Quality Control

October 2024

Our Case Study

- A car manufacturer aims to enhance and automate the inspection of critical vehicle labels during the production process.

- Different label type:
    - vehicle identification number (VIN)
    - tire pressure labels

- Inspection Process :
    - Photographed using stationary cameras and mobile devices

Challenge

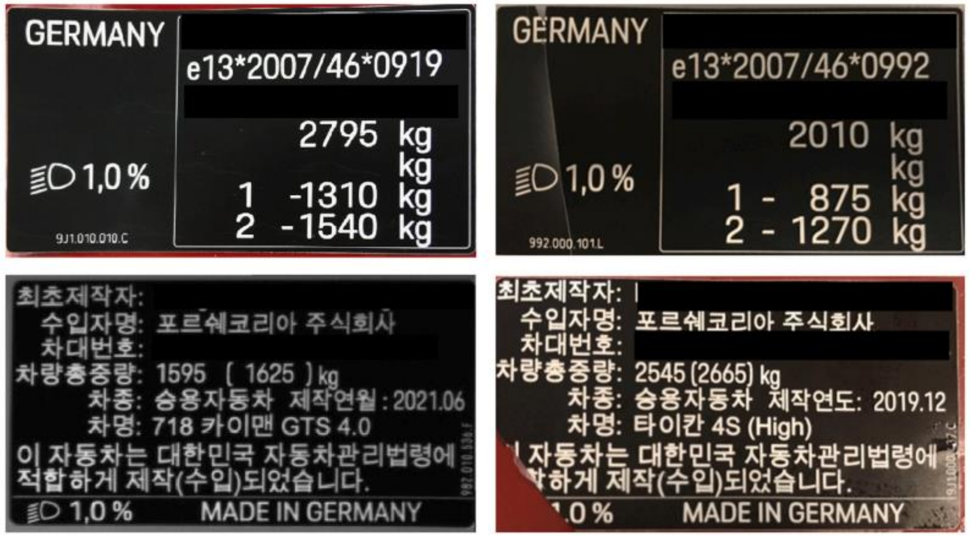
Varying lighting environment

Camera angles and distances

Use mobile camera for taking picture

Large variety of training data

- Labels differ by country with no consistent structure



Supervised Learning

- labeled instances are rarely available
- Impossible to collect sufficient number of labeled
- Costly to synthetically samples
- Often dont known which types of defects may happen

Unsupervised Learning

Traine only on normal instances.

Learn normal quality then deviations can be detected

Not relying on pre-defined classes and labeled samples

Dataset

- 2703 undamaged images
- 970 damaged images
- Average resolution of the original images was 1347 × 723
- Images were cropped out, standardized, and reduced to a unified size.
- All images resized to 256 × 256 pixels
- Use Mask R-CNN to segment labels

Approaches

01

Skip-GANomaly(unsupervised)

02

PaDiM(unsupervised)

