MASTER THESIS

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering at the University of Applied Sciences Technikum Wien - Degree Program Mechatronics/Robotics

Visual Semantic Context Encoding for Data Harvesting and Domain Prediction

By: Andreas Kriegler, BSc

Student Number: 1810331016

Supervisors: Daniel Steininger, MSc (AIT);
Wilfried Wöber, MSc
Dr.rer.nat. Andrea Ojdanic

Vienna, September 15, 2020

FH University of Applied Sciences
TECHNIKUM WIEN

Declaration

“As author and creator of this work to hand, I confirm with my signature knowledge of the relevant copyright regulations governed by higher education acts (see Urheberrechtsgesetz /Austrian copyright law as amended as well as the Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien as amended).

I hereby declare that I completed the present work independently and that any ideas, whether written by others or by myself, have been fully sourced and referenced. I am aware of any consequences I may face on the part of the degree program director if there should be evidence of missing autonomy and independence or evidence of any intent to fraudulently achieve a pass mark for this work (see Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien as amended).

I further declare that up to this date I have not published the work to hand nor have I presented it to another examination board in the same or similar form. I affirm that the version submitted matches the version in the upload tool.“

Vienna, September 15, 2020 Signature

Kurzfassung

Visueller semantischer Kontext beschreibt die Zusammenhänge zwischen Objekten und deren Umgebung in Bildern. Eine Analyse dieses Kontexts kann Informationen liefern, mit Hilfe derer ein holistisches Verständnis von Bildszenen ermöglicht wird. Betrachtet man den Kontext über mehrere Bilder hinweg werden Domänen erkenntlich. Domänen bestimmen die vorwiegenden Umgebungen von Objekten und sind definiert durch charakteristische visuelle Merkmale. Menschen fällt es leicht diese semantischen Beziehungen zu verwenden um die Umgebungswahrnehmung zu fördern – im Bereich des maschinellen Sehens existieren jedoch noch wenige Arbeiten die dieses Kontextwissen umfassend verwenden.

Während Kontext oft implizit in künstlichen neuronalen Netzwerken gelernt wird, zeigt diese Arbeit eine Möglichkeit, Kontext explizit auszudrücken und diese statistische Darstellung in zweierlei Hinsicht zu verwenden. Die Verwendung des Kontexts ermöglicht es, in der Aggregierung eines Datensatzes irrelevante und ungewollte Bilder auszufiltern, ein wichtiger Schritt um die Abdeckung der Domänen für nachfolgende Lernaufgaben zu verbessern, was besonders notwendig ist, wenn mit öffentlich zugänglichen Datensätzen gearbeitet wird. Darüber hinaus wird Kontext dazu verwendet, um die Domänen bestimmter Objekte zu erkennen, was wiederum eine weitergehende Adaption der Parametersätze von domän-spezifisch trainierten Modellen ermöglichen könnte.

Das entwickelte Framework wird speziell im Bereich der Luftfahrt getestet, um Kontext von Flugzeugen aus den Datensätzen *ADE20K-SceneParsing*, *COCO-Stuff* und *PASCAL-Context* zu verarbeiten. Als Zwischenergebnis werden Statistiken auf diesen Datensätzen gezeigt, welche wiederum die Formulierung des abgeleiteten, vereinten Datensatzes "*SemanticAircraft*" ermöglichen. Drei unterschiedliche Methoden werden zur Klassifizierung der Domänen verwendet: Ein Algorithmus, der mit Schwellwerten arbeitet, sowie unüberwachte Clusteralgorithmen wie Bayesische Gaussische Mixtur-Modelle verwenden die explizite Repräsentation des Kontexts, während faltende neuronale Netzwerke mit den Bildern per-se sowie Domänannotierungen arbeiten. Allen Modellen gelingt die Klassifizierung der Domänen mit hoher Genauigkeit, wobei das neuronale Netzwerk mit Genauigkeiten von 69% bis 85%, abhängig von der Variante von *SemanticAircraft*, am performantesten ist. Die Ergebnisse entsprechen somit den Erwartungen – die mit den Mixtur-Modellen gefundenen Clusterstrukturen stimmen nicht zwangsmässig mit den definierten Domänen überein, aber ermöglichen eine allgemeinere Analyse der Kontextstatistiken.

Schlagworte: Semantischer Kontext, Datenaggregierung, Domänenklassifizierung, Luftfahrtbilder

Abstract

Visual semantic context describes the relationship between objects and their environment in images. Analyzing this context yields important cues for more holistic scene understanding. Domains are an extension of semantic context across multiple images, specified by visual features and form the environment objects most commonly appear in. Humans naturally use these semantic relations to improve their environment perception but in computer vision literature only a handful of works exist that exploit context to significant extent.

While context is often learned implicitly in neural networks, this work provides an explicit representation of context and utilizes context statistics in two ways. Using semantic context, irrelevant images can be filtered during data aggregation, a key step to improving domain coverage for a specific learning task, especially working with public datasets. Secondly, context is used to predict the domains of objects of interest, which could enable later model adaptation of fine-tuned models.

The framework is applied to the aerial domain, specifically the context around airplanes from *ADE20K-SceneParsing*, *COCO-Stuff* and *PASCAL-Context*. As intermediate results, the context statistics were obtained on these datasets to guide design and label-mapping choices for a merged dataset, referred to as *SemanticAircraft* in this work. Three different methods were employed to predict domains of airplanes: an original threshold-algorithm and unsupervised clustering via variational Bayesian mixture models use explicit context priors, a supervised CNN on the other hand works on input images with annotated domain-labels. All three models were able to achieve satisfactory prediction results, with the CNN obtaining highest accuracies of 69% to 85% depending on the subset of *SemanticAircraft*. The results therefore meet expectations – clusters found with the mixture models do not necessarily correspond to the predefined domains and instead allow a more general analysis of the context statistics.

Keywords: semantic context, data harvesting, domain prediction, aerial data

Acknowledgements

I would like to thank both of my thesis supervisors Daniel Steininger and Wilfried Wöber for their guidance during the writing of this thesis. The shared knowledge, critical comments and discussions were a significant component necessary to complete this thesis.
I would like to thank the Austrian Institute of Technology for supporting this project and the UAS Technikum Vienna for approving it.
Lastly I am grateful to my extended family for their everlasting support, compassion and kind words. The last few years have been the most exciting in my life by far, and with this thesis another major chapter comes to a close.

Contents

1 Introduction

Humans intuitively incorporate contextual information when trying to understand the environment they perceive. Objects appearing in an unfamiliar semantic context, or out-of-context objects [1] such as airplanes on a highway, attract the observer's attention since they are typically related to other domains. Incorporating this kind of prior information has the potential to improve computer vision (CV) models by assigning meaning to objects and is essential for solving upcoming challenges in scene understanding. In particular, autonomous systems operating in the real world struggle to stay robust when traversing multiple domains, or the domains look significantly different due to weather, atmospheric effects, or time of day. It stands to reason that such influencing factors can define a new domain itself if the factors dominate the visual features. In the context of this work a domain is a very flexible and all-encompassing concept used to describe characteristics about visual scenes in a high-level manner.

Semantics of images or semantic parsing in the field of CV refers to the recognition and understanding of the relationship between objects of interest other objects and their environment, oftentimes related by a common task. Natural occurrences of objects and corresponding environments are analyzed to transfer this information into a logical-form representation, understandable for machines and artificial intelligence (AI). On a micro level pixel-wise classification of images – semantic segmentation – yields information regarding both foreground objects, commonly referred to as things [2] and background scenery, known as stuff [3] and enables recognition and understanding of the image's content [4]. Following this segmentation and applying ideas from natural language processing (NLP), where understanding the semantics of a word or sentence is a well-researched problem, semantic relations between things and stuff can be formulated. These relationships answer questions such as "What does seeing this object in this environment mean?". In CV, an example image might get translated with the sentence "Two men riding a bicycle in front of a building on a road". Knowledge about the characteristic features of male humans, bicycles, buildings and roads as well as spatial awareness about the scene geometry are necessary to make such statements. Going one step further and looking at the bigger picture yet, other descriptions that provide more detail can be "The time of day is afternoon, the surroundings seem to be the neighborhood suburbs of a city in the northern hemisphere, the season appears to be early-summer". Such broad observations are easily surmised by humans and necessary for complete scene understanding but nevertheless largely unexplored in today's CV research.

The close surroundings of objects of interest feature distinctive semantic context and can give such objects different meanings. An airplane on top of a bed is assumed to be a toy-plane or maybe an illustration in a brochure while an aircraft surrounded by clear sky and clouds is both "real" and in its natural environment or domain – undertaking the flying task. At the smallest level, context can be the surrounding pixel-classes of objects of interest, e.g.

within the bounding-box, but context can also be given across an entire image or even dataset. Incorporating a sense of direction or location of the context results in spatial context. In past works, during inference with conditional random field (CRF) models, contextual information was usually used as shared prior for classification: a boat has a higher probability of appearing above or on the sea than a house, but at the same time the sea has a higher probability of appearing underneath a boat than a house.

In the scope of this work, a domain is a collection of dominant or characteristic classes or higher-level superclasses. The syntactic evolution from semantic context to domains is natural, when considering the focus for the latter lies on some kind of object, person or autonomous agent around which the context is formulated. As such it is closely related to the idea of semantic context. In fact domains can be understood as a result of the analysis of an objects semantic context, placing things into a distinctive domain. The idea of domains is especially of interest when discussing intelligent agents in the real world, be it self-driving cars for advanced driver assistance systems (ADAS) applications or aircraft, drones and unmanned aerial vehicles (UAV) in the aerial domain. An agent during its operation works in a known set of domains. This set can be defined by human experts and is thus exhaustive to describe the expected working environment. These domains often-times contain multiple subdomains and the granularity is an important distinction to make. A new subdomain could be defined if tasks undertaken within are of great importance or the visual semantic context is distinctive enough.

Two example domains are indoor vs. $\mathrm{outdoor}$. Both domains are comprised of distinct features that are very natural for humans to understand. Indoor environments are generally enclosed by walls and a ceiling, feature artificial flooring & lighting as well as a strong human presence. Because of these characteristics, aircraft for example are unlikely to be encountered indoor, at least during regular operation. If aircraft do appear indoor this already gives a strong clue of the task taking place – there might be an exhibition of sorts or the aircraft is undergoing maintenance. This simple example shows the potential understanding gained when domain-awareness is included in image processing algorithms.

The term domain is sometimes used in CV to differentiate between real and synthetic data – data captured in the real world vs. data generated synthetically. Latent representations usually differ (real-world lighting conditions and atmospheric effects are difficult to model) giving rise to the task of domain adaptation which seeks to minimize differences between the data distributions. In an ideal scenario divergence would become zero and the domains would merge into one. In this work domains cannot merge to such an extend, and at least a few distinct domains always remain, even on a very coarse level. Furthermore, domains are also always used in reference to objects of interest and describe the environment these things appear in.

It is well known that cues referring to any kind of semantic relationship stem from the semantic context surrounding objects and this context is therefore a necessity for more complete scene understanding [5]. Although initial works leveraging context cues exist [6, 7, 8], there is still significant unexplored potential in the full application of these context priors. Wang *et al.* [9] describe domains, the natural extension of context, with a distinct feature space $\mathcal{X}$ and marginal probability distribution $P(X)$, where $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \in \mathcal{X}$ are observed data

samples in that domain, such as domain-specific images.[1] Following the analysis of $P(X)$, characteristic statistics can give an understanding of datasets which can in turn be used to guide e.g. data-aggregation strategies for various learning tasks as a first step. Further along the learning pipeline, due to the convolutional kernel in convolutional neural networks (CNN), contextual information is usually learned implicitly regardless of the actual learning task at hand [10, 11]. The semantic-context information is embedded as features in the feature space of the layers and the specific characteristics of these kernel-parameters lead to the prevalent bias/-variance dilemma [12]. Algorithms tuned to perform well in one domain, regardless of learning task, are biased in a sense, and this bias is in conflict with an algorithm learned to perform across multiple domains, featuring higher variance in its learned feature representations. An explicit representation of context in the form of distinct domains might allow intelligent systems to swap between models trained and fine-tuned on specific domains instead of relying on a generalized model. This in turn improves model accuracy on numerous perceptions tasks. This application for parameter-swapping is not part of this thesis, but the preliminary step of predicting domains is studied in detail. The complete procedure could be understood as a novel form of domain adaptation, since common domain adaptation methods usually deal with the discrepancy between real vs. synthetic data.

Keeping the usefulness of semantic context for a variety of problems in CV in mind, this work makes the following contributions: It brings forth both a simple way of extracting semantic context using semantically segmented images, and secondly shows applications of this context in different stages of a learning pipeline. In particular, extraction builds on the idea of label co-occurrences, argues why co-occurrence is not enough, and forms statistical measures for per-class distributions of pixels. Furthermore, it gives label transitions in the Von-Neumann neighborhood (VNN) to give a narrow-range sense of direction. Obtained measures are on an instance, image and dataset level and are used as statistical priors to improve data harvesting procedures in target domains. The application in a learning procedure includes first the formal description of the domain prediction task in a general sense. Experiments in predicting domains, specifically the common environments of commercial airplanes across three public datasets, are conducted. A baseline for this prediction was developed and various models from both unsupervised and supervised machine learning (ML), particularly clustering algorithms and mixture-models for the former and deep CNNs trained from scratch using manually obtained domain-labels for the latter, were employed and evaluated for domain prediction

The structure of this thesis is as follows: First, in chapter 2 a brief introduction of semantic context and its history in NLP is given. The usage of visual semantic context is then described pre- deep-learning and with CNNs respectively. The aerial domain and applicable public datasets and their shortcomings are introduced and the data aggregation procedure described in chapter 3. Afterwards chapter 4 shows the particular methods used for obtaining semantic context and statistics on images featuring airplanes from three datasets (*ADE20K-SceneParsing*, *COCO-Stuff* and *PASCAL-Context*) are presented. The acquired statistics are used to derive the target dataset *SemanticAircraft*. A set of filters for well-tuned aggregation using context priors is formulated. In chapter 5 the task of domain prediction is formally described

[1] The mathematical notation used in this work is described in Appendix A.

and the manual annotation procedure labeling ground truth domains for the image patches is explained. Since no other works regarding the task of domain prediction in aerial scenes exist, an original baseline algorithm is formulated. The developed baseline model as well as applied unsupervised ML models and various CNNs for the task of domain prediction are detailed and results presented and discussed. Following a brief summary, chapter 7 provides an outlook on the possible future work with visual semantic context, including other methods of obtaining context as well as possible learning tasks that likely benefit from semantic contextual knowledge.

2 Related Works

The aspirations to achieve a more complete scene understanding has gained traction over the past few years with semantic segmentation becoming an increasingly more popular learning task for deep learning (DL) models. Following image-wide classification, object detection and semantic segmentation, a clear trend towards more complex representations and a fuller understanding of visual imagery is noticeable [13, 14, 15]. Very recently the task of panoptic vision [16], the combination of semantic and instance segmentation, has emerged.

This chapter provides an overview of existing works dealing with visual semantic context. In particular, it shows existing methods for obtaining an explicit representation of the context in images in some feature space X and the way semantic context priors, both explicit and implicit, are used. In particular, knowing X and the marginal $P(X)$ what methods exist for obtaining the domain $\mathcal{D}$, defined by Wang *et al.* [9] as $\mathcal{D} = \{X, P(X)\}$.

We begin by looking at the early usages of visual semantic context and the way in which ideas were derived from the field of NLP. This is done to gain a deeper understanding of semantic context while working in a well-established field of ML. One can model both individual letters as well as pixels with random variables l and p respectively and see that the same idea of using neighboring letters, i.e. words, sentences and paragraphs, to establish the context around l is the same as using the neighboring pixels of p.

2.1 Semantic Context and Natural Language Processing

The idea of obtaining and using information surrounding objects of interest in the visual domain stems from the same premise to leverage cues from the context of words in a language-processing setting. NLP is a well established learning task with numerous ML and recently DL models developed over the years to process and analyze large amounts of natural language data, predominately written text but also spoken language.

Natural languages are languages that have evolved over time through the usage by humans without a conscious effort to plan words, syllables and sounds by some logic. Nevertheless, Montague has shown in his treatment [17] and subsequent works [18, 19] that any natural language can be treated like a formal language. A similar argument can be made about physical objects and the perception. The shape, size, color, structure and placement of objects in the real world has also evolved naturally, although it stands to reason that humans are consciously aware of these characteristics, more so than language syntax. Parameters of object appearance are therefore partly premeditated and expressing perception data in a formal setting is possible just the same. Both words and images are representations of 3D space, although the first being one-dimensional and the latter two-dimensional. Due to his higher-dimensional

representation in images more information gets preserved but so do many of the common disruptive factors that make semantic parsing of images increasingly difficult. A car looks very different during the day, the night, during sun, rain or snow.

The basic idea of combining visual semantic context with the context of words in text documents is to take the representations of the visual images or image regions $\mathcal{I}$ and those of written language $\mathcal{L}$ and embed both in some feature space with different dimensionality, denoted $\mathcal{F}$. These embedding functions, $f(\mathbf{P}) : \mathcal{I} \rightarrow \mathcal{F}$ and $g(\mathbf{l}) : \mathcal{L} \rightarrow \mathcal{F}$ working on a matrix of pixels $\mathbf{P}$ and vector of letters $\mathbf{l}$ can take different forms and this is where the works mainly differ. Once both representations are present in $\mathcal{F}$, various measures for vector closeness or similarity are typically used.

An example for $g(\cdot)$ is given by Murphy *et al.* [20] who attempt to embed the most important information in text documents, i.e. reduce dimensionality of the text, via non-negative sparse embedding (NNSE), a derivation of singular value decomposition (SVD) but the embeddings can be overcomplete which strengthens sparse decomposition. They formulate the optimization problem of decomposing input matrix $\mathbf{X}$ into $\mathbf{A}$ and $\mathbf{D}$ respectively, subject to a sparse constraint on the rows of $\mathbf{A}$ among st other constraints.

The DeViSE model from Frome *et al.* [21] combines language and visual embedding. They teach a language model, in particular the skip-gram architecture [22] to represent 155.000 terms from Wikipedia documents with a semantic vector each. At the same time, they remove the softmax layer from their visual model pre-trained on ImageNet [23] for object recognition, and replace it with a projection layer for the embedding space. Once in $\mathcal{F}$, they argue that such vector-nearest-neighbor evaluation is a ranking problem and formulate a combination of hinge and dot-product loss. They assess that L_2 loss only aims to bring image and word vectors together but stays agnostic to incorrect word vectors closer to the target image.

As part of zero shot learning (ZSL) for image classification, Norouzi *et al.* [24]'s ConSE model does not learn a direct regression function $f(\cdot)$ but instead they train a classifier to predict image labels in a maximum a posteriori (MAP) fashion. The semantic embedding vector is then deterministically predicted by combining the semantic word embeddings from skip-gram weighted with the classifier output probabilities. Cosine similarity is used in $\mathcal{F}$ to compare the embeddings of images and class labels.

Kiros *et al.* [25] and their follow-up work [26] show end-to-end DL frameworks for caption generation and multi-label classification of images respectively. In both cases a CNN is used as $f(\cdot)$ for feature embedding in a multimodal space. In [25] a long short-term memory (LSTM) model, a networks capable of capturing a larger context extent due to gated memory units, also encodes the image description to the same space. A neural language decoder then recursively generates captions. In [26] and [27] embeddings of features from extracted regions of interest (RoI) are combined in a loss layer with embedded text features. Ren *et al.* [27]'s work utilizes the GloVe model [28] for text labels.

Liu *et al.* [29] use a similar approach of deep visual-semantic embedding for selecting thumbnails of videos following word queries. They use the GloVe model for word embedding and also replace the softmax layer with a projection layer mapping to the embedding space. Pan *et al.* [30] jointly model visual-semantic embedding and translation to words with a LSTM-recurrent

neural network (RNN) jointly learnig video and sentence embedding. Cao *et al.* [31] combine a visual-semantic fusion network (using LSTMs) with a modality hashing network to retrieve images given image or sentence queries. Niu *et al.* [32] provide dense visual-semantic embeddings for specific image-region features and phrases instead of entire images or sentences. Pérez-Arnal *et al.* [33]'s work put forward a concept of visually assessing the distance between semantic concepts or words in the vast lexical database known as WordNet.

The main thing to take away from this overview is the similarity of semantic context in either visual images or written language. Multiple methods exist for obtaining such a context embedding using ML as well as DL methods. Nevertheless, attention has shifted away from taking larger-dimensional features and projecting them to lower-dimensional joint embedding space and instead lies in determining the semantics on a micro-level namely the classification of individual image pixels in the learning task semantic segmentation.

2.2 Semantic Context in Co-occurrence Models

Before CNNs, semantic segmentation was done with either CRFs or large-scale tree models similar to Markov networks [34, 35]. CRFs are a type of statistical discriminative model building on the idea of Markov random fields (MRF). A MRF is an undirected graph $G = (V, E)$ where the vertices V are random variables (often superpixels in CV) and edges E are the dependencies between the vertices/superpixels. When adding additional vertices, one might divide the set of random variables into two sets $\mathbf{x}$ and $\mathbf{y}$ respectively. For a CRF the graph needs to fulfill the Markov property, in the sense that conditioning on $\mathbf{x}$ globally means all variables in $\mathbf{y}$ follow $p(\mathbf{y}_u \mid \mathbf{x}, \mathbf{y}_v) = p(\mathbf{y}_u \mid \mathbf{x}, \mathbf{y}_x)$ where $u \neq v$ and $\mathbf{y}_u$ and $\mathbf{y}_x$ are in the Markov blanket (neighboring nodes in the graph): Using $\mathbf{x}$ as observed variables or evidence, the value for any single label $\mathbf{y}_i$ is only dependent on its neighboring nodes. This can be formulated in a likelihood expression using parameters θ for label transitions and finding optimized θ to yield the highest likelihood for $p(\mathbf{y} \mid \mathbf{x})$ with log-odds. In practice, CRFs are often used to incorporate object co-occurrence statistics. Superpixels or coherent image regions become a node in a CRF and inference allows detection of harder object that co-occur with easily detected objects. Most works focus on pairwise co-occurrence due to computational complexity, although many relationships might require richer representations [1]. Furthermore, the distribution on co-occurrence statistics is often too flat in practice and not peaky enough to make decisive arguments for or against the appearance of objects [36].

One early work is Choi *et al.* [1]'s model and their task of out-of-context estimation. They argue that contextual information is useful for more than simply boosting object recognition frameworks, and formulate the task of out-of-context estimation. A tree model is used to capture the co-occurrence of 107 *SUN* [37] categories, featuring binary variables to represent whether a category is present in the image and latent variables representing scenes, meta-objects or superclasses, using the gist-descriptor [38] as measurements for latent variables. They then continue to construct a support tree (road supporting e.g. cars) relating bounding box location and classes and were able to detect out-of-context objects due to unlikely co-occurrence or uncharacteristic support relationships.

In a similar vein, Fu *et al.* [39] postulate a class label graph computed via an absorbing Markov-chain process (AMP), similar to CRFs except nodes can reach an absorbed state in which their state can no longer change, to model an entire semantic manifold in embedding space, which goes beyond simple cosine distance similarity between embedded vectors. Classes are grouped into superclasses that span distinct semantic manifolds. They argue that the two most common spaces for semantic embedding are either the attribute space with an attribute ontology used to represent class labels or the semantic word vector space. Their graph is formed using clustering methods, in particular K-nearest neighbors [40].

Mottaghi *et al.* [4] introduced the *PASCAL-Context* dataset, providing label images (semantically segmented masks) with 540 categories for *PASCAL-VOC 2010* [41] and go on to develop a new contextual model exploiting global context (presence or absence of a class in the scene) and local context (contextual classes in object vicinity). They establish object roots and formulate random variables for appearance parts (object semantics) and contextual parts (surroundings outside root). The detection problem is viewed as MRF inference, with support vector machines (SVM) learned via loss on intersection over union (IoU). SuperParsing [42] and O2P [43] are used for segmentation, and they finally combine the different models into a single contextual-detection model.

The idea of co-occurrence will be revisited at later points in the work but we will now look at the possibilities that have arisen from using CNNs as very powerful feature extractors capturing context naturally with the 2D convolution kernels.

2.3 Semantic Context and Segmentation with Deep Learning

For a brief introduction into DL and its common terms and ideas, the reader is referred to [44]. The convolution operation $f*g$ combined with learnable weights in the kernel and the practice of stacking multiple such layers on top of another has allowed CNNs to become versatile feature extractors with wide-ranging applications in CV.

A simple application for semantic context is Doersch *et al.* [45]'s work of extracting an image patch and another patch in the Moore neighborhood of the first and predicting the spatial location of the second patch in the neighborhood. This is somewhat related to the generative task of image inpainting and was used by Pathak *et al.* [46], who postulate context encoders as CNNs that predict missing parts of a scene from the parts surroundings, encoding surrounding context and decoding onto missing spaces.

In a similar vein to previous works using CRFs is Wang *et al.* [6]'s multiple-label classification on *NUS-WIDE* [47], *COCO* [2] and *VOC-2007* [48]. They combine a VGG [49] CNN pretrained on ImageNet to embed features with a LSTM-RNN to embed label information in a joint space. Similar to Choi *et al.* [1], they argue that CRFs only capture pairwise label co-occurrences, and even though the graph is extendable it quickly becomes computationally intractable – while recurrent neurons can represent higher-order correlations. The task of multi-label prediction is represented as an ordered prediction path problem, and no Markov property is assumed, in-

stead (greedy) beam-search is used to find probable multi-labels. The labeling order is given by occurrence frequency in the dataset. After predicting a label the RNN shifts its visual attention away from regions with that label, to predict the next label on the ordered path. Unfortunately this only partially deals with the semantic redundancy of separate labels, and having the labeling order decided by the occurrence frequency introduces bias, and it is unclear if this model would perform as well with more uniform label frequencies.

Zhang *et al.* [8] pose the question whether capturing contextual information with a CNN is the same as simply increasing the receptive field size. This is certainly a way to capture semantic context, so perhaps the question should be "How much can one increase the receptive field size and still capture relevant contextual information?". They contribute a context encoding module to produce scaling factors conditioned on the encoded semantics. These scaling factor are then use to highlight class-dependent feature maps, a strategy taken from style-transfer and SE-Net [50]. The highlights get channel-wise multiplied with unencoded dense convolutional featuremaps from the CNN. Dilated convolutional layers are used to create the final semantic segmentation.

Chen *et al.* [7] use a pre-trained GloVe model to get semantic embedding vectors from convolutional feature maps for every category, which are in turn fused with image features at various locations to obtain attentional coefficients and category-related feature vectors. These get related in a graph neural network akin to CRFs. The message propagation is gated to encourage correlated categories (nodes) to talk. Finally, scores for every category are obtained corresponding to the likelihood that category x appears in this image with high scoring categories used as multi-labels for the image. Their ablation study shows that while semantic decoupling is important, the co-occurrence graph is less so. The shortcomings are again unpeaky pairwise co-occurrence can only every give $p(x \mid y)$ and $p(x \mid z)$ but never $p(x \mid y, z)$ or similar higher-order concepts like domains. Also since the normalization is only done over location, categories with fewer instances are naturally correlated less to any other category.

In a similar vein with Zhang *et al.* [8]'s work, Fu *et al.* [51] state that the method with which to effectively capture pixel or region-aware context in an end-to-end training framework is still an open research question. They go on to say that this is a valuable research topic for achieving comprehensive and accurate scene parsing of visual imagery. They use a weighting scheme to include both global and local context to satisfy respective context demands of specific pixels to help with semantic segmentation.

While these works provide a very solid foundation for capturing context in a DL manner, it stands to reason that the semantically segmented output masks are much lower-level in their representation of the context than one might desire. It could be argued that pixel-wise classification as final model output is less representative of the actual content of an image than multi-label classification or the concept of domains. We will therefore take a step back and look at existing works on the concept of domains and how they relate to the definition in this work.

2.4 Domain Adaptation and Prediction for Context Generalization

Following the definitions and notation of [52, 53, 9], described specifically in [54], a domain $\mathcal{D}$ consists of a d-dimensional feature space $\mathcal{X} \subset \mathbb{R}^d$ with marginal probability distribution $P(\mathbf{X})$, task $\mathcal{T}$ defined by label space $\mathcal{Y}$ and conditional probability distribution $P(\mathbf{y} \mid \mathbf{X})$ where $\mathbf{X}$ and $\mathbf{y}$ are random variables or more specifically a sample set $\mathbf{X} = (\mathbf{x}_1, \dots \mathbf{x}_n)$ of $\mathcal{X}$ with corresponding labels $\mathbf{y} = (y_1, \dots y_n)$ from $\mathcal{Y}$. Using feature-label pairs $\{\mathbf{x}_i, y_i\}$ the conditional $P(\mathbf{y} \mid \mathbf{X})$ can in general be learned in a supervised manner with some discriminative objective function $f(\cdot)$. In literature, the problem most commonly deals with two domains, source domain $\mathcal{D}^s$ and source task $\mathcal{T}^s$ as well as target domain $\mathcal{D}^t$ and target task $\mathcal{T}^T$. If the domains and tasks correspond, i.e. $\mathcal{D}^s = \mathcal{D}^t$ and $\mathcal{T}^s = \mathcal{T}^t$, $\mathcal{D}^s$ becomes the training set and $\mathcal{D}^t$ the test set. If the domains are somewhat related this corresponding information can be used to learn $P(\mathbf{y}^t \mid \mathbf{X}^t)$, a process known as transfer learning (TL). As Wang *et al.* [9] explain, one-step adaptation can use discrepancy based (maximum mean discrepancy (MMD) or Wasserstein distance in a reproducing kernel Hilbert space (RKHS)), specifically class, statistic, architectural or geometric criterion. Another way to achieve domain adaptation (DA) is to use adversarial based objectives to encourage domain confusion during reconstructions in an encoder/decoder setup. Most commonly, the source domain consists of synthetic, labeled images (see e.g. [55] for a physics-engine to generate synthetic images in avionics) while unlabeled target images form the target domain. Multi-step DA has intermediate domains that are more related with source or target domains respectively. These intermediate domains can be hand-crafted, instance-based or representation-based. We will begin by taking a look at common domain adaptation works building on this methodology and then formulate a general extension to multiple domains that differs from the idea of intermediate domains.[1]

To begin with, it should be noted that the distinction between supervised and unsupervised methods regarding DA is difficult to establish, many ideas being instead semi-supervised. Wilson *et al.* [57] argue in their survey of unsupervised deep DA that supervised methods oftentimes build on the premise of data in $\mathcal{D}^s$ and $\mathcal{D}^t$ being drawn from the same distribution, an assumption that does not hold in real life (specifically ADAS) applications very often. Pan *et al.* [53] in their survey on TL define DA as a type of transductive TL where the task remains the same, but domain marginal probability distributions differ. They point out that any kind of DA working with feature representations assume said representations are domain invariant. An early example of unsupervised DA building on the premise of minimizing discrepancy in distributions is that of Ganin *et al.* [58]. The objective is to maximize the loss of a domain-classifier that discriminates between source and target domains, bringing source (partially synthetic and labeled) and target (real and unlabeled) images together, making the feature distributions over

[1] Recently the task of predictive domain adaptation (PDA) has emerged, where the aim is to generalize even further, to new, previously unseen target domains. Mancini *et al.* [56] have proposed AdaGraph, a deep architecture leveraging metadata to build a graph, where nodes represent domains and edge-strength models domain similarity. Although promising, this holds less relevance for a problem where the domains are assumed to be known and somewhat exhaustive in capturing the entire environment, as is the case in this work dealing with aerial scenes.

the two domains similar. The formulation of their gradient-reversal layer functions very similar to $H\delta H$-distance. As Ben-D. *et al.* [59] point out, $H\delta H$ distance is often used as a discrepancy metric between two distributions w.r.t. a hypothesis set H. This metric makes the assumption that such a hypothesis both exists and enables well-behaved classification in both domains (i.e. $\mathcal{T}^s = \mathcal{T}^t$). Furthermore, it only deals with binary classification (two domains) in the XOR case, i.e. no image can come from both or neither domains. Building on the concept of confusion, Tzeng *et al.* [60]'s work transfers category information between domains by maximizing domain confusion and forwarding the class-correlations to the target domain.

Bolländer [61] provides a brief review of deep domain adaptation (DDA) techniques common in CV with a focus on applications in the ADAS domain. It shows a distinction similar to the one of unsupervised DA between divergence-based, MMD or correlation alignment DDA, adversarial-based methods using generative adversarial networks (GAN) and reconstruction-based approaches employing deep reconstruction classification networks (DRCN) and cycle-GANs.

For the purpose of making a segmentation network robust across multiple domains, Chen *et al.* [62] propose to treat different cities as distinct domains and go on to learn both class-wise and global domain adaptation adversarially in an unsupervised manner. The concept of domains are treated as a means-to-the-end for boosting segmentation accuracy which is a common approach. Similarly, Sakaridis *et al.* [63] use the idea of guided curriculum model adaptation for improving semantic segmentation of nighttime ADAS images. Curriculum learning describes the process of first learning on easy examples and then gradually increasing the difficulty during training for a specific task. Having captured the same scene at daytime, twilight and night using labeled synthetic stylized and unlabeled real data, models are transferred from daytime to night with twilight as an intermediate domain, using the Dark Zürich dataset. A cross-bilateral filter is used to smooth small camera pose errors at same locations and a cycle-GAN is used for style transfer. The created stylized images contain human annotations of original counterparts but also unrealistic artifacts, whereas the real images have less reliable pseudo-labels but are characterized by artifact-free textures. They then propose invalid regions for highly uncertain pixels, while at the same time argue for the inclusion of these invalid regions in evaluation. They finally include a sense of uncertainty by proposing the uncertainty-aware intersection over union (UIoU) metric which is guaranteed to be larger than IoU under the assumption that predictions on invalid regions are incorporated.

In a similar vein are the works of Zhang *et al.* [64] and their follow-up paper [65]. They argue that the assumption of one discriminator (classifier) existing in both source and target domains, does not hold, arguing that $P(\mathbf{y} \mid \mathbf{X})$ is likely not shared between two domains if the classification boundary is sophisticated which it has to be for segmentation. They learn global label distributions over images and local distributions over landmark superpixels and feed those properties into a segmentation network to boost semantic segmentation performance. The former, global image-level label distribution informs the network how to update the predictions while distributions over superpixels tell the network where to do so, thus guiding it towards the target domain. The occupancy proportion of every category over entire images and superpixels is calculated by counting pixel-class labels and dividing it by the size of the region ($p_t(c)$). The

same calculation is done for network predictions ($\hat{p}_t(c)$) and the cross-entropy between $p_t(c)$ and $\hat{p}_t(c)$ is minimized. To obtain these proportions, for global label distribution, they extract image features using an Inception-ResnetV2 network and combine those features with ground truth label distribution from source images and finally feed this into a logistic regression and nearest neighbors algorithm to forward per image label distribution to target images. For local label distributions they segment each image into 100 superpixels via linear spectral clustering. The dominant label and the features of these superpixels are used to train a multi-class SVM yielding a label distribution. In their follow-up paper [65] they aim to search for more "easy" tasks to learn, that can help guide the hard task (segmentation). They state that these tasks should be "easier to solve" than pixel-wise labeling and can also be written as a function of the pixel-wise labels. They recognize the weakness of using superpixels but state their necessity as anchors to decide the location to penalize the segmentation network output. For example, keeping only the top 60% most confident superpixels, as they do, biases towards "easier" superpixels, while hard superpixels might contain unusual and by extension significantly relevant context features. They shy away from using domain-wide superpixel label distributions for network guidance noting possibly inaccurate estimates as troublesome. In their ablation studies, they show that the inclusion of label distributions across images creates CNNs that perform better on small objects. According to the confusion matrices, accuracy improvements largely come from the fact, that the baseline tends to confuse road and sidewalk, while they clear up this confusion with image-wide label distribution, which shows the strength of having an image-wide context measure.

Finally of note is Sikirić *et al.* [66]'s work very strongly rooted in the ADAS domain. The task is image-wide classification of images captured in various traffic scenes in Croatia. While they do not apply any semantic context representation, their treatment of different traffic scenes is similar to the idea of domains in this work: as a concept to describe the environment for scene parsing. They differentiate between eight scenes encountered during driving in total: highway, road, tunnel, exit, settlement, overpass, booth, traffic. They tackle this classification problem using a SVM with a radial basis function (RBF) kernel, and it performs exceedingly well, most likely due to the fact that training, validation and test sets are virtually indistinguishable. Sequences were captured driving around and every x-th frame is assigned to either the training, validation or test set. This process, combined with the significant class imbalance, contributes to the suspiciously high accuracies.

Extending the idea of source and target domains to the n-dimensional case, we assume to have n domains $\mathcal{D}^i = \{\mathcal{X}^i, P(\mathbf{X}^i)\}$ with target $\mathcal{T}^i = \{\mathcal{Y}^i, P(\mathbf{y}^i \mid \mathbf{X}^i)\}$. Thinking of domains in this general way, one goal is to quantify and improve the coverage of subdomains in datasets instead of transferring from one domain to another. If the data is reflecting the real world more closely, any following learning algorithms are expected to perform better on tasks situated in real world environments. As already hinted in the introduction, domains in this sense feature specific object lists based on semantic context analysis and characteristic distribution per object category: per image, but also in image regions. The representation of the semantic context then, among other factors, leads towards the selection of model type, be it supervised or unsupervised, for domain prediction.

Starting at semantic context and NLP the path of visual context in CV was outlined in this chapter beginning with non-deep models most notably CRFs that capture context dependencies, followed by numerous DL works that deal with context most notably in terms of semantic segmentation, until having arrived at works detailing DDA. While this overview has shown little overlap between research tackling semantic context and domain adaptation, it is in the author's opinion that the latter is a natural extension to the former, especially when using the concept of domains in its natural, applied form as different characteristics of scenes in the real word. This is in line with the idea of traffic scene classification shown by Sikirić *et al.* [66]. We will now move on to inspect one specific domain in particular, the aerial domain, outline the various subdomains and highlight public datasets providing aerial scenes images for later evaluation in the work. The aerial domain was chosen since it is underrepresented with research in CV focusing largely on ADAS or indoor domains.

3 Aerial Domains and Public Aircraft Data

The methods developed in this work are kept as general as possible to allow the application in multiple domains, nevertheless, evaluation is done in one particular domain: aerial scenes. This chapter provides an introduction into the aerial domain, its most common subdomains, outlines important dataset factors and image characteristics and showcases publicly available aerial datasets. We begin by discussing aircraft and the domains they commonly appear in, in the real world.

3.1 Aircraft and Common Aerial Domains

In public data the most commonly featured types of aircraft are commercial, jet-engine propelled airplanes used for passenger and goods-transport by different airlines across the world. When considering the domains airplanes traverse, three distinct domains can be identified quickly:

Apron
: In aviation the area where airplanes are usually parked, loaded or unloaded with goods, boarded by passengers or refueled is generally referred to as apron. It is best known as the area passengers traverse on their way to the airplane. Images captured in this domain are usually cluttered, showing a large variety of (occluded) objects – predominately persons and many types of unusual vehicles not seen in any other domain such as mobile loading ramps, taxing vehicles and moving stairways.

Runway
: The strip of asphalt or concrete used primarily for takeoff and landing of the airplanes is referred to as runway. It is usually directly enclosed by grass or other types of soil, with more vegetation such as bushes and trees appearing to the sides. Neither vehicles nor persons are usually encountered in this domain.

Sky
: Airplanes that are in-flight and have reached a certain altitude are in the domain sky. More often than not sky is a smooth blue or grey background to the airplane, but clouds and time of day can significantly alter its appearance. Furthermore, as was outlined previously, context in CV stems from the surrounding pixels of object instances in the 2D image plane. Since this is a projection, elevation angle of the capturing camera has a significant influence. Images of airplanes from above show field, forests and sea as context, but for observers from the ground the actual context would be sky and clouds. This is an issue that will be explored in more detail in section 3.3.

Table 1 compares common classes in the three major domains using typical avionics terms on the left vs. classes as probable counterparts found in public datasets with oftentimes ambiguous labeling on the right. This discrepancy can only be lowered by capturing more data, but any representation of the real world can never be complete. Additionally, no normed, standardized or internationally adopted class-hierarchy for visual data exists but is instead defined by well-known publications.

Table 1: Comparison of common things and stuff classes in the three main aerial domains between ideal and public data, specifically the datasets *ADE20K-SceneParsing*, *COCO-Stuff* and *PASCAL-Context*. Individual rows group classes in a semantic sense. Labels given in italics denote classes that do not occur in all three datasets. Empty cells on the left side are for classes occurring in data that would not occur in this domain in the real world – empty cells on the right signify things or stuff occurring in the domains in reality, but lacking a descriptive label in the dataset.

	Optimal data	Public data
Apron	Airplane (non-target)	Airplane, *Helicopter*
	Terminal, Jet-bridge	Building, *House*
	Staircase	*Stairs*
	Person	Person
	Luggage items	*Backpack*, *Suitcase*
	Taxi, Transport-bus, Truck	Car, Bus, Truck
	Concrete-floor	*Runway*, *Path*, *Road*
	—"—	*Floor*, *Ground-other*
	Luggage-cart and Loading-conveyor	—
Runway	Runway, white ground-markings	*Runway*, *Path*, *Road*
	—"—	*Floor*, *Ground-other*
	Tree, Bush, Grass	Tree, *Bush*, *Grass*, *Field*
	Runway-lights	—
	—	Airplane, Building, *House*, *Fence*
Sky	Sky, Clouds, Fog	*Sky*, *Sky-other*, *Clouds*, *Fog*, *Sun*
	Contrails	—
	—	*Sea*, *River*, *Lake*
	—	*Land*, Mountain, *Hill*
	—	*Forest*, *Road*

The list of subdomains can never be complete and more domains could always be defined – section 4.3 actually introduces three new domains. For now it suffices to say that a fourth domain, other is also necessary to hold out-of-context airplanes.

To establish an understanding of and feel for the domains, Figure 1 gives example images from Google-Images and aircraft-spotting videos. As such they are not an accurate representation of instances from public datasets, in fact the latter are often times more ambiguous and difficult to assess. While the examples in Figure 1 give an ideal representation of the domains, many images in public datasets are often influenced by a variety of factors which degrade the image quality and lower the amount of extractable information. Some of those relevant factors are outlined in the upcoming section.

