University of Münster

Department of Information Systems

DINO: Learning Robust Visual Features without Supervision

Seminar Thesis

in the context of the seminar

Advanced Topics in Deep Learning

submitted by

Tobias Zimmermann

Chair of Data Science:
Machine Learning and Data Engineering

Principal Supervisor	Prof. Dr. Fabian Gieseke
Supervisor	Christian Lülf, M.Sc.
	Chair for Data Science:
	Machine Learning and
	Data Engineering
Matriculation Number	454407
Field of Study	Information Systems
Contact Details	t_zimm11@uni-muenster.de
Submission Date	29.02.2024

Contents

List of Figures

List of Abbreviations

BYOL	Bootstrap Your Own Latent
DINO	knowledge DIsitllation with NO labels
DPT	Dense Prediction Transformer
EMA	Exponential Moving Average
FlashAttention	Fast and Memory-Efficient Exact Attention with IO-Awareness
FSDP	Fully-Sharded Data Parallel
iBOT	Image BERT Pre-Training with Online Tokenizer
k-NN	k-Nearest Neighbor
MAE	Mean Absolute Error
MLP	Multi-layer Perceptron
ResNet	Residual Network
R^2	R^2 Regression Score
SGD	Stochastic Gradient Descent
SSL	Self-Supervised Learning
ViT	Vistion Transformer

1 Introduction

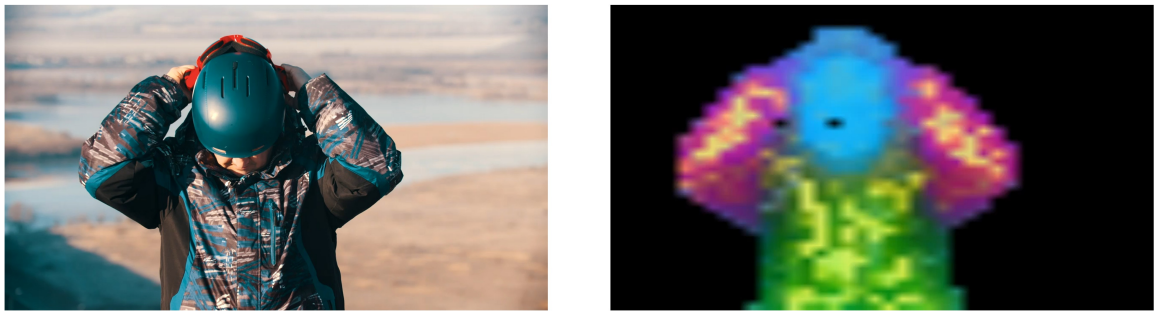


Figure 1 Exemplary Visualization of DINOs Image Understanding Capabilities

In the field of machine learning, Self-Supervised Learning (SSL) and knowledge distillation have become essential techniques. SSL, a form of supervised learning, removes the requirement for labeled data by using the data itself to create labels. On the other hand, knowledge distillation entails transferring knowledge from a larger, more intricate model to a smaller, more streamlined one.

The knowledge DIsitllation with NO labels (DINO) approach combines the concepts of SSL and knowledge distillation. This innovative technique leverages the strengths of both methodologies to enhance model training efficiency and effectiveness, setting a new standard in the field. An introduction to both concepts, and the DINO approach, is given in Chapter 2.

Since its initial release, the DINO approach has undergone significant improvements, transitioning to DINOv2. These advancements include employing larger datasets, refining pre-training techniques, and improving overall implementation for increased efficiency and performance. These improvements and eventually the consequences of them are described in Chapter 3. Figure 1 shows the ability of DINOv2 models to capture the information present in an image and to understand the images more deeply, rather than just learning characteristics for one downstream task.

One promising application of DINOv2 is in the field of satellite imagery. By applying DINOv2, we can extract more accurate and meaningful information from satellite data, useful for enhanced performance in various applications such as canopy height mapping, which is essential for monitoring world forest changes. Thus, the application of DINOv2 in the field of canopy height mapping is presented in Chapter 4.

2 The DINO Approach

The DINO approach is based on two rather advanced deep learning techniques: SSL and knowledge distillation. Due to this, the general concepts of SSL and knowledge distillation are first introduced. Subsequently, the combination of these two concepts called Self-Distillation as described in Caron et al. [Car+21] will be explained. Finally, Vistion Transformer (ViT) and its role in the DINO approach will be discussed.

2.1 Self-Supervised Learning

SSL is a variant of supervised learning that eliminates the need for costly and time-consuming data labeling by generating labels from the data itself. Instead of training the model to predict a specific ground truth, the objective is to model the underlying observations themselves. There are various types of SSL techniques.

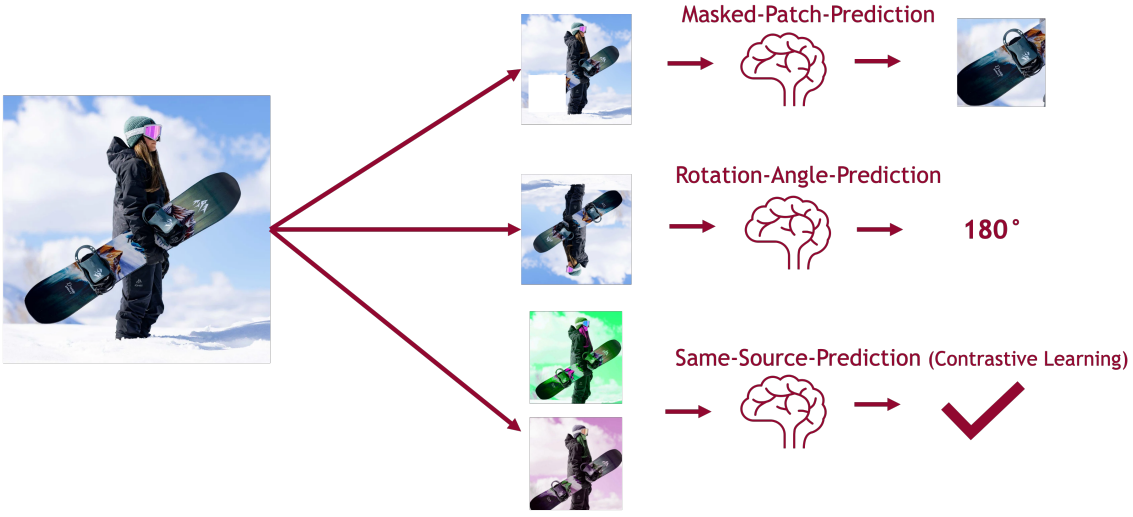


Figure 2 Different Approaches of Self-Supervised Learning

Figure 2 illustrates three exemplary forms of SSL. For instance, in Masked-Patch-Prediction, random patches from the input image are masked, and the model learns to predict these masked patches. Then, a classical learning algorithm is performed on the loss between the actual and predicted mask. This form of SSL is also known as generative learning. Another example is Rotation-Angle-Prediction, where the model is presented with a randomly rotated image and is tasked with predicting the rotation angle. The loss for training the model is then calculated based on the difference between the predicted and actual rotation angles. Additionally, Same-Source-Prediction, a form of contrastive learning, requires the model to determine whether two randomly augmented images originate from the same source image. In other words, this is a classical binary classification problem for which there are

multiple training algorithms. Compared to supervised learning, SSL aims to create a more human-like learning experience for the model and develop a better perception of the underlying observations that can be applied to multiple downstream tasks. Due to this, SSL is often referred to as pre-training, because there is another training or fine-tuning for the downstream task afterward [LeC20; LM21].

2.2 Knowledge Distillation

Larger models have greater capabilities to model the underlying data, yielding better results than smaller models. However, they come at the cost of being more computationally complex, leading to a larger demand for computation resources and time during training and also later on during inference. While at training time this larger demand of time and resources can sometime be fulfilled, many use cases do not allow for that much time and resource consumption during inference. Because of this, it would be helpful to get a smaller version from a larger model with similar results. This process of distilling the knowledge of a larger model into a smaller one is called knowledge distillation [HVD15].

