A SEMANTICS-AWARE NORMALIZING FLOW MODEL FOR ANOMALY DETECTION

wei ma[◇], *Shiyong Lan* [◇†*], *Weikang Huang*[†], *Wenwu Wang*[‡], *Hongyu Yang*[◇], *Yitong Ma*[◇], *Yongji Ma*[◇]

[◇]College of Computer Science, Sichuan University, China.
[†] National Key Laboratory of Fundamental Science on Synthetic Vision, China.
[‡]Center for Vision Speech and Signal Processing, University of Surrey, UK.

ABSTRACT

Anomaly detection in computer vision aims to detect outliers from input image data. Examples include texture defect detection and semantic discrepancy detection. However, existing methods are limited in detecting both types of anomalies, especially for the latter. In this work, we propose a novel semantics-aware normalizing flow model to address the above challenges. First, we employ the semantic features extracted from a backbone network as the initial input of the normalizing flow model, which learns the mapping from the normal data to a normal distribution according to semantic attributes, thus enhances the discrimination of semantic anomaly detection. Second, we design a new feature fusion module in the normalizing flow model to integrate texture features and semantic features, which can substantially improve the fitting of the distribution function with input data, thus achieving improved performance for the detection of both types of anomalies. Extensive experiments on five well-known datasets for semantic anomaly detection show that the proposed method outperforms the state-of-the-art baselines. The codes will be available at https://github.com/SYLan2019/SANF-AD.

Index Terms— Semantic anomaly detection, Normalizing flow, Density estimation, Feature fusion, Class attention

1. INTRODUCTION

Anomaly detection in computer vision is a binary classification task which determines whether a given image deviates from a predetermined distribution. It is widely used in various fields, such as defect detection in industry [1], novelty detection [2], health monitoring [3], and video surveillance [4]. In general, we can divide it into texture defect detection and semantic anomaly detection. The former mainly focuses on the local information of the image, while the latter focuses on the image in semantic meaning.

In real world scenarios, the anomaly in data is complicated and sometimes unexpected. It is difficult for us to clearly define the specific abnormality of a system. Moreover, it is time-consuming to manually label the abnormal data required for training. Therefore, constructing an anomaly detection system based on the normal data is an alternative.

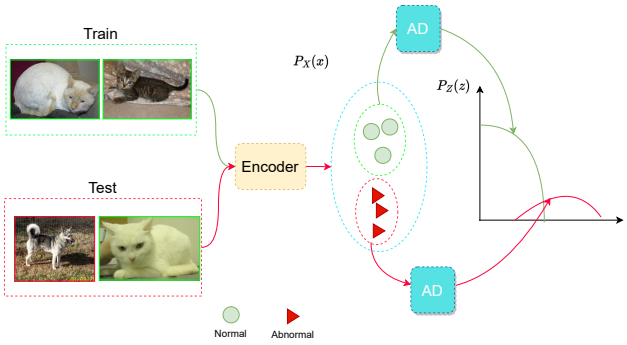

Fig. 1: An illustration of the proposed semantic anomaly detection method. We set *cat* as the normal class and *dog* as the abnormal class. Only the normal data is used for training. Our detector learns the distribution $P_X(\boldsymbol{x})$ of the normal input features extracted using a backbone, then transfers it to a Gaussian distribution $P_Z(\boldsymbol{z})$ by the normalizing flow-based model. In inference stage, the likelihood of the input features from the test data is computed through the distribution function fitted by the normalizing flow based model and compared with a threshold to detect the anomalies.

Existing anomaly detection methods can be mainly divided into four categories: unsupervised reconstruction-based methods [5, 6], self-supervised methods using proxy tasks [7], methods combining pretrained models and traditional clustering [8, 9], and methods based on normalizing flow and pretrained features [10, 11, 12].

In the reconstruction-based methods [6], such as generative adversarial network (GAN) based methods, a generator is often used to reconstruct the normal data, and the reconstruction error is then used to detect anomalies. However, this type of methods could be limited by the instability in model training, due to the use of GAN [13].

