Darmstadt University of Applied Sciences

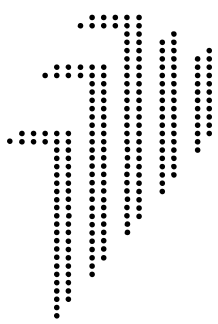
Computer Vision Seminar

Unearthing Surgical-Dino

Author
Tilman Seeßelberg

Reviewer
Prof. Dr. Andreas Weinmann, Darmstadt University of Applied Sciences
Vladyslav Polushko, Darmstadt University of Applied Sciences

Date of Submission: July 24, 2024



h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Abstract

Kurzzusammenfassung

Contents

List of Figures

List of Tables

List of abbreviations

MLP Multi Layer Perceptron

CNN Convolutional Neural Network

FM Feature Map

kNN k-Nearest Neighbors

MDE Monocular Depth Estimation

NLP Natural Language Processing

RNN Recurrent Neural Network

SOTA State of the Art

ViT Vision Transformer

Chapter 1
Introduction

[**WIP**:*Rewrite and get to the topic (Surgical Dino) quicker*] Foundation models are gaining significant traction in the field of computer vision, driven by extensive research in unsupervised pretraining tasks for large-scale models such as Vision Transformers (ViTs) and Convolutional Neural Networks (CNNs). These models promise to become the new standard in machine learning for computer vision, offering substantial advantages over training networks from scratch. By leveraging pretrained models and fine-tuning them, researchers can achieve impressive results, often surpassing previous state-of-the-art performance while reducing computational resources and the need of large labled datasets, due to the foundation models already having learned meaningful representations of the input data. This is particularly advantageous in the medical field. In disciplines like endoscopy, where creating labeled datasets is expensive and time-consuming, foundation models offer a promising solution. For instance, in endoscopy, accurate depth maps are crucial for precisely navigating the endoscope within the patient. By fine-tuning a pretrained foundation model, significant advancements can be made compared to training from scratch.

This report delves into the findings of the paper *Surgical-DINO: Adapter Learning of Foundation Models for Depth Estimation in Endoscopic Surgery* by Cui et al. [1]. The paper introduces a fine-tuning regimen for a computer vision foundation model, DINOv2, developed by Oquab et al. [2], specifically tailored for the task of Monocular Depth Estimation (MDE) in endoscopy. Using the comparatively small dataset SCARED dataset [3] with 17 607 unique images, each accompanied by a corresponding ground truth depth map, this work highlights the potential of foundation models to accelerate the development of state-of-the-art techniques, even in domains with limited data availability.

This report will first dive into Vision Transformer (ViTs) in Section refsec:vision-transformers, followed by a detailed explanation of the used foundation model DINOv2 in Section 2.3 after which the basics of depth estimation are layed out in Section 2.4 before the details of fine-tuning foundation models are explained in Section 2.5 before. [**WIP**:*polish the last sentence*]

Chapter 2

Theoretical Foundations

2.1 Vision Transformer

In order to understand the paper *Surgical-Dino* being examined in this report, it is crucial to first comprehend the algorithm used as the backbone of the suggested model: Vision Transformers (ViTs). ViTs, as introduced by Dosovitskiy et al. in 2020 [4], build on the work of Vaswani et al. [5], adapting the transformer architecture for image classification, which until then was predominantly used for natural language processing (NLP). ViTs leverage the transformer's strengths in handling sequential data by splitting an image into a sequence of smaller patches, e.g., 16×16 pixels, and processing them through the transformer architecture as if they were tokens, as explained below.

The advantage of ViTs over conventional Convolutional Neural Networks (CNNs), which most previous state-of-the-art (SOTA) models for computer vision tasks were based on, lies in their ability to capture not only local features hierarchically but also global context information that may span different regions of the image [6]. Additionally, they tend to scale better with the amount of data and parameters compared to popular CNN architectures but tend to perform worse when trained on smaller datasets from scratch [7]. The original transformer architecture, designed for machine translation, consists of an encoder-decoder structure that processes input sequences through an *Embedding Layer*, *Positional Encoding*, *Encoder* containing *attention heads* and *MLPs*, a *Decoder* and a *prediction head*. The following describes these components in the context of the original transformer architecture illustrated in Figure 2.1.