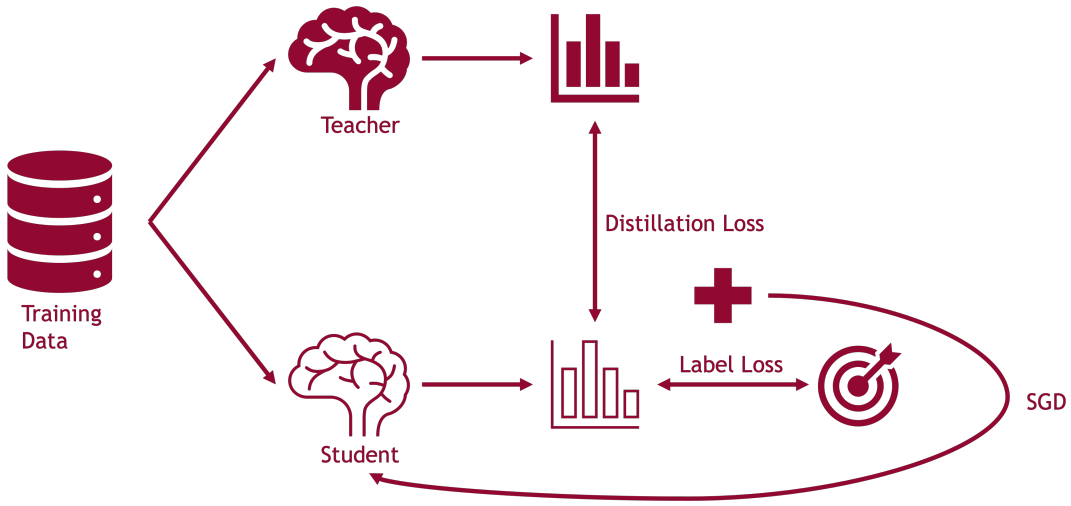


Figure 3 Illustration of Knowledge Distillation

The process of knowledge distillation is illustrated in Figure 3. During knowledge distillation, the larger model is called the teacher, while the smaller model is called the student, because the larger model is teaching the smaller one to achieve better results. The training data is passed through both models, and the loss between both predictions builds the distillation loss. It should be noted that this loss is calculated on the raw output of both models. So, for a classification problem, for example, the distillation loss is not calculated on the final class prediction, but on the class probabilities over all possible classes. In this way, the distillation loss not only punishes misclassifications and rewards correct ones, but also guides the student to

model class probabilities just like the teacher. Furthermore, this distillation loss is then combined with the label loss on the basis of the difference between the prediction of the students and the actual ground truth. This combination of both losses is then used with Stochastic Gradient Descent (SGD), to train the student. However, when knowledge distillation is applied to achieve a smaller model and due to this less consumptive model, this always comes at the cost of some quality loss. This means that the smaller model can become quite as good as the larger one, but will not match its quality completely [HVD15].

2.3 Self-Distillation

The DINO approach introduced by Caron et al. [Car+21] synthesizes the previous two concepts into a process called Self-Distillation. This process, inspired by the Bootstrap Your Own Latent (BYOL) approach from Grill et al. [Gri+20], merges the techniques of SSL with those of knowledge distillation. In contrast to the traditional application of knowledge distillation, the DINO framework leverages identical model architectures for both the teacher and student models. This architecture consists of a ViT from Dosovitskiy et al. [Dos+21] or Residual Network (ResNet) introduced by He et al. [He+15] that serves as the backbone, complemented by a Multi-layer Perceptron (MLP) that functions as a projection head for training only and is then discarded [Car+21].

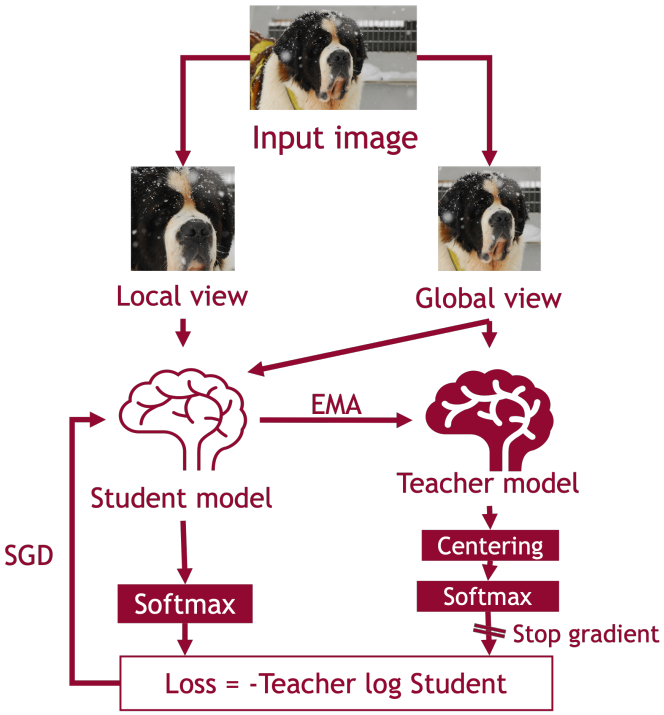


Figure 4 Self-Distillation Process Illustrated

The process of self-distillation within this framework is illustrated in Figure 4. SSL is implemented through a multi-crop strategy, where multiple subsets of the input image, called views, are generated. These views are categorized on the basis of the percentage of the original image's area they cover. Local views cover less than 50 percent of the original image area, whereas global views exceed this threshold. The student model is fed with both view types, while the teacher model processes only global views. Thus, the student learns to find a representation of the local views corresponding to the global ones. Both models generate embeddings from their respective inputs, after applying centering to the teacher's output, a Softmax activation function, and a Cross-Entropy loss function are employed to align the embeddings of the student with those produced by the teacher. While the student learns based on SGD, the teacher's parameters are updated using an Exponential Moving Average (EMA) derived from the student's weights. As shown in Caron et al. [Car+21], the teacher model consistently outperforms the student during the training process, thus providing superior quality feature guidance [Car+21].

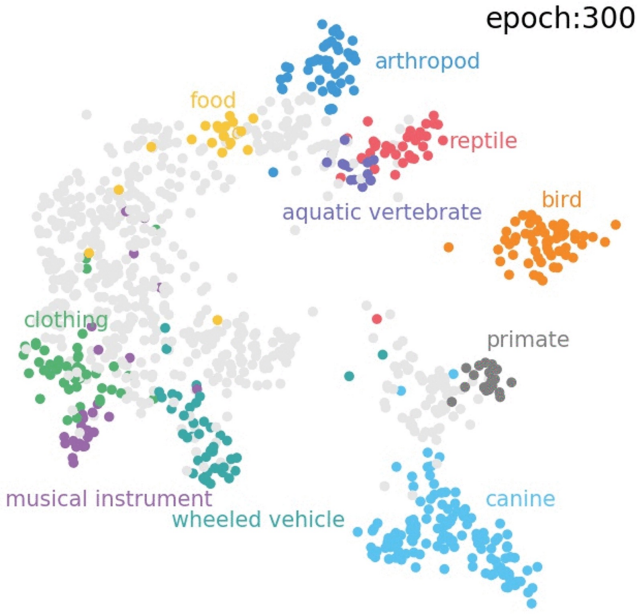
Algorithm 1 DINO approach in pseudo Python/PyTorch code

```
teacher, student, C, tp_t, tp_s, l, m = init_dino()
for image in loader:
    g_v = sample_global_views(image)
    l_v = sample_local_views(image)
    t_o = teacher(g_v)
    s_o = student(g_v + l_v)
    t_o = softmax((t - C) / tp_t)
    t_o = t_o.detach().chunk(len(g_v)
    s_o = (s / tp_s).chunk(len(g_v + l_v))
    total_loss = 0
    n_loss_terms = 0
    for i, t in enumerate(t_o):
        for j, s in enumerate(s_o):
            if i == j:
                continue
            loss = sum(-t * log_softmax(s))
            total_loss += loss.mean()
            n_loss_terms += 1
    total_loss /= n_loss_terms
    total_loss.backward()
    update(student)
    teacher.params = l * teacher.params + (1-l) * student.params
    C = m*C + (1-m) * t_o.mean()
```

Pseudocode for the DINO approach is given in Algorithm 1 based on the pseudocode from Caron et al. [Car+21] and the actual implementation[1] of it.