The self-supervised learning based methods have been used to obtain the features from raw images for anomaly detection, with various proxy tasks, including rotation classification [14], geometric transformations classification [15] and

*Corresponding author. E-mail: lanshiyong@scu.edu.cn.
This work was funded by 2035 Innovation Pilot Program of Sichuan University, China. This paper is accepted by ICME2023.

contrastive learning based classification [7].

A pretrained model has been used in [8] to extract prior information which could potentially mitigate the over-fitting problem caused by the lack of training data, as shown in the above methods. In Cohen [9], the pretrained features are refined with a modified the traditional knowledge distillation model to improve the anomaly detection results. However, these methods usually focus more on enhancing the features obtained from the pretrained model, but ignore the important role of semantic information extracted by the pretrained model.

In Rudolph et al [10], the pretrained features are used as the input to a normalizing flow model [16] in order to learn the distribution of normal data, leading to Differnet [10] and CSFlow [11], which were shown to achieve the state-of-the-art (SOTA) performance in texture defect detection. However, Kirichenko et al [12] pointed out that normalizing flow based models focus only on how to reconstruct feature information at the texture level, which is effective in discriminating semantic attributes. Nevertheless, for the semantic anomaly detection task, the global semantic information from the input images is crucial. As a result, existing normalizing flow based methods may have inherent limitations for semantic anomaly detection, as they are prone to model collapse when dealing with complex textural features [12].

To capture the semantic information of input images, global average pooling could be applied on local feature maps as conventional CNN models [17]. However, the pooling operation may lead to loss of some useful information. Recently, transformer-based vision (ViT) model [18] has achieved outstanding performance in many visual processing tasks, which can not only obtain the spatial textual feature map, but also the semantic information via the use of class token. Therefore, in this paper, we explicitly introduce the class token output from ViT as basic semantic information for semantic anomaly detection. In addition, we design a class attention feature fusion (CAFF) module that can further integrate some useful semantic information from the spatial features to enhance semantic distribution learning of the normalizing-flow-based model.

The principle of our proposed normalizing flow based semantic anomaly detection is shown in Fig. 1. Our main contribution can be summarized as follows.

- A new architecture of anomaly detection based on the normalizing flow model is proposed to map normal data to a normal distribution, in which the semantic attributes are explicitly used as learning objective in the process of fitting the distribution of normal data.
- We design a new feature fusion module in each layer of the normalizing flow model to integrate texture features and semantic features, which can improve the fitting of the distribution function to input data, thereby the performance for both types of anomaly detection.
- Extensive experiments on well-known datasets for semantic anomaly detection have demonstrated that our method outperforms the state-of-the-art baselines.

2. PROPOSED METHOD

2.1. Model overview

We propose an unsupervised method, by using only the normal data to train a model. During testing, both normal data and abnormal data are observed. According to [8], a good anomaly detection algorithm requires not only accurate feature representation but also high-performance classifiers. Therefore, we first summarize these two aspects of our model, i.e., feature extractor and the normalizing flow, where the normalizing flow is used as the high-performance classifier in our model. Then the detailed design of our method is introduced in Sections 2.2-2.4.

Feature Extractor: Pretrained CNN-based methods [17] have been used widely in current popular anomaly detection methods. However, these methods are sensitive to local information and suitable for handling local anomalies. Different from CNN-based methods [17], the attention based vision transformer can capture not only the global spatial features, but also the semantic attributes of an input image [9]. In addition, the unique design of the class token in the transformer structure allows the model to obtain semantic features by aggregating all the patches of the image without using a pooling operation. Furthermore, we design a new class attention feature fusion (CAFF) module to enhance the semantic feature representation, as detailed in Section 2.3.