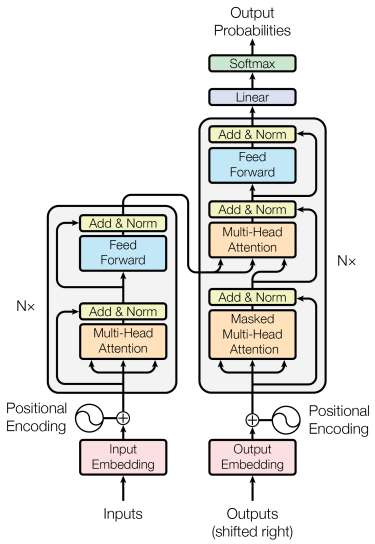


Figure 2.1: The transformer architecture (image from [5])

Embedding Layer: To ingest a sequence or an image into a transformer, it is first split into smaller pieces called tokens, which can be words, subwords and are then mapped into continuous vector representations with dimension d_{model} called *token vectors*. The number of input tokens

is denoted as N. Vision transformer handles embedding slightly differnt creating tokens by splitting the images of dimension $\mathbb{R}^{H\times W\times C}$ with H being the height, W the width and C the number of channels into N patches of dimension $\mathbb{R}^{P\times P\times C}$ where $N = \frac{H\times W}{P^2}$. Each patch then is flattened into an 1D vector and transformed to a length of d_{model} using a linear layer resulting in the *token vector*. The trade-off is to split the sequence or image in as few parts as possible, which improves the inference of the model later on, while staying flexible enough for new unseen data, which is especially relevant for NLP tasks where often dictionaries are used to convert text to token vectors unlike ViTs where the data is already numeric and can directly be used as a token vector withouth a dictionary. These *token vectors* form the columns of the *Embedding Matrix* which, after the modification of the next step, will be the input of the *Encoder*.

Positional Encoding: Following the embedding a *positional encoding* is added to each *token vector* in the *Embedding Matrix*, which gives the model a hint of the order of the tokens in the sequence. This is needed as Transformers do not have an inductive bias about token order, unlike predecessors like Recurrent Neural Networks (RNNs) [8] or Long Short-Term Memory (LSTM) networks [9]. The *Positional Encoding* is usually either learned in form of a linear layer, or calculated using a fixed function such as the sine-cosine function as done for the original Transformer.

Encoder: Typically an *Encoder* consists of multiple *Transformer Blocks* arranged sequentialy where on is build as described below: The *Embedding Matrix* with the *Positional Encoding* added, first is processed by a *Multi-Head Attention* consisting of *h Attention Heads*. The original paper achieved good results with $h = 8$. An Attention head first runs the *Embedding Matrix* through three linear layers W^Q, W^K & W^V in parallel, producing three matrices: *query matrix* Q, *key matrix* K, and *value matrix* V each $\in \mathbb{R}^{d_{\text{model}}\times d_k}$ with d_{model} defining the dimension the token vector is projected on to (512 in the original Transformer) and $d_k = \frac{d_{\text{model}}}{h}$ reducing the number of trainable parameters as it seams to be more parameter efficient to traine more attention heads with fewer parameters than fewer attention heads with more parameters. These terms are analogous to database operations, where the query matrix is the search query, the key matrix is the database, and the value matrix is the data in the database. This should convey the intuition behind the attention mechanism. In *Multi-Head Attention* the *Embedding Matrix* is processed by several *Attention Heads* in parallel to calculate the attention score using the Scaled Dot-Product Attention method as shown in Equation 2.1. First the query matrix Q and the transposed key matrix K are multiplied, devided by the square of the dimension of the matrices d_k, to increase the gradients of the softmax function, and then scaled using softmax to a range of $0-1$. This dot product is the *Attention Matrix* of dimension $\mathbb{R}^{N^2}$ which contains the *Attention Scores* of each token relating, or attending, to each other which can be interpreted as probabilities of how much a certain token is correlated to another token [7]. This *Attention Matrix* is then multiplied with the value matrix V resulting in the weighted value matrix. Having h attention heads leads to h weighted value matrices which get concatenated and processed by a final linear layer $W^O \in \mathrm{R}^{hd_k\times d_{\text{model}}}$ which transformes the outputs back to the dimension of the *Embedding Matrix*. This transformed *Embedding Matrix* then is added to the unmodified input of the Transformer Block, which is called the *Skip Connection*.