[1] https://github.com/facebookresearch/dino

A common challenge in SSL is model collapse, where the model output becomes excessively uniform or dominated by a single dimension. To mitigate this issue, the DINO framework incorporates centering, which prevents one dimension from dominating by adding a bias term to the teacher's output that is updated with an EMA based on the output from the teacher, as implemented in lines seven and 23 of Algorithm 1. Additionally, sharpening is incorporated in lines seven and nine of Algorithm 1, which is the division by a very small number, also called temperature, to amplify minor differences between activations, thus preserving the distinctiveness of the output and preventing it from becoming excessively uniform [Car+21].



Method	Arch.	Param.	im/s	Linear	k-NN
Supervised	RN50	23	1237	79.3	79.3
SCLR [12]	RN50	23	1237	69.1	60.7
MoCov2 [15]	RN50	23	1237	71.1	61.9
InfoMin [67]	RN50	23	1237	73.0	65.3
BarlowT [81]	RN50	23	1237	73.2	66.0
OBoW [27]	RN50	23	1237	73.8	61.9
BYOL [30]	RN50	23	1237	74.4	64.8
DCv2 [10]	RN50	23	1237	75.2	67.1
SwAV [10]	RN50	23	1237	**75.3**	65.7
DINO	RN50	23	1237	**75.3**	**67.5**
Supervised	ViT-S	21	1007	79.8	79.8
BYOL* [30]	ViT-S	21	1007	71.4	66.6
MoCov2* [15]	ViT-S	21	1007	72.7	64.4
SwAV* [10]	ViT-S	21	1007	73.5	66.3
DINO	ViT-S	21	1007	**77.0**	**74.5**
Comparison across architectures					
SCLR [12]	RN50w4	375	117	76.8	69.3
SwAV [10]	RN50w2	93	384	77.3	67.3
BYOL [30]	RN50w2	93	384	77.4	–
DINO	ViT-B/16	85	312	78.2	76.1
SwAV [10]	RN50w5	586	76	78.5	67.1
BYOL [30]	RN50w4	375	117	78.6	–
BYOL [30]	RN200w2	250	123	79.6	73.9
DINO	ViT-S/8	21	180	79.7	**78.3**
SCLRv2 [13]	RN152w3+SK	794	46	79.8	73.1
DINO	ViT-B/8	85	63	**80.1**	77.4

Figure 5 DINO Results from Bojanowski et al. [Boj+21] and Caron et al. [Car+21]

Figure 5 shows the results of applying the DINO approach. As shown by Bojanowski et al. [Boj+21] after just 300 epochs, the model learns meaningful representations for ImageNet classes, showing that without supervised information about those classes, the model learns to understand the underlying data and distinguish the content in it. This allows the usage of these embeddings together with a linear or k-Nearest Neighbor (k-NN) classifier to achieve a higher top-1 accuracy than supervised and previous SSL methods, as shown in the table in Figure 5 [Car+21; Boj+21].

2.4 The Role of ViT in DINO

As stated above, the table in Figure 5 shows the performance of the DINO approach for ImageNet classification together with a linear and k-NN classifier. However, this table also shows that the DINO approach achieves better results with the ViT backbone than with the ResNet backbone. This lies in the nature of transformer architec-

tures, as they have already shown superior performance following similar approaches in other fields [Car+21].

The ViT architecture, shown in Figure 6, is based on sequence processing. Therefore, it processes the image as a sequence of contiguous non-overlapping image patches of $N \times N$ resolution. These patches are passed through a linear layer to form a set of embeddings. Also, an extra learnable token is added, called class or 'cls' token. The model learns to aggregate the information from the entire sequence into that token, and then a projection head on top of it might be used for the final output. The transformer encoder is a sequence of multi-head-attention and feed-forward layers, paralleled with skip connections. The multi-head-attention layers update a token representation by looking at the representations of other tokens with the so-called attention mechanism. Therefore, the transformer encoder can process each patch in the context of all other patches [Dos+21].

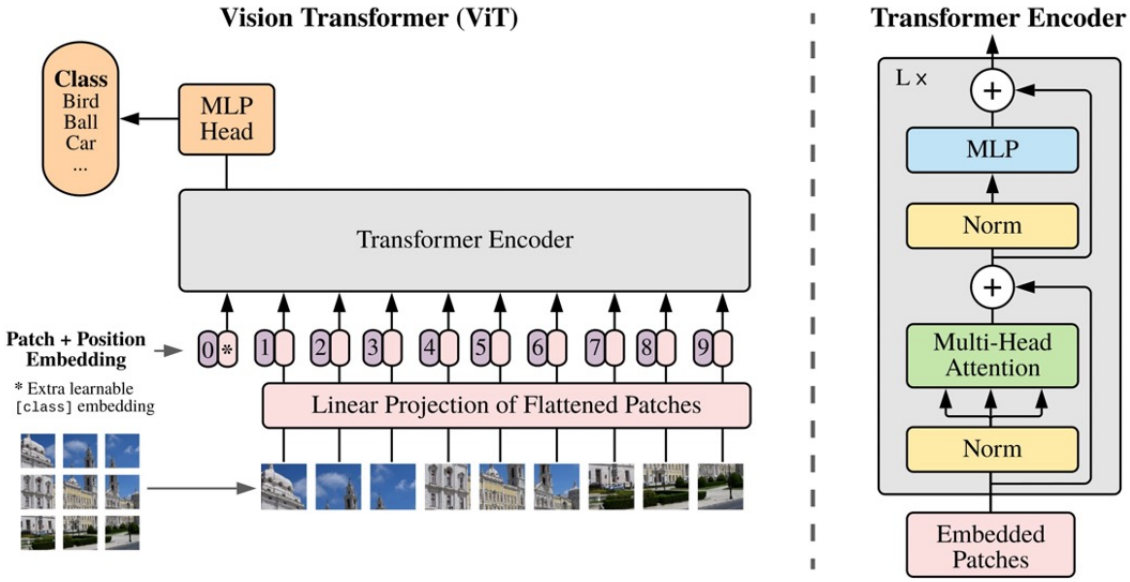


Figure 6 ViT Architecture from Dosovitskiy et al. [Dos+21]

Transformers have already shown superior results in other fields like natural language processing, for example. But this performance comes at the cost of higher computational demands and the need for longer and more diversified training with more training data than previous model architectures have needed before. Inspired by approaches from natural language processing, Caron et al. [Car+21] came up with a BERT-like vision encoder and overcame the obstacles of training transformers by using SSL in their DINO approach. They leverage transformers' capability to capture a broader range of dependencies in the data. Finally, by visualizing the local attention maps they show that with the DINO approach, ViT models learn more meaningful aspects of the images [Boj+21; Car+21].

3 Taking DINO to the Next Step

After the first version of DINO published by Caron et al. [Car+21], around two years later Oquab et al. [Oqu+23] came up with the next iteration of the approach. Taking DINO to the next step, they called their approach DINOv2. Their improvements can be divided into three main parts. Namely, the larger dataset they used for pretraining and their automatic curation approach, their extension of discriminative self-supervised pretraining, and their implementation improvements that allow efficient usage of larger transformer architectures. These three main parts will be explained in the following sections. Finally, in the last section, the need for registers in larger ViT architectures will be discussed.

3.1 Automatic Dataset Curation

The first major improvement was to use a much larger dataset. For this, Oquab et al. [Oqu+23] came up with an approach to automatic data curation, which led to their LVD-142M dataset that contains 142 million diverse images. Just like the original DINO approach, this dataset curation approach was inspired by a similar approach from the field of natural language processing [Oqu+23].

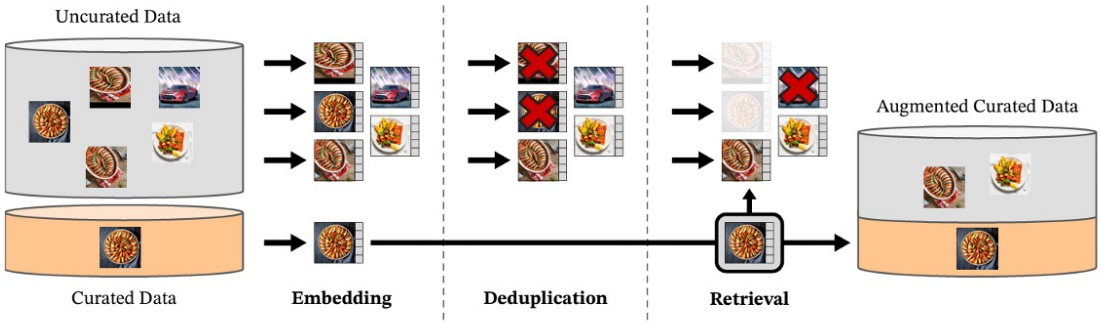


Figure 7 DINOv2 Data Collection Approach from Oquab et al. [Oqu+23]

The approach is visualized in Figure 7. First, 1.2 billion images from various sources were collected and stored as a pool of uncurated data. The curated data pool consisted of images from various curated datasets. Based on cosine-similarity and specific embeddings for deduplication by Pizzi et al. [Piz+22], copies and near-duplicates were then removed from the uncurated pool in a deduplication phase. Furthermore, images from the uncurated pool that are copies of or too similar to images from the evaluation set were completely removed from the uncurated pool using a more restrictive cosine-similarity threshold [Piz+22; Oqu+23].