The Normalizing Flow: Our proposed method estimate the density of the image semantic feature based on the normalizing flow. In order to pay more attention to semantic attributes, we take semantic features (i.e. class token) as the main input of our model. In addition, The CAFF module is embedded into each flow based learning layer of our model, which can further enhance the semantic feature representation by integrating useful information from spatial features. As shown in Fig. 2, our normalizing flow based model maps the semantic feature $\boldsymbol{y}$ of the image to the latent space $\boldsymbol{z}$ (i.e., $\boldsymbol{z} = F_{flow}(\boldsymbol{y})$), where $\boldsymbol{z}$ and $\boldsymbol{y}$ have same dimensions. We assume the feature $\boldsymbol{y}$ from space $\boldsymbol{Y}$ follows the distribution $P_Y(y)$, and the set $\boldsymbol{z}$ from space $\boldsymbol{Z}$ has a standard normal distribution (i.e., $\boldsymbol{z} \sim \mathcal{N}(0, I)$). During training, we maximize the likelihood of $\boldsymbol{z}$ transformed from $\boldsymbol{y}$, and due to the bijection of the normalizing flow [19], the likelihood of $\boldsymbol{z}$ can represent the distribution of the original $\boldsymbol{y}$. While at inference stage, the likelihood of the features from normal data will be greater than that from abnormal one, which can be compared with a threshold to determine the anomalies.

2.2. Normalizing flow for semantic anomaly detection

Following [19], $F_{flow} = f_1 \circ f_2 \circ ... \circ f_K$ in our model is shown in Fig. 2, which consists of a chain of flow-

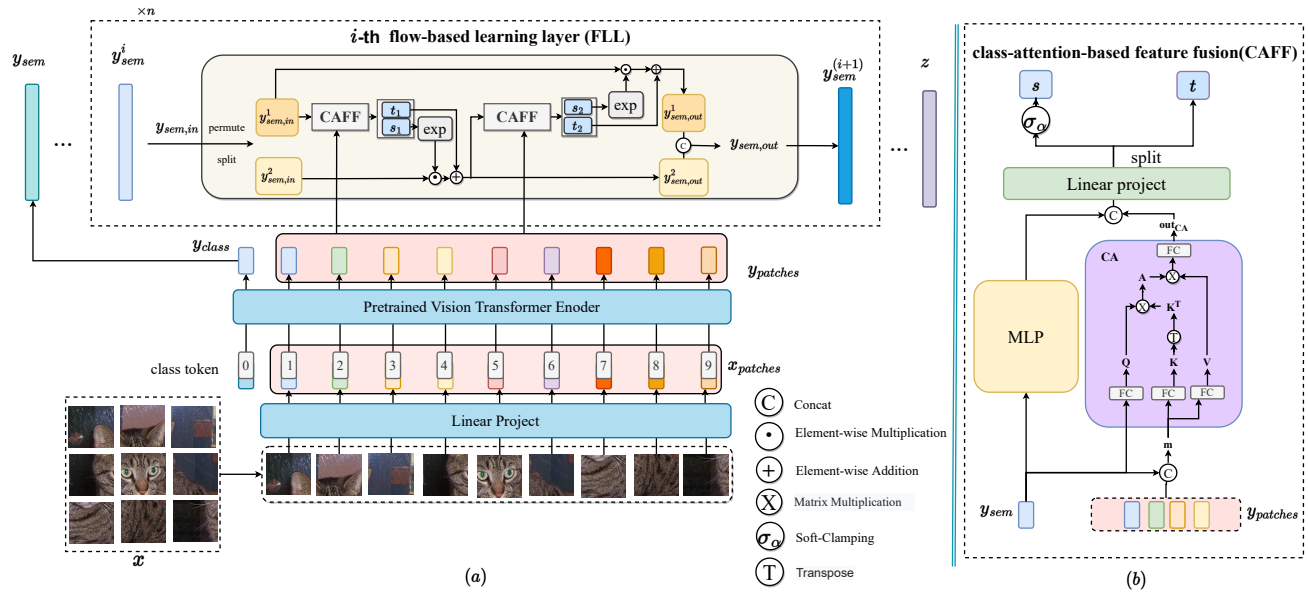

Fig. 2: Overview of the proposed method: As shown in (a), we use the pretrained ViT to extract semantic features $\boldsymbol{y}_{sem}$ and patch token $\boldsymbol{y}_{patches}$ of image X. Then we use n stacked flow-based learning layers (FLL) to transform $\boldsymbol{y}_{sem}$ into the latent variable $\boldsymbol{z}$. The detail of class-attention feature fusion (CAFF) module is shown in (b), which is used to fuse $\boldsymbol{y}_{patches}$ and $\boldsymbol{y}_{sem}$.

