# Unsupervised Semantic Segmentation of Cracks in Concrete Specimens

Amir Sadrnia[1], Mohamed S. H. Alabassy[1] and Andrea Osburg[1]

[1]Bauhaus-Universität Weimar, F. A. Finger Institut für Baustoffkunde, Professur Bauchemie und Polymere Werkstoffe

E-mail(s): mohamed.said.helmy.alabassy@uni-weimar.de

**Abstract:** Reliable and automated detection of cracks is increasingly used for quantitatively assessing the condition of internal damage to concrete subject to cyclic freezing and thawing. Numerous methods already exist to detect cracks, but semantic segmentation has demonstrated superior results in comparison to other conventional methods utilizing heuristic image processing techniques. However, training such networks requires large human-annotated datasets. To eliminate the need for labor-intensive pixel-annotation of images, unsupervised semantic segmentation offers an alternative solution. In this research, two state-of-the-art frameworks, Self-supervised Transformer with Energy-based Graph Optimization (STEGO) and Eigen Aggregation Learning (EAGLE), were chosen to be trained on the $\mu$CT and DeepCrack datasets due to their success on benchmarks. The quantitative results of STEGO trained on $\mu$CT are promising, but the qualitative results show that it failed to detect them, whereas EAGLE with nearly identical metrics could partially detect cracks, when class imbalances are not addressed. Based on the qualitative results on DeepCrack, both models detected cracks in this dataset, however, EAGLE performed worse than STEGO in metrics.

*Keywords:* Crack segmentation, Unsupervised learning segmentation, Contrastive learning, Vision transformer

DOI: 10.18154/RWTH-CONV-254881. Published in the conference proceedings of the 36. Forum Bauinformatik 2025, Aachen, Germany,

## 1 Introduction

Cracks may occur when water gets absorbed into the voids of a heterogeneous multi-phase composite porous material, like concrete, then freezes causing pressure forces inside, as the volume of freezing water expands exerting expansive force beyond its tensile strength. To efficiently assess freezing durability and serviceability of concrete, a significant number of approaches, varying from traditional and manual techniques to image-based methods, were employed to achieve crack detection.

Although previous works, particularly in image-based methods, have achieved significant success in crack identification and detection by development of Deep Learning (DL) methods, automated crack detection is receiving close review and of current interest. Deep learning algorithms can automatically learn meaningful features from the images and are capable of handling large-scale data and complex scenes, where such approaches ranging from supervised semantic segmentation to unsupervised semantic segmentation were used to tackle the automated crack detection tasks.

Supervised semantic segmentation approaches are reliable methods for crack detection tasks, however, these methods require manually labelled data in order to train the deep learning models. Unsupervised semantic segmentation models, on the other hand, can segment meaningful features without the need for annotated masks. These approaches include reduced reliance on extensive labeled datasets and human expertise, adaptability to diverse environments, the potential for real-time applications in infrastructure maintenance and enables scalable and cost-effective crack detection, especially for large infrastructures.

In this research the potential of unsupervised techniques to detect cracks in concrete specimens is explored. State-of-the-art models Self-supervised Transformer with Energy-based Graph Optimization (STEGO) [1] and Eigen Aggregation Learning (EAGLE) [2] were chosen for their significant performances on benchmarks [3]. These models have demonstrated promising results in various applications showcasing their ability to handle complex data.

# 2 Methodology

## 2.1 Dataset

One of the datasets used for this study was obtained at Bauhaus-Universität Weimar, Chair of Construction Chemistry and Polymer Materials. The dataset consists of $\mu$CT-scans of a selection of cylindrical concrete specimens, which were exposed to cyclic freezing and thawing according to the standard German procedures for the Capillary Suction, Internal Damage and Freeze Thaw (CIF) test, then scanned using the “nanotom m research | edition” micro and nano computed tomography system. The system, housed in a compact and radiation-shielded cabinet, is equipped with a high-power nanofocus X-ray tube and an internally cooled detector for extremely high magnification and resolution.

A total number of 3115 grayscale slices of 3D image stack and their corresponding masks were generated. Each mask consists of two classes for segmentation, crack and non-crack. Approximately 2% of the dataset belongs to cracks, whereas 98% are assigned as non-crack pixels. The segmentation scheme was adopted to enable detailed analysis of the internal structures and damage processes within the concrete samples.