3.2 Image Attributes for Aerial Scenes and Autonomous Driving Applications

Visual data in the aerial and ADAS domain is influenced by a number of parameters which make robustness of algorithms operating in the real world difficult. These imperfections are prohibitive to the transition of ML models benchmarked on datasets to real-life applications and this transition is perhaps the biggest challenge in CV yet. In the outlook of this work (chapter 7) the possibility of capturing some of these parameters with the concept of domains will be given, for now they are simply listed as factors to keep-in-mind[1]:

Lighting

Both time of day and weather affect lighting to a significant extent. Strong sunlight and clear skies lead to distinct shadows and heavily contrasted images. In diffuse lighting conditions (covered skies), shadows are indistinguishable or non-existent During civil or nautical twilight, both at dusk or dawn, the sun is very low resulting in long cast shadows and strong visual appearance difference between opposite sides of an object due to the amount of light hitting the surface. Nighttime means very little natural but predominately artificial lighting instead, which generally carries a blue tint and low color temperature.

Atmospheric effects

The most common kind of atmospheric effects are various types of precipitation, such as rain, snow and fog. Rain is troublesome since it can cause reflections on the ground or other surface, particularly if the skies clear shortly after the rainfall. Raindrops stuck to windshields can also obstruct the field of view for the visual sensors recording images. Fog is a very noticeable occurrence during the morning hours and airplanes during take-off, landing or in-flight in the sky can be completely occluded as a consequence. Snow significantly changes the appearance of any surface it covers, hiding the color and texture completely. Another effect comes from air traveling from hotter to colder regions in the surroundings. This stream is observable even to cameras and is not uncommon to occur

[1] Effects due to nature and their extent and frequency are of course dependent on even broader parameters, such as geographic location and current season.

over the tarmac of the runway, especially during hot temperatures where the ground is significantly heated up from oncoming sun rays and much warmer than the surrounding air.

Problematic lighting conditions and atmospheric effects are very common when dealing with images for ADAS applications, but are nonetheless rarely discussed in CV literature, although a variety of weather conditions are dealt with in *WildDash* [75].

Another class of influencing factors stem from the setup of the capturing device or related technical circumstances:

Camera-setup

The camera can be static, which is common on airports with cameras being mounted on the tower or on installations in the field. The mechanics usually allow them to rotate in one or more axis, but the overall setup is largely fixed in place. This is in contrast to dynamic cameras which are mounted on the moving system itself such as cameras on autonomous buses or other vehicles. They are less common in the aerial domain at least referring to the detection of large-scale aircraft such as airplanes. Almost every UAV, such as drones, on the other hand do feature some kind of camera setup, but the cameras capture only the surroundings and not the aircraft itself.

Visual degradation

Describes all kinds of effects that occur during the capture of the images themselves or the processing on the sensor chip. Over/underexposure can be due to incorrect shutter speeds for the particular scene causing light to hit the sensor for too long/short a time and can cause strong distortions in brightness and contrast. Allowing too much light to hit the sensor can cause bloom[2], leading to white circles. This can occur due to a particularly strong light source, prolonged exposure or light reflecting and being focused on certain surfaces. Noise is a common problem which corrupts the signal in complex ways, can take many forms such as Gaussian or salt-and-pepper noise and can be attributed to the physical properties of the electronic circuit which are influenced by the circuit board temperature. Blur, most notably motion blur, occurs when object points move fast in relation to the camera, specifically perpendicular to the optical axis. This fast moving part of the scene is represented by a collection of pixels and skips many pixels from image to image instead of being assigned to pixels close to the original. Other forms of blur can be attributed to distortion from the lenses, such as radial distortion. Another effect is chromatic aberration, which can be attributed to dispersion[3]. This means the lens is not able to focus all color waves in the same point, causing a distortion in the color scheme.

These effects all strongly influence the images and their content and referring to public data for CV tasks, most public datasets in particular tend to suffer from some or multiple shortcomings, due to the uncontrolled nature in which source images were taken.

[2] During bloom the charges in particular transistors overflow and neighboring transistors take up the charge.

[3] Dispersion refers to the phenomena, where light of different wavelengths take different paths after traversing a lens since the refractive index of the lens material is dependent on the wavelength.

(a) Apron: Example images showing the domain apron on an airport. Wide-range captures similar to the four images on the left are very rare in annotated and publicly available dataset, many photos look more like the four on the right (Sources from t.l. to b.r.: modified and taken from [67, 68, 69, 70, 71, 72, 73, 74]).

(b) Runway: Example images showing the domain runway on an airport. Captures from this domain are usually blurry due to the fast lateral movement of the airplane in relation to the camera. Images are often taken from afar due to the general inaccessibility of the area for the public.

(c) Sky: Example images showing the domain sky. Captures of airplanes oftentimes show the aircraft in a pose facing upwards and to the side. The context is lacking structural features and mostly consists of few grayish or blue colors.

Figure 1: Example images not part of any public dataset showing the three most common domains of airplanes during operation: apron, runway and sky. Example ground-truth domain labels according to the labeling scheme outlined in section 5.2 are given in the upper-left corner of the individual images.

Most public visual datasets additionally come with some kind of labeling annotation, since supervised DL methods (CNNs and other types of neural networks) are used so commonly for different learning tasks. The type of annotation is usually dependent on the CV task the dataset was created for and one can follow a trend from coarse, image-wide classification to detection to semantic segmentation when it comes to scene parsing tasks:

Classification

Classically done by assigning one label to an entire image as image-wide classification [76, 77, 78, 79]. In recent times specific object instances or image regions get assigned a class label instead. The basic premise with CNNs is to obtain a feature representation of the images and derive single values for every class from said representations, followed by maximum-likelihood classification.

Detection

Object detection [80, 81] usually means finding object instances from certain classes in images. Common methods have a separate algorithm for yielding region proposals where certain objects might be found. Within those proposals the objectness of the set of pixels is judged. If objectness is high enough, classification of those pixels takes place.

Semantic segmentation

Taking classification to the micro-level, individual pixels can be assigned class-labels, which is the task of semantic segmentation [82, 83, 84]. In practice, for the creation of semantically segmented training data, annotation tools usually assign labels to superpixels instead of truly individual pixels when creating semantic masks.

Panoptic vision

Combining semantic segmentation with instance segmentation (detection and segmentation of each object instance) has led to the task of panoptic segmentation [16], as a richer and more complete scene understanding method.

While CNNs are certainly very capable of capturing context (see section 2.3), and are also used in this work for supervised domain prediction (see section 5.5), semantic masks already provide the basis for rich semantic information retrieval. When considering methods to use context in an explicit representation and enabling domain prediction with this representation, it is natural to use semantic masks as the basis to simply extract the visual context. This is the premise for the data distillation and unsupervised prediction algorithms employed in this work, using a context-vector $\mathbf{c}$ derived from masks directly.

Having established the various aerial domains of airplanes in images, the next section gives an overview of public datasets that provide semantically annotated images featuring aircraft in different domains. At the same time, further drawbacks of working with public data will also be discussed in more detail.

3.3 Public Datasets and Shortcomings of Public Data

Publicly available datasets providing semantically segmented images are fairly numerous (around 10-15 [85]) and a fair number are specific to ADAS (around 8: Apollo Scape [86], Audi A2D2 [87], Berkeley DeepDrive [88], Cityscapes [89], India Driving Dataset [90], KITTI [91], Mapillary [92] and WildDash [75]). While RailSem19 [93] provides the basis for scene understanding of trains and railway scenes, no semantically-segmented and annotated dataset specifically created for aircraft and aerial scenes exists as of yet. The aerial datasets that do exist usually feature images captured from aircraft such as drones flying over street scenes and although a domain prediction of ground-scenes is definitely feasible it is not the focus of this work. These restriction, in combination with other necessities and desirable feature of datasets summarized below, significantly restrict the dataset selection.

Public
: Since the process of capturing and semantically annotating an aerial dataset would be beyond the scope of this work, publically available datasets need to suffice.

Sementically segmented
: As mentioned already, semantic masks enable simple extraction of the context and representations as pixel-based distributions with high accuracy [64]. Another method for obtaining context is using the complex feature embeddings in CNNs either implicitly, as is done in this work, or transforming it into an explicit representation first.

Aircraft instances
: For this work, the focus is put on predicting the domain of aircraft, although application in other domains only requires the exchange of source data (see chapter 7).

Common classes
: When deriving from multiple datasets, the labels and data they represent are required to be somewhat similar. Since no globally accepted class-ontology for CV exists and the variety of objects in the real world is significant, this is a non-trivial requirement.

Scale
: The scale of a dataset is of great importance two-fold. Both images should be sizable, 32×32 CIFAR [94] images are insufficient for context estimation, and the dataset itself should be vast, featuring multiple thousand images. In both senses scale contributes to increased variability of image content.

Additionally, domain labels for the training of the supervised CNN will have to be manually annotated, since no labeling for this CV task exists yet. Since evaluation will take place on an image-quadrant and instance basis, bounding box (BBox) annotations are also required to allow the extraction of instances – although if BBoxes are not already present in the public datasets, they can be generated from the segmentation masks (see section 3.4). Instance segmentation annotations used in the panoptic learning task are not of relevance.

Three public datasets that feature aircraft and also pixel-wise annotation were identified and used: A derivative of the *ADE20K* dataset [95] for scene parsing [96] referred to as *ADE20K-SceneParsing*, an extension to the *MS COCO* [2] annotation for stuff classes [3] denoted as *COCO-Stuff* and the semantic extension to *PASCAL-VOC* [41]: *PASCAL-Context* [97]. For simplicity sake these special derivations will sometimes be referred to as *ADE*, *COCO* and *PASCAL* in the scope of this work.

Another dataset which would fit the requirements is *OpenImages* [98] with 2.7 million segmentation masks on 9 million pictures. Because *OpenImages* was created using Flickr image search without tags, images of aircraft are varied to such a significant extent, which makes roughly every third image irrelevant for most algorithms trained for specific real life applications. Aircraft can include LEGO toys, illustrations, missiles and many images captured from the inside of an aircraft looking outwards, showing nothing more but the wing, although indoor captures are also a problem with *COCO*. Filtering using the tag "airplanes" yields more specific images removing some other airborne objects, such as a hot air balloon with the likeness of Darth-Vader, which is not very likely to appear all that often in the real world. The sheer size of the dataset (750GB + annotations) is also prohibitive to work with and it was consequently decided to forego OpenImages.

Besides the number of images, there are other dataset statistics that allow insight at a glance: The number of labels for example is a good indicator into the exhaustiveness of the class hierarchy of a dataset. Too many labels usually lead to long-tailed datasets [99], while too few labels hint at a possible oversimplification of the real world. Labeling consistency is a related factor, describing whether the same things or stuff from the real world are labeled consistently across the dataset and ideally across multiple datasets with the same label. This is very hard to achieve and we will see that using more classes often times conflicts with labeling consistency. The percentage of void or unlabeled pixels for segmented data reflects the quality of the human annotation procedure. A certain number of void pixels is unavoidable since annotaters themself might not be able to recognize the content, which is not unusual. Furthermore, some void pixels always get created due to the annotation tool itself. Many datasets, including *ADE*, *COCO* and *PASCAL* have void pixels when transitioning from one label to the next in image space. Some datasets have borrowed the taxonomy of separating pixels into things vs. stuff and provide a things-stuff ratio. Datasets usually feature more stuff pixels, which might appear counterproductive in a learning scheme, because one generally wants to learn the features that correlate to things instead. But when talking about holistic scene understanding, these stuff labels are very useful, providing strong priors especially for the estimation of object domains. Various image metric such as minimum, maximum and mean image and instance width and height are also often provided. A dataset purely for airplane instances for examples is expected to have its images be wider than they are high, which can already give a clue to the sort of data augmentation or network architecture required.

Some datasets, more often those for object detection, also provide Boolean flags, or integer values for various characteristics of the instances. *PASCAL-VOC* is such a dataset and it has flags signifying whether an object is either: 1) occluded, 2) truncated and/or 3) difficult. The first two parameters are straight-forward and useful especially for tracking and detection tasks,

while the concept of image difficulty is an ongoing topic of discussion in CV. In general, image difficulty is always related to the specific learning-task at hand. Tudor *et al.* [100] introduced the concept of visual search difficulty using the response time of humans to queries phrased "is object X in the next image?". As it turns out, for humans, this is mostly related to the clutterness of an image but level of clutterness can be seen as just one metric for image-wide difficulty. Lighting conditions and many of the previously mentioned variabilities influence the difficulty as well but are harder to parse. On an instance level, perhaps the most important metric of difficulty is the size. Oversized instances might be truncated while tiny instances yield few features that help distinguishing. This is a prevalent problem when dealing with aircraft since photographs are often taken from afar which makes them look small and bird-like in appearance. In the aerial domains, particularly in sky, the canonical image would show a white commercial plane in their "natural" flying-pose in the sky, with clear blue skies as background. Airplanes of higher visual difficulty might also have unusual appearance such as military aircraft, they might be captured in an unusual pose, or placed in rare environments such as snow-covered runways, on the beach or on lakes. Hard examples would include any of the above factors combined with tiny instance size, strong motion blur, occlusion or airplanes that are exceptionally unusual in structural design, texture or livery.

Finally, camera parameters would also be a nice asset to have access to but are very hard to obtain, particularly in the case of public datasets. Extrinsic parameters, such as setup type, roll, pitch and yaw combined with intrinsic parameters e.g. focal length, field of view and frame rate, would allow very early categorization of images on a global level.

Turning out attention back to the aircraft themselves, in the real world, many different types of aircraft exist. When talking about manned aircraft a distinction can be made following the type of propulsion: jet engines or propellers. Planes can also be distinguished regarding their usage, e.g. for business, commercial, military, utility or leisure activities. For military-purpose aircraft, not only transporter-type planes but also fighters exist, and in a stealth variant as bombers. Another types of aircraft are helicopters, which exist in similar variety and follow the same distinction by usage-type as aircraft. Another recently emerging type of aircraft are UAVs which also come in various different styles. Military and leisure drones are very common, but commercial usages are becoming more prevalent. In public data, the most common variant are jet-propelled commercial airplanes by a large margin, and these are the kind of aircraft which will be aggregated and inspected in this work.

A final point of discussion is the granularity of subdomains used. Looking at aircraft-spotting videos and inspecting the datasets on-hand, at least two other aerial subdomains can be observed:

Taxiway

The area airplanes traverse between apron and runway is commonly referred to as taxiway. It features many of the characteristics of runway, although the airplane is always on the ground and not air-bound. It is further possible for multiple airplanes to appear in one scene, which is highly unlikely for airplanes in runway. The paths and lines on the ground are often curved – intersections of paths are not uncommon. Even though the task is very different for airplanes (slow maneuvering instead of take-off/landing) the

visual similarities to runway are too high and it was decided to not use taxiway as a distinct domain, with instances being assigned either apron or runway depending on the proximity to either domain.

Take-off or landing (TOL)

Aircraft in TOL are air bound either approaching the airport or shortly after take-off, their landing gear is extended. Using this definition a majority of captures from aircraft spotting imagery found on YouTube or similar platforms would be classified as TOL. Since the only difference to sky is an extracted landing gear (which is not even a certainty for all types of aircraft) and the distinguishing factor between runway and TOL is at worst in a very strict sense only a tiny margin of air between landing gear and tarmac, it was also decided to forego this domain. TOL is instead substituted by runway and sky.

While taxiway and TOL are not included as domains, one other, artificial domain is included instead: other. All aircraft that do not fall into any of the domains apron, runway or sky are assigned other. This includes aircraft in indoor environments, images showing the indoor of aircraft, airplanes used for public exhibitions, highly unusual types of aircraft such as military types or toys and finally illustrations or other 2D depictions of aircraft. Another common occurrence are images of airplanes surrounded by a border-frame.Figure 2 shows a selection of such out-of-context, other, airplanes from the datasets *ADE*, *COCO*, *PASCAL*. Examples of unusual aircraft and airplanes from OpenImages can be seen in Figure 3. The proportion of such images is particularly high in OpenImages, due to the liberal harvesting scheme. Figure 4 provides some example images for the two superseded domains taxiway and TOL as well as images that can not be assigned to any of those domains, and are therefore other.

To deeper explore the topic of labeling consistency and labeling incompleteness, in combination with beforementioned specific aerial domains, a comparison of classes actually encountered in the real world vs. those appearing in *ADE*, *COCO* & *VOC* was undertaken. Table 1 provided this comparison for the three major domains apron, runway and sky – Table 2 now gives the same overview for the borderline domains taxiway and TOL at the top and the artificial additional domain other at the bottom.

To summarize, the following limitations of the three datasets, for context work with airplanes in particular, are noticeable:

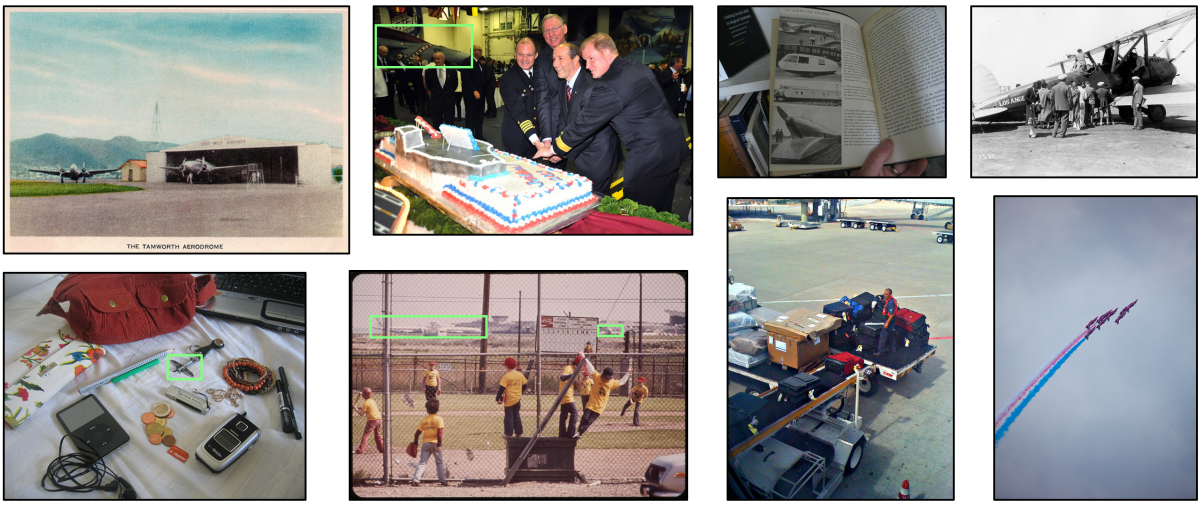
Few datasets

Only few datasets showing aircraft featuring semantically segmented images exist to begin with. Denoted "aerial" datasets usually feature images taken from aircraft such as drones instead, looking downwards with a birds-eye-view.

Incomplete and inconsistent labeling

For example, the labels used to described the paths the airplanes travel going from apron to runway are used interchangeably, making the tarmac in apron, taxiway and runway semantically indistinguishable. Object part annotations are very rare, meaning the landing gear is never annotated which could enable the inclusion of TOL as a distinct domain. As Tables 1 and 2 show, many classes present in one dataset do not exist in the others.

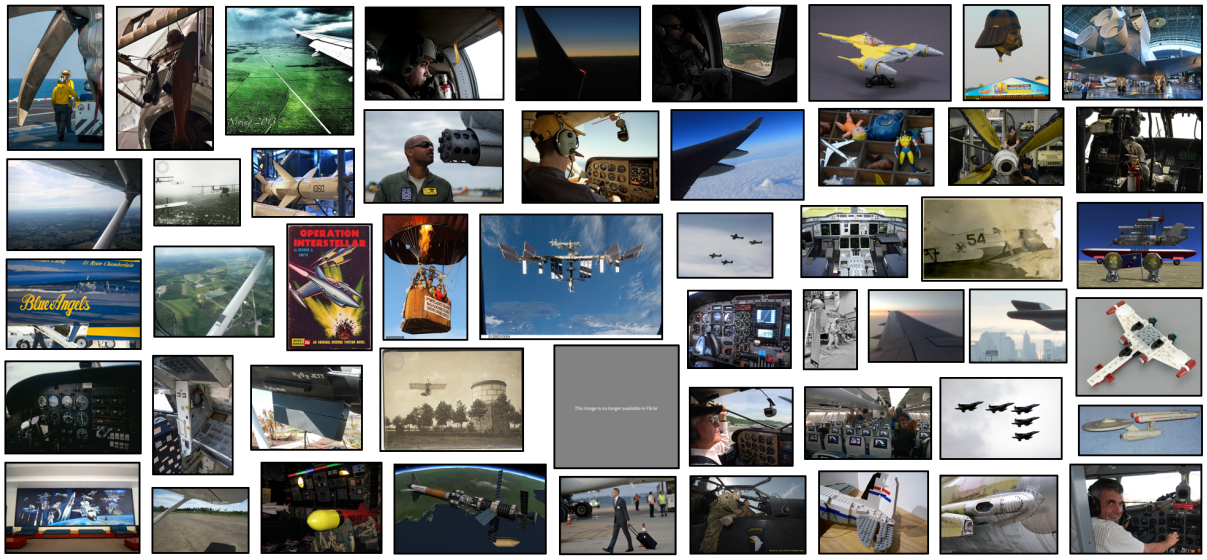
(a) Examples showing images from *ADE* with the tag airfield. Leisure two-seat airplanes often appear in environments similar to the ones shown here.



(b) Examples of difficult or out-of-context airplanes from *COCO*. The second image from the left in the top row barely shows the side of an aircraft in the background covered by groups of people. The two leftmost images in the bottom row show particularly hard examples: the bottom left shows a tiny metal airplane as keychain – in the right image, airplanes are barely distinguishable in the far background behind the fence. The airplanes were outlined for better visibility.

(c) Example images of airplanes from the *PASCAL* dataset. The aircraft in the t.r. and b.l. corners are not only highly unusual, the point-of-view also renders the images exceedingly similar and redundant.

Figure 2: This figure shows the limitations of existing datasets that include aerial scenes, by highlighting images which are so unusual or devoid of significant information, making them prohibitive in a learning scheme. This is one of the reasons exercising data distillation of public datasets can have significant benefits for the learning pipeline, not only in the aerial domain.



(a) Examples of aircraft from OpenImages. While variability in images is desirable for generalization of models, most of these images are so varied they carry little significant information for a learning procedure.

(b) Restricting the search from aircraft to airplanes yields better results, although various illustrations of airplanes and largely truncated instances are still very common.

Figure 3: OpenImages is a vast dataset aggregated freely from Flicker, meaning many images (at least aircraft and airplane images) are hugely out-of-context and irrelevant for most learning procedures. A significant number of illustrations or drawings of airplanes is also present.

(a) Example images showing the unused domain taxiway on an airport (Sources: top row from l. to r.: modified and taken from [101, 102, 103]).

(b) Example images showing airplanes during TOL. For detailed information explaining the separation of similar airplane-images into either runway or sky during the annotation procedure, see section 5.2.

(c) Images from *COCO* depicting airplanes significantly out-of-context. Indoor airplanes, illustrations or seaplanes are fairly common.

Figure 4: The first two subfigures show two other domains aircraft traverse, taxiway and TOL. These images are idealized examples and do not appear as such in any of the used datasets. At the bottom out-of-context images from *COCO* are shown. An example ground-truth domain label is shown in the upper left corner of the individual images, showing the split of taxiway and TOL images to apron/runway and runway/sky respectively.

Table 2: Comparison of common classes appearing in the two transition domains taxiway and TOL, as well as other. The distinct lack of classes representing objects from taxiway and TOL in datasets prohibits learning a proper distinction, while the vast amount of classes appearing out-of-context reflect the necessity of the other domain. Classnames in italics reference categories exlusive to some of the three datasets, not occurring in all three.

	Optimal data	Public data
Taxiway	Airplane(non-target)	Airplane
	Taxiway	*Runway*, *Path*, *Road*
	Taxiway intersections	*Floor*, *Ground-other*
	Grass, Dirt	Tree, *Bush*, *Grass*, *Field*
	Yellow direction signs	—
	—	Building
	—	Car, Bus, Truck
TOL	Runway	*Runway*, *Path*, *Road*
	—"—	*Floor*, *Ground-other*
	Sky, Clouds, Fog	*Sky*, *Sky-other*, *Clouds*, *Fog*, *Sun*
	Tree, Grass	Tree, *Bush*, *Grass*, *Field*
	Runway-lights, Extended-landing-gear	—
	—	Building, *Fence*
	—	Car, Truck
	—	*Mountain*, *Hill*
Other	Domain does not exist	*Ceiling*, *Floor*, *Walls*
	—	*Lake*, *River*, *Sea*
	—	Crowd of people - not annotated
	—	*Poster*, *Banner*, *Magazine*, *Photograph*
	—	*Chair*, *Window*
	—	*Museum*, *Exhibition*
	—	etc.

All of these factors are motivations for creating superclasses: aggregating and merging individual classes (see section 4.3).

Void transitions
Pixels on the outskirts of objects and background classes are often void, which comes from the usage of imprecise annotation tools. This is especially prevalent in *PASCAL-Context*.

Tiny/huge instances
Airplanes in sky are often seen from afar and are therefore very small, while instances on the apron are oftentimes larger than the image and thus truncated. Tiny airplanes in the sky further resemble birds.

Aircraft variability
While many different types of aircraft exist, more often than not the only classes that appear are airplane and sometimes helicopter. Although this is only a minor drawback, since the context of airplanes are of more interest in this work anyway.

Domain variability
A significant amount of images show airplanes in sky in front of clear blue skies, leading to domain imbalance.

Out-of-context
Many aircraft instances are out-of-context, especially in *COCO*, appearing in museums, as toys, on banners, photos, publicity or as attractions during an exhibition. Seaplanes and many small private planes on fields or beaches are also a common occurrence. These out-of-context airplanes necessitated the creation of the other domain.

Keeping these limitations in mind for future reference, the usage of public datasets is nevertheless significantly less labor intensive than own data collection and annotation, especially when semantic masks are required. The method with which these datasets were aggregated and some simple preliminary dataset-statistics will now be shown.

3.4 Preliminary Aggregation of Airplane Images

All images with airplanes from *ADE20K-SceneParsing*, *COCO-Stuff* and *PASCAL-Context* were aggregated. *ADE* features 317 labels for related object parts. The label definition is ambitious and in some cases exhaustive – nevertheless the amount of images showing airplanes is small. In the SceneParsing subset only 150 labels were used and although *ADE* includes the classes airfield and runway, the airplanes in airfield are largely out-of-context (as Figure 2 has shown) and runway does unfortunately not exist in either *COCO* or *PASCAL*. Runway being a specific class would be desirable for later context estimations and domain predictions but *COCO* or *PASCAL* use other path-like classes listed in Table 1 instead. *COCO-Stuff* extends the things classes with roughly 90 classes for stuff, with many variations for certain stuff textures. *PASCAL-Context*(59), the subset used in this work, only features 59 of the entire 456

classes. This is the same subset used by the authors for their evaluations – roughly 13% of *PASCAL-Context*(59) pixels are void to begin with [4].

Using the original semantic masks, mapped masks can get created holding a custom defined class label at every pixel, although for this first aggregation step the label values were left unchanged. It is further possible to create custom RGB masks for ease of visualization. For aggregation every image featuring at least one $\mathrm{airplane}$ pixel was aggregated.

If the dataset provides BBox annotations, those are used for extraction of instances. If it does not, bounding boxes are put around blobs of same pixels. The algorithm for this simply extends a rectangle encompassing 1 pixel at the start and iteratively includes all other pixels of the same target class touching any already included pixels. This procedure has one downside, namely separate airplane instances that overlap in the semantic mask due to occlusion are treated as one instance. This is the case for the aircraft from *ADE* and *PASCAL* but it is a very rare occurrence.

Afterwards, BBoxes were also enlarged percent-wise, by ten, twenty, thirty and forty percent individually. The algorithm for this works as follows: the width of the instance w gets multiplied by the set percentage p: $x = w \times p$. Value x finally gets halved and this half substracted from x_{min} and added to x_{max} (the range of the BBox) resulting in e.g. twenty percent increased width for $p = 0.2$. The same procedure gets applied to the height. If the new coordinates of the BBox would exceed the image borders, the increase is stopped on that side and not appended to the opposite end, i.e. instances always get enlarged around the BBox center.

Every dataset was inspected more closely following aggregation and the following observations could be made:

ADE20K-SceneParsing

ADE only features 146 images with airplanes. 33 of the 150 classes are of interest for aerial applications, although some of them describe very similar concepts. It only features airplanes and no other aircraft. The class runway is unique to this dataset as well as fence and field. Images on average are around 600×600 in size. Two pairs of duplicate images exist, one pair being a true duplicate and in the other pair the "duplicate" is the same image except surrounded by a white border. One of the duplicates in both pairs get removed respectively: during aggregation true duplicates get detected and removed automatically, using image hash values. The border-padded image is manually discarded.

COCO-Stuff

The largest of the three datasets in terms of number of $\mathrm{airplane}$ images with 3079 images featuring airplanes. *COCO-Stuff* has 171 classes in total, 41 of which are applicable, with many of them describing similar objects. Particularly for stuff classes, six labels exist to describe various types of walls, and a few for ground and sky. Class names using the "other" suffix are also common, e.g. $\mathrm{ground\text{-}other}$, $\mathrm{floor\text{-}other}$ or $\mathrm{wall\text{-}other}$ for pixels that annotaters were unsure of. The only type of featured aircraft are airplanes. Images are around 640×480 in size. Besides a significant portion of out-of-context airplanes, *COCO* also features some synthetic images, in the sense of screenshots from computer programs or digital drawings.

PASCAL-Context

The total number of images with airplanes in this dataset is 597. It also features a handful helicopter instances but since this class is unique to *PASCAL* they were not used. It has 456 classes in total of which around 30 are applicable. Here, many variations for building and soil exist. The average image size is around 470×386. *PASCAL-Context* only includes *PASCAL-VOC* images up to *VOC2010*, so any images from *VOC2011* featuring airplanes have no semantic annotation.

As output, all images and masks from the three datasets showing airplanes are obtained. Also, a file for every dataset gets created listing all instances by image name and BBox coordinates, specifically upper left and lower right corners of the BBox. This concludes the preliminary aggregation of airplanes from *ADE*, *COCO* and *PASCAL*. We will next take a look at the process of obtaining semantic context from these images and an application of context for improved data harvesting.

4 Semantic Context Extraction and Exploitation for Data Distillation

Having aggregated all airplanes from the three datasets *ADE20K-SceneParsing*, *COCO-Stuff* and *PASCAL-Context* in the form of color images, semantic masks and BBox annotations, the next step was to obtain the semantic context using this data. This chapter first outlines the method used for obtaining semantic context, similar to the concept of label occurrence frequency presented by Zhang *et al.* [64], followed by the definition of the *SemanticAircraft* dataset deriving from *ADE*, *COCO* and *PASCAL* using context priors to guide the class-mapping process. After filtering of images using context-priors the final context statistics on *SemanticAircraft* are presented.

4.1 Obtaining Semantic Context and Label-Neighborhood Measures

When working with pixel-wise labeled data other statistics in addition to those outlined in section 3.4 can be formulated:

Label co-ocurrence

The distribution assessing the frequency that any pairs of labels x and y from the set of labels l appear in the same image or instance, or co-occurrence of labels x and y, has been shown to be not peaky enough as one would like [36]. Nevertheless, it does provide some preliminary information into the visual context, and can point out classes that possibly describe similar things or stuff, which in turn helps the creation of superclasses.

Label frequency

Counting the number of pixels per class and in certain image regions yields the occurrence frequency. This simple measure provides answers in the form of: In this image/instance 40% percent of pixels are sky, 20% sea and so forth and works akin to the concept of occurrence frequency from [64].

Label location

When placing a grid on images, and counting the number of pixels per class in each grid, a measure can be obtained similar to occurrence frequency although in a reversed sense, instead stating pixels of sky occur to 40% in the upper left quadrant of images, 15% in the lower right and so forth. For both label frequency and location photographer bias has a significant impact, although more so for label location, since frequency generally uses a much coarser (if any) grid on images to formulate statistics. Oftentimes the focus is put

on specific object instances, so objects occur most commonly in the center of an image. For things such as airplanes or birds, which one would assume to be at the top of the image when talking about scenery photography, bias is especially noticeable oftentimes appearing in the center or even bottom center. At this point it should also be noted that in an infinitely large dataset, the assumption of vertical symmetry holds, meaning objects on the left and right side of the vertical symmetry axis appear equally often. The same can not be said for horizontal symmetry, since the land-surface with its various objects can be distinctly separated from the sky and horizon above.

Label-neighborhood

Measuring the frequency of certain label transitions in a directional sense allows the formulation of neighborhood (or label-transition) statistics. Normally this is given using a finite sets of bins, similar to label location, and most commonly in the VNN or Moore neighborhood. Label-neighborhood gives an idea how may road pixels are directly below $\mathrm{airplane}$ pixels for example. This transitional information highlights very broadly how our world is built and can be seen as a generalization of the concept of support-relationships shown by Choi *et al.* [1].

To summarize these measures using the example classes of sky and road: label co-occurrence quantifies the co-occurrence of sky and road in the same image (patch), label frequency measures how many pixels of sky and road appear in that patch, label location shows the distribution of occurrences of sky and road over different patches, while label-neighborhood finally measures the direct label transition from sky to road and vice versa in some neighborhood. Since co-occurrence is known to yield flat distributions it was decided to forego this measure. Frequency and location provide similar knowledge although the latter is more heavily biased, so label frequency is the measure of interest instead. Finally label-neighborhood was obtained as well, although its applications have proven limited and it was not used for later domain prediction.

The semantic context module described in algorithm 1 extracts label frequency for a set $\mathcal{M}$ of masks $\mathbf{m}$. Besides the masks, the list of classes $\mathbf{x} \in \mathbb{R}^{u \times 1}$ is required (u is 150 for *ADE*, 171 for *COCO* and 59 for *PASCAL*). The pixel values for classes void and the target class, $\mathrm{airplane}$ in this instance, are of particular interest. As a first step, to deal with beforementioned void pixels at label transitions, the boundaries of the target instance are expanded by five pixels in every direction, i.e. the instance gets dilated by 1 for five times and this becomes the separate, target mask.

Then, as operations on masks are significantly faster than pixel-loops, for every class in $\mathbf{x}$ the number of pixels in a certain patch of $\mathbf{m}$ is obtained and normalized with the total number of pixels in that patch. For void pixels, they can either be ignored entirely or included and ignored only if they occur in the dilated instance. As a result, a number of context distribution vectors $\mathbf{c}$ are obtained. The context across an image is denoted $\mathbf{c}_i$, for every quadrant in that image $\mathbf{c}_{i,q-I...IV}$. For the area in the BBox around the target $\mathbf{c}_t$ is used, and context on quadrants in that instance is denoted $\mathbf{c}_{t,q-I...IV}$. All of these vectors are also summed up and averaged across the entire dataset of images, resulting in vectors: $\mathcal{C}_{d,i}$, $\mathcal{C}_{d,i,q-I...IV}$, $\mathcal{C}_{d,t}$ and $\mathcal{C}_{d,t,q-I...IV}$. So vector $\mathcal{C}_{d,i,q-II}$ for example gives the context in dataset d across the second

image quadrants. Quadrant-I is the top-right image quadrant counting counter-clockwise.

Algorithm 1 Obtaining semantic context

Requisites: A single mask $\mathbf{m} : \omega \rightarrow \mathbb{R}^1$, holding a class-label for every pixel, or a set of masks $\mathcal{M}$, with $\omega = \{1, \dots H\} \times \{1, \dots W\}$ being the set of pixels. A list of class labels $\mathbf{x} \in \mathbb{R}^{u \times 1}$, in particular the value of label t for the target class and label v for void pixels.

function GETCONTEXT($\mathcal{M}, \mathbf{x}$)
  for $\mathbf{m} \in \mathcal{M}$ **do**
    $\mathbf{m_{ext}} \leftarrow \mathrm{dilate}(\mathbf{m}, t, 1, 5)$ ▷ Dilate the target outline five times by one pixel
    for x in $\mathbf{x}$ **do**
      if $x \neq t, v$ **then** ▷ By default, target and void pixels are ignored
        $\mathbf{c}_{\mathbf{m},x} \leftarrow \sum_{i=0,j=0}^{i=W,j=H} (\mathbf{m_{ext}}_{i,j} == x)$
  $\forall x \in \mathbf{x} : \mathbf{c} \leftarrow 100 \times \frac{\mathbf{c}_{\mathbf{m},x}}{\sum(\mathbf{c_m})}$
  return $\mathbf{c}$ ▷ c : $(u-2)$-dimensional vector with percentage pixel-count for every class x

Entries in $\mathbf{c}$ are also sorted by value i.e. the most common context class is first followed by decreasingly frequent classes. The maximum number of recorded classes can be limited to deal with long-tailedness, which is useful for datasets with a large amount of classes when the majority of them occur only in small amounts.

In a second algorithm the direct label transition as label-neighborhood was obtained in the VNN of $\mathrm{airplane}$ pixels. Once again, masks $\mathbf{m}$ and list of labels $\mathbf{x}$ are required. While algorithm 1 works with masks instead of nested for-loops, algorithm 2 instead casts "beams" from the image borders into the image until a target pixel is found. Once found, up to five void pixels are stepped over backwards of the beam, and the next non-target pixel is noted as label transition in that direction. This procedure is only accurate for convex shapes, while in-between areas in concave surfaces are ignored.[1] Furthermore, all pixels of the target are treated as one instance so images with multiple instances lead to slightly unintuitive results. Instance segmentation annotation would alleviate this last shortcoming. Label-neighborhood is obtained on an image basis in the VNN around the target and averaged across an entire dataset. The average runtime per 640×420 image is 0.7 seconds for neighborhood estimation and less than 20 milliseconds for label frequency. This module allows the calculation of context and label-neighborhood in a general and intuitive way and it was applied to the datasets *ADE*, *COCO* and *PASCAL* to obtain first statistical context measures.

[1] One other way of obtaining neighborhood would be to dilate concave regions until the convex hull is obtained and then take pixels closest on the perpendicular to the tangent of the hull in every point.

Algorithm 2 Obtaining neighborhood label transition

Requisites: A single mask $\mathbf{m} : \omega \to \mathbb{R}^1$, holding a class-label for every pixel, or a set of masks $\mathcal{M}$, with $\omega = \{1, \dots H\} \times \{1, \dots W\}$ being the set of pixels. A list of class labels $\mathbf{x} \in \mathbb{R}^{u \times 1}$, in particular the value of label t for the target class and label v for void pixels.

function GETNEIGHBORHOOD($\mathcal{M}, \mathbf{x}$)
  for $\mathbf{m} \in \mathcal{M}$ **do**
    for $(i, j) \leftarrow (0, 0)$ to (W, H) **do**
      if $\mathbf{m}_{i,j} == t$ **then**
        for $d \leftarrow 0$ to 3 **do** ▷ For all four directions, create vectors p
          if $d == 0$ **then**
            $\mathbf{p} \leftarrow (-1, 0)$
          else if $d == 1$ **then**
            $\mathbf{p} \leftarrow (0, -1)$
          else if $d == 2$ **then**
            $\mathbf{p} \leftarrow (1, 0)$
          else if $d == 3$ **then**
            $\mathbf{p} \leftarrow (0, 1)$
          $x \leftarrow 0$
          while $(\mathbf{m}_{((i,j)+p)} == v \text{ or } t)$ and $x \leq 5$ **do** ▷ Step over void pixels around t
            $\mathbf{p} \leftarrow \mathbf{p} \times x$
            $x \leftarrow x + 1$
          $c \leftarrow \mathbf{m}_{((i,j)+\mathbf{p})}$
          if $c \neq t, v$ **then**
            $\mathbf{n}_{\mathbf{p},c} \leftarrow \mathbf{n}_{\mathbf{p},c} + 1$ ▷ $\mathbf{n}_{\mathbf{p},c}$: class c neighboring pixels in direction p
    $\mathbf{N} \leftarrow \mathbf{N} + \mathbf{n}$
  $\forall d \in [0, 3] : \mathbf{N}_d \leftarrow 100 \times \frac{\mathbf{n}_{\mathbf{m},d}}{\sum(\mathbf{n}_{\mathbf{m},d})}$
  return $\mathbf{N}$

4.2 Context Statistics for Airplanes from *ADE*, *COCO* and *PASCAL*

The statistics shown in this section generally serve two purposes. First, they help give a general understanding and semantic "feeling" for the dataset. This helps to understand the scenes the airplanes appear in across the datasets without having to look at all the images individually. The second point of the shown statistics is the influence on the decision-making process of designing the target dataset, specifically which classes to derive from the source datasets. This process is described in more detail in the next section.

For each of the three source datasets (*ADE*, *COCO* and *PASCAL*) the context is shown in Tables 3, 4 and 5 respectively. The first column shows the dataset-wide context across whole

images $\mathbf{c}_{d,i}$.[2] Using the previously stated infinite-sized dataset symmetry assumption, contexts for quadrants I/III and II/IV become identical: $\mathbf{c}_{d,i,q-I} \equiv \mathbf{c}_{d,i,q-III}, \mathbf{c}_{d,i,q-II} \equiv \mathbf{c}_{d,i,q-IV}$. Columns two and three show $\mathbf{c}_{d,i,q-I}$ and $\mathbf{c}_{d,i,q-III}$ respectively. Lastly, column four gives the context in the bounding boxes around aircraft $\mathbf{c}_{d,t}$. The last row additionally shows the number of respective image regions without any context. In other words, these image segments contain only $\mathrm{airplane}$ and/or void pixels. For more detailed statistics, in particular the context in enlarged bounding boxes and quadrants of (enlarged) bounding, see Appendix B. Averaged dataset-wide image label-neighborhood around airplanes is also given in Appendix B.

Overall the context statistics meet expectations, with a significant number of pixels labeled as sky. The amount of sky pixels is particularly significant in *PASCAL*. Sky as context in the lower image half is much more common in *COCO* and *PASCAL*, compared to *ADE*. The tables also show some overlapping classes and synonymous class-names across datasets, therefore, in order to create one coherent dataset of semantic aircraft images, another, refined step of aggregation was undertaken.

[2] For the sake of visualization, results are rounded to one digit precision, the %-sign is omitted and only the top twenty classes are shown.

Table 3: Image, image-quadrant and instance-wide semantic context around airplanes in the dataset *ADE20K-SceneParsing*. The fairly high amount of sea context stems from fifteen images showing an aircraft carrier and various fighter planes at sea. Sky context is significantly higher in the top half of images than the bottom. A significant percentage of instances are without context, while only some lower-half image quadrants are context-void.