based learning layers (FLL). For the i-th FLL, we assume that the input is the semantic feature $\boldsymbol{y}_{sem,in}$, and the output is $\boldsymbol{y}_{sem,out}$. First, we randomly shuffle $\boldsymbol{y}_{sem,in}$ along the channel, and then evenly divide it into $\boldsymbol{y}^1_{sem,in}$ and $\boldsymbol{y}^2_{sem,in}$ along the channel. Then, $\boldsymbol{y}^1_{sem,in}$ and $\boldsymbol{y}^2_{sem,in}$ are fed into the class attention based feature fusion (CAFF) module to successively generate the scale and shift parameters $[\boldsymbol{s}_1, \boldsymbol{t}_1]$ and $[\boldsymbol{s}_2, \boldsymbol{t}_2]$. These two parameters will be applied to their respective counterparts (i.e., $\boldsymbol{y}^1_{sem,in}$, $\boldsymbol{y}^2_{sem,in}$) to calculate the output $[\boldsymbol{y}^1_{sem,out}, \boldsymbol{y}^2_{sem,out}]$. For simplicity of loss function calculation and the affinity property, we use an exponential function before scaling. These processes are described by:

$$[\boldsymbol{y}^1_{sem,in}, \boldsymbol{y}^2_{sem,in}] = split(\boldsymbol{y}_{sem,in}) \tag{1}$$

$$\boldsymbol{y}^2_{sem,out} = \boldsymbol{y}^2_{sem,in} \odot e^{\boldsymbol{s}_1(\boldsymbol{y}^1_{sem,in})} + \boldsymbol{t}_1(\boldsymbol{y}^1_{sem,in}) \tag{2}$$

$$\boldsymbol{y}^1_{sem,out} = \boldsymbol{y}^1_{sem,in} \odot e^{\boldsymbol{s}_2(\boldsymbol{y}^2_{sem,out})} + \boldsymbol{t}_2(\boldsymbol{y}^2_{sem,out}) \tag{3}$$

$$\boldsymbol{y}_{sem,out} = Concat(\boldsymbol{y}^1_{sem,out}, \boldsymbol{y}^2_{sem,out}) \tag{4}$$

where $\odot$ denotes the the element-wise multiplication, and $split(\cdot)$ means to evenly divide a vector into two vectors along the channel dimension.

Finally, following [20], we use soft-clamping to preserve the model stability through the activation:

$$\sigma_\alpha(\boldsymbol{s}) = \frac{2\alpha}{\pi} arctan \frac{\boldsymbol{s}}{\alpha} \tag{5}$$

where α is a hyperparameter, $\boldsymbol{s}$ is the scale parameter produced by the CAFF module. When the input of CAFF is $\boldsymbol{y}^1_{sem,in}$, $\boldsymbol{s} \triangleq \boldsymbol{s}_1$. Similarly, $\boldsymbol{s} \triangleq \boldsymbol{s}_2$, when input is $\boldsymbol{y}^2_{sem,in}$.