The second dataset used in this research, DeepCrack [4], consists of 537 RGB images of concrete surfaces showing a varying levels of crack severity. The images include differences in exposure, brightness, shadow, texture and crack characteristics, and resemble to a large extent the morphological features cracks captured in computed tomographic scans. It reflects the real-world conditions, where cracks may occur in various environments. The images also vary in resolution and quality to ensure that the model encounters a wide range of data during training. This dataset also contains two classes, crack and non-crack, and almost 3% of the the images are labeled as cracks and the rest are non-crack regions.

## 2.2 Preprocessing

Preprocessing the dataset involves manipulating, cleaning, filtering, and encoding the data in order to feed the input to the network, and ensuring that it is free of errors, such as missing or NaN values. The

$\mu$CT dataset was originally grayscale so they were converted to RGB values to ensure the compatibility with Self-distillation with no labels (DINO) [5] backbones. Furthermore, all images were set to a fixed resolution of 224×224 pixels for a balanced memory usage and computational efficiency as well as ensuring the compatibility with DINO backbones. Moreover, for consistent intensity ranges and the reduction of the effect of lighting variations can be achieved through normalization. Normalization takes the mean and standard deviation of the pixel values and normalize them to a fixed range of [0,1] or [-1,1].

Annotation or consistent label mapping is essential for aligning with the model's predictions for meaningful evaluation. In this research, the masks of the datasets are not used for training, however, they are important for an accurate evaluation, therefore, label mapping is a vital preprocessing technique. CT reconstruction artefacts filled or inconsistent images are considered outliers and may cause errors during training, the presence of noisy data could contribute to over-fitting and severely impact the model's performance in learning meaningful features. The $\mu$CT dataset was reviewed and the noisy data were manually removed. After this step, the total number images were downsampled from 3154 Pixels to 2512. A reduction of almost 20% made the dataset cleaner and prepared it for the next preprocessing steps.

The datasets in this study both consist of images and annotation masks. It should be noted that the labels of the datasets are never used during the training process; however they are solely used for evaluating the performance of the model [1]. The $\mu$CT dataset was divided into two parts; training and validation datasets. A 70:30 data-split ratio (i.e., 70% of the dataset is used for training and the other 30% for validation purposes) was taken into account, while a ratio of 80:20 was considered for the DeepCrack dataset, as the whole dataset contains significantly fewer images in compared to the $\mu$CT dataset. For $\mu$CT, 1759 images and their corresponding labels belong to training dataset, whereas 753 images and their corresponding annotation masks belong to validation dataset. For DeepCrack, the numbers are 300 and 237 for the training and validation datasets respectively. Data augmentation is used to artificially generate more varying images from the existing data. Zooming-in, jittering and cropping out images can simulate different data acquisition conditions and geometric perspectives and reduce bias. In STEGO and EAGLE data augmentation is carried out by five-crop technique, which divides an image into 5 different crops, top-left corner, top-right corner, bottom-left corner, bottom-right corner, center crop with desired resolution.

## 2.3 Models

STEGO is a Self-Supervised Learning (SSL) model, which uses Vision Transformers (ViT) instead of Convolutional Neural Network (CNN) to detect patterns in images without relying on manually annotated data. EAGLE takes a different approach by clustering similar features in images by using eigenvector analysis. Both models have demonstrated solid performance on benchmark segmentation tasks, showcasing their potential for real-world application, such as crack detection. However, the repository of EAGLE was developed based on that of STEGO.

Both models leverage contrastive learning. STEGO learns how to semantically cluster a group of similar pieces, whereas EAGLE aggregates features by identifying shared patterns across different images, which can be compared with recognizing dominant eigenvectors in a dataset. For the purpose of this study, a refactored version of STEGO is used for training on the respective datasets. Frey et al. [6] refactored the STEGO project to achieve a clean and more organized code. Their implementation is more efficient in usage of Random Access Memory (RAM) and is therefore more time efficient for training purposes of the model locally. Training of the EAGLE and STEGO models on DeepCrack and $\mu$CT datasets was conducted on a laptop with NVIDIA GeForce RTX 4080 (12 GB) Graphics Processing Unit (GPU), Intel Core i9-13980HX Computing Processing Unit (CPU), and 64 GB DDR5-5600 (2x 32 GB Dual-Channel) of RAM.

