Anomaly Detection via Reverse Knowledge Distillation. Analysis, comparison and extension.
ADLMM 2022 Project

Francesco Masciulli
Politecnico di Milano
francesco.masciulli@mail.polimi.it

Abstract

This work focus on the Student-Teacher framework applied on Anomaly Detection and Localization (AD&L) problems and its relation with the concept of Knowledge Distillation (KD). The main scope is to asses the performance of the architecture behind Reverse KD on the standard dataset MVTecAD, that measures the SOTA performance of AD&L task, and its comparison with other previously published architectures. A further step is to extend it in order to work with data from a different domain, such as the medical one. While the former is compared with the results published by the authors, the usage of data from different domains can be a powerful indicator on how the model will behave in such different field. The architecture was reproduced and trained, then the performance were measured and compared. The obtained model is able to reproduce the results on MVTecAD objects, while need some extension in order to catch anomalies on textures and medical datasets. The work shows that the proposed architecture solve the major problem of Uninformed students, that is dealing with multi-scale anomalies. Also, it is well suited to be rotational independent, how shown in our tests being the samples heterogeneously distributed.

1. Introduction

AD&L problems may be of interest for many different technical fields, from the manufacturing processes to the medical diagnosis. Multiple solutions based on Deep Learning have been proposed to face this problem. Their scope is to detect and localize anomalies within data samples that differs from a standard. In fact automatic detection and localization will help technicians to shrink the intervention time but also giving a powerful and precise visualization tool, in the case of a meaningful localization. Depending on the specific domain, the accuracy in the localization and the fast response could also be vital. For instance, the medical field is nowadays exploiting Machine Learning and Deep Learning algorithms in order to assist doctors in the diagnosis process. This is enhanced with an automatic tool that is faster and less human-error prone. The majority of AD&L solutions relies on semi-supervised learning, so that the model is trained as a one-class (the normal samples) learning problem and is able to recognize the unexpected anomalies, that are seen only at inference time. They will differ in any way from the normal samples. This different training paradigm arise from the difficulties of categorizing anomalies, focusing on the learning of normal samples. This technique helps the data collection process, focusing only on known and easily collectible normal samples.

The Student-Teacher Framework [6] proposed a simple but precise model for AD&L. This model trains an ensemble of (smaller) student networks to reproduce the hidden representation generated by a teacher network, which usually is a pretrained network used for transfer learning. This solution achieved a good accuracy while maintaining a relatively small computational demand with respect to the Generative counterparts. The mayor problem of this framework consists in handling anomalies with different region sizes. In practice, the final prediction is inferred by averaging output of numerous Student-Teacher architectures working with crops of different sizes. Furthermore, comparing the features extracted only by the last layer is not enough to fully exploit the knowledge of the teacher network.

A recent evolution proposes the use of KD in order to catch features generated by intermediate layers. Multiresolution KD [23] and Reverse KD [11] were proposed to exploit this technique. The Multiresolution KD is an encoder-like architecture that uses, during training, the discrepancy between latent representations generated by the (single) student and teacher networks. In this model the KD is

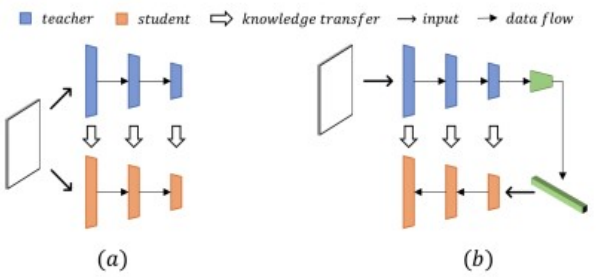

Figure 1. Comparison between Teacher-Student Multiresolution KD and Reverse KD [11].

exploited to concatenate student and teacher hidden representations. In Reverse KD, instead, the student is in practice a mirrored architecture with respect to the teacher one, thus creating a full autoencoder model. Also, knowledge is distilled in "reverse order", i.e. high-level representation is first distilled, followed by low-level features. The proposed architecture also use a One Class Bottleneck Encoder placed in between encoder and decoder. This helps to prohibit the propagation of unusual perturbations to the student model representation discrepancy on anomalies. It works by aligning the multi-scale encoder features and condensing them before being fed to the decoder part. The comparison of the two proposed methods (Multiresolution KD and Reverse KD) is reported in (Fig.1).