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

Decoder: The original Transformer architecture includes an decoder block which iterativley generates the output sequence token by token. While both encoder and decoder consist of the same building blocks, the decoder uses a masked multi-headed attention mechanism, zeroing

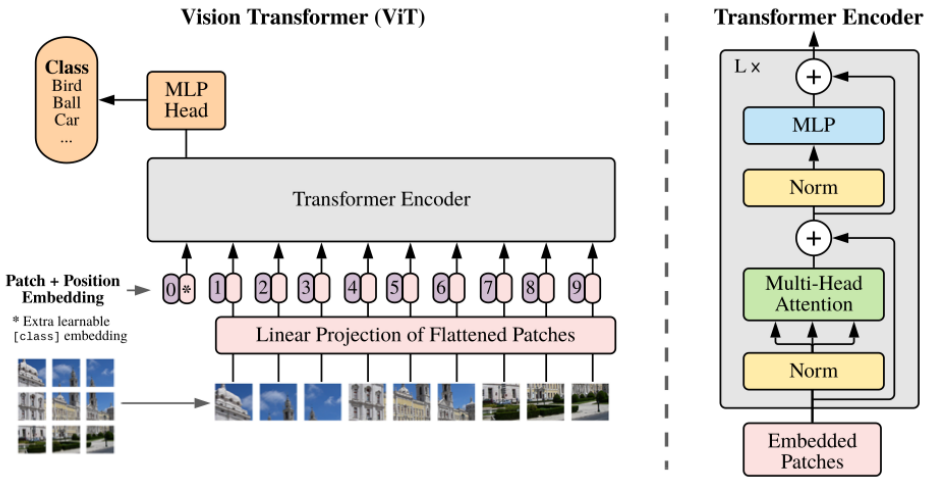


Figure 2.2: The Vision Transformer architecture (copied from [4]).

out the lower left triangle of the *Attention Matrix* containing the *Attention Scores* to prevent the model from attending to future tokens. Following that a multi-head cross attention is applied connecting the output of the encoder with the decoder. Since then works such as the BERT framework by Devlin et al. [10] have shown that the decoder is not inherently needed for tasks like language translation, or text generation allowing for a simplified *Encoder Transformer* architecture which is also used in ViTs.

Prediction Head: Following the *Encoder* and *Decoder* blocks usually a MLP or CNN follows to processe the transformed *Embedding Matrix* depending whether the task is a classification, regression or generation task. For classification and regression the prediciton heads often only process one or more specific tokens, either the last sequence token as in the original transformer, or a prepending class token as done in the ViT or BERT architectures. For generative ViTs such as ones used for depth estimation, use the whole *Embedding Matrix*, sometimes even from different transformer blocks, to generate the output[1].

A compact summary of differences between the Transformer and ViT architecure can be seen in the Appendix A.

Using this simplified Transformer architecture, as visualized in Figure 2.2, ViTs have demonstrated superior performance over state-of-the-art convolutional neural networks (CNNs) in several image classification tasks [11]. Since the original ViT paper, transformers have found numerous applications in the computer vision field beyond image classification. For example, ViTs have been successfully applied in anomaly detection to identify unusual patterns in manufacturing images, and in medical imaging for monocular depth estimation in endoscopy [12]. These applications benefit from ViTs' ability to analyze the global context of images, and their performance is validated in diverse domains [13]. Although initially requiring significantly more training data than CNNs when trained from scratch, ViTs match or surpass state-of-the-art performance when finetuned, highlighting the potential of pre-training on large datasets and the flexibility of transformers in processing different types of data.

2.2 Unsupervised Pretraining

[**WIP**:*This Section is undergoing corrections*] Now that ViTs allow for the creation of large-scale models for computer vision tasks, the need arises to train these models in an unsupervised manner to achieve even better performance by using even larger datasets without any lables. Unsupervised learning involves training a network without any labels for the data, which facilitates the creation of huge datasets with millions or even billions of samples. While these models often already show impressive results using clustering algorithms such as k-nearest-neighbor (KNN) clustering, they usually exhibit significant improvements when fine-tuned in a supervised manner for different downstream tasks, even when using small datasets and with limited computational resources available. The overall term for these general models is foundation models [14]. For NLP tasks, unsupervised approaches such as masked language modeling, as used in BERT [10], or generative pretraining, as used in GPT [15], have been successfully applied. While there are some works utilizing the same idea of predicting the next token, or a missing token, for computer vision tasks [16], [17] there are alternative approaches tending to yield even better results.

One alternative approach that has gained popularity is contrastive learning, as proposed by Hadsell et al. [18], using siamese networks consisting of two identical networks in parallel to learn to differentiate between similar and different images. The task of predicting whether the images presented to both networks are similar or different is called a contrastive task and is needed to prevent model collapse to a trivial solution, which would occur if the training task were to predict the output of the other network always given the same input. This idea was further developed by Chen et al. [19] in 2020, who successfully applied contrastive learning to a ResNet50 Network improving its accuracy on the ImageNet-2012 Dataset with over 10 000 classes to almost 86% being within the top-5 models at the time.