2.3. The class attention based feature fusion method

Except the semantic feature $\boldsymbol{y}_{sem}$, there may be some useful information in the remaining $\boldsymbol{y}_{patches}$. Different from the pooling layer commonly used in CNN, as [21], we use $\boldsymbol{y}_{sem}$ as the query to extract some helpful information from $\boldsymbol{y}_{patches}$, and generate the parmater $\boldsymbol{s}$ and $\boldsymbol{t}$. First, we use $\boldsymbol{y}_{sem}$ to produce Q, and concat $\boldsymbol{y}_{sem}$ and $\boldsymbol{y}_{patches}$ into $\boldsymbol{m}$ to produce K and V:

$$Q = W_q \boldsymbol{y}_{sem} + b_q \tag{6}$$

$$K = W_k \boldsymbol{m} + b_k \tag{7}$$

$$V = W_v \boldsymbol{m} + b_v \tag{8}$$

where $W_q, W_k, W_v \in \mathbb{R}^{d \times d}$ and $b_q, b_k, b_v \in \mathbb{R}^d$ are the parameters of the fully-connected (FC) layer (used to generate the embedding vector) on each branch of the attention module, which we call the class-attention (CA) module, here d is the size of each embedded vector. Then the class-attention weighting matrix is given by:

$$A = Softmax(QK^T)/\sqrt{d} \tag{9}$$

After that, the vector V weighted by the matrix A can be input into a FC layer to produce Out_{CA}:

$$Out_{CA} = W_o AV + b_o \tag{10}$$

where $W_o \in \mathbb{R}^{d \times d}$ and $b_o \in \mathbb{R}^d$ are the parameters of the FC layer. As shown in Fig. 2(b), $\boldsymbol{y}_{sem}$ is further encoded by an MLP module. Finally, the output of MLP and Out_{CA} are concated and fed into a linear projection module to produce the final results of the CAFF, i.e., $\boldsymbol{s}, \boldsymbol{t}$, as follows

$$[\boldsymbol{s}, \boldsymbol{t}] = split(FC(Concat(MLP(\boldsymbol{y}_{sem}), Out_{CA}))) \tag{11}$$

where $MLP(\cdot)$ represents a Multi-Layer Perceptron module.

2.4. Loss functions

We define the likelihood of the feature $\boldsymbol{y}_{sem}$:

$$P_Y(\boldsymbol{y}_{sem}) = P_Z(\boldsymbol{z}) \left| \ det \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{y}_{sem}} \ \right| \tag{12}$$

where $\boldsymbol{z} = F_{flow}(\boldsymbol{y}_{sem})$ and $f : Y \rightarrow Z$ is our model. It projects semantic feature $\boldsymbol{y}_{sem} \in P_Y(\boldsymbol{y})$ into latent feature $\boldsymbol{z} \in P_Z(\boldsymbol{z})$ in a reversible bijective manner. Thus, we can estimate the likelihood for $\boldsymbol{y}_{sem}$ from $P_Z(\boldsymbol{z})$. During training, we want to maximize the likelihood $P_Y(\boldsymbol{y}_{sem})$ of the features from normal data, which is equivalent to maximizing the likelihood $P_Z(\boldsymbol{z})$ of the projected latent variable $\boldsymbol{z}$.

To simplify the calculation, we take the log on both sides of the equation and set $\boldsymbol{z} \sim \mathcal{N}(0, I)$, we can get:

$$logP_Y(\boldsymbol{y}_{sem}) = logP_Z(\boldsymbol{z}) + log \left| \ det \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{y}_{sem}} \ \right| \tag{13}$$

Then we optimize our model by minimizing the negative log-likelihood $-logP_Y(\boldsymbol{y}_{sem})$:

$$\mathcal{L}oss = -logP_Y(\boldsymbol{y}) = \frac{\|\boldsymbol{z}\|_2^2}{2} - log \left| det \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{y}_{sem}} \right| \tag{14}$$

where $\|\cdot\|_2^2$ is L_2 norm, and $\left| det \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{y}_{sem}} \right|$ denotes the absolute determinant of the Jacobian matrix, whose derivation can be found in [16].

3. EXPERIMENTS

3.1. Datasets and Metrics

Datasets: Following [8, 9], our paper mainly focuses on semantic anomaly detection, so we use Cifar10 [22], Cifar100 [23], CatsVSDogs [8] and STL10 [24] datasets to evaluate the performance of our method. In order to further evaluate our model in other settings, we also tested our model on Mvtec [1] and Lbot [5] datasets. The proportion of training and testing splits in all datasets are shown in Table 1. In Fig. 3, we show several examples of images from these datasets.