After that, in a retrieval phase, two approaches were used to augment the datasets from the curated pool with images from the cleaned uncurated one. Both were performed based on the cosine-similarity between embeddings produced by a self-supervised pre-trained ViT model. The first retrieval approach is called sample-based retrieval, which was used for larger curated datasets with more than one million images. Here, for each image in a curated dataset, a fixed number of N images close to it were sampled from the cleaned uncurated pool. For smaller curated datasets with less than one million images, a cluster-based retrieval approach was used. For each cluster, in which three or more images from the curated dataset fell, a fixed number of M images were retrieved from it. The total number of images retrieved for each curated dataset was capped at one million images for this approach in order to balance them. Both N and M are hyperparameters that were tuned based on visual assessment. The application of both approaches led to the augmented curated LVD-142M dataset [Oqu+23].

3.2 Discriminative Self-Supervised Pre-Training

The next major improvement that led to DINOv2 was the amendments made to the original training objective. In addition to the image-level objective, described in Chapter 2, the so-called patch-level objective was added, inspired by the Image BERT Pre-Training with Online Tokenizer (iBOT) approach of Zhou et al. [Zho+22]. Here, the input patches given to the student were randomly masked, whereas the inputs given to the teacher were not modified. Then a cross-entropy loss was added between the embeddings of both networks for these patches. Thus, the teacher not only guided the student with embeddings based on inputs with larger views as before, but also guided the student with more information even within the smaller area of local views. After the introduction of the second objective, an issue with overfitting at the image-level and simultaneously underfitting at the patch-level was observed. This could be resolved by untying the weights that were related to both objectives [Oqu+23].

Furthermore, increasing the resolution of the input image led to better performance in downstream tasks such as segmentation or detection. But because this resolution increase was more time and memory demanding, the resolution was just increased during a short period at the end of pre-training. This led to better performance in downstream tasks, but was not resource-consuming [Oqu+23].

3.3 Efficient Implementation for Larger Transformers

Last but not least, another major improvement was an efficient implementation, which enhanced the usage of larger transformer architectures. For this, a custom version

of Fast and Memory-Efficient Exact Attention with IO-Awareness (FlashAttention) by Dao et al. [Dao+22] was implemented to improve the computational performance of the self-attention layers. Because of the hardware specific improvements, the ViT architecture used was slightly modified in order to perform better with the corresponding warp sizes of the GPUs used for training. This led to a ViT architecture with 1.1 billion parameters for the largest version. The usage of nested tensors in the self-attention layers allowed the local and global views for the image-level objective to be processed using the same forward and backward passes, even though they differ in the number of patch tokens needed to process them. Also for stochastic depth, which was originally developed by Huang et al. [Hua+16], an optimized version was implemented. This optimized implementation does not compute the residuals that are affected by dropout anyway[Oqu+23].

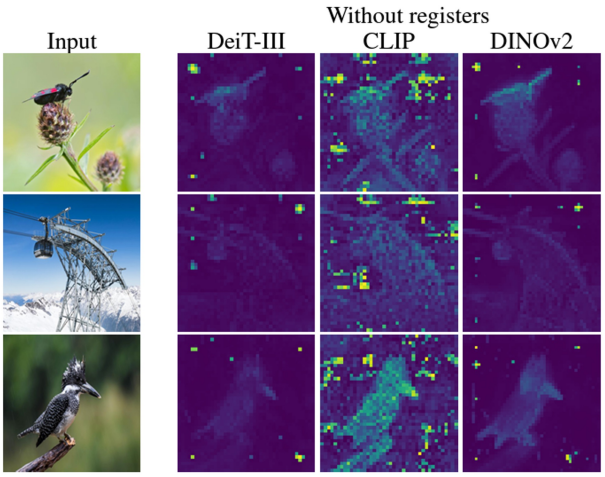
In addition to these implementation improvements, further improvements were made to the training process itself. Memory usage per GPU was reduced by splitting model replicas, needed for the AdamW optimizer, across GPUs using a Fully-Sharded Data Parallel (FSDP) approach implemented in PyTorch. Thus, the model size used was not capped by the memory capacity of the smallest GPU used for training, but by the sum of memory capacity from all GPUs used. Furthermore, by using a less precise datatype, the communication time between GPUs required by the FSDP approach could be reduced [Oqu+23].

Finally, because all these improvements were optimized for larger models, instead of training the smaller variants from scratch as before, they were distilled from the largest variant using a knowledge distillation approach. Here, as adaptations to the process described in Chapter 2, the largest variant was used as a frozen teacher, and a reserved EMA variant of the student was used as the final smaller variant. Also, masking and stochastic depth were removed and the iBOT loss from Zhou et al. [Zho+22] was applied to global views [Oqu+23].

3.4 Larger ViTs develop a Need for Registers

As stated above, one of the major improvements from Oquab et al. [Oqu+23] for DINOv2, but also from other researchers in the same field, was the usage of larger ViT architectures. When Darcet et al. [Dar+23] discovered that there were outlier tokens without any useful information in the visualized attention maps of DINOv2 in contrast to those of DINOv1, they also found this phenomenon in other larger ViTs. These outlier tokens are shown in the illustration of Figure 8 from Darcet et al. [Dar+23]. They could lead to possible false model output when they start to dominate and make the model less interpretable due to the inaccurate attention

maps. Darcet et al. [Dar+23] found out that these outlier tokens appeared when training larger ViTs on sufficient data for long enough time.



	IN1k	P205	Airc.	CF10	CF100	CUB	Cal101
[CLS]	**86.0**	**66.4**	**87.3**	**99.4**	**94.5**	**91.3**	96.9
normal	65.8	53.1	17.1	97.1	81.3	18.6	73.2
outlier	69.0	55.1	79.1	99.3	93.7	84.9	**97.6**

	Cars	DTD	Flow.	Food	Pets	SUN	VOC
[CLS]	**91.5**	**85.2**	**99.7**	**94.7**	**96.9**	**78.6**	89.1
normal	10.8	63.1	59.5	74.2	47.8	37.7	70.8
outlier	85.2	84.9	99.6	93.5	94.1	78.5	**89.7**

Figure 8 Outlier Tokens in larger ViTs adapted from Darcet et al. [Dar+23]

Furthermore, these outlier tokens appeared in the attention maps at positions where the patches contained redundant information. During their research, they trained a classifier on actual class tokens, normal ones, and these outliers. The results for 14 classification benchmark datasets are shown in the table in Figure 8. They showed that these outlier tokens actually contain useful information, since the classifier trained on them achieves higher accuracies for all classification benchmarks than the one trained on normal tokens. Additionally, for two classification benchmarks, the classifier trained on the outlier tokens scores higher accuracies than the one trained on the actual class token [Dar+23].