Grill et al. [20] then introduced the Bootstrap Your Own Latent (BYOL) framework eliminating the need for negative samples to prevent model collaps by changing the way both networks udpate their weights. Grill et al. figured, it, to prevent model collaps, they would randomly initialize the second network, the teacher network, and freeze its weights, the student network would still while not achieving good results in down stream tasks, definitly learns meaninfull representations of the images. To further improve this they figured that updating the teacher weights based on the moving average of the student weights would prevent the model from collapsing to a trivial solution. This however is yet to be mathematically proven, but the authors empirically show, that the model does not collaps and intentionally made designchoices to prevent that.**dig into the reasoning they give in the paper**

Grill et al. [20] then introduced the Bootstrap Your Own Latent (BYOL) framework, eliminating the need for negative samples needed for contrastive learning, to prevent model collapse by changing the way both networks update their weights. Grill et al. found that randomly initializing the second network, the teacher network, and freezing its weights and only updating the student networks weights during training already led the student network to learn meaningful representations of the images. The student network, while not achieving high accuracies in classification tasks, still learned meaningful representations of the images. To further improve this, they found that updating the teacher weights based on the moving average of the student weights, instead of using backpropagation would prevent the model from collapsing to a trivial solution. This has yet to be mathematically proven, but the authors empirically show that the model does not collapse which was veryfied by numerouse works[**WIP**:*Add references*].

The pipeline of BYOL starts with generating several augmented crops and providing each network, the student and teacher network, with a different random augmentation of the same

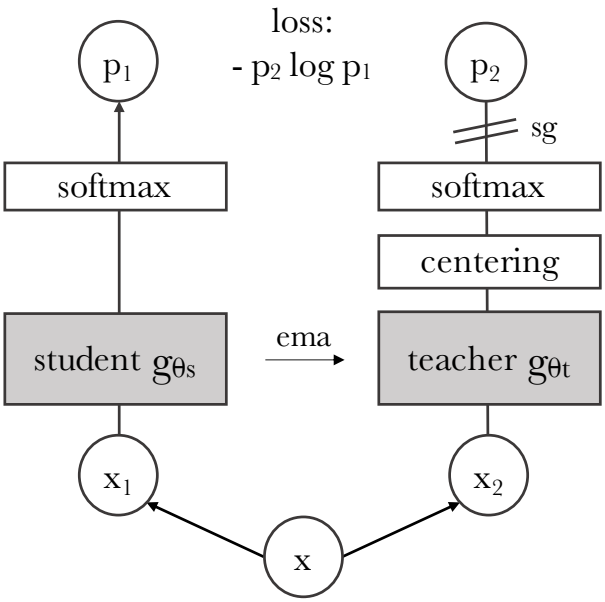


Figure 2.3: The DINOv1 Architecture showing the Student-Teacher Setup. *sg* stands for stop gradient (image from [21])

image. Next, both networks process their input inside a ViT encoder, called the representation block. This is followed by a few MLPs for dimensionality reduction. The student network also has another prediction MLP at the end, which introduces asymmetry into the model, likely making the trivial solution of outputting a constant vector less likely. Afterward, a scaled cosine similarity between the output of the student network $q_\theta(z_\theta)$ and the teacher network $\bar{z}'_\xi$ is calculated as shown in Equation 2.2. Then, the inputs of the networks are switched, so both networks get the images of the other one. The cosine similarity is again calculated, and the loss is calculated as the sum of both similarities and used to optimize the student network's weights. The weights of the teacher network ξ are updated as the exponential moving average of the student network's weights θ as shown in Equation 2.3, with τ set to 0.99 in the BYOL paper. After training the representation block inside the student network can be used for downstream tasks.

$$\mathcal{L}_{\theta,\xi} \triangleq \left\| \bar{q}_\theta(z_\theta) - \bar{z}'_\xi \right\|_2^2 = 2 - 2 \cdot \frac{\langle q_\theta(z_\theta), z'_\xi \rangle}{\|q_\theta(z_\theta)\|_2 \cdot \|z'_\xi\|_2} \tag{2.2}$$

$$\xi \leftarrow \tau\xi + (1-\tau)\theta \tag{2.3}$$

2.3 DINO: Unsupervised Learning with Vision Transformers

[**WIP**:*This Section is undergoing corrections*]

