Darmstadt University of Applied Sciences

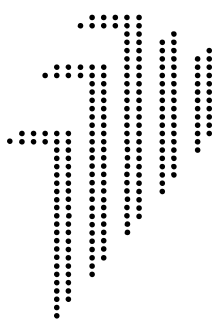
Computer Vision Seminar

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 12, 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

Chapter 2

Theoretical Foundations

2.1 Vision Transformer

Vision Transformers (ViTs), as introduced by Dosovitskiy et al. in 2020 [1], build on the work of Vaswani et al. [2], adapting the transformer architecture for image classification. ViTs leverage the transformer's strengths in handling sequential data by splitting an image into a sequence of 16×16 pixel patches and processing them through the transformer architecture just like tokens as explained below.

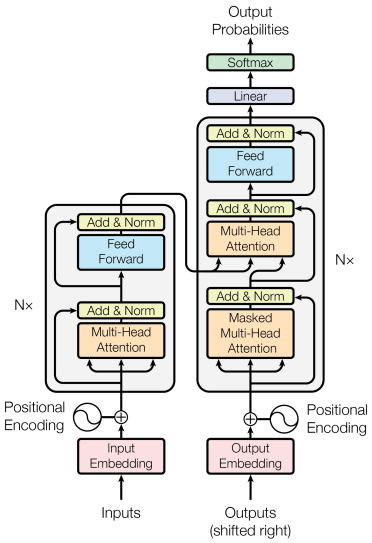


Figure 2.1: The transformer architecture (image from [2])

The original transformer architecture, designed for machine translation, consists of an encoder-decoder structure that processes input sequences through *embedding layers*, *positional encoding*, *attention mechanisms*, and *multi-layer perceptrons*. 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*. These *token vectors* form 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) [3] or Long Short-Term Memory (LSTM) networks [4].

Encoder: Next, the *Embedding Matrix* 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 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 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 [5]**. 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 $W^O \in \mathrm{R}^{hd_k \times d_{\text{model}}}$.

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

Following the multi-headed attention mechanism, the resulting

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. [6] have shown that the decoder is not inherently needed for tasks like language translation, or text generation allowing for a simplified *Encoder Transformer* architecture.

This methodology is largely copied by the ViT architecture, though slightly adapted as shown in Table 2.1.

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 [7]. Since the original ViT paper, many more use cases beyond image classification have emerged for Transformers in the computer vision field, including anomaly detection, object/instance detection and segmentation, image compression and upscaling, dense/depth estimation [8], and video image generation [9]. 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 Dino and DinoV2

As mentioned in the privious Section 2.1, ViTs are work very well for tasks like image classification and have shown superior performance over state-of-the-art convolutional neural networks (CNNs) in several image classification tasks. So far ViTs were mostly trained in a supervised

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 Embedding	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 fully connected feed-forward network.	Same as in Transformers.
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 fully connected feed-forward network.	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.	Same as Transformer.
Class Token	Not applicable.	Introduces a special class token appended to the sequence of patch embeddings, used for classification tasks by aggregating information from all patches.

Table 2.1: Comparison of Components in Transformers and Vision Transformers

manner and needed a lot of labeled data to achieve state-of-the-art performance. As labeled training data is expensive and time consuming to generate, it is very desirable to train models in an unsupervised manner. Just a year after the introduction of the Vision Transformer, Caron et al. [10] introduced an approach to achieve exactly this called DINO (DIstilled NOt labeled), which is a framework for training in theory any network capable of ingesting image data using self-distillation without labels. However the paper in particular emphasizes the synergy of Dino with Vision Transformers as a good compromise between performance and computational cost. Dino is a usefull framework for tasks such as segmentation, depth estimation, and object detection, as it provides a good initialization for supervised fine-tuning.

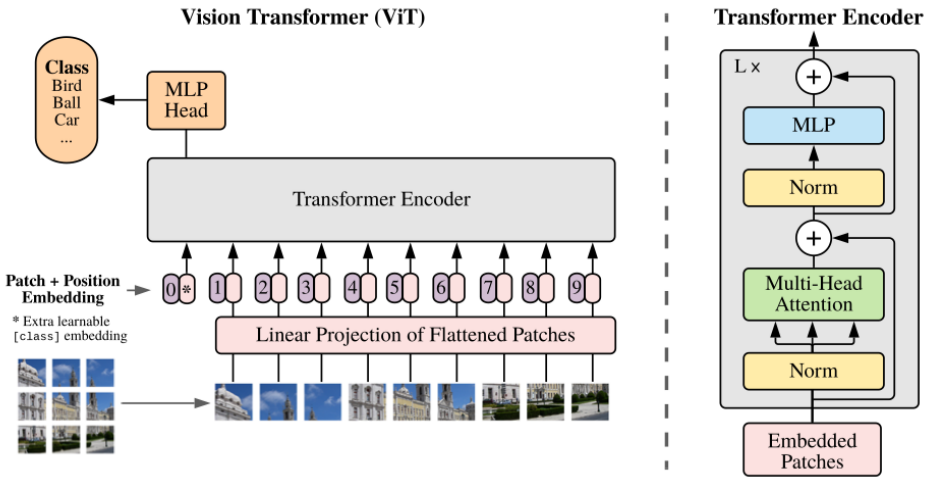


Figure 2.2: The Vision Transformer architecture (copied from [1]).

