from "The Cityscapes Dataset for Semantic Urban Scene Understanding" [18].

4.2 GTA5

GTA5 [16] is a synthetic image dataset consisting of 24966 labeled images with resolution 1914×1052 obtained from the open-world Grand Theft Auto 5 video game. The images are always captured from the car perspective in the streets of American-style virtual cities (see an example in Figure 4.2), and the labels, which are dense pixel-level annotations using the 19 semantic classes of Cityscapes [18], are used for segmentation tasks. The dataset comes without splits in a single training set and, typically, in tasks such as Domain Adaptation and Domain Generalization, where it is ubiquitous, it is used as a source dataset to solve Semantic Segmentation while the final performance is assessed on a real target dataset such as Cityscapes [18] or, for example, ACDC [22]. Not by chance, the class frequencies of GTA5 are very close (class-wise) to those of Cityscapes, so the shift from one to the other, in both UDA and DG, is

attenuated by similar frequencies of the object categories in the scenes. The need for synthetic datasets such as GTA5 was evident at that time due to the time-consuming labeling process of real images and, to date, it remains the most used dataset, along with SYNTHIA [17], when it comes to generalize to real data starting from synthetic ones since large-scale real segmentation datasets are still missing.

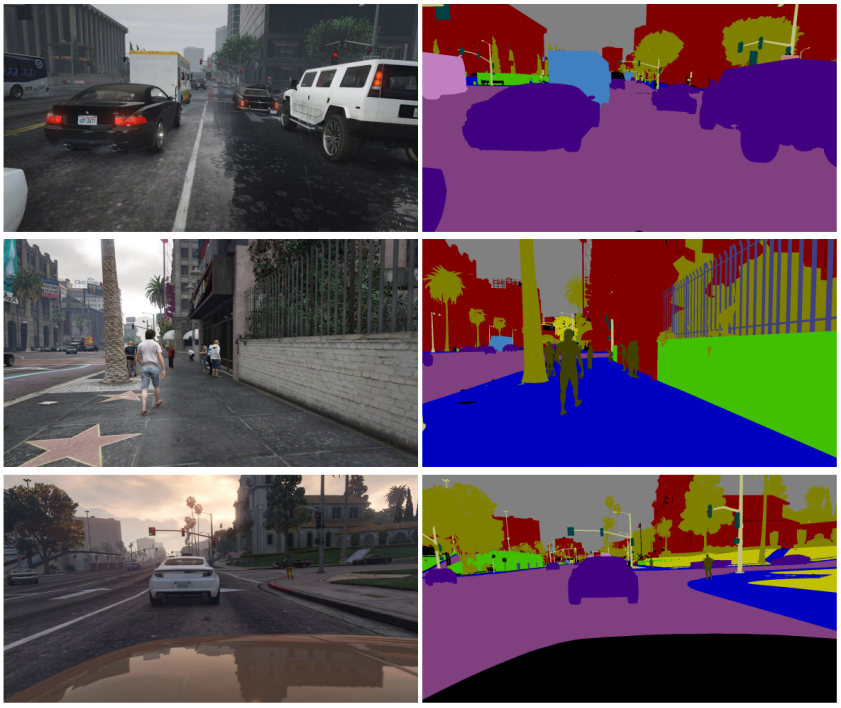

Figure 4.2: GTA5 sample images with their corresponding ground-truth segmentation maps. Note that, in these samples, the ground-truth labels do not follow the standard Cityscapes color-map [18]. Figure excerpt from "Playing for Data: Ground Truth from Computer Games" [16].

4.3 ACDC

The Adverse Conditions Dataset with Correspondences (ACDC) [22] is a real image dataset thought for testing models robustness on semantic driving scene understanding against different adverse visual conditions (sample images in Figure 4.3). The images are accurately chosen from a set of videos recorded at 1920×1080 resolution while driving around with a car in Switzerland, mostly in urban areas but also on highways and rural regions. The former version of the dataset is divided by the adverse conditions and counts 1006 nighttime, 1000 rainy, 1000 foggy, and 1000 snowy images.

Figure 4.3: ACDC [22] sample images show fog, night, rain, and snow adverse conditions column-wise, respectively. Figure excerpt from "Object Detection in Adverse Weather for Autonomous Driving through Data Merging and YOLOv8" [25].

The samples together sum up to 4006 pixel-level labeled images split into train, validation, and test sets, each comprising, respectively, 1600, 406, and 2000 images. A subsequent update of the dataset leveraged the camera GPS coordinates to introduce image-level correspondences between normal-condition (no adverse conditions) and adverse-condition images to allow normal-to-adverse generalization tasks while increasing the total dataset size to 5509 images. The semantic classes are the same as Cityscapes, thus facilitating real-to-real tasks where the dataset is often employed along with others like Mapillary Vistas [20] and BDD100K [19]. As in Cityscapes, the test set labels are not publicly available.

4.4 Mapillary Vistas

Mapillary Vistas [20] (or just Mapillary) is a real image dataset comprising 25000 pixel-level labeled driving scene images from all around the world (it covers parts of Europe, North and South America, Asia, Africa and Oceania) captured at different weather, season and daytime conditions and, with different imaging devices (*e.g.,* mobile phones and action cameras) and

by differently experienced photographers (sample images in Figure 4.4). The images are all in high definition, at least at FHD resolution (1920×1080), and 5 times more than those of the Cityscapes dataset [18].

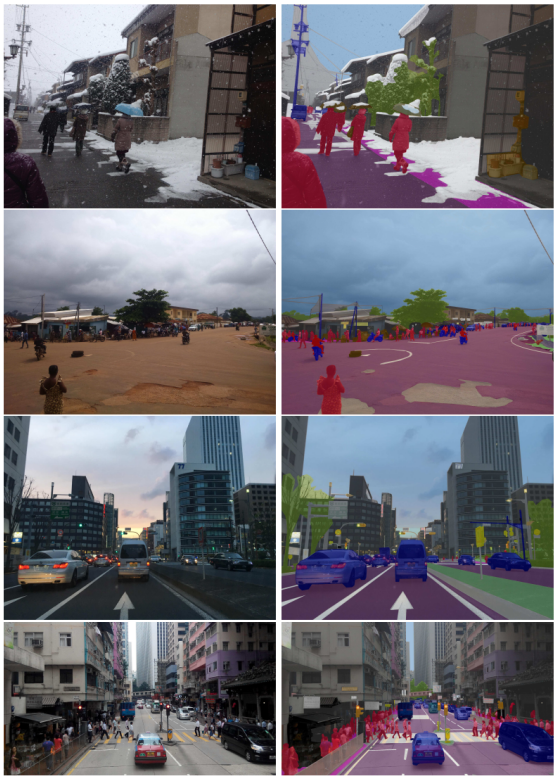

Figure 4.4: Mapillary Vistas sample images with their corresponding ground-truth segmentation maps. Figure excerpt from "The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes" [20].

The freely available dataset provides labels for 66 semantic classes, while the commercial one increases the number of classes up to 100. The dataset split consists of 18000 training, 2000 validation, and 5000 test images for which the labels are not publicly available, as common practice, for fair comparisons. Vistas is typically used, similarly to BBD100K [19], as the target dataset for synthetic-to-real and real-to-real generalization tasks.

4.5 SELMA

SEmantic Large-scale Multimodal Acquisitions (SELMA) [24] is a multimodal synthetic dataset for autonomous driving built using a modified version of the CARLA simulator [26] (game-engine). The multi-modality resides in the multiple sensors used to acquire the data as, in fact,

images are captured in 30909 independent locations from 7 RGB cameras, 7 depth cameras, 7 semantic cameras, and 3 LiDARs coupled with semantic information. Each acquisition is generated in variable weather, daytime, and viewpoint conditions and across 8 maps, totalizing 216 unique settings.