Images	Quadrant-I	Quadrant-III	Instances
sky: 40.4	sky: 65.8	runway: 37	sky: 30.3
runway: 19.4	building: 7.1	sea: 9.6	building: 16.8
sea: 7.4	sea: 5.9	grass: 8.3	runway: 16.4
building: 6.4	ceiling: 4.4	sky: 7.1	sea: 7.5
grass: 3.8	wall: 4.1	building: 7.0	wall: 4.6
wall: 3.1	runway: 2.8	floor: 6.2	grass: 4.5
floor: 3.1	tree: 2.6	person: 5.6	mountain: 3.2
ceiling: 2.4	windowpane: 1.4	earth: 3.4	tree: 2.2
earth: 2.1	earth: 0.9	road: 2.3	windowpane: 2
tree: 1.9	mountain: 0.9	wall: 1.8	floor: 2.0
person: 1.7	hill: 0.6	field: 1.5	road: 1.8
road: 1.1	grass: 0.6	tree: 1.3	person: 1.3
mountain: 0.9	person: 0.6	truck: 1.0	earth: 1.3
field: 0.9	escalator: 0.4	car: 0.9	field: 0.9
windowpane: 0.9	skyscraper: 0.3	sand: 0.7	ceiling: 0.8
car: 0.6	field: 0.3	seat: 0.7	car: 0.7
skyscraper: 0.5	car: 0.3	fence: 0.7	sand: 0.6
truck: 0.5	truck: 0.2	skyscraper: 0.6	hill: 0.5
hill: 0.5	column: 0.1	mountain: 0.5	signboard: 0.4
sand: 0.4	pole: 0.1	escalator: 0.5	pole: 0.3
0/142 (0%)	0/142 (0%)	4/142 (2.82%)	29/272 (10.66%)

Table 4: Semantic context around airplanes in the dataset *COCO-Stuff*. Sky-like classes such as sky-other, clouds and fog are dominating the context. A much lower percentage of image-patches are void of any context, compared to *ADE20K*.

Images	Quadrant-I	Quadrant-III	Instances
sky-other: 35.4	sky-other: 49.1	sky-other: 20.8	sky-other: 32.6
clouds: 16.0	clouds: 22.2	road: 17.1	clouds: 16.2
road: 8.8	fog: 4.9	clouds: 9.2	building-other: 9.1
grass: 5.1	tree: 3.9	pavement: 9.2	road: 7.2
pavement: 5.0	building-other: 3.7	grass: 9.1	tree: 5.1
tree: 3.5	mountain: 2.2	ground-other: 4.1	grass: 4.1
building-other: 3.4	ceiling-other: 1.5	building-other: 3.0	fog: 3.6
fog: 3.2	road: 1.4	tree: 3.0	pavement: 3.4
mountain: 2.4	grass: 1.3	mountain: 2.5	mountain: 2.5
ground-other: 2.2	person: 1.0	sea: 2.4	ground-other: 1.6
sea: 1.6	pavement: 1.0	person: 1.9	person: 1.3
person: 1.4	wall-other: 0.7	truck: 1.5	floor-other: 1.1
ceiling-other: 0.8	window-other: 0.7	fog: 1.4	sea: 1.0
truck: 0.8	sea: 0.7	floor-other: 1.4	metal: 1.0
floor-other: 0.8	metal: 0.6	dirt: 1.4	hill: 1.0
metal: 0.7	ground-other: 0.6	fence: 1.0	window-other: 1.0
dirt: 0.7	hill: 0.4	sand: 0.9	ceiling-other: 0.8
fence: 0.6	wall-concrete: 0.3	metal: 0.9	fence: 0.7
wall-other: 0.6	truck: 0.3	snow: 0.7	wall-other: 0.7
window-other: 0.6	wall-panel: 0.3	bush: 0.5	truck: 0.6
9/3077 (0.29%)	20/3077 (0.65%)	18/3077 (0.59%)	29/5270 (0.55%)

Table 5: Semantic context around airplanes in the dataset *PASCAL-Context*. As column 2 shows, the upper half of images is heavily dominated by sky. The proportion of context-less image-regions is similarly low to *COCO-Stuff*.

Images	Quadrant-I	Quadrant-III	Instances
sky: 58.0	sky: 77.5	sky: 37.2	sky: 54.0
ground: 12.0	building: 7.4	ground: 23.0	building: 10.6
grass: 7.4	tree: 4.4	grass: 13.5	ground: 10.4
building: 6.1	mountain: 2.6	road: 7.0	grass: 6.3
tree: 4.0	grass: 1.7	building: 5.5	tree: 5.2
road: 3.5	ground: 1.5	tree: 3.6	road: 2.7
mountain: 2.0	ceiling: 1.3	floor: 1.9	mountain: 2.4
floor: 1.0	wall: 1.2	water: 1.5	wall: 2.3
water: 0.9	person: 0.5	mountain: 1.4	floor: 1.4
wall: 0.9	road: 0.5	person: 1.1	person: 0.9
person: 0.8	truck: 0.3	snow: 1.0	ceiling: 0.9
ceiling: 0.8	water: 0.3	fence: 0.9	water: 0.7
snow: 0.6	fence: 0.2	truck: 0.7	truck: 0.6
fence: 0.5	light: 0.2	wall: 0.5	snow: 0.5
truck: 0.5	window: 0.2	car: 0.4	fence: 0.4
car: 0.2	snow: 0.1	boat: 0.4	car: 0.3
boat: 0.2	floor: 0.1	window: 0.2	window: 0.3
window: 0.2	car: 0.04	food: 0.2	light: 0.2
sign: 0.1	sign: 0.02	sign: 0.1	boat: 0.1
food: 0.1	boat: 0.01	chair: 0.1	sign: 0.04
2/582 (0.34%)	5/582 (0.86%)	5/582 (0.86%)	5/811 (0.62%)

4.3 Data Distillation for Aggregation of *SemanticAircraft*

When training a more sophisticated learning framework relying on annotated public datasets, one single pass of data aggregation does not yield the desired data in terms of quantity, redundancy, distribution characteristics, object size and many of the other already discussed parameters. After preliminary aggregation a step of data distillation helps filter out irrelevant or redundant data. This leads to a target dataset which has been appropriately tuned for desired tasks, environment or other requirements.

Following the context analysis, only a subset of the total available classes per dataset appear in images with airplanes and the long-tailedness is very noticeable. Looking for example at the context on *COCO*, multiple classes exist for the concept of ground: road, $\mathrm{pavement}$, $\mathrm{ground\text{-}other}$, $\mathrm{floor\text{-}other}$, dirt and merging those classes allows the higher-level superclass representation to include similar, although slightly different classes from *ADE* and *PASCAL* such as runway, floor, earth, road, field and ground. Additionally, this reduces sparsity of the vectors $\mathbf{c}$ representing context which is a benefit for learning models. Therefore, all three datasets were again aggregated while mapping and merging many classes to superclasses, to form *SemanticAircraft*. Figure 5 shows the semantic hierarchy of *SemanticAircraft* with source and target classes.

Images were aggregated using this mapping scheme and instances further increased by ten to forty percent in size. The context for this dataset was obtained using the semantic context module and can be observed in Tables 6 and 7 for images and instances respectively.

Increasing the instance area means capturing more distant context while small instances provide a narrow field-of-view around the airplane. Following empirical analysis it was decided to proceed with thirty percent increased instances, which struck a good balance between incorporating more distant context elements while at the same time leaving out irrelevant features.

A significant portion of images and instances feature undesirable context traits, namely a high-percentage of void and indoor pixels, signifying out-of-context airplanes. Since semantic context encapsulates a high-level understanding of the scene, it can be used to implement distillation algorithms that work on such a level. These filter can be combined with lower level filters observing object size or aspect ratio for strong data distillation. Semantic context naturally lends itself to the removal of redundancies from the data, or the selection of specific instances/images that only fit a certain semantic characteristic. This can be done by filtering out all patches where the context in specific classes is higher or lower than a certain threshold. Another alternative is to filter out the top x-percent of patches via quantiles. The algorithm applied in this work follows the latter methodology although in a way that ends up filtering out (almost) all patches with any indoor pixels.

In particular, using the set $\mathcal{C}$ of context vectors $\mathbf{c}$ for any patch ($\mathcal{C}_i$ for instances and $\mathcal{C}_{q-I\ldots IV}$ for quadrants), sort $\mathcal{C}$ using the values of the context for any desirable class that is to be filtered. Then, obtain the quantile value q_p at the threshold p and remove all patches with context in that class above q_p, while using the mean of the quantile-values for quadrants-I/II and III/IV respectively (large-scale symmetry assumption). The filter algorithm can be observed in algorithm 3.

Table 6: Context across images and image-quadrants of *SemanticAircraft* pre-filtering. Sky, $\mathrm{pavement}$ and $\mathrm{building}$ are the most common context elements. Void labels were included in this estimation and the difference in void pixels between upper and lower image halves are significant. This can be explained due to nature of scenery images naturally being more cluttered in the bottom half, with buildings, pavement variants and plants appearing in close proximity leading to an increased number of void pixels, either due to the uncertainty during annotation or the numerous labeling transitions carrying broad void-pixelated edges. Since void pixels were included in this calculation, the last row shows the number of image-regions only holding $\mathrm{airplane}$ pixels, in other words heavily truncated airplanes.

Images	Quadrant-I	Quadrant-II	Quadrant-III	Quadrant-IV
sky: 53.2	sky: 74.5	sky: 74.7	sky: 30.7	sky: 30.7
pavement: 15.9	building: 4.5	building: 4.3	pavement: 30.2	pavement: 29.5
soil: 6.9	plant: 4.0	plant: 4.0	soil: 12.4	soil: 12.4
void: 5.4	void: 3.9	void: 3.7	void: 6.7	void: 7.0
building: 4.2	indoor: 3.7	indoor: 3.7	building: 3.9	building: 4.1
plant: 3.8	pavement: 2.6	pavement: 3.0	plant: 3.6	plant: 3.6
indoor: 2.9	elevation: 2.5	elevation: 2.6	waterbody: 2.8	waterbody: 2.9
elevation: 2.5	soil: 1.6	soil: 1.6	elevation: 2.5	elevation: 2.5
waterbody: 1.8	waterbody: 0.8	waterbody: 0.8	indoor: 2.1	indoor: 2.0
object: 1.3	object: 0.8	person: 0.6	object: 1.9	object: 2.0
person: 1.2	person: 0.7	object: 0.6	vehicle: 1.7	person: 1.8
vehicle: 1.0	vehicle: 0.3	vehicle: 0.4	person: 1.6	vehicle: 1.6
0/3801 (0%)	6/3801 (0.16%)	3/3801 (0.08%)	4/3801 (0.11%)	8/3801 (0.21%)

Due to the overwhelming proportion of patches having 100-percent sky context, no filtering using sky quantiles took place.[3] – patches were only filtered using void and indoor statistics. Following the heuristic of filtering all patches showing (almost) any indoor pixels while not wanting to filter patches with no indoor pixels and filtering the same amount of patches per class, the p-values were determined empirically as $\mathbf{p} = (0.93, 0.93)$. The process of determining p-values and corresponding quantiles can be seen in Table 8.

Some observations can be made using the mean between q_{q-I} / q_{q-II} and q_{q-III} / q_{q-IV} respectively. Without taking the mean, after filtering for either indoor or void with $p = 0.85$, 11843 quadrants remain. When using the mean, 11838 quadrants after indoor-filtering and 11850 after void-filtering remain instead. More patches were dropped during the former filtering process, i.e. more patches have indoor context in $[q_{mean}, q_{upper}]$ than $[q_{lower}, q_{mean}]$. This in turn means, that once patches do have indoor context, this context is significantly large, hinting

[3] $q_{0.9}$ for sky was 100-percent, i.e. at least 10-percent of patches had only sky as context.

Table 7: Context across instances in various scale sizes of *SemanticAircraft* – pre-filtering. In the header row, the first number describes the percentage increase size of bounding boxes, while the second number gives instances without any context (only $\mathrm{aircraft}$ pixels) in that scaling, from a total of 6354 instances. The number of context-less instances decreases as the bounding box area increases.

	0% (14)	10% (11)	20% (6)	30% (5)	40% (2)
Sky	50.7	50.5	50.3	50.3	50.3
Pavement	12.7	13.3	13.9	14.3	14.6
Building	9.8	9.5	9.2	8.9	8.6
Soil	5.5	5.6	5.6	5.7	5.8
Plant	5.3	5.1	4.9	4.8	4.8
Void	4.3	4.5	4.6	4.7	4.7
Indoor	3.5	3.5	3.5	3.5	3.5
Elevation	3.3	3.2	3.1	3	3
Object	1.6	1.6	1.6	1.6	1.6
Waterbody	1.4	1.4	1.4	1.4	1.4
Person	1.1	1	1	1	1
Vehicle	0.8	0.8	0.8	0.8	0.8

Algorithm 3 Semantic context filter

Requisites: Set $\mathcal{C}$ of vectors $\mathbf{c}$ holding statistics for every patch to be filtered. Labels $\mathbf{x}$ (in this case void and indoor) that shall get filtered with quantile percentages $\mathbf{p} = (0.93, 0.93)$.

function FILTERCONTEXT($\mathcal{C}, \mathbf{x}, \mathbf{p}$)
  for (x, p) in $(\mathbf{x}, \mathbf{p})$ **do**
    $q_{i,x} \leftarrow \mathrm{quantile}(\mathcal{C}_{(i,x)}, p)$
    $q_{q-I\ldots IV,x} \leftarrow \mathrm{quantile}(\mathcal{C}_{(q-I\ldots IV,x)}, p)$
    $\mathcal{S}_{i,x} \leftarrow \mathcal{C}_{i,x} < q_{i,x}$ ▷ $\mathcal{S}_i$: instances after filtering with context class x
    for $i \leftarrow I$ to IV **do**
      if $i \leq II$ **then**
        $\mathcal{S}_{q-i,x} \leftarrow \mathcal{C}_{i,x} < mean(q_{(q-I,x)}, q_{(q-II,x)})$
      else ▷ $\mathcal{S}_{q-i,x}$: quadrants at location i after filtering for x
        $\mathcal{S}_{q-i,x} \leftarrow \mathcal{C}_{i,x} < mean(q_{(q-III,x)}, q_{(q-IV,x)})$
  return $\mathcal{S}_{indoor}, \mathcal{S}_{\mathrm{void}}$

Table 8: Results of filtering *SemanticAircraft* by context using classes void and indoor. Top row gives the p-values for the quantiles while table entries are the corresponding values q_p. Filtering the top five percent by indoor is too lenient, allowing up to 18-percent of indoor pixels in instances, while setting $p_{\mathrm{indoor}} = 0.85$ patches without any indoor pixels are arbitrarily filtered. Final values $(0.93, 0.93)$ allow barely any indoor pixels while e.g. instances with 16.8-percent of void pixels are still accepted, striking a good balance.

	p_{void}			p_{indoor}		
	0.8	0.85	$\mathbf{0.93}$	0.95	0.85	$\mathbf{0.93}$
Instances	3.26	5.94	$\mathbf{16.82}$	18.08	0.0	$\mathbf{2.61}$
Quadrant-I	0.8	2.78	$\mathbf{16.31}$	15.23	0.0	$\mathbf{0.91}$
Quadrant-II	0.88	2.52	$\mathbf{13.44}$	15.94	0.0	$\mathbf{0.92}$
Quadrant-III	6.56	11.27	$\mathbf{29.25}$	4.25	0.0	$\mathbf{0.31}$
Quadrant-IV	7.14	11.96	$\mathbf{31.85}$	4.46	0.0	$\mathbf{0.86}$

towards the fact that few patches are in highly indoor environments. On the other hand, less patches after void-filtering were dropped, i.e. if there is void in an image patch it is usually not that dominant.

Only the set of patches $\mathcal{S}$, both quadrants and instances, was used which satisfied both constraints, i.e.: $\mathcal{S} = \mathcal{S}_{\mathrm{indoor}} \cap \mathcal{S}_{\mathrm{void}}$. As a last filtering step, some low-level filters were applied. In particular, all instances with width or height shorter than 60 pixels were discarded. Lastly, patches with aspect ratio larger than 6:1 in either direction were also discarded, since this causes strong distortions when resizing images for the CNN. This leaves 3854 instances and 13265 image-quadrants respectively.

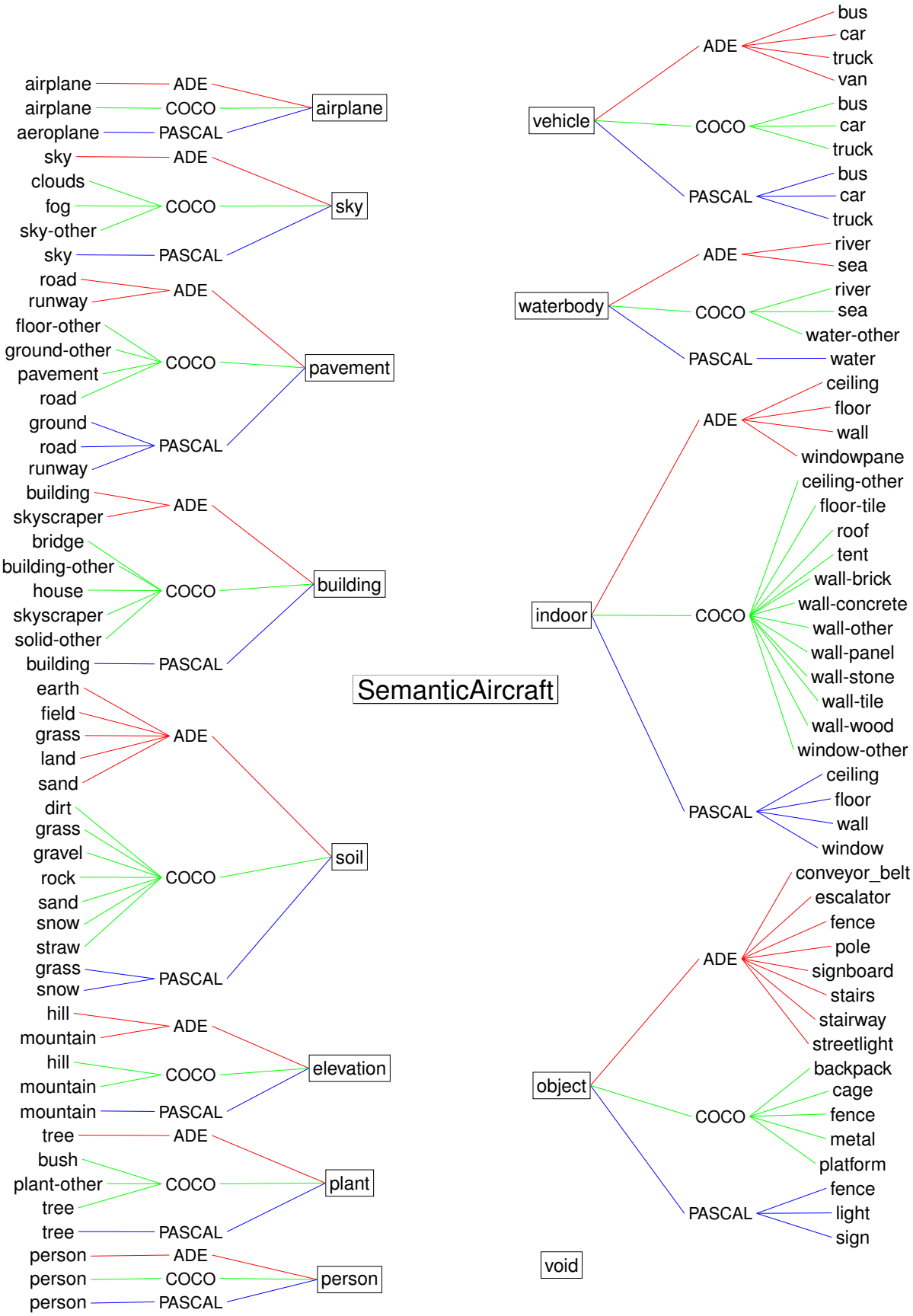


Figure 5: The class hierarchy of *SemanticAircraft*, deriving from *ADE20K-SceneParsing*, *COCO-Stuff* and *PASCAL-Context*. *SemanticAircraft* consists of twelve classes in total: Aircraft, void and ten context superclasses. Indoor consists of many types of surfaces predominately found in indoor environments.

4.4 Context Statistics on *SemanticAircraft*

For this final, distilled collection of images, semantic masks and bounding box annotations, semantic context statistics were obtained using the semantic context module – every individual quadrant and instance is now treated as a single image. Table 9 shows the results for this context extraction.

Table 9: Visual context of *SemanticAircraft*. First column gives the average context across all 3854 instances. Second and third column give the context in quadrants I and III of the instances respectively. Last column shows the context across all 13265 quadrants, treated as individual images. Void pixels were ignored during calculation. No image-regions without any semantic context are included any longer.

Instances	Instance-Q-I	Instance-Q-III	Image-Quadrants
sky: 57.2	sky: 71.6	sky: 38.5	sky: 58.6
pavement: 15.8	building: 8.6	pavement: 30.9	pavement: 17.4
building: 7.5	plant: 6.3	soil: 10.4	soil: 7.8
soil: 6.2	pavement: 4.3	building: 6.1	building: 4.3
plant: 5.1	elevation: 3.7	plant: 3.8	plant: 4.0
elevation: 3.2	soil: 2.8	object: 2.7	elevation: 2.8
object: 1.5	waterbody: 1.2	elevation: 2.4	waterbody: 1.9
waterbody: 1.4	object: 0.8	vehicle: 1.9	object: 1.2
person: 1.2	person: 0.6	person: 1.8	person: 1.1
vehicle: 0.9	vehicle: 0.2	waterbody: 1.5	vehicle: 1.0
indoor: 0.01	indoor: 0.02	indoor: 0.1	indoor: 0.02

Most noticeable is the significant drop of indoor-context due to filtering. For this estimation void-pixels were again ignored, since they do not contribute any semantic information and the unsupervised models building on context vectors $\mathcal{C}$ for domain prediction only use semantically relevant classes. $\mathrm{Airplane}$ (or generally target) pixels are excluded since context is assumed to be independent of the target itself – section 5.2 shows that while this assumption holds for semantic context reasoning itself, in domain prediction the presence or absence of the target can influence the domain category.

The correlation matrices across instances $\widehat{\mathrm{cor}}(\mathcal{I})$ and quadrants $\widehat{\mathrm{cor}}(\mathcal{Q})$ were also obtained, showing the correlation coefficients between individual superclasses as variables. Correlation is dimensionless in comparison to covariance so changes in the scales of the values do not affect it. Additionally, while covariance matrices hold individual variances $\widehat{\mathrm{var}}(\mathbf{x})$ in the diagonal, the correlation of a variable with itself is always 1. A positive value denotes positive relationship, i.e. $\widehat{\mathrm{cov}}(\mathrm{building}, \mathrm{pavement}) = 0.07$ means any certain context amount for $\mathrm{buildings}$ coincides

with similar amounts for $\mathrm{pavement}$. On the other hand, $\widehat{\mathrm{cov}}(\mathrm{sky}, \mathrm{building}) = -0.44$ means high amounts of sky context usually occurs when the amount of $\mathrm{building}$ pixels is small and vice versa. Figure 6 shows the correlation matrices for instances and quadrants at the top and bottom respectively.

Correlation between sky and any other context class is negative, again highlighting the fact that many patches carry exclusively sky as context. The strongest positive correlations are found in instances: $\widehat{\mathrm{cor}}(\mathrm{plant}, \mathrm{soil}) = 0.18$ and $\widehat{\mathrm{cor}}(\mathrm{pavement}, \mathrm{vehicle}) = 0.14$. The pairwise correlation between label frequencies are akin to co-occurrence and in this light the correlation matrices show that the distribution over co-occurrences between classes is very flat, only sky causing significant peaks.

The direct labeling transition as label-neighborhood was additionally calculated. Figures 7 and 8 shows these statistics. For every class in *SemanticAircraft* the figures show the direction, up, left, down or right, pixels from said class appear in relation to $\mathrm{airplane}$ pixels. These statistics state, that for example whenever $\mathrm{pavement}$ pixels directly border $\mathrm{airplane}$ pixels, in 40.6 these pixels appear below the airplane for instances and 32.4 for quadrants. The statistics are spread out more equally for quadrants, which is to be expected, since all image-quadrants were grouped together and e.g. sky pixels appearing in the upper quadrants but below some $\mathrm{airplane}$ are pixels counted as "down" or below the airplane. Without depth or 3D information, some transitions can appear unintuitive, for example the fact, that $\mathrm{building}$-pixels seem to appear above airplanes, when in reality the building are located in some distances behind the airplanes instead. For classes $\mathrm{pavement}$ and soil the results meet expectations – they are consistently placed below airplanes.

While this answers the distribution of pixels per class, the absolute percentages of labeling transitions per-direction are also of interest to determine the classes in closest proximity to airplanes. Tables 10 and 11 show, that the labeling transitions are dominated by the $\mathrm{airplane}$-sky pair, with $\mathrm{pavement}$ or $\mathrm{building}$ as the next most common transition classes. Downward-transitions are the most varied, with different things or stuff appearing below airplanes somewhat frequently. Furthermore, strong differences between Up-Down transitions are noticeable while Left-Right transitions are very similar. While the insights provided into the scene geometry of the images is noteworthy, no further application of the label-neighborhood transition statistics has taken place in this work.

With these context statistics in mind, the aggregation of *SemanticAircraft* is concluded. In the next chapter, the applicability of these context statistics for the prediction of domains of the airplanes will be shown.

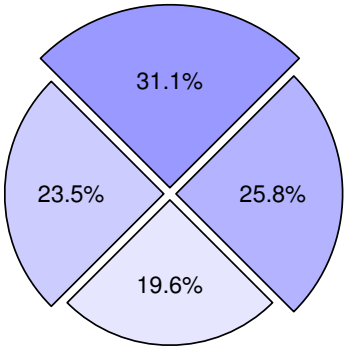
	Sky	Pavement	Building	Soil	Elevation	Plant	Person	Vehicle	Waterbody	Indoor	Object
Sky	1	−.57	−.44	−.4	−.25	−.37	−.13	−.15	−.18	−.08	−.19
Pavement	−.57	1	.07	−.01	−.07	−.03	.03	.14	−.07	.07	−.02
Building	−.44	.07	1	−.06	−.04	−.03	−.05	.05	−.03	.03	.01
Soil	−.4	−.01	−.06	1	−.01	.18	.01	−.04	−.02	−.01	−.01
Elevation	−.25	−.07	−.04	−.01	1	−.03	−.03	−.04	.01	.01	.01
Plant	−.37	−.03	−.03	.18	−.03	1	−.02	−.04	−.03	.03	.01
Person	−.13	.03	−.05	.01	−.03	−.02	1	.01	−.01	−.00	.01
Vehicle	−.15	.14	.05	−.04	−.04	−.04	.01	1	−.03	.02	.03
Waterbody	−.18	−.07	−.03	−.02	.01	−.03	−.01	−.03	1	−.01	.06
Indoor	−.08	.07	.03	−.01	.01	.03	−.00	.02	−.01	1	.04
Object	−.19	−.02	.01	−.01	.00	.01	.01	.03	.06	.04	1

(a) Correlation matrix for *SemanticAircraft* instances. Correlation is low, be it positive or negative, for most context classes other than sky.

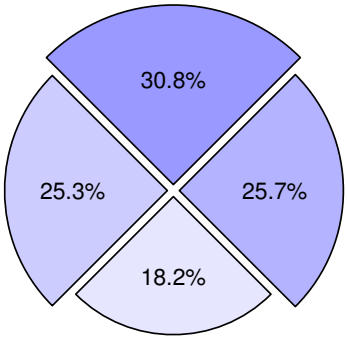
	Sky	Pavement	Building	Soil	Elevation	Plant	Person	Vehicle	Waterbody	Indoor	Object
Sky	1	−.66	−.24	−.44	−.17	−.23	−.17	−.22	−.17	−.03	−.18
Pavement	−.66	1	−.04	−.02	−.09	−.1	.05	.18	−.08	.01	.03
Building	−.24	−.04	1	−.04	−.02	−.01	−.03	−.00	−.00	.03	.00
Soil	−.44	−.02	−.04	1	−.05	.03	.01	−.02	−.02	.01	.03
Elevation	−.17	−.09	−.02	−.05	1	−.02	−.03	−.04	−.01	−.00	−.02
Plant	−.23	−.1	−.01	.03	−.02	1	−.02	−.03	−.01	.03	−.00
Person	−.17	.05	−.03	.01	−.03	−.02	1	.02	−.00	−.00	.03
Vehicle	−.22	.18	−.00	−.02	−.04	−.03	.02	1	−.03	−.00	.04
Waterbody	−.17	−.08	−.00	−.02	−.01	−.01	−.00	−.03	1	−.00	−.00
Indoor	−.03	.01	.03	.01	−.00	.03	−.00	−.00	−.00	1	.02
Object	−.02	.03	.00	.03	−.02	−.00	.03	.04	−.00	.02	1

(b) Correlation matrix for quadrants of *SemanticAircraft*. Correlations between context classes are similar to those observed on instances.

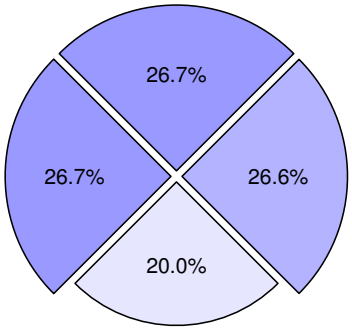
Figure 6: Correlation matrices of the instances and quadrants of *SemanticAircraft*, giving insight into the semantic content of images similar to the concept of label co-occurrence. Notable is the change of correlation between building and pavement from instances to quadrants, possibly alluding to the fact that buildings were truncated in quadrants.



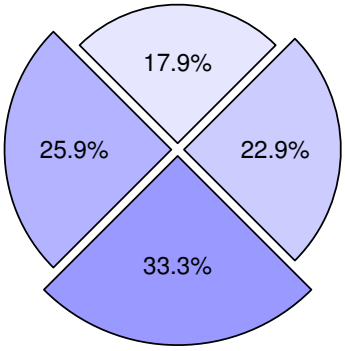
(a) Building



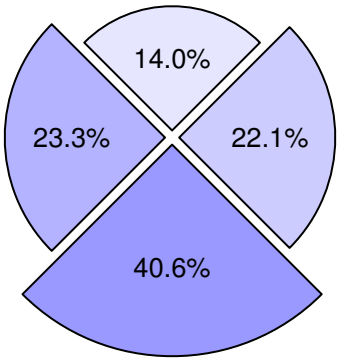
(b) Elevation



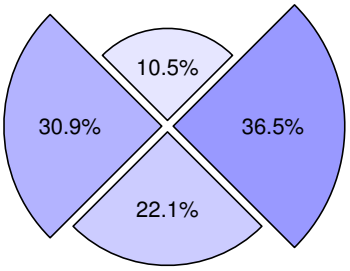
(c) Indoor



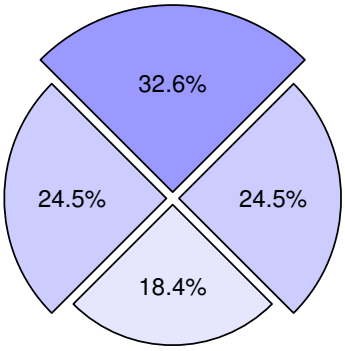
(d) Object



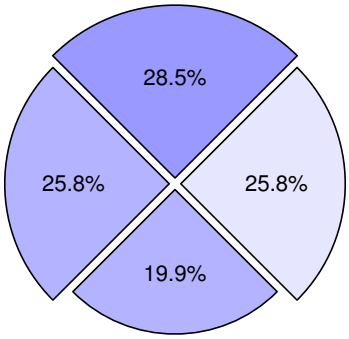
(e) Pavement



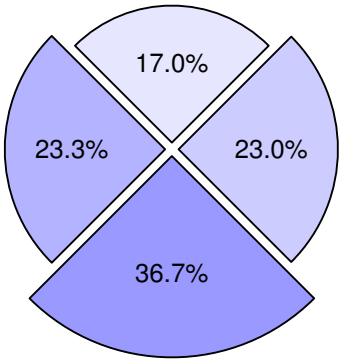
(f) Person



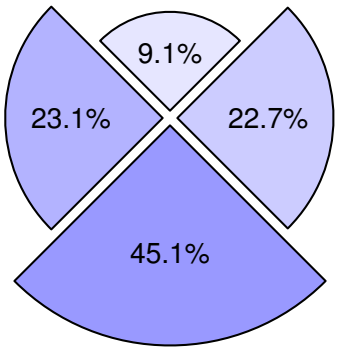
(g) Plant



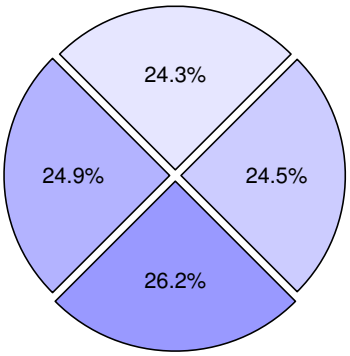
(h) Sky



(i) Soil

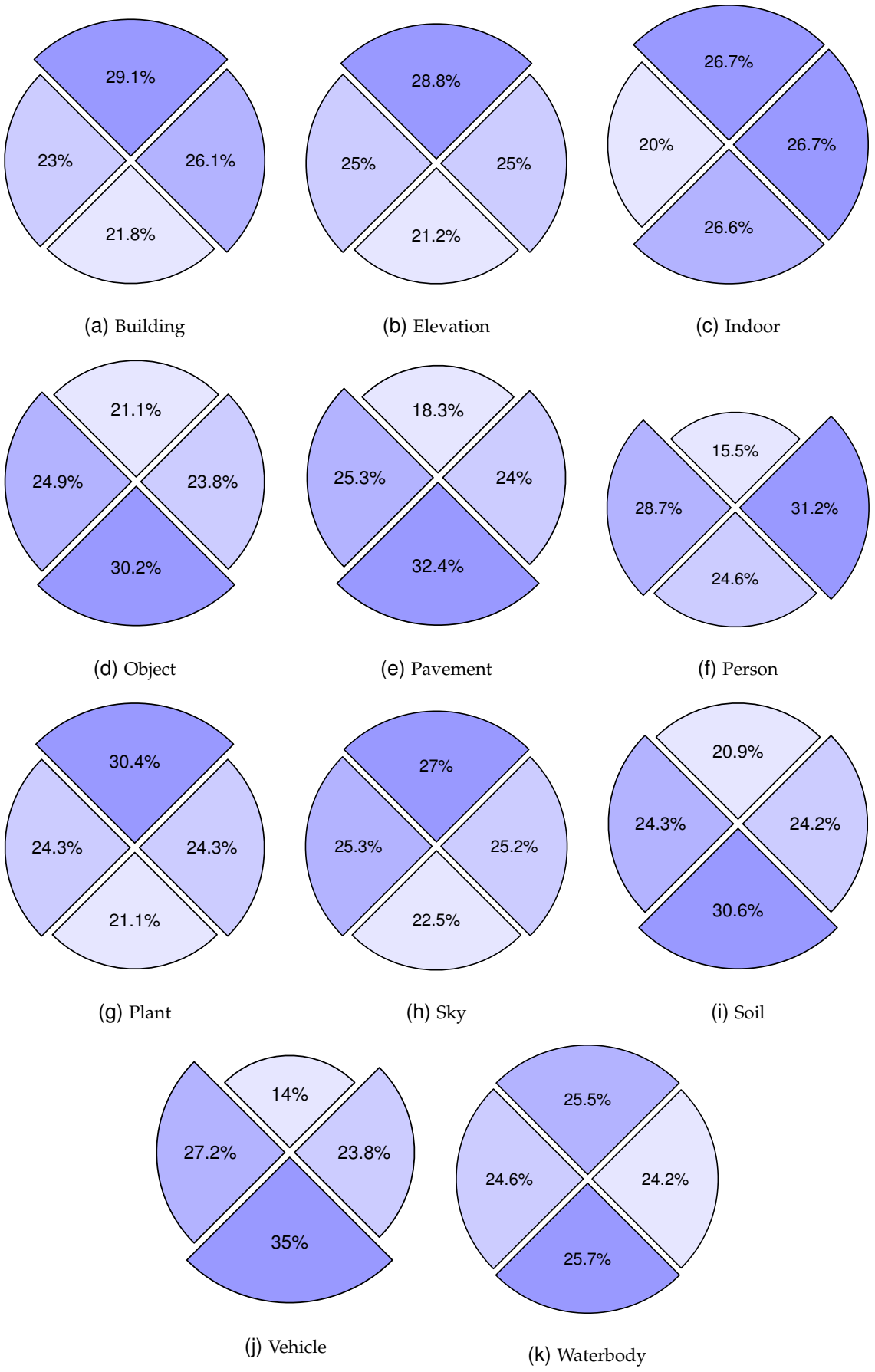


(j) Vehicle



(k) Waterbody

Figure 7: Per-class label-neighborhood transitions around airplanes in *SemanticAircraft* instances.



(a) Building (b) Elevation (c) Indoor

(d) Object (e) Pavement (f) Person

(g) Plant (h) Sky (i) Soil

(j) Vehicle (k) Waterbody

Figure 8: Per-class label-neighborhood around airplanes in *SemanticAircraft* quadrants.

Table 10: Direction-wise label transition from airplane pixels to other classes in instances of *SemanticAircraft*. Sky is not only the most prevalent context class as a whole, it is also oftentimes the stuff-class immediately surrounding airplanes.

Up	Down	Left	Right
sky: 59.2	sky: 41.4	sky: 53.8	sky: 53.7
building: 11.1	pavement: 27.1	pavement: 15.6	pavement: 14.8
pavement: 9.4	soil: 10.5	building: 8.4	building: 9.2
plant: 7.7	building: 7.0	soil: 6.7	soil: 6.6
soil: 4.9	plant:4.4	plant: 5.8	plant:5.8
elevation: 3.7	object: 3.3	elevation: 3.0	elevation: 3.1
object: 1.8	elevation: 2.2	object: 2.6	person: 2.7
waterbody: 1.3	person: 1.6	person: 2.3	object: 2.3
person: 0.8	waterbody: 1.4	waterbody: 1.3	waterbody: 1.3
vehicle: 0.2	vehicle: 1.2	vehicle: 0.6	vehicle: 0.6
indoor: 0.04	indoor: 0.03	indoor: 0.04	indoor: 0.04

Table 11: Label-neighborhood transitions from airplane pixels to other classes in *SemanticAircraft*-quadrants.

Up	Down	Left	Right
sky: 48.3	sky: 40.4	sky: 45.4	sky: 45.2
pavement: 10.0	pavement: 17.7	pavement: 13.8	pavement: 13.1
building: 7.2	soil: 7.3	soil: 5.8	building: 6.5
plant: 5.9	building: 5.4	building: 5.7	soil: 5.8
soil: 5.0	plant: 4.1	plant: 4.7	plant: 4.7
elevation: 2.8	elevation: 2.1	elevation: 2.5	elevation: 2.5
object: 1.4	object: 2.0	person: 2.3	person: 2.5
person: 1.2	person: 1.9	object: 1.6	object: 1.5
waterbody: 1.2	waterbody: 1.2	waterbody: 1.1	waterbody: 1.1
vehicle: 0.3	vehicle: 0.9	vehicle: 0.7	vehicle: 0.6
indoor: 0.02	indoor: 0.02	indoor: 0.01	indoor: 0.02

5 Domain Prediction on *SemanticAircraft*

Having used semantic context to gain insights into the datasets and filter irrelevant and undesirable patches, this chapter will introduce the application of semantic context for the task of domain prediction. Figure 9 gives a general overview of the framework for domain prediction using either RGB input images or semantic masks.

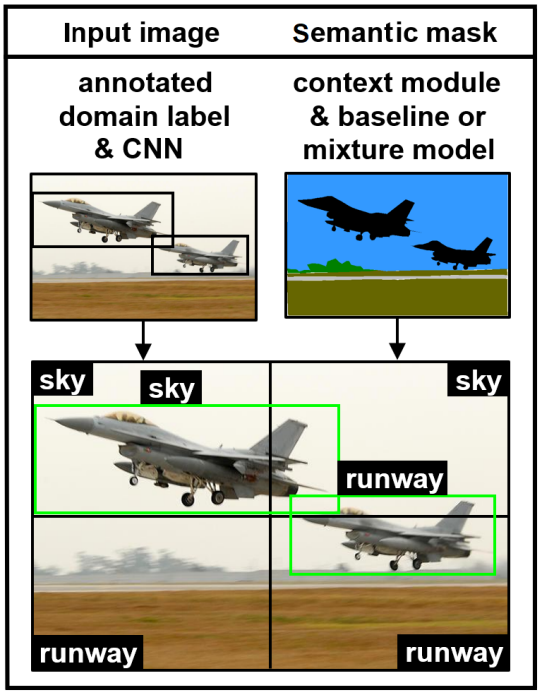


Figure 9: The developed framework can predict domains of *SemanticAircraft* instances and quadrants using either input images with domain-labels training a CNN or use the extracted context vectors $\mathcal{C}$ with either a (quasi) deterministic threshold algorithm as $\mathrm{baseline}$ or unsupervised mixture models for clustering of context vectors.

5.1 Domain Prediction as Computer Vision Task

Domain adaptation outlined in section 2.4 aims to produce a model that performs well simultaneously in two domains, most commonly source $\mathcal{S}$ and target $\mathcal{T}$, the former consisting of synthetic and the latter of real images. When using domains as characteristic expanded semantic context around objects, domain prediction instead aims to parse the scene to allow decision-making about the specific domain an object is in. As such it is a simple classification task not limited to images – instead learning the domain predictive function $f(\cdot)$ uses any measure or representation of semantic context. An implicit representation is the one found in CNN

featuremaps, using convolutional filters to capture context. An explicit, statistical measure are the context vectors $\mathcal{C}$ obtained from the semantic context module. While function $f(\cdot)$ is used for domain prediction, $g(\cdot)$ might be used for any other CV task, such as object detection or tracking. Using the predicted domains from $f(\cdot)$ as input for the learning framework encapsulating $g(\cdot)$, predicted domains allow the latter function to be trained and fine-tuned on any one specific target domain separately – no generalization across vastly different visual domains is required. Instead the parameter set of $g(\cdot)$ is exchanged dynamically according to predicted domain labels and $g(\cdot)$ is expected to yield much better results than one model trained on all three domains instead. In this work, only the problem of obtaining $f(\cdot)$ is tackled.

To summarize, the domain prediction module is used to separate instances and image quadrants from datasets into domains based on semantic context priors. Three models were employed to this end: a specifically developed threshold algorithm as baseline, unsupervised mixture models and supervised CNNs. The three models take broadly the following approaches:

Baseline

First, define domains an as ontology of classes and superclasses and using the context vectors $\mathcal{C}_i$ and $\mathcal{C}_q$ for instances and quadrants respectively, run a threshold algorithm using ranges and weights for specific context superclasses. This does not necessitate ground truth (GT) labels, although feedback using classification accuracy is used to tune the parameters.