In this work we will focus on the Student-Teacher extensions based on KD. We want to asses the validity of these methodologies in both production and medical domain, in order to confirm a recent, but simple and powerful approach. We will start with an analysis of the Reverse KD architecture. Its implementation is recreated and tested against MVTecAD [5] Dataset. Then, it will be used to asses its performance on dataset from medical domain in order to prove its generalization power. The used data came from the Head-CT [15] dataset for brain hemorrhage, RetinalOCT [18] and Brain MRI Images For Brain Tumor Detection [9]. The main sections of the work can be resumed as:

- Brief description of the related works.
- Deeper analysis of the Reverse KD approach.
- Implementation choices and description, sum-up of the obtained results.
- Conclusions.

2. Related work

Early studies in AD&L proposed either miss anomaly localization [3, 8], or tackle the problem splitting images into smaller patches to determine the sub-regional abnormality [19]. This is computationally expensive and often leads to inaccurate localization. The majority of Deep Learning models that face AD&L are based on generative approaches, such as autoencoders [7, 1] or GAN [24]. These are trained from scratch, and results in poor performance due to per-pixel comparisons and reconstruction errors.

Some successful works have been presented, working with pretrained models that are been transferred and fine-tuned. Andrews et al. [3] uses latent features of a pretrained VGG [25] network and model the anomaly-free training distribution with a SVM. However, they only apply their method to image classification and do not consider the segmentation of anomalous regions. Napoletano et al. [19] extract features from a pretrained ResNet-18 [14] for a large number of cropped training patches and model their distribution using K-Means clustering after prior dimensionality reduction with PCA. The main throwback of their approach is that, extracting patches from input images they were not using the whole information during training. It was necessary using a very deep network and feature extraction was onerous. In order to circumvent the cropping, Sabokrou et al. [22] propose to extract features maps of earlier layers from a pretrained AlexNet [16]. The pooling lead to a downsampling of the input image, giving a worst resolution of Anomaly maps with larger receptive fields.

Methods like uninformed-students [7] focus on the meaningful information of normal data by building a compact feature representation through a convolutional network (the Teacher). An ensemble of (Students) networks will be trained to reproduce the output of the teacher. At inference time, the Students will assign a score to each pixel, measuring how much it is distant from the training manifold. The rationale behind it is that students will fail to reconstruct teacher output for anomalies because they were never seen during training. Here KD is performed by training the Teacher to act as an encoder and to mimic the hidden representation generated by a pretrained network for classification. The multi-scale problem, caused by the different possible sizes of anomalies, is tackled by using different ensembles with different receptive fields and averaging them to compute the final result.

Following studies [23, 10] highlights the representative power generated by intermediate layers, that are a solid representation of their input images. Salehi et al. propose the extension of KD by distilling the output of each intermediate layer of a pretrained VGG-16 [25] into one simpler (Student) network. Here, the main difference from Bergmann's work is that the student architecture is equivalent to the teacher. Consequently, the student learns the normal data

manifold thoroughly and yet earns no knowledge from the source about other possible input data, differently from the pretrained teacher. The Student was initially a clone of the Teacher architecture but the authors highlight how a simpler network have improved performance by avoiding not relevant filters to be coupled with the Teacher ones. This multiple distillation is done for including both more general and precise features from shallow and deeper layers, resulting in a Student more aware of different abstraction levels and more robust in representing normal data. Anomaly Detection is performed by measuring the loss between Teacher and Student embedding, while localization arise from the attribution map, where noise is reduced through Gaussian blur. The major attention must be posed on the fact that nonetheless Teacher and Student architectures are different, they follow the same "encoding" flow and so the anomaly map computation will be based on the discrepancy of latent features at the same deep level.