Figure 4.5: SELMA RGB samples from the DESK viewpoint for all weather conditions at Noon daytime (the location is the same for all images). Figure excerpt from "SELMA: SEmantic Large-scale Multimodal Acquisitions in Variable Weather, Daytime and Viewpoints" [24].

Semantic labeling, for both camera and LiDAR data, is accomplished for 36 distinct classes while maintaining a complete class overlap with the Cityscapes dataset [18]. The 30909 pixel-level labeled samples come in 1280×640 resolution and are optionally split into training (24735 samples), validation (3087 samples), and test (3087 samples) sets. The test labels are, therefore, publicly available to the research community. The dataset is a valid alternative to the standard GTA5 [16] for DA and DG semantic segmentation tasks with the key addition of the better weather and daytime variability, and it has also around 6K samples more compared to the latter. Some RGB samples from the DESK viewpoint for all weather conditions at Noon daytime are shown in Figure 4.5.

5 Method

In this chapter, the methodology, comprising the models and the architectures used in the experiments, is outlined. Section 5.1 presents the models employed to solve the Domain Generalization Semantic Segmentation task, while Section 5.2 discusses the architectures and the method chosen to tackle the problem.

5.1 Models

This section introduces the models that are used in the experiments and describes them from a high level with a focus on the architectural side while highlighting important details useful to understand the methodology later proposed. Section 5.1.1 presents CLIP [6], a VLM model for image-text matching, Section 5.1.2 presents Mask2Former [7], a vision decoder for segmentation tasks.

5.1.1 CLIP

Contrastive Language-Image Pre-training (CLIP) [6] is an efficient method of learning images from natural language supervision that leverages noisy image-text pairs to perform Contrastive Learning (CL) and achieve excellent zero-shot performance on over 30 existing image datasets w.r.t. prior task-specific image-only supervised models. To this end, the authors built a very large dataset made of 400 million image-text pairs collected from a variety of publicly avail-

able Internet sources, called WIT for WebImageText, and having a total word count similar to WebText, the dataset used to train GPT-2 [27]. Following the typical CL approach, given a batch of N (image,text) pairs, CLIP is trained to match the correct pairs among all the possible $N \times N$ combinations, by maximizing the cosine similarity between the correct N text and image embedding pairs while minimizing that of the remaining incorrect $N^2 - N$ ones. The text and image embeddings lie in a shared multi-modal embedding space where the direction of vectors are similar when they convey similar information, in this case, similar visual concepts. The architecture of CLIP is simple, consisting of a text and image encoders sharing the same layer architecture, namely, the Multi-Head Self-Attention (MHSA) (Section 2.1.1). The text encoder, named Text Transformer, and the image encoder ViT [9] (Section 2.2) are then directly updated via the last layer `[CLASS]` token embeddings through which the similarity score is computed. The original work employs, as Vision Transformer encoder, ViT-B/32, ViT-B/16, and ViT-L/14 while the Text Transformer encoder follows [27] and uses causal attention mask with maximum sequence length capped at 76. Inference on image classification datasets is carried out by feeding the text encoder with prompts in the form "`a photo of a {label}.`" for each label and then classifying by similarity to exploit the original pre-training method.

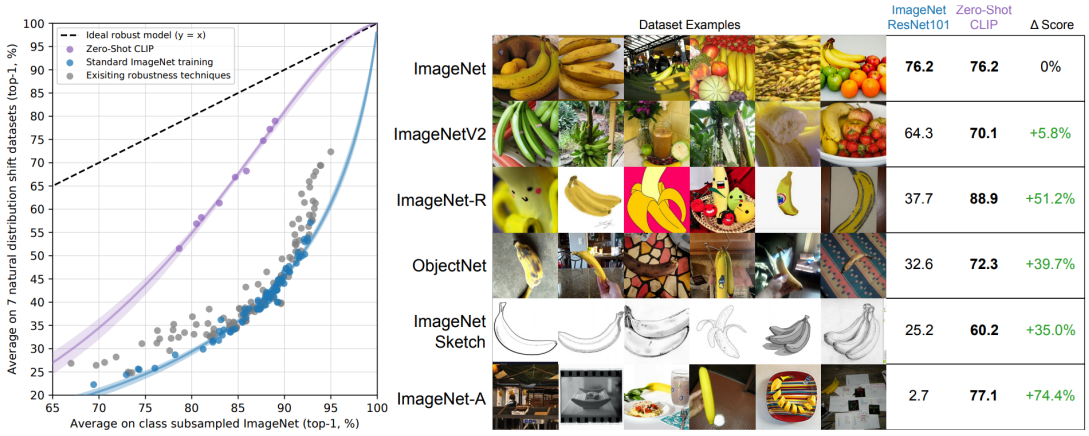

Figure 5.1: (Left) Zero-shot CLIP is much more robust to domain shift than any standard ImageNet model, closing the gap with the ideal robust model (dashed line), namely the model achieving ImageNet performance on all the natural distribution datasets, by up to 75%. (Right) The performance of the best zero-shot CLIP in its family is compared with ResNet101 [13] - a model that achieves the same ImageNet validation performance - on the bananas class occurring in 5 of the 7 natural distribution datasets. Figure excerpt from "Learning Transferable Visual Models From Natural Language Supervision" [6].

The most relevant result, w.r.t. to our work, is CLIP's impressive zero-shot robustness to domain shifts that strongly surpasses that of models solely trained on ImageNet [28], indeed, as Figure 5.1 shows, the model clearly outperforms, in average on 7 natural distribution shift

datasets, even models trained with ad-hoc robustness techniques.

5.1.2 Mask2Former

Masked-attention Mask Transformer (Mask2Former) [7] is a universal segmentation model capable of addressing all segmentation tasks (semantic, instance, and panoptic) using the same architecture. Built upon its predecessor MaskFormer [29], Mask2Former solves the segmentation problem by performing mask classification, that is, by computing a set of binary masks and a class probability distribution for each of them. The advantages of mask classification over per-pixel classification - typically used in semantic segmentation - are, of course, generalization and adaptability to all segmentation tasks since the semantics of segments predicted in the binary masks are arbitrary as, in fact, they can be used to represent both instances and categories.

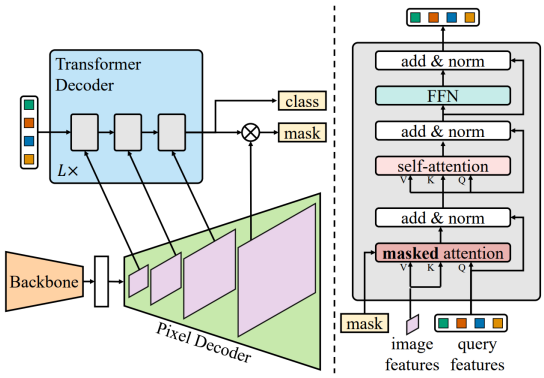

Figure 5.2: (Left) Mask2Former architecture comprising pixel decoder and transformer decoder. (Right) Transformer decoder layer architecture comprising Masked Cross-Attention with randomly initialized queries followed by Self-Attention. Figure excerpt from "Masked-attention Mask Transformer for Universal Image Segmentation" [7].

Mask2Former architecture (Figure 5.2) follows Deformable DETR [10] one so consists of a pixel decoder comprising a stack of Deformable Self-Attention layers followed by transformer decoder layers composed of, in order, Cross-Attention with the object queries and Self-Attention (see Section 2.1 for details of these attention mechanism variants). In DETR [30] `100` feature vectors called object queries are used as Regions of Interest (RoIs) to perform object detection and, subsequently, in MaskFormer [29] to solve the segmentation tasks. Mask2Former improves upon its predecessor and contributes with multiple novel additions, first introduces masked (cross) attention in the transformer decoder module to focus the attention on objects or regions based on the segments predicted in the previous layer to improve performance and

speed up convergence, second the authors leverage multi-scale high-resolution features to improve the model ability to predict small objects/regions and, finally, let the object queries, originally zero-initialized, be learnable from the beginning. Because of the high memory requirements - already present in MaskFormer [29] - they also avoid computing the exact loss for the binary masks and instead sample a few points at random and compute the loss only for those, resulting in a $3\times$ training memory reduction. Mask2Former performs on par or better than specialized models in all segmentation tasks using the same architecture and generalizing well across datasets.

