Quantized FCA: Efficient Zero-Shot Texture Anomaly Detection

SUPPLEMENTARY MATERIAL

Andrei-Timotei Ardelean and Patrick Rückbeil and Tim Weyrich
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

S1. Summary

In this supplementary material, we include additional visualizations, comparisons, evaluations, and details useful for the reproducibility of our results.

S2. Detailed quantitative results

In Table 4 we include a detailed breakdown of our scores for each texture class in the evaluated datasets.

As mentioned in the main text, the image-level AUROC$_c$ is saturated on the MVTec AD textures. Since the image-level anomaly score is computed as the maximum across the image, the result is quite sensitive to the final smoothing of the results. Table 6 reports the AUROC$_c$ metric for QFCA and QFCA+ at different smoothing levels σ_s. In the best case (QFCA+ with $\sigma_s = 2.0$) the metric is very close to 100%. To be exact, the anomalies are perfectly detected in 3 out of 5 textures (grid, leather, and wood).

S3. Evaluation on generic objects

Our method is designed to work on textures or surfaces that are largely stationary. While that is the case, our feature preprocessing in QFCA+ significantly improves the performance on images that are just partly texture-like. Table 5 includes an evaluation on all objects in MVTec in a comparison with FCA. For some classes (such as transistor) the difference between QFCA+ and FCA is rather small. However, objects that can be characterized as two intertwined textures (such as screw and toothbrush) see large improvements since they benefit most from our PCA-based preprocessing. Moreover, as seen in Tab. 7, despite being designed for textures and using limited pretraining (WideResnet on ImageNet), QFCA+ outperforms WinCLIP [JZK*23] and is comparable to more recent VLM-based methods.

S4. Proof of algorithm correctness

In this section, we prove the correctness of the algorithm introduced for computing the mismatch score between a patch and a reference histogram (Alg. 1). In this context, correctness means that the algorithm is equivalent to the feature correspondence mismatch from FCA, *i.e.* $M(x,y;P)$ in [AW24b]. Specifically, the computed errors for each bin match the errors obtained using the FCA algorithm if it were run on the quantized values.

For simplicity, we analyze the algorithm for a single patch, assume integer weights, and use the same notation as in Alg. 1. Additionally, let $\{X_i\}_{i=1}^{T^2}, \{Y_i\}_{i=1}^{T^2}$ be the feature values of the patch and the reference, respectively. FCA computes the mismatch score by sorting the X and Y vectors and mapping elements of the same rank. The error associated with each element of a patch is given by the absolute difference to its matched value; *i.e.*, $\texttt{FCA}_k := |X_{O_k} - Y_{L_k}|$, where O and L are the indices of the sorted order of X and Y. For the case where the histogram weight of a bin is one, *i.e.* $P_i = 1$, it is easy to see that $E_i = |Q_i - Q_j|$, as per lines 6 or 10 of Alg. 1. Therefore, the error E_i corresponds to $\texttt{FCA}_k$ if and only if i and j correspond to the same ordered rank. P and R represent the vectors as histograms, where the bins correspond to quantiles Q, which are inherently in sorted order. It follows that i and j track the ranks in sorted order iff at each iteration the cumulative weight of the processed bins in histogram P matches the cumulative weight in histogram R; that is, for the order statistic k, $\sum_{b=1}^{j-1} R_b < k \leq \sum_{b=1}^{j} R_b$ and $\sum_{b=1}^{i-1} P_b < k \leq \sum_{b=1}^{i} P_b$. Since we process the order statistics for one bin at a time, this translates in our algorithm to $\sum_{b=1}^{j-1} R_b < \sum_{b=1}^{i} P_b \leq \sum_{b=1}^{j} R_b$ if the algorithm enters the condition on line 5 and $\sum_{b=1}^{i-1} P_b < \sum_{b=1}^{j} R_b \leq \sum_{b=1}^{i} P_b$ otherwise.

It can be verified by induction that these identities hold. At each iteration, the cumulative sum for both histograms is adjusted by the same amount: the index of the smaller bin is moved forward and the discarded weight is taken from the larger bin to mark the transport. Since the weights are nonnegative, the new cumulative sum is bounded by the interval defined by the bins of the other histogram. This remark is sufficient for the case where $P_i = 1$; however, in practice the elements will not be unique due to quantization. When the vectors X and Y have duplicate values, the FCA mismatch score for these elements is ill-defined, since different sorted orders are possible. Our QFCA algorithm alleviates this issue by defining the mismatch score of a bin as the weighted average of the scores pertaining to a bin. The multiplication with P_i on line 6 (and R_j on line 10, respectively) is balanced by the division by $\hat{P}_i$ on line 15 to create this weighted average. Intuitively, in the context of FCA, this translates to computing the mismatch scores in sorted order as before, and then averaging the errors of elements with the same value (because they are arbitrarily mapped to different values in the reference).

We have thus shown that the histogram-based algorithm yields the same mismatch scores as the standard FCA used on quantized values. Note that this also implies that the quantized algorithm exactly converges to the patch statistics comparison of FCA as the number of bins goes to infinity.