03

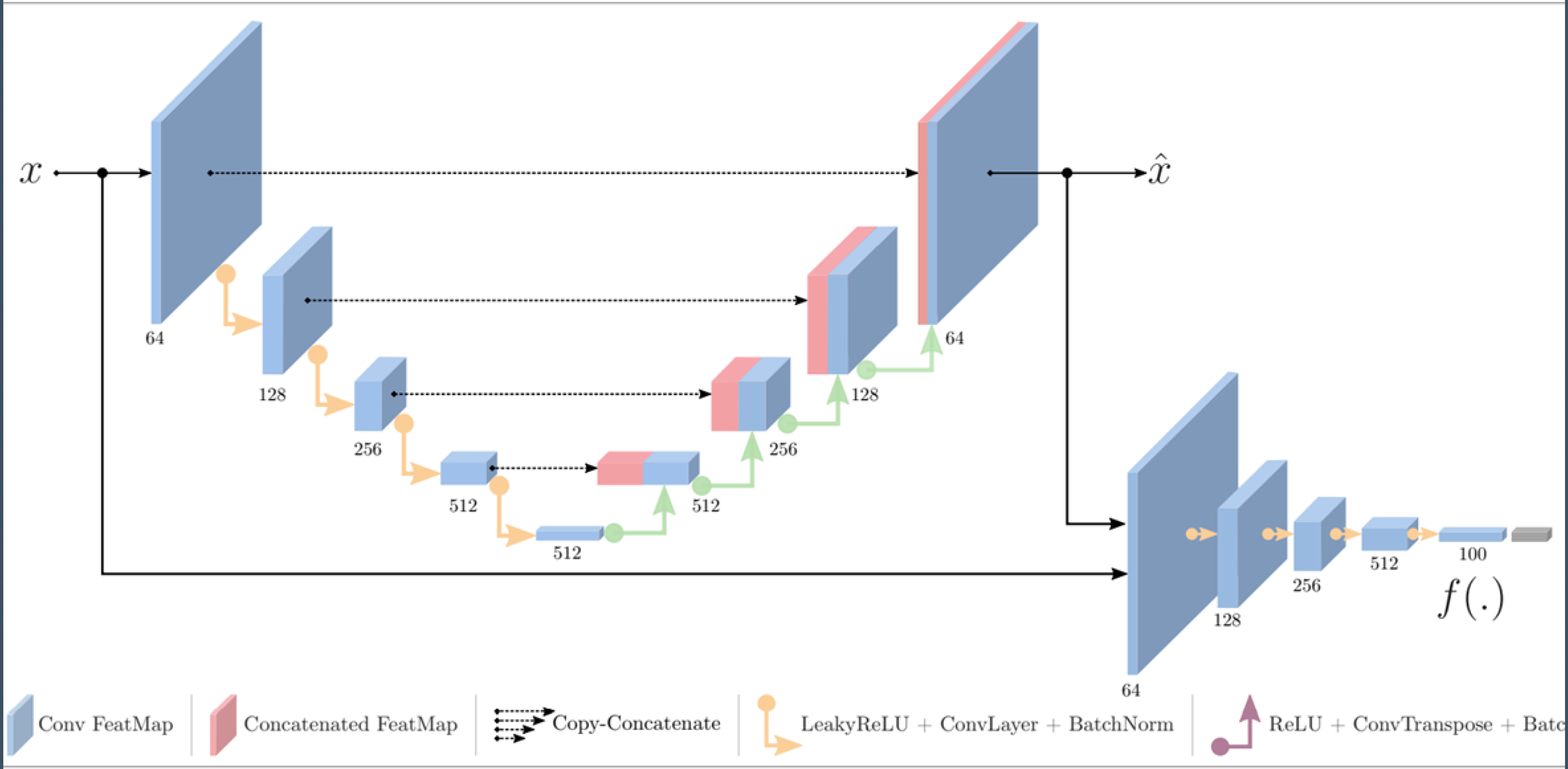
PatchCore(unsupervised)

04

Auto-Classifier(supervised)

Skip-GANomaly

- Train on normal images
- Test on both normal and damaged
- Similar to GAN
  - generator based on an autoencoder
  - discriminator network
- Discriminator
  - distinguish reconstructed images from the original ones
- Anomaly score is weighted sum of the:
  - reconstruction loss
  - Latent-representation loss (L2-norm)



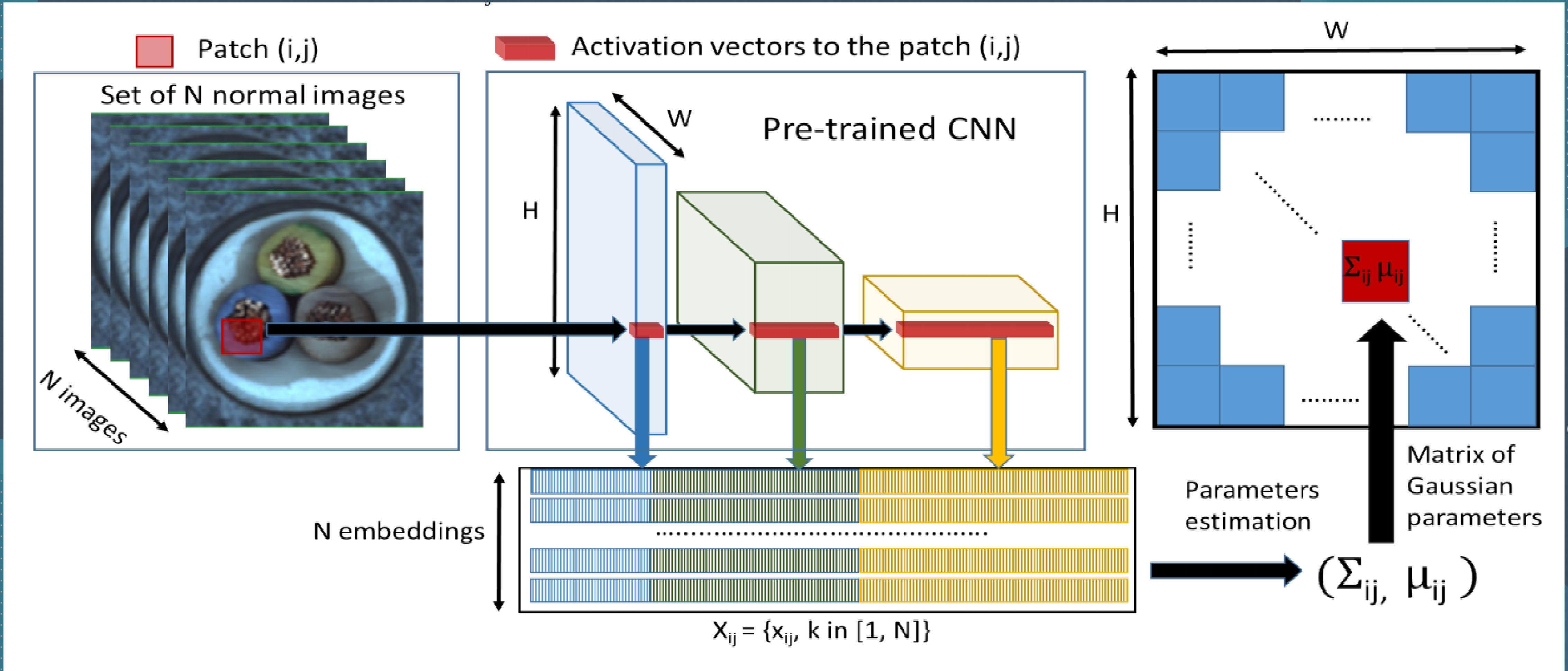
x
$\hat{x}$
64
128
256
512
512
512
256
128
64
64
128
256
512
100
$f(.)$
Conv FeatMap
Concatenated FeatMap
Copy-Concatenate
LeakyReLU + ConvLayer + BatchNorm
ReLU + ConvTranspose + Batc

