Local Affine Frame-based Vision Transformer

Anna Kawai, Yanze Liu, Yiting Chen, Yu Ge

I. Task

We aim to develop a novel approach to tokenize images for vision transformers (ViT) [1] by guiding image patching with local feature detectors for the task of image classification.

In our case, local features are derived from key points, which is the output of KeyNet [2]. KeyNet includes hand-crafted and convolutional neural network (CNN) filters to detect key points, denoted as local affine frames (LAFs) [2]. The patches/tokens are generated around LAFs, with invariance of scale and orientation. Once flattened, these patches and LAFs are the inputs of the encoder of the modified ViT. We expect this guided patching approach will improve classification accuracy.

II. Related Work

In natural language processing, transformers have become the dominant architecture due to their exceptional ability to model long-range dependencies and process sequential data efficiently [3]. This success has inspired researchers to adapt transformers for computer vision tasks. Unlike traditional CNNs, which rely on local connectivity and spatial hierarchies, ViTs process images as sequences of patches, leveraging self-attention mechanisms to capture global dependencies. ViTs have demonstrated state-of-the-art performance in various vision tasks, making them a strong competitor to CNNs.

Despite their successes, transformers lack certain inductive biases inherent to CNNs, such as translation equivariance and local feature aggregation [4]. These biases often enable CNNs to generalize effectively with fewer training samples and better exploit the spatial structure of images. But even in CNNs, learning from a large amount of data is the only way to get robust responses for features in different scales and orientations. Addressing this limitation, researchers have proposed hybrid approaches that integrate the strengths of both architectures. For instance, the Pyramid Vision Transformer (PVT) [5] introduces a CNN-like pyramid structure to handle multi-scale features. Extracting local and global features using CNNs and ViTs has also been explored to detect anomalies in images [6] . CNNs achieve superior performance in local feature extraction, but have difficulty extracting global features [6]. Conversely, ViT only approaches can learn the distribution of global features, but ignore the local context [6]. Motivated by the challenges and successes of these approaches, Wahid et al. developed a combined model of CNNs and ViTs that outperformed traditional models in the field of anomaly detection. Their model, NN2ViT, uses Single Shot Dector (SSD) and the Segment Anything Model (SAM) for capturing local feature detection and global feature segmentation, respectively [6]. ViTALNet is another example of a hybrid transformer for anomaly detection in images, designed to overcome the challenges of learning local and global features [7]. This model is constituted of two modules, a ViT to encode the effective image context, and a hybrid CNN-ViT pyramidal feature reconstruction architecture to estimate local and global anomaly regions precisely. VitALNet outperformed competing methods in best mean pixel AUROC by 0.2-40.7%, evidencing the potential of combining these tools in computer vision tasks [7].

For more induced bias in transformer architecture, tokenization strategies have also evolved to adapt transformers to vision tasks. Movable tokenization, as introduced in works like Deformable Patch-based Transformer (DPT) [8], replaces square patches with rectangular regions. These regions are learned dynamically during training, allowing the patches to focus on the area with more information and have different scales. Another variation of tokenization is superpixel-based tokenization, [9], which partitions an image into regions of perceptually similar pixels, rather than fixed square patches. Also inspired by the potential of combining self-attention and CNNs, Liu et al. developed SWIN transformers. Their work uses an alternative patching approach to optimize self-attention and the detection of hierarchical relationships between visual entities. They implement shifted-windows to create hierarchical feature maps that merge patches in deeper layers, allowing cross-window connections and simplifying the standard quadratic complexity of ViTs to a linear complexity to the input image size [10]. Their method has achieved notable performance in image classification (87.3 top-1 accuracy on ImageNet-1K), and has offered a generalizable framework compatible with object detection and semantic segmentation [10]. Their work emphasizes the need for improved patching methods to capture meaningful relationships between visual entities to optimize the performance of ViT.