5.2 Architectures

This section describes the architectures of the models actually used in the experiments of this work and introduces the main method chosen to solve the DGSS task that exploits language priors information. Section 5.2.1 presents a image-only architecture that we use as the baseline for our more advanced experiments; the results obtained from the models following this architecture are necessary in order to assess the effectiveness and validity of our method, explained in Section 5.2.2, exploiting the language priors.

5.2.1 Image

In order to solve the DGSS problem, a vision encoder and decoder have to be chosen. In our case, we opted, to simplify, for two different vision encoders sharing the same architecture, namely ViT and CLIPViT, and a single vision decoder, Mask2Former.

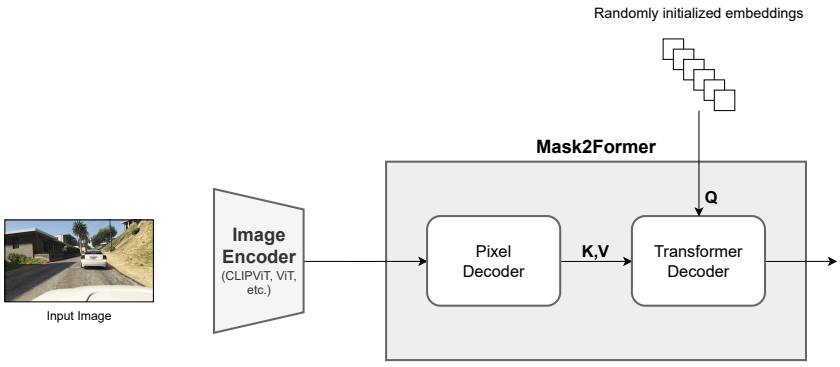

Figure 5.3: The architecture of the model for the image-only experiments. The vision encoder can be one of choice, whilst the vision decoder is Mask2Former. No learnable neck is employed.

The motivations behind these choices are mainly driven by the excellent results Transformer-based models achieved in many vision tasks to date, and, regarding the Mask2Former decoder, it is because of its versatility and scalability properties with the possibility of a multi-modal usage (as done in the current SOTA tqdm [5] and in our method, later discussed). Figure 5.3 depicts the chosen architecture for the image-only experiments. The diagram shows the simplest encoder-decoder design, highlighting the interchangeability of the encoder (it can be ViT, SegFormer, ResNet, etc.). Contrary to the common practice in literature, we do not employ a learnable neck for upsampling the encoder feature maps. Even though this choice probably impacts the final performance, it drastically reduces the computational need, and the resulting model is lighter. We collect the results of various experiments to create a solid baseline for our true method involving language priors (Section 5.2.2). Regarding the indices of the 3 features maps passed to Mask2Former, we follow the prior literature ([3, 4, 5]) and select the 6^{th}, 8^{th}, and 12^{th} features maps (out of the 12 total ones).

5.2.2 Image-Text

Extending the architecture introduced in Section 5.2.1 to one that leverages natural language priors and facilitates the convergence to the DGSS solution is not straight-forward but starting from the results obtained from the contributions of the previous works in the field (see Chapter 3) helps us moving to the correct direction.

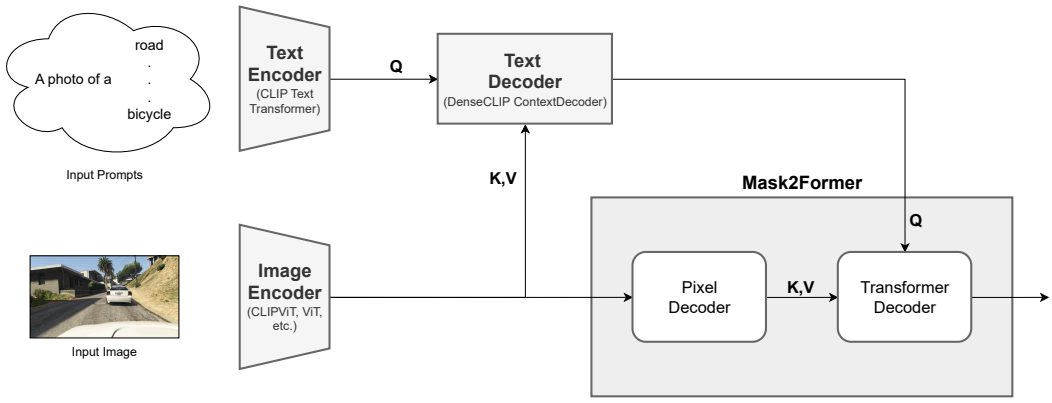

Figure 5.4: The architecture of the model for the image-text experiments. The vision encoder can be one of choice whilst the vision decoder is Mask2Former. The text encoder is CLIP Text Transformer, while the text decoder is DenseCLIP Context Decoder. No learnable neck is employed.

DenseCLIP [3] introduces the Context Decoder, an attention-based module that leverages

cross-attention to contextualize language priors with vision features and compute an auxiliary segmentation loss. tqdm [5] feeds the Mask2Former Transformer Decoder with the contextualized texts as batched queries. This work, on the same line as tqdm, exploits a Text Decoder for Mask2Former as a textual queries decoder and compares the resulting method with different vision encoders (ViT and CLIPViT) to assess its consistency across different vision features and the contribution of the language priors as well as their generalization properties needed in order to solve a Domain Generalization problem. The architecture leveraging the language priors is shown in Figure 5.4. Our method, as DenseCLIP, is model-agnostic since the vision and language encoders can be any models in literature even though, in this work, we focus on the vision-model interchangeability, keeping the text encoder - CLIP Text Transformer - fixed. As in recent works and in CLIP official work [6], we feed the text encoder with prompts in the form of `"a photo of a {class_name}."` with `class_name` corresponding to the Cityscapes class names, resulting in 19 text prompts. The text decoder contextualizes the text embeddings with all the last-layer vision tokens features (projected into the multi-modal space in the CLIPViT case) through 9 layers, each composed of Self-Attention, Cross-Attention, and a FFN. Additionally, to further strengthen our model, we employ the DenseCLIP auxiliary segmentation loss - Pixel-Text Matching Loss - computed by multiplying the normalized 19 output contextualized text embeddings by the normalized input vision features embeddings obtaining a small prediction map to which Cross Entropy is applied by using as labels the downsampled original labels.

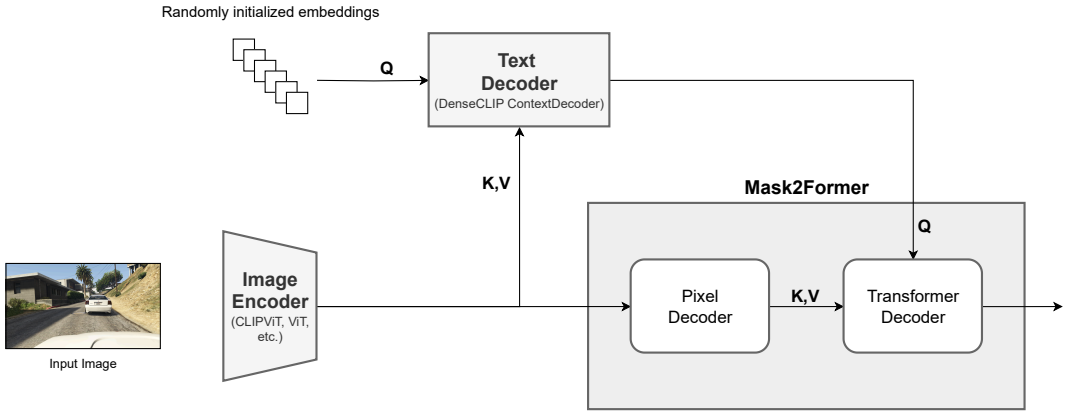

Figure 5.5: The architecture of the model for the image-text experiments with the Text Encoder ablated. The text embeddings are replaced with randomly initialized embeddings.