### 2.4 Evaluation Methods

To evaluate the results of a semantic segmentation process, numerous metrics can be employed. In STEGO and EAGLE, the performance of the trained models are evaluated through Accuracy and mIoU. Accuracy (Acc) is calculated by the number of correct predictions over the whole scene. It is calculated using the formula $Acc = \frac{TP+TN}{TP+TN+FP+FN}$, while Intersection over Union in terms of semantic segmentation calculates the overlap of the ground truth and the prediction (True Positive) over the union of the sample sets. The formula for Mean Intersection over Union (mIoU) is $IoU = \frac{TP}{TP+FP+FN}$, where TP, TN, FP and FN are the number of True Positive, True Negative, False Positive and False Negative values respectively.

Both STEGO and EAGLE are evaluated using two primary methods, namely the Linear Probe Evaluation and Clustering Evaluation. The linear probe assesses the quality of the learned features via a supervised task by training a linear classifier on top of a frozen feature extractor using Cross-Entropy loss, with ground-truth labels used only for evaluation [1]. In contrast, clustering evaluation measures the alignment between unsupervised predictions and ground-truth labels using the Hungarian algorithm to ensure label permutation invariance, to evaluate the model's ability to group similar regions meaningfully [1].

## 3 Experiments and Results

The STEGO model with the architecture *"DINO ViT_base_8"* was trained on the $\mu$CT and DeepCrack datasets with 500 number of steps, batch size 32 and 0 and 5 extra clusters, additional learned representations help to improve segmentation performance, for $\mu$CT and DeepCrack respectively. After hyperparameter tuning, it was determined that the architecture, batch size and number of extra clusters are the key parameters influencing STEGO's training. The base version of DINO with 8 patches showed the best performance related to mIoU and accuracy among the tested backbones. Increasing the batch size to 32 could significantly improve the model's performance. The value of extra clusters can influence the quantitative and qualitative results of the training. Training STEGO on the $\mu$CT dataset with zero extra clusters yielded the best performance for mIoU.

EAGLE with the architecture *"DINO ViT_base_8"* was also trained on $\mu$CT and DeepCrack using a batch size of 32, with 600 and 500 maximum training steps, and 5 and 0 extra clusters, respectively. Similar to the findings regarding the sensitivity and importance of hyperparameters in STEGO, EAGLE exhibited comparable behavior with respect to its own hyperparameter settings. It is noteworthy that the parameters $b_{\text{knn}}$, $b_{\text{rand}}$, $b_{\text{self}}$, $\gamma_{\text{knn}}$, $\gamma_{\text{rand}}$, and $\gamma_{\text{self}}$, negative inter-cluster alignment, positive inter-cluster relationship, positive intra-cluster alignments, negative inter-shift, positive inter-shift and positive intra-shift respectively, demonstrated significant importance in the hyperparameter optimization of STEGO. Among these, the parameters $b_{\text{knn}}$, $b_{\text{rand}}$, and $b_{\text{self}}$ were particularly crucial for optimizing EAGLE's performance. The quantitative and qualitative results of training STEGO and EAGLE on $\mu$CT and DeepCrack are shown in Table 1, Figure 1, Figure 2 and Figure 3 respectively.

Table 1: Quantitaive results of Training STEGO and EAGLE on $\mu$CT and DeepCrack datasets respectively.

| Model | Dataset | Unsupervised | | Linear Probe | |
|---|---|---|---|---|---|
| | | Accuracy (%) | mIoU (%) | Accuracy (%) | mIoU (%) |
| STEGO | $\mu$CT | 92.11 | 46.07 | 97.40 | 48.70 |
| STEGO | cropped-$\mu$CT | 97.81 | 49.89 | 99.85 | 50.02 |
| STEGO | DeepCrack | 98.96 | 49.53 | 99.68 | 49.90 |
| EAGLE | $\mu$CT | 99.96 | 45.33 | 99.23 | 48.89 |
| EAGLE | cropped-$\mu$CT | 99.98 | 45.76 | 99.97 | 50.11 |
| EAGLE | DeepCrack | 68.49 | 34.25 | 99.83 | 49.91 |