LOSS FUNCTIONS

- Use 3 diffrent loss function at same time:
  - Adversarial Loss:
    - ensures that the network G reconstructs image as realistically as possible.
  - Contextual Loss:
    - explicitly learn contextual information
  - Latent Loss:
    - reconstruct latent representations for the input x and generated image as similar as possible

PaDiM

- Based on embedding vectors similarity
- Several pre-trained CNN backbone networks
  - ResNet18
  - WideResNet50
  - EfficientNetB5
- **extract features** without weight adjustment or backpropagation.
- **localizes** the regions of anomaly



Patch (i,j)
Activation vectors to the patch (i,j)
Set of N normal images
N images
W
H
Pre-trained CNN
N embeddings
$X_{ij} = \{x_{ij}, k \text{ in } [1, N]\}$
W
H
$\Sigma_{ij}\ \mu_{ij}$
Parameters estimation
Matrix of Gaussian parameters
(Σ_{ij}, μ_{ij})

Other information

To reduce size of embedding vectors, randomly selecting few dimensions

More efficient than PCA

Anomaly score is distance between the test patch embedding xij and learned distribution

$$M(x_{ij}) = \sqrt{(x_{ij} - \mu_{ij})^T \Sigma_{ij}^{-1} (x_{ij} - \mu_{ij})}$$