Unsupervised

Using the (training) set of context features $\mathcal{C}_i$ and $\mathcal{C}_q$, fit an unsupervised ML model to predict the domain for unseen context vectors, i.e. interpret label statistics per patch as features and use unsupervised learning for clustering – reinterpret the clusters for a classification setting. This procedure does not use GT labels in any way to influence the learning, but only for final evaluation against the other models.

Supervised

Instances and quadrants of color-images from *SemanticAircraft* were assigned a GT domain label manually. The CNN uses these images and annotations for supervised classification.

The general premise when using supervised vs. unsupervised models is different: Supervised algorithms are highly capable of learning/memorizing the data structure and infer a category for unseen samples. As such, supervised models such as neural networks are highly accurate, but provide little insight into the training data itself. Unsupervised models, and especially generative methods, instead provide further insight into the distributions that created the data itself, an explainability that is not present in CNNs. For the purpose of domain prediction, classification accuracy is the primary goal and all models are tuned in a way to maximize this accuracy – although this is not straightforward for unsupervised models if no ground-truth is known. Accuracy with supervised methods is expected to surpass other models, but with unsupervised clustering algorithms the created cluster-structures can additionally provide insights into possible subdomains of the data.

Since the supervised CNN as classifier requires GT domain labels, instances and quadrants from *SemanticAircraft* were manually annotated with domain labels in a one-hot scheme, i.e. every instance/quadrant is assigned exactly one domain label. Using GT labels also allows the calculation of classification accuracy, specifically recall and F_1, across the different prediction models to enable cross-evaluation.

5.2 Domain Annotation for *SemanticAircraft*

The four domain-labels are: Apron (0), runway (1), sky (2) and other (3) and any one labels is assigned to any instance and image-quadrant of *SemanticAircraft*, after context filtering. It should be noted that the scale in which context is to be considered is still an open question. In this work it was decided to work with increased instances and image quadrants to provide both narrow-range context around the airplane and far-range context across the image. Additionally, separating images into quadrants makes one-hot classification better behaved: Assigning a single label to an image patch is difficult – this is already the case with quadrants and instances but it is exacerbated on an image level, where the airplane might only be shown in front of the sky, but the bottom half might be filled with airport buildings, runways or similar objects.

A number of rules were established to guide the annotation process for both instances and quadrants. It should be noted that the annotation process is not straight-forward and no two annotaters would give the exact same labels on any sufficiently large and varied set of images. In particular, instances or quadrants would often-times show features strongly related to $\mathrm{apron/runway}$ at the bottom while clear sky was visible in the upper segments. It should further be noted that all patches were shuffled for annotation, i.e. no instances or quadrants from the same image appear in sequence (unless by chance). This was done to prevent bias while annotating, which would arise from including information seen in a previous quadrant to form a decision on the next. Finally, exceptions to any of the established rules were made in some cases. Figures 13 and 14 in chapter 6 show example instances and quadrants from all domains.

- Patches in grayscale or sepia tones were classified as other.
- Images with natural or artificial (padded) borders, such as scanned photographs or illustrations were classified as other.
- Airplanes shown on unusual airfields, such as lawns, fields, beaches or on water were labeled as other.
- Airplanes in otherwise atypical environments such as a snow-covered scenery were assigned a respective domain label, if a distinction between apron and runway could be made. If this distinction was not possible they were assigned other.
- The few airplanes left in indoor scenes that were not filtered out were classified as other.
- Airplanes from *ADE* on or around the aircraft carrier were classified as other.

- Monochrome patches with no airplane visible were labeled as sky if the patch was blue or grey and some textures were recognizable. If no image texture was present at all, patches were labeled as other.
- Since the semantic context is derived from neighboring pixel-labels in the image plane, the point-of-view when observing objects plays a significant role. Images captured from airplanes themselves show e.g. fields, forests or the sea as context while images from below would show mainly sky and clouds.
- Some patches show an airplane in the foreground landing or taking off from the runway, while in the back the rest of the airport apron is visible – or the focused instance is small in the background while the foreground is dominated by another airplane. The assigned label always deals with the surroundings of the instance in question, even if more image area falls to context at other distances different from the target.
- Another distinguishing factor between apron and runway was the presence of people. If people are present and were, for example, boarding an airplane, the patch was labeled as apron, even if the strip of asphalt is mostly likely used for take-off and landing as well. This was prevalent at small airports in rural places, where the runway often times functions as the apron as well.
- Since the amount of patches showing mainly sky as context is by far the largest, to help preemptive class balancing while still trying to properly assign TOL instances to either runway and sky, patches were labeled as runway even if the context was largely sky, as long as treetops or other ground features expected in the vicinity of airfields appear near the bottom edge. If the airplane appears to be far away from the actual runway, but a few treetops are visible, the domain would be sky regardless.[1]
- In general, viewing angle, landing-gear visibility and relative size of the airplane vs. other objects was used to judge whether an airplane was bound to be landing soon/just took-off and assigned runway or if it was in-flight and assigned sky.
- Very few aircraft were captured in the taxiway domain and even if, the markings on the ground give a clue as to the type of path the aircraft is on, allowing the distinction between apron and runway. In general, prolonged straight white lines mark the runway, while curved yellow lines signify paths in close proximity to the apron.
- A trend across all datasets could be noticed: Many airplanes were in fact captured during take-off or landing.
- Captures of airplanes at the apron were often of lower image quality, taken through a window, from the inside of airplanes or would show close-by, heavily occluded airplanes.

[1] The thirty-percent increase of bounding boxes makes a significant impact in this regard. If instances were instead not increased in size at all, an increased number would instead be assigned sky.

- It should be noted, that a small amount of quadrants showed no airplane-pixels at all. Such patches were not excluded and assigned a domain label depending on the respective content. This means, that two image quadrants from the same image with similar background, but one quadrant showing the airplane and the other not, look the same context-wise to the baseline and unsupervised models, since target pixels were excluded for calculation of $\mathcal{C}$. The CNN on the other hand works on input images and incorporates this presence/absence in its reasoning.

- Finally, if it was not possible to decide on either apron, runway or sky for example due to the absence of an airplane with no other distinguishable features visible, or for any other reason, the patch was assigned the other label.

After annotation, the dataset *SemanticAircraft* consisted of a set of 3854 instances and 13265 quadrants. Every patch in turn consists of the triplet of: RGB input image from either *ADE*, *COCO* or *PASCAL*, corresponding context vector $\mathbf{c}_{i/q}$ obtained with the semantic context module from the semantic masks, and GT domain label from the annotation process. As such, *SemanticAircraft* consists of two classes of sets of images ($\mathcal{I}_\mathbf{i}$ and $\mathcal{I}_\mathbf{q}$), context vectors ($\mathcal{C}_\mathbf{i}$ and $\mathcal{C}_\mathbf{q}$) and labels ($\mathbf{l}_i$ and $\mathbf{l}_q$) respectively. The model search and learning process took place in two phases. In the first phase, hyperparameters and architectural designs were tuned following the evaluation of method-specific metrics, using the training and validation split. Once the optimal version of a model was found, final evaluation used reshuffled training and test data. To avoid having strongly similar images in any two of the splits, all instances and quadrants taken from the same source image are assigned the same split.

Specifically, before training begins in phase-1, both instances and quadrants are split into five folds each making up twenty-percent of the data. This follows the idea of k-fold cross-validation. The basic premise behind using k-fold cross-validation is to remove the randomness introduced when splitting the dataset into training/validation/test arbitrarily. For example, it is possible that all the "easy" samples happen to end up in the test split, skewing the actual performance of the model. Folds are introduced to allow evaluation on every example while still keeping training and test data mutually exclusive per fold. In the beginning 20% were completely excluded and left untouched during model tuning. From the remaining 80% four "active" folds are created with mutually-exclusive validation data. This means for fold-1 the first 60% of the entire dataset are used as training data while $60-80\%$ are validation, for fold-2 percentages $20-80\%$ are used for training, $0-20\%$ for validation and so forth. The results during model search are all given using average scores on validation data across all four folds.

For phase-2 two separate versions of *SemanticAircraft* were used. The first set consists of all instances and quadrants. For the second set all instances and quadrants with the GT domain label other are removed. Other patches are not included during training or evaluation in this case.[2] The dataset is shuffled again, split into five equal-sized subsets used to create five different folds. In this way, every sample is used once for validation/testing and three times for training. In phase-2 reported results are all averaged across the test-sets of all five folds.

[2] If one quadrant or instance was assigned other during annotation, only this patch is excluded not all instances/quadrants from that source image.

In the following three sections for every method the used algorithm is first described, followed by model-tuning in phase-1 and final application and evaluation in phase-2.

5.3 Baseline Threshold Model

The algorithm proposed in this section works (almost) deterministically and predicts domains using context-priors. It is highly parameterized by design and serves as the baseline for domain prediction evaluation.

5.3.1 Hierarchical Threshold as Domain Prediction Baseline

The basic premise of the baseline was to develop an algorithm that works similar to human intuition: The quantity of every context-class contributes towards a certain domain-belief with a set strength if that quantity is "as-expected" for any domain . For example, apron samples are commonly expected to feature vehicles in the context while runway and sky are not. Observing individual classes of the context for any patch, e.g. $\mathbf{c}_{vehicle} = 0.4$ means forty percent of pixels are $\mathrm{vehicle}$, the ranges of expected vehicle context for all three domains are checked and every domain with bounds including 0.4 gets assigned a score, symbolizing the level of distinction the context provides. If $\mathrm{vehicle}$ is a superclass and context estimates are only available for the respective sub-classes, the scores of all its individual classes are first added together. This is done for every superclass with scores adding up until the final domain-scores are reached. The scores or weights are between $[0, 100]$ and the sum of scores for all superclasses per domain is 100. For classification, an image patch then has to reach a configurable score-threshold. All parameters were implement using expert knowledge. This baseline domain-prediction method is outlined in algorithm 4.

If $argmax(\cdot)$ sees two domains with the same score, the label is assigned at random between the tied domains. If all domain-scores do not meet the score-requirement (80 by default), threshold is decreased in steps of 5 until a score exceeds it. This is slightly different than assigning the highest-scoring domain, and allows collecting information how many times the threshold was decreased as an uncertainty measure. While simple to configure for first time-usage, this algorithm shows multiple shortcomings:

- It is incredibly parameter-heavy, making both tuning for a set of domains and extension to other domains tedious.
- All parameters are partly dependent on expert-knowledge, and need to be informed by previous dataset-wide semantic context analysis.
- Equal domain-scores lead to ambiguity – in this case, this ambiguity was solved with random tie-breaks but an alternative could be to allow multi-labels or assign the domain which met the superclass requirement with the highest weight w. Although the latter alternative would bias towards domains with a less uniform distribution across the weights.

Algorithm 4 Predict domains (quasi) deterministically via hierarchical thresholding

Requisites: Set of context vectors $\mathcal{C}_{\mathbf{i/q}}$ to predict domains upon. List of n domains $\mathbf{d}$ and for every domain and every superclass $\mathbf{s}$ consisting of classes c in that domain a certain range e.g. $\mathbf{r}_{apron,vehicle} = [0.0, 0.5]$ and corresponding weight $\mathbf{w}_{apron,vehicle} = 20$ that the superclass falls in that range. Finally, a domain-prediction threshold of th for all domains and possibly a decrease th_d.

1: **function** THRESHOLDDOMAINPREDICTION($\mathcal{C}_{\mathbf{i/q}}, \mathcal{D}, \mathcal{S}, \mathcal{R}, \mathcal{W}, th, th_d$)
2: **for** $\mathbf{c}$ in $\mathcal{C}$ **do**
3: domain_scores $\leftarrow \mathbf{0} : \mathbf{0} \in \mathbb{R}^{n \times 1}$
4: **for** d in $\mathcal{D}$ **do**
5: **for** $\mathbf{s}$ in $\mathcal{S}$ **do**
6: super_score $\leftarrow \sum_i c_i, \forall c \in \mathbf{s}$
7: **if** super_score $\in [\mathbf{r}_{d,s,lower}, \mathbf{r}_{d,s,upper}]$ **then**
8: domain_scores$_d$ $\leftarrow$ domain_scores$_d$ $+ \mathbf{w}_{d,s}$
9: **if** $\max$(domain_scores) $> th$ **then**
10: $\mathbf{l_c} \leftarrow argmax$(domain_scores) ▷ Break ties randomly
11: **else**
12: $th \leftarrow th - th_d$ ▷ Go-to line 9 until domain found
13: **return** $\mathbf{l}$ ▷ $\mathbf{l}$: vector holding a domain-label for every patch

- Patches not meeting the threshold th signify high uncertainty in the context or out-of-context patches. Instead of lowering the threshold it might be more advantageous to discard the patch altogether.

Despite these drawbacks, once set-up for a set of domains and datasets, results are largely reproducible due to the deterministic nature and inference-time is negligible since the computational complexity and dimensionality of the context vectors is low, enabling fast tuning of the parameters.

5.3.2 Parameter Tuning of the Baseline Model

Three different parameter-sets were used during model-tuning – Table 12 shows the sets of configurations.

As metric for evaluation GT domain labels were used and classification accuracy (recall) calculated – Table 13 provides the results.[3] With configuration-I many patches were wrongly classified as sky since it is so prevalent as context and the restrictions via ranges were not strong enough. Therefore, the lower bound of $\mathbf{r}_{sky,sky}$ (sky is both a domain and a superclass occurring in every domain) was raised from 60 to 90, while the upper bounds of sky in the other domains were raised. This led to definitive improvements in apron and runway, while accuracy

[3] Because of this inclusion of GT knowledge and subsequent parameter tuning, the application of this algorithm is not entirely unsupervised in its nature, even if no supervised learning procedure takes place.

Table 12: Ranges and weights used for domain prediction using the baseline, during all three configurations. For every domain (column) and every (contextually relevant) superclass of *SemanticAircraft* (rows), the left-hand subcolumn gives the specific range, while the right subcolumn gives the corresponding weight. Changes to parameters are reflected in the color, with black being the optimal found value. For example, for the domain other the upper bound of sky-context was changed from 50 to 60 back to 50 again from configurations I to III.

	Apron		Runway		Sky		Other	
	$\mathbf{r}_{apron}$	$\mathbf{w}_{apron}$	$\mathbf{r}_{runway}$	$\mathbf{w}_{runway}$	$\mathbf{r}_{sky}$	$\mathbf{w}_{sky}$	$\mathbf{r}_{other}$	$\mathbf{w}_{other}$
Sky	[0, (40) 60]	(6) 16	[0, (99) 99.5]	(25) 20	[(60) 90, 100]	10	[0, (50, 60) 50]	(2, 6) 2
Pavement	[(8) 3, 100]	(25) 18	[0, (50) 60]	(5) 3	[0, 0]	15	[0, 25]	(3, 6) 3
Building	[0, (90) 98]	3	[0, 10]	(5) 3	[0, 2]	10	[0, 40]	(3, 6) 3
Soil	[0, (18) 25]	(15) 14	[0, 70]	(5) 3	[0, 5]	5	[0, 70]	(3, 6) 3
Elevation	[0, (2) 5]	3	[0, (25) 35]	3	[0, 65]	1	[0, 65]	(3, 6) 3
Plant	[0, 1]	(11) 10	[0, (85) 95]	3	[0, (15) 10]	1	[0, 85]	(3, 6) 3
Person	[0, (70) 80]	10	[0, 0.2]	30	[0, 0]	20	[0, 100]	(37, 23) 37
Vehicle	[0, (40) 50]	3	[0, 5]	(10) 16	[0, 0]	15	[0, 60]	(3, 6) 3
Waterbody	[0, 5]	(11) 10	[0, 25]	2	[0, 85]	10	[0, 100]	(3, 6) 3
Indoor	[0, (5) 10]	3	[0, 0]	2	[0, 0]	1	[0, 100]	(37, 23) 37
Object	[0, (20) 30]	10	[0, 0]	(10) 15	[0, 0]	12	[0, 60]	(3, 6) 3

Table 13: Recall of the baseline model during model-tuning. The model performs best on sky – the simplest of domains in terms of image difficulty. Since sky is also the most frequent domain, total recall is higher average recall.

Configuration	Apron	Runway	Sky	Other	**Average**	**Total**
			Instances			
I	0.2065	0.3697	0.8893	0.6658	0.5328	0.5358
II	0.3865	0.4489	0.8882	0.5303	0.5635	0.5768
III	0.3751	0.4193	0.8904	0.592	**0.5692**	**0.5825**
			Quadrants			
I	0.4689	0.3145	0.8946	0.5899	0.567	0.633
II	0.5192	0.5156	0.8823	0.4588	**0.594**	0.6376
III	0.5133	0.4255	0.8807	0.5178	0.5843	**0.6423**

in the other domain dropped off significantly. To remedy this effect, changes to the ranges and weights of other were reverted, which results in configuration-III.

Following total accuracy, configuration-III is the best variant for both instances and quadrants. Looking at average accuracy, configuration-III performs best on instances but is outperformed by configuration-II on quadrants, due to a 9-percent difference in runway prediction. Also, simply reverting the changes made to the other-parameters for configuration-III was not enough to restore the strong original performance. Since configuration-III provided the best results in three out of four metrics it was also used for application on the test set.

5.3.3 Evaluation of the Baseline

As previously outlined, four subsets of *SemanticAircraft* were created for final evaluation: instances and quadrants with and without the domain other. Every subset has five folds with training, validation and test splits. The algorithm was applied to all patches and Table 14 shows results as recall, precision and F_1 score (harmonic mean of precision and recall), relating true positive (TP), false positive (FP) and false negative (FN) classifications.[4] Additionally, the standard deviation with bessel's correction for recall is given.

Table 15 lastly provides the confusion matrices for all four subsets. Additionally, for every domain the number of patches assigned to that domain ($TP+FP$) vs. the true number patches following domain annotation ($TP + FN$) is visible.

While results obtained with the baseline are reasonably good, the limited nature of the model due to the strong parameterization significantly restricts its usability. Therefore, it was decided to proceed with a learning based approach in the form of unsupervised ML models.

[4] The average F_1 score is the arithmetic mean of other F_1 scores not the harmonic mean of average precision and recall.

Table 14: Final domain prediction results using the baseline. Classification results improved significantly when excluding other patches. The difference in accuracy between instances and quadrants is insignificant. Notable also is the strong precision on apron patches across the board. Low precision with other samples indicate a significant amount of false-positive assignments to the other domain during classification. For total measures, precision is equivalent to recall.

	Apron	Runway	Sky	Other	**Average**	Total
			Instances			
			Including Other			
Precision	0.7445	0.4166	0.939	0.3152	0.6039	0.588
Recall	0.382 ± 0.033	0.425 ± 0.035	0.89 ± 0.037	0.579 ± 0.018	0.569 ± 0.031	0.588 ± 0.015
F_1	0.5049	0.4208	0.9138	0.4082	**0.5619**	0.588
			Excluding Other			
Precision	0.8312	0.5265	0.9849	—	0.7809	0.796
Recall	0.649 ± 0.025	0.75 ± 0.033	0.968 ± 0.013	—	0.789 ± 0.024	0.796 ± 0.011
F_1	0.7289	0.6187	0.9764	—	**0.7747**	0.796
			Quadrants			
			Including Other			
Precision	0.7248	0.4354	0.8778	0.3727	0.6027	0.639
Recall	0.511 ± 0.028	0.417 ± 0.032	0.884 ± 0.011	0.502 ± 0.022	0.5785 ± 0.023	0.639 ± 0.017
F_1	0.5994	0.426	0.8809	0.4278	**0.5835**	0.639
			Excluding Other			
Precision	0.8037	0.5584	0.9411	—	0.7677	0.799
Recall	0.632 ± 0.0224	0.734 ± 0.0149	0.922 ± 0.0148	—	0.7627 ± 0.0174	0.799 ± 0.006
F_1	0.7076	0.6343	0.9315	—	**0.7578**	0.799

Table 15: Confusion matrices for baseline results, showing significant confusion between runway and other as well as apron and other. Actual labels (rows) are visible on the left vs. predicted labels (columns). Sky is predicted with the least confusion. In instances, 1399 samples are assigned other, which is more than any other domain, while only 762 instances were actually annotated with the other label.

	Including Other				Excluding Other			
Instances								
	Apron	Runway	Sky	Other	Apron	Runway	Sky	
Apron	0.382	0.199	0.001	0.418	0.649	0.349	0.002	1207
Runway	0.070	0.425	0.015	0.490	0.228	0.750	0.022	675
Sky	0.001	0.008	0.89	0.101	0.004	0.028	0.968	1210
Other	0.144	0.200	0.077	0.579				762
	619	689	1147	1399	942	961	1189	
Quadrants								
Apron	0.511	0.150	0.024	0.315	0.632	0.327	0.041	2966
Runway	0.098	0.417	0.062	0.423	0.189	0.734	0.077	2351
Sky	0.002	0.045	0.884	0.069	0.002	0.076	0.922	5233
Other	0.124	0.217	0.157	0.502				2715
	2089	2251	5268	3657	2333	3091	5126	

5.4 Unsupervised Clustering and Mixture Models

In this section, the unsupervised ML models applied for the prediction of domains on the sets of context vectors $\mathcal{C}_i$ and $\mathcal{C}_q$ are detailed. As a reminder, the dimension for every context vector $\mathbf{c}$ is $M = 11$, while the number of domains is $D = 4$. First, the mathematical background of unsupervised clustering and mixture models is given, largely derived from Bishop [104]. It should be noted, that any created cluster are an internal mathematical construct and do not resemble the set of predefined domains. This makes interpretation in a classification setting not as straightforward as using softmax scores in CNNs. Afterwards, the tuning and application of the models on *SemanticAircraft* is given.

5.4.1 Unsupervised Clustering Algorithms and Mixture Models

Chapters 9 and 10 from Bishop*et al.* [104] provide many details for clustering algorithms, mixture models, the idea of expectancy maximization (EM) and the extension for variational methods.

Without the knowledge of GT domain-labels for the context vectors, classification can be done by finding clusters in the set of data points, where every cluster corresponds to a domain that generated the observed data. A popular non-probabilistic technique for clustering is the K-means algorithm [105] creating cluster regions similar to Voronoi diagrams.

Assuming D clusters in the data, a set of M-dimensional vectors μ_d where $d = 1, \ldots, D$ is a cluster prototype, or the center of cluster d. The goal is to find an assignment of data points to clusters to minimize the sum-of-squares of (euclidean) distances from data to center points μ_d. For every data point $\mathbf{c}$, binary variable $r_{nd} \in 0, 1$ describes which cluster d the vector $\mathbf{c}$ is assigned to (one-hot). This leads to the objective function J which is to be minimized for every one of N observations:

$$J = \sum_{n=1}^{N} \sum_{d=1}^{D} r_{nd} \mid \mathbf{c}_n - \mu_d \mid^2 \tag{1}$$

Optimization takes place in two steps, the E (expectation) and M (maximization) steps. First, r_{nd} is set 1 for the cluster d that gives the minimal $\mid \mathbf{c}_n - \mu_d \mid^2$, i.e. every context vector $\mathbf{c}_n$ is assigned to the cluster of the closest cluster center. Then, having chosen r_{nd}, the objective function J can be minimized using the derivative with respect to μ_d, giving:

$$\mu_d = \frac{\sum_n r_{nd} \mathbf{c}_n}{\sum_n r_{nd}} \tag{2}$$

The interpretation of this step is to set μ_d equal to the mean of all datapoints assigned cluster d. These steps are repeated until convergence, although K-means is not guaranteed to converge to a global optimum.

Taking a probabilistic approach, in simple terms, a Gaussian mixture models (GMM) is a probabilistic model that assumes all data-points were generated from a finite mixture of Gaussians with unknown parameters. It can be seen as a way of generalizing K-means clustering by incorporating information about the covariance structure of the data. As Bishop [104] write,

the Gaussian mixture distribution can be written as linear superposition of d Gaussians with mixing coefficients π_d:

$$p(\mathbf{c}) = \sum_{d=1}^{D} \pi_d \mathcal{N}(\mathbf{c} \mid \mu_d, \mathbf{\Sigma}_d) \tag{3}$$

Introducing a D-dimensional binary (latent) variable $\mathbf{z}$ with all the properties for a well-defined marginal probability $p(\mathbf{z})$, the conditional $\gamma(z_d) \coloneqq p(z_d = 1 \mid \mathbf{c})$ is known as the "responsibility" that component d takes towards explaining the observation $\mathbf{c}$. This responsibility plays an important role in the expectation step: using the log likelihood of posterior observations, the steps of EM for Gaussian mixtures are generally as follows:

1. Initialize the means μ_d, often times with a preliminary run of K-means, covariances $\mathbf{\Sigma}_d$ and mixing coefficients π_d.
2. Expectation: Evaluate responsibilities $\gamma(z_d)$ using observations $\mathbf{c}$ and current parameters.
3. Maximization: Update μ_d, $\mathbf{\Sigma}_d$ and mixing coefficients π_d using the responsibilities $\gamma(z_d)$.
4. Evaluate the log likelihood of the model and repeat steps 2-4 until convergence criterion is satisfied.

As Blei *et al.* [106] point out, for variational reasoning, if any unknown parameters in the probabilistic model are given prior distributions, these parameters can be absorbed into the set of latent variables $\mathbf{z}$, resulting in a fully Bayesian model. Given infinite computation time, they promise to find a global optimum for the parameters of the distributions that generated observations $\mathbf{c}$. As Bishop [104] write, approximation schemes are often required due to analytic intractability or prohibitively expensive calculation. Variational inference (VI) or variational Bayes is such an approximation technique which has been widely applied [107]. At this point it should be noted, that VI in this classical Bayesian interpretation is largely intractable for any high-dimensional problem, prohibiting the application on images directly, which are oftentimes 10-s of thousands or millions-dimensional. The representation in the form of 11-dimensional context vectors makes this problem tractable with VI.[5] For the scope of this work, it suffices to say, that all model parameters are absorbed in the set of latent variables $\mathbf{z}$. As Blei *et al.* [106] write, when postulating a family of densities $\mathcal{Q}$ the problem of optimization becomes one of minimizing the Kullback-Leibler (KL) divergence.

$$q^*(\mathbf{z}) = \underset{q(\mathbf{z} \in \mathcal{Q})}{\operatorname{argmin}} KL(q(\mathbf{z}) \mid p(\mathbf{z} \mid \mathbf{c})) \tag{4}$$

The posterior $q(\mathbf{z})$ is then approximated with the optimized member of the family $q^*(\cdot)$. This approximation is typically done using the evidence lower bound. The latent variable from $\mathbf{z}$ describing the probability of the observation belonging to any of the D domains is categorically distributed. In Bayesian settings, if the posterior and prior belong to the same probability

[5] From sparse Gaussian processes (GP) [108] to Bayesian GP latent variable models (LVM) [109] to finally deep Gaussian processes (DGP) [110], DGP are one way of merging DL capabilities of processing high-dimensional data while keeping variational/Bayesian reasoning intact.

distribution family, the distributions are conjugate and the prior is a conjugate prior. A conjugate prior is an algebraic convenience allowing closed-form expression of the posterior. The Dirichlet distribution is a conjugate prior to the categorical, and thus, for variational Bayesian gaussian mixture models (VBGMM) the prior distribution type is typically Dirichlet. Finally, for infinite non-parametric discrete distributions, the Dirichlet process is the conjugate prior. For an introduction to Dirichlet processes see [111].

Lastly, to compare the performance of K-means, GMMs and VBGMMs, different metrics provided by [112] were used. In a probabilistic setting and for selection among a finite set of models, the Bayesian information criterion (BIC) is commonly used, which maximizes likelihood while introducing a penalty term for high numbers of model parameters to prevent overfitting. BIC cannot be formulated for K-means, due to the non-probabilistic nature of the algorithm, or VBGMM due to the infinite set of possible models. Another common metric which works with the clusters explicitly was instead used, an extension of the silhouette score [113], the silhouette coefficient [114]. The silhouette score is a measure of how similar a data-point is to its own cluster compared to other clusters. It is bound by $[-1, 1]$ where 1 marks perfect cluster assignment, 0 symbolizes overlapping clusters and -1 is the worst possible "inverse" assignment. Silhouette score is calculated with any distance metric and is calculated for every data-point, although the selection of distance metric, by default euclidean, is not straightforward, since distance metrics behave differently in higher dimensions [115]. The mean of all the silhouette-scores over a cluster is then proportional to the tightness of all grouped points in that cluster. Kaufman *et al.* [114] then introduced the silhouette coefficient, as the maximum of the means of individual silhouette scores across the entire dataset.

Before model setup, principal component analysis (PCA) was used to obtain the covariance ratio of every feature of the training data as a preliminary step.[6] The covariance ratio in every component following PCA using all eigenvectors was obtained across the four folds in phase-1 and can be observed in Table 16. Singular values of the specific eigenvectors are omitted, they ranged from 124 for the first component to 5.579×10^{-4} for the last, with a similar distribution characteristic as the covariance ratios, i.e. singular values were significantly less for the last 1 or 2 components (after standardization). The PCA-analysis was done both before and after standardization. Following direct observations, it appears that 1.) every component except one (presumably indoor) contributes to significant extent, 2) the difference between the covariance $\mathbf{\Sigma}$ of training vs validation set is negligible, in other words training and validation data were drawn (almost) i.i.d., 3) standardization significantly effects the covariance-ratios, bringing the significance of multiple features to light which would otherwise be lost due to differences in the relative scales of the features, in particular context being shadowed by sky. Since almost all context features hold relevance in terms of covariance, PCA was not used for the sake of dimensionality reduction in the following experiments. As a reminder, the semantic context module always already normalizes per-class context with the pixel-sum – context is further also always scaled between $[0, 1]$.

[6] T-distributed stochastic neighborhood embedding (t-SNE) was also shortly experimented with, although results proved more unstable, presumably due to the strong sparsity of most context vectors [116].

Table 16: Covariance ratio in all principal components following PCA of *SemanticAircraft*. Before standardization, it appears one component is dominating the covariance (presumably sky). With standardized data, the covariance is much more evenly spread. The last component, which is presumed to be indoor and almost entirely removed from the context, holds very little covariance.

Instances				Quadrants			
Training		Validation		Training		Validation	
—	Stand.	—	Stand.	—	Stand.	—	Stand.
0.6237	0.1810	0.6307	0.1855	0.6237	0.1776	0.6307	0.1757
0.1595	0.1155	0.1612	0.1211	0.1595	0.1091	0.1615	0.1111
0.0725	0.1065	0.0681	0.1007	0.0725	0.098	0.0681	0.0971
0.0419	0.0952	0.0436	0.0999	0.0419	0.0944	0.0436	0.0961
0.0412	0.0919	0.0332	0.0934	0.0412	0.0928	0.0332	0.093
0.0319	0.0906	0.0264	0.0893	0.0319	0.091	0.0264	0.0911
0.0129	0.087	0.0166	0.087	0.0129	0.0877	0.0166	0.0889
0.0095	0.0826	0.0133	0.0767	0.0095	0.0867	0.0133	0.0872
0.0069	0.0763	0.0071	0.074	0.0069	0.0858	0.0071	0.084
2.4×10^{-7}	0.0735	4.5×10^{-7}	0.0724	2.4×10^{-7}	0.0771	4.5×10^{-7}	0.0757
2.8×10^{-10}	7.4×10^{-10}	2.8×10^{-10}	7.7×10^{-10}	2.8×10^{-10}	2.8×10^{-10}	2.8×10^{-10}	2.8×10^{-10}

5.4.2 Model Search of Unsupervised Models

For unsupervised prediction a K-means, a GMM and a VBGMM were fit on the context vectors and used to infer domains for unseen context samples. The scikit-learn toolbox was used for implementation [112]. A number of other clustering algorithm are implemented in scikit – unfortunately, all of these models can only fit and predict on the same set of samples, i.e. a learning procedure using training, validation and test splits separately is not possible. It was therefore decided to not explore these algorithms.

While optimization of the parameters in the probabilistic models follows the EM scheme, the numerous hyperparameters of any of the models were tuned using grid-search methodology. Since models are fit individually on every fold, optimal hyperparameters can differ from fold to fold. In this case, the most common value or category for any hyperparameter was chosen. If this was not possible, the average was instead used. Also, the random state was fixed to 42 across all experiments for fair comparison and to allow reproducibility.

A number of methods exist for obtaining the "natural" number K of clusters within a dataset – information about the true nature of distributions might get lost when forcing a certain number of clusters. One such approach for any clustering algorithm is to use silhouette plots and means. Nevertheless, since unsupervised models' performance for domain prediction will be evaluated against both baseline and CNN models in a classification setting using GT label

information, it was decided to fix the number of clusters to be equal to the number of domains, both during phase-1 and phase-2.

K-Means Clustering

For the K-means model, the only tuned hyperparameter was the method of initialization. For configuration-I, the initialization method was set to $k - means++$ [117]. Alternatively, the cluster centers can be defined explicitly. It was considered to include the dataset-wide context statistics, either from *ADE*, *COCO* and *PASCAL* or *SemanticAircraft* to this end but using the dataset specific context averages as cluster-centers would lead the model to believe the various clusters symbolize different datasets, instead of domains. Incorporating the mean of context across the domains instead of datasets on the other hand would incorporate supervised information. For configuration-II of K-Means, it was instead experimented with using best-guesses for cluster-centers of every class and per domain, an idea similar to the ranges used in the baseline for thresholding. In this case, the number of runs defined by $n - init$ was forceably set to 1 since initialization parameters were passed explicitly. The initialization matrices used in configuration-II can be observed in Figure 10.

	Sky	Pavement	Building	Soil	Elevation	Plant	Person	Vehicle	Waterbody	Object	Indoor
Apron	0.4	0.2	0.05	0.05	0.01	0.01	0.08	0.05	0.01	0.02	0.01
Runway	0.6	0.2	0.01	0.15	0.03	0.05	0.01	0.01	0.01	0.01	0.01
Sky	0.92	0.01	0.01	0.01	0.03	0.01	0.01	0.01	0.03	0.01	0.01
Other	0.5	0.1	0.1	0.05	0.03	0.03	0.01	0.05	0.05	0.01	0.01

(a) Initialization matrix for means (cluster-centers) of instances during K-means clustering.

	Sky	Pavement	Building	Soil	Elevation	Plant	Person	Vehicle	Waterbody	Object	Indoor
Apron	0.3	0.4	0.1	0.01	0.01	0.01	0.05	0.05	0.01	0.02	0.01
Runway	0.65	0.25	0.01	0.2	0.03	0.05	0.01	0.01	0.01	0.01	0.01
Sky	0.98	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
Other	0.5	0.1	0.1	0.05	0.03	0.03	0.1	0.05	0.05	0.01	0.01

(b) Initialization matrix for means (cluster-centers) of quadrants.

Figure 10: For the K-means model it was attempted to set explicit cluster centers, using best-guess values derived from semantic context analysis. For every domain (row), an estimate was made regarding the average amount of context observed in that domain for every class (column).

Furthermore, every configuration was applied using either the context vectors $\mathcal{C}$ directly, or the data-matrix was first whitened via PCA-whitening [118], using the eigen-system of $\mathbf{\Sigma}$ obtained during PCA (although PCA was not used for dimensionality reduction). Other parameters for the K-means model are the number of times the algorithm is run with different centroid seeds: $n-init = 30$ – the maximum number of iterations during a single run: $max-iter = 300$ – and the relative tolerance regarding the (Frobenius norm of) cluster center difference in two consecutive runs: $tol = 0.0001$. In a final run $max - iter$ was raised from 300 to 1000. Observed silhouette results where the same, implying convergence into the tolerance boundary was achieved during 300 iterations already.

The best performing K-means model in terms of silhouette coefficient for both instances and quadrants used both standardization followed by PCA-whitening for data preprocessing and initialized the centroid seeds using $k - means++$. Achieved silhouette-coefficients were 0.4895 and 0.5173 for instances and quadrants respectively, meaning cluster-tightness was slightly more difficult to achieve with instances than quadrants.

Gaussian Mixture Models

For the GMMs, data-preprocessing steps again included the possibility of standardizing and/or whitening the data. The tuned hyperparameters were: 1) the initialization method for the means μ and covariances $\mathbf{\Sigma}$ of the Gaussians, using either random parameters or centers from a preliminary K-means run for the means and 2) the covariance-type, which can be either spherical, diagonal, tied or full.

Two experiments were conducted with either no preprocessing of the data, or standardization and whitening. In both attempts the optimal hyperparameters are those, where the silhouette-coefficient is at a maximum. The results and corresponding hyperparameters can be observed in Table 17. For a small number of sub experiments BIC was also recorded. While optimal hyperparameters differed, BIC and silhouette-coefficient itself correlated positively, i.e. a change in hyperparameters that caused an improvement (decrease) in BIC also caused and improvement (increase) of the silhouette coefficient.

To summarize, the best performing GMM could be obtained by first standardizing and whitening the data, using K-means for parameter initialization and employing tied covariance-matrices with silhouette-coefficients of 0.698 and 0.758 for instances and quadrants respectively, an improvement from 0.4895 and 0.5173 achieved with basic K-means.

Variational Bayesian Gaussian Mixture Models

For the final model, the VBGMM implementation from scikit-learn was used. It should be noted, that one advantage of variational methods is the non-requirement of a a pre-defined number of clusters – this is desirable in a truly unsupervised setting with next to no knowledge about the type and number of distributions that generated the data. In that context, letting the model decide the number of clusters yields clustering closer to the actual underlying distributions, without "hiding" information by forcing data-points to any fixed cluster. Here the number of clusters is known, or at least assumed to be equal to the number of domains. Experiments with

Table 17: Optimal hyperparameters according to the silhouette coefficients of GMMs fit on *SemanticAircraft* for domain prediction. A grid search was run to find the best combination of initialization and covariance type with respective data pre-processing. For all models it was discovered that the initialization of Gaussian means using K-means cluster centers improved the performance. The best performing models according to the observed silhouette coefficients also used either diagonal or tied covariance matrices, instead of a full covariance matrix. This is perhaps surprising, although Pedregosa *et al.* [112] argue, that mixture models with full covariance are prone to overfitting, since the mixture components can take any shape and location of distributions. Data pre-processing in the form of standardization and whitening provided similarly positive results.

		Initialization	Covariance	Standardization	Whitening	**Silhouette**
I	Instances	k-means	diagonal	F	F	0.645
	Quadrants	—"—	tied	—"—	—"—	0.545
II	Instances	—"—	—"—	T	T	**0.698**
	Quadrants	—"—	—"—	—"—	—"—	**0.758**

not setting the number of active components were conducted but since any number of clusters might be created, no evaluation against baseline or CNNs with four fixed domains would be possible. To be exact, the number of effective components can be inferred from the data, and this was done for configurations-I to III, setting the maximum number of components to 15 and letting the model itself decide what mixture components to activate. For configurations-IV onward the number of components is fixed at 4 – observation has shown that the VBGMM always activated those four components.

In addition to the hyperparameters of the GMMs, the prior on the weight-concentration of the Dirichlet distributions, in literature referred to as γ was also tuned. The values of γ could range from 10×10^{-5} to 10×10^{5}: large γ puts more mass in the center of a distribution resulting in more components being active, while smaller γ puts more mass at the edges. Priors on the covariance degrees of freedom and for the distribution on covariance itself are given using the Wishart distribution. For the initialization of the means and covariances of the responsibilities either K-means or random initialization can be employed, similar to the GMMs. For every run of the model, five different initializations were performed and the result with the highest lower bound on the likelihood kept. For the Dirichlet-process, which can be understood as a distribution whose range is itself a set of distributions, the stick-breaking representation is used to obtain a finite model. In terms of data-preprocessing, for every run the data can be either standardized, standardized and whitened or neither. The recorded hyperparameters at maximum silhouette-coefficients and the silhouette-coefficients themselves can be observed in Table 18. During configurations-I through III, the model itself chose the number of mixture components to activate. It was observed, that for instances up to 13 components were active and up to 9 for quadrants. This points towards the fact that the number of domains (4) might be too strong a restriction for the data. A number of other clusters/subdomains can be attributed

significant responsibility for creating the context, and besides the inclusion of the domains $\mathrm{taxiway}$ and TOL a few other subdomains seem to exist in *SemanticAircraft*.

The highest recorded silhouette-coefficient for instances was obtained with configuration-VI and a Dirichlet distribution prior. For quadrants, the highest value was actually obtained by letting the model decide on the number of active components, 9 were activated in this case. Since this would lead to domain prediction across 9 domains, but all other models (baseline and CNN) are restricted to 4, the second highest scoring configuration for test-set evaluation was instead used, a Dirichlet process prior in configuration-VI. Thus, the achieved coefficients are 0.702 and 0.766 for instances and quadrants respectively, just barely outperforming the non-variational GMM with coefficients 0.698 and 0.758

5.4.3 Evaluating the Mixture Model for Domain Prediction

For testing purposes, the maximum number of EM iterations to perform was raised to 500 to ensure convergence. Once the model was fit on the training data one final caveat was necessary to evaluate this clustering algorithm in a classification sense. Since clusters are mathematical constructs of data points, context vectors in this case, and not direct representations of domains, to obtain classification accuracy (recall), the best-performing permutation of possible cluster-assignments had to be found. This was done by expressing the confusion matrix across clusters as a cost matrix and obtaining the minimization over all permutations of possible row/column combinations [119], $4!$ when including other, $3!$ otherwise. This yields the permutation of the confusion matrix with the highest diagonal sum (total recall). Some sample images from the assigned clusters were drawn – the cluster resembling the sky domain was clearly noticeable, confusion between apron, runway and other exists, making the samples largely indistinguishable. The final results can be observed in Table 19.

Table 19: Final results of the application of unsupervised models, specifically a VBGMM, for domain prediction. It should be noted, that this obtained recall from permutations is the total recall, that is $TotalRecall = \frac{TP}{TP+FN}$ across the entire dataset, which is always higher (or at least equal) than average recall due to class-imbalance. In *SemanticAircraft* this is particularly relevant, since the largest number of patches stem from the easiest domain, sky. The deviation is again provided, since the test data is split across five separate folds.

Instances		Quadrants	
Including Other	Excluding Other	Including Other	Excluding Other
$\mathbf{0.586 \pm 0.048}$	$\mathbf{0.712 \pm 0.06}$	$\mathbf{0.539 \pm 0.029}$	$\mathbf{0.637 \pm 0.083}$

Having concluded the experiments using unsupervised algorithms, the next section will use the manually annotated GT domain-labels to train a supervised model: a CNN for domain prediction in the sense of image-classification.