3. Proposed approach

Problem formulation. Let $I^N = \{i_1^N, .., i_k^N\}$ being k images of normal samples, and let $I^T = \{i_1^T, .., i_l^T\}$ being images of both normal and anomalies. The problem can be defined as discriminating samples of I^T between the two classes (AD) or localizing anomalies within the input image, if present.

Model overview The Reverse KD architecture proposed [11] is composed of three main components: a pretrained encoder *E*, a trainable one-class bottleneck embedding module *OCBE*, and a student decoder *D*. The described model is shown in Fig.2. At training time, each image from I^N is fed to *E*, which extract multiscale representation of the input. The output at each scale is passed to the *OCBE* module and from its output *D* is trained to reconstruct the output of *E*. At inference time, the normal samples will be correctly reconstructed while the anomalies will be easily encoded but *D* will likely fail, as highlighted in Fig.3.

3.1. Component analysis

Teacher encoder. The Teacher module is composed of a pretrained, deep network. It aims to extract meaningful representations of the input, that are distilled into the Student network. In order to avoid this powerful network to converge to trivial solutions, each layer of *E* is freezed during training. At each convolutional layer, the network will extract features of higher level, relative to wider receptive fields. The multi-scale distillation ensures that descriptor of both low and high level are distilled. Its implementation, among with all the other implementation choices, is described in Section 4.1.

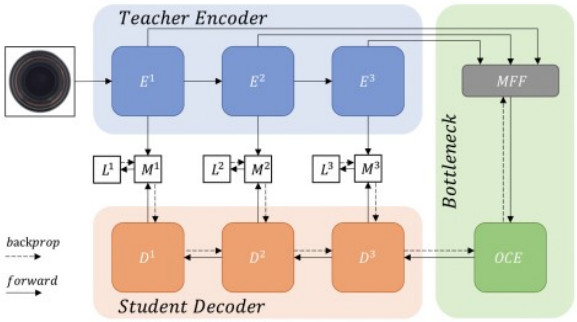

Figure 2. Reverse KD architecture overview [11]. The pretrained *E* output is compared with the one produced by *D*, and loss is computed starting from their distance.

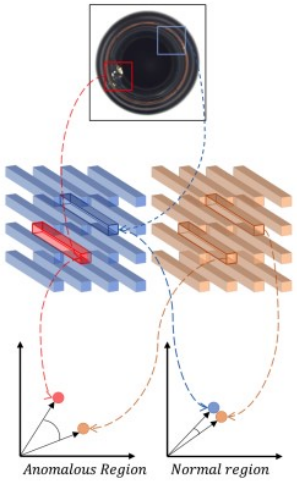

Figure 3. Example of discrepancies between reconstructed and encoded features, in case of anomalous or normal region [11].

OCBE module. The OCBE component is in charge of processing the output of *E* before feeding it to the Student. The authors state that by doing this step the descriptors extracted from the (complex) Teacher are less redundant. Then, the activation of the last encoder block in backbone usually characterizes semantic and structural information of the input data. Due to the reverse order of knowledge distillation, directly feeding this to the student decoder set a challenge for low-level features reconstruction. The OCBE module (Fig.4) works by processing and concatenating the feature extracted by each convolutional layer of *E* and then embed them using a Residual block.

Student decoder. In other studies [23, 7], the student network is a similar or identical neural network with respect to the teacher model, it accepts raw data/images as input, and it is trained to match its feature to the teacher's. In Reverse

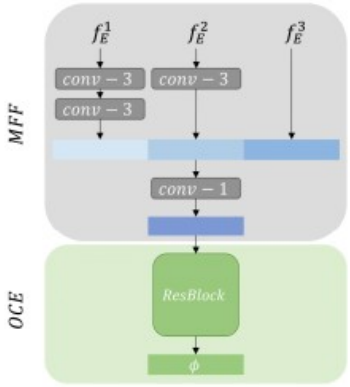

Figure 4. The OCBE module architecture [11]. The MFF module concatenate each feature f_i^E and the resulting descriptor is then embedded by the one-class embedding component.

KD framework, the student architecture is reversed, or mirrored, with respect to the teacher. In practice it acts as a decoder that aims to reconstruct the input features for each multi-scale level of features extracted by the teacher.

