Darmstadt University of Applied Sciences

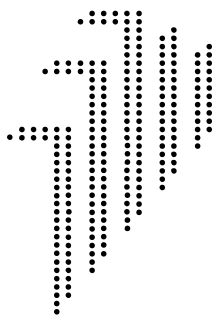
Computer Vision Seminar

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 19, 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

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, the first thing to understand is the algorithm used as the backbone of the suggested model which are 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, which until then were mostly used for natural language processing (NLP), for image classification. ViTs leverage the transformer's strengths in handling sequential data by splitting an image into a sequence of smaller, e.g. 16×16 pixel patches, and processing them through the transformer architecture just like tokens as explained below.

The advantage of ViTs over conventional Convolutional Neural Networks (CNNs), which most previouse state-of-the-art (SOTA) models for computer vision tasks were based on, is the ability to not only capture local features in a hirachical manner, but also to capture global context information of the image not necessarily being in the same region of the image [6]. Additionally they seem to scale better with the amount of data and paramters compared with popular CNN architectures, but are tending 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 *Embedding Layer*, *Positional Encoding*, *Encoder* containing *attention mechanisms* and *MLPs*, and a *prediction head*. The following describes these components in the context of the original transformer architecture which is illustrated in Figure 2.1.

Embedding Layer: To ingest a sequence into a transformer, it is first split into smaller pieces called tokens, which can be words, subwords, or characters, and are then mapped into continuous vector representations called *token vectors*. The goal is to split the sequence in as few üarts as possible while staying flexible for new unseen data. These *token vectors* form the columns of the *Embedding Matrix*.

Positional Encoding: Following the embedding a *positional encoding matrix* is added to 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].

Encoder: Typically an *Encoder* consists of multiple *Transformer Blocks* out in sequentialy which are described in the following: The *Embedding Matrix* with the *Positional Encoding* added, is then 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 learnable linear layers W^Q, W^K & W^V in parallel, producing three matrices: *query matrix* Q, *key matrix* K, and *value matrix* V each $\in \mathrm{R}^{d_{\text{model}} \times d_k}$ with d_{model} being the length of a token vector (512 in the original paper) and $d_k = \frac{d_{\text{model}}}{h}$. 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. These terms try to convey the intuition behind

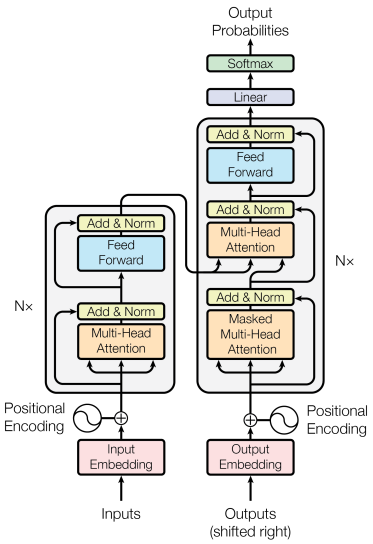


Figure 2.1: The transformer architecture (image from [5])

the attention mechanism. These matrices are processed by one or more *Attention Heads* as shown in Equation 2.1 to calculate the attention score using the Scaled Dot-Product Attention method. First the query matrix Q and the transposed key matrix K is multiplied, devided by the square of the dimension of the matrices d_k and then normalized using softmax to prevent exploding gradients. This dot product is the *Attention Matrix* of dimension $d_{\text{model}} \times d_{\text{model}}$ which contains the *Attention Scores* which can be interpreted as probabilities of how much a certain token is correlated to another token **Explained in a good manner here [7]**. This *Attention Matrix* then multiplied with the value matrix V resulting in the weighted value matrix. Having h attention heads leads to h weighted value matrices which are 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 encoder-decoder structure, where the encoder processes the input sequence and the decoder 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 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 was 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.

Prediction Head: Following the *Encoder* and *Decoder* blocks a usual MLP follows doing the job the network is trained to do, so either classification, such as the next token, a class the sequence is belonging to or processing the processed input sequence in any other way. These 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.

This methodology is largely copied by the ViT architecture, though slightly adapted as shown in Table A.

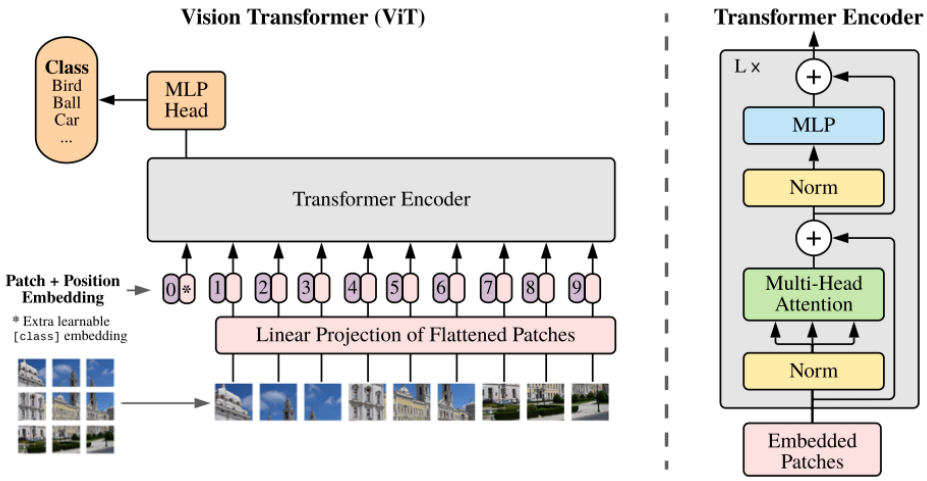


Figure 2.2: The Vision Transformer architecture (copied from [4]).