The **DI**stillation with **NO** labels (DINO) framework, introduced by Caron et al. in 2021 [21], is an approach for implementing self-distillation for Vision Transformers (ViTs). This method is inspired by the findings of the BYOL paper, as introduced in Section 2.2, and is built upon by the *Surgical-Dino* paper examined in this report. The goal of DINO is similar to that of BYOL: to train a ViT encoder for image representation learning from scratch in an unsupervised manner to be later finetuned for downstream tasks. The key differences between DINO and BYOL are the training task, image augmentation, and the architecture of the student network s and the teacher network t. The DinoV1 architectur is shown in Figure 2.3:

- **Training Task:** DINO minimizes the mean cross-entropy (Equation 2.4) between the student and teacher network outputs of two different crops of each image.
- **Image Augmentation:** DINO uses cropping as the sole form of augmentation. It differentiates between two kinds of crops: global crops, which contain more than 50 % of the image, and local crops, which contain less than 50 % of the image. The teacher only receives the global crops, while the student sees both global and local crops. This way, the student learns to transfer information from local to global contexts.
- **Architecture:** In the DINO framework, both the student and teacher networks have the same architecture, unlike BYOL, which uses an asymmetric architecture with an additional projection head only added to the student.

The loss function for DINO is given by:

$$\mathcal{L} = -\sum_{x} P_t(x) \log P_s(x) \tag{2.4}$$

where $P_t(x)$ is the output probability distribution of the teacher network and $P_s(x)$ is the output probability distribution of the student network as calcualted with Equation 2.5.

The output probability distribution $P_t(x)$ is computed using a softmax function, scaled by a temperature parameter τ:

$$P_t(x) = \text{softmax}\left(\frac{g_{\theta_t}(x)}{\tau}\right) \tag{2.5}$$

DINO introduces a novel approach to unsupervised learning by leveraging self-distillation and using the same network architecture for both the student and teacher. This method has shown promising results in learning high-quality image representations that can be effectively used in various downstream tasks and is even further improved in the updated DinoV2 Framework. [**WIP**:*Information about how DinoV2 Changes*]

2.4 Monocular Depth Estimation

Monocular Depth Estimation (MDE), also known as Monocular Dense Estimation, is a discipline where, given a single image, the model learns to predict the depth of every pixel. This task often scales depth values from 0 to 255. Monocular depth estimation holds significant potential in fields such as robotics, autonomous driving, and medicine and also gets more and more important for generative image and video creation. These areas require precise 3D representations of the environment, but sensors capable of capturing depth information can be too expensive or impractical due to space constraints or limited availability in general.

In generall there are two main approaches to monocular depth estimation: generative models e.g. based on diffusion networks and discriminative models e.g. based on CNNs or more recently hybrid models combining CNNs with Vision Transformers (ViTs) as demonstrated by works such as [2], [12], [22].

Recently, there has been a shift towards using hybrid models that combine CNNs with Vision Transformers (ViTs), as demonstrated by works like Ranftl et al. [12]. In these hybrid models, the ViT typically encodes the image, transforming it into feature tokens enriching them with

additional information. These tokens are then usually processed by a CNN-based prediction head to generate a depth map.

Monocular Depth Estimation (MDE), also known as Monocular Dense Estimation, is a task where, given a single image, the model learns to predict the depth of every pixel. This task often scales depth values from 0 to 255. Monocular depth estimation holds significant potential in fields such as robotics, autonomous driving, and medicine and is becoming increasingly important for generative image and video creation. These areas require precise 3D representations of the environment, but sensors capable of capturing depth information can be too expensive or impractical due to space constraints or limited availability in general.

In general, there are two main approaches to monocular depth estimation: generative models (e.g., based on diffusion networks) and discriminative models (e.g., based on CNNs or more recently, hybrid models combining CNNs with Vision Transformers (ViTs) as demonstrated by works such as [2], [12], [22]).

Recently, there has been a shift towards using hybrid models that combine CNNs with Vision Transformers (ViTs), as demonstrated by works like Ranftl et al. [12]. In these hybrid models, the ViT typically encodes the image, transforming it into feature tokens enriched with additional information. These tokens are then usually processed by a CNN-based prediction head to generate a depth map.

Using MLPs however can also yield good results as shown by the Surgical-Dino. [**WIP**:*Improve wording for segway to Surgical-Dino*]

2.5 Model Fine-Tuning

Foundation Models (FMs), as defined in [14], are models trained on a large variety of data using self-supervised learning, sometimes for weeks on hundreds of graphics processing units (GPUs), as exemplified by the GPT-3 model [23] or the Florence CV Foundation Model by Yuan et al. [24]. These FMs can then be fine-tuned using few-shot training to tailor the FM to a specific task using several orders of magnitude less data than would be needed for training a model from scratch. This approach saves significant computational cost and time since it is often sufficient to retrain only small parts of the FM, with most of the parameters/weights being frozen.