Previous work evidences the need for a computer vision model capable of local and global feature extraction, as well as optimized image tokenization. Traditional tokenization methods disrupt the continuity of image data, limiting their potential in preserving image features during tokenization. Drawing inspiration from scale-invariant feature transform (SIFT) detectors [11], we propose a vision transformer that utilizes scale- and rotation-invariant tokens. Later, we find that KeyNet is more efficient for our image processing pipeline, which aligns with previous work on coupling CNNs and ViTs. By introducing inductive bias during token segmentation, our approach maximizes information retention

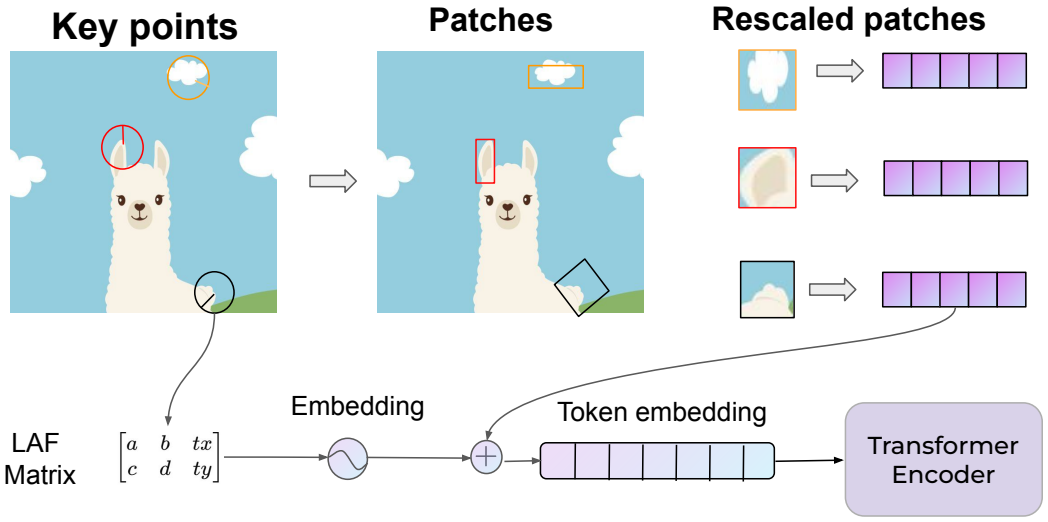

Fig. 1. The illustration for the LAF-based Transformer.

without requiring modifications to the underlying transformer architecture.

III. Approach

In vanilla ViT, the whole image was cut into multiple 16×16 patches and the patches contain all the information of the image. We note that not all the pixels are equally important for the images. For instance, on the image classification task, the object is more important than the background, making the patches containing the background redundant. However, ViT doesn't utilize such a difference between parts in an image. This inspires us to refine the patches inputed to the ViT, i.e. we only input the "important" patches. In this project, the "important" patches will be constructed based on the key point detected by existing algorithms/models, such as SIFT and KeyNet.

A. LAF Generation

We will generate scale-invariant key points from the KeyNet algorithm, which has a uniform output format,the LAF matrix. Scale, position, and orientation information can be retrieved from LAF matrix. Hence, each key point and the related information can be solely determined by the LAF matrix. The key point extraction algorithms are provided by an open-sourced project Kornia [12]. Kornia includes algorithms like SIFT, KeyNet, and so on. Notice that the computation of LAF can be done offline, we pre-process the whole training set as follows:

1) Generate key points with the feature method. Each point corresponds to a 2x3 matrix representing local affine frames (LAF), which includes position and (local) linear transform.
2) Save the computed LAFs for the whole datasets, which will be used in the embedding part in Transformer.

The pre-processing could help saving a large amount of time since we don't have to recompute the LAF in each epoch. The additional cost is having extra memory to storage the LAF. Such cost is pretty cheap since only a tensor of shape $(N, 2, 3)$ needed to be saved per image, no matter what resolution of the image is, where N is the number of keypoints (equivalently, the number of patches we need). In our experiment, N is set as 64 or 256.

LAFs play a central role in the construction of patches and embedding part in the proposed Transformer. Its first usage is to guide the patch extraction from the images. Then, the LAFs are also applied to a linear layer to become the embedding, which replaces the positional embedding module in the standard ViT.

B. Patching Embedding

Given a LAF matrix (i.e. 2×3 matrix), we construct a patch in the following way:

Step 1. Denote the LAF as $\begin{bmatrix} a & b & t_x \\ c & d & t_y \end{bmatrix}$ and $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$. Let set E be the image of the unit 2-D disk under linear transform $z \mapsto Az$, i.e. $E := \left\{ \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} : x^2 + y^2 \leq 1 \right\}$.