The ViT is first trained on a comparably large dataset with a large number of classes. Following this the and then is trained for a downstream classification task on a smaller dataset with a higher resolution.

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, many more use cases beyond image classification have emerged for Transformers in the computer vision field. For example, ViTs have been successfully applied in anomaly detection where they identify unusual patterns in manufacturing images, and in medical imaging [12] for monocular depth estimation in the endoscopy domain. These applications benefit from ViTs' ability to analyze the global context of images, demonstrating their potential in diverse and critical domains, [13]. Despite requiring significantly more training data than previous state-of-the-art CNN architectures, ViTs effectively leverage the transformer architecture to surpass the previous state of the art, highlighting the importance of unsupervised pre-training on large datasets and the flexibility of transformers in processing different types of data beyond natural language.

2.2 Unsupervised Pretraining

Now that ViTs allow to create large scale models for computer vision tasks the need arises to train these models in an unsupervised manner in order to yield even better performance. Unsupervised learning means the training of a network without any lables for the data, which facilitates the creation of huge datasets with millions or even billions of samples. While these models often already are showing impressive clustering results and can be used for clustering algorithms such as K Nearest Neigbhors (KNN) or similar, usually these models are afterwards fine-tuned in a supervised manner for different downstream tasks which usually takes comparably little datasets and little computational resources to achieve state-of-the-art performance. The overall term for these models is foundation models [14]. For NLP tasks approaches such as masked language modeling, as used in BERT [10], or generative pretraining, as used in GPT [15], have been successfully applied. For Images however the methodology of predicting the next token does not make as much sense as for text, as images itself are not sequential data.

There are already several works such as [16], [17] using the same methodologie of masked input modeling as suggested for Bert [10] which already improve the performance by several percent on the ImageNet classification dataset. However with around 86 % accuracy there is still more room for improvement and a need to find alternative approaches.

One of these was made popular among other works by hadsell et al. [18] and is called contrastive learning, using siemese networks consisting of two identical networks in parallel, to learn to differentiate between similar and different images. The task to predict if the images both networks were presented with are similar, or different is called a contrastive task and is needed to prevent model collaps to a trivial solution, if the train task would be to predict the output of the other network given the same input. This was idea was develpoed further by Chen et al. [19] in 2020 who applied contrastive learning to a ResNet50 Network.

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** The pipeline of BYOL starts with generating several augmented crops and providing each, the online and target network with a different randome augmentation of the same image. Next both networks process their input inside a ViT Encoder, called the representation Block. This is followed by a few MLP for dimensionality reduction. The student also has another prediction MLP which introduces an asymmetry into the model, likely making the trivial solution of outputting a constant vector less likely. Afterwards a scaled cosin similarity between the output of the online Network $q_\theta(z_\theta)$ and the target network $\bar{z}'_\xi$ is calculated as shown in Equation 2.2. Afterwards the inputs of the networks are switched, so both networks get the images of the other one. Then the cosin similarity is again calculated and the loss is calculated as the sum of both similarities and used to optimize the online networks weights. The weights of the target netowork ξ are updated as the exponential moving average of the online networks weights θ as shown in Equation 2.3 with τ being set to 0.99 in the BYOL paper.

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

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

2.3 DINO: Unsupervised Learning with Vision Transformers

The **DI**sitllation with **NO** labels (DINO) framework, introduced by Caron et al. in 2021 [21], is another approach of implementing self-distillation for ViTs inspired by the findings of the BYOL paper as introduced in Section 2.2 and being build on by the *Surgical-Dino* paper being examined in this paper. The goal of DINO is the same as for BYOL, to train a ViT Encoder for image representation learning from scratch in an unsupervised manner for later use in downstream tasks. The key differences from DINO to BYOL are the training task, the image augmentation,

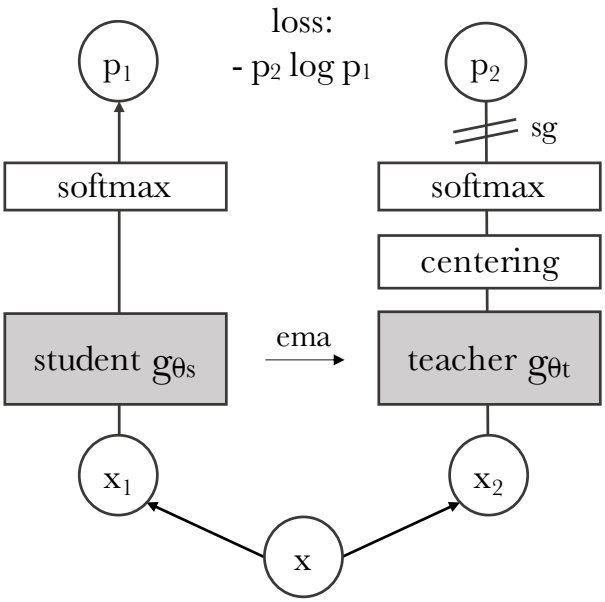


Figure 2.3: The DINOv1 Architecture showing the Student Teacher Setup. *sg* stands for stop Gradient (image from [21])

and the architecture of the student network s and the teacher network t which can be seen in Figure 2.3:

- **Training Task:** DINO tries to minimize the mean of the cross-entropy (Equation 2.4) between the student and teacher network's outputs of two different crops each.
- **Image Augmentation:** DINO augments the images only by cropping. DINO differentiates between two different kind of crops: Global crops contain more then 50 % of the image and local crops containing less then 50 % of the image. The teacher only gets the global crops while the student sees all the crops. This way the student has to learn the transfer from local to global.
- **Architecture:** The DINO framework uses a student teacher network where both the student and the teacher have the same architecture instead of having an asymetric architecture with another projection head added to the student.

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

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

[**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.

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 even for weeks on hunderds of graphic processing units (GPUs) as e.g. the GPT3 model [23] or the Florence CV Foundation Model by Yuan et al. [24]. These FMs can then fine-tuned doing few shot training, to tailor the FM to a specific task using several magnitutes less data as would be needed for training a model from scratch. This saves a lot of computational cost and time as it often is enough to retrain only small parts of the FM, and most of the parameters/weights get frozen as well as that cost can be saved dataset which can be relativley small. 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 Adaption (LoRA) as suggested by Hu et al. [25] in the context of transformer fine-tuning. Hut et al. assume based on the 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 then their dimensionality. In order to utilize this assumption to more efficiently fine-tune a transformer 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. set the goal to use a maximum of 18M parameters for fine tuning the 173B parameter GPT3 model and created a grid search for optimizing the rank r and which weight matrices to update. Eventually they figured for their usecase a rank $r = 4$ and only applying updates to all key matrices K and value matrices V in the attention heads of the transformer performed the best. Besides the advantage of significantly lowering the computational cost of only optimizing the A and B matrices, fine-tuning using LoRa does not increase inference times as the decomposed matrices are recomposed and added onto the original weight matrices W_0 after the fine tuning and therefor do not add any additional computation to the forward pass of the model.

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

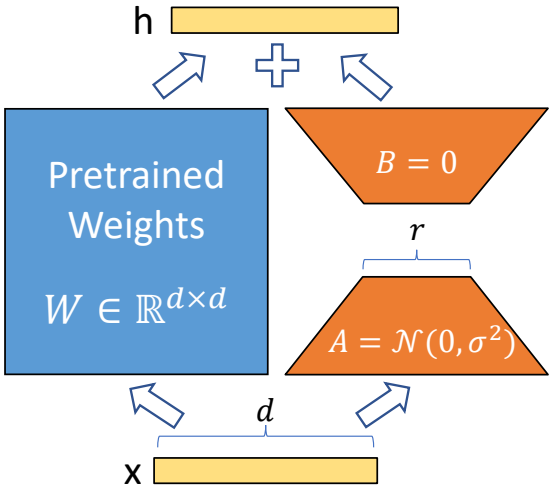


Figure 2.4: LoRa architecture with the docomposed $\Delta W_n = AB$ and their initialization method (graphic lifted from [25])

There are more approaches for fine tuning Transformer which can be read more about here [27].

Chapter 3

Unearthing Surgical-Dino

3.1 Surgical-Dino

[**WIP**:*Information about the motivation of Dino*] Surgical Dino as introduced by Cui et al. [1] adapts the DinoV2 architecture as introduced in Section 2.3 to the task of monocular depth estimation to use for endoscopy. Endoscopy is a medical procedure where a camera is inserted into the body to visualize the interior of organs or cavities for diagnostic or surgical purposes. While there are endoscopy instruments available using stereo cameras for depth estimation, these are often bulky and expensive. Hence the desire to create appropriate depth estimations using only a singel, monocular camera using ML. DinoV2 was choosen as starting point as it was the most recent foundation model and already achieved promising results in several computer vision tasks including depth estimation.

[**WIP**:*Information about the motivation of the authors*] To do this the authors develope a pipeline to adapt the smallest pretrained DinoV2 Encoder *DINOv2-S* with 12 Transformer Blocks and each multi-head attention heaving 6 hedas, and train a new depth prediction head to transfer the Transformer outputs to a depth map. The Surgical-Dino Model uses this *DINOv2-S* model, freezes its weighst and adds LORA matrices to the Attention Heads and adds a untrained linear layer following the Encoder to predict the depth map. Instead of fine tuning the model by updating all weights the authors applied the sugested method LORA as introduced in Section 2.5 as it reduces the number of parameters to be trained significantly.

[**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.

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.