3.2. Mathematics

Loss function As was said, the features extracted by E are distilled into D, which input is their embedding generated by *OCBE*. The loss function must account for this distillation in order to train the model to match each level of hidden representations. Notice also that *OCBE* is trained in one pass, leading to a end-to-end training of the whole model, except for the freezed E. Let ϕ being the projection of an input I into *OCBE* space. Let also be $f_E^k = E^k(I)$ the feature extracted by the k^{th} encoder layer, and $f_D^k = D^k(\phi)$ the features of the k^{th} decoder layer. We know that $f_E^k, f_D^k \in \mathbb{R}^{C_k, H_k, W_k}$, where C_k, H_k, W_k are respectively the channels (i.e. number of filters), height and weight of the k^{th} layer. We can compute, for each f_E^k, f_D^k, their cosine similarity and the relative loss with respect to the channels axis, that will generate a 2-D anomaly map $M^k(h,w) \in \mathbb{R}^{H_k, W_k}$:

$$M^k(h,w) = 1 - \frac{f_E^k(h,w) \cdot f_D^k(h,w)}{\|f_E^k(h,w)\| \|f_D^k(h,w)\|} \tag{1}$$

Considering the multi-scale KD [12], the scalar knowledge distillation loss function L_{KD} (2) used for model optimization is obtained by accumulating $M^k(h,w)$ for each k^{th} hidden representation generated by both E and D, with K equal to the number of layers used in the specific implementation.

$$L_{KD} = \sum_{k=1}^{K} \frac{1}{H_k W_k} \sum_{h=1}^{H_k} \sum_{w=1}^{W_k} M^k(h,w) \tag{2}$$

It is worth to mention also the loss proposed by [23]. As will be better explained in Section 4.2, we have compared the results of training the model with both losses. The equation of L_{MKD} is shown in (4), where L_{val} (3) is the sum of euclidean distances between each Teacher and Student activation values and λ is a parameter that scales the cosine distance in order to be of the same order of magnitude of the other term.

$$L_{val} = \sum_{k=1}^{K} \frac{1}{H_k W_k} \sum_{h=1}^{H_k} \sum_{w=1}^{W_k} (f_E^k(h,w) - f_D^k(h,w))^2 \tag{3}$$

$$L_{MKD} = L_{val} + \lambda L_{KD} \tag{4}$$

Anomaly scoring At inference, regarding Anomaly Localization (AL), the afore-mentioned anomaly maps $M^k(h,w)$ are computed, giving a pixel-wise (but in the feature space) anomaly score. This score represent the fact that Student has not been able to reconstruct the features extracted by the teacher. To localize the anomalies within the input image, $M^k(h,w)$ is upsampled using bilinear interpolation. Let Ψ being the interpolation operation, we have that the pixel-wise anomaly map S_{AL} is computed (5). Also, a Gaussian filter is applied in order to smooth the resulting map. For what instead concerns Anomaly Detection (AD), being this a classification task, S_{AD} is used as threshold. It is defined as the maximum value within S_{AL}. It is worth to highlight that this score is different from what is used by [23], that instead uses directly the loss measure. This comparison will be inspected in Section 4.

$$S_{AL} = \sum_{l=1}^{L} \Psi(M^l) \tag{5}$$

4. Experiments

In the following section the implementation of our work is presented. It is important to highlight that we have followed, as much as possible, the directives presented in [11]. The whole project is implemented in *Python 3.9* [26], using *Pytorch* [20] for what concerns the every aspect of Deep Learning. Other important libraries used are *Scikit Learn* [21] and *Numpy* [13]. Our implementation is available on *GitHub*[17]. It is also fundamental to account *Anomalib* [2], that published (on 06/09/2022) an implementation of Reverse KD architecture. Our work started before that this implementation was made available, but as soon as it was released some of their components were integrated in our implementation, discarding ours. In particular, we have adapted from [2] the following components, in order to make them compatible with our work: OCBE module, Decoder module, Anomaly Maps generator, and Model definition. For what instead concerns the Encoder and each other concept used, we have defined them from scratch. More in-depth analysis will follow.