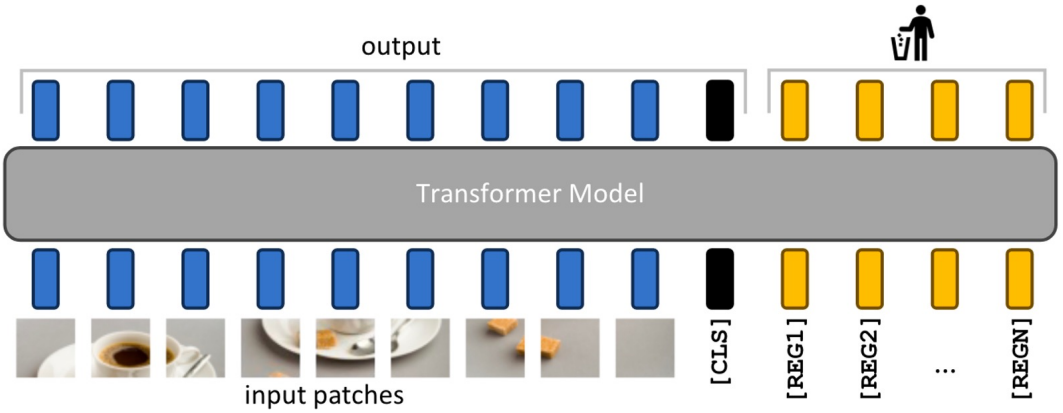


Figure 9 ViT Architecture with Registers from Darcet et al. [Dar+23]

Thus, they stated the hypothesis that the model starts to use these tokens that carry redundant information to store global information in them. Therefore, they intended to give the model extra space to store this global information instead of

using redundant tokens. So, they added register tokens to the ViT architecture. The modified ViT architecture is shown in Figure 9. By visualizing the attention maps after applying their modified architecture, Darcet et al. [Dar+23] showed that the model produced interpretable attention maps again, just as the smaller models of DINOv1 before. Also, by visualizing the specific attention maps of the register tokens, they could show that the model learns to use them to store global information in them, without being explicitly instructed to do so. However, they stated that the register tokens have no aligned behavior and thus show some sort of irregularity [Dar+23].

4 Applying DINOv2 on Satellite Imagery

After an introduction to the DINO approach in Chapter 2 and its enhancements in Chapter 3, this chapter will present the application of the most recent DINO model in satellite imagery, inspired by the research of Tolan et al. [Tol+24]. Due to the limitations in time and resources, the focus is only on phase two of their DINOv2 application. For this, the used data set and the necessary data preprocessing steps will be first introduced. Subsequently, the process of fine-tuning a Dense Prediction Transformer (DPT) decoder model on top of the DINOv2 backbone will be shown, reproducing this step from Tolan et al. [Tol+24] as closely as possible.

4.1 Satellite Imagery and Lidar Dataset Preparation

The satellite imagery dataset was received divided into two pars, a train area[2] and a test area[3]. Both areas are in the Münsterland in Germany. The original dataset contained one satellite image for each area and multiple LiDAR data files, all of them stored as tag image files.

To train on this dataset, the patches in the two satellite images that match the area covered by the LiDAR images are first extracted. The Python code for this step can be found in the GitHub repository[4] for this work, in the "extract_patches" notebook. Because the satellite and LiDAR images differ in their resolution, the resolution of the satellite images is reduced to match the LiDAR image resolution. These pairs of satellite image patches and the corresponding LiDAR patches are then stored as zipped NumPy files. This is done for the train and test area, respectively. Oriented at Tolan et al. [Tol+24], thumbnails with a size of 256×255 pixels are created from the patches of the previous step. For this step, the Python code can be found in the "create_thumbnails" notebook in the GitHub repository[4]. As before, the satellite image and LiDAR thumbnail pairs are stored as zipped NumPy files. This leads to the final dataset for the application, with $4,480$ elements for the training set and $3,840$ elements for the test set.

4.2 Fine-tuning a DPT-Decoder on Top of DINOv2

After the preparation of the dataset, it is used to train a DPT model on top of a frozen DINOv2 model as the backbone, which is used as an image encoder. This

[2] http://bboxfinder.com/#51.747070,7.149510,51.799078,7.286562
[3] http://bboxfinder.com/#51.693032,7.149510,51.745101,7.286562
[4] https://github.com/tZimmermann98/dinov2-dpt-depth-estimation

process is oriented towards phase two of the approach from Tolan et al. [Tol+24], which is illustrated in Figure 10. Here, the DINOv2-small model is used as the frozen ViT encoder. The resulting SSL features are passed to the reassemble blocks of the DPT decoder, where they are downsampled. After that, the features are progressively upsampled in the fusion blocks, followed by a specific head for depth estimation, which progressively reduces the feature dimension to a single dimension and upsamples it to the original resolution of the input image. This results in a single-dimensional canopy height map prediction. The model is then trained with a supervised loss between the prediction and the ground-truth LiDAR canopy height map.

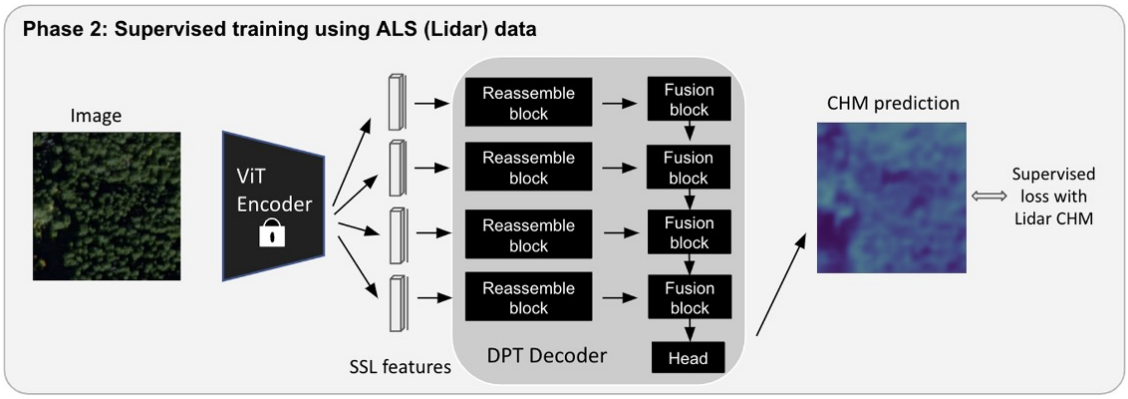


Figure 10 Training Process from Tolan et al. [Tol+24]

The training configurations are kept as close as possible to those described by Tolan et al. [Tol+24]. Thus, the training thumbnail images are augmented with random 90 degree rotations and color jittering with 0.1 as the parameter for brightness and contrast. To accommodate hardware capacities, a batch size of 32 is used. The Sigloss described by Tolan et al. [Tol+24] is implemented as a custom loss function. An AdamW optimizer is used together with a cosine annealing learning scheduler with a linear warm-up phase of $12,000$ iterations. After warming up, the learning rate is scheduled from 1×10^{-8} to 1×10^{-4}. The Python code for model training can be found in the "train" notebook in the GitHub repository[4]. During model training, an issue with exploding gradients is experienced. Due to this gradient clipping and a slight adaption to the loss function, also used by Oquab et al. [Oqu+23] is used. This stabilizes the training slightly, but the issue is not resolved completely. Thus, for the best trained model, the training is stopped after 102 epochs due to exploding gradients.

4.3 Canopy Height Mapping Results

After training the model as described above, the performance of the model was evaluated on the test set that contained satellite images from another area, in addition

to the training set. As a baseline, the model is compared with the two depth estimation models released by Oquab et al. [Oqu+23]. That is, the "KITTI" model that was fine-tuned on the KITTI depth estimation dataset using a "DINOv2-small" backbone. And also the "NYUd" model that was fine-tuned on the NYUd dataset using the same backbone. The three models are evaluated on the basis of the two metrics Mean Absolute Error (MAE) and R^2 Regression Score (R^2). For the MAE, a lower value is better, while for R^2 a higher value indicates better performance. These metrics are chosen for comparability reasons, as they are also used by Tolan et al. [Tol+24].

Model	**MAE**	**R^2**
KITTI	9.97	-0.87
NYUd	9.04	-0.87
Scratch	**4.15**	**0.53**

Table 1 Results of the Trained Model Compared to the Baseline Models

The results of the evaluation are shown in Table 1. The NYUd model only has a slightly better MAE than the KITTI model, but both baseline models share the same value at R^2. However, the proposed model, which is trained from scratch on top of the same frozen backbone, shows much better performance on both metrics. This indicates that the DINOv2 backbone already shows decent performance for downstream tasks in satellite imagery, without a specific pre-training in this imagery.