Mathematically, the straight forward way of fine-tuning a model is by retraining all parameters, which can be written as in Equation 2.6 with W_0 being the pretrained weights and ΔW the change in weights during fine-tuning. This However still is relativley computationally intesive as the backpropagation has to be done for all parameters.

$$W = W_0 + \Delta W \tag{2.6}$$

A more efficient way of fine-tuning weight matrices is Low-Rank Adaptation (LoRA), as suggested by Hu et al. [25] in the context of transformer fine-tuning. Hu et al. assume, based on the work of Aghajanyan et al. [26], that the updated weight matrices δW_n inside the attention heads of the transformer have a rank much lower than their dimensionality. To utilize this assumption for more efficient finetuning, the updated weight matrices δW_n are decomposed into two matrices $A \in \mathbb{R}^{d \times r}$ and $B \in \mathbb{R}^{r \times k}$ with $r \ll d$ and $r \ll k$, as shown in Equation 2.7. Hu et al. aimed to use a maximum of 18M parameters for fine-tuning the 173B parameter GPT-3

model and conducted a grid search for optimizing the rank r and which weight matrices to update. They found that for their use case, a rank $r = 4$ and applying updates only to all key matrices K and value matrices V in the attention heads of the transformer performed the best. Besides significantly lowering the computational cost by optimizing only the A and B matrices, fine-tuning using LoRA does not increase inference times, as the decomposed matrices are recomposed and added to the original weight matrices W_0 after fine-tuning, thus not adding any additional computation to the model's forward pass. LoRas are also referenced as LoRa side layers, as it does not add new parameters sequentialy, but extends the existing weight matrixes in parallel as can be seen in Figure 2.4.

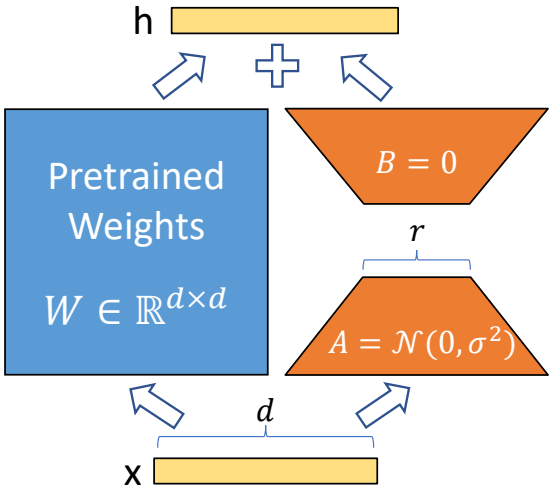


Figure 2.4: LoRa architecture with the docomposed $\Delta W_n = AB$ and their initialization method (graphic lifted from [25])

$$W = W_0 + BA \tag{2.7}$$

There are more approaches for fine-tuning transformers, which can be explored in further detail in [27].

Chapter 3

Unearthing Surgical-Dino

3.1 Surgical-Dino

Motivation

[**WIP**:*Information about the motivation of Dino*] Surgical Dino as introduced by Cui et al. [1] addresses the challange of monocular depth estimation in endoscopy using a fine-tuned version of the DinoV2 ViT foundation model, which is one of the biggest foundation models available to the public. Most current works using foundation models in the surgical domain deal with segmentation and object detection tasks. There, however, is also a need for using foundation models for accurate depth estimation as it is a crucial step to improve technologies in the field of robotic surgery and has the potential of enhacing 3d reconstruction of interal organs and surgical navigation. While there are endoscopy instruments available using stereo cameras for depth estimation, these are only available on the most advanced surgical robots such as the Davinci Xi, which are not available in most hospitals. Hence the desire to create appropriate depth estimations using only a singel image using ML algorithms such as CNNs and more recently transformer. While vision foundation models could already proof their value in more natural domains where there are large datasets available, but the authors of Surgical-Dino figured that the unadapted foundation models experience a significant drop in performance when applied to the surgical domain, which makes sense as this kind of data usually is not inculded in the training data of the foundation models as it often is confidential and not availbale at the scale needed for training such models from scratch. DinoV2 was choosen as the basis of Surgical-Dino by the authors as it was the most recent foundation model available and already achieved promising results in several computer vision tasks including depth estimation in more general settings such as autonomous driving or general segmentation.