Metrics: We use the Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) Curve as our evaluation metric. It calculates the true positive rate (TPR) and the false positive rate (FPR) by gradually changing the threshold of anomaly scores. Under different threshold conditions, there are different TPR and FPR. With FPR as the abscissa and TPR as the ordinate, we can approximately draw the ROC curve and get the AUC score. The AUC score represents the performance of the models under evaluation.

Table 1: Details of the datasets used in the experiments, including the number of classes in each dataset,and the average number of datasets for each class during training and testing.

Dataset	$N_{classes}$	N_{train}	N_{test}
Cifar10	10	5,000	10,000
Cifar100	20	6,000	10,000
Lbot	1	4,000	1,000
CatsVsDogs	2	10,000	5,000
STL10	10	500	8,000
Mvtec	15	242	1725

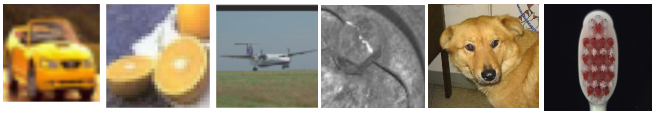

Fig. 3: Representative images from the datasets (from left to right, Cifar10, Cifar100, STL10, Lbot, CatsVSDogs and Mvtec). In the experiments, we have specified abnormal data for each category in the Mvtec and Lbot datasets. For other datasets, we follow the protocol of one-class anomaly detection, that is, set one class as normal and other classes as anomaly.

3.2. Implementation Details

As [9], we mainly follow the OCC setting, i.e., one class of data is normal and the remaining is abnormal. Our method is implemented by PyTorch 1.10.0 and CUDA 11.3, on a NVIDIA GeForce RTX 3090 GPU with 24Gb Memory. In the training process, the objective function is optimized by the Adam optimizer with momentum $\beta_1 = 0.8$, $\beta_2 = 0.8$, weight-decay 1e-5, and the initial learning rate 5e-5. The hyperparameter α is 3. For each dataset, the batchsize is set to 32. For the exteractor, we use a PyTorch implementation of ViT [18] as our pretrained extractor. The number of stacked flow-based learning layers (FLL) in our model is 4, and the number of hidden channels in each FLL is 1024.

3.3. Results

Comparision with the SOTA semantic anomaly detection methods: The compared methods include unsupervised Sagan [5] and AnoTrans [6], self-supervised CSI [7], and the pretrained Panda [8], MSAD [25] and Trans [9]. As shown in Table 2, the AUC of our proposed method on Cifar10 exceeds all other methods. We further compare our methods with the latest methods on additional datasets, as shown in Table 3. It demonstrates the effectiveness of our model on semantic anomaly detection.

Comparision with flow based methods: Considering that we are the first to apply the normalizing flow into semantic anomaly detection, we compare some flow-based anomaly detection methods in defect detection field including Differnet

Table 2: Auc results compare with other methods on Cifar10 dataset. Bold represent optimal results.

Model	Plane	Car	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck	Average
Sagan [5]	**1.000**	0.979	0.945	0.981	**0.999**	0.937	**1.000**	0.978	**1.000**	0.970	0.976
AnoTrans [6]	**1.000**	0.961	0.950	0.987	0.998	0.955	**1.000**	0.957	**1.000**	0.979	0.978
Panda [8]	0.975	0.987	0.936	0.895	0.975	0.942	0.981	0.973	0.978	0.977	0.962
MSAD [25]	0.975	0.987	0.955	0.941	0.973	0.971	0.983	0.982	0.986	0.982	0.974
CSI [7]	0.899	0.991	0.931	0.864	0.939	0.932	0.951	0.987	0.979	0.955	0.943
Trans [9]	0.966	0.985	0.966	0.960	0.985	0.983	0.985	0.991	0.984	0.991	0.981
Ours	0.995	**0.995**	**0.990**	**0.988**	0.995	**0.989**	0.999	**0.998**	0.997	**0.993**	**0.993**

[10], CSFlow [11] and FastFlow [26] on cifa10 and STL10. For Differnet and CSFlow, we keep the same settings as in their original paper. For FastFlow, we use the extractor as in our method. In Table 4, we can see that our method achieves the best results and it is demonstrated that our method is more suitable for semantic anomaly detection.