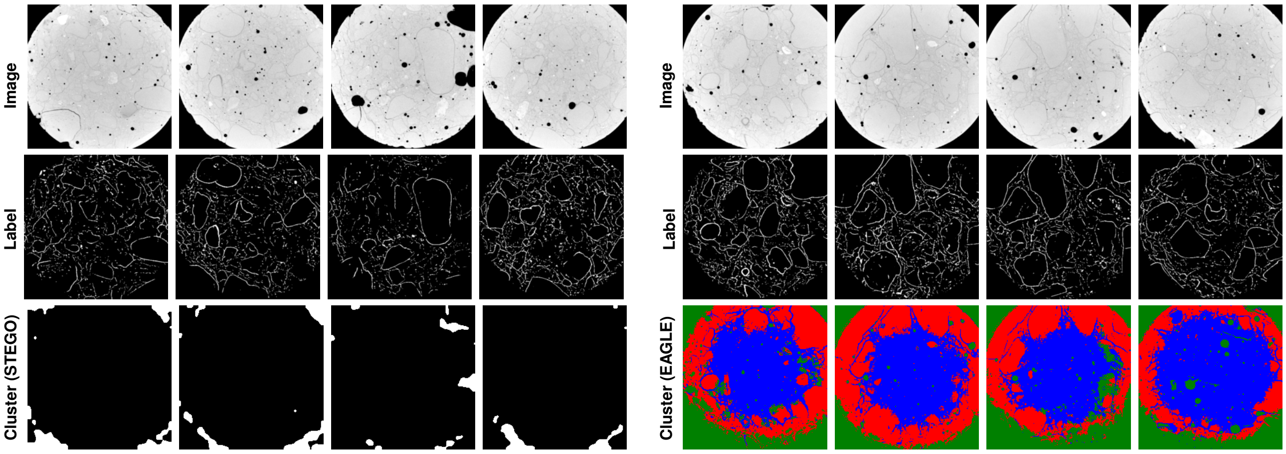

Figure 1: Qualitative results of the models on $\mu$CT dataset. On the right STEGO and on the left EAGLE. The blue color for results of EAGLE on Cluster depicts the 'Crack' class, whereas red demonstrates the anything on the concrete except for cracks. Green does not indicate a third class but it was used to distinguish the background from the sample.

# 4 Discussion

## 4.1 Comparison of the Results

The evaluation of the experiments shows that STEGO could provide a reasonable quantitative results for both datasets, but the qualitative results show that STEGO failed to detect any cracks in the $\mu$CT dataset, whereas it detected cracks in the DeepCrack dataset. EAGLE produced nearly identical quantitative results for the $\mu$CT dataset but detected partially some cracks in the images. EAGLE showed a weaker quantitative performance for DeepCrack, however its qualitative results were promising. It was found that the models' metrics alone do not fully reflect the segmentation quality attained due to the disproportionality of class weights, but through visualization of the results and how the model segments the respective classes, the performance of the approach can be interpreted more accurately.

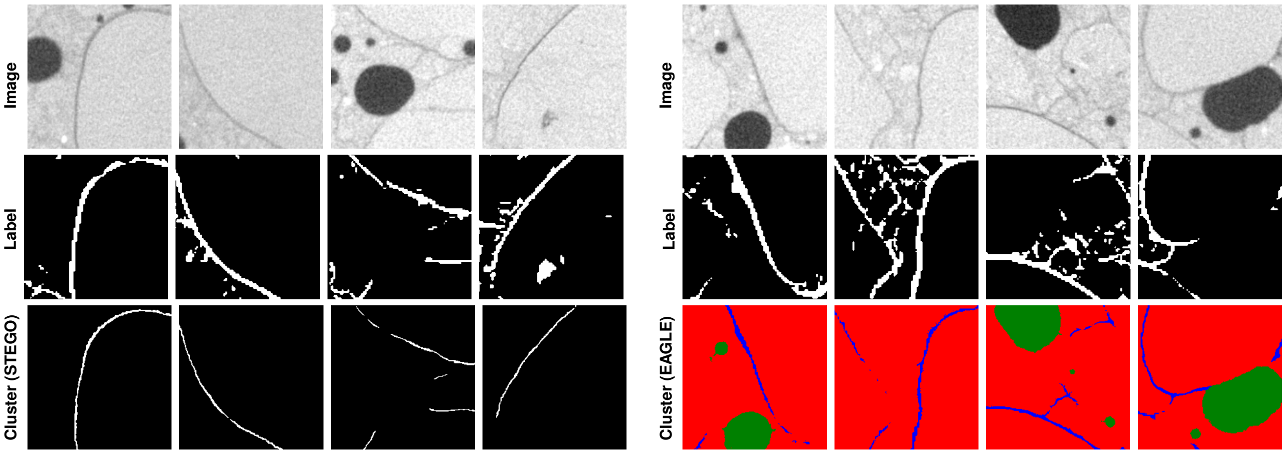

Figure 2: Qualitative results of the models on cropped $\mu$CT dataset. On the right STEGO and on the left EAGLE. The blue color for results of EAGLE on Cluster depicts the 'Crack' class.