Step 2. Shifting E to position $\begin{bmatrix} t_x \\ t_y \end{bmatrix}$, denoted as E'. Therefore, $E' := \left\{ \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} : \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} \in E \right\}$ is also a rectangle region.

Step 3. Notice that E' is always a solid ellipsoid/oval centered at $\begin{bmatrix} t_x \\ t_y \end{bmatrix}$, we let R be the minimal rectangle covering E' and R is centered at $\begin{bmatrix} t_x \\ t_y \end{bmatrix}$.

Step 4. Getting the patch corresponding to the rectangle region R from the image and resizing it to the given patch size.

To determine the minimal rectangle region at **Step 3**, it suffices to determine the two edges of the rectangle. Let s_1

Fig. 2. The key points captured by KeyNet and the patches constructed from the LAFs.

and s_2 be the two singular values of A, and the associated right singular vector is $\mathbf{v}_1$ and $\mathbf{v}_2$, i.e. $\|A\mathbf{v}_1\|_2 = s_1\|\mathbf{v}_1\|_2$ and $\|A\mathbf{v}_2\|_2 = s_2\|\mathbf{v}_2\|_2$. Then the first edge of the rectangle is parallel to $\mathbf{v}_1$, of length s_1 and its distance to the point $\begin{bmatrix} t_x \\ t_y \end{bmatrix}$ is $\frac{s_2}{2}$. Similarly, the second edge of the rectangle is parallel to $\mathbf{v}_2$, of length s_2 and its distance to the point $\begin{bmatrix} t_x \\ t_y \end{bmatrix}$ is $\frac{s_1}{2}$.

We handle the images from the whole dataset during every epoch. Here we will take one single image as an example to show our proposed method. In Figure 2, we implemented Keynet to detect 64 the key points and get the associated LAFS. The ellipsoid described as in **Step 2** for each key point is marked with red and the associated patches described as **Step 4** are arranged as in a 8×8 format.

Once the patches are constructed, we will flatten them and conduct the embedding through a Linear Layer, up to appropriate normalization (Pytorch LayerNorm).

C. LAF as an Embedding

Each LAF corresponds to a key point is a 2×3 matrix. LAF will be flattened and embedded through a Linear Layer, up to appropriate normalization (Pytorch LayerNorm).

IV. Datasets

The dataset we used is ImageNet-100, which is an adaptation of ImageNet-1k, a benchmark dataset for object detection and classification models [13]. ImageNet-100 was retrieved from Huggingface, and it contains 100 classes, with a split of 117k training images, 13k validation images, and 5k test images. We tested two different types of resizing, 256x256 and 64x64.

The transformer training requires more data than convolutional neural networks. Generally, training a vision transformer requires a dataset on the order of millions of images, which is satisfied only on large datasets like original ImageNet. Considering the capacity of the project and computing resources, ImageNet-100 is a compromise choice. For some with low resolution like CIFAR-10 and CIFAR-100, the number of patches is too few to demonstrate the true power of transformers.

V. Metrics

The metric used was validation accuracy during training, which is the percentage of correctly labeled images from the validation dataset. The classification accuracy is defined as the ratio of the number of corrected classified image and the total image number.

VI. Experiment Results

A. Experiment Setting

The concrete environment used in SCC to run the experiment is as follows: (i) number of GPU $= 1$; (ii) GPU compute capability $= 6.0$; number of cores $= 8$.

B. Extraction of key points (LAFs)

The following mentioned directory is located in the Github repository. This part is implemented in the modules in `main/lafs/get_patch_and_feature.py` and `main/lafs/add_laf.py`. Besides, there are also an illustration for this process in `main/illustration_for_getting_patches.ipynb`.

The maximum number of key points output by KeyNet is set as 5000. In this setting, KeyNet will return 5000 LAFs associated with key points at most per image and all the LAFs will be returned if the number of key points detected by KeyNet is less than 5000. Let N be the number of patches needed to input to the model. Then if N is greater than the number of LAFs returned by KeyNet, the multiple 2×3 matrix of zeros will be included to ensure that there are always N LAFs for one image. If N is less than the number of LAFs output by KeyNet, we select the LAFs such that their indices form an arithmetic sequence.