5.2.3 Image-Random

To assess the direct contribution of the text embeddings, we also experiment with a variant of the image-text architecture with the Text Encoder ablated. In these models, the 19 text embeddings are replaced with 19 randomly initialized embeddings. Figure 5.5 reports the third architecture.

6 Results

This chapter reports the results of the experiments performed in this work. Section 6.1 lists the main implementation details needed to understand the training settings and scenarios, Section 6.2 discusses the results obtained by the models trained only on images which constitute the baseline references for the rest of the work and, Section 6.3 discusses the results obtained by the models trained with both images and texts data leveraging our contributing method for solving the DGSS problem. Finally, Section 6.4 reports the final complete domain generalization results for the synthetic-to-real, real-to-real, and synthetic-to-synthetic scenarios.

6.1 Implementation Details

The experiments are performed with the models from the `huggingface` python library. We use as image encoders ViT/16 (ImageNet-21k pre-train) and CLIPViT/16 (WebText pre-train) and, as text encoder, CLIP Text Transformer while we use Mask2Former as image decoder and DenseCLIP Context Decoder as text decoder. We do not use any kind of learnable neck to upsample the image encoder feature maps so as to speed up the experiments and facilitate the combination of the vision and language embeddings (see more details about the neck in Section 5.2). To study the effects of the method on a smaller model first, we use TinyCLIP/16 (ViT-8M/16) [31] - a smaller distilled version of CLIP obtained via affinity mimicking and weight inheritance strategies - with a shallow Mask2Former variant that uses $\frac{1}{3}$ of the original number of layers - we name it Mask2Former-Small (M2F-S). We set batch size at 2 and train

the models on the synthetic GTA5 dataset for 40k iterations, randomly cropping images at 512×512 resolution with no further augmentations. The sliding-window inference is performed with the $40k^{th}$ iteration weights on the Cityscapes validation set to assess the synthetic-to-real generalization performance using the mean Intersection over Union (mIoU) as metric - expressed in percentage. As typical in UDA, during data preprocessing, we first rescale the GTA5 and Cityscapes images to 1024×512 and 1280×720 resolutions, respectively. When no text encoder is employed, Gradient Clipping with max norm value at 0.01 is used to regulate the Mask2Former large magnitude of gradients as in the original work. In the experiments involving textual data, the text decoder - DenseCLIP Context Decoder - is set, unless otherwise specified, to have 9 layers. Linear warmup of 1500 iterations and linear decay to zero schedules are used for the backbone encoders. The base learning rate is set at 1×10^{-5} with the image decoder Mask2Former learning $10\times$ faster. Because of the model training instability, we perform two runs for each experiment and choose the best one based on the in-domain curves. Finally, for the real-to-real experiments, we train the models on Cityscapes with the same training settings and evaluate them on the additional target datasets ACDC and Mapillary Vistas, whose samples are resized as in the synthetic-to-real experiments, using the mIoU and rPD (relative Performance under Domain shift) metrics. The synthetic-to-synthetic (SELMA $\rightarrow$ GTA5) experiments follow the same argument as the real-to-real ones.

6.2 Vision baselines

The models trained only on visual data follow the general architecture discussed in Section 5.2.1. The vision encoder is either ViT or CLIPViT, while the vision decoder is Mask2Former. We do not employ a learnable neck that upsamples the encoder feature maps, as typical in literature, because of the increased computational complexity and to create a consistent baseline for the more advanced models used in Section 6.3. The training scenario is synthetic-to-real, specifically, GTA5 $\rightarrow$ Cityscapes generalization with the mIoU metric (details are reported in Section 6.1). Table 6.1 (Top) reports both the in-domain (GTA5, 1^{th} column) and domain generalization (Cityscapes, 2^{th} column) resulting mIoU of the image-only experiments for the two vision encoders. Note that we obtain a validation set for GTA5 by sampling a random subset having the same size as the Cityscapes one (500) from the dataset and keeping it unseen during training.

Table 6.1 (Top) shows a much higher in-domain generalization performance by CLIPViT compared with the ViT counterpart, even though the former reaches a lower domain gener-

Method	mIoU (%)	
	GTA5 → Cityscapes	
ViT + M2F	69.83	**36.67**
CLIPViT + M2F	**76.05**	31.34
ViT (Frozen) + M2F	**66.94**	**33.82**
CLIPViT (Frozen) + M2F	54.79	32.86

Table 6.1: In-domain (GTA5) and domain generalization (Cityscapes) mIoU results of ViT and CLIPViT as vision encoders and Mask2Former (M2F) as vision decoder. **[Top]** The backbone is fine-tuned. **[Bottom]** The backbone is frozen. Gradient Clipping with max norm 0.01 is applied as in the original Mask2Former [7] work to stabilize the training.

alization mIoU. The reasons for CLIPViT's low DG performance could be addressed to the setup and/or to Mask2Former training instability, which we tackle as in the original work by using Gradient Clipping. In fact, we try a smaller variant of Mask2Former having 1 pixel decoder layer (instead of 6) and 3 transformer decoder layers (instead of 10) and report the results in Table 6.2 (Top).

Method	mIoU (%)	
	GTA5 → Cityscapes	
ViT + M2F-S	66.36	35.2
CLIPViT + M2F-S	**73.71**	**36.24**
ViT (Frozen) + M2F-S	**58.71**	**30.58**
CLIPViT (Frozen) + M2F-S	56.67	28.85

Table 6.2: In-domain (GTA5) and domain generalization (Cityscapes) mIoU results of ViT and CLIPViT as vision encoders and a smaller variant of Mask2Former (M2F-S) having 1 pixel decoder layer and 3 transformer decoder layers as vision decoder. **[Top]** The backbone is fine-tuned. **[Bottom]** The backbone is frozen. No Gradient Clipping, M2F-S learning rate kept constant in time, no standardization.

Table 6.2 (Top) reveals, in the DG scenario, the hidden potential of CLIPViT that reaches a similar value to its counterpart trained with the full version of Mask2Former. In-domain generalizations are, in this case, penalized by the smaller amount of parameters, but note that the aim of the experiment was rather to confirm the equal DG potential of both the vision encoders. Moreover, the smaller version of Mask2Former avoids the use of Gradient Clipping since more stable and allows a training with constant (base) learning rate.

To verify the zero-shot generalization capacity of the vision encoders, we perform the experiments again but with the backbone frozen, limiting the set of learnable parameters during the fine-tuning to those of Mask2Former (or Mask2Former-Small). Table 6.1 (Bottom) and Table 6.2 (Bottom) report the results of these experiments for both versions of the vision decoder.

In general, the results are lower when the backbone is frozen. We also notice that the ViT backbone, in this training setting (and in our setup), generalizes better in both domains.

6.3 Vision with Language Priors

6.3.1 Textual queries

This section reports and exhaustively discusses the results obtained in the experiments involving the natural language priors. As in Section 6.2, we experiment with, as vision encoders, ViT and CLIPViT and, as vision decoder, Mask2Former without a learnable neck. The addition of the text information to the visual processing imposes the need for a secondary decoder - the Text Decoder - that somehow has to learn to join language and vision representations to produce a meaningful and final performance-contributing output. We choose as text encoder CLIP Text Transformer - a causal model sharing the same architecture of ViT - and, as text decoder, the DenseCLIP Context Decoder module comprising layers composed of, in order, Self-Attention, Cross-Attention, and a Feed Forward Network (discussed in Section 5.2.2). The contextualized textual embeddings resulting from the text decoder function are fed to the Mask2Former Transformer Decoder as batched queries.