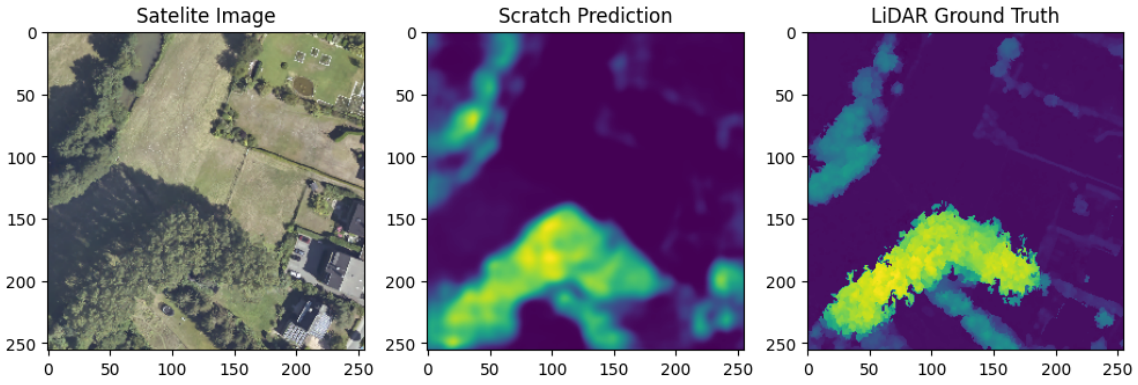


Figure 11 Visualization of the Proposed Models Canopy Height Map Prediction

The visualization of the prediction of the canopy height map of the proposed models for an exemplary satellite thumbnail can be seen in Figure 11. In Appendix A.1 further visualizations of exemplary thumbnails can be found, also compared to visualizations of the predictions from the two baseline models. The Python code for the evaluation and visualization of exemplary predictions can be found in the "evaluation" notebook in the GitHub repository[4].

5 Conclusion and Outlook

This seminar paper discussed the concepts of DINO and DINOv2, focusing on how SSL can be used in combination with knowledge distillation to learn meaningful feature extraction and understand image content.

For the practical application, the usage with satellite images to map forest heights was examined. It could be shown that DINOv2, the most recent version, works effectively for this specific task with satellite images even though it was pre-trained on images from mostly other domains.

However, due to resource constraints and the scope of this seminar paper, only fine-tuning could be tested on top of a pre-trained DINOv2 model. At the time of the work on the seminar paper, unfortunately, the models of Tolan et al. [Tol+24] were not yet released. Since these models have been published[5] at the time of writing, for future works, it would be interesting to compare their performance with the purposed model on the same dataset to analyze the effectiveness of pre-training a DINOv2 based encoder specifically for satellite images. Furthermore, the effect of continuing pre-training for the application of a pre-trained DINOv2 based encoder would be interesting to look into, just like previous work did for the BERT based encoder in the field of natural language processing.

In conclusion, using DINO and DINOv2 for satellite images is very promising, even though there are some hurdles, such as the issues with exploding gradients. For further implementations, these issues should be taken into account and would require further consideration.