Method	Image Resizing	Num of Patches	DA	Patch Method	Training Time	Accuracy
Our method_256	256x256	256	No	KeyNet LAF-based	23h 11min 48s	58.9%
ViT_256	256x256	256	No	Standard ViT Patching	17h 22min 41s	45.6%
ViT_with_DA_256	256x256	256	Yes	Standard ViT Patching	18h 2min 55s	56.46%
Our method_64	128x128	64	No	KeyNet LAF-based	5h 24min 32s	48.6%
ViT_64	128x128	64	No	Standard ViT Patching	4h 16min 39s	33.3%
ViT_with_DA_64	128x128	64	Yes	Standard ViT Patching	4h 6min 40s	37.43%

TABLE I

Comparison of Methods Based on Image Resizing, Data Augmentation, Patch Method, Training Time, and Accuracy. DA: Data Augmentation.

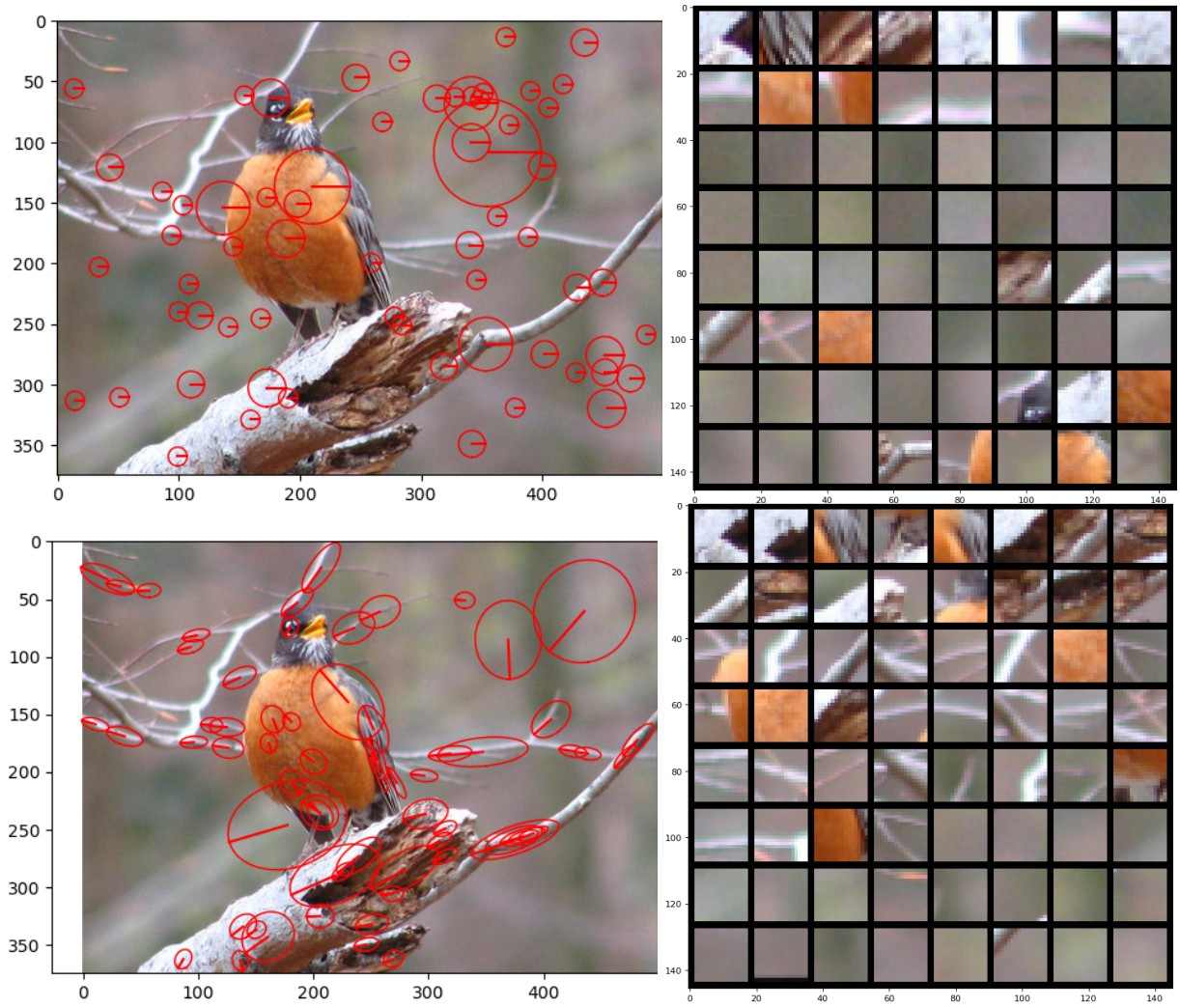

Fig. 3. The key points comparison of SIFT and KeyNet. The first row is the result from SIFT, and the second row is the result from KeyNet.