4.1. Model implementation

Datasets. We have reproduced the results obtained by Reverse KD [11] on **MVTecAD** [5] for AL and extended them on three dataset from medical domain, that are **Retinal-OCT** [18], **HeadCT** [15] and **BrainMRI** [9], comparing them with Multiresolution KD performance on AD. **MVTecAD** is the standard dataset used in AD&L studies. It is an industrial dataset with over 5k high-resolution images in 15 categories of objects and textures. Each category comprises Normal samples for training and a test set with both normal and anomalous samples with different defection. **Retinal-OCT** is a huge dataset of Retinal optical coherence tomography (OCT) images. It comprises over 26000 normal samples for training and 1000 test images divided in four categories between normal and anomalies. For computational reasons we had used only half of the training set. **HeadCT** is a medical dataset containing 100 normal head CT images and 100 with hemorrhage. **BrainMRI** consist of 98 normal MRI images and 155 with tumors. For both of these last dataset, 10 normal images are used for testing among with all the anomalous. For what instead regard the first two dataset, we had train and test set already defined. Each input is resized to $(224, 224)$ and preprocessed with the function relative to the Encoder backbone.

Experiments setup. Following [11], the Encoder module is composed of the first three convolutional blocks of *Pytorch Wide ResNet 50* [29]. A custom module is created to keep only the desired layers and to return each generated latent representation, that will be used for KD. The OCBE module consist in a series of convolutions that are applied, in different number, to the Encoder features. They are then concatenated and fed to a Residual block before being embedded. The Decoder is the mirrored architecture with respect to the selected Encoder blocks, that upsamples the embedded representation, trying to match Encoder features at each block. The KD is applied through the loss (2), leading to an end-to-end optimization process. Initially, the model is trained for 200 epochs, without being validated. It uses Adam optimizer with learning rate of 0.005 and beta = (0.5, 0.99).
At inference, the anomaly map is generated and the AUROC computed comparing ground truth with the model output, following the anomaly scoring previously presented *Eq. 5* for AD or the anomaly map in case of AL. In this latter case, also the Per-Region Overlap (PRO) [4] is computed between each pixel of ground truth and anomaly map. For this, the optimal threshold was computed from Youden's J statistic [28] in order to set the optimal classification threshold. The problem is ultimately reduced to binary classification. Some extensions to Reverse KD approach are presented in 4.3, whenever the expected results deviated from the ones obtained.

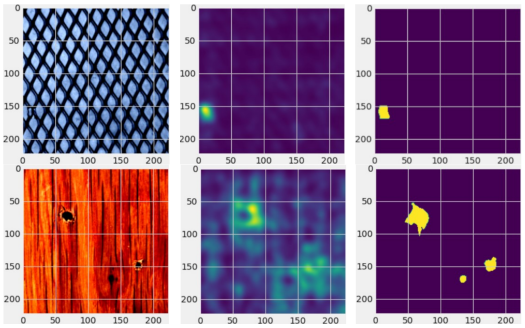

Figure 5. Comparison between best and worst object categories, which are *grid* and *wood* classes. We show, in order, a preprocessed sample, the anomaly map generated and the ground truth. While the former category is perfectly predicted, the latter anomaly map is not catching the anomalies.

4.2. Results evaluation and comparison

MVTecAD. The complete results obtained by our implementation, compared with the performances of Multiresolution KD [23] and Reverse KD [11] in AL task, are shown in *Table 1* and *Table 2*.
The complete results obtained by our implementation are shown , where are compared with the performances of Multiresolution KD [23] and Reverse KD [11] in AL task.
It is worth to highlight that almost each object obtained comparable AUROC and PRO scores, while our results on textures were notable different. Starting from this point, we have tested different approaches that are better explained in Section 4.3. Shortly, the trained was performed by using L_{MKD} as expressed in Section 3.2. For what instead concerns the PRO score, our results are better. Using the Youden's J statistic we have tuned the threshold that classifies each anomaly map pixel. In *Fig.5, 6* we show the generated anomaly maps among with ground truth and the correspondent preprocessed image. While the objects categories are correctly predicted, the model does not catch correctly the anomalies in textures. For what regards *wood* category, probably the non uniform texture led to a more difficult learning. Furthermore, not validating the model training had caused an overfitting of the training set.