As explained in [11] destillation in machine learning is a method facilitating the transfer of knowledge from one model to another using a teacher-student model approach. Often the teacher is a larger complex pretrained model which is used to train a much smaller stundent model. However, in the case of Dino the teacher is a non-pretrained model being trained in parallel with the student model. This process is called self-distillation, as no pretrained model is used to inject prexisting knowledge into the student model.

In a teacher stundent architecture the student tries to predict the output of the teacher model. In Dino the students weights get updated using the classical gradient decend approach, while the teacher model is updated using a exponential moving average of the student model weights. Caron et al. figured that in order to prevent a trivial solution where both teacher and student would start outputing the same output no regard to the input, it is enough to center the output of the teacher model to a mean of 0 and to apply a sharpened softmax function to the output by applying a temperatur parameter τ called temperatur to the softmax function as shown in equation 2.2. It makes very large values even larger and very small values even smaller.

Architecture

Dino is trained using a classical student teacher network approach, though the teacher is not a pretrained model but a model trained in parallel to the student model. The Tracher model is largely the same as the student model, but the outputs follow a normalisation applying centering subtracting the mean and running the resulting outputs thorugh a shapened softmax function as shown in equation 2.2.

$$q_i = \frac{f_i}{|f_i|_2} \cdot \tau \tag{2.2}$$

- Attention Map right after training puts much attention on the main object of the image and less on the background. This already is a good start for segmentation - Dino is trained unsupervised and places, when trained, images in to a latent vector, where similar images are close to each other, similar to how similar words are embedded close to each other when training a embedding on them. This could already be used for a k-nearest neighbor approach to classify images.

2.2.1 Depth Estimation

2.2.2 Segmentation

2.3 Model Fine-Tuning

Foundation Models (FMs), as defined in [12], 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 [13] or the Florence CV Foundation Model by Yuan et al. [14]. 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 2.3 with W_0 being the pretrained weights and δW the change in weights during fine-tuning.

$$W' = W_0 + \delta W \tag{2.3}$$

This However still is relativley computationally intesive as the backpropagation has to be done for all parameters. **DISCUSS IN GENERALL HOW PRETRAINING WORKS???**

A more efficient way of fine-tuning weight matrices is Low Rank Adaption (LoRA) as suggested by Hu et al. [15] in the context of transformer fine-tuning. Hut et al. assume based on the the work of Aghajanyan et al. [16], 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 $r << d$ and $r << k$ as shown in equation 2.4. 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.

$$W = W_0 + BA \tag{2.4}$$

Another methodology of finetuning Neural Networks (NN) is to

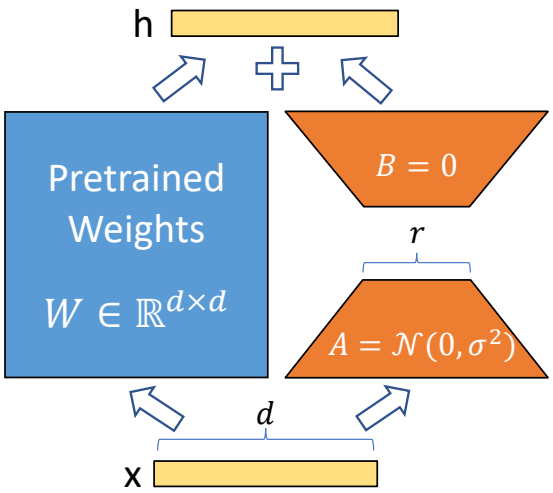


Figure 2.3: LoRa architecture with the docomposed $\delta W_n = AB$ and their initialization method (graphic lifted from [15])

Chapter 3
Unearthing Surgical-Dino

3.1 Improvements of Surgical-Dino

Chapter 4
Discussion

Chapter 5
Conclusion

Appendix A
Code Tools

Bibliography

[1] 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.

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

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] M. Phuong and C. Lampert, "Towards understanding knowledge distillation," in *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, Eds., ser. Proceedings of Machine Learning Research, vol. 97, PMLR, Jul. 2019, pp. 5142–5151. [Online]. Available: https://proceedings.mlr.press/v97/phuong19a.html.

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

[13] 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.

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

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

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