[5] https://github.com/facebookresearch/HighResCanopyHeight

A Appendix

A.1 Further Canopy Height Mapping Results

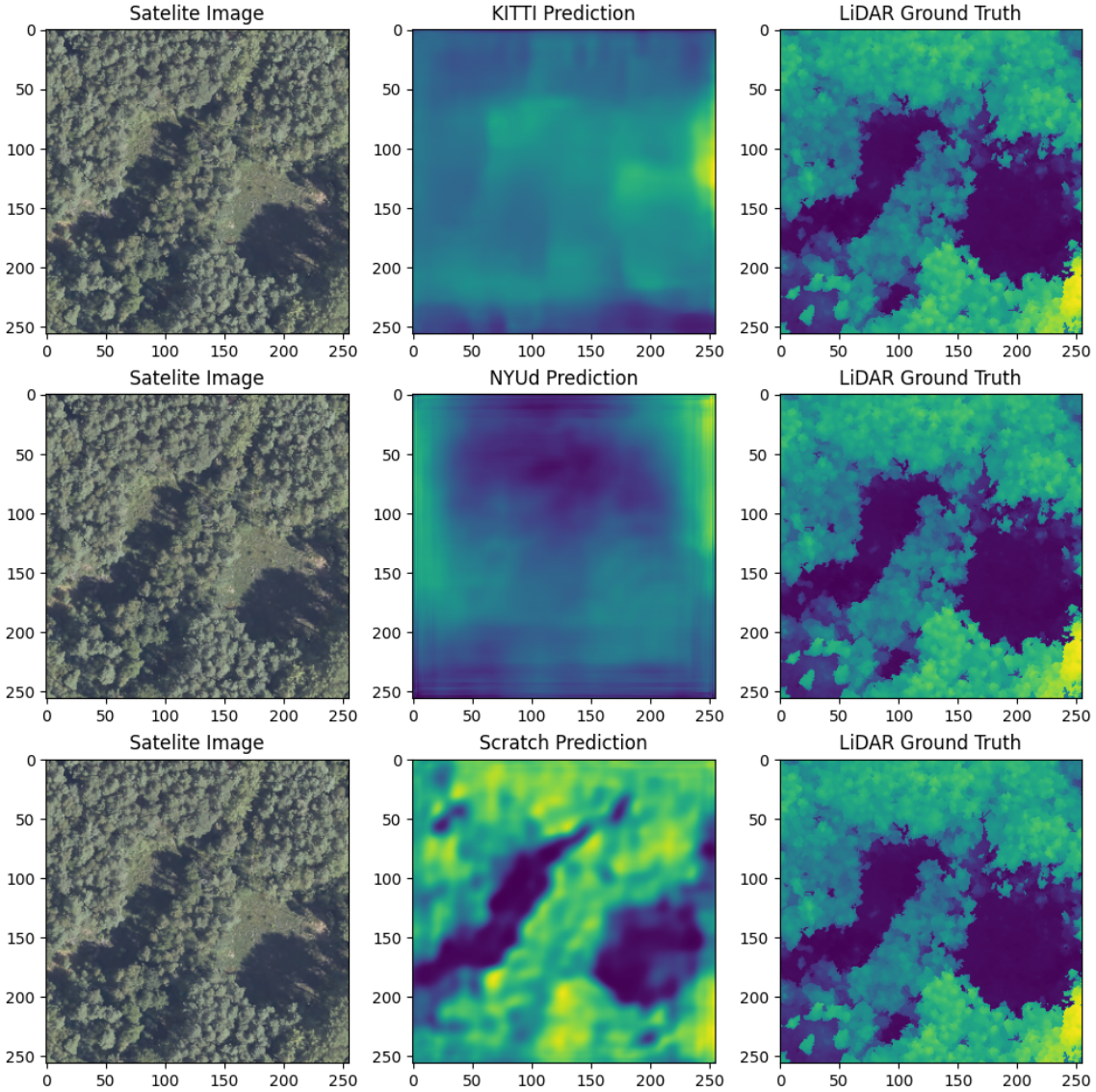


Figure 12 Visualization of Canopy Height Map Predictions

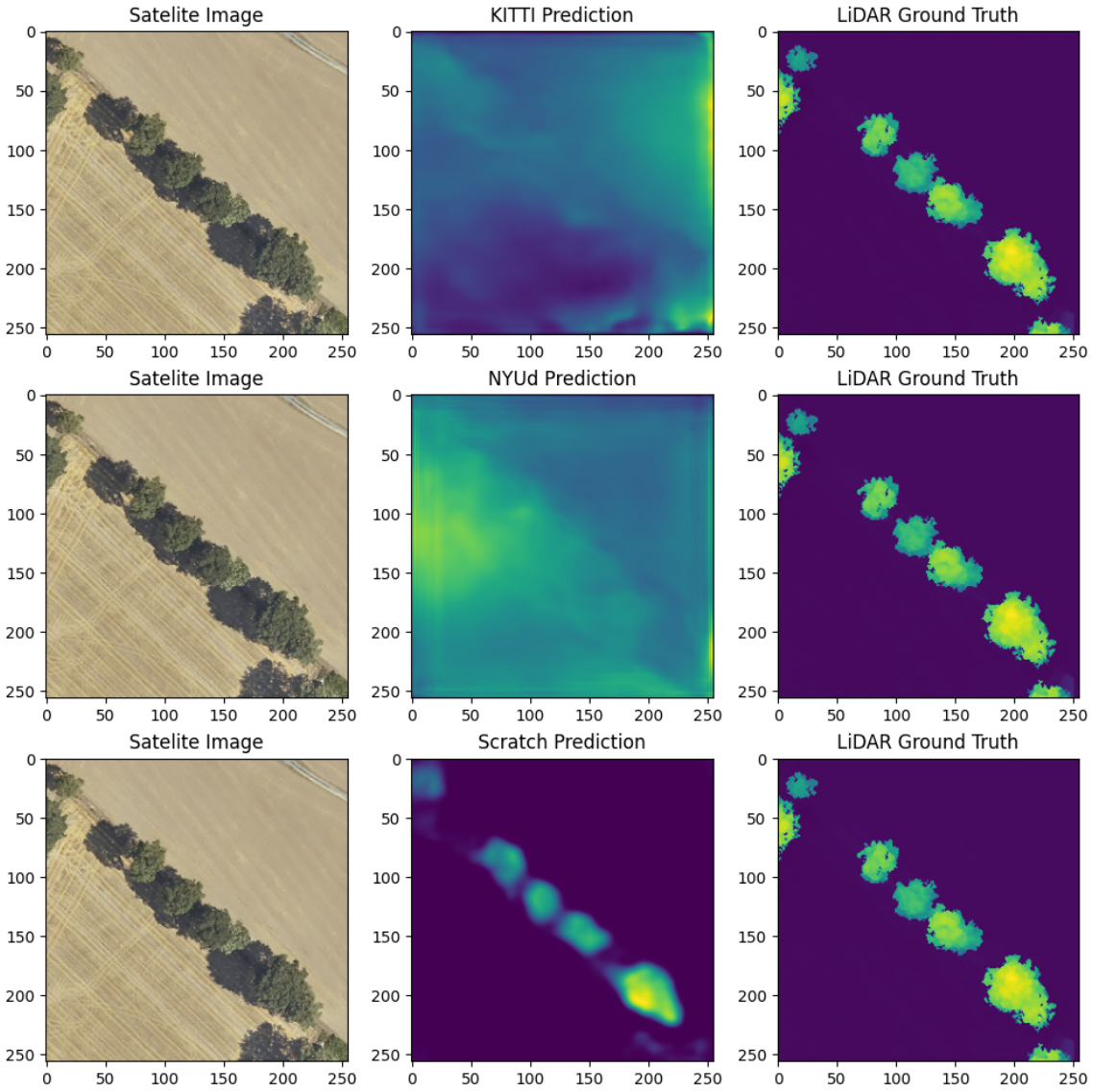


Figure 13 Visualization of Canopy Height Map Predictions

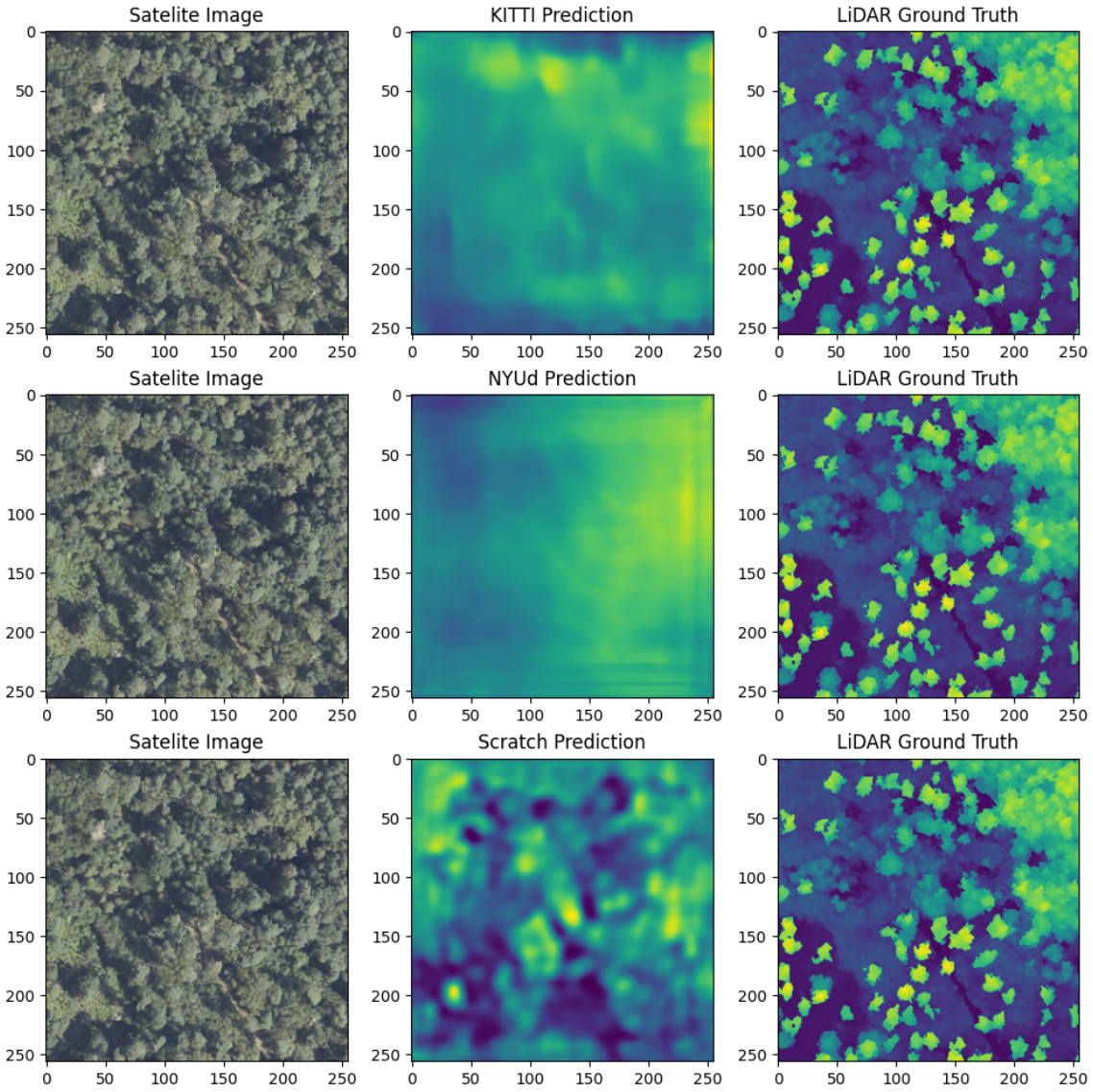


Figure 14 Visualization of Canopy Height Map Predictions

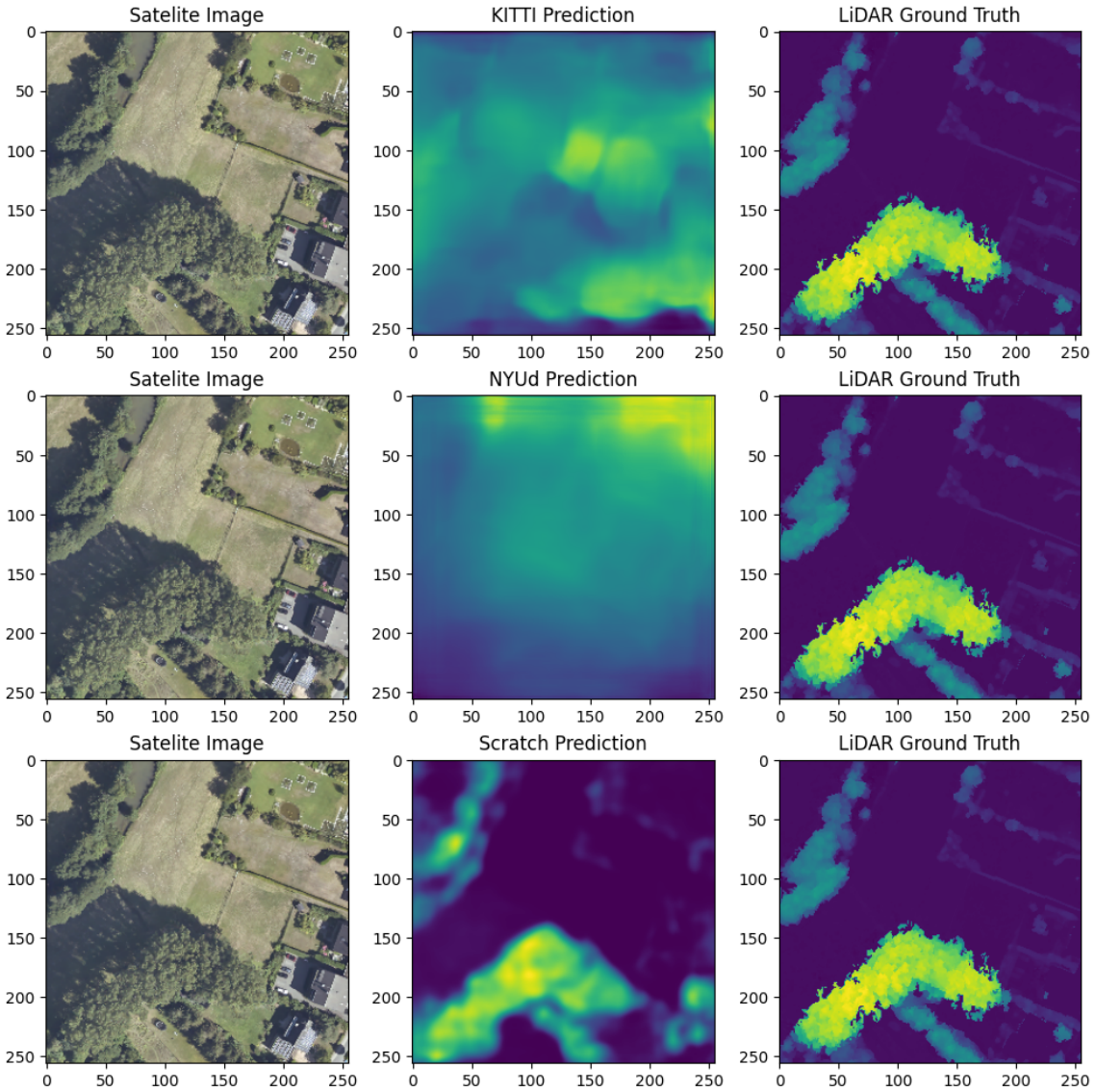


Figure 15 Visualization of Canopy Height Map Predictions

Bibliography

[Boj+21] Piotr Bojanowski et al. *DINO and PAWS: Advancing the state of the art in computer vision*. Apr. 30, 2021. URL: https://ai.meta.com/blog/dino-paws-computer-vision-with-self-supervised-transformers-and-10x-more-efficient-training/ (visited on 02/21/2024).

[Car+21] Mathilde Caron et al. *Emerging Properties in Self-Supervised Vision Transformers*. May 24, 2021. DOI: 10.48550/arXiv.2104.14294. arXiv: 2104.14294[cs]. URL: http://arxiv.org/abs/2104.14294 (visited on 11/02/2023).

[Dao+22] Tri Dao et al. *FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness*. June 23, 2022. DOI: 10.48550/arXiv.2205.14135. arXiv: 2205.14135[cs]. URL: http://arxiv.org/abs/2205.14135 (visited on 02/27/2024).

[Dar+23] Timothée Darcet et al. *Vision Transformers Need Registers*. Sept. 28, 2023. DOI: 10.48550/arXiv.2309.16588. arXiv: 2309.16588[cs]. URL: http://arxiv.org/abs/2309.16588 (visited on 11/02/2023).

[Dos+21] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. June 3, 2021. DOI: 10.48550/arXiv.2010.11929. arXiv: 2010.11929[cs]. URL: http://arxiv.org/abs/2010.11929 (visited on 02/20/2024).

[Gri+20] Jean-Bastien Grill et al. *Bootstrap your own latent: A new approach to self-supervised Learning*. Sept. 10, 2020. DOI: 10.48550/arXiv.2006.07733. arXiv: 2006.07733[cs,stat]. URL: http://arxiv.org/abs/2006.07733 (visited on 02/27/2024).

[He+15] Kaiming He et al. *Deep Residual Learning for Image Recognition*. Dec. 10, 2015. DOI: 10.48550/arXiv.1512.03385. arXiv: 1512.03385[cs]. URL: http://arxiv.org/abs/1512.03385 (visited on 02/20/2024).

[Hua+16] Gao Huang et al. *Deep Networks with Stochastic Depth*. July 28, 2016. DOI: 10.48550/arXiv.1603.09382. arXiv: 1603.09382[cs]. URL: http://arxiv.org/abs/1603.09382 (visited on 02/27/2024).

[HVD15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. *Distilling the Knowledge in a Neural Network*. Mar. 9, 2015. DOI: 10.48550/arXiv.1503.02531. arXiv: 1503.02531[cs,stat]. URL: http://arxiv.org/abs/1503.02531 (visited on 02/16/2024).

[LeC20] Yann LeCun. “Yann LeCun - Self Supervised Learning | ICLR 2020”. ICLR 2020. June 6, 2020. URL: https://www.youtube.com/watch?v=8TTK-DdOH9U (visited on 02/03/2024).

[LM21] Yann LeCun and Ishan Misra. *Self-supervised learning: The dark matter of intelligence*. Mar. 4, 2021. URL: https://ai.meta.com/blog/self-supervised-learning-the-dark-matter-of-intelligence/ (visited on 02/03/2024).

[Oqu+23] Maxime Oquab et al. *DINOv2: Learning Robust Visual Features without Supervision*. Apr. 14, 2023. DOI: 10.48550/arXiv.2304.07193. arXiv: 2304.07193[cs]. URL: http://arxiv.org/abs/2304.07193 (visited on 11/02/2023).