Comparision about computational efficiency: We compare the number of floats and parameters of our method with those of Panda [8], Trans [9], CSFlow [11] and Differnet [10]. For Panda [8], we investigate its network configuration, as used in the original paper. Considering that Trans [9] used a knowledge distillation model, what we compare here are the parameters and flops of the student network in the method (i.e. ViT(base)). As shown in Table 6, the number of parameters (used in training) and flops of our method are much smaller than those of the compared methods. However, our method still offers competitive detection accuracy as compared with the baselines, for example, our AUC index is over 3% higher than that of Panda, as shown in Table 2.

Table 3: The AUC results on additional datasets. SAD refers to semantic anomaly detection, while TDD refers to texture defect detection.

Type	Dataset	CSI [7]	Panda [8]	MSAD [25]	Trans [9]	Ours
SAD	Cifar100	0.896	0.941	0.964	0.973	**0.984**
SAD	CatsvsDogs	0.863	0.973	0.993	0.995	**0.996**
TDD	Mvtec	0.636	0.865	0.872	0.879	**0.941**

Table 4: The AUC results compared with existing flow-based anomaly detection methods.

Dataset	Differnet	CSFlow	FastFlow	Ours
Cifar10	0.695	0.955	0.961	**0.993**
STL10	0.814	0.989	0.991	**0.997**

Ablation study: Here we evaluate the impact of the use CAFF module on integrating potential semantic information from the spatial feature tokens of the ViT [18] extractor. As shown in Table 5, using the spatial feature patches is indeed helpful to improve the detection accuracy.

We further test the influence of different number of stacked FLLs (i.e., n_{blocks}) in our model. As shown in Table 7, with the increase in n_{blocks}, the performance of our model is greatly improved at first, and then decreases gradually after reaching the maximum. In order to achieve a balance between computational efficiency and model performance, we mainly set the extactor as ViT (large) and n_{blocks} as 4.

Table 5: The average AUC results of our model with and without CAFF.

Module	dataset	ViT (base)	ViT (large)
with CAFF	Cifar10	**0.986**	**0.993**
without CAFF	Cifar10	0.985	0.991
with CAFF	Mvtec	**0.941**	**0.951**
without CAff	Mvtec	0.881	0.892
with CAFF	Lbot	**0.911**	**0.941**
without CAFF	Lbot	0.874	0.892

Table 6: The number of parameters compared with baselines.

Method	Panda	Trans	CSFlow	Differnet	FastFlow$^{\triangle}$	Ours
params	60.2M	85.5M	275M	172M	31.8M	**13.6M**
flops	11.6G	1.93G	65.9G	**0.17G**	18.3G	0.69G

$^{\triangle}$ denotes re-implementation because of no official code released.

Table 7: Ablation study with the number of stacked FLLs on the dataset Cifar10.

n_{blocks}	2	4	6	8
ViT(base)	0.968	0.979	**0.985**	0.982
ViT(large)	0.992	**0.993**	**0.993**	0.991

4. CONCLUSION

In this paper, we have designed a new normalizing flow model to address problems of semantic anomaly detection, focusing on the learning of a mapping of semantic features, rather than the representation of spatial features as in conventional methods. In addition, we have considered utilizing spatial features to enhance the learning of semantic distribution. Our extensive experiments have shown that our model has achieved competitive performance for both textual defect detection and semantic anomaly detection, especially, obtaining a new SOTA performance for semantic anomaly detection on several well-known datasets. The number of parameters in our model is far less than that in baselines. In addition, the class-attention feature fusion module can indeed leverage useful semantic information from spatial features. In the future, we will explore its performance in anomaly detection for more types of anomalies.