PatchCore

Only requires normal images for training

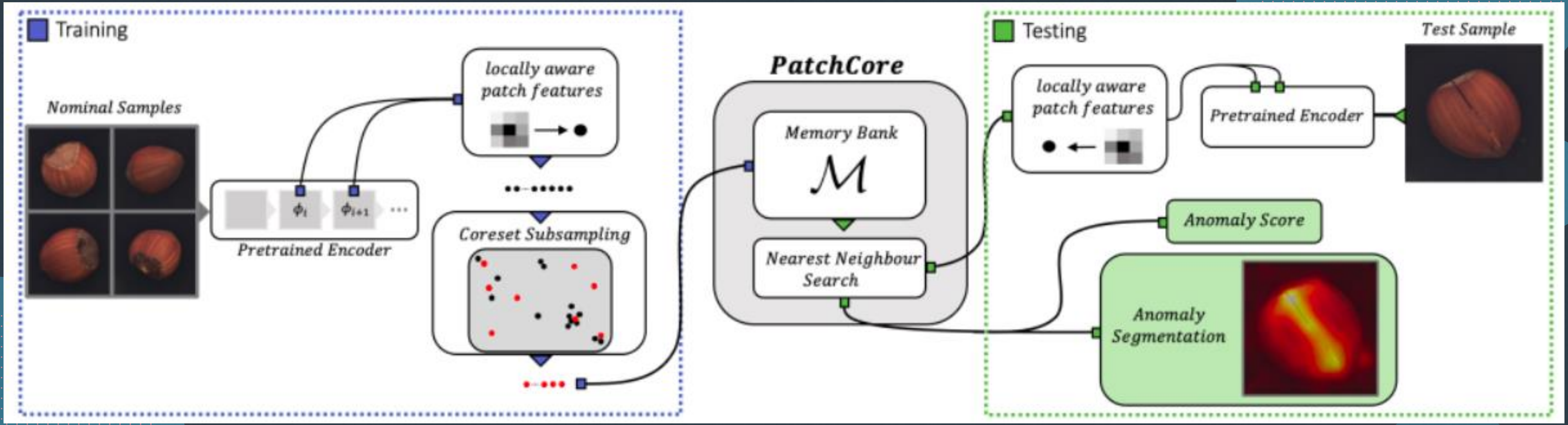
Like PaDiM, extract embedding vectors

Use ImageNet

localizes the regions of anomaly

Process:

- First, embedding vectors stored in a memory bank
- To reduce memory bank size, subsampled by applying a k-center-greedy algorithm
  - sampling only 1% of the patch representations to be in the memory bank is sufficient to get good performance



Training
Nominal Samples
Pretrained Encoder
ϕ_i
ϕ_{i+1}
locally aware
patch features
Coreset Subsampling
PatchCore
Memory Bank
$\mathcal{M}$
Nearest Neighbour
Search
Testing
Test Sample
locally aware
patch features
Pretrained Encoder
Anomaly Score
Anomaly
Segmentation

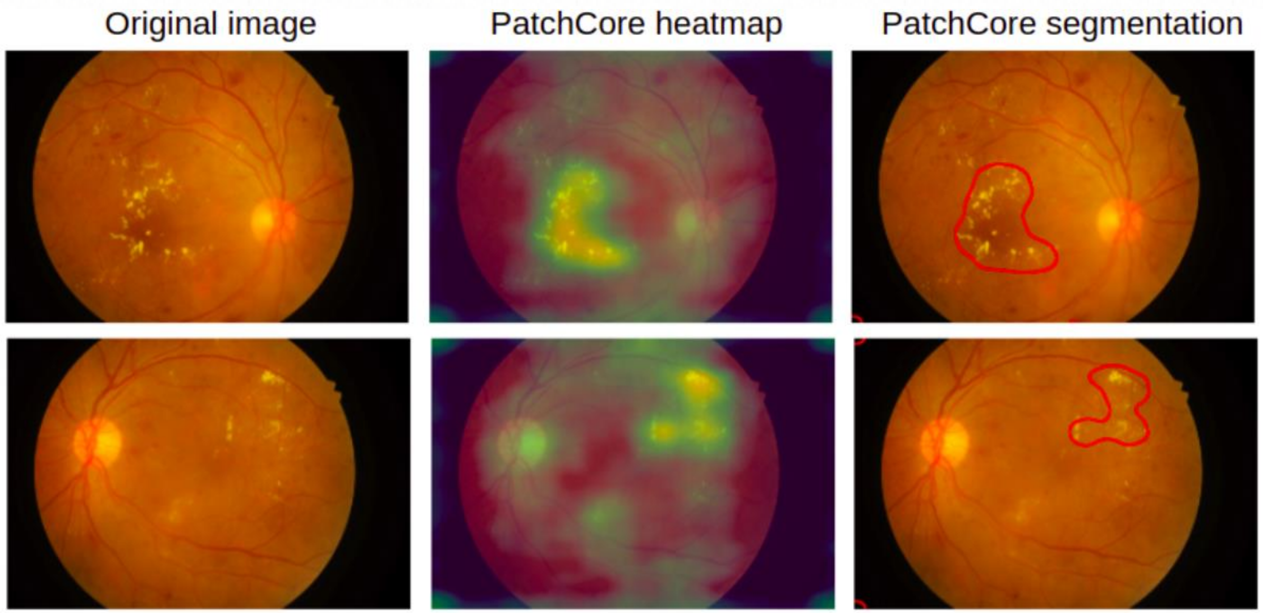
Testing

Extract embedding vectors of the input image in the first step

Calculate distances to the vectors of the embedding coreset using the k-nearest neighbor

anomaly score is calculated based on nearest neighbor weighted by the other k distances

Example


Original image
PatchCore heatmap
PatchCore segmentation

Auto-Classifier

- A supervised approach
- Works on 2D images
- Balance the importance of each network through the process of normalization
- Networks with higher AUC scores have higher confidence