Table 18: This table shows the the tuning and model-search process of the VBGMM.

	# - Components	Initialization	Covariance	Stand.	Whitening	γ	**Silhouette**
			Instances				
			Dirichlet distribution				
I	15	k-means	tied	F	F	10^3	0.468
II	—"—	—"—	—"—	T	—"—	2778	0.459
III	—"—	random	spherical	—"—	T	1	0.671
IV	4	k-means	tied	F	F	10^-5	0.446
V	—"—	—"—	—"—	T	—"—	1	0.398
VI	—"—	—"—	—"—	—"—	T	10^4	**0.702**
			Dirichlet process				
I	15	k-means	spherical	F	F	10^-5	0.468
II	—"—	random	tied	T	—"—	27503	0.457
III	—"—	—"—	spherical	—"—	T	27750	0.685
IV	4	k-means	—"—	F	F	10^-5	0.44
V	—"—	random	—"—	T	—"—	10^-5	0.397
VI	—"—	k-means	—"—	—"—	T	10^-5	0.698
			Quadrants				
			Dirichlet distribution				
I	15	random	tied	F	F	10^-5	0.66
II	—"—	k-means	—"—	T	—"—	278	0.66
III	—"—	random	—"—	—"—	T	10	0.778
IV	4	k-means	tied	F	F	10^-5	0.667
V	—"—	random	—"—	T	—"—	10^3	0.518
VI	—"—	k-means	—"—	—"—	T	10^5	0.744
			Dirichlet process				
I	15	random	tied	F	F	10^-5	0.653
II	—"—	k-means	—"—	T	—"—	10^-5	0.66
III	—"—	random	—"—	—"—	T	37000	0.765
IV	4	k-means	tied	F	F	10^-5	0.667
V	—"—	random	—"—	T	—"—	10^-5	0.514
VI	—"—	k-means	—"—	—"—	T	10^-5	**0.766**

5.5 Supervised Convolutional Neural Networks

For the CNN, not the set of context vectors $\mathcal{I}_{\mathbf{c}}$ and $\mathcal{Q}_{\mathbf{c}}$ are used as input data, but instead the images $\mathcal{I}_{\mathbf{RGB}}$ / $\mathcal{Q}_{\mathbf{RGB}}$ and corresponding class-labels $\mathcal{I}_{\mathrm{l}}$ / $\mathcal{Q}_{\mathrm{l}}$ obtained from domain annotation.

5.5.1 Convolutional Neural Networks for Classification

For implementation of the different models PyTorch was used, particularly TorchParallel. All models were loaded onto the graphical processing units (GPU) where training takes place. Three GTX-2080TIs with 12GB RAM each were available although the models are much smaller which allowed training in multiple configurations at a time, significantly speeding up the hyperparameter tuning process. While significantly dependent on input and model size, on average training one epoch took approximately two minutes for instances and ten minutes for quadrants. During phase-1, the model-tuning, validation took place after every training epoch. During final application in phase-2, a total of ten validation steps were done.

The loss function in all models is cross-entropy loss. Regarding parameter initialization, most notably the weights in convolutional layers, according to Dellinger *et al.* [120] for all deep neural networks using asymmetric non-linear activation functions, Kaiming [121] weight initialization performs better, in the sense that gradients do not vanish nearly as fast, than Xavier [122]. While symmetric activation functions such as $sigmoid(\cdot)$ and $tanh(\cdot)$ were commonly used in the beginning of CNNs, rectified linear unit (ReLU)-type functions are now the norm. All models trained in this work use ReLU activation functions and Kaiming-uniform weight initialization. Bias neurons are initialized with 0s while weights and biases in batch-normalization layers are initialized with 1s and 0s respectively [123].

Almost all models use adaptive average pooling right before the fully connected (FC) layer to stay agnostic to different input sizes. The pooled area increases with input size, resulting in constant output size after pooling. Input images are still required to be squared for most architectures to function properly. This is somewhat troublesome for the airplane instances $\mathcal{I}_{\mathbf{RGB}}$ due to their aspect ratio. Therefore, among other preprocessing steps, input patches were often resized using cubic interpolation (4×4 neighborhood).

On the topic of image augmentation, Shorten *et al.* [124] provide a survey of common techniques used for DL, Mikołajczyk *et al.* [125] highlight techniques particularly useful for image classification. The image augmentation library of Jung *et al.* [126] is used in this work for some augmentation techniques. As the next section will show, models tend to overfit on the training split of *SemanticAircraft*, resulting in suboptimal performance on the validation and test data. One method generally accepted to reduce overfitting is to use stronger augmentation techniques. This creates larger variance in the input data, making it harder for the model to fit its parameters perfectly. Three strengths of image augmentation were used in this work, each consisting of a number of different augmentation techniques:

Light image augmentation includes some of the most commonly used techniques

- HorizontalFlip(0.5), i.e. every patch has a fifty-percent chance of being flipped horizontally

- Resize($x \times x$), resizes the input patch to be $x \times x$, where x is a power of 2 and set to 256 for the experiments[7]

Medium strength augmentations include affine transformations

- Crop($\cdot$) to crop some patches mostly in width followed by
- HorizontalFlip(0.5)
- AdditiveGaussianNoise($scale = 0.05 * 255$) is used to add some noise followed by
- Affine($rotate = (-5, 5)$), a slight affine rotation in the range $[-5, 5]$

Strong augmentation includes all of the above mentioned techniques. In addition it was experimented with

- MotionBlur($\cdot$) to simulate fast moving objects
- LinearContrast($\cdot$) to change the contrast in images
- Translate($\cdot$) as a different affine transformation
- GaussianBlur($\cdot$) which adds blur following normal distributions
- Scale($\cdot$) as an alternative to Resize($\cdot$) in combination with
- Padding($\cdot$) to pad the image borders with set values and
- Dropout($\cdot$) of pixels with different levels of granularity

All augmentation techniques are applied only on the training set. Patches from the validation and test-splits are only resized to meet the input requirements. Data from any split is further standardized by substracting the per-channel mean from all pixels of the training set, followed by division with the standard deviation.

5.5.2 Searching and Tuning Applicable Neural Networks

While advanced techniques for visual explanations of deep networks exist [11], for hyperparameter tuning of CNNs it often suffices to observe the loss and accuracy recorded during training and validation. One such graph can be observed in Figure 11. The plot was obtained using one of the applied networks – the $\mathrm{ResNet50}$ [127] architecture – on fold-1 of instances.

Observing this plot in particular a few things are immediately noticeable: The model achieves almost 100% training accuracy during the last epoch, which is a strong indication that the model is unnecessarily powerful for the comparatively simple task. The model also overfits, and does so very early on, which can be observed from the gap between training and validation accuracy. The loss shows the convergence of the model, consistently dropping up until epoch 24. From epoch 30 to 31 a significant step in training accuracy can be observed, this coincides with the reduction of the learning rate at epoch 30. The optimizer does not get stuck in a local minima, or at least the minima it convergences to is not far above the global minima.

Following these results, multiple methods were employed to combat the noticeable overfitting:

[7] There is no requirement for patches to have power-of-2 width/height, but it does lead to faster performance due to the architectural design of the processing hardware.

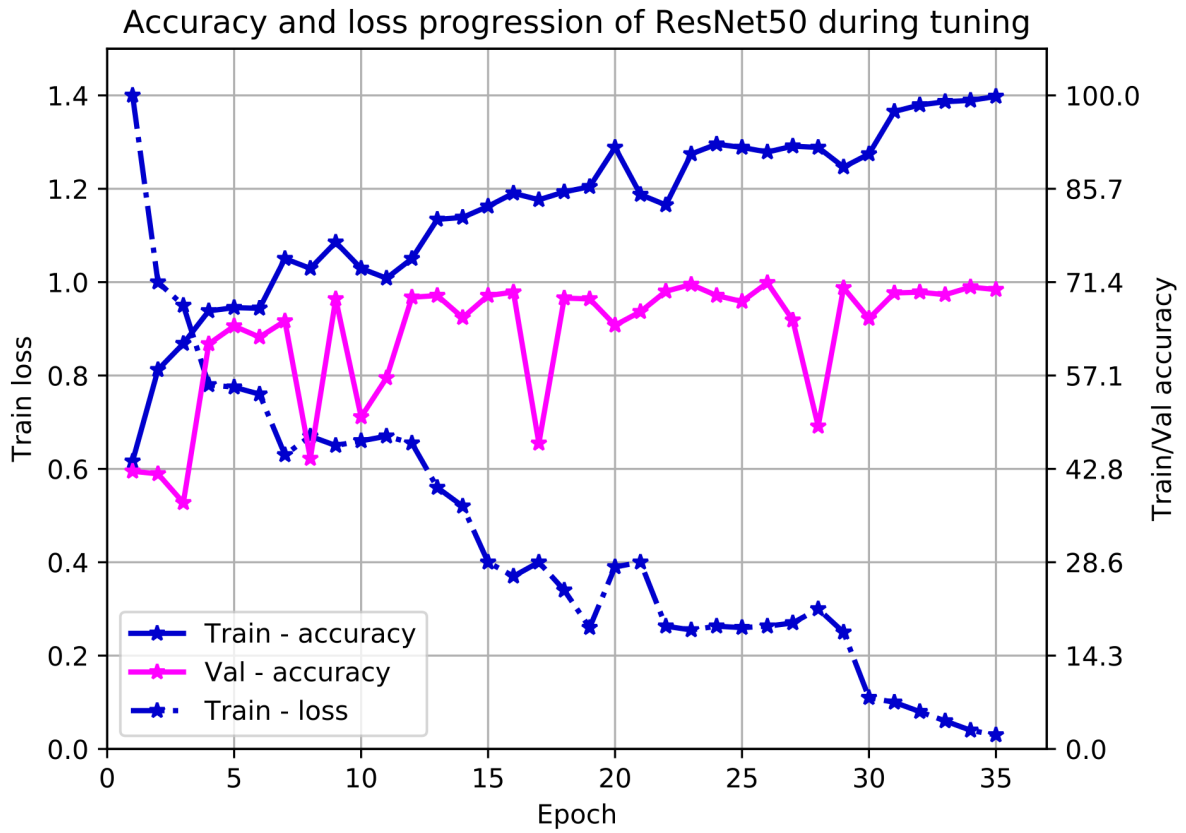


Figure 11: Plot showing the training and validation accuracy, as well as the training-loss during tuning of the ResNet50 model in configuration-I. A number of things can be observed at a glace, for example the fact that the model is overfitting on the training data.

Model-size

The model size was reduced by changing the architecture to that of ResNet34, ResNet18 and various Dilated Residual Networks (DRN) [128] and MobileNet [129] architectures. Even SqueezeNet [130] was briefly experimented with, although this did not converge. It could be observed, that simply exchanging the model and reducing the model size while keeping all other parameters fixed proved successful in alleviating overfitting.

Image-augmentation

Various strengths of image-augmentation were applied to create artificial variance in the input data. Stronger augmentations generally led to worse performance across all models in this experiment. While this may seem unintuitive, it is difficult to draw the line where augmentation of images becomes too strong.

Batch-size

Reducing the batch-size leads to an increased number of backpropagation steps over smaller data-subsets which makes it harder for parameters to overfit on the data points. While reducing batch-size does mean prolonged training time, it did provide slight performance improvements.

Dropout

A common regularization technique on an architectural level is to use dropout layers. None of the CNN architecture originally used dropout, so a layer had to be added to the best-performing (ResNet) models. Using dropout improved performance slightly.

Two different optimizers were employed for $\mathrm{ResNet50}$: Adam [131] and stochastic gradient descent (SGD) with Nesterov momentum [132] of 0.9. It was observed that they converge to the same minimum, although Adam does so significantly faster. Besides the augmentation strength, batch-size and dropout, the learning rate (LR) was also changed occasionally. The learning rate significantly impacts the learning process: if it is set too high, learning plateaus after a few epochs, while a rate set too low leads to very slow, if any, convergence. With the Adam optimizer the concept of learning rate works differently compared to other common optimizers [44]: Adam keeps a specific learning rate for every single model parameter which allows it to handle sparse gradients. Nevertheless, a global learning rate can be set which acts as the upper-bound for all local, parameter-specific learning rates. For decay of the learning rate, a simple step-decay was implemented that multiplies the rate by a factor of 0.1 every 30 epochs. The number of training epochs was 35 for almost all models, meaning one learning rate decay took place right before the end. Weight decay or L2-regularization was set to 0.0.

To obtain accuracy results, the total validation accuracies across the last five epochs and all four folds were averaged. Table 20 shows the process of finding the best performing model for both instances and quadrants of *SemanticAircraft* using the total recall as metric.

The best performing models on both instances and quadrants are $\mathrm{ResNet18}$ variants. For both models, only light image-augmentation yielded the best results. A key difference between the models is the presence of dropout for the quadrants model. A probability of 0.5 led to find the best performing model on quadrants. The plots recorded during training on the first of the phase-1 folds for both models respectively can be observed in Figure 12. The full collection of accuracy-loss plots can be found in Appendix C.

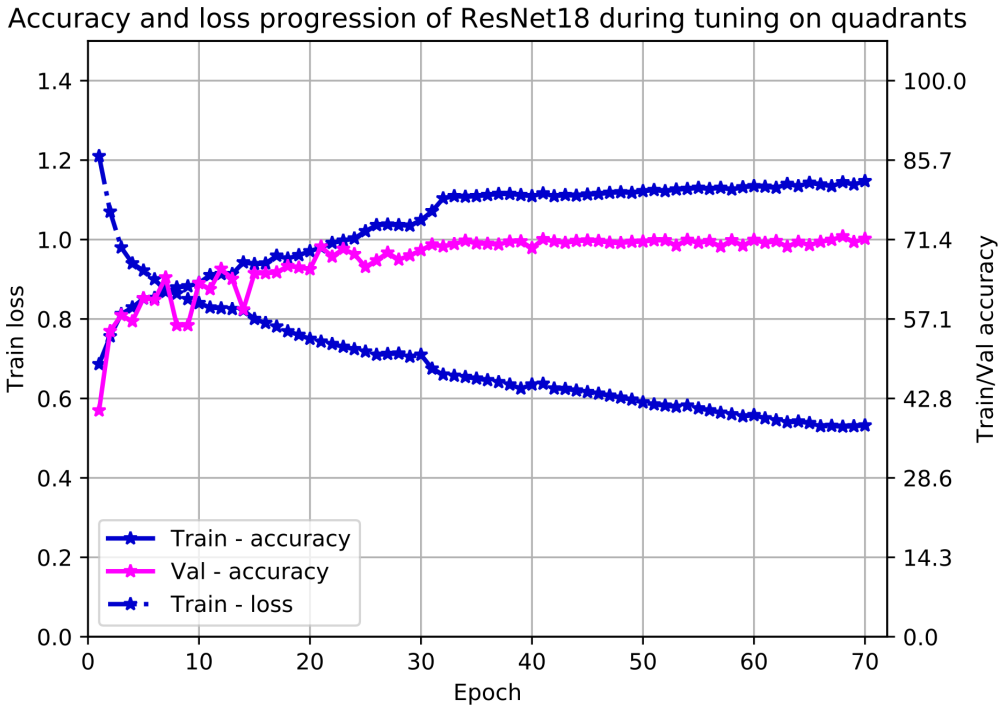
5.5.3 Evaluation of the ResNet18 Model

For the final evaluation of the $\mathrm{ResNet18}$ models in phase-2, the CNNs were trained with above mentioned parameters for 35 and 150 epochs on every fold of the instances and quadrants respectively. Once training concluded, the models were applied on the test data once and the per-class and total accuracy recorded and averaged across the folds. The results can be observed in Table 21. For more detailed per-class prediction results, Table 22 provides the four normalized confusion matrices.

Table 20: Parameter search of the supervised models. Multiple architectures, image-augmentation techniques and other hyperparameters were applied and tuned. To combat the problem of overfitting, reducing model size and adding a dropout layer proved successful. The last column gives the total recall for instances $\mathcal{I}$ and quadrants $\mathcal{Q}$ respectively.

	Augmentation	Batch-size	Dropout	LR	Epochs	**Total Recall**	
			ResNet50			$\mathcal{I}$	$\mathcal{Q}$
I	light	128	0.0	0.001	35	0.7	0.694
II	medium	—"—	—"—	—"—	—"—	0.68	0.67
III	light	—"—	—"—	0.0005	—"—	0.666	0.686
			ResNet18				
I	light	128	0.0	0.001	35	**0.728**	0.704
II	medium	—"—	—"—	—"—	—"—	0.695	0.677
III	light	—"—	—"—	0.0005	—"—	0.673	0.694
IV	—"—	16	—"—	0.001	—"—	0.73	0.685
V	—"—	—"—	0.2	—"—	—"—	0.702	0.679
VI	—"—	—"—	0.5	—"—	70	0.722	**0.718**
			ResNet34				
I	light	16	0.0	0.001	35	0.679	0.657
			MobileNetV2				
I	light	16	0.0	0.001	35	0.658	0.639
II	strong	—"—	—"—	—"—	—"—	0.528	0.552
			DRN-C26				
I	light	32	0.0	0.001	35	0.683	0.684
II	—"—	16	—"—	—"—	—"—	0.695	0.670
			DRN-D38				
I	light	16	0.0	0.001	35	0.682	0.680

(a) Training accuracy and loss as well as validation accuracy of the model best performing on instances for domain prediction, ResNet18 in configuration-I. While overfitting is still present, a strong validation accuracy of 72.8% could be achieved.



(b) Accuracy-loss plot for the model best performing on quadrants, ResNet18 in configuration-VI. Due to the manual inclusion of a dropout-layer, the model no longer achieves 100% training accuracy and overfitting was significantly reduced. A step in the loss is noticeable after adjusting the learning rate in epoch 30.

Figure 12: Accuracy-loss plots for the best performing models on instances and quadrants. Plots were captured during model-tuning.

Table 21: Final results of the ResNet18 models trained for domain prediction on *SemanticAircraft*. Notable is the significant increase across all metrics when excluding other images, as well as the difference of 3 to 5 percent between average and total accuracy, due to the many "easy" sky patches. Prediction of quadrants was slightly less accurate, presumably due to the annotation difficulties.

	Apron	Runway	Sky	Other	**Average**	Total
			Instances			
			Including Other			
Precision	0.6725	0.6158	0.93	0.535	0.6883	0.716
Recall	0.669 ± 0.046	0.622 ± 0.029	0.945 ± 0.035	0.517 ± 0.041	0.688 ± 0.038	0.716 ± 0.015
F_1	0.6708	0.6189	0.9374	0.5259	**0.6883**	0.716
			Excluding Other			
Precision	0.8335	0.6833	0.9694	—	0.8287	0.853
Recall	0.829 ± 0.045	0.69 ± 0.058	0.969 ± 0.019	—	0.829 ± 0.041	0.853 ± 0.011
F_1	0.8312	0.6865	0.9692	—	**0.829**	0.853
			Quadrants			
			Including Other			
Precision	0.6553	0.5421	0.8587	0.5358	0.648	0.692
Recall	0.556 ± 0.027	0.608 ± 0.017	0.893 ± 0.019	0.527 ± 0.033	0.646 ± 0.096	0.692 ± 0.013
F_1	0.6016	0.5732	0.8755	0.5314	**0.6454**	0.692
			Excluding Other			
Precision	0.8037	0.5584	0.9411	—	0.7677	0.778
Recall	0.711 ± 0.022	0.686 ± 0.013	0.938 ± 0.01	—	0.778 ± 0.015	0.778 ± 0.006
F_1	0.7545	0.6157	0.9396	—	**0.7699**	0.778

To summarize, the trained CNNs were capable of predicting domains with fairly high accuracy. The results are in line with previous expectations, showing mild confusion between apron and runway, strong performance on sky patches, and other images proving somewhat troublesome. The problem of model overfitting could be addressed to some extent be reducing model-size, batch-size and adding a dropout layer to the architectures. With hyperparameter tuning and application of the models concluded, they can now be directly compared against another.

Table 22: Confusion matrices following CNN-based domain prediction. Noticeable is the confusion between apron and other patches, particularly for the instances. Prediction of the sky domain was again the most accurate. When excluding other, the expected confusion between apron and runway comes to light.

	Including Other				Excluding Other			
Instances								
	Apron	Runway	Sky	Other	Apron	Runway	Sky	
Apron	0.669	0.14	0.005	0.186	0.829	0.162	0.008	1207
Runway	0.231	0.622	0.034	0.113	0.27	0.69	0.04	675
Sky	0.007	0.013	0.945	0.035	0.015	0.017	0.969	1210
Other	0.301	0.101	0.081	0.517				762
	1200	682	1230	737	1201	682	1209	
Quadrants								
Apron	0.556	0.193	0.047	0.205	0.711	0.209	0.08	2966
Runway	0.162	0.608	0.077	0.153	0.194	0.686	0.12	2351
Sky	0.015	0.04	0.893	0.052	0.025	0.037	0.938	5233
Other	0.149	0.158	0.165	0.527				2715
	2515	2638	5441	2671	2698	2425	5427	

6 Results and Discussion of Domain Prediction Models

For direct comparison of the three models, the total recall is used since this is the only metric obtained with the VBGMM. Table 23 shows this comparison.

Table 23: Total recall of all three models predicting domains of $\mathrm{airplane}$ instances and quadrants from *SemanticAircraft*. The CNN shows strong classification results and performs the best across all metrics except quadrants when excluding other samples. The second most accurate model has proven to be the threshold algorithm baseline followed by the unsupervised VBGMM.

	Instances		Quadrants	
	Including Other	Excluding Other	Including Other	Excluding Other
Baseline	0.588 ± 0.015	0.796 ± 0.011	0.639 ± 0.017	$\mathbf{0.799 \pm 0.006}$
VBGMM	0.586 ± 0.048	0.712 ± 0.06	0.539 ± 0.029	0.637 ± 0.083
ResNet18	$\mathbf{0.716 \pm 0.015}$	$\mathbf{0.854 \pm 0.011}$	$\mathbf{0.692 \pm 0.013}$	0.778 ± 0.006

While the CNN has given the best prediction performance, the requirement of annotations for the domains are a significant drawback, limiting its further application. For the baseline, the meticulous parameterization makes tuning and extension to other domains very difficult. Significant effort would be required to apply the model to other domains. While the VBGMM performs the worst in terms of prediction accuracy, the insights the model can provide into the dataset structure, hinting at possible other subdomains besides apron, runway, sky and other, are noteworthy. In short, all three models have benefits and drawbacks but for the explicit task of domain prediction, the supervised CNN performed the best. Both the baseline and CNN require domain annotation but are in turn the most accurate models. The highly parameterized nature of the baseline makes it both tedious to set-up and tune, and extension to new domains incorporating new (super-)classes is cumbersome. Nevertheless, it does perform well in this setting with a limited number of domains and superclasses. Unsupervised models' classification is the most variant, this is due to the general premise of GMMs but also the permutation taking place to allow accuracy interpretation. One possibility to improve clustering performance might be the usage of evidence ensembles, where clustering is based on evidence from the co-association matrix [133]. With quadrants the annotation procedure was the most troublesome, this is reflected in unsupervised model's performance, while CNNs are large enough to encompass these faults. The exclusion of any samples belonging to other does improve all model's performance, most significantly the baseline.

One final note about the unsupervised learning methods: the poor classification performance compared to the supervised CNNs was to be expected. The neural network is very capable of memorizing the feature representations from the images and using its trained parameters for classification of unseen samples and achieves high domain prediction accuracies in this way. The purpose of unsupervised algorithms goes beyond simple classification, since the models yield a cluster structure mathematically derived free of the constraints imposed via the limited set of domains, apron, runway, sky and other. The clusters might in fact not at all resemble the set of domains defined in this application. Individual samples within a cluster correlate in the sense that the context distributions are similar but the patterns found in the semantic context statistics might show a multitude of further domains [134], or perhaps the concept of visual domains is not appropriate to describe the clusters altogether. Therefore, the resulting clusters themselves are of more interest than achieved domain prediction accuracies. One straightforward way to gain a slight understanding of the kind of clusters created is to observe the images corresponding to the context vectors in every cluster. Having done so, almost all unsupervised models use one or two clusters to capture sky-like images with overwhelmingly sky context, which is a significant portion of the samples in *SemanticAircraft*. Other clusters feature images mixed across the domains apron and runway. This confusion can be a result of multiple factors and is most likely a combination thereof:

- The distinction is more noticeable in images due to differences in lighting, scene structure or other features not captured via context
- The set of labels in *SemanticAircraft* is incomplete and more distinct classes are necessary
- The transition between the two domains is so ambiguous it can only be learned with a CNN
- Apron vs. runway is not an appropriate distinction to make
- The domain annotation procedure itself used the images as basis and not the context statistics

Lastly, in all experiments with clustering algorithms, if the number of clusters was not specified as it was the case in some VBGMM experiments, more than three or four clusters were created. Finding the optimal (according to some criterion) number of clusters with unsupervised learning is a non-trivial task, using metrics such as silhouette-scores - the preliminary definition of domains made this a non-requirement. The results indicate, that the limitation to apron, runway and sky was perhaps too strict. In future work, further analysis with clustering algorithms could provide important insights.

To conclude the domain prediction procedure and provide visual results, Figures 13 and 14 lastly show a set of randomly selected, classified instances and quadrants from *SemanticAircraft* for every domain. Patches that were classified incorrectly by any of the three models are outlined with a red border.

Images from instances and quadrants look decidedly different: Samples from the former appear much more cluttered and tend to feature the airplane in the center, while quadrants all

(a) Apron instances: The bottom left corner shows an airplane in a rural environment where apron and runway coincide – all models were able to correctly predict the domain.

(b) Runway instances: Many images were confused and assigned to the apron domain instead.

(c) Sky instances: Perhaps a surprising amount of sky instances shown here were wrongly classified, many incorrect predictions made by the baseline.

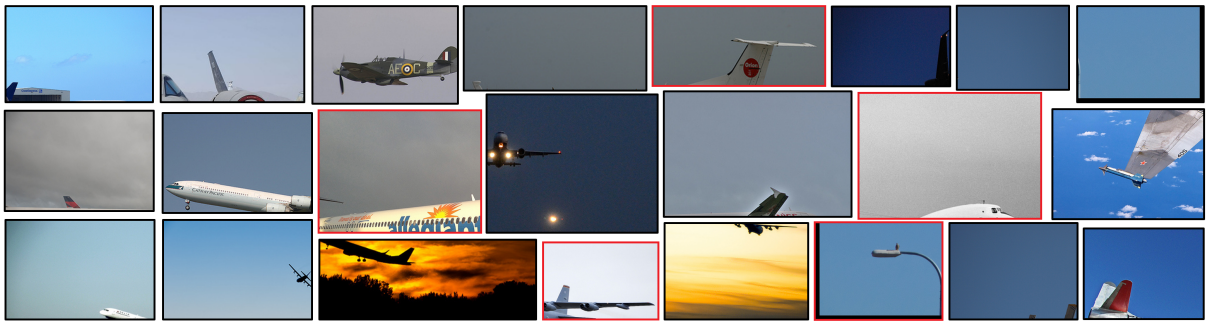
(d) Other instances: While some images show airplanes completely out-of-context, the fourth image from the left in the top row on the other hand shows a very canonical capture of an airplane, except the image is computer generated.

Figure 13: A randomly drawn selection of images showing instances of *SemanticAircraft*. Images that were wrongly classified by any of the Baseline, VBGMM or ResNet18 models are highlighted in red. Many runway patches were classified as apron instead.

provide a truncated, clear and neat scene image. Having successfully predicted domains for all airplanes in *SemanticAircraft* the experiments are concluded.

(a) Apron quadrants: In the lower row the second image from the left for example can only be assigned to the domain apron because a small sun-umbrella has been extended above the cockpit, signifying the fact that the airplane is not in motion.

(b) Runway quadrants: The confusion of the CNN between apron and runway patches was lower with quadrants than instances, and these images show perhaps why. Images are not as cluttered as apron quadrants, and instead often times provide a landscape-esque view of the scene with little airport context near the bottom.



(c) Sky quadrants: These images are exceedingly "clean" and clutter-free in comparison to other domains. It is interesting to note, that all three methods were able to correctly predict the image on the bottom row third from left, although the baseline and VBGMM are not aware of the unusual color scheme but only the context vectors.

(d) Other quadrants: It is difficult to make sense of other quadrants even for humans and therefore not surprising that the exclusion of other patches led to performance increase across all models and dataset-variations.

Figure 14: A randomly drawn selection of quadrants from $\mathcal{Q}_{\mathrm{RGB}}$ treated as individual images. Wrong domain predictions by any model are again highlighted in red. With quadrants the annotation procedure was exceedingly difficult. The level of image clutterness is significantly lower than with instances, across all domains.

7 Summary and Outlook

This work provided a broad overview for working with visual semantic context in CV. Many of the existing works learn or express semantic context only implicitly. Using the proposed semantic context module instead, context vectors can be extracted directly from semantically segmented masks. These context vectors were then employed for data distillation, filtering out unwanted samples during aggregation. The concept of domains for objects of interest as an extension of semantic context was introduced. As an application of semantic context, the task of domain prediction was formulated, specifically for images from the aerial domain, predicting the environment of airplanes in the merged dataset *SemanticAircraft*. Context was used in its explicit representation for domain prediction, using a heavily-parameterized threshold algorithm and unsupervised ML models. Images were further manually annotated with GT domain-labels allowing the application of CNNs for domain-prediction in the form of image-classification. Results show that all three models were capable of predicting domains with sufficient accuracy, the $\mathrm{ResNet18}$ CNN was able to achieve the best accuracy.

The entire procedure can be summarized briefly as follows:

1. Data aggregation and context extraction of airplane images from *ADE20K-SceneParsing*, *COCO-Stuff* and *PASCAL-Context*. Analyze the context statistics to identify important/distinct classes as well as similar classes that can be merged to superclasses.

2. Design *SemanticAircraft* deriving from *ADE*, *COCO* and *PASCAL* and separate the images into instances of airplanes as well as individual image-quadrants. Provide the corresponding masks to the semantic context module and filter highly void, indoor or inappropriately sized patches.

3. Manual domain annotation by assigning one of four domain labels (apron, runway, sky, other) to every instance and quadrant. Further create two variations of the dataset, one excluding other samples, and setup a 5-fold cross validation scheme.

4. Three separate models were used for the domain prediction task

    a) Baseline - an original algorithm performing prediction in the form of hierarchical thresholding over the context vectors. Despite decent classification accuracy, multiple shortcomings exist most notably the parameter-heavy nature of the model.

    b) Unsupervised algorithms - K-means, GMMs and VBGMMs were used for context-vector clustering as classification. The silhouette-coefficient was used to evaluate the models against one-another. Applying the VBGMM on the test set has yielded acceptable prediction performance, although the clusters created by the VBGMM themselves could provide further insights into context distributions.

c) Supervised CNN - Using the images and domain-annotation labels, various CNN architectures were trained and applied. While overfitting was very noticeable at the start, its effect was successfully reduced by lowering model size and implementing a dropout layer. Final application of the ResNet models yielded the best performance for all models, with prediction accuracy between 69 and 85 percent.

For the scope of this work, experiments concluded with the prediction of domains, showcasing one possible application of semantic context directly.

In future works, a number of things can be explored when working with visual semantic context. Perhaps a straightforward idea might be the application in other domains, such as indoor scenes [135], for helping improve seasonal consistency [136] of algorithms or estimating the time of day [137]. In a larger learning framework, the domain prediction results could be used to guide parameter-selection for models fine-tuned on specific domains. A closer integration into the aggregation procedure with early domain predictions could further help create a more balanced subdomain coverage in the target dataset.

For obtaining semantic context in a learning procedure, ideas similar to saliency maps might be employed. Masking objects of interests in images can lead CNNs to only learn the context, although it should be noted that priors are always bidirectional, i.e. the presence of context influences the objects and vice versa. Training two separate streams of deep neural networks for either context or object respectively, then combining the two in a Bayesian inference scheme might give insight what is truly learned for e.g. object detection. For synthetic image generation with GANs, context might be applied as a feature space in the adversary to possibly generate images with certain desirable domain-specific characteristics.

The clusters created with the unsupervised models could be further analyzed for deeper insights into the visual context of the datasets. Other probabilistic domain prediction procedures could be employed, working more closely on the context distributions – or using distributional reinterpretation of softmax scores to help model uncertainty more accurately. The possibility of expressing context in an even lower low-dimensional form as was presented here could allow the application of generative models.

Bibliography

[1] M. J. Choi, A. Torralba, and A. S. Willsky, “Context models and out-of-context objects,” *Pattern Recognition Letters*, vol. 33, no. 7, pp. 853–862, 2012.

[2] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[3] H. Caesar, J. Uijlings, and V. Ferrari, “Coco-stuff: Thing and stuff classes in context,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[4] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, “The role of context for object detection and semantic segmentation in the wild,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 891–898.

[5] X. Chen, “Context driven scene understanding,” Ph.D. dissertation, University of Maryland, 2015.

[6] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, and W. Xu, “Cnn-rnn: A unified framework for multi-label image classification,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2285–2294.

[7] T. Chen, M. Xu, X. Hui, H. Wu, and L. Lin, “Learning semantic-specific graph representation for multi-label image recognition,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 522–531.

[8] H. Zhang, K. Dana, J. Shi, Z. Zhang, X. Wang, A. Tyagi, and A. Agrawal, “Context encoding for semantic segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7151–7160.

[9] M. Wang and W. Deng, “Deep visual domain adaptation: A survey,” *Neurocomputing*, vol. 312, pp. 135–153, 2018.

[10] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, “On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation,” *PloS one*, vol. 10, no. 7, 2015.

[11] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” *International Journal of Computer Vision*, vol. 128, no. 2, p. 336–359, Oct 2019. [Online]. Available: http://dx.doi.org/10.1007/s11263-019-01228-7

[12] S. Geman, E. Bienenstock, and R. Doursat, “Neural networks and the bias/variance dilemma,” *Neural computation*, vol. 4, no. 1, pp. 1–58, 1992.

[13] A. G. Kendall, “Geometry and uncertainty in deep learning for computer vision,” Ph.D. dissertation, University of Cambridge, 2019.

[14] L.-J. Li, R. Socher, and L. Fei-Fei, “Towards total scene understanding: Classification, annotation and segmentation in an automatic framework,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 2036–2043.

[15] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun, “Unified perceptual parsing for scene understanding,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 418–434.

[16] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, “Panoptic segmentation,” 2018.

[17] R. Montague, “English as a formal language, chapter linguaggi nella societa e nella tecnica, b,” *Visentini et al eds*, pp. 189–224, 1970.

[18] ——, “Universal grammar,” *Theoria*, vol. 36, no. 3, pp. 373–398, 1970.

[19] ——, “The proper treatment of quantification in ordinary english,” in *Approaches to natural language*. Springer, 1973, pp. 221–242.

[20] B. Murphy, P. Talukdar, and T. Mitchell, “Learning effective and interpretable semantic models using non-negative sparse embedding,” in *Proceedings of COLING 2012*. Mumbai, India: The COLING 2012 Organizing Committee, dec 2012, [Online]. Available: https://www.aclweb.org/anthology/C12-1118 [Accessed: 22.03.2020].

[21] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. A. Ranzato, and T. Mikolov, “Devise: A deep visual-semantic embedding model,” in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, [Online]. Available: http://papers.nips.cc/paper/5204-devise-a-deep-visual-semantic-embedding-model.pdf [Accessed: 22.03.2020].

[22] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.

[23] 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*. Ieee, 2009, pp. 248–255.

[24] M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G. S. Corrado, and J. Dean, “Zero-shot learning by convex combination of semantic embeddings,” *arXiv preprint arXiv:1312.5650*, 2013.

[25] R. Kiros, R. Salakhutdinov, and R. S. Zemel, “Unifying visual-semantic embeddings with multimodal neural language models,” *CoRR*, vol. abs/1411.2539, 2014, [Online]. Available: http://arxiv.org/abs/1411.2539 [Accessed: 22.03.2020].

[26] M. Ren, R. Kiros, and R. Zemel, “Image question answering: A visual semantic embedding model and a new dataset,” *Proc. Advances in Neural Inf. Process. Syst*, vol. 1, no. 2, 2015.

[27] Z. Ren, H. Jin, Z. Lin, C. Fang, and A. Yuille, “Multi-instance visual-semantic embedding,” *arXiv preprint arXiv:1512.06963*, 2015.

[28] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[29] W. Liu, T. Mei, Y. Zhang, C. Che, and J. Luo, “Multi-task deep visual-semantic embedding for video thumbnail selection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[30] Y. Pan, T. Mei, T. Yao, H. Li, and Y. Rui, “Jointly modeling embedding and translation to bridge video and language,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[31] Y. Cao, M. Long, J. Wang, Q. Yang, and P. S. Yu, “Deep visual-semantic hashing for cross-modal retrieval,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016.

[32] Z. Niu, M. Zhou, L. Wang, X. Gao, and G. Hua, “Hierarchical multimodal lstm for dense visual-semantic embedding,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[33] R. Pérez-Arnal, A. Vilalta, D. Garcia-Gasulla, U. Cortés, E. Ayguadé, and J. Labarta, “A visual distance for wordnet,” 2018.

[34] L. Rabiner and B. Juang, “An introduction to hidden markov models,” *ieee assp magazine*, vol. 3, no. 1, pp. 4–16, 1986.

[35] M. Richardson and P. Domingos, “Markov logic networks,” *Machine learning*, vol. 62, no. 1-2, pp. 107–136, 2006.

[36] D. Steininger and C. Beleznai, “Semantic labeling enhanced by a spatial context prior,” *fh-ooe*, 2016.

[37] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, “Sun database: Large-scale scene recognition from abbey to zoo,” in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 3485–3492.

[38] A. Oliva and A. Torralba, “Modeling the shape of the scene: A holistic representation of the spatial envelope,” *International journal of computer vision*, vol. 42, no. 3, pp. 145–175, 2001.

[39] Z. Fu, T. Xiang, E. Kodirov, and S. Gong, “Zero-shot object recognition by semantic manifold distance,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

[40] T. Cover and P. Hart, “Nearest neighbor pattern classification,” *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.

[41] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (voc) challenge,” *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[42] J. Tighe and S. Lazebnik, “Superparsing: scalable nonparametric image parsing with superpixels,” in *European conference on computer vision*. Springer, 2010, pp. 352–365.

[43] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu, “Semantic segmentation with second-order pooling,” in *European Conference on Computer Vision*. Springer, 2012, pp. 430–443.

[44] A. Amidi and S. Amidi. (2018) Stanford-cs-230-deep-learning. [Online]. Available: https://github.com/afshinea/stanford-cs-230-deep-learning/blob/master/en/super-cheatsheet-deep-learning.pdf [Accessed: 16.12.2019].

[45] C. Doersch, A. Gupta, and A. A. Efros, “Unsupervised visual representation learning by context prediction,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[46] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, “Context encoders: Feature learning by inpainting,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[47] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng, “Nus-wide: a real-world web image database from national university of singapore,” in *Proceedings of the ACM international conference on image and video retrieval*, 2009, pp. 1–9.

[48] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes challenge 2007 (voc2007) results,” 2007.

[49] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[50] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[51] J. Fu, J. Liu, Y. Wang, Y. Li, Y. Bao, J. Tang, and H. Lu, “Adaptive context network for scene parsing,” in *Proceedings of the IEEE international conference on computer vision*, 2019.

[52] K. Weiss, T. M. Khoshgoftaar, and D. Wang, “A survey of transfer learning,” *Journal of Big data*, vol. 3, no. 1, p. 9, 2016.

[53] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, 2009.

[54] G. Csurka, “Domain adaptation for visual applications: A comprehensive survey,” *arXiv preprint arXiv:1702.05374*, 2017.

[55] S. Shah, D. Dey, C. Lovett, and A. Kapoor, “Airsim: High-fidelity visual and physical simulation for autonomous vehicles,” in *Field and service robotics*. Springer, 2018.

[56] M. Mancini, S. R. Bulò, B. Caputo, and E. Ricci, “Adagraph: Unifying predictive and continuous domain adaptation through graphs,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[57] G. Wilson and D. J. Cook, “A survey of unsupervised deep domain adaptation,” *arXiv preprint arXiv:1812.02849*, 2019.

[58] Y. Ganin and V. Lempitsky, “Unsupervised domain adaptation by backpropagation,” *arXiv preprint arXiv:1409.7495*, 2014.

[59] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, “A theory of learning from different domains,” *Machine learning*, vol. 79, no. 1-2, 2010.

[60] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, “Simultaneous deep transfer across domains and tasks,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[61] B. Bolländer. (2019) Deep domain adaptation in computer vision. [Online]. Available: https://towardsdatascience.com/deep-domain-adaptation-in-computer-vision-8da398d3167f [Accessed: 18.11.2019].

[62] Y.-H. Chen, W.-Y. Chen, Y.-T. Chen, B.-C. Tsai, Y.-C. Frank Wang, and M. Sun, “No more discrimination: Cross city adaptation of road scene segmenters,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[63] C. Sakaridis, D. Dai, and L. V. Gool, “Guided curriculum model adaptation and uncertainty-aware evaluation for semantic nighttime image segmentation,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[64] Y. Zhang, P. David, and B. Gong, “Curriculum domain adaptation for semantic segmentation of urban scenes,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[65] Y. Zhang, P. David, H. Foroosh, and B. Gong, “A curriculum domain adaptation approach to the semantic segmentation of urban scenes,” *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[66] I. Sikirić, K. Brkić, P. Bevandić, I. Krešo, J. Krapac, and S. Šegvić, “Traffic scene classification on a representation budget,” *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[67] O. Franěk. (2003) Apron sever - výstavba terminálu, červen 2003. [Online]. Available: http://gallery.vacc-cz.org/album04/abz [Accessed: 19.03.2020].

[68] A. Hunt. (2015) Boeing everett factory. [Online]. Available: http://www.crash-aerien.news/forum/les-avions-oublies-qui-pourrissent-sur-un-bout-de-tarmac-t2387-705.html [Accessed: 19.03.2020].

[69] S. Ahmed. (2014) List of international / domestic airports in india – general awareness (ga) for bank exams. [Online]. Available: https://allindiaroundup.com/general/international-domestic-airports-in-india-general-awareness/ [Accessed: 18.03.2020].

[70] png cloud. (2019) Person walking towards airplane on airport. [Online]. Available: https://png.cloud/f/person-walking-towards-airplane-on-airport/8ff499d8bbda49e381c2-201907080619.html [Accessed: 18.03.2020].

[71] King 5. (2019) Boeing max 737 planes parked on airport apron in moses lake. [Online]. Available: https://www.youtube.com/watch?v=QrMf6ys3yj4 [Accessed: 19.03.2020].