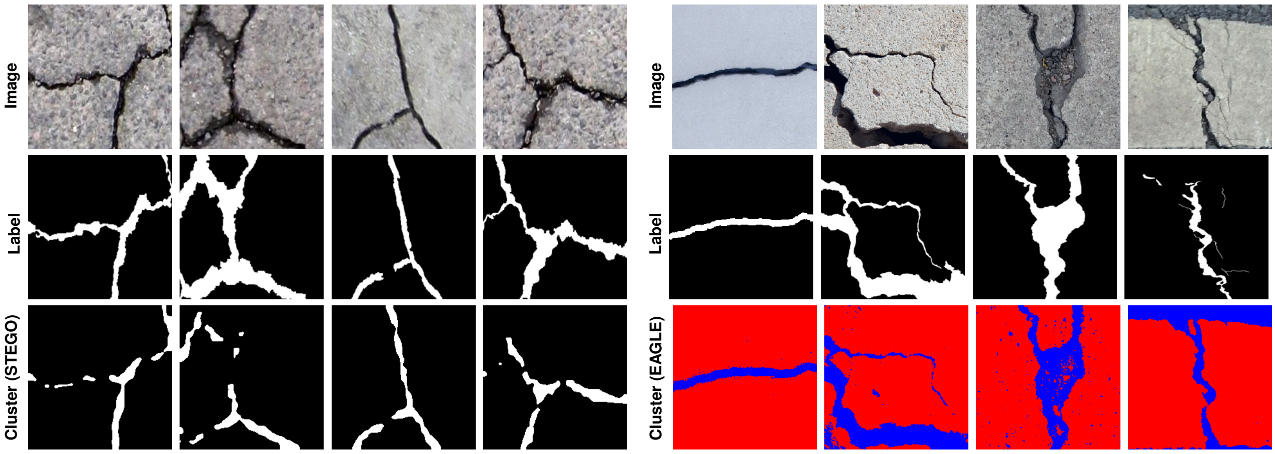

Figure 3: Qualitative results of the models STEGO and EAGLE on DeepCrack dataset.

## 4.2 Challenges and Countermeasures

### 4.2.1 Data-Level Approaches

Despite successful application of unsupervised semantic segmentation using STEGO and EAGLE, several challenges were encountered that affected the performance of the models. The greatest challenge is the class imbalance. The class imbalance encourages the model to become biased and leads to overfitting. The second challenge mostly regards the $\mu$CT dataset and its one-channel (grayscale) structure. Most of the existing models are designated to be trained on RGB datasets. However, several strategies were implemented to attack the aforementioned problems.

The FiveCrop augmentation was applied to the DeepCrack dataset to increase diversity by cropping each image into five parts. However, for $\mu$CT images, which contain cracks, concrete, and background but still follow a binary classification (crack vs. non-crack) it was avoided to prevent segmentation challenges. Instead, standard cropping was used to reduce background interference and enhance crack visibility. This improved both models' mIoU by approximately 3% each.

Manually removing the images with a lesser presence of cracks was employed for DeepCrack. This approach would allow the dataset to magnify the crack class. This approach could improve the mIoU STEGO and EAGLE by approximately 1% and 0.9%. It should be noted that the conventional augmentation techniques, such as rotation, translation and jittering was applied during the training.

Grayscale structure of $\mu$CT was also problematic, as most of the semantic segmentation models expect RGB images. However, duplicating grayscale channels to generate fake RGB images was taken into account. This approach could only make the dataset appropriate for being fed to the models. Furthermore, transfer learning could not be effective because pre-trained models are trained on RGB datasets. Moreover, traditional color-based augmentations such as hue and saturation could not be applied to grayscale images.

### 4.2.2 Algorithm-Level Approaches

Switching between DINO architectures was explored to improve model performance. While DINOV2 is theoretically more advanced, it did not consistently achieve high metrics in this study. The best results were scored by *"ViT_base"* with 8 patches. Adding extra clusters helped STEGO on DeepCrack (+4% mIoU) and EAGLE on $\mu$CT (+2% mIoU), but the same strategy failed when applied conversely. This highlights dataset-dependent model behavior. Koenig et al. [7] suggest that a lower-dimension embedding space, $D_{\text{STEGO}}$, focuses on global pattern and generates coarse segmentations, whereas higher values keep more granular details but causes noise and therefore a less effective segmentation. The target value 90 for *"dim"* showed the best performance for all four scenarios.