Implementation

[**WIP**:*Information about the technicalities of the model*] To adapt DinoV2, which was introduced in 2.3, for depth estimation in the surgical domain the authors develope a pipeline to finetune the smallest of the available pretrained DinoV2 Encoders called *DINOv2-S* with 12 transformer blocks and a token vector size of $d_{\text{model}} = 784$ each block containing multi-head attention blocks with 6 heads and additionally add a new depth prediction head to transfer the Transformer Block outputs, also known as *Embedding Matrix* or *Feature Map* into a grayscale depth map with pixel values between 0 to 255. For the finetuning all weights of the *DINOv2-S* model were frozen and extented by adding trainable LoRa side layers as introduced in Section 2.5 to all Attention Heads' query and value matrices inspired by the findings of Zhang et al. [28]. Using LoRa side layers was decided on by the authors as they allow to fine tune the model with very limited computational ressources and small datasets, do not increase the inference time later on and are

easy to implement. In case of this work a single highend consumer grade graphics card was enough to train the model in a reasonable amount of time.

Following the transformer encoder blocks a depth decoder head is appended which converts not only the *Feature Maps* of the last transformer block, but the outputs of every third block. Before the decoder head the *Feature Maps* are unflattend

Following the transformer encoder blocks a depth decoder head is appended which converts not only the *Feature Maps* of the last transformer block, but the outputs of every third block. Before the decoder head, each *token vector* of inside the *Feature Maps* gets unflattened and upsampled to $f\times$ the original patch resolution using bilinear interpolation to increase the resolution of the depth map. These transformed features are then concatenated along the channel dimension to form Each Feature Matrix then is reassambled and the different Features Maps are concatenated along the channel dimension. Finally, the depth decoder utilizes a 1×1 convolutional layer to predict the depth of each pixel individually, treating depth prediction as a classification problem by dividing the depth range into discrete bins.

First the outputs not just of the last transformer block but also of the transformer blocks 3, 6 and 9 are processed by the depth prediction head. Before that the transformer block outputs are unflattend, to have the same shape as the input patches, bi-linearly upsampled 4x to increase the resolution for the depth map. At the end the Depth map is downsampled again which improves the overall quality of the depth map.

Training

[**WIP**:*Results of the Paper*]

The first step for creating Surgical-DINO is to fine-tune the DinoV2 Encoder following the DinoV2 pre-training procedure except with the encoder weights frozen

Related Works

Reasoning of the authors for how they derive at the fine-tuning regiment as they do

Chapter 4
Discussion

[**WIP**:*Discussing the Results in context of related works*]

Bibliography

[1] B. Cui, M. Islam, L. Bai, and H. Ren, "Surgical-dino: Adapter learning of foundation models for depth estimation in endoscopic surgery," *International Journal of Computer Assisted Radiology and Surgery*, vol. 19, no. 6, pp. 1013–1020, Mar. 2024, ISSN: 1861-6429. DOI: 10.1007/s11548-024-03083-5.

[2] M. Oquab, T. Darcet, T. Moutakanni, *et al.*, "Dinov2: Learning robust visual features without supervision," 2023. DOI: https://doi.org/10.48550/arXiv.2304.07193.

[3] M. Allan, J. Mcleod, C. Wang, *et al.*, *Stereo correspondence and reconstruction of endoscopic data challenge*, 2021. DOI: 10.48550/ARXIV.2101.01133.

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, *An image is worth 16x16 words: Transformers for image recognition at scale*, 2020. DOI: 10.48550/ARXIV.2010.11929.

[5] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, 2017. DOI: 10.48550/ARXIV.1706.03762.

[6] A. Khan, Z. Rauf, A. Sohail, *et al.*, "A survey of the vision transformers and their cnn-transformer based variants," *Artificial Intelligence Review*, vol. 56, no. S3, pp. 2917–2970, Oct. 2023, ISSN: 1573-7462. DOI: 10.1007/s10462-023-10595-0.

[7] K. Han, Y. Wang, H. Chen, *et al.*, "A survey on vision transformer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 1, pp. 87–110, Jan. 2023, ISSN: 1939-3539. DOI: 10.1109/tpami.2022.3152247.

[8] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, *Recent advances in recurrent neural networks*, 2018. DOI: 10.48550/ARXIV.1801.01078.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, ISSN: 1530-888X. DOI: 10.1162/neco.1997.9.8.1735.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2018. DOI: 10.48550/ARXIV.1810.04805.

[11] J. Maurício, I. Domingues, and J. Bernardino, "Comparing vision transformers and convolutional neural networks for image classification: A literature review," *Applied Sciences*, vol. 13, no. 9, p. 5521, Apr. 2023, ISSN: 2076-3417. DOI: 10.3390/app13095521.