[Piz+22] Ed Pizzi et al. *A Self-Supervised Descriptor for Image Copy Detection*. Mar. 25, 2022. DOI: 10.48550/arXiv.2202.10261. arXiv: 2202.10261[cs]. URL: http://arxiv.org/abs/2202.10261 (visited on 02/26/2024).

[Tol+24] Jamie Tolan et al. “Very high resolution canopy height maps from RGB imagery using self-supervised vision transformer and convolutional decoder trained on aerial lidar”. In: *Remote Sensing of Environment* 300 (Jan. 1, 2024), p. 113888. ISSN: 0034-4257. DOI: 10.1016/j.rse.2023.113888. URL: https://www.sciencedirect.com/science/article/pii/S003442572300439X (visited on 02/29/2024).

[Zho+22] Jinghao Zhou et al. *iBOT: Image BERT Pre-Training with Online Tokenizer*. Jan. 27, 2022. DOI: 10.48550/arXiv.2111.07832. arXiv: 2111.07832[cs]. URL: http://arxiv.org/abs/2111.07832 (visited on 02/27/2024).

Declaration of Authorship

I hereby declare that, to the best of my knowledge and belief, this thesis titled *DINO: Learning Robust Visual Features without Supervision* is my own, independent work. I confirm that each significant contribution to and quotation in this thesis that originates from the work or works of others is indicated by proper use of citation and references; this also holds for tables and graphical works.

Münster, 29.02.2024

Tobias Zimmermann

Consent Form

Name: Tobias Zimmermann
Title of Thesis: DINO: Learning Robust Visual Features without Supervision

What is plagiarism? Plagiarism is defined as submitting someone else's work or ideas as your own without a complete indication of the source. It is hereby irrelevant whether the work of others is copied word by word without acknowledgment of the source, text structures (e.g. line of argumentation or outline) are borrowed or texts are translated from a foreign language.

Use of plagiarism detection software. The examination office uses plagiarism software to check each submitted bachelor and master thesis for plagiarism. For that purpose the thesis is electronically forwarded to a software service provider where the software checks for potential matches between the submitted work and work from other sources. For future comparisons with other theses, your thesis will be permanently stored in a database. Only the School of Business and Economics of the University of Münster is allowed to access your stored thesis. The student agrees that his or her thesis may be stored and reproduced only for the purpose of plagiarism assessment. The first examiner of the thesis will be advised on the outcome of the plagiarism assessment.

Sanctions Each case of plagiarism constitutes an attempt to deceive in terms of the examination regulations and will lead to the thesis being graded as "failed". This will be communicated to the examination office where your case will be documented. In the event of a serious case of deception the examinee can be generally excluded from any further examination. This can lead to the exmatriculation of the student. Even after completion of the examination procedure and graduation from university, plagiarism can result in a withdrawal of the awarded academic degree.

I confirm that I have read and understood the information in this document. I agree to the outlined procedure for plagiarism assessment and potential sanctioning.

Münster, 29.02.2024

Tobias Zimmermann