Method	**mIoU (%)**	
	GTA5 → Cityscapes	
ViT + TE + TD + M2F	55.10	30.57
CLIPViT + TE + TD + M2F	**77.74**	**37.95**
ViT + TE (Frozen) + TD + M2F	69.62	**39.68**
CLIPViT + TE (Frozen) + TD + M2F	**78.77**	39.47

Table 6.3: In-domain and domain generalization mIoU results exploiting a Text Encoder (TE) and a Text Decoder (TD) (architecture in Figure 5.4). Neither the vision (in CLIPViT) nor the text embeddings are projected into their shared text-image embedding space. **[Top]** The text encoder, CLIP Text Transformer, is finetuned. **[Bottom]** The text encoder, CLIP Text Transformer, is frozen. Note also that the text decoder is DenseCLIP Context Decoder with 9 layers. No Gradient Clipping, linear warmup, and linear decay for both vision and language encoders (details in Section 6.1), while the remaining modules have constant learning rate, no standardization.

The first experiment consists of trying our method with both the vision encoders, evaluating the models on both in-domain generalization (GTA5) and domain generalization (Cityscapes) to assess the performance in terms of the mIoU metric. As explained in Section 6.2, the validation set of GTA5 consists of 500 randomly chosen samples from the dataset kept unseen during training. Note that to achieve a fair comparison between the two vision encoders, we

project neither the vision (in CLIPViT) nor the text embeddings into their shared text-image embedding space. Table 6.3 (Top) reports the GTA5 → Cityscapes results (note that TE = Text Encoder and TD = Text Decoder) with the text encoder parameters pre-trained but still learnable during fine-tuning. It is clear, in this setting, the advantage of CLIPViT over ViT, expressed as a mIoU difference of 7% even without using the final projections to the shared embedding space, most probably due to the correlation in weights between CLIPViT and its text encoder CLIP Text Transformer originally trained together via CL. In literature, the text encoder is generally not fine-tuned to preserve the language priors, so in Table 6.3 (Bottom), we repeat the experiments with TE weights frozen.

Method	**mIoU (%)**	
	GTA5 → Cityscapes	
ViT + TE (Frozen) + TD + M2F-S	60.96	30.12
CLIPViT + TE + TD + M2F-S	**75.07**	**38.95**

Table 6.4: Generalization mIoU results exploiting a Text Encoder (TE) fed with prompts in the form `"a photo of a {class_name}."` and a Text Decoder (TD). Neither the vision (in CLIPViT) nor the text embeddings are projected into their shared text-image embedding space. Mask2Former-Small (M2F-S) as vision decoder. Details as in Table 6.3.

The results are surprising but not in the way one may expect; in fact, the two vision encoders seem to perform the same in domain generalization (GTA5 → Cityscapes). The reasons for that are to be investigated, but, in general, the possibilities are that either the Text Decoder manages to fuse vision and language information even when the encoders do not share a coupled pretraining (as in the ViT-CLIP Text Transformer case) or the Text Decoder computations themselves are enough to improve the final mIoU whenever the text embeddings are frozen (*i.e.,* language information is not crucial). Before trying to answer these doubts, we repeat again the experiments with Mask2Former-Small (introduced in Section 6.2 as a smaller variant of Mask2Former having only 1 pixel decoder layer and 3 transformer decoder layers) to appreciate the linkage with the vision baselines of Section 6.2. Table 6.4 shows the results.

Method	**mIoU (%)**	
	GTA5 → Cityscapes	
CLIPViT + RandEmb + TD + M2F	**78.77**	39.21
CLIPViT + RandEmb (Frozen) + TD + M2F	76.82	**40.68**

Table 6.5: Generalization mIoUs of the model ablated of its Text Encoder (architecture in Figure 5.5). RandEmb corresponds to 19 randomly initialized (`torch.randn`) embeddings used in place of the frozen text embeddings. Performances similar to those reported in Table 6.3 (Bottom).

Unexpectedly, when the text encoder is frozen, the models perform poorly with M2F-S (Mask2Former-Small) as the vision decoder. Despite the results in Table 6.3 (Bottom) foretell a superior efficacy of the method when TE (Text Encoder) is frozen, Table 6.4 seems to lead to the opposite conclusion. It may be that using the full version of M2F guarantees a better convergence and that, when using the small version of it, the models have fewer degrees of freedom to learn Vision-Language representations, so they also rely on the TE parameters to converge, but further experiments are needed to confirm this.
In Table 6.5, we verify with CLIPViT that, as suggested by the results in Table 6.3 (Bottom), the 19 text embeddings can be replaced with 19 randomly initialized vectors without affecting the performance. Indeed, the resulting model achieves roughly the same in-domain generalization mIoU and even higher domain generalization mIoU (when the embeddings are frozen), proving that, in this setting, the language information is not relevant or, at least, safely replaceable with randomly initialized embeddings.

6.3.2 Pixel-Text Matching Loss

As discussed in Section 5.2.2, we strengthen our method by adding an auxiliary segmentation loss, the DenseCLIP Pixel-Text Matching (PTM) loss (Figure 3.1), computed as Cross Entropy between the matrix obtained from the normalized contextualized texts and vision embedding matrices multiplication and, the downsampled labels - because of the vision encoder stride (16 in our experiments).

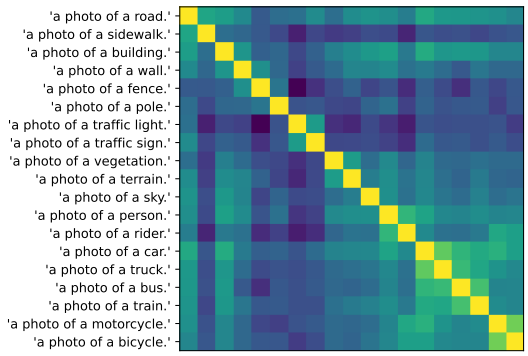

Figure 6.1: Similarity map obtained via matrix multiplication of the normalized text embeddings (output of the text encoder). The y-axis reports the original corresponding prompts for each Cityscapes class.

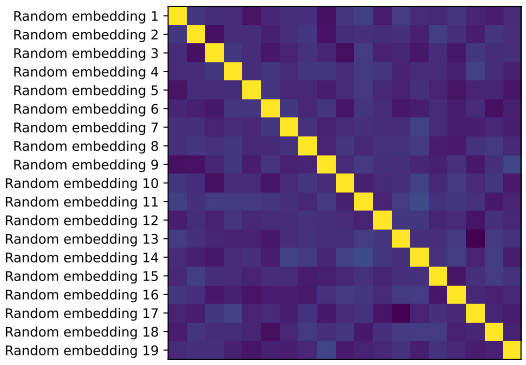

Figure 6.2: Similarity map obtained via matrix multiplication of the normalized random embeddings belonging to the standard normal distribution created using the `torch.randn` function.

In this new setting, we project the text `[CLASS]` token and all the vision tokens (in the ViT

case, we exploit the ViT pooler final layer) to their shared multi-modal embedding space (originally used to perform Contrastive Learning), as done in DenseCLIP [3]. We notice that CLIP Text Transformer's final linear projection follows a Layer Normalization (LN), so we know that the raw text vectors follow a Normal distribution. We then visualize the 19×19 similarity maps obtained by normalizing the embedding vectors over the channel dimension and multiplying each other in a matrix multiplication (recall that Cityscapes has 19 classes). The plots are reported in Figure 6.1 and Figure 6.2, respectively, for the text embeddings and the random embeddings obtained from the `torch.randn` function as standard normal vectors.

Method	**mIoU (%)**	
	GTA5 $\rightarrow$ Cityscapes	
ViT + TE (Frozen) + TD + PTM + M2F	72.32	38.78
CLIPViT + TE (Frozen) + TD + PTM + M2F	73.73	**41.77**
CLIPViT + RandEmb + TD + PTM + M2F	69.16	33.29
CLIPViT + RandEmb (Frozen) + TD + PTM + M2F	**69.23**	**38.22**

Table 6.6: Generalization mIoU results exploiting the DenseCLIP Pixel-Text Matching (PTM) auxiliary segmentation loss. **[Top]** The text embeddings are frozen. **[Bottom]** The text embeddings are replaced with randomly initialized embeddings created using the `torch.randn` function.