[72] Induced. (2019) Stansted airport stn. [Online]. Available: http://induced.info/?s=Stansted+Airport+STN [Accessed: 18.03.2020].

[73] Tawsif Salam. (2015) Qatar airways airbus a380-800 at heathrow airport terminal 4 before flying to doha, 6 jan 2015. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/0/07/Qatar_Airways_Airbus_A380-800_at_Heathrow_Airport_Terminal_4_before_Flying_to_Doha%2C_6_Jan_2015.jpg [Accessed: 18.03.2020].

[74] Rtitb airside. (2018) How to decrease aircraft turnaround times. [Online]. Available: https://www.rtitb-airside.com/wp-content/uploads/2018/11/How-to-Decrease-Aircraft-Turnaround-Times-1024x683.jpg [Accessed: 18.03.2020].

[75] O. Zendel, K. Honauer, M. Murschitz, D. Steininger, and G. Fernandez Dominguez, "Wilddash-creating hazard-aware benchmarks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 402–416.

[76] R. M. Haralick, K. Shanmugam, and I. H. Dinstein, "Textural features for image classification," *IEEE Transactions on systems, man, and cybernetics*, no. 6, pp. 610–621, 1973.

[77] D. Lu and Q. Weng, "A survey of image classification methods and techniques for improving classification performance," *International journal of Remote sensing*, vol. 28, no. 5, pp. 823–870, 2007.

[78] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *European conference on computer vision*. Springer, 2010, pp. 143–156.

[79] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[80] C. P. Papageorgiou, M. Oren, and T. Poggio, “A general framework for object detection,” in *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*. IEEE, 1998, pp. 555–562.

[81] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[82] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[83] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[84] H. Noh, S. Hong, and B. Han, “Learning deconvolution network for semantic segmentation,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1520–1528.

[85] L. Huang, J. Peng, R. Zhang, G. Li, and L. Lin, “Learning deep representations for semantic image parsing: a comprehensive overview,” *Frontiers of Computer Science*, vol. 12, 08 2018.

[86] X. Huang, X. Cheng, Q. Geng, B. Cao, D. Zhou, P. Wang, Y. Lin, and R. Yang, “The apolloscape dataset for autonomous driving,” *arXiv: 1803.06184*, 2018.

[87] J. Geyer, Y. Kassahun, M. Mahmudi, X. Ricou, R. Durgesh, A. S. Chung, L. Hauswald, V. H. Pham, M. Mühlegg, S. Dorn, T. Fernandez, M. Jänicke, S. Mirashi, C. Savani, M. Sturm, O. Vorobiov, M. Oelker, S. Garreis, and P. Schuberth, “A2D2: Audi Autonomous Driving Dataset,” Audi, Tech. Rep., 2020, [Online]. Available: https://www.a2d2.audi [Accessed: 07.09.2020].

[88] 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,” 2018.

[89] 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,” in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[90] G. Varma, A. Subramanian, A. Namboodiri, M. Chandraker, and C. Jawahar, “Idd: A dataset for exploring problems of autonomous navigation in unconstrained environments,” in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019, pp. 1743–1751.

[91] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3354–3361.

[92] G. Neuhold, T. Ollmann, S. Rota Bulo, and P. Kontschieder, "The mapillary vistas dataset for semantic understanding of street scenes," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4990–4999.

[93] O. Zendel, M. Murschitz, M. Zeilinger, D. Steininger, S. Abbasi, and C. Beleznai, "Railsem19: A dataset for semantic rail scene understanding," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[94] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," CS, University of Toronto, Tech. Rep., 2009.

[95] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba, "Semantic understanding of scenes through the ade20k dataset," *International Journal of Computer Vision*, vol. 127, no. 3, pp. 302–321, 2016.

[96] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ade20k dataset," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 633–641.

[97] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, "The role of context for object detection and semantic segmentation in the wild," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[98] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, T. Duerig *et al.*, "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," *arXiv preprint arXiv:1811.00982*, 2018.

[99] Z. Liu, Z. Miao, X. Zhan, J. Wang, B. Gong, and S. X. Yu, "Large-scale long-tailed recognition in an open world," 2019.

[100] R. Tudor Ionescu, B. Alexe, M. Leordeanu, M. Popescu, D. P. Papadopoulos, and V. Ferrari, "How hard can it be? estimating the difficulty of visual search in an image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2157–2166.

[101] EHQ Production. (2018) Airfield planes aerial. [Online]. Available: https://my.melbourneairport.com/taxiway-zulu [Accessed: 19.03.2020].

[102] B. Boettger. (2016) Fresh asphalt paving on the taxiways and tie-down area of the kenai municipal airport sits ready for use. [Online]. Available: https://www.alaskajournal.com/2016-10-02/kenai-airport-taxiway-renovation-now-complete [Accessed: 19.03.2020].

[103] Nagpuru Updates. (2020) Amravati-mum scheduled flights from next year. [Online]. Available: https://nagpurupdates.com/amravati-mum-scheduled-flights-from-next-year/ [Accessed: 19.03.2020].

[104] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[105] S. Lloyd, “Least squares quantization in pcm,” *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[106] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, “Variational inference: A review for statisticians,” *Journal of the American Statistical Association*, vol. 112, no. 518, p. 859–877, Feb 2017. [Online]. Available: http://dx.doi.org/10.1080/01621459.2017.1285773

[107] F. Hirschberger, D. Forster, and J. Lücke, “Large scale clustering with variational em for gaussian mixture models,” 2018.

[108] M. Titsias, “Variational learning of inducing variables in sparse gaussian processes,” in *Artificial Intelligence and Statistics*, 2009, pp. 567–574.

[109] M. Titsias and N. D. Lawrence, “Bayesian gaussian process latent variable model,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 844–851.

[110] A. Damianou and N. Lawrence, “Deep gaussian processes,” in *Artificial Intelligence and Statistics*, 2013, pp. 207–215.

[111] K. El-Arini. (2008) Dirichlet processes - a gentle tutorial. [Online]. Available: https://www.cs.cmu.edu/~./kbe/dp{_}tutorial.pdf [Accessed: 20.04.2020].

[112] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, “Scikit-learn: Machine learning in python,” *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[113] P. J. Rousseeuw, “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis,” *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[114] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009, vol. 344.

[115] S. says Reinstate Monica (https://stats.stackexchange.com/users/22311/sycorax-says-reinstate-monica). Why is euclidean distance not a good metric in high dimensions? Cross Validated. [Online]. Available: https://stats.stackexchange.com/q/99191 [Accessed: 20.04.2020].

[116] user11852 (https://stats.stackexchange.com/users/11852/us%ce%b5r11852). Are there cases where pca is more suitable than t-sne? Cross Validated. [Online]. Available: https://stats.stackexchange.com/q/249520 [Accessed: 20.04.2020].

[117] D. Arthur and S. Vassilvitskii, “k-means++: The advantages of careful seeding,” Stanford, Tech. Rep., 2006.

[118] J. H. Friedman, “Exploratory projection pursuit,” *Journal of the American statistical association*, vol. 82, no. 397, pp. 249–266, 1987.

[119] S. Morbieu. (2019) Accuracy: from classification to clustering evaluation. [Online]. Available: https://smorbieu.gitlab.io/accuracy-from-classification-to-clustering-evaluation/ [Accessed: 29.05.2020].

[120] J. Dellinger. Weight initialization in neural networks: A journey from the basics to kaiming. [Online]. Available: https://towardsdatascience.com/weight-initialization-in-neural-networks-a-journey-from-the-basics-to-kaiming-954fb9b47c79 [Accessed: 16.12.2019].

[121] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[122] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[123] F.-F. Li, R. Krishna, and X. Danfei. (2020) Cs231n: Convolutional neural networks for visual recognition. [Online]. Available: https://cs231n.github.io/neural-networks-2/ [Accessed: 29.05.2020].

[124] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, 2019.

[125] A. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," in *2018 international interdisciplinary PhD workshop (IIPhDW)*. IEEE, 2018, pp. 117–122.

[126] A. B. Jung, K. Wada, J. Crall, S. Tanaka, J. Graving, C. Reinders, S. Yadav, J. Banerjee, G. Vecsei, A. Kraft, Z. Rui, J. Borovec, C. Vallentin, S. Zhydenko, K. Pfeiffer, B. Cook, I. Fernández, F.-M. De Rainville, C.-H. Weng, A. Ayala-Acevedo, R. Meudec, M. Laporte *et al.* (2020) ImgAug. [Online]. Available: https://github.com/aleju/imgaug [Accessed: 25.03.2020].

[127] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[128] F. Yu, V. Koltun, and T. Funkhouser, "Dilated residual networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 472–480.

[129] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[130] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[131] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[132] N. Qian, “On the momentum term in gradient descent learning algorithms,” *Neural networks*, vol. 12, no. 1, pp. 145–151, 1999.

[133] C. Zhong, L. Hu, X. Yue, T. Luo, Q. Fu, and H. Xu, “Ensemble clustering based on evidence extracted from the co-association matrix,” *Pattern Recognition*, vol. 92, pp. 93–106, 2019.

[134] H. Alashwal, M. El Halaby, J. J. Crouse, A. Abdalla, and A. A. Moustafa, “The application of unsupervised clustering methods to alzheimer’s disease,” *Frontiers in computational neuroscience*, vol. 13, p. 31, 2019.

[135] J. Yang, J. Xu, K. Li, Y.-K. Lai, H. Yue, J. Lu, H. Wu, and Y. Liu, “Learning to reconstruct and understand indoor scenes from sparse views,” 2019.

[136] M. Larsson, E. Stenborg, L. Hammarstrand, T. Sattler, M. Pollefeys, and F. Kahl, “A cross-season correspondence dataset for robust semantic segmentation,” 2019.

[137] L. Sun, K. Wang, K. Yang, and K. Xiang, “See clearer at night: Towards robust nighttime semantic segmentation through day-night image conversion,” 2019.

List of Figures

List of Tables

List of Algorithms

List of Abbreviations and Acronyms

ADAS advanced driver assistance systems

AI artificial intelligence

AMP absorbing Markov-chain process

BBox bounding box

BIC Bayesian information criterion

CNN convolutional neural networks

CRF conditional random field

CV computer vision

DDA deep domain adaptation

DA domain adaptation

DGP deep Gaussian processes

DL deep learning

DRN Dilated Residual Networks

DRCN deep reconstruction classification networks

EM expectancy maximization

FC fully connected

FN false negative

FP false positive

GAN generative adversarial networks

GMM Gaussian mixture models

GP Gaussian processes

GPU graphical processing units

GT ground truth

IoU intersection over union

KL Kullback-Leibler

LSTM long short-term memory

LVM latent variable models

MAP maximum a posteriori

MMD maximum mean discrepancy

MRF Markov random fields

ML machine learning

NNSE non-negative sparse embedding

NLP natural language processing

PCA principal component analysis

PDA predictive domain adaptation

RBF radial basis function

ReLU rectified linear unit

RKHS reproducing kernel Hilbert space

RNN recurrent neural network

RoI regions of interest

SGD stochastic gradient descent

SVD singular value decomposition

SVM support vector machines

TL transfer learning

TOL Take-off or landing

TP true positive

t-SNE T-distributed stochastic neighborhood embedding

UAV unmanned aerial vehicles

UIoU uncertainty-aware intersection over union

VBGMM variational Bayesian gaussian mixture models

VI Variational inference

VNN Von-Neumann neighborhood

ZSL zero shot learning

Appendices

A Mathematical Notation

The mathematical content in this work is kept to the minimum necessary to achieve an understanding in the field of semantic context. Nevertheless, an understanding of the relevant concepts from ML, DL and CV is essential.

Scalar variables are denoted with a single letter such as x, while vectors are given in bold lower case such as $\mathbf{x}$ and assumed to be column vectors. Upper case bold letters are used for matrices $\mathbf{X}$ and calligraphy style $\mathcal{X}$ is used for the concept of datasets or other higher dimensional spaces. The notation (x, y) denotes a row vector of size 1×2 with entries being the scalar x and y respectively, while $\mathbf{x} = (x, y)^{\mathrm{T}}$ is the corresponding column vector. The notation $[x,\, y]$ is the interval from x to y including x and y themselves. The variance is given by $\widehat{\mathrm{var}}(\mathbf{x})$ while the covariance and correlation are given by $\widehat{\mathrm{cov}}(\mathcal{X})$ and $\widehat{\mathrm{cor}}(\mathcal{X})$ respectively.

B Context and Neighborhood Statistics

This appendix provides the complete dataset-wide context and neighborhood statistics obtained with the semantic context module in full detail.

B.1 *ADE20K - SceneParsing*

B.1.1 Images

Semantic Context

Context across the entire image
sky: 40.42, runway: 19.41, sea: 7.43, building: 6.40, grass: 3.84, wall: 3.09, floor: 3.08, ceiling: 2.43, earth: 2.06, tree: 1.91, person: 1.66, road: 1.12, mountain: 0.92, field: 0.91, windowpane: 0.90, car: 0.58, skyscraper: 0.49, truck: 0.49, hill: 0.45, sand: 0.36, escalator: 0.32, fence: 0.31, land: 0.20, case: 0.13, pole: 0.13, ship: 0.08, seat: 0.07, signboard: 0.06, water: 0.06, conveyor_belt: 0.06, stairway: 0.06, plant: 0.05, river: 0.05, streetlight: 0.04, crt_screen: 0.04, bus: 0.04, van: 0.03, column: 0.03, box: 0.03, bulletin_board: 0.03, house: 0.03, door: 0.02, tower: 0.02, painting: 0.02, stairs: 0.02, railing: 0.02, cabinet: 0.02, step: 0.02, path: 0.02, sculpture: 0.01, other: 0.03
images with no context: 0/142

Context in specific image quadrants
quadrant-I:
sky: 65.79, building: 7.14, sea: 5.87, ceiling: 4.38, wall: 4.09, runway: 2.83, tree: 2.56, windowpane: 1.41, earth: 0.92, mountain: 0.87, hill: 0.62, grass: 0.59, person: 0.55, escalator: 0.35, skyscraper: 0.33, field: 0.26, car: 0.26, truck: 0.24, column: 0.13, pole: 0.13, bulletin_board: 0.11, signboard: 0.09, streetlight: 0.09, floor: 0.06, water: 0.06, path: 0.05, tower: 0.04, road: 0.03, land: 0.02, railing: 0.02, bannister: 0.02, van: 0.01, stairway: 0.01, stairs: 0.01, door: 0.01, fence: 0.01, conveyor_belt: 0.01
quadrant-II:
sky: 66.12, sea: 5.81, building: 5.66, wall: 5.15, ceiling: 4.49, runway: 3.18, tree: 2.36, mountain: 1.05, person: 0.96, earth: 0.86, hill: 0.80, windowpane: 0.77, grass: 0.54, car: 0.49, skyscraper: 0.37, field: 0.25, road: 0.17, truck: 0.14, pole: 0.10, ship: 0.10, water: 0.09, escalator: 0.08, box: 0.07, door: 0.07, signboard: 0.06, plant: 0.06, conveyor_belt: 0.06, streetlight: 0.04, fence: 0.02, land: 0.02, floor: 0.02, path: 0.01, stairway: 0.01, pier: 0.01, sculpture: 0.01
quadrant-III:
runway: 37.04, sea: 9.63, grass: 8.32, sky: 7.06, building: 7.02, floor: 6.17, person: 5.54,

earth: 3.40, road: 2.31, wall: 1.82, field: 1.48, tree: 1.33, truck: 0.95, car: 0.87, sand: 0.73, seat: 0.72, fence: 0.66, skyscraper: 0.64, mountain: 0.50, escalator: 0.49, windowpane: 0.40, ceiling: 0.37, case: 0.37, land: 0.32, crt_screen: 0.32, stairway: 0.25, pole: 0.15, river: 0.15, bus: 0.14, plant: 0.13, hill: 0.12, ship: 0.11, house: 0.08, conveyor_belt: 0.07, painting: 0.07, sculpture: 0.05, stairs: 0.04, van: 0.04, railing: 0.03, bulletin_board: 0.03, signboard: 0.03, box: 0.01, door: 0.01, cabinet: 0.01, water: 0.01, streetlight: 0.01
quadrant-IV:
runway: 37.33, sea: 10.03, grass: 8.27, sky: 7.48, building: 7.30, floor: 6.52, earth: 3.46, road: 3.13, person: 2.64, wall: 1.68, field: 1.63, mountain: 1.30, car: 1.22, tree: 1.14, windowpane: 1.13, truck: 0.75, sand: 0.70, skyscraper: 0.66, land: 0.65, fence: 0.62, ceiling: 0.43, escalator: 0.34, pole: 0.20, river: 0.17, ship: 0.13, conveyor_belt: 0.12, van: 0.10, box: 0.10, crt_screen: 0.10, signboard: 0.07, water: 0.07, streetlight: 0.06, tower: 0.05, cabinet: 0.04, boat: 0.04, house: 0.04, step: 0.04, door: 0.04, railing: 0.04, stairway: 0.03, bannister: 0.03, stairs: 0.02, bus: 0.02, bag: 0.02, plant: 0.01, trade_name: 0.01, seat: 0.01, pot: 0.01
image quadrants with no context:
quadrant-I: 0/142
quadrant-II: 0/142
quadrant-III: 4/142
quadrant-IV: 6/142

Label-neighborhood

left:
sky: 28.82, runway: 15.70, building: 15.24, sea: 6.48, wall: 5.73, grass: 4.48, tree: 2.68, earth: 2.53, mountain: 2.51, person: 2.21, floor: 1.77, windowpane: 1.75, ceiling: 1.58, road: 0.94, hill: 0.77, car: 0.73, skyscraper: 0.60, stairway: 0.59, door: 0.54, field: 0.52, truck: 0.42, stairs: 0.33, sand: 0.28, pole: 0.26, land: 0.22, fence: 0.17, conveyor_belt: 0.13, box: 0.13, streetlight: 0.12, water: 0.09, path: 0.07, bus: 0.06, trade_name: 0.03, flag: 0.03, signboard: 0.03, van: 0.02
up:
sky: 31.27, building: 21.66, runway: 10.20, wall: 6.73, sea: 6.31, tree: 4.28, mountain: 3.48, grass: 3.43, ceiling: 2.14, earth: 2.01, windowpane: 1.53, hill: 0.75, floor: 0.74, land: 0.67, field: 0.55, truck: 0.55, road: 0.53, person: 0.49, skyscraper: 0.43, door: 0.12, stairway: 0.08, plant: 0.08, trade_name: 0.07, stairs: 0.07, box: 0.06, pole: 0.06, fence: 0.05, car: 0.05, sand: 0.04, water: 0.04, path: 0.02, flag: 0.01, crt_screen: 0.01, van: 0.01, bus: 0.01, streetlight: 0.01, tower: 0.01
right:
sky: 29.15, runway: 16.34, building: 15.61, sea: 7.12, wall: 5.69, grass: 4.43, tree: 2.83, earth: 2.53, mountain: 2.16, person: 1.80, floor: 1.71, windowpane: 1.65, ceiling: 1.57, road: 0.95, hill: 0.72, door: 0.48, car: 0.46, field: 0.42, truck: 0.40, land: 0.36, stairs: 0.31, skyscraper: 0.29, pole: 0.28, sand: 0.25, stairway: 0.25, box: 0.18, crt_screen: 0.11, water: 0.11, van: 0.08, streetlight: 0.08, path: 0.06, bus: 0.05, plant: 0.04, flag: 0.03, trade_name: 0.03, fence:

0.02, signboard: 0.01
down:
runway: 35.46, sky: 15.30, building: 8.63, sea: 7.40, grass: 6.47, wall: 4.71, earth: 3.29, floor: 2.74, windowpane: 1.71, person: 1.61, tree: 1.50, road: 1.29, ceiling: 1.28, field: 1.07, car: 1.04, fence: 0.92, mountain: 0.77, hill: 0.72, truck: 0.67, skyscraper: 0.45, stairs: 0.33, sand: 0.22, stairway: 0.20, van: 0.15, bus: 0.11, door: 0.10, path: 0.09, box: 0.08, pole: 0.08, trade_name: 0.07, conveyor_belt: 0.04, signboard: 0.03, land: 0.02, flag: 0.01, water: 0.01

B.1.2 Instances

Default Instance Size

Context across the entire instance
sky: 30.30, building: 16.84, runway: 16.41, sea: 7.50, wall: 4.64, grass: 4.46, mountain: 3.24, tree: 2.24, windowpane: 1.99, floor: 1.97, road: 1.83, person: 1.34, earth: 1.27, field: 0.92, ceiling: 0.83, car: 0.71, sand: 0.61, hill: 0.45, signboard: 0.35, pole: 0.28, truck: 0.28, skyscraper: 0.27, streetlight: 0.16, land: 0.15, stairway: 0.14, conveyor_belt: 0.11, water: 0.08, bus: 0.08, stairs: 0.07, crt_screen: 0.06, van: 0.06, path: 0.06, fence: 0.06, box: 0.05, cabinet: 0.04, door: 0.04, bulletin_board: 0.04, tower: 0.03, painting: 0.02
Instances with no context: 29/272

Context in specific instance quadrants
quadrant-I:
sky: 38.29, building: 19.34, runway: 7.58, sea: 7.15, wall: 6.05, grass: 4.41, mountain: 3.53, tree: 3.44, windowpane: 3.12, earth: 1.16, ceiling: 1.10, field: 0.77, road: 0.75, hill: 0.53, sand: 0.43, person: 0.36, streetlight: 0.31, land: 0.28, tower: 0.24, pole: 0.16, signboard: 0.14, conveyor_belt: 0.13, skyscraper: 0.12, water: 0.12, cabinet: 0.11, car: 0.09, truck: 0.08, bulletin_board: 0.08, crt_screen: 0.05, door: 0.04, trade_name: 0.02, floor: 0.01
quadrant-II:
sky: 39.35, building: 19.67, runway: 8.00, wall: 6.84, sea: 6.42, mountain: 4.73, tree: 3.19, grass: 3.10, windowpane: 2.15, ceiling: 1.34, road: 0.95, earth: 0.75, field: 0.71, hill: 0.46, skyscraper: 0.43, sand: 0.41, person: 0.41, signboard: 0.24, land: 0.19, water: 0.16, truck: 0.12, floor: 0.08, door: 0.05, conveyor_belt: 0.04, streetlight: 0.04, plant: 0.04, fence: 0.03, car: 0.03, stairway: 0.03, pole: 0.02
quadrant-III:
runway: 30.48, building: 15.75, sky: 15.28, grass: 5.82, sea: 5.03, floor: 4.19, person: 3.33, road: 2.71, wall: 2.67, mountain: 1.99, windowpane: 1.81, earth: 1.74, field: 1.37, stairway: 1.04, tree: 0.93, car: 0.90, sand: 0.87, ceiling: 0.79, truck: 0.54, signboard: 0.50, bus: 0.48, skyscraper: 0.44, hill: 0.44, fence: 0.23, van: 0.22, box: 0.10, stairs: 0.10, painting: 0.08, conveyor_belt: 0.08, path: 0.02, land: 0.02, bulletin_board: 0.02, streetlight: 0.01, pole: 0.01, flag: 0.01
quadrant-IV:
runway: 30.14, sky: 14.89, building: 14.57, sea: 7.12, grass: 4.88, floor: 4.71, person: 2.58,

wall: 2.25, road: 2.10, mountain: 2.06, windowpane: 2.05, car: 2.01, earth: 1.71, field: 1.38, tree: 1.24, ceiling: 0.85, sand: 0.81, truck: 0.74, pole: 0.66, signboard: 0.46, hill: 0.44, crt_screen: 0.44, streetlight: 0.38, stairway: 0.32, stairs: 0.27, van: 0.19, path: 0.12, conveyor_belt: 0.12, skyscraper: 0.10, box: 0.09, bus: 0.08, cabinet: 0.07, door: 0.06, land: 0.05, tower: 0.02, fence: 0.01, bulletin_board: 0.01, seat: 0.01

10% - Increased Instances

Context across the entire instance

sky: 29.95, runway: 17.23, building: 16.06, sea: 8.25, wall: 4.56, grass: 4.34, mountain: 2.91, tree: 2.20, floor: 2.01, windowpane: 1.97, road: 1.90, person: 1.38, earth: 1.27, field: 0.97, ceiling: 0.83, car: 0.67, sand: 0.59, hill: 0.46, signboard: 0.34, truck: 0.30, pole: 0.28, skyscraper: 0.26, stairway: 0.19, streetlight: 0.15, land: 0.15, conveyor_belt: 0.12, bus: 0.08, water: 0.08, fence: 0.08, van: 0.06, stairs: 0.06, crt_screen: 0.06, path: 0.05, tower: 0.05, cabinet: 0.04, box: 0.04, bulletin_board: 0.03, door: 0.03, painting: 0.02
Instances with no context: 27/272

Context in specific instance quadrants

quadrant-I:
sky: 39.62, building: 18.70, sea: 7.81, runway: 7.26, wall: 5.96, grass: 4.11, tree: 3.38, mountain: 3.30, windowpane: 3.00, earth: 1.10, ceiling: 1.05, field: 0.74, road: 0.72, hill: 0.53, sand: 0.42, person: 0.40, streetlight: 0.33, land: 0.24, tower: 0.22, conveyor_belt: 0.16, signboard: 0.14, pole: 0.13, cabinet: 0.11, water: 0.10, skyscraper: 0.10, car: 0.10, truck: 0.09, bulletin_board: 0.07, door: 0.03, crt_screen: 0.03, floor: 0.02, trade_name: 0.01, fence: 0.01
quadrant-II:
sky: 40.44, building: 18.74, runway: 8.11, wall: 6.75, sea: 6.74, mountain: 4.48, tree: 3.14, grass: 2.96, windowpane: 2.16, ceiling: 1.33, road: 0.87, earth: 0.73, field: 0.70, hill: 0.49, skyscraper: 0.43, person: 0.40, sand: 0.38, signboard: 0.23, land: 0.19, water: 0.15, truck: 0.12, pole: 0.09, floor: 0.09, conveyor_belt: 0.08, door: 0.04, streetlight: 0.04, plant: 0.03, stairway: 0.03, fence: 0.03, car: 0.02
quadrant-III:
runway: 33.34, building: 14.69, sky: 14.08, grass: 5.49, sea: 4.97, floor: 4.35, person: 3.36, road: 2.80, wall: 2.49, windowpane: 1.72, earth: 1.70, field: 1.48, mountain: 1.47, stairway: 1.11, car: 1.09, tree: 0.93, sand: 0.85, ceiling: 0.80, truck: 0.50, signboard: 0.48, bus: 0.45, skyscraper: 0.44, hill: 0.44, fence: 0.23, van: 0.19, stairs: 0.09, box: 0.09, pole: 0.08, painting: 0.08, conveyor_belt: 0.07, land: 0.04, path: 0.04, bulletin_board: 0.02, streetlight: 0.01, plant: 0.01
quadrant-IV:
runway: 32.73, building: 14.34, sky: 14.00, sea: 7.04, floor: 4.74, grass: 4.67, person: 2.38, wall: 2.18, road: 2.13, windowpane: 1.98, car: 1.94, earth: 1.71, field: 1.58, mountain: 1.38, tree: 1.09, sand: 0.81, ceiling: 0.79, truck: 0.78, pole: 0.51, signboard: 0.44, hill: 0.44, crt_screen: 0.44, streetlight: 0.33, stairway: 0.32, fence: 0.22, stairs: 0.22, van: 0.17, path:

0.11, bus: 0.10, skyscraper: 0.09, conveyor_belt: 0.08, cabinet: 0.06, tower: 0.06, box: 0.05, door: 0.04, land: 0.03, seat: 0.01, bulletin_board: 0.01

20% - Increased Instances

Context across the entire instance

sky: 29.38, runway: 18.51, building: 15.40, sea: 8.51, wall: 4.39, grass: 4.26, mountain: 2.62, tree: 2.14, floor: 2.11, windowpane: 1.98, road: 1.94, person: 1.63, earth: 1.24, field: 0.94, ceiling: 0.81, car: 0.65, sand: 0.55, hill: 0.46, signboard: 0.31, truck: 0.30, pole: 0.27, skyscraper: 0.25, stairway: 0.20, streetlight: 0.15, conveyor_belt: 0.13, land: 0.13, fence: 0.10, bus: 0.09, water: 0.07, van: 0.06, stairs: 0.06, cabinet: 0.05, tower: 0.05, crt_screen: 0.05, path: 0.04, box: 0.03, bulletin_board: 0.03, door: 0.02, plant: 0.02, painting: 0.02
Instances with no context: 24/272

Context in specific instance quadrants

quadrant-I:
sky: 40.53, building: 18.22, sea: 8.10, runway: 6.72, wall: 5.92, grass: 4.11, tree: 3.40, mountain: 3.13, windowpane: 2.93, earth: 1.04, ceiling: 0.99, road: 0.69, field: 0.66, hill: 0.53, person: 0.44, sand: 0.41, pole: 0.38, streetlight: 0.35, land: 0.22, tower: 0.21, conveyor_belt: 0.20, signboard: 0.12, cabinet: 0.12, water: 0.09, car: 0.09, skyscraper: 0.09, truck: 0.08, bulletin_board: 0.07, fence: 0.05, floor: 0.04, door: 0.03, crt_screen: 0.01, trade_name: 0.01
quadrant-II:
sky: 41.28, building: 17.89, runway: 7.46, sea: 7.08, wall: 6.69, mountain: 4.22, grass: 3.56, tree: 3.04, windowpane: 2.16, ceiling: 1.32, road: 0.82, earth: 0.69, person: 0.68, field: 0.62, hill: 0.52, skyscraper: 0.43, sand: 0.32, signboard: 0.22, land: 0.17, conveyor_belt: 0.16, truck: 0.14, water: 0.13, pole: 0.10, floor: 0.09, streetlight: 0.04, plant: 0.04, door: 0.03, cabinet: 0.03, stairway: 0.03, car: 0.02, fence: 0.02
quadrant-III:
runway: 36.43, building: 13.30, sky: 12.91, grass: 5.20, sea: 4.87, floor: 4.48, person: 3.87, road: 2.87, wall: 2.40, windowpane: 1.68, earth: 1.63, field: 1.46, mountain: 1.17, car: 1.16, stairway: 0.93, tree: 0.90, ceiling: 0.79, sand: 0.78, truck: 0.49, signboard: 0.46, skyscraper: 0.43, hill: 0.43, bus: 0.43, fence: 0.20, van: 0.16, pole: 0.09, stairs: 0.08, box: 0.08, painting: 0.08, conveyor_belt: 0.06, plant: 0.05, land: 0.05, path: 0.03, bulletin_board: 0.02, streetlight: 0.02, ship: 0.01
quadrant-IV:
runway: 35.05, building: 13.40, sky: 13.02, sea: 7.27, floor: 4.74, grass: 4.45, person: 2.75, road: 2.36, wall: 2.08, windowpane: 1.94, car: 1.78, earth: 1.64, field: 1.63, mountain: 1.15, tree: 1.01, sand: 0.79, truck: 0.73, ceiling: 0.71, hill: 0.43, crt_screen: 0.43, pole: 0.42, signboard: 0.41, fence: 0.31, stairway: 0.29, streetlight: 0.26, stairs: 0.17, van: 0.15, bus: 0.13, path: 0.10, skyscraper: 0.09, conveyor_belt: 0.06, tower: 0.06, cabinet: 0.05, door: 0.04, box: 0.03, land: 0.03, column: 0.02, ship: 0.01, seat: 0.01

30% - Increased Instances

Context across the entire instance

sky: 29.02, runway: 19.15, building: 14.71, sea: 9.14, wall: 4.26, grass: 4.16, mountain: 2.42, floor: 2.20, tree: 2.18, windowpane: 1.98, road: 1.96, person: 1.69, earth: 1.25, field: 1.01, ceiling: 0.78, car: 0.62, sand: 0.52, hill: 0.47, truck: 0.31, signboard: 0.28, pole: 0.27, skyscraper: 0.25, stairway: 0.22, conveyor_belt: 0.14, streetlight: 0.14, land: 0.12, fence: 0.10, bus: 0.09, cabinet: 0.07, water: 0.06, stairs: 0.06, tower: 0.05, crt_screen: 0.05, van: 0.05, path: 0.04, plant: 0.04, box: 0.03, bulletin_board: 0.02, ship: 0.02, door: 0.02, painting: 0.02
Instances with no context: 21/272

Context in specific instance quadrants

quadrant-I:
sky: 41.21, building: 17.65, sea: 8.74, runway: 6.55, wall: 5.86, grass: 3.87, tree: 3.49, mountain: 2.98, windowpane: 2.87, earth: 1.00, ceiling: 0.96, road: 0.67, field: 0.58, hill: 0.54, person: 0.52, sand: 0.41, pole: 0.38, streetlight: 0.33, conveyor_belt: 0.22, tower: 0.21, land: 0.20, cabinet: 0.13, signboard: 0.11, water: 0.09, truck: 0.08, skyscraper: 0.08, car: 0.07, bulletin_board: 0.07, fence: 0.05, floor: 0.04, door: 0.02, trade_name: 0.01, crt_screen: 0.01
quadrant-II:
sky: 42.12, building: 16.99, sea: 7.42, runway: 7.00, wall: 6.63, mountain: 3.98, grass: 3.61, tree: 3.52, windowpane: 2.09, ceiling: 1.32, road: 0.77, person: 0.72, earth: 0.62, field: 0.61, hill: 0.54, skyscraper: 0.42, sand: 0.27, conveyor_belt: 0.21, signboard: 0.20, land: 0.16, truck: 0.15, pole: 0.12, water: 0.12, floor: 0.09, cabinet: 0.08, plant: 0.06, streetlight: 0.04, stairway: 0.04, crt_screen: 0.04, door: 0.03, car: 0.02, fence: 0.02
quadrant-III:
runway: 37.60, building: 12.54, sky: 12.14, sea: 5.24, grass: 4.97, floor: 4.63, person: 3.88, road: 2.92, wall: 2.44, earth: 1.70, windowpane: 1.66, field: 1.63, car: 1.22, mountain: 1.00, stairway: 0.94, tree: 0.86, sand: 0.74, ceiling: 0.69, truck: 0.54, skyscraper: 0.43, hill: 0.43, signboard: 0.41, bus: 0.40, fence: 0.19, van: 0.14, pole: 0.12, plant: 0.09, painting: 0.07, stairs: 0.07, box: 0.07, conveyor_belt: 0.05, ship: 0.05, land: 0.05, path: 0.03, bulletin_board: 0.02, streetlight: 0.01, door: 0.01, cabinet: 0.01
quadrant-IV:
runway: 37.15, building: 12.39, sky: 11.94, sea: 7.87, floor: 4.73, grass: 4.35, person: 2.79, road: 2.47, wall: 1.98, windowpane: 1.90, earth: 1.80, field: 1.70, car: 1.60, mountain: 1.05, tree: 0.94, sand: 0.78, truck: 0.67, ceiling: 0.63, hill: 0.42, crt_screen: 0.41, pole: 0.40, signboard: 0.37, stairway: 0.27, fence: 0.23, stairs: 0.18, streetlight: 0.18, bus: 0.15, van: 0.12, path: 0.09, skyscraper: 0.09, column: 0.06, ship: 0.06, conveyor_belt: 0.05, tower: 0.05, cabinet: 0.05, box: 0.03, door: 0.02, land: 0.02, seat: 0.01

40% - Increased Instances

Context across the entire instance

sky: 28.72, runway: 19.65, building: 14.09, sea: 9.84, wall: 4.15, grass: 4.11, floor: 2.28,

mountain: 2.27, tree: 2.15, windowpane: 1.98, road: 1.93, person: 1.76, earth: 1.19, field: 0.98, ceiling: 0.78, car: 0.62, sand: 0.51, hill: 0.47, truck: 0.31, pole: 0.31, signboard: 0.26, skyscraper: 0.25, stairway: 0.23, conveyor_belt: 0.14, streetlight: 0.12, land: 0.11, fence: 0.09, bus: 0.08, cabinet: 0.08, tower: 0.07, crt_screen: 0.07, water: 0.06, stairs: 0.05, plant: 0.05, van: 0.05, path: 0.03, ship: 0.03, box: 0.03, column: 0.02, bulletin_board: 0.02, painting: 0.02, door: 0.02
Instances with no context: 19/272

Context in specific instance quadrants
quadrant-I:
sky: 41.81, building: 16.96, sea: 9.05, runway: 6.33, wall: 6.18, grass: 3.77, tree: 3.45, mountain: 2.87, windowpane: 2.79, earth: 1.00, ceiling: 0.97, person: 0.65, road: 0.64, hill: 0.54, field: 0.47, pole: 0.42, sand: 0.40, streetlight: 0.31, tower: 0.28, conveyor_belt: 0.21, land: 0.18, signboard: 0.10, cabinet: 0.10, skyscraper: 0.09, water: 0.08, truck: 0.08, bulletin_board: 0.07, car: 0.06, floor: 0.05, fence: 0.04, door: 0.02, trade_name: 0.01
quadrant-II:
sky: 42.72, building: 16.24, sea: 8.11, runway: 6.64, wall: 6.53, grass: 3.97, mountain: 3.76, tree: 3.42, windowpane: 2.00, ceiling: 1.31, person: 0.83, road: 0.73, field: 0.56, hill: 0.54, earth: 0.54, skyscraper: 0.41, sand: 0.23, conveyor_belt: 0.21, signboard: 0.19, pole: 0.15, truck: 0.14, land: 0.14, cabinet: 0.12, water: 0.11, crt_screen: 0.08, floor: 0.07, plant: 0.07, stairway: 0.05, streetlight: 0.04, door: 0.02, car: 0.02, fence: 0.02
quadrant-III:
runway: 38.91, building: 12.04, sky: 11.42, sea: 5.22, grass: 4.86, floor: 4.80, person: 3.81, road: 2.95, wall: 2.47, field: 1.72, windowpane: 1.62, earth: 1.62, car: 1.32, stairway: 0.91, mountain: 0.88, tree: 0.84, sand: 0.72, ceiling: 0.64, truck: 0.60, skyscraper: 0.43, hill: 0.43, signboard: 0.38, bus: 0.37, fence: 0.19, pole: 0.16, van: 0.14, plant: 0.12, ship: 0.07, painting: 0.06, stairs: 0.06, box: 0.06, conveyor_belt: 0.06, land: 0.05, path: 0.03, crt_screen: 0.02, bulletin_board: 0.02, door: 0.01, streetlight: 0.01, cabinet: 0.01
quadrant-IV:
runway: 38.23, building: 11.69, sky: 11.47, sea: 8.22, floor: 4.81, grass: 4.42, person: 2.74, road: 2.54, windowpane: 1.92, wall: 1.91, earth: 1.75, field: 1.73, car: 1.54, mountain: 1.00, tree: 0.88, sand: 0.78, truck: 0.63, ceiling: 0.59, pole: 0.42, hill: 0.42, crt_screen: 0.38, signboard: 0.34, stairway: 0.27, fence: 0.19, stairs: 0.18, bus: 0.15, streetlight: 0.14, van: 0.10, skyscraper: 0.10, column: 0.09, path: 0.08, ship: 0.08, tower: 0.05, conveyor_belt: 0.05, cabinet: 0.04, box: 0.04, escalator: 0.02, door: 0.02, land: 0.02, seat: 0.01

B.2 *COCO - Things and Stuff*

B.2.1 Images

Semantic Context

Context across the entire image

sky-other: 35.42, clouds: 16.03, road: 8.81, grass: 5.07, pavement: 4.97, tree: 3.47, building-other: 3.38, fog: 3.18, mountain: 2.40, ground-other: 2.20, sea: 1.55, person: 1.40, ceiling-other: 0.80, truck: 0.78, floor-other: 0.76, metal: 0.71, dirt: 0.69, fence: 0.63, wall-other: 0.57, window-other: 0.56, sand: 0.51, hill: 0.38, snow: 0.32, bush: 0.31, wall-concrete: 0.31, plant-other: 0.24, car: 0.24, water-other: 0.22, wall-panel: 0.21, solid-other: 0.19, skyscraper: 0.19, river: 0.16, chair: 0.15, bus: 0.14, floor-tile: 0.14, bridge: 0.14, platform: 0.12, table: 0.12, structural-other: 0.11, gravel: 0.11, roof: 0.11, stairs: 0.11, house: 0.09, branch: 0.09, rock: 0.08, railing: 0.08, boat: 0.07, banner: 0.07, floor-wood: 0.07, textile-other: 0.06, other: 1.44
images with no context: 9/3077

Context in specific image quadrants

quadrant-I:
sky-other: 49.05, clouds: 22.22, fog: 4.91, tree: 3.87, building-other: 3.71, mountain: 2.18, ceiling-other: 1.46, road: 1.40, grass: 1.33, person: 0.97, pavement: 0.95, wall-other: 0.74, window-other: 0.74, sea: 0.69, metal: 0.59, ground-other: 0.55, hill: 0.41, wall-concrete: 0.30, truck: 0.27, wall-panel: 0.26, fence: 0.23, skyscraper: 0.20, solid-other: 0.18, roof: 0.17, bush: 0.15, floor-other: 0.13, table: 0.12, sand: 0.11, bus: 0.11, structural-other: 0.10, banner: 0.09, plant-other: 0.09, water-other: 0.08, chair: 0.08, dirt: 0.07, river: 0.07, window-blind: 0.07, branch: 0.06, textile-other: 0.06, wall-brick: 0.05, paper: 0.05, house: 0.05, tent: 0.05, plastic: 0.04, stairs: 0.04, bridge: 0.04, light: 0.04, floor-tile: 0.04, snow: 0.04, furniture-other: 0.04, other: 0.77
quadrant-II:
sky-other: 49.16, clouds: 22.22, fog: 4.84, tree: 3.92, building-other: 3.69, mountain: 2.22, road: 1.63, ceiling-other: 1.45, grass: 1.31, pavement: 1.06, person: 0.80, wall-other: 0.73, window-other: 0.68, sea: 0.62, ground-other: 0.61, hill: 0.44, metal: 0.42, wall-concrete: 0.32, truck: 0.32, wall-panel: 0.30, fence: 0.20, skyscraper: 0.20, dirt: 0.16, bush: 0.15, solid-other: 0.14, floor-other: 0.14, roof: 0.14, water-other: 0.12, sand: 0.11, table: 0.10, structural-other: 0.09, house: 0.09, car: 0.08, river: 0.08, banner: 0.08, plant-other: 0.08, textile-other: 0.07, bus: 0.07, wall-brick: 0.06, light: 0.06, bridge: 0.06, branch: 0.06, floor-tile: 0.05, refrigerator: 0.05, paper: 0.05, stairs: 0.05, snow: 0.04, cloth: 0.03, window-blind: 0.03, tent: 0.03, other: 0.60
quadrant-III:
sky-other: 20.78, road: 17.14, clouds: 9.23, pavement: 9.15, grass: 9.12, ground-other: 4.06, building-other: 3.00, tree: 2.98, mountain: 2.54, sea: 2.40, person: 1.91, truck: 1.45, fog: 1.42, floor-other: 1.35, dirt: 1.35, fence: 1.02, sand: 0.89, metal: 0.89, snow: 0.66, bush: 0.47, wall-other: 0.46, car: 0.40, plant-other: 0.38, window-other: 0.37, water-other: 0.37, hill:

0.36, wall-concrete: 0.31, river: 0.26, platform: 0.24, gravel: 0.24, floor-tile: 0.23, bridge: 0.23, solid-other: 0.23, chair: 0.18, stairs: 0.16, rock: 0.16, railing: 0.15, bus: 0.14, wall-panel: 0.14, floor-wood: 0.13, table: 0.13, skyscraper: 0.13, ceiling-other: 0.13, branch: 0.12, structural-other: 0.12, boat: 0.12, house: 0.11, cage: 0.10, straw: 0.10, roof: 0.09, other: 1.91
quadrant-IV:
sky-other: 20.73, road: 16.59, clouds: 9.39, grass: 9.24, pavement: 9.12, ground-other: 3.86, building-other: 3.17, tree: 2.95, mountain: 2.57, sea: 2.49, person: 2.25, floor-other: 1.44, fog: 1.41, dirt: 1.30, truck: 1.26, fence: 1.08, metal: 1.08, sand: 0.91, snow: 0.57, bush: 0.48, plant-other: 0.43, window-other: 0.41, wall-other: 0.39, car: 0.34, water-other: 0.33, hill: 0.31, wall-concrete: 0.30, river: 0.26, solid-other: 0.24, chair: 0.24, bridge: 0.23, floor-tile: 0.22, gravel: 0.21, bus: 0.21, skyscraper: 0.20, platform: 0.20, stairs: 0.20, structural-other: 0.16, rock: 0.14, ceiling-other: 0.13, house: 0.13, table: 0.13, wall-panel: 0.12, floor-wood: 0.12, branch: 0.11, railing: 0.11, suitcase: 0.11, straw: 0.10, boat: 0.09, cage: 0.09, other: 1.84
image quadrants with no context:
quadrant-I: 20/3077
quadrant-II: 20/3077
quadrant-III: 18/3077
quadrant-IV: 16/3077

Label-neighborhood

left:
sky-other: 35.06, clouds: 15.57, road: 7.77, building-other: 5.41, tree: 4.69, grass: 4.22, pavement: 4.15, fog: 3.01, person: 2.85, mountain: 2.42, ground-other: 1.93, metal: 1.69, sea: 1.23, window-other: 0.79, fence: 0.70, ceiling-other: 0.62, floor-other: 0.59, dirt: 0.57, wall-other: 0.56, hill: 0.51, truck: 0.45, sand: 0.35, bush: 0.30, wall-concrete: 0.30, solid-other: 0.24, snow: 0.23, wall-panel: 0.22, plant-other: 0.20, water-other: 0.19, car: 0.16, river: 0.15, table: 0.14, stairs: 0.14, house: 0.13, platform: 0.10, banner: 0.10, chair: 0.10, structural-other: 0.10, floor-tile: 0.10, bridge: 0.09, bus: 0.07, paper: 0.06, plastic: 0.06, textile-other: 0.06, skyscraper: 0.06, roof: 0.06, branch: 0.06, railing: 0.06, wall-brick: 0.05, refrigerator: 0.05, other: 0.90
up:
sky-other: 38.43, clouds: 17.24, building-other: 7.45, tree: 6.29, road: 4.68, fog: 3.36, grass: 3.21, mountain: 2.81, pavement: 2.38, ground-other: 1.46, sea: 1.25, metal: 1.19, person: 0.92, window-other: 0.87, ceiling-other: 0.86, wall-other: 0.76, hill: 0.63, fence: 0.53, wall-concrete: 0.41, floor-other: 0.34, wall-panel: 0.34, plant-other: 0.29, bush: 0.29, dirt: 0.25, sand: 0.23, solid-other: 0.21, water-other: 0.20, truck: 0.17, house: 0.15, snow: 0.15, river: 0.14, table: 0.14, banner: 0.13, bridge: 0.11, structural-other: 0.10, skyscraper: 0.09, roof: 0.09, stairs: 0.08, paper: 0.08, wall-brick: 0.07, platform: 0.07, chair: 0.06, textile-other: 0.06, floor-tile: 0.06, straw: 0.06, branch: 0.05, car: 0.05, bus: 0.05, refrigerator: 0.04, boat: 0.04, other: 0.71
right:
sky-other: 34.61, clouds: 15.64, road: 7.14, building-other: 5.99, tree: 4.69, grass: 4.33,

pavement: 3.87, person: 3.31, fog: 2.99, mountain: 2.56, ground-other: 1.84, metal: 1.57, sea: 1.19, window-other: 0.82, fence: 0.70, ceiling-other: 0.66, floor-other: 0.61, wall-other: 0.59, hill: 0.54, dirt: 0.45, truck: 0.45, wall-concrete: 0.38, sand: 0.33, wall-panel: 0.27, plant-other: 0.26, solid-other: 0.26, bush: 0.25, snow: 0.22, stairs: 0.20, water-other: 0.16, river: 0.16, table: 0.14, house: 0.14, car: 0.12, chair: 0.12, structural-other: 0.11, platform: 0.10, bridge: 0.09, floor-tile: 0.09, bus: 0.09, banner: 0.09, skyscraper: 0.08, textile-other: 0.07, straw: 0.06, playingfield: 0.06, paper: 0.06, kite: 0.06, dining table: 0.05, floor-wood: 0.05, railing: 0.05, other: 0.91
down:
sky-other: 27.97, road: 13.67, clouds: 12.06, pavement: 6.94, grass: 6.92, building-other: 4.25, tree: 3.44, ground-other: 3.14, fog: 2.21, person: 2.05, mountain: 2.03, metal: 2.01, sea: 1.26, floor-other: 0.99, fence: 0.99, dirt: 0.92, truck: 0.86, window-other: 0.70, wall-other: 0.49, sand: 0.47, hill: 0.44, snow: 0.40, bush: 0.34, ceiling-other: 0.32, wall-concrete: 0.31, solid-other: 0.30, plant-other: 0.30, car: 0.26, river: 0.23, stairs: 0.23, platform: 0.21, water-other: 0.18, table: 0.16, chair: 0.14, bridge: 0.14, house: 0.14, wall-panel: 0.13, bus: 0.12, floor-tile: 0.11, structural-other: 0.11, gravel: 0.10, straw: 0.09, playingfield: 0.09, cage: 0.09, railing: 0.08, banner: 0.08, floor-wood: 0.07, dining table: 0.06, plastic: 0.06, textile-other: 0.06, other: 0.91

B.2.2 Instances

Default Instance Size

Context across the entire instance
sky-other: 32.64, clouds: 16.24, building-other: 9.13, road: 7.17, tree: 5.09, grass: 4.07, fog: 3.61, pavement: 3.35, mountain: 2.53, ground-other: 1.56, person: 1.30, floor-other: 1.09, sea: 1.04, metal: 1.04, hill: 1.00, window-other: 0.97, ceiling-other: 0.84, fence: 0.65, wall-other: 0.65, truck: 0.63, sand: 0.60, dirt: 0.48, wall-concrete: 0.31, plant-other: 0.23, bush: 0.23, platform: 0.22, solid-other: 0.21, wall-panel: 0.20, structural-other: 0.16, water-other: 0.15, floor-tile: 0.15, snow: 0.14, river: 0.14, car: 0.12, bridge: 0.12, roof: 0.11, banner: 0.11, bus: 0.11, house: 0.10, skyscraper: 0.10, straw: 0.10, stairs: 0.10, table: 0.09, playingfield: 0.07, chair: 0.07, wood: 0.06, light: 0.05, railing: 0.05, cage: 0.05, floor-wood: 0.05, other: 0.74
Instances with no context: 29/5270

Context in specific instance quadrants
quadrant-l:
sky-other: 38.22, clouds: 18.94, building-other: 10.32, tree: 6.16, fog: 4.25, road: 2.95, mountain: 2.80, grass: 2.39, hill: 1.19, ceiling-other: 1.16, window-other: 1.07, pavement: 1.06, sea: 1.01, metal: 0.98, wall-other: 0.79, person: 0.74, ground-other: 0.66, floor-other: 0.56, sand: 0.48, wall-concrete: 0.43, fence: 0.39, wall-panel: 0.26, plant-other: 0.25, solid-other: 0.18, structural-other: 0.18, truck: 0.16, bush: 0.15, roof: 0.15, dirt: 0.14, water-other: 0.13, banner: 0.12, river: 0.10, skyscraper: 0.10, bridge: 0.09, house: 0.09, table: 0.08, floor-tile:

0.08, platform: 0.07, straw: 0.07, light: 0.06, wood: 0.06, window-blind: 0.06, wall-brick: 0.05, bus: 0.05, branch: 0.04, snow: 0.04, chair: 0.04, paper: 0.04, wall-wood: 0.04, textile-other: 0.03, other: 0.54
quadrant-II:
sky-other: 38.19, clouds: 18.75, building-other: 9.97, tree: 6.29, fog: 4.27, road: 3.25, mountain: 2.61, grass: 2.45, pavement: 1.25, hill: 1.19, ceiling-other: 1.13, window-other: 1.02, sea: 1.01, metal: 0.85, wall-other: 0.80, person: 0.72, ground-other: 0.71, floor-other: 0.59, sand: 0.51, fence: 0.38, wall-concrete: 0.35, wall-panel: 0.26, plant-other: 0.23, bush: 0.21, truck: 0.19, platform: 0.18, banner: 0.17, structural-other: 0.17, solid-other: 0.16, dirt: 0.16, skyscraper: 0.15, roof: 0.14, water-other: 0.14, river: 0.13, bridge: 0.11, house: 0.11, light: 0.08, table: 0.08, car: 0.07, wood: 0.07, paper: 0.05, snow: 0.05, floor-tile: 0.05, cage: 0.04, bus: 0.04, window-blind: 0.04, refrigerator: 0.04, wall-wood: 0.04, textile-other: 0.03, railing: 0.03, other: 0.49
quadrant-III:
sky-other: 24.62, road: 13.66, clouds: 12.18, building-other: 7.47, pavement: 6.54, grass: 6.29, tree: 3.84, ground-other: 2.94, fog: 2.57, mountain: 1.99, person: 1.94, floor-other: 1.64, metal: 1.51, truck: 1.34, sea: 1.06, dirt: 0.98, fence: 0.97, window-other: 0.91, sand: 0.81, hill: 0.64, wall-other: 0.55, ceiling-other: 0.47, platform: 0.37, bush: 0.32, snow: 0.30, solid-other: 0.27, wall-concrete: 0.26, river: 0.21, plant-other: 0.21, car: 0.20, water-other: 0.17, stairs: 0.17, bus: 0.15, floor-tile: 0.15, wall-panel: 0.14, bridge: 0.14, structural-other: 0.13, straw: 0.12, gravel: 0.11, house: 0.10, table: 0.10, roof: 0.08, cage: 0.08, playingfield: 0.07, banner: 0.07, railing: 0.07, chair: 0.07, wood: 0.07, floor-wood: 0.06, light: 0.06, other: 0.84
quadrant-IV:
sky-other: 24.52, road: 13.30, clouds: 12.04, building-other: 8.38, pavement: 6.40, grass: 6.40, tree: 3.64, ground-other: 2.69, fog: 2.49, person: 2.18, mountain: 2.05, floor-other: 1.67, metal: 1.61, truck: 1.20, sea: 1.10, fence: 0.98, window-other: 0.93, dirt: 0.89, sand: 0.74, hill: 0.66, wall-other: 0.51, ceiling-other: 0.50, snow: 0.28, solid-other: 0.27, wall-concrete: 0.26, platform: 0.26, bush: 0.25, stairs: 0.24, plant-other: 0.24, car: 0.21, straw: 0.21, river: 0.20, floor-tile: 0.16, structural-other: 0.15, water-other: 0.14, bridge: 0.13, wall-panel: 0.12, bus: 0.11, house: 0.11, gravel: 0.10, playingfield: 0.10, table: 0.09, chair: 0.09, banner: 0.08, railing: 0.08, floor-wood: 0.08, skyscraper: 0.08, textile-other: 0.07, cage: 0.07, roof: 0.06, other: 0.88

10% - Increased Instances

Context across the entire instance
sky-other: 32.54, clouds: 16.19, building-other: 8.89, road: 7.56, tree: 4.93, grass: 4.08, fog: 3.59, pavement: 3.52, mountain: 2.44, ground-other: 1.62, person: 1.21, floor-other: 1.09, metal: 1.07, sea: 1.04, window-other: 0.98, hill: 0.97, ceiling-other: 0.84, fence: 0.65, truck: 0.65, wall-other: 0.64, sand: 0.61, dirt: 0.49, wall-concrete: 0.32, plant-other: 0.23, bush: 0.23, platform: 0.22, solid-other: 0.21, wall-panel: 0.20, structural-other: 0.17, snow: 0.15, floor-tile: 0.15, water-other: 0.15, river: 0.15, car: 0.12, bridge: 0.12, roof: 0.11, bus: 0.11, banner: 0.10, skyscraper: 0.10, straw: 0.10, house: 0.10, stairs: 0.10, table: 0.09, playingfield: 0.08,

chair: 0.07, wood: 0.06, light: 0.05, railing: 0.05, cage: 0.05, floor-wood: 0.05, other: 0.75
Instances with no context: 29/5270

Context in specific instance quadrants
quadrant-I:
sky-other: 38.64, clouds: 19.14, building-other: 10.11, tree: 6.04, fog: 4.35, road: 2.83, mountain: 2.72, grass: 2.35, ceiling-other: 1.17, hill: 1.15, window-other: 1.08, pavement: 1.03, sea: 1.02, metal: 0.97, wall-other: 0.79, person: 0.66, ground-other: 0.65, floor-other: 0.55, sand: 0.47, wall-concrete: 0.41, fence: 0.38, wall-panel: 0.26, plant-other: 0.24, solid-other: 0.18, structural-other: 0.18, truck: 0.16, roof: 0.16, dirt: 0.16, bush: 0.14, water-other: 0.13, banner: 0.12, skyscraper: 0.12, river: 0.11, house: 0.09, bridge: 0.09, floor-tile: 0.08, table: 0.08, straw: 0.07, light: 0.07, platform: 0.07, wood: 0.06, window-blind: 0.06, bus: 0.05, wall-brick: 0.05, snow: 0.04, chair: 0.04, paper: 0.04, branch: 0.04, wall-wood: 0.04, textile-other: 0.03, other: 0.51
quadrant-II:
sky-other: 38.64, clouds: 18.95, building-other: 9.79, tree: 6.22, fog: 4.32, road: 3.12, mountain: 2.57, grass: 2.39, pavement: 1.20, hill: 1.17, ceiling-other: 1.15, window-other: 1.01, sea: 1.00, metal: 0.86, wall-other: 0.80, ground-other: 0.71, person: 0.65, floor-other: 0.58, sand: 0.51, fence: 0.37, wall-concrete: 0.36, wall-panel: 0.26, plant-other: 0.22, bush: 0.21, truck: 0.19, structural-other: 0.17, banner: 0.16, solid-other: 0.16, platform: 0.16, dirt: 0.16, skyscraper: 0.15, water-other: 0.14, roof: 0.14, river: 0.13, bridge: 0.12, house: 0.11, light: 0.08, table: 0.08, wood: 0.07, car: 0.06, paper: 0.05, floor-tile: 0.05, snow: 0.05, window-blind: 0.04, refrigerator: 0.04, cage: 0.04, bus: 0.04, wall-wood: 0.04, wall-brick: 0.03, textile-other: 0.03, other: 0.48
quadrant-III:
sky-other: 24.22, road: 14.46, clouds: 11.97, building-other: 7.20, pavement: 6.84, grass: 6.33, tree: 3.59, ground-other: 3.03, fog: 2.48, mountain: 1.91, person: 1.83, floor-other: 1.67, metal: 1.57, truck: 1.33, sea: 1.05, dirt: 0.99, fence: 0.96, window-other: 0.90, sand: 0.81, hill: 0.62, wall-other: 0.55, ceiling-other: 0.45, platform: 0.38, bush: 0.33, snow: 0.32, wall-concrete: 0.28, solid-other: 0.28, river: 0.23, plant-other: 0.21, car: 0.21, water-other: 0.18, bus: 0.18, stairs: 0.16, floor-tile: 0.15, bridge: 0.15, wall-panel: 0.14, structural-other: 0.14, straw: 0.12, gravel: 0.11, table: 0.09, house: 0.09, roof: 0.08, cage: 0.08, chair: 0.08, railing: 0.07, playingfield: 0.07, floor-wood: 0.06, wood: 0.06, banner: 0.06, light: 0.06, other: 0.86
quadrant-IV:
sky-other: 24.02, road: 14.09, clouds: 11.84, building-other: 8.08, pavement: 6.65, grass: 6.40, tree: 3.43, ground-other: 2.82, fog: 2.44, person: 2.08, mountain: 2.00, metal: 1.69, floor-other: 1.68, truck: 1.22, sea: 1.10, fence: 1.03, window-other: 0.95, dirt: 0.91, sand: 0.76, hill: 0.66, wall-other: 0.52, ceiling-other: 0.48, snow: 0.29, wall-concrete: 0.28, solid-other: 0.27, platform: 0.26, bush: 0.25, plant-other: 0.24, stairs: 0.23, river: 0.22, straw: 0.21, car: 0.21, floor-tile: 0.17, structural-other: 0.15, water-other: 0.14, bridge: 0.13, wall-panel: 0.11, bus: 0.11, house: 0.11, gravel: 0.10, playingfield: 0.10, table: 0.09, chair: 0.09, railing: 0.08, floor-wood: 0.08, roof: 0.08, textile-other: 0.08, banner: 0.07, cage: 0.07, skyscraper:

0.07, other: 0.90

20% - Increased Instances

Context across the entire instance

sky-other: 32.49, clouds: 16.16, building-other: 8.60, road: 7.91, tree: 4.76, grass: 4.12, pavement: 3.68, fog: 3.58, mountain: 2.38, ground-other: 1.68, person: 1.15, floor-other: 1.10, metal: 1.07, sea: 1.04, window-other: 0.98, hill: 0.94, ceiling-other: 0.85, truck: 0.66, fence: 0.65, wall-other: 0.64, sand: 0.62, dirt: 0.50, wall-concrete: 0.32, bush: 0.23, plant-other: 0.23, platform: 0.22, solid-other: 0.21, wall-panel: 0.21, structural-other: 0.19, river: 0.16, snow: 0.16, floor-tile: 0.15, water-other: 0.15, car: 0.12, bridge: 0.12, roof: 0.11, bus: 0.11, skyscraper: 0.11, straw: 0.10, banner: 0.10, stairs: 0.10, house: 0.10, table: 0.09, playingfield: 0.08, chair: 0.07, wood: 0.06, light: 0.06, railing: 0.05, gravel: 0.05, cage: 0.05, other: 0.75

Instances with no context: 28/5270

Context in specific instance quadrants

quadrant-I:

sky-other: 39.07, clouds: 19.33, building-other: 9.89, tree: 5.94, fog: 4.41, road: 2.75, mountain: 2.67, grass: 2.25, ceiling-other: 1.19, hill: 1.13, window-other: 1.08, sea: 1.02, pavement: 1.00, metal: 1.00, wall-other: 0.78, ground-other: 0.65, person: 0.62, floor-other: 0.54, sand: 0.46, wall-concrete: 0.40, fence: 0.36, wall-panel: 0.26, plant-other: 0.24, structural-other: 0.19, solid-other: 0.18, roof: 0.17, dirt: 0.15, bush: 0.15, truck: 0.14, water-other: 0.13, skyscraper: 0.12, banner: 0.12, river: 0.11, house: 0.09, bridge: 0.09, floor-tile: 0.08, table: 0.08, straw: 0.07, light: 0.07, wood: 0.07, platform: 0.06, window-blind: 0.06, bus: 0.06, wall-brick: 0.04, paper: 0.04, chair: 0.04, wall-wood: 0.04, snow: 0.04, branch: 0.04, textile-other: 0.03, other: 0.51

quadrant-II:

sky-other: 39.09, clouds: 19.20, building-other: 9.61, tree: 6.09, fog: 4.40, road: 2.98, mountain: 2.53, grass: 2.30, ceiling-other: 1.18, hill: 1.16, pavement: 1.14, window-other: 1.01, sea: 0.99, metal: 0.85, wall-other: 0.81, ground-other: 0.71, person: 0.61, floor-other: 0.56, sand: 0.49, wall-concrete: 0.36, fence: 0.36, wall-panel: 0.27, plant-other: 0.21, bush: 0.19, truck: 0.19, skyscraper: 0.17, solid-other: 0.16, dirt: 0.16, banner: 0.15, structural-other: 0.15, platform: 0.15, water-other: 0.15, roof: 0.14, river: 0.13, bridge: 0.12, house: 0.11, light: 0.08, table: 0.07, car: 0.06, wood: 0.06, paper: 0.05, floor-tile: 0.05, snow: 0.04, window-blind: 0.04, refrigerator: 0.04, cage: 0.04, bus: 0.04, wall-wood: 0.04, wall-brick: 0.04, railing: 0.03, other: 0.47

quadrant-III:

sky-other: 23.88, road: 15.21, clouds: 11.79, pavement: 7.13, building-other: 6.86, grass: 6.46, tree: 3.37, ground-other: 3.14, fog: 2.39, mountain: 1.87, person: 1.76, floor-other: 1.68, metal: 1.51, truck: 1.33, sea: 1.04, dirt: 1.02, fence: 0.97, window-other: 0.90, sand: 0.81, hill: 0.58, wall-other: 0.56, ceiling-other: 0.44, platform: 0.40, snow: 0.34, bush: 0.33, wall-concrete: 0.28, solid-other: 0.28, river: 0.24, plant-other: 0.22, car: 0.21, water-other: 0.18,

bus: 0.17, bridge: 0.16, floor-tile: 0.15, stairs: 0.15, wall-panel: 0.15, structural-other: 0.13, gravel: 0.13, straw: 0.12, table: 0.09, house: 0.08, railing: 0.08, cage: 0.08, chair: 0.08, roof: 0.08, playingfield: 0.07, floor-wood: 0.07, banner: 0.06, wood: 0.06, light: 0.06, other: 0.87
quadrant-IV:
sky-other: 23.57, road: 14.83, clouds: 11.64, building-other: 7.75, pavement: 6.94, grass: 6.53, tree: 3.20, ground-other: 2.94, fog: 2.39, person: 2.06, mountain: 1.95, floor-other: 1.70, metal: 1.64, truck: 1.24, sea: 1.09, fence: 1.03, window-other: 0.94, dirt: 0.93, sand: 0.79, hill: 0.62, wall-other: 0.51, ceiling-other: 0.46, snow: 0.30, wall-concrete: 0.28, solid-other: 0.27, platform: 0.26, bush: 0.25, river: 0.24, plant-other: 0.24, stairs: 0.23, car: 0.21, straw: 0.21, floor-tile: 0.17, water-other: 0.16, structural-other: 0.15, bridge: 0.13, bus: 0.12, wall-panel: 0.12, house: 0.10, gravel: 0.10, playingfield: 0.10, chair: 0.09, table: 0.09, railing: 0.08, floor-wood: 0.08, textile-other: 0.08, banner: 0.07, roof: 0.07, cage: 0.07, skyscraper: 0.07, other: 0.92

30% - Increased Instances

Context across the entire instance
sky-other: 32.47, clouds: 16.18, building-other: 8.34, road: 8.14, tree: 4.61, grass: 4.19, pavement: 3.79, fog: 3.57, mountain: 2.33, ground-other: 1.73, person: 1.13, floor-other: 1.11, metal: 1.05, sea: 1.04, window-other: 0.97, hill: 0.91, ceiling-other: 0.85, truck: 0.68, fence: 0.64, wall-other: 0.64, sand: 0.62, dirt: 0.51, wall-concrete: 0.32, bush: 0.23, plant-other: 0.23, platform: 0.22, solid-other: 0.21, wall-panel: 0.21, structural-other: 0.19, river: 0.17, snow: 0.16, floor-tile: 0.15, water-other: 0.15, car: 0.13, bridge: 0.12, playingfield: 0.12, bus: 0.11, roof: 0.11, skyscraper: 0.11, straw: 0.10, banner: 0.10, stairs: 0.10, house: 0.09, table: 0.08, chair: 0.07, railing: 0.06, light: 0.06, wood: 0.06, gravel: 0.05, floor-wood: 0.04, other: 0.76
Instances with no context: 24/5270

Context in specific instance quadrants
quadrant-I:
sky-other: 39.50, clouds: 19.54, building-other: 9.68, tree: 5.88, fog: 4.47, mountain: 2.63, road: 2.63, grass: 2.17, ceiling-other: 1.21, hill: 1.11, window-other: 1.08, sea: 1.02, metal: 0.98, pavement: 0.96, wall-other: 0.78, ground-other: 0.64, person: 0.61, floor-other: 0.52, sand: 0.45, wall-concrete: 0.39, fence: 0.36, wall-panel: 0.26, plant-other: 0.23, structural-other: 0.19, solid-other: 0.18, roof: 0.17, dirt: 0.15, bush: 0.15, truck: 0.14, water-other: 0.12, skyscraper: 0.12, banner: 0.11, river: 0.11, bridge: 0.08, house: 0.08, floor-tile: 0.07, table: 0.07, light: 0.07, straw: 0.07, wood: 0.07, bus: 0.06, window-blind: 0.06, platform: 0.05, paper: 0.04, wall-brick: 0.04, wall-wood: 0.04, chair: 0.04, snow: 0.03, branch: 0.03, textile-other: 0.03, other: 0.51
quadrant-II:
sky-other: 39.44, clouds: 19.48, building-other: 9.38, tree: 5.99, fog: 4.47, road: 2.85, mountain: 2.51, grass: 2.23, ceiling-other: 1.20, hill: 1.16, pavement: 1.11, window-other: 1.00, sea: 0.99, metal: 0.82, wall-other: 0.80, ground-other: 0.69, person: 0.59, floor-other: 0.55, sand:

0.48, wall-concrete: 0.36, fence: 0.35, wall-panel: 0.27, plant-other: 0.20, truck: 0.19, bush: 0.18, skyscraper: 0.18, solid-other: 0.16, dirt: 0.15, structural-other: 0.15, water-other: 0.15, banner: 0.14, platform: 0.14, roof: 0.13, river: 0.13, bridge: 0.12, house: 0.11, light: 0.08, table: 0.07, wood: 0.06, car: 0.06, paper: 0.05, floor-tile: 0.05, snow: 0.04, playingfield: 0.04, window-blind: 0.04, bus: 0.04, refrigerator: 0.04, wall-brick: 0.04, tv: 0.04, cage: 0.03, other: 0.48
quadrant-III:
sky-other: 23.62, road: 15.65, clouds: 11.68, pavement: 7.33, grass: 6.66, building-other: 6.60, ground-other: 3.22, tree: 3.18, fog: 2.32, mountain: 1.84, person: 1.73, floor-other: 1.69, metal: 1.46, truck: 1.33, dirt: 1.04, sea: 1.03, fence: 0.97, window-other: 0.89, sand: 0.83, wall-other: 0.58, hill: 0.55, ceiling-other: 0.43, platform: 0.41, snow: 0.36, bush: 0.33, wall-concrete: 0.29, solid-other: 0.27, river: 0.25, plant-other: 0.22, car: 0.22, water-other: 0.18, bus: 0.17, bridge: 0.16, floor-tile: 0.16, stairs: 0.15, wall-panel: 0.15, gravel: 0.14, structural-other: 0.13, straw: 0.12, table: 0.09, railing: 0.09, playingfield: 0.09, chair: 0.08, cage: 0.08, house: 0.08, roof: 0.07, floor-wood: 0.07, banner: 0.06, light: 0.06, wood: 0.05, other: 0.87
quadrant-IV:
sky-other: 23.32, road: 15.30, clouds: 11.60, building-other: 7.41, pavement: 7.12, grass: 6.73, tree: 3.06, ground-other: 3.00, fog: 2.35, person: 2.02, mountain: 1.89, floor-other: 1.71, metal: 1.58, truck: 1.27, sea: 1.09, fence: 1.01, dirt: 0.96, window-other: 0.92, sand: 0.80, hill: 0.60, wall-other: 0.51, ceiling-other: 0.44, snow: 0.31, wall-concrete: 0.28, platform: 0.27, solid-other: 0.26, bush: 0.25, river: 0.25, plant-other: 0.24, stairs: 0.22, car: 0.21, straw: 0.21, floor-tile: 0.18, structural-other: 0.16, water-other: 0.16, bridge: 0.13, bus: 0.13, wall-panel: 0.11, gravel: 0.10, house: 0.10, playingfield: 0.10, chair: 0.09, table: 0.09, railing: 0.08, floor-wood: 0.08, textile-other: 0.08, banner: 0.07, cage: 0.07, roof: 0.07, skyscraper: 0.07, other: 0.93

40% - Increased Instances

Context across the entire instance
sky-other: 32.49, clouds: 16.21, road: 8.34, building-other: 8.08, tree: 4.48, grass: 4.27, pavement: 3.89, fog: 3.56, mountain: 2.29, ground-other: 1.77, floor-other: 1.12, person: 1.11, metal: 1.04, sea: 1.04, window-other: 0.97, hill: 0.88, ceiling-other: 0.85, truck: 0.69, fence: 0.64, wall-other: 0.63, sand: 0.63, dirt: 0.52, wall-concrete: 0.32, bush: 0.23, plant-other: 0.22, platform: 0.22, solid-other: 0.21, wall-panel: 0.21, structural-other: 0.18, river: 0.17, snow: 0.17, floor-tile: 0.15, water-other: 0.15, car: 0.13, bridge: 0.12, playingfield: 0.12, bus: 0.11, roof: 0.11, skyscraper: 0.11, straw: 0.10, banner: 0.10, stairs: 0.09, house: 0.09, table: 0.08, chair: 0.07, railing: 0.06, gravel: 0.06, light: 0.06, wood: 0.05, floor-wood: 0.04, other: 0.76
Instances with no context: 22/5270

Context in specific instance quadrants
quadrant-I:
sky-other: 39.93, clouds: 19.74, building-other: 9.45, tree: 5.80, fog: 4.53, mountain: 2.59,

road: 2.52, grass: 2.09, ceiling-other: 1.24, hill: 1.10, window-other: 1.08, sea: 1.02, metal: 0.96, pavement: 0.94, wall-other: 0.76, ground-other: 0.63, person: 0.59, floor-other: 0.51, sand: 0.44, wall-concrete: 0.39, fence: 0.35, wall-panel: 0.26, plant-other: 0.22, structural-other: 0.19, solid-other: 0.18, roof: 0.18, bush: 0.15, dirt: 0.14, truck: 0.13, skyscraper: 0.12, water-other: 0.12, banner: 0.11, river: 0.10, bridge: 0.08, house: 0.08, light: 0.07, floor-tile: 0.07, table: 0.07, wood: 0.07, straw: 0.06, bus: 0.06, window-blind: 0.06, platform: 0.05, paper: 0.04, wall-brick: 0.04, chair: 0.04, wall-wood: 0.04, textile-other: 0.03, snow: 0.03, branch: 0.03, other: 0.51

quadrant-II:

sky-other: 39.81, clouds: 19.74, building-other: 9.18, tree: 5.90, fog: 4.54, road: 2.72, mountain: 2.48, grass: 2.15, ceiling-other: 1.25, hill: 1.15, pavement: 1.07, window-other: 0.99, sea: 0.97, wall-other: 0.80, metal: 0.80, ground-other: 0.68, person: 0.55, floor-other: 0.54, sand: 0.47, wall-concrete: 0.36, fence: 0.34, wall-panel: 0.27, truck: 0.20, plant-other: 0.19, skyscraper: 0.18, bush: 0.17, solid-other: 0.16, dirt: 0.15, water-other: 0.15, structural-other: 0.14, platform: 0.14, banner: 0.14, roof: 0.13, bridge: 0.13, river: 0.12, house: 0.11, light: 0.09, table: 0.07, car: 0.06, wood: 0.06, paper: 0.05, floor-tile: 0.05, snow: 0.04, bus: 0.04, wall-brick: 0.04, playingfield: 0.04, window-blind: 0.04, refrigerator: 0.04, tv: 0.04, railing: 0.03, other: 0.47

quadrant-III:

sky-other: 23.34, road: 16.04, clouds: 11.58, pavement: 7.49, grass: 6.91, building-other: 6.34, ground-other: 3.28, tree: 3.02, fog: 2.27, mountain: 1.80, floor-other: 1.72, person: 1.70, metal: 1.46, truck: 1.35, dirt: 1.06, sea: 1.03, fence: 0.97, window-other: 0.88, sand: 0.84, wall-other: 0.56, hill: 0.51, platform: 0.42, ceiling-other: 0.42, snow: 0.37, bush: 0.33, wall-concrete: 0.29, solid-other: 0.28, river: 0.26, car: 0.23, plant-other: 0.22, water-other: 0.17, bus: 0.17, floor-tile: 0.16, bridge: 0.16, wall-panel: 0.15, gravel: 0.15, stairs: 0.15, structural-other: 0.13, straw: 0.12, railing: 0.10, table: 0.09, playingfield: 0.09, chair: 0.08, cage: 0.08, house: 0.07, floor-wood: 0.07, roof: 0.06, banner: 0.06, floor-stone: 0.05, light: 0.05, other: 0.88

quadrant-IV:

sky-other: 23.08, road: 15.66, clouds: 11.55, pavement: 7.30, building-other: 7.08, grass: 6.96, ground-other: 3.07, tree: 2.94, fog: 2.31, person: 2.01, mountain: 1.85, floor-other: 1.73, metal: 1.56, truck: 1.28, sea: 1.10, fence: 1.00, dirt: 0.99, window-other: 0.91, sand: 0.81, hill: 0.56, wall-other: 0.50, ceiling-other: 0.43, snow: 0.32, wall-concrete: 0.28, platform: 0.27, solid-other: 0.26, river: 0.26, bush: 0.25, plant-other: 0.25, stairs: 0.21, car: 0.21, straw: 0.20, floor-tile: 0.18, structural-other: 0.16, water-other: 0.16, bus: 0.13, bridge: 0.13, wall-panel: 0.11, gravel: 0.11, chair: 0.10, playingfield: 0.10, house: 0.10, table: 0.09, railing: 0.08, floor-wood: 0.08, textile-other: 0.08, banner: 0.07, cage: 0.07, roof: 0.07, skyscraper: 0.06, other: 0.94

B.3 *PASCAL-Context59*

B.3.1 Images

Semantic Context

Context across the entire image

sky: 58.02, ground: 12.04, grass: 7.44, building: 6.09, tree: 3.97, road: 3.53, mountain: 1.99, floor: 1.03, water: 0.92, wall: 0.90, person: 0.84, ceiling: 0.81, snow: 0.60, fence: 0.54, truck: 0.50, car: 0.21, boat: 0.18, window: 0.15, sign: 0.08, food: 0.06, chair: 0.02, light: 0.02, bus: 0.01, bench: 0.01, table: 0.01, shelves: 0.01
images with no context: 2/582

Context in specific image quadrants

quadrant-I:
sky: 77.46, building: 7.43, tree: 4.35, mountain: 2.56, grass: 1.72, ground: 1.45, ceiling: 1.33, wall: 1.19, person: 0.54, road: 0.51, truck: 0.26, water: 0.25, fence: 0.24, light: 0.20, window: 0.19, snow: 0.11, floor: 0.11, car: 0.04, sign: 0.02, boat: 0.01, door: 0.01, bus: 0.01, shelves: 0.01
quadrant-II:
sky: 78.07, building: 6.43, tree: 4.34, mountain: 2.61, grass: 1.85, ground: 1.53, wall: 1.48, ceiling: 1.06, road: 0.62, person: 0.50, water: 0.28, window: 0.27, truck: 0.20, light: 0.19, floor: 0.17, fence: 0.17, snow: 0.08, car: 0.04, sign: 0.04, boat: 0.03, bus: 0.03, cabinet: 0.01
quadrant-III:
sky: 37.19, ground: 22.98, grass: 13.52, road: 7.03, building: 5.47, tree: 3.64, floor: 1.87, water: 1.49, mountain: 1.42, person: 1.11, snow: 0.95, fence: 0.87, truck: 0.70, wall: 0.45, car: 0.39, boat: 0.38, window: 0.17, food: 0.15, sign: 0.11, chair: 0.05, table: 0.02, cloth: 0.01, bench: 0.01, cabinet: 0.01, light: 0.01
quadrant-IV:
sky: 36.69, ground: 23.04, grass: 13.23, road: 6.82, building: 5.32, tree: 3.65, floor: 2.20, water: 1.69, mountain: 1.44, person: 1.38, fence: 0.92, snow: 0.90, truck: 0.90, car: 0.44, wall: 0.35, boat: 0.30, sign: 0.19, ceiling: 0.17, food: 0.14, window: 0.10, chair: 0.03, bench: 0.02, table: 0.02, shelves: 0.02, rock: 0.01, wood: 0.01, pottedplant: 0.01, bus: 0.01
image quadrants with no context:
quadrant-I 5/582
quadrant-II: 7/582
quadrant-III: 5/582
quadrant-IV: 8/582

Label-neighborhood

left:
sky: 52.91, ground: 11.12, building: 9.28, grass: 8.01, tree: 5.31, road: 3.59, mountain: 2.32,

person: 1.28, wall: 1.22, floor: 1.14, water: 0.81, snow: 0.66, truck: 0.45, fence: 0.43, ceiling: 0.38, window: 0.26, car: 0.20, boat: 0.12, light: 0.08, sign: 0.05, bench: 0.02, bus: 0.01, shelves: 0.01, table: 0.01
up:
sky: 59.97, building: 11.64, tree: 7.12, grass: 5.36, ground: 5.28, mountain: 3.14, wall: 1.55, road: 1.42, water: 0.77, person: 0.71, floor: 0.65, ceiling: 0.61, snow: 0.44, fence: 0.28, window: 0.18, truck: 0.15, light: 0.15, car: 0.12, boat: 0.05, sign: 0.04, bus: 0.03
right:
sky: 53.06, ground: 10.74, building: 9.96, grass: 7.88, tree: 5.23, road: 3.44, mountain: 2.34, person: 1.39, floor: 1.24, wall: 1.02, water: 0.78, snow: 0.74, ceiling: 0.44, fence: 0.40, truck: 0.30, car: 0.29, window: 0.27, boat: 0.05, sign: 0.05, bus: 0.01, light: 0.01, shelves: 0.01
down:
sky: 44.67, ground: 15.89, grass: 11.64, building: 9.34, road: 5.04, tree: 4.26, floor: 1.55, mountain: 1.41, person: 1.15, water: 1.04, snow: 0.85, wall: 0.80, truck: 0.62, car: 0.48, fence: 0.34, boat: 0.21, window: 0.13, ceiling: 0.12, sign: 0.05, chair: 0.03, bench: 0.02, table: 0.02, bus: 0.02

B.3.2 Instances

Default Instance Size

Context across the entire instance
sky: 53.97, building: 10.60, ground: 10.40, grass: 6.26, tree: 5.21, road: 2.73, mountain: 2.37, wall: 2.34, floor: 1.36, person: 0.87, ceiling: 0.85, water: 0.74, truck: 0.62, snow: 0.46, fence: 0.39, car: 0.27, window: 0.26, light: 0.16, boat: 0.07, sign: 0.04, bus: 0.02
Instances with no context: 5/811

Context in specific instance quadrants
quadrant-I:
sky: 64.62, building: 11.53, tree: 6.51, mountain: 3.37, ground: 3.32, grass: 3.29, wall: 2.71, ceiling: 1.00, road: 0.70, person: 0.63, floor: 0.63, water: 0.59, snow: 0.30, fence: 0.30, window: 0.27, truck: 0.15, car: 0.06, light: 0.02, sign: 0.01, bus: 0.01, shelves: 0.01
quadrant-II:
sky: 65.01, building: 10.45, tree: 6.32, grass: 3.82, mountain: 3.35, ground: 3.13, wall: 2.84, ceiling: 1.08, road: 0.78, floor: 0.65, water: 0.65, person: 0.52, window: 0.33, snow: 0.32, light: 0.28, fence: 0.26, truck: 0.10, car: 0.04, bus: 0.04, boat: 0.01
quadrant-III:
sky: 40.13, ground: 18.89, building: 10.51, grass: 10.31, road: 5.49, tree: 3.66, floor: 1.90, wall: 1.60, person: 1.35, mountain: 1.26, truck: 1.09, water: 0.90, car: 0.72, snow: 0.60, fence: 0.50, ceiling: 0.47, boat: 0.27, window: 0.20, sign: 0.07, light: 0.03, bench: 0.03, cloth: 0.01, chair: 0.01, shelves: 0.01
quadrant-IV:
sky: 39.88, ground: 19.61, building: 10.50, grass: 9.97, road: 5.33, tree: 4.07, floor: 2.19,

wall: 1.57, person: 1.35, mountain: 1.18, water: 1.00, truck: 0.98, car: 0.57, snow: 0.52, fence: 0.51, ceiling: 0.35, window: 0.15, boat: 0.12, sign: 0.08, bench: 0.02, bus: 0.01, table: 0.01, light: 0.01, pottedplant: 0.01

10% - Increased Instances

Context across the entire instance
sky: 53.81, ground: 10.86, building: 10.46, grass: 6.19, tree: 5.07, road: 2.87, wall: 2.32, mountain: 2.30, floor: 1.37, ceiling: 0.87, person: 0.83, water: 0.73, truck: 0.59, snow: 0.46, fence: 0.41, car: 0.27, window: 0.25, light: 0.16, boat: 0.08, sign: 0.05, bus: 0.02
Instances with no context: 4/811