Medical Datasets. We have trained our model on three different dataset belonging to medical domain, and compared our results with the one obtained in [23]. They are all facing AD task, and any mask was available in order to test also AL. The results are shown in *Table 3*. The model was initially trained as described in [11], and some extensions were performed in order to understand how to increase the accuracy. These extensions are explained in Section 4.3, shortly we have used the loss as anomaly score following

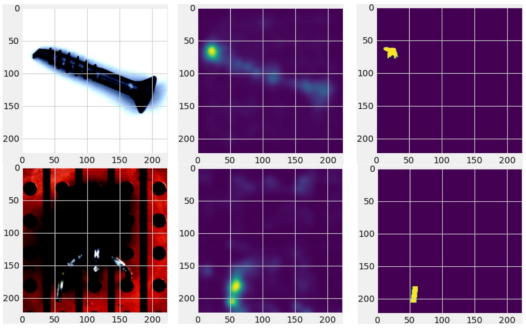

Figure 6. Comparison between best (*screw*) and worst (*transistor*) object categories. We show, in order, a preprocessed sample, the anomaly map generated and the ground truth. The latter almost perfectly overlap with our prediction in both cases.

Category/Method		AUROC%/PRO%		
		MKD	RKD	Ours
Textures	Carpet	95.6/_	**98.9**/**97.0**	82.0/73.6
	Grid	91.8/_	**99.3**/97.6	99.2/**98.1**
	Leather	98.1/_	**99.4**/**99.1**	91.7/93.5
	Tile	82.8/_	**95.6**/**90.6**	83.4/68.5
	Wood	84.8/_	**95.3**/**90.9**	79.0/68.0
	Average	90.6/_	**97.7**/**95.0**	87.1/80.3

Table 1. Results on **MVTecAD** textured categories, compared with [23, 11]. While RKD outperforms MKD, our performance are lower for each category except Grid. In this case, we have a better PRO value thanks to the optimal threshold tuning. Values in bold are the best among the compared.

Category/Method		AUROC%/PRO%		
		MKD	RKD	Ours
Objects	Bottle	96.3/_	**98.7**/96.6	98.4/**97.4**
	Cable	82.4/_	**97.4**/91.0	97.0/**91.7**
	Capsule	95.9/_	**98.7**/**95.8**	96.9/90.2
	Hazelnut	94.6/_	**98.9**/95.5	97.5/**96.4**
	Metal Nut	86.4/_	97.3/92.3	**98.4**/**96.3**
	Pill	89.6/_	98.2/**96.4**	**98.7**/94.5
	Screw	96.0/_	**99.6**/**98.2**	99.5/97.7
	Toothbrush	96.1/_	**99.1**/94.5	**99.1**/**98.3**
	Transistor	76.5/_	**92.5**/78.0	92.1/**81.7**
	Zipper	93.9/_	**98.2**/**95.4**	95.7/87.9
	Average	90.8/_	**97.9**/**93.4**	97.3/93.2

Table 2. Results on **MVTecAD** object categories, compared with [23, 11]. Is highlighted that the AUROC is similar for almost each category, while PRO is slightly better in our implementation. This may be because of the optimal threshold computation. Values in bold are the best among the compared.

what was done by [23], we have validated the model using early stopping in order to avoid overfitting and finally we have trained the model optimizing L_{MKD}. The main result is that, using a huge dataset such as **Retinal-OCT**, our model easily achieves comparable performance in its original configuration. Being the training with that dataset very computational expensive, we had to use only half of the training set. Nonetheless, we achieved an AUROC over 94%. For this reason we had not extended the model for this dataset. In the other cases, we obtained divergent results. While training on **Head CT** we obtained very similar performace, we fall on **Brain MRI**. In both cases we had improved our results by changing the anomaly scoring measure and by optimizing the loss function used in Multiresolution KD. In the former dataset, the usage of L_{MKD} in both training and inference (as anomaly score) had not improved the results, while in the latter it gave the best performance for our configuration

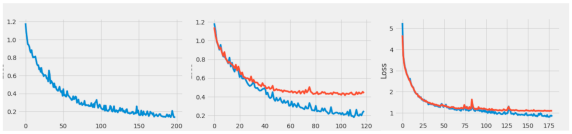

Figure 7. Comparison on **Head CT** loss during training in case early stopping is performed and the L_{MKD} is used.