We keep the Text Encoder (TE) frozen as in literature and perform the experiment with both the vision encoders using the new training loss defined as follows:

$$\mathcal{L}_{model} = \mathcal{L}_{M2F} + \lambda_{PTM}\mathcal{L}_{PTM} \tag{6.1}$$

where $M2F$ stays for Mask2Former and PTM stays for Pixel-Text Matching. As the prior work [4] advises, to avoid overfitting any dataset, we set $\lambda_{PTM} = 1$. Therefore, the loss in Equation 6.1 becomes:

$$\mathcal{L}_{model} = \mathcal{L}_{M2F} + \mathcal{L}_{PTM} \tag{6.2}$$

Table 6.6 (Top) reports the results. We see how, compared to the results in Table 6.3 (Bottom), the in-domain generalization decreased by about 5% mIoU with a domain generalization improvement of around 2% mIoU, indicating a better generalization capacity of the overall method which this time exploits the CLIP final projections layers and an additional segmentation loss. Table 6.6 (Bottom) reports the experiments with CLIPViT in the same setting but with randomly initialized embeddings (both in the learnable and frozen case) in place of the text embeddings to appreciate the contribution of the language information. Conversely to

the results found in Table 6.5, in this setting, the model using random embeddings does not reach the performance of that using the text embeddings.

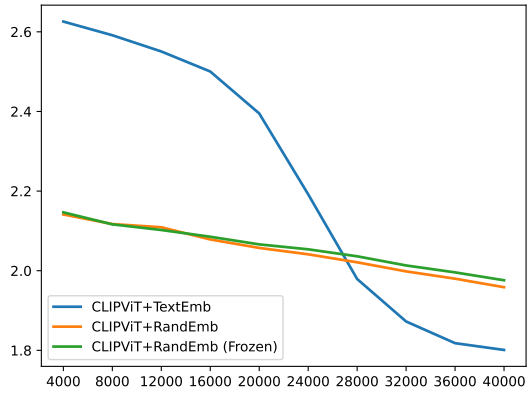

Figure 6.3: Pixel-Text Matching (PTM) loss on the GTA5 validation set. Evaluations are performed every 4k iterations.

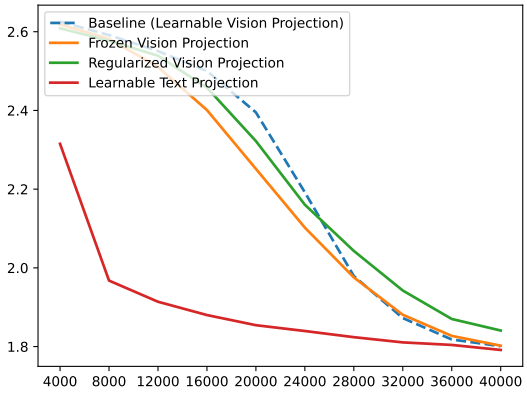

Figure 6.4: Pixel-Text Matching (PTM) loss on the GTA5 validation set. The three proposed solutions to improve on the baseline mIoU decaying problem are tested. Evaluations are performed every 4k iterations.

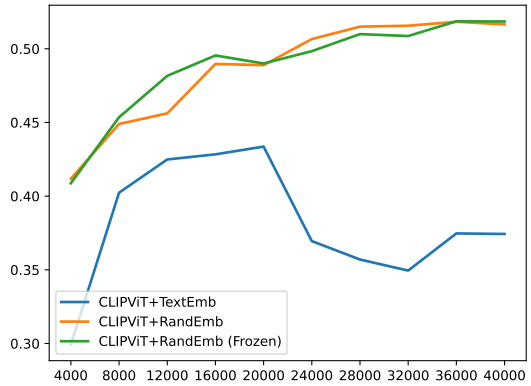

Figure 6.5: Pixel-Text Matching (PTM) mIoU on the GTA5 validation set. Evaluations are performed every 4k iterations.

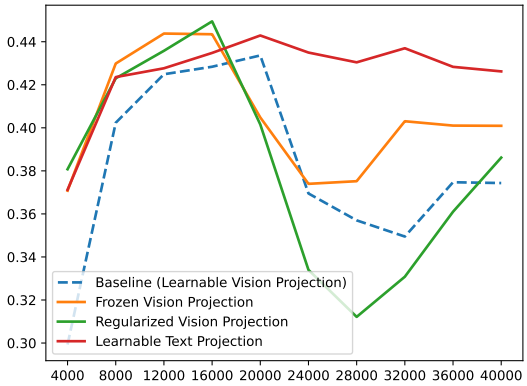

Figure 6.6: Pixel-Text Matching (PTM) mIoU on the GTA5 validation set. The three proposed solutions to improve on the baseline mIoU decaying problem are tested. Evaluations are performed every 4k iterations.

To visualize the efficacy of the auxiliary loss, Figure 6.3 and Figure 6.5 plot, respectively, the loss and mIoU of the small segmentation map, obtained as explained before (see Figure 3.1), on the GTA5 validation set. Note that we evaluate our models every 4k iterations during training. Even if, in this training settings, the text embeddings allow for higher performance on the final task (as Table 6.6 shows), it could not be said, at least apparently, for the inner segmentation task. In fact, the PTM loss with text embeddings is smoother, but the corresponding mIoU is lower and decaying after around 20k iterations, probably due to the vision projection layer that,

during fine-tuning, loses correlation with the text projection one (which is frozen). To tackle this, we propose three solutions: first, we try freezing the vision final projection, second we try to regularize the vision projection by the Mean Square Error (MSE) between its predictions and those of its pretrained original version (implying the use of another instance of the encoder which remains frozen) and, third, we try to fine-tune also the text final projection at the same learning rate keeping the rest of the text encoder frozen. In the second solution, the total final loss can be formulated as follows:

$$\mathcal{L}_{model} = \mathcal{L}_{M2F} + \mathcal{L}_{VReg} + \mathcal{L}_{PTM} \tag{6.3}$$

where $M2F$ stays for Mask2Former, $VReg$ stays for Vision Regularization and, PTM stays for Pixel-Text Matching. Again, analogously to Equation 6.1 and Equation 6.2, we do not employ a multiplier for $\mathcal{L}_{VReg}$ to not overfit. Figure 6.4 and Figure 6.6 show, respectively, the PTM loss and PTM mIoU of the three proposed solutions along with the baseline to appreciate the differences.

Method	**mIoU (%)**	
	GTA5 $\rightarrow$ Cityscapes	
I: Frozen Vision Projection	71.38	**37.30**
II: Regularized Vision Projection	66.82	28.21
III: Learnable Text Projection	**77.54**	37.13

Table 6.7: Generalization mIoUs of the three proposed solutions for the decaying PTM mIoU problem.

The plots, as expected, lead to the conclusion that fine-tuning both the vision and text final projection layers at the same learning rate yields the best results. Nevertheless, Table 6.7 reports poor domain generalization mIoUs for all three solutions, particularly for the regularization solution, indicating, in this case, a low correlation between the PTM mIoU and the final generalization mIoU.