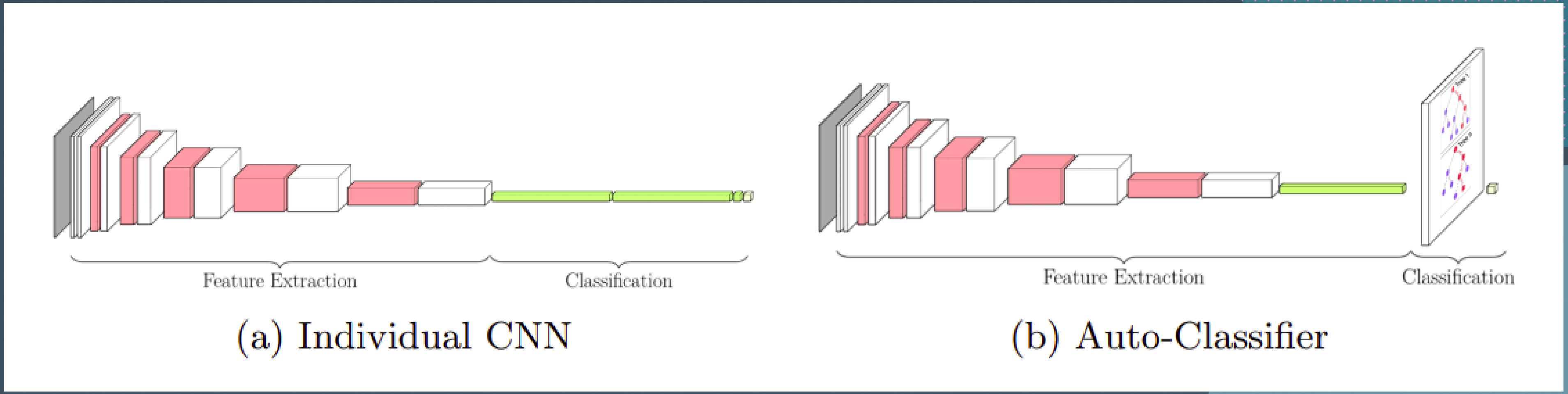
Process:

- Consider several CNN architectures
  - VGG11
  - VGG16
  - VGG19
  - ResNet18
- Fusing all the individual predictions into a final, weighted, prediction by making a weighted sum of each class

$$w_i = \frac{V_i}{\sum_{j=1}^{n} V_j}, i \in 1, \ldots, n$$

Auto-Classifier

Second part:



Results

Table 3

Evaluation results of the anomaly detection performance based on the test set.

Model	Backbone	*AUROC*	*F1 Score*	$Recall_{Pr=0.996}$	$Recall_{Pr=0.95}$	$Recall_{Pr=0.9}$
Auto-Classifier	ResNet50	1.000 ± 0.000	0.995 ± 0.005	0.995 ± 0.009	1.000 ± 0.000	1.000 ± 0.000
	Fusion	1.000 ± 0.001	0.995 ± 0.006	0.792 ± 0.443	1.000 ± 0.000	1.000 ± 0.000
	AutoML	1.000 ± 0.000	0.992 ± 0.003	0.978 ± 0.022	1.000 ± 0.000	1.000 ± 0.000
Skip-GANomaly	–	0.930 ± 0.006	0.757 ± 0.009	0.130 ± 0.081	0.214 ± 0.166	0.275 ± 0.160
PaDiM	EfficientNetB5	0.992 ± 0.002	0.911 ± 0.009	0.438 ± 0.240	0.884 ± 0.031	0.940 ± 0.020
	ResNet18	0.978 ± 0.003	0.874 ± 0.010	0.230 ± 0.092	0.486 ± 0.206	0.790 ± 0.048
	ResNet50	0.982 ± 0.005	0.897 ± 0.017	0.205 ± 0.143	0.697 ± 0.112	0.841 ± 0.080
	WideResNet50	0.984 ± 0.005	0.893 ± 0.033	0.318 ± 0.113	0.712 ± 0.136	0.867 ± 0.087
PatchCore	WideResNet50	0.996 ± 0.002	0.948 ± 0.015	0.451 ± 0.245	0.941 ± 0.033	0.986 ± 0.017

Training time

Table 4
Evaluation results of training and inference times for a single run.

Model	Backbone	Train time [h]	Inference time [ms]
Auto-Classifier	ResNet50	1.401	17.48
	Fusion	15.528	199.96
	AutoML	2.433	10.65
Skip-GANomaly	–	4.935	150.39
PaDiM	ResNet18	0.211	57.89
	ResNet50	0.236	111.98
	WideResNet50	0.241	119.96
	EfficientNetB5	0.248	103.62
PatchCore	WideResNet50	44.491	1818.52