5. REFERENCES

[1] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, "Mvtec ad–a comprehensive real-world dataset for unsupervised anomaly detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9592–9600.

[2] M. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*. IEEE, 2008, pp. 722–729.

[3] Z. Li, C. Wang, M. Han, Y. Xue, W. Wei, L. Li, and F. F. Li, "Thoracic disease identification and localization with limited supervision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8290–8299.

[4] Y. Liu, C. Li, and B. Póczos, "Classifier two sample test for video anomaly detections.," in *British Machine Vision Conference*, 2018, p. 71.

[5] G. Liu, S. Lan, T. Zhang, W. Huang, and W. Wang, "Sagan: Skip-attention gan for anomaly detection," in *2021 IEEE International Conference on Image Processing*, 2021, pp. 2468–2472.

[6] C. Yang, S. Lan, W. Huang, W. Wang, G. Liu, H. Yang, W. Ma, and P. Li, "A transformer-based gan for anomaly detection," in *International Conference on Artificial Neural Networks*, 2022, pp. 345–357.

[7] J. Tack, S. Mo, J. Jeong, and J. Shin, "Csi: Novelty detection via contrastive learning on distributionally shifted instances," *Advances in Neural Information Processing Systems*, vol. 33, pp. 11839–11852, 2020.

[8] T. Reiss, N. Cohen, L. Bergman, and Y. Hoshen, "Panda: Adapting pretrained features for anomaly detection and segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2806–2814.

[9] M. Cohen and S. Avidan, "Transformaly-two (feature spaces) are better than one," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2022, pp. 4060–4069.

[10] M. Rudolph, B. Wandt, and B. Rosenhahn, "Same same but differnet: Semi-supervised defect detection with normalizing flows," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer vision*, 2021, pp. 1907–1916.

[11] M. Rudolph, T. Wehrbein, B. Rosenhahn, and B. Wandt, "Fully convolutional cross-scale-flows for image-based defect detection," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 1088–1097.

[12] P. Kirichenko, P. Izmailov, and A. Wilson, "Why normalizing flows fail to detect out-of-distribution data," *Advances in Neural Information Processing Systems*, vol. 33, pp. 20578–20589, 2020.

[13] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," *arXiv preprint arXiv:1701.04862*, 2017.

[14] D. Hendrycks, M. Mazeika, S. Kadavath, and D. Song, "Using self-supervised learning can improve model robustness and uncertainty," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[15] I. Golan and R. El-Yaniv, "Deep anomaly detection using geometric transformations," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[16] D. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE onference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[18] T. Ridnik, E. Ben-Baruch, A. Noy, and L. Zelnik-Manor, "Imagenet-21k pretraining for the masses," *arXiv preprint arXiv:2104.10972*, 2021.

[19] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," *arXiv preprint arXiv:1605.08803*, 2016.

[20] L. Ardizzone, C. Lüth, J. Kruse, C. Rother, and U. Köthe, "Guided image generation with conditional invertible neural networks (2019)," *arXiv preprint arXiv:1907.02392*, 2018.

[21] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, and H. Jégou, "Going deeper with image transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 32–42.

[22] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[23] P. D'Oro, E. Nasca, J. Masci, and M. Matteucci, "Group anomaly detection via graph autoencoders," in *Neural Information Processing Systems Workshop*, 2019, vol. 2.

[24] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Journal of Machine Learning Research Workshop and Conference Proceedings, 2011, pp. 215–223.

[25] T. Reiss and Y. Hoshen, "Mean-shifted contrastive loss for anomaly detection," *arXiv preprint arXiv:2106.03844*, 2021.

[26] J. Yu, Y. Zheng, X.g Wang, W. Li, Y. Wu, R. Zhao, and L. Wu, "Fastflow: Unsupervised anomaly detection and localization via 2d normalizing flows," *arXiv preprint arXiv:2111.07677*, 2021.