We also compared the key points generation quality between SIFT and KeyNet. The quality of the patches generated by KeyNet outperforms the one generated by SIFT as observed in fig. 3. One can see that the patches constructed from LAFs output by KeyNet are less related to the background. Thus, the patches used later in Transformer are generated by KeyNet in this work.

C. Vision Transformer

There are two section for the part: model and training, which are `main/model/` and `main/training/`. There are three cases for the comparison reason: the proposed method, the original ViT, and the ViT with data augmentation. The results are shown in Figure 4. Each case contains two different settings for the number of patches.

The data augmentation we applied in this experiment is sequentially applying the two transforms:

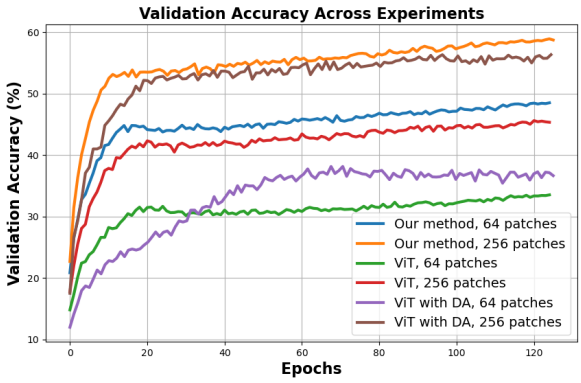

Fig. 4. Validation accuracy comparison between different cases

- torchvision.transforms.RandomHorizontalFlip();
- torchvision.transforms.RandomCrop(IMAGE_SIZE, padding=4).

The first transform applies a random horizontal flip to the input image with a specified probability. By default, this probability is 0.5. The flip transform enhances the variations of the dataset in terms of orientation. The second transform will pad the input image and then randomly selected and cropped a region of size (IMAGE_SIZE $\times$ IMAGE_SIZE) from the padded image. For instance, if the resolution of the input image is 256×256 and IMAGE_SIZE $= 256$, the padding image is of size 264×264 (with 4 pixels padding on each side) and the output of the transform will be a 256×256 sub-image, randomly selected from the padding image. All the cases are run 150 epochs with patch size of 16×16. The following are the experiment results:

- Proposed method with 64 patches: we resized all the images as 128*128, and input 64 patches per images to ViT, data augmentation is not applied, the patches are taken based on the lafs computed by Keynet, accuracy: $48.6\%(0.4864615499973297)$, training time: 5h 24m 32s.
- Proposed method with 256 patches: we resized all the images as 256*256, and input 256 patches per images to ViT, data augmentation is not applied, the patches are taken based on the lafs computed by Keynet, accuracy: $58.9\%(0.5885384678840637)$, training time: 23h 11m 48s
- Original ViT with 64 patches: we resized all the images as 128*128, and input 64 patches per images to ViT, data augmentation is not applied, the patches are taken on the same way as Original ViT. Since the patch sizes are 16*16, so the 64 patches will be input to ViT per images, accuracy: $33.3\%(0.33307692408561707)$, training time: 4h 16m 39s
- Original ViT with 256 patches: we resized all the images as 256*256, and input 256 patches per images to ViT, data augmentation is not applied, the patches are taken on the same way as Original ViT. Since the patch sizes are 16*16, so the 256 patches will be input to ViT per images. accuracy: $45.6\%(0.4563846290111542)$, training time: 17h 22m 41s
- ViT with data augmentation and 64 patches: we resized all the images as 128*128, and input 64 patches per image to ViT, data augmentation is applied, the patches are taken on the same way as Original ViT. Since the patch sizes are 16*16, so the 64 patches will be input to ViT per images. Accuracy: $37.43\%(37.430769205093384)$, training time: 4h 6m 40s
- ViT with data augmentation and 256 patches: we resized all the images as 256*256, and input 256 patches per images to ViT, data augmentation is applied, the patches are taken on the same way as Original ViT. Since the patch sizes are 16*16, so the 256 patches will be input to ViT per images. Accuracy: $56.46\%(55.46153783798218)$, training time: 18h 2m 55s

For reference, we summarize the results in Table I. In conclusion, our proposed method shows the advantage in terms of classification accuracy over validation set compared with the other two methods, standard ViT and the ViT with data augmentation.