For the sake of completeness, Table 6.8 (Top) shows that, in practice, it is possible to achieve the performance of the models in Table 6.6 (Top) using randomly initialized embeddings in place of the text embeddings by feeding the Text Decoder with the last hidden layer tokens without involving the final vision projection layer (conversely to the standard DenseCLIP method). By comparing the model using frozen random embeddings in Table 6.8 (Top) with that in Table 6.6 (Top) using the text embeddings we see equal domain generalization performances but a difference of almost 5% in the in-domain mIoU may indicating a better train-to-test generalization capacity of the model exploiting language priors. In general, we note that the models

Method	mIoU (%)	
	GTA5 → Cityscapes	
CLIPViTw/oProj + RandEmb + TD + PTM + M2F	**78.29**	39.42
CLIPViTw/oProj + RandEmb (Frozen) + TD + PTM + M2F	78.27	**41.64**
ViTw/oPooler + RandEmb + TD + PTM + M2F	**70.39**	38.45
ViTw/oPooler + RandEmb (Frozen) + TD + PTM + M2F	68.23	**39.39**

Table 6.8: Generalization mIoU results exploiting the DenseCLIP Pixel-Text Matching (PTM) auxiliary segmentation loss. The text embeddings are replaced with randomly initialized embeddings created using the `torch.randn` function. **[Top]** CLIPViTw/oProj means using the encoder without involving its final projection layer originally used in pretraining to perform CL. **[Bottom]** ViTw/oPooler means using the encoder without involving its final pooler layer originally used in pretraining for classification.

exploiting frozen representations generalize better independently of the nature of the representations (both random and textual representations can work). These constant vectors act as external priors (or initializers, in the random case) that supervise the segmentation task and stabilize the training, leading to faster and better convergence. This is particularly true for the CLIP model, probably because of its pretraining, and thinking about achieving the same results on a text-encoder-decoupled vision encoder such as ViT with ImageNet pretraining may not be realistic in practice but, of course, further study is needed to confirm this fully. Again, for completeness, we also report the results of the experiments with ViT as the vision encoder and random embeddings in place of the text embeddings in Table 6.8 (Bottom).

6.4 Domain generalization

In this section, we evaluate the best vision models of Section 6.2 and vision-language models of Section 6.3 on multiple real target datasets to assess their effective synthetic-to-real domain generalization capability. Moreover, we also re-train the models on Cityscapes and SELMA [24] to assess their real-to-real and synthetic-to-synthetic generalization performances, respectively. Apart from the mIoU scores, for each model, we also compute the rPD (relative Performance under Domain shift) metric, introduced in VLTSeg [4] to assess the domain shift resilience of models, that given a source dataset S and a set of target datasets $\{T_k\}$ is defined as:

$$\text{rPD} = \frac{\frac{1}{K}\sum_{k=1}^{K} \text{mIoU}^{T_k}}{\text{mIoU}^{S}} \tag{6.4}$$

6.4.1 Synthetic-to-Real

For the models in Section 6.2 (image) and Section 6.3 (image-text) the source dataset is GTA5, so we evaluate them with their $40k^{th}$ iteration weights on the real target datasets ACDC (night, fog, rain, snow) and Mapillary Vistas (more about these datasets in Chapter 4) in terms of final mIoU and rPD (Equation 6.4) percentages reporting again also the in-domain and Cityscapes mIoUs in order to visualize all the results together.

Synthetic-to-Real

	Method [Table]	**mIoU (%)**					**rPD (%)**
		G → G	G → C	G → A	G → V	DG Avg	
CLIPViT	CLIPViT [6.2]	76.13	38.70	31.00	45.32	38.34	50.36
	CLIPViT + Text [6.6]	73.73	**41.77**	30.01	46.55	39.44	53.49
	CLIPViT + Text [6.6]*	73.62	40.68	**34.04**	46.67	40.46	**54.96**
	CLIPViT + Random [6.8]	**78.27**	41.64	33.81	**48.60**	**41.35**	52.83
	CLIPViT + Random [6.8]*	74.26	38.58	33.06	46.42	39.35	52.99
ViT	ViT [6.1]	70.71	36.82	30.28	42.18	36.42	51.51
	ViT + Text [6.6]	**72.32**	38.78	28.92	43.87	37.19	51.42
	ViT + Random [6.8]	68.23	**39.39**	**31.05**	**45.19**	**38.54**	**56.49**

Table 6.9: Synthetic-to-Real generalization final results of the best models found in Section 6.2 (image) and Section 6.3 (image-text) that are trained on GTA5 (G) and evaluated after 40k iterations on Cityscapes (C), ACDC (A), and Mapillary Vistas (V). Performance comparison is expressed in terms of both mIoU and rPD (relative Performance under Domain shift) metrics. The rPD metric consists of the ratio between the mean mIoU on the target datasets and the mIoU on the source dataset. The * mark means that we used a 3-layer Text Decoder (instead of a 9-layer one) as in the original DenseCLIP [3] work. Note also that G → G represents the in-domain generalization.

Table 6.9 hence reports the final domain generalization (synthetic-to-real) results for both metrics. It is clear the advantage of using CLIPViT as the vision encoder instead of ViT, especially when the Text Decoder is employed. Regarding CLIPViT, from Table 6.9, we understand how much our method improves the final DG mIoU even though it is still not clear whether the text embeddings are superior to the random ones. Of course, they achieve a similar DG mIoU score, but, in general, we notice that the gap between the in-domain and the DG mIoUs is larger for the models trained with the text embeddings, as the rPD scores show. This may be attributed to the language priors and their superior train-to-test generalization capacity.

6.4.2 Real-to-Real

We re-train the models of Section 6.2 (image) and Section 6.3 (image-text) on Cityscapes for 40k iterations and evaluate them with their last iteration weights on the remaining real target

datasets ACDC (all adverse conditions) and Mapillary Vistas (more about these datasets in Chapter 4) in terms of final mIoU and rPD (Equation 6.4). We also report the in-domain mIoU in order to understand the rPD score.

Real-to-Real

	Method [Table]	mIoU (%)				rPD (%)
		C → C	C → A	C → V	DG Avg	
CLIPViT	CLIPViT [6.2]	**69.22**	39.27	49.55	44.41	64.16
	CLIPViT + Text [6.6]	67.98	**40.65**	**51.54**	**46.09**	**67.80**
	CLIPViT + Text [6.6]*	66.46	40.15	46.67	43.41	65.32
	CLIPViT + Random [6.8]	66.59	39.71	49.35	44.53	66.87
ViT	ViT [6.1]	59.65	36.81	44.24	40.52	67.94
	ViT + Text [6.6]	**60.59**	**37.83**	**44.40**	**41.11**	67.86
	ViT + Random [6.8]	54.92	36.91	43.50	40.20	**73.21**

Table 6.10: Real-to-Real generalization final results of the best models found in Section 6.2 (image) and Section 6.3 (image-text) re-trained on Cityscapes (C) and evaluated after 40k iterations on ACDC (A) and Mapillary Vistas (V). Performance comparison is expressed in terms of both mIoU and rPD (relative Performance under Domain shift) metrics. The rPD metric consists of the ratio between the mean mIoU on the target datasets and the mIoU on the source dataset. The * mark means that we used a 3-layer Text Decoder (instead of a 9-layer one) as in the original DenseCLIP [3] work. Note also that C → C represents the in-domain generalization.

Table 6.10 then reports the final domain generalization (real-to-real) results. Again, as in the synthetic-to-real results (Table 6.9), CLIPViT remains superior to ViT in all the configurations tried. In this real-to-real scenario, we notice a higher performance of the model using the text embeddings over the random ones that also results in a higher rPD score, indicating a larger gap between in-domain and DG mIoUs (as found also in the synthetic-to-real experiments, Section 6.4.1). Regarding ViT, we notice a very high rPD score when using the method with random embeddings (the same holds in Table 6.9 for the synthetic-to-real results), proving that the method, in general, brings benefits and is superior to the baseline vision model. In the ViT case, we note that it is not expected for the text embeddings to work better than the random ones since the vision and text encoders are decoupled (they do not share a coupled pretraining).

6.4.3 Synthetic-to-Synthetic

For the sake of completeness and to assess the superiority of the text embeddings over the random ones, we also test the models on the synthetic-to-synthetic scenario. We then re-train the models of Section 6.2 (image) and Section 6.3 (image-text) on SELMA (information about this

dataset in Section 4.5) for 40k iterations and evaluate them with their last iteration weights on GTA5 in terms of final mIoU and rPD (Equation 6.4). We also report the in-domain mIoU in order to understand the rPD score.