4.3. Model extensions

Validating the model. The first extension that we have made with respect to what is explained in [11] is validating the training process against a never-seen set of normal sample. For each epoch we measure the loss on the validation set and early stop the training if it have not decreased for 15 epochs. This is done in order to avoid overfitting. The validation set is extracted from the training set, keeping the 25% of samples. This is applied to two out of the three medical dataset (see *Table 3*) and generally improve the model performance (see *Table 4*).

Using a different measure for Anomaly Scoring For what concerns AD, we had also changed the anomaly score from S_{AD} to what is proposed in [23]. In practice, we had measured directly the loss of the model at inference time. Doing this, the performance on **Brain MRI** and **Head CT** increases while the opposite happens for **Retinal-OCT**.

Using a different Loss function The last extension is performed by changing the loss function that is optimized (*See Fig. 7*). As described in Section 3.2, we have trained the model with both L_{KD} [11] and L_{MKD} [23]. The results (*Table 3, 4*) highlights how the more complex L_{MKD}, accounting also for the similarity of "straight" activations among with their directions, is improving the learning of normal distributions.

Dataset	MKD	Ours	Ours+ES	Ours+L_{KD}AS	Ours+L_{MKD}	Ours+L_{MKD}AS
Retinal-OCT	**97.0**	94.6	_	90.24	_	_
Brain MRI	**95.0**	85.6	84.1	85.2	91.2	93.0
Head CT	78.0	77.2	81.3	82.9	**85.2**	79.1

Table 3. Results on medical datasets, compared with [23]. Ours+ES is validated with Early stopping, Ours+L_{KD}AS uses L_{KD} as anomaly score, Ours+L_{MKD} is trained with L_{MKD} and L_{MKD}AS is trained with L_{MKD} and uses it as anomaly score. The results shows how Reverse KD performance generally improves when L_{MKD} is used. Values in bold are the best among the compared.

Category/Method		AUROC%/PRO%		
		RKD	Ours	Ours+L_{MKD}
Textures	Carpet	**98.9**/**97.0**	82.0/73.6	97.9/92.7
	Grid	**99.3**/97.6	99.2/**98.1**	-/-
	Leather	**99.4**/**99.1**	91.7/93.5	92.6/98.3
	Tile	95.6/**90.6**	83.4/68.5	**95.9**/89.6
	Wood	**95.3**/**90.9**	79.0/68.0	91.9/84.3
	Average	**97.7**/**95.0**	87.1/80.3	94.58/91.23

Table 4. Results on **MVTecAD** textures categories, compared with [11] and with ours model optimized with L_{MKD}. The usage of this different loss function arise performance on each object, leading to a more comparable result with respect to the original study.

5. Conclusion

Our work first aim was to reproduce the Reverse KD architecture proposed by *Deng et al.* [11]. Thanks to [2], the work later focused on integrating the already available implementation in what was already done. This lead us to extend the configuration and test some possible extensions and solutions to improve the performance that we have obtained. The main problem was with textures, but was partially solved using L_{MKD} loss function among with validation techniques. For what concerns medical datasets, the main problem still is related to the pretrained backbone. ImageNet samples are different from the images of medical domain. In order to account for these, the recent *RadImageNet* [27] was published. We believe that this can help, but the obtained results are also optimistic as they are. The main conclusion is that both Multiresolution and Reverse KD are facing and solving the most important issue of Uninformed students, that is the multi-scale problem.