Notably, we also explored directly using the orientation of patches and shifting them, and introducing them directly into the ViT. However, we encountered issues with the training time and improvement of the loss function, further validating the versatility and efficiency of coupling KeyNet with the ViT for LaF-guided patching.

VII. Conclusion

In conclusion, our method aims to improve the performance of ViT by modifying the patching and embedding. The LAF matrix, which implicitly contains the information regarding patch scale, orientation, and position, is used for the Transformer embedding. Experiments with different settings are applied to show the advantages over the baselines based on ImageNet 100 dateset. The results of the experiment show the potential of combining CNNs-based technique (KeyNet) and ViTs for image classification tasks. The proposed method outperformed standard ViT and remained superior even when ViT patching was enhanced with data augmentation. This improvement was observed when using the same number of patches and even less.

VIII. Appendix

A. Detailed Roles

Please check Table II.

B. Code Repository

Please check https://github.com/lyz2d/EC523Proj.git

References

[1] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," Jun. 2021.

Team Member	Role
Yanze Liu	Contributed to the development of the LAFs-based Transformer model by modifying and training the model on SCC. Responsibilities also included visualizing training results, writing the report, and updating the GitHub readme repository.
Yiting Chen	Led the development of image pre-processing part and worked on LAFs-based Transformer model. Responsibilities included developing code for LAFs computation, modifying the positional embedding component, and facilitating the integration of other modules.
Yu Ge	Inspired our initial project, researched the available tools to make our model, explored the feasibility of using KeyNet for key point detection, collaborated with Anna in exploring the possibility to directly incorporating orientation features into the ViT, and created the diagrams and graphical abstract of our approach for project deliverables.
Anna Kawai	Uploaded the datasets and other SCC-related tasks, collaborated with Yu in exploring the possibility to directly incorporating orientation features into the ViT, created the presentation structure and slides, and worked on documentation and the report.

TABLE II

Contribution of each member

[2] A. Barroso-Laguna, E. Riba, D. Ponsa, and K. Mikolajczyk, "Key.net: Keypoint detection by handcrafted and learned cnn filters," 2019. [Online]. Available: https://arxiv.org/abs/1904.00889

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," Aug. 2023.

[4] A. Goyal and Y. Bengio, "Inductive biases for deep learning of higher-level cognition," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 478, no. 2266, p. 20210068, Oct. 2022.

[5] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, "Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions," Aug. 2021.

[6] J. A. Wahid, M. Ayoub, M. Xu, X. Jiang, L. Shi, and S. Hussain, "Nn2vit: Neural networks and vision transformers based approach for visual anomaly detection in industrial images," *Neurocomputing*, vol. 615, p. 128845, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231224016163

[7] X. Tao, C. Adak, P.-J. Chun, S. Yan, and H. Liu, "Vitalnet: Anomaly on industrial textured surfaces with hybrid transformer," *IEEE Transactions on Instrumentation and Measurement*, vol. 72, pp. 1–13, 2023.

[8] Z. Chen, Y. Zhu, C. Zhao, G. Hu, W. Zeng, J. Wang, and M. Tang, "Dpt: Deformable patch-based transformer for visual recognition," in *Proceedings of the ACM International Conference on Multimedia*, 2021.

[9] M. Aasan, O. Kolbjørnsen, A. S. Solberg, and A. R. Rivera, "A Spitting Image: Modular Superpixel Tokenization in Vision Transformers," Aug. 2024.

[10] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.

[11] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2. IEEE, 1999, pp. 1150–1157.

[12] E. Riba, D. Mishkin, D. Ponsa, E. Rublee, and G. Bradski, "Kornia: an open source differentiable computer vision library for pytorch," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 3674–3683.

[13] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.