[12] R. Ranftl, A. Bochkovskiy, and V. Koltun, "Vision transformers for dense prediction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2021, pp. 12 179–12 188.

[13] S. Jamil, M. Jalil Piran, and O.-J. Kwon, "A comprehensive survey of transformers for computer vision," *Drones*, vol. 7, no. 5, p. 287, Apr. 2023, ISSN: 2504-446X. DOI: 10.3390/drones7050287.

[14] R. Bommasani, D. A. Hudson, E. Adeli, *et al.*, *On the opportunities and risks of foundation models*, 2021. DOI: 10.48550/ARXIV.2108.07258.

[15] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, *et al.*, "Improving language understanding by generative pre-training," 2018.

[16] C. Wei, H. Fan, S. Xie, C.-Y. Wu, A. Yuille, and C. Feichtenhofer, "Masked feature prediction for self-supervised visual pre-training," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 14 668–14 678.

[17] H. Bao, L. Dong, S. Piao, and F. Wei, *Beit: Bert pre-training of image transformers*, 2021. DOI: 10.48550/ARXIV.2106.08254.

[18] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR'06)*, IEEE, 2006. DOI: 10.1109/cvpr.2006.100.

[19] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*, PMLR, 2020, pp. 1597–1607.

[20] J.-B. Grill, F. Strub, F. Altché, *et al.*, "Bootstrap your own latent - a new approach to self-supervised learning," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 21 271–21 284. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/f3ada80d5c4ee70142b17b8192b2958e-Paper.pdf.

[21] M. Caron, H. Touvron, I. Misra, *et al.*, "Emerging properties in self-supervised vision transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2021, pp. 9650–9660.

[22] L. Yang, B. Kang, Z. Huang, *et al.*, *Depth anything v2*, 2024. DOI: 10.48550/ARXIV.2406.09414.

[23] B. Yuan, Y. He, J. Davis, *et al.*, "Decentralized training of foundation models in heterogeneous environments," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35, Curran Associates, Inc., 2022, pp. 25 464–25 477. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/a37d615b61f999a5fa276adb14643476-Paper-Conference.pdf.

[24] L. Yuan, D. Chen, Y.-L. Chen, *et al.*, *Florence: A new foundation model for computer vision*, 2021. DOI: 10.48550/ARXIV.2111.11432.

[25] E. J. Hu, Y. Shen, P. Wallis, *et al.*, *Lora: Low-rank adaptation of large language models*, 2021. DOI: 10.48550/ARXIV.2106.09685.

[26] A. Aghajanyan, L. Zettlemoyer, and S. Gupta, *Intrinsic dimensionality explains the effectiveness of language model fine-tuning*, 2020. DOI: 10.48550/ARXIV.2012.13255.

[27] Y. Xin, S. Luo, H. Zhou, *et al.*, *Parameter-efficient fine-tuning for pre-trained vision models: A survey*, 2024. DOI: 10.48550/ARXIV.2402.02242.

[28] L. Zhang, C. Bao, and K. Ma, "Self-distillation: Towards efficient and compact neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021, ISSN: 1939-3539. DOI: 10.1109/tpami.2021.3067100.

Appendix A

Architecture Comparison Transformer and Vision Transformer

Component	Transformers	Vision Transformers (ViTs)
Embedding	Devides input sequence (e.g., a sentence) into tokens (e.g., syllables) and converts them into continuous vector representations, forming the embedding matrix.	Devides the input image into non-overlapping patches of size 16×16. Each patch is flattened and used as an embedding vector. All embedding vectors together form the embedding matrix.
Position Encoding	Uses fixed 1-d sinusoidal positional encodings to encode the position of each token in the sequence.	Uses learnable position embeddings to better capture the spatial relationships between image patches.
Encoder	Consists of a stack of identical layers, each with a multi-head self-attention mechanism and a position-wise MLP.	
Decoder	Consists of a stack of identical layers, each with a multi-head self-attention mechanism and an multi-head cross attention connecting the output of the encoder with the decoder. The cross attention is followed by a position-wise MLP.	Not applicable, as ViTs only use an encoder.
Attention Mechanism	Computes attention scores using query (Q), key (K), and value (V) matrices. Multi-head attention allows the model to attend to different parts of the input sequence simultaneously.	
Class Token	Not applicable.	Prepends a special class token to the embedding matrix which the prediction head uses for classification

Table A.1: Comparison of Components in the original Transformers by Vaswani et al. [5] and the original ViT by Dosovitskiy et al. [4], detailing key adaptations such as the use of image patches instead of textual tokens and positional embeddings tailored for spatial data, which allow ViTs to effectively handle visual contexts.