It is noteworthy to mention that the difference between STEGO and EAGLE in detecting cracks, even partially, in $\mu$CT dataset lies in their approaches to capture meaningful data. EAGLE employs EiCue (Eigen Cues) as a spectral clustering technique. EiCue helps EAGLE segment sharp, fine structures rather than just grouping similar features. STEGO's feature correspondence approach works better to distinguish larger semantic regions but may fail dealing with thin cracks. For reference, the UP-CrackNet method [8] achieved an IoU of 58.738 trained on DeepCrack, and 61.073 trained on Crack500, demonstrating strong performance among unsupervised approaches. These IoU values are higher than those obtained in our experiments, although the qualitative results from UP-CrackNet are less clear in capturing fine crack details.

## 5 Conclusion

In this research, the unsupervised models STEGO and EAGLE were trained on $\mu$CT and DeepCrack datasets for binary crack segmentation, showing that while STEGO achieved better quantitative results and EAGLE yielded more promising qualitative outputs, visual inspection revealed that segmentation quality, aside from metrics values, better reflects model performance. The difference between STEGO and EAGLE performance lies in their techniques to capture cracks. EAGLE utilises an EiCue (i.e., Eigen Cue) technique, which avails the model to extract sharper and finer details, whereas STEGO relies on feature correspondence approach, which is more suitable for distinguishing larger semantic regions. Furthermore, the qualitative differences in model performance, particularly the failure of STEGO to detect cracks in $\mu$CT, indicates that class imbalance play the central role. Numerous methods such as data-level approaches as well as algorithm level approaches were employed. Minor improvements were observed after applying the techniques discussed earlier, but these techniques could not fully solve the problem.

Future research could explore alternative methods to overcome class imbalance. Moreover, more robust, less noisy and class-balanced datasets might help improve crack detection. It should be noted that unsupervised approaches are still in their early stages and the current results motivate further improvements in the architectures and algorithms. It is still too soon to draw conclusions, whether such approaches are appropriate or the definitive solution for crack detection. Nevertheless, the completion of this research represents a significant step toward leveraging existing models for crack detection. In conclusion, this research demonstrates the potential of unsupervised techniques for crack detection, opening new avenues for further exploration in this field.

## Data Availability and Acknowledgements

Trained checkpoint files of the trained models available on GitHub. A Master's Thesis pertaining further detailed results of this paper is available via this URL. This research is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - Project number 432767097. $\mu$CT Dataset used is part of the aforementioned project.

## References

[1] M. Hamilton, Z. Zhang, B. Hariharan, N. Snavely, and W. Freeman, *Unsupervised semantic segmentation by distilling feature correspondences*, arXiv preprint, arXiv:2203.08414, 2022.

[2] C. Kim, W. Han, D. Ju, and S. Hwang, *Eagle: Eigen aggregation learning for object-centric unsupervised semantic segmentation*, arXiv preprint, arXiv:2403.01482, 2024.

[3] Papers with Code, *Unsupervised semantic segmentation*, Available at: https://paperswithcode.com/task/unsupervised-semantic-segmentation.

[4] Y. Liu, J. Yao, X. Lu, R. Xie, and L. Li, "Deepcrack: A deep hierarchical feature learning architecture for crack segmentation", *Neurocomputing*, vol. 338, pp. 139–153, 2019. DOI: 10.1016/j.neucom.2019.01.036

[5] Yannic Kilcher, *Dino: Emerging properties in self-supervised vision transformers (facebook ai research explained)*, 2021. [Online]. Available: https://youtu.be/h3ij3F3cPIk

[6] J. Frey, M. Mattamala, N. Chebrolu, C. Cadena, M. Fallon, and M. Hutter, *Fast traversability estimation for wild visual navigation*, 2023. arXiv: 2305.08510 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2305.08510

[7] A. Koenig, M. Schambach, and J. Otterbach, *Uncovering the inner workings of stego for safe unsupervised semantic segmentation*, 2023. arXiv: 2304.07314 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2304.07314

[8] N. Ma, R. Fan, and L. Xie, *Up-cracknet: Unsupervised pixel-wise road crack detection via adversarial image restoration*, 2024. arXiv: 2401.15647 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2401.15647