Synthetic-to-Synthetic

	Method [Table]	mIoU (%)		rPD (%)
		S → S	S → G	
CLIPViT	CLIPViT [6.2]	69.28	36.60	52.82
	CLIPViT + Text [6.6]	75.68	39.32	51.96
	CLIPViT + Text [6.6]*	75.81	**44.60**	**58.83**
	CLIPViT + Random [6.8]	**76.13**	41.17	54.08
ViT	ViT [6.1]	63.59	33.23	52.26
	ViT + Text [6.6]	68.97	35.22	51.07
	ViT + Random [6.8]	**69.47**	**36.66**	**52.77**

Table 6.11: Synthetic-to-Synthetic generalization final results of the best models found in Section 6.2 (image) and Section 6.3 (image-text) re-trained on SELMA (S) and evaluated after 40k iterations on GTA5 (G). Performance comparison is expressed in terms of both mIoU and rPD (relative Performance under Domain shift) metrics. The rPD metric consists of the ratio between the mean mIoU on the target datasets and the mIoU on the source dataset. The * mark means that we used a 3-layer Text Decoder (instead of a 9-layer one) as in the original DenseCLIP [3] work. Note also that S → S represents the in-domain generalization.

Table 6.11 then reports the final domain generalization (synthetic-to-synthetic) results. In this scenario, the text embeddings work very well, outperforming, in the CLIPViT case, the random ones by $\approx$ 3.5% DG mIoU and the baseline vision model by 8% DG mIoU, thus achieving the highest rPD score. However, we note that these results do not imply a superiority of the text embeddings over the random ones since the synthetic-to-synthetic scenario is not used in practice and since these depend on our setup and on the specific architectures chosen. Regarding ViT, we notice the same trend of the synthetic-to-real (Section 6.4.1) and real-to-real experiments (Section 6.4.2), with the highest scores achieved when the random embeddings are employed, since, in this case, the vision and text encoders do not share a coupled pretraining.

7 Conclusion

The aim of the work was to assess the contribution of VLMs text embeddings in solving the DGSS task on multiple generalization scenarios over a variety of different autonomous driving datasets. The results of the experiments proved that our method, involving an architecture that exploits the Vision-Language features from the CLIP VLM, applied to the DGSS task, brings benefits and increases the performances, outperforming the baseline vision models fine-tuned solely on images. This holds true for all three generalization tasks, particularly for the real-to-real and synthetic-to-synthetic tasks for which the method seems to suit the most. We tried to apply our method to ViT, which does not share a coupled pre-training with a text encoder, and, as expected, achieved lower results, thus confirming the efficacy of VLMs on the task. We compared text embeddings to random ones for initializing the Text Decoder, and the former, overall, achieved the highest scores even though, from a computational point of view, the latter also proved to be valid initializers. Of course, further experiments with different architectures and models are needed to draw the final conclusions, but this is left as future work.

References

[1] M. Chen, Z. Zheng, and Y. Yang, "Transferring to real-world layouts: A depth-aware framework for scene adaptation," 2024. [Online]. Available: https://arxiv.org/abs/2311.12682

[2] M. Schwonberg, J. Niemeijer, J.-A. Termöhlen, J. P. Schäfer, N. M. Schmidt, H. Gottschalk, and T. Fingscheidt, "Survey on unsupervised domain adaptation for semantic segmentation for visual perception in automated driving," 2023. [Online]. Available: https://arxiv.org/abs/2304.11928

[3] Y. Rao, W. Zhao, G. Chen, Y. Tang, Z. Zhu, G. Huang, J. Zhou, and J. Lu, "Denseclip: Language-guided dense prediction with context-aware prompting," 2022. [Online]. Available: https://arxiv.org/abs/2112.01518

[4] C. Hümmer, M. Schwonberg, L. Zhou, H. Cao, A. Knoll, and H. Gottschalk, "Vltseg: Simple transfer of clip-based vision-language representations for domain generalized semantic segmentation," 2024. [Online]. Available: https://arxiv.org/abs/2312.02021

[5] B. Pak, B. Woo, S. Kim, D. hwan Kim, and H. Kim, "Textual query-driven mask transformer for domain generalized segmentation," 2024. [Online]. Available: https://arxiv.org/abs/2407.09033

[6] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," 2021. [Online]. Available: https://arxiv.org/abs/2103.00020

[7] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar, "Masked-attention mask transformer for universal image segmentation," 2022. [Online]. Available: https://arxiv.org/abs/2112.01527

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023. [Online]. Available: https://arxiv.org/abs/1706.03762

[9] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021. [Online]. Available: https://arxiv.org/abs/2010.11929

[10] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," 2021. [Online]. Available: https://arxiv.org/abs/2010.04159

[11] A. Kirillov, R. Girshick, K. He, and P. Dollár, "Panoptic feature pyramid networks," 2019. [Online]. Available: https://arxiv.org/abs/1901.02446

[12] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba, "Semantic understanding of scenes through the ade20k dataset," 2018. [Online]. Available: https://arxiv.org/abs/1608.05442

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[14] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," 2021. [Online]. Available: https://arxiv.org/abs/2103.14030

[15] Q. Sun, Y. Fang, L. Wu, X. Wang, and Y. Cao, "Eva-clip: Improved training techniques for clip at scale," 2023. [Online]. Available: https://arxiv.org/abs/2303.15389

[16] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," 2016. [Online]. Available: https://arxiv.org/abs/1608.02192

[17] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3234–3243.

[18] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," 2016. [Online]. Available: https://arxiv.org/abs/1604.01685

[19] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," 2020. [Online]. Available: https://arxiv.org/abs/1805.04687

[20] G. Neuhold, T. Ollmann, S. R. Bulò, and P. Kontschieder, "The mapillary vistas dataset for semantic understanding of street scenes," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 5000–5009.

[21] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "Segformer: Simple and efficient design for semantic segmentation with transformers," 2021. [Online]. Available: https://arxiv.org/abs/2105.15203

[22] C. Sakaridis, H. Wang, K. Li, R. Zurbrügg, A. Jadon, W. Abbeloos, D. O. Reino, L. V. Gool, and D. Dai, "Acdc: The adverse conditions dataset with correspondences for robust semantic driving scene perception," 2024. [Online]. Available: https://arxiv.org/abs/2104.13395

[23] Z. Wei, L. Chen, Y. Jin, X. Ma, T. Liu, P. Ling, B. Wang, H. Chen, and J. Zheng, "Stronger, fewer, & superior: Harnessing vision foundation models for domain generalized semantic segmentation," 2024. [Online]. Available: https://arxiv.org/abs/2312.04265

[24] P. Testolina, F. Barbato, U. Michieli, M. Giordani, P. Zanuttigh, and M. Zorzi, "Selma: Semantic large-scale multimodal acquisitions in variable weather, daytime and viewpoints," 2022. [Online]. Available: https://arxiv.org/abs/2204.09788

[25] D. Kumar and N. Muhammad, "Object detection in adverse weather for autonomous driving through data merging and yolov8," *Sensors*, vol. 23, no. 20, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/20/8471

[26] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," 2017. [Online]. Available: https://arxiv.org/abs/1711.03938

[27] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[28] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[29] B. Cheng, A. G. Schwing, and A. Kirillov, "Per-pixel classification is not all you need for semantic segmentation," 2021. [Online]. Available: https://arxiv.org/abs/2107.06278

[30] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," 2020. [Online]. Available: https://arxiv.org/abs/2005.12872

[31] K. Wu, H. Peng, Z. Zhou, B. Xiao, M. Liu, L. Yuan, H. Xuan, M. Valenzuela, Xi, Chen, X. Wang, H. Chao, and H. Hu, "Tinyclip: Clip distillation via affinity mimicking and weight inheritance," 2023. [Online]. Available: https://arxiv.org/abs/2309.12314

Acknowledgments

I would like to thank my supervisors for allowing me working on a topic I was interested in and for the help I received during the work.