Context in specific instance quadrants
quadrant-I:
sky: 65.01, building: 11.42, tree: 6.39, grass: 3.38, mountain: 3.31, ground: 3.20, wall: 2.63, ceiling: 1.08, road: 0.65, floor: 0.60, water: 0.57, person: 0.56, snow: 0.29, fence: 0.28, window: 0.27, truck: 0.15, light: 0.14, car: 0.05, sign: 0.01, bus: 0.01
quadrant-II:
sky: 65.40, building: 10.52, tree: 6.19, grass: 3.76, mountain: 3.27, ground: 3.02, wall: 2.82, ceiling: 1.10, road: 0.73, floor: 0.63, water: 0.62, person: 0.47, light: 0.41, window: 0.33, snow: 0.28, fence: 0.26, truck: 0.09, car: 0.04, bus: 0.03, boat: 0.01, sign: 0.01
quadrant-III:
sky: 39.61, ground: 19.84, grass: 10.10, building: 10.08, road: 5.79, tree: 3.55, floor: 1.91, wall: 1.75, person: 1.28, mountain: 1.20, truck: 1.07, water: 0.87, car: 0.70, snow: 0.61, fence: 0.54, ceiling: 0.45, boat: 0.27, window: 0.21, sign: 0.07, bench: 0.03, light: 0.03, chair: 0.02, cloth: 0.01, shelves: 0.01
quadrant-IV:
sky: 39.43, ground: 20.63, building: 10.01, grass: 9.66, road: 5.72, tree: 3.81, floor: 2.20, wall: 1.55, person: 1.32, mountain: 1.12, water: 0.97, truck: 0.96, snow: 0.56, fence: 0.55, car: 0.53, ceiling: 0.46, window: 0.16, boat: 0.13, sign: 0.12, bench: 0.03, bus: 0.02, table: 0.02, shelves: 0.01, chair: 0.01, pottedplant: 0.01, light: 0.01

20% - Increased Instances

Context across the entire instance
sky: 53.74, ground: 11.28, building: 10.14, grass: 6.24, tree: 4.93, road: 2.99, wall: 2.29, mountain: 2.25, floor: 1.38, ceiling: 0.90, person: 0.80, water: 0.72, truck: 0.61, snow: 0.47, fence: 0.42, car: 0.27, window: 0.25, light: 0.13, boat: 0.09, sign: 0.06, bench: 0.02, bus: 0.01, shelves: 0.01
Instances with no context: 4/811

Context in specific instance quadrants
quadrant-I:

sky: 65.51, building: 11.37, tree: 6.27, mountain: 3.43, grass: 3.24, ground: 3.04, wall: 2.55, ceiling: 1.16, road: 0.61, floor: 0.59, water: 0.55, person: 0.51, window: 0.27, snow: 0.26, fence: 0.26, truck: 0.15, light: 0.14, car: 0.05, sign: 0.01, bus: 0.01
quadrant-II:
sky: 66.03, building: 10.43, tree: 6.01, grass: 3.62, mountain: 3.24, wall: 2.88, ground: 2.87, ceiling: 1.13, road: 0.69, floor: 0.63, water: 0.59, person: 0.46, light: 0.40, window: 0.33, fence: 0.25, snow: 0.25, truck: 0.09, car: 0.03, bus: 0.03, sign: 0.02, boat: 0.01
quadrant-III:
sky: 39.08, ground: 20.74, grass: 10.08, building: 9.60, road: 5.97, tree: 3.43, floor: 2.05, wall: 1.76, person: 1.25, mountain: 1.13, truck: 1.06, water: 0.85, car: 0.67, snow: 0.63, fence: 0.58, ceiling: 0.43, boat: 0.27, window: 0.23, sign: 0.08, bench: 0.03, light: 0.03, chair: 0.02, cloth: 0.01, shelves: 0.01
quadrant-IV:
sky: 38.94, ground: 21.36, grass: 9.80, building: 9.48, road: 6.05, tree: 3.60, floor: 2.22, wall: 1.53, person: 1.27, truck: 1.06, mountain: 1.04, water: 0.96, fence: 0.60, snow: 0.58, car: 0.51, ceiling: 0.45, window: 0.17, boat: 0.14, sign: 0.12, bench: 0.04, shelves: 0.02, bus: 0.02, table: 0.02, chair: 0.01, pottedplant: 0.01, rock: 0.01

30% - Increased Instances

Context across the entire instance
sky: 53.74, ground: 11.58, building: 9.88, grass: 6.30, tree: 4.77, road: 3.06, wall: 2.26, mountain: 2.24, floor: 1.35, ceiling: 0.93, person: 0.82, water: 0.71, truck: 0.61, snow: 0.47, fence: 0.43, car: 0.28, window: 0.25, light: 0.11, boat: 0.09, sign: 0.06, bench: 0.02, bus: 0.01, shelves: 0.01
Instances with no context: 4/811

Context in specific instance quadrants
quadrant-I:
sky: 66.14, building: 11.18, tree: 6.12, mountain: 3.46, grass: 3.15, ground: 2.91, wall: 2.49, ceiling: 1.23, road: 0.60, floor: 0.58, water: 0.54, person: 0.49, window: 0.27, fence: 0.25, snow: 0.24, light: 0.14, truck: 0.14, car: 0.04, sign: 0.01, bus: 0.01
quadrant-II:
sky: 66.66, building: 10.27, tree: 5.81, grass: 3.53, mountain: 3.25, wall: 2.83, ground: 2.75, ceiling: 1.19, road: 0.66, floor: 0.62, water: 0.57, person: 0.45, light: 0.40, window: 0.34, fence: 0.25, snow: 0.22, truck: 0.09, car: 0.03, bus: 0.03, sign: 0.02, boat: 0.01
quadrant-III:
sky: 38.69, ground: 21.43, grass: 10.18, building: 9.20, road: 6.10, tree: 3.34, floor: 2.05, wall: 1.77, person: 1.22, mountain: 1.07, truck: 1.06, water: 0.84, car: 0.67, snow: 0.65, fence: 0.59, ceiling: 0.41, boat: 0.27, window: 0.23, sign: 0.09, bench: 0.06, light: 0.02, chair: 0.02, cloth: 0.01, shelves: 0.01
quadrant-IV:
sky: 38.59, ground: 22.00, grass: 9.97, building: 9.13, road: 6.14, tree: 3.44, floor: 2.20, wall:

1.52, person: 1.28, truck: 1.04, mountain: 0.97, water: 0.95, fence: 0.63, snow: 0.60, car: 0.51, ceiling: 0.43, window: 0.17, boat: 0.16, sign: 0.11, bench: 0.04, shelves: 0.03, table: 0.02, bus: 0.02, chair: 0.02, rock: 0.01, pottedplant: 0.01

40% - Increased Instances

Context across the entire instance
sky: 53.70, ground: 11.85, building: 9.74, grass: 6.35, tree: 4.61, road: 3.11, wall: 2.25, mountain: 2.23, floor: 1.32, ceiling: 0.95, person: 0.82, water: 0.70, truck: 0.61, snow: 0.46, fence: 0.43, car: 0.28, window: 0.24, boat: 0.10, light: 0.09, sign: 0.06, bench: 0.02, bus: 0.01, chair: 0.01, shelves: 0.01
Instances with no context: 3/811

Context in specific instance quadrants
quadrant-I:
sky: 66.62, building: 11.13, tree: 5.93, mountain: 3.48, grass: 3.05, ground: 2.82, wall: 2.46, ceiling: 1.27, road: 0.58, floor: 0.55, water: 0.53, person: 0.48, window: 0.27, fence: 0.24, snow: 0.22, light: 0.14, truck: 0.14, car: 0.04, sign: 0.02, bus: 0.01
quadrant-II:
sky: 67.19, building: 10.24, tree: 5.60, grass: 3.39, mountain: 3.27, wall: 2.80, ground: 2.67, ceiling: 1.24, road: 0.63, floor: 0.60, water: 0.55, person: 0.44, light: 0.39, window: 0.33, fence: 0.26, snow: 0.21, truck: 0.09, car: 0.03, sign: 0.02, bus: 0.02, boat: 0.01
quadrant-III:
sky: 38.46, ground: 22.04, grass: 10.36, building: 8.81, road: 6.18, tree: 3.25, floor: 2.05, wall: 1.79, person: 1.15, mountain: 1.03, truck: 1.00, water: 0.83, car: 0.66, snow: 0.65, fence: 0.59, ceiling: 0.38, boat: 0.28, window: 0.23, sign: 0.10, bench: 0.07, chair: 0.03, light: 0.02, cloth: 0.01, table: 0.01, motorbike: 0.01
quadrant-IV:
sky: 38.30, ground: 22.63, grass: 10.11, building: 8.78, road: 6.22, tree: 3.31, floor: 2.19, wall: 1.52, person: 1.28, truck: 1.01, water: 0.96, mountain: 0.92, fence: 0.65, snow: 0.60, car: 0.50, ceiling: 0.41, boat: 0.18, window: 0.18, sign: 0.11, bench: 0.03, shelves: 0.02, table: 0.02, chair: 0.02, bus: 0.01, rock: 0.01, pottedplant: 0.01

B.4 SemanticAircraftV1

B.4.1 Images

Semantic Context

Context across the entire image
sky: 53.21, pavement: 15.90, soil: 6.86, void: 5.35, building: 4.20, plant: 3.83, indoor: 2.87, elevation: 2.54, waterbody: 1.82, object: 1.28, person: 1.19, vehicle: 1
images with no context: 0/3801

Context in specific image quadrants

quadrant-I:
sky: 74.53, building: 4.54, plant: 4.01, void: 3.91, indoor: 3.67, pavement: 2.64, elevation: 2.51, soil: 1.58, waterbody: 0.84, object: 0.75, person: 0.71, vehicle: 0.31
quadrant-II:
sky: 74.67, building: 4.33, plant: 4.03, void: 3.71, indoor: 3.68, pavement: 2.99, elevation: 2.56, soil: 1.63, waterbody: 0.82, person: 0.63, object: 0.57, vehicle: 0.37
quadrant-III:
sky: 30.68, pavement: 30.15, soil: 12.41, void: 6.74, building: 3.89, plant: 3.58, waterbody: 2.79, elevation: 2.49, indoor: 2.11, object: 1.89, vehicle: 1.66, person: 1.61
quadrant-IV:
sky: 30.66, pavement: 29.47, soil: 12.37, void: 7.03, building: 4.08, plant: 3.58, waterbody: 2.87, elevation: 2.48, indoor: 2.04, object: 2.04, person: 1.81, vehicle: 1.56
image quadrants with no context:
quadrant-I: 6/3801
quadrant-II: 3/3801
quadrant-III: 4/3801
quadrant-IV: 8/3801

Label-neighborhood

left:
sky: 52.84, pavement: 14.73, building: 6.85, soil: 6.10, plant: 5.19, indoor: 3.25, elevation: 2.88, person: 2.78, object: 2.25, waterbody: 1.68, vehicle: 0.73
up:
sky: 58.31, building: 9.13, pavement: 8.71, plant: 6.85, soil: 4.34, indoor: 3.92, elevation: 3.44, waterbody: 1.68, object: 1.59, person: 0.98, vehicle: 0.30
right:
sky: 52.58, pavement: 13.93, building: 7.46, soil: 6.06, plant: 5.17, indoor: 3.46, person: 3.13, elevation: 3.01, object: 2.10, waterbody: 1.67, vehicle: 0.68
down:
sky: 41.79, pavement: 24.86, soil: 9.57, building: 5.77, plant: 4.11, object: 2.89, indoor: 2.74, elevation: 2.29, person: 2.09, waterbody: 1.85, vehicle: 1.29

B.4.2 Instances

Default Instance Size

Context across the entire instance

sky: 50.72, pavement: 12.71, building: 9.75, soil: 5.53, plant: 5.26, void: 4.34, indoor: 3.51, elevation: 3.30, object: 1.61, waterbody: 1.38, person: 1.12, vehicle: 0.77
Instances with no context: 14/6354

Context in specific instance quadrants
quadrant-I:
sky: 59.59, building: 10.85, plant: 6.24, pavement: 4.93, indoor: 4.20, elevation: 3.82, void: 3.78, soil: 3.22, waterbody: 1.32, object: 1.23, person: 0.62, vehicle: 0.20
quadrant-II:
sky: 59.59, building: 10.51, plant: 6.36, pavement: 5.35, indoor: 4.00, void: 3.79, elevation: 3.72, soil: 3.33, waterbody: 1.32, object: 1.20, person: 0.59, vehicle: 0.24
quadrant-III:
sky: 37.73, pavement: 23.97, soil: 8.65, building: 8.35, void: 5.03, plant: 4.00, indoor: 2.91, elevation: 2.38, object: 2.37, person: 1.65, vehicle: 1.54, waterbody: 1.42
quadrant-IV:
sky: 37.46, pavement: 23.29, building: 9.05, soil: 8.63, void: 5.20, plant: 3.87, indoor: 2.94, elevation: 2.44, object: 2.38, person: 1.82, waterbody: 1.52, vehicle: 1.39

10% - Increased Instances

Context across the entire instance
sky: 50.50, pavement: 13.33, building: 9.50, soil: 5.55, plant: 5.10, void: 4.46, indoor: 3.52, elevation: 3.18, object: 1.64, waterbody: 1.39, person: 1.04, vehicle: 0.79
Instances with no context: 11/6354

Context in specific instance quadrants
quadrant-I:
sky: 60.22, building: 10.66, plant: 6.10, pavement: 4.76, indoor: 4.19, void: 3.85, elevation: 3.72, soil: 3.18, waterbody: 1.35, object: 1.21, person: 0.55, vehicle: 0.20
quadrant-II:
sky: 60.19, building: 10.34, plant: 6.27, pavement: 5.16, indoor: 4.02, void: 3.92, elevation: 3.62, soil: 3.24, waterbody: 1.31, object: 1.19, person: 0.51, vehicle: 0.22
quadrant-III:
sky: 36.96, pavement: 25.11, soil: 8.65, building: 8.03, void: 5.32, plant: 3.81, indoor: 2.92, object: 2.44, elevation: 2.26, vehicle: 1.55, person: 1.53, waterbody: 1.42
quadrant-IV:
sky: 36.72, pavement: 24.42, building: 8.69, soil: 8.61, void: 5.48, plant: 3.66, indoor: 2.94, object: 2.48, elevation: 2.37, person: 1.72, waterbody: 1.52, vehicle: 1.39

20% - Increased Instances

Context across the entire instance
sky: 50.34, pavement: 13.90, building: 9.18, soil: 5.61, plant: 4.93, void: 4.62, indoor: 3.51, elevation: 3.08, object: 1.64, waterbody: 1.39, person: 1.00, vehicle: 0.8
Instances with no context: 6/6354

Context in specific instance quadrants
quadrant-I:
sky: 60.85, building: 10.43, plant: 6.00, pavement: 4.59, indoor: 4.16, void: 3.94, elevation: 3.67, soil: 3.06, waterbody: 1.35, object: 1.23, person: 0.52, vehicle: 0.19
quadrant-II:
sky: 60.88, building: 10.17, plant: 6.12, pavement: 4.92, indoor: 4.04, void: 3.97, elevation: 3.58, soil: 3.15, waterbody: 1.29, object: 1.17, person: 0.49, vehicle: 0.22
quadrant-III:
sky: 36.29, pavement: 26.21, soil: 8.77, building: 7.61, void: 5.61, plant: 3.62, indoor: 2.90, object: 2.40, elevation: 2.16, vehicle: 1.54, person: 1.48, waterbody: 1.41
quadrant-IV:
sky: 35.98, pavement: 25.55, soil: 8.76, building: 8.29, void: 5.74, plant: 3.44, indoor: 2.88, object: 2.45, elevation: 2.27, person: 1.70, waterbody: 1.54, vehicle: 1.40

30% - Increased Instances

Context across the entire instance
sky: 50.29, pavement: 14.30, building: 8.91, soil: 5.68, plant: 4.79, void: 4.71, indoor: 3.49, elevation: 3.01, object: 1.62, waterbody: 1.40, person: 0.98, vehicle: 0.82
Instances with no context: 5/6354

Context in specific instance quadrants
quadrant-I:
sky: 61.46, building: 10.21, plant: 5.93, pavement: 4.40, indoor: 4.16, void: 4.04, elevation: 3.62, soil: 2.94, waterbody: 1.35, object: 1.20, person: 0.50, vehicle: 0.19
quadrant-II:
sky: 61.53, building: 9.94, plant: 6.01, pavement: 4.73, indoor: 4.04, void: 4.04, elevation: 3.55, soil: 3.04, waterbody: 1.29, object: 1.13, person: 0.47, vehicle: 0.22
quadrant-III:
sky: 35.80, pavement: 26.89, soil: 8.98, building: 7.30, void: 5.83, plant: 3.46, indoor: 2.89, object: 2.36, elevation: 2.08, vehicle: 1.55, person: 1.45, waterbody: 1.41
quadrant-IV:
sky: 35.51, pavement: 26.30, soil: 8.97, building: 7.92, void: 5.99, plant: 3.30, indoor: 2.83, object: 2.38, elevation: 2.19, person: 1.66, waterbody: 1.54, vehicle: 1.42

40% - Increased Instances

Context across the entire instance
sky: 50.30, pavement: 14.63, building: 8.64, soil: 5.76, void: 4.82, plant: 4.66, indoor: 3.47, elevation: 2.95, object: 1.60, waterbody: 1.39, person: 0.97, vehicle: 0.81 Instances with no context: 2/6354

Context in specific instance quadrants
quadrant-I:
sky: 62.09, building: 9.97, plant: 5.83, pavement: 4.26, indoor: 4.17, void: 4.08, elevation: 3.57, soil: 2.84, waterbody: 1.34, object: 1.17, person: 0.49, vehicle: 0.18
quadrant-II:
sky: 62.13, building: 9.73, plant: 5.88, pavement: 4.55, void: 4.14, indoor: 4.06, elevation: 3.52, soil: 2.94, waterbody: 1.27, object: 1.11, person: 0.46, vehicle: 0.22
quadrant-III:
sky: 35.38, pavement: 27.46, soil: 9.23, building: 7.00, void: 6.01, plant: 3.31, indoor: 2.85, object: 2.36, elevation: 2.01, vehicle: 1.58, person: 1.41, waterbody: 1.40
quadrant-IV:
sky: 35.15, pavement: 26.90, soil: 9.21, building: 7.58, void: 6.13, plant: 3.18, indoor: 2.79, object: 2.35, elevation: 2.10, person: 1.63, waterbody: 1.54, vehicle: 1.44

B.5 SemanticAircraft

B.5.1 Instances

Semantic Context

Context across the entire instance
sky: 57.21, pavement: 15.81, building: 7.48, soil: 6.23, plant: 5.11, elevation: 3.18, object: 1.54, waterbody: 1.37, person: 1.15, vehicle: 0.91, indoor: 0.01

Context in specific instance quadrants
quadrant-I:
sky: 71.56, building: 8.60, plant: 6.31, pavement: 4.33, elevation: 3.68, soil: 2.83, waterbody: 1.19, object: 0.78, person: 0.55, vehicle: 0.15, indoor: 0.02
quadrant-II:
sky: 71.67, building: 8.12, plant: 6.26, pavement: 4.56, elevation: 3.63, soil: 2.89, waterbody: 1.17, object: 0.89, person: 0.56, vehicle: 0.23, indoor: 0.02
quadrant-III:
sky: 38.47, pavement: 30.90, soil: 10.38, building: 6.13, plant: 3.82, object: 2.69, elevation: 2.38, vehicle: 1.89, person: 1.78, waterbody: 1.50, indoor: 0.07
quadrant-IV:
sky: 38.36, pavement: 30.31, soil: 10.30, building: 6.87, plant: 3.74, object: 2.69, elevation: 2.34, person: 1.97, vehicle: 1.78, waterbody: 1.61, indoor: 0.02

Label-neighborhood

left:
sky: 53.75, pavement: 15.55, building: 8.38, soil: 6.69, plant: 5.78, elevation: 3.04, object: 2.55, person: 2.28, waterbody: 1.33, vehicle: 0.61, indoor: 0.04

up:
sky: 59.21, building: 11.08, pavement: 9.35, plant: 7.70, soil: 4.85, elevation: 3.70, object: 1.77, waterbody: 1.30, person: 0.77, vehicle: 0.24, indoor: 0.04
right:
sky: 53.66, pavement: 14.79, building: 9.21, soil: 6.56, plant: 5.79, elevation: 3.08, person: 2.69, object: 2.26, waterbody: 1.31, vehicle: 0.60, indoor: 0.04
down:
sky: 41.37, pavement: 27.11, soil: 10.48, building: 6.97, plant: 4.35, object: 3.29, elevation: 2.18, person: 1.63, waterbody: 1.40, vehicle: 1.19, indoor: 0.03

B.5.2 Quadrants

Context across the entire quadrants
sky: 58.57, pavement: 17.44, soil: 7.76, building: 4.26, plant: 4.04, elevation: 2.76, waterbody: 1.88, object: 1.19, person: 1.08, vehicle: indoor: 1.02
Context in quadrants of quadrants of original images are omitted

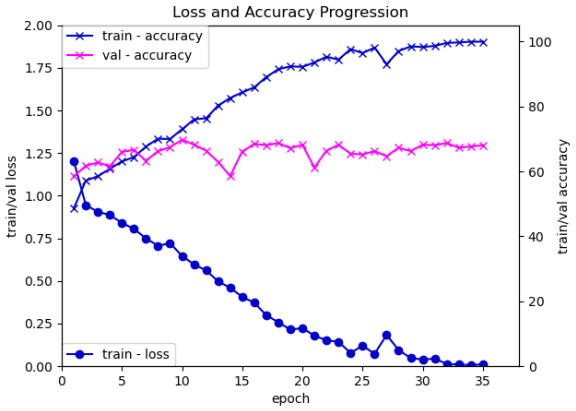
Label-neighborhood

left:
sky: 45.38, pavement: 13.76, soil: 5.81, building: 5.70, plant: 4.71, elevation: 2.45, person: 2.25, object: 1.61, waterbody: 1.14, vehicle: 0.66, indoor: 0.01
up:
sky: 48.32, pavement: 9.98, building: 7.22, plant: 5.89, soil: 5.00, elevation: 2.82, object: 1.36, person: 1.22, waterbody: 1.18, vehicle: 0.34, indoor: 0.02
right:
sky: 45.22, pavement: 13.06, building: 6.49, soil: 5.79, plant: 4.72, elevation: 2.46, person: 2.45, object: 1.54, waterbody: 1.12, vehicle: 0.58, indoor: 0.02
down:
sky: 40.36, pavement: 17.65, soil: 7.32, building: 5.42, plant: 4.09, elevation: 2.08, object: 1.95, person: 1.93, waterbody: 1.19, vehicle: 0.85, indoor: 0.02

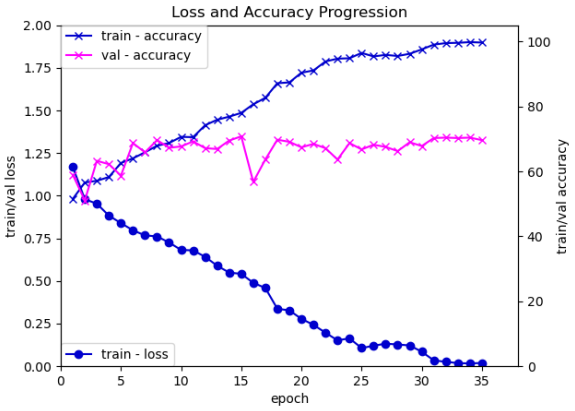
C Accuracy-Loss Plots of CNNs During Hyperparameter Tuning

This appendix provides the full collection of plots showing the learning process of various CNNs trained for domain prediction.

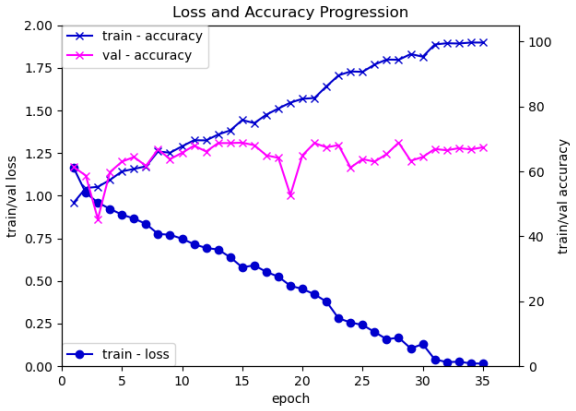
C.1 Plots on Instances



(a) DRN-C26 in configuration-I.

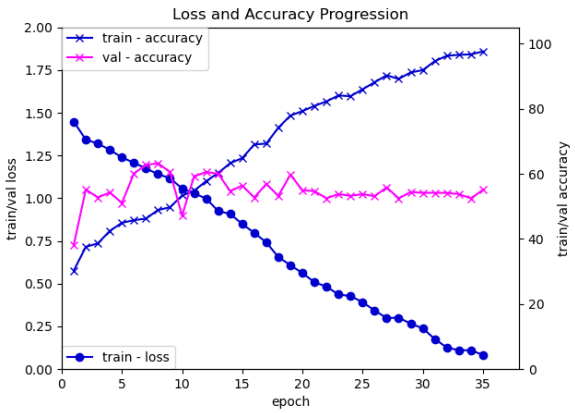


(b) DRN-C26 in configuration-II.



(c) DRN-D38 in configuration-I.

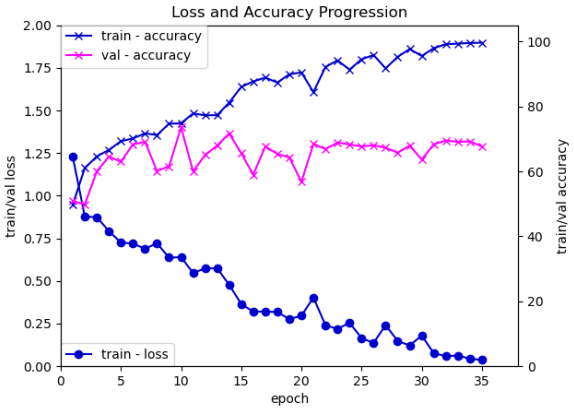
(d) MobileNetV2 in configuration-I.



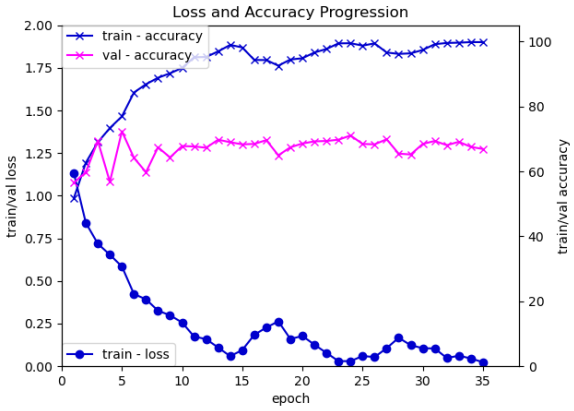
(e) MobileNetV2 in configuration-II.

Figure 15: Plots of various models trained on instances of *SemanticAircraft*.

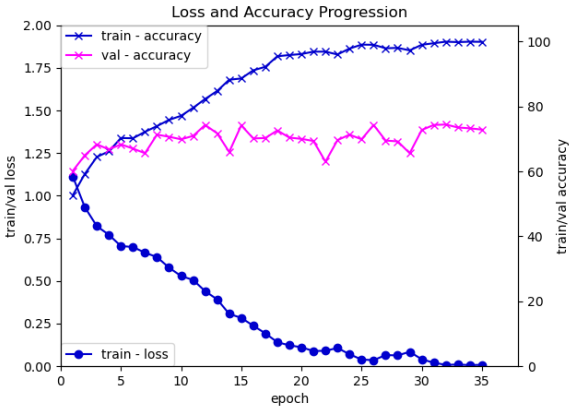
(a) ResNet18 in configuration-I.



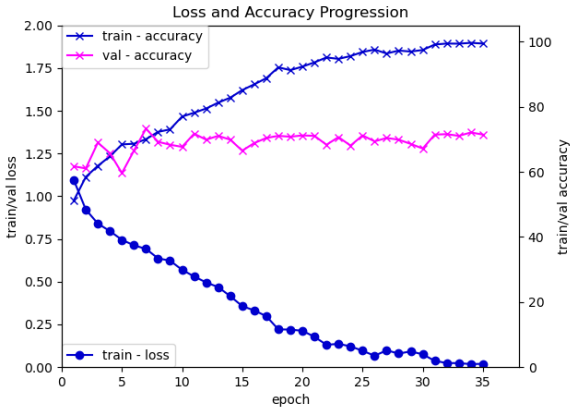
(b) ResNet18 in configuration-II.



(c) ResNet18 in configuration-III.



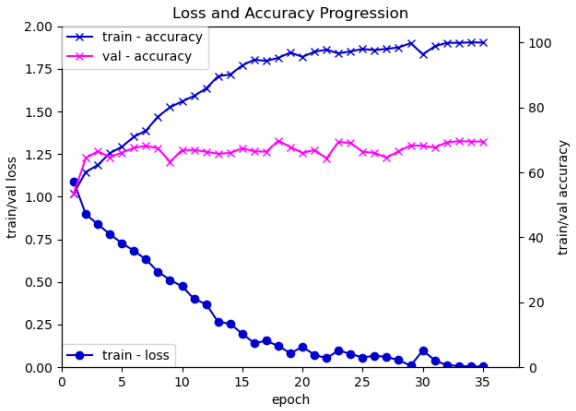
(d) ResNet18 in configuration-IV.



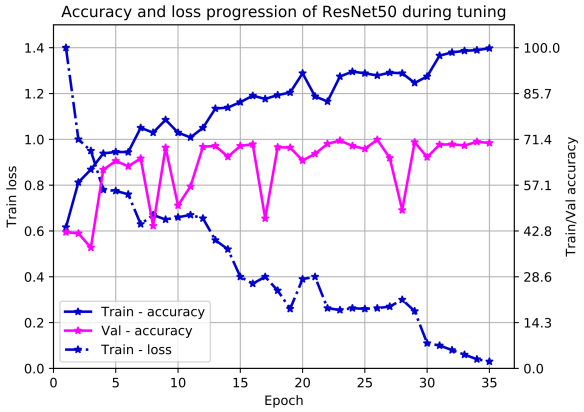
(e) ResNet18 in configuration-V.

(f) ResNet18 in configuration-VI.

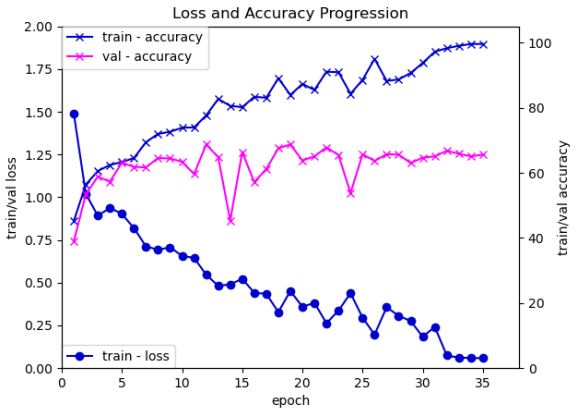
Figure 16: Plots of the six different ResNet18 configurations.



(a) ResNet34 in configuration-I.



(b) ResNet50 in configuration-I.

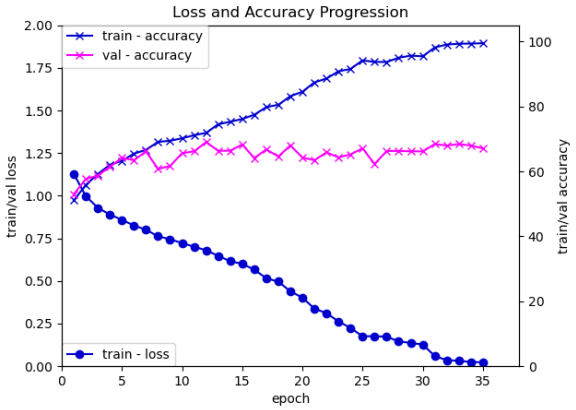


(c) ResNet50 in configuration-II.

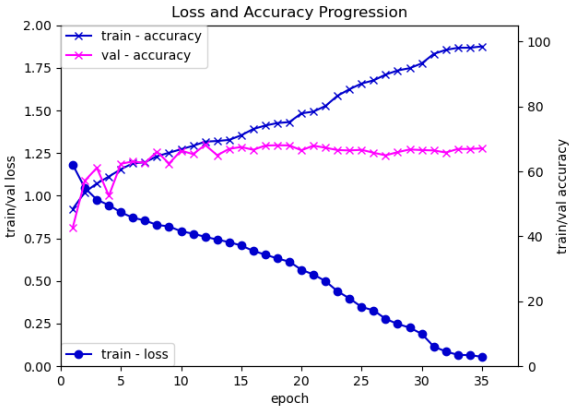
(d) ResNet50 in configuration-III.

Figure 17: Four more plots of models trained on *SemanticAircraft* instances.

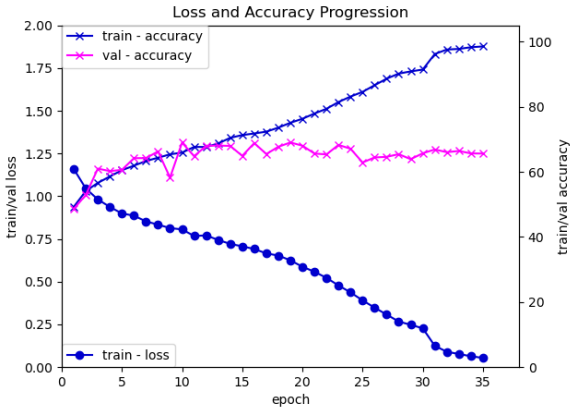
C.2 Plots on Quadrants



(a) DRN-C26 in configuration-I.

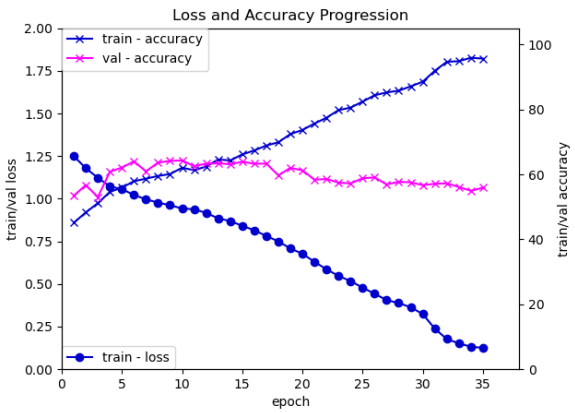


(b) DRN-C26 in configuration-II.



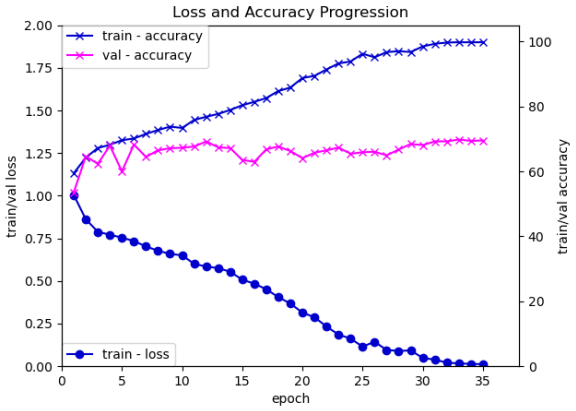
(c) DRN-C26 in configuration-III.

(d) MobileNetV2 in configuration-I.

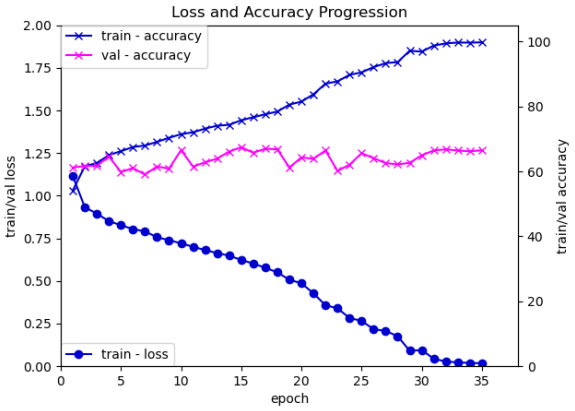


(e) MobileNetV2 in configuration-II.

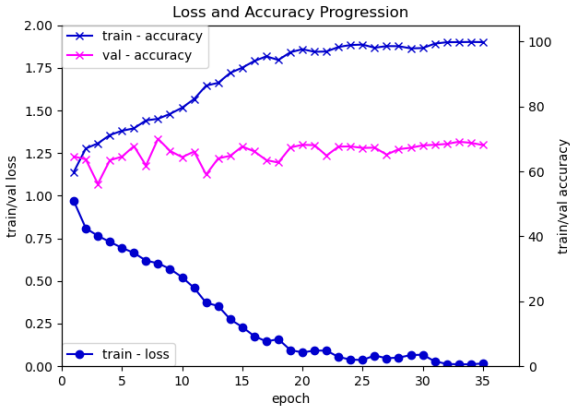
Figure 18: Plots of different models trained on quadrants of *SemanticAircraft*.



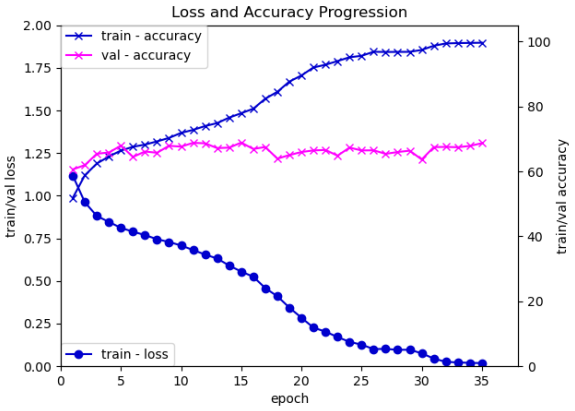
(a) ResNet18 in configuration-I.



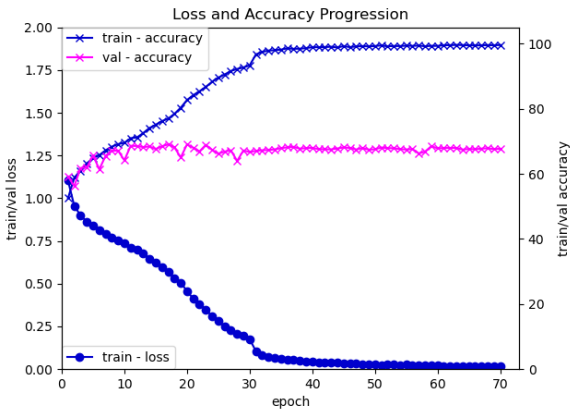
(b) ResNet18 in configuration-II.



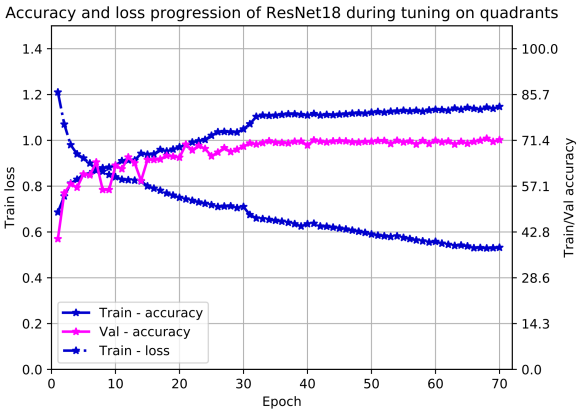
(c) ResNet18 in configuration-III.



(d) ResNet18 in configuration-IV.

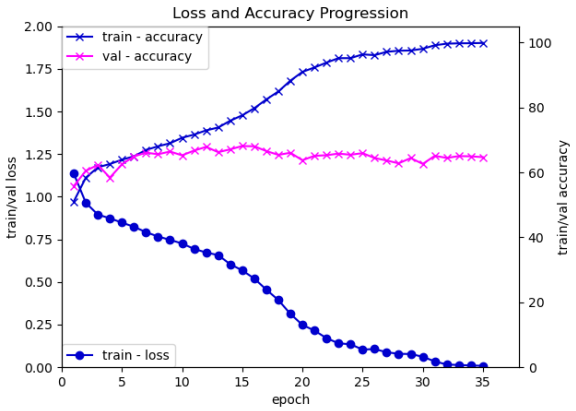


(e) ResNet18 in configuration-V.

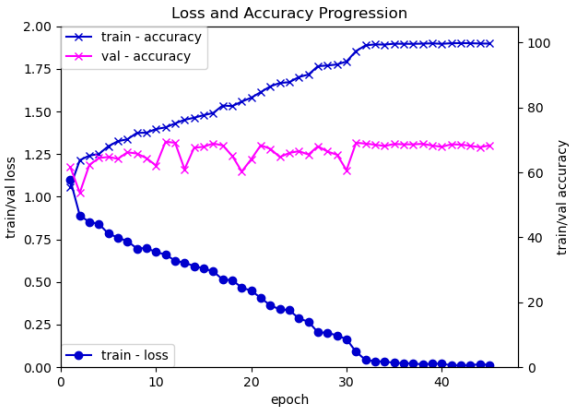


(f) ResNet18 in configuration-VI.

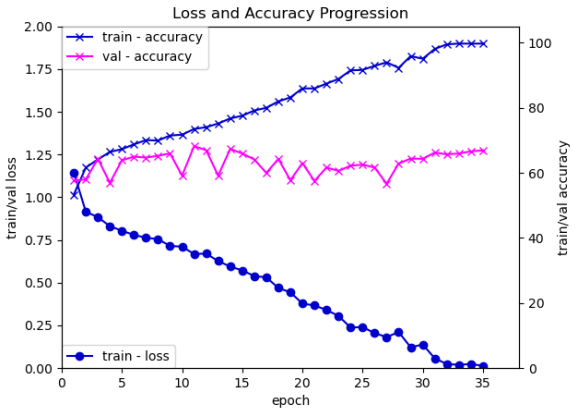
Figure 19: Plots of all ResNet18 models.



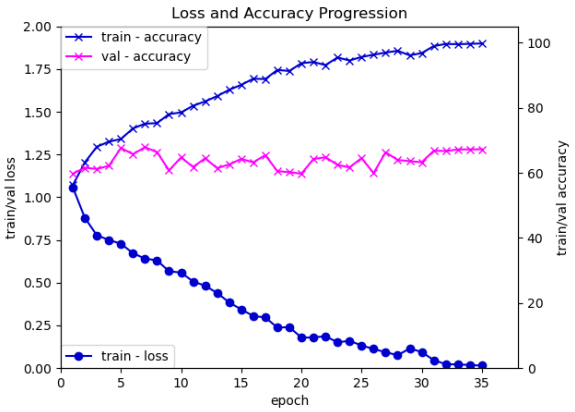
(a) ResNet34 in configuration-I.



(b) ResNet50 in configuration-I.



(c) ResNet50 in configuration-II.



(d) ResNet50 in configuration-III.

Figure 20: Plots of ResNet34 and ResNet50 models trained on quadrants.