References

[1] D. Abati, A. Porrello, S. Calderara, and R. Cucchiara. Latent space autoregression for novelty detection. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 481–490, 2019. 2

[2] S. Akcay, D. Ameln, A. Vaidya, B. Lakshmanan, N. Ahuja, and U. Genc. Anomalib: A Deep Learning Library for Anomaly Detection, 2 2022. 4, 7

[3] J. Andrews, T. Tanay, E. Morton, and L. Griffin. Transfer representation-learning for anomaly detection. 07 2016. 2

[4] P. Bergmann, K. Batzner, M. Fauser, D. Sattlegger, and C. Steger. Beyond dents and scratches: Logical constraints in unsupervised anomaly detection and localization. *International Journal of Computer Vision*, 130, 04 2022. 5

[5] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger. Mvtec ad — a comprehensive real-world dataset for unsupervised anomaly detection. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9584–9592, 2019. 2, 5

[6] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger. Uninformed students: Student-teacher anomaly detection with discriminative latent embeddings. *CoRR*, abs/1911.02357, 2019. 1

[7] P. Bergmann, S. Löwe, M. Fauser, D. Sattlegger, and C. Steger. Improving unsupervised defect segmentation by applying structural similarity to autoencoders. *CoRR*, abs/1807.02011, 2018. 2, 3

[8] P. Burlina, N. Joshi, and I.-J. Wang. Where's wally now? deep generative and discriminative embeddings for novelty detection. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11499–11508, 2019. 2

[9] N. Chakrabarty. Brain mri images for brain tumor detection. https://www.kaggle.com/datasets/navoneel/brain-mri-images-for-brain-tumor-detection. 2, 5

[10] H. Cheng, L. Yang, and Z. Liu. Relation-based knowledge distillation for anomaly detection. In H. Ma, L. Wang, C. Zhang, F. Wu, T. Tan, Y. Wang, J. Lai, and Y. Zhao, editors, *Pattern Recognition and Computer Vision*, pages 105–116, Cham, 2021. Springer International Publishing. 2

[11] H. Deng and X. Li. Anomaly detection via reverse distillation from one-class embedding. *CoRR*, abs/2201.10703, 2022. 1, 2, 3, 4, 5, 6, 7

[12] J. Gou, B. Yu, S. J. Maybank, and D. Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, mar 2021. 4

[13] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020. 4

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015. 2

[15] F. Kitamura. Head ct - hemorrhage. https://www.kaggle.com/datasets/felipekitamura/head-ct-hemorrhage/metadata. 2, 5

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc. 2

[17] F. Masciulli. Reverse kd implementation. https://github.com/FraqMasq/RKD_AD. 4

[18] P. Mooney. Retinal oct images (optical coherence tomography). https://www.kaggle.com/datasets/paultimothymooney/kermany2018. 2, 5

[19] P. Napoletano, F. Piccoli, and R. Schettini. Anomaly detection in nanofibrous materials by cnn-based self-similarity. *Sensors (Basel, Switzerland)*, 18, 2018. 2

[20] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 4

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 4

[22] M. Sabokrou, M. Fayyaz, M. Fathy, and R. Klette. Fully convolutional neural network for fast anomaly detection in crowded scenes. *CoRR*, abs/1609.00866, 2016. 2

[23] M. Salehi, N. Sadjadi, S. Baselizadeh, M. H. Rohban, and H. R. Rabiee. Multiresolution knowledge distillation for anomaly detection, 2020. 1, 2, 3, 4, 5, 6, 7

[24] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. *CoRR*, abs/1703.05921, 2017. 2

[25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition, 2014. 2

[26] G. Van Rossum and F. L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. 4

[27] Y. Yang, X. Mei, P. Robson, B. Marinelli, M. Huang, A. Doshi, A. Jacobi, K. Link, T. Yang, C. Cao, Y. Wang, H. Greenspan, T. Deyer, and Z. Fayad. Radimagenet: A large-scale radiologic dataset for enhancing deep learning transfer learning research. 06 2021. 7

[28] W. J. Youden. Index for rating diagnostic tests. *Cancer*, 3(1):32–35, 1950. 5

[29] S. Zagoruyko and N. Komodakis. Wide residual networks, 2016. 5