2-Wasserstein distance We provide here an additional mathematical justification for the efficacy of FCA and QFCA for localizing anomalies. Ardelean and Weyrich [AW24b] present the patch com-

	QFCA				QFCA+			
MVTec AD	PRO	AUROC$_s$	F_1	AUROC$_c$	PRO	AUROC$_s$	F_1	AUROC$_c$
carpet	95.54	98.38	72.43	99.64	96.72	98.77	75.75	99.48
grid	98.16	99.50	62.29	99.84	98.87	99.67	67.59	100.00
leather	98.89	99.44	65.87	99.63	99.03	99.47	65.26	99.52
tile	95.83	98.00	82.20	99.50	95.88	98.02	81.12	98.19
wood	97.24	98.27	76.59	99.30	97.37	98.22	75.65	97.63
DTD	PRO	AUROC$_s$	F_1	AUROC$_c$	PRO	AUROC$_s$	F_1	AUROC$_c$
Blotchy_099	97.21	99.42	76.21	99.63	97.22	99.13	68.82	99.94
Marbled_078	97.17	99.05	73.45	100.00	97.32	99.00	71.49	100.00
Mesh_114	94.08	97.63	63.85	95.80	96.83	98.60	66.23	97.74
Stratified_154	98.78	99.13	64.87	100.00	98.83	99.10	63.10	100.00
Woven_068	96.98	98.79	68.70	99.88	97.57	99.02	70.85	100.00
Woven_125	98.08	99.37	74.85	100.00	98.23	99.38	74.79	100.00
Fibrous_183	97.06	99.01	71.02	97.75	97.40	99.04	68.99	99.56
Matted_069	89.80	99.41	74.65	100.00	90.53	99.56	74.67	100.00
Perforated_037	94.88	96.82	66.00	98.88	96.75	97.99	67.93	99.94
Woven_001	96.70	99.30	65.04	96.00	98.57	99.69	67.53	98.38
Woven_104	90.85	97.25	66.65	95.00	93.92	98.10	68.46	98.94
Woven_127	87.23	92.55	62.31	97.11	94.90	96.05	70.60	98.11
WFT	PRO	AUROC$_s$	F_1	AUROC$_c$	PRO	AUROC$_s$	F_1	AUROC$_c$
texture_1	92.12	97.91	79.81	–	94.25	98.06	79.68	–
texture_2	86.83	98.57	77.92	–	91.73	98.95	79.73	–

Table 4: *Per-texture detailed results of our method (QFCA and QFCA+).*

	QFCA+				FCA			
MVTec AD	PRO	AUROC$_s$	F_1	AUROC$_c$	PRO	AUROC$_s$	F_1	AUROC$_c$
bottle	**44.58**	**56.53**	**21.27**	**47.62**	26.53	42.46	17.36	28.73
capsule	**81.57**	**87.64**	**13.39**	**49.30**	70.38	85.34	10.48	35.14
grid	**98.87**	**99.67**	**67.59**	**100.00**	98.07	99.46	61.62	99.84
leather	**99.03**	**99.47**	65.26	99.52	98.90	**99.45**	**66.06**	**99.63**
metal_nut	**39.47**	55.63	35.80	**67.89**	26.38	**61.00**	**36.41**	59.58
tile	**95.88**	**98.02**	81.12	98.19	95.41	97.84	**81.70**	**99.36**
transistor	**40.95**	59.72	16.65	**34.92**	39.13	**62.35**	**17.63**	30.42
zipper	**50.55**	**85.61**	**17.09**	**81.20**	42.17	82.60	16.21	43.96
cable	26.64	70.02	14.23	**49.16**	**32.84**	**75.22**	**26.88**	45.46
carpet	**96.72**	**98.77**	**75.75**	99.48	95.44	98.31	72.58	**99.60**
hazelnut	**93.72**	**92.67**	**56.71**	**97.82**	91.08	91.78	50.31	95.00
pill	**78.14**	77.02	22.17	**52.48**	76.08	**80.84**	**25.62**	45.39
screw	**86.37**	**96.08**	**16.39**	50.46	77.85	93.39	5.94	**53.93**
toothbrush	**82.88**	**92.09**	**31.29**	**91.11**	64.26	87.20	16.22	73.06
wood	**97.37**	**98.22**	75.65	97.63	97.18	**98.22**	**76.34**	**99.30**
Average	**74.18**	**84.48**	**40.69**	**74.45**	68.78	83.70	38.76	67.23

Table 5: *Per-class results of our QFCA+ compared to FCA on all objects in the MVTec AD dataset.*

	σ_s				
Method	1.2	1.4	1.6	1.8	2.0
QFCA	99.83	99.83	99.82	99.82	99.80
QFCA+	99.38	99.70	99.77	99.84	**99.89**

Table 6: *Average image-level AUROC$_c$ on MVTec AD textures at different levels of smoothing.*

Method	PRO	AUROC$_s$	F_1
WinCLIP [JZK*23]	64.6	85.1	31.7
April-GAN [CHZ23]	44.0	87.6	43.3
AnoVL [DZBL23]	77.8	90.6	36.5
SDP [CZT*24]	79.1	88.7	35.3
SDP+ [CZT*24]	85.6	91.2	41.9
SAA [CXS*23]	31.9	67.7	23.8
SAA+ [CXS*23]	42.8	73.2	37.8
ClipSAM [LCY*25]	88.3	92.3	47.8
QFCA+ (Ours)	74.2	84.5	40.7

Table 7: *Comparison to VLM-based methods on all objects in MVTec AD; metrics taken from [LCY*25]. While our method is specific to textures, it is on par with most VLM-based methods that make use of significantly more pretraining as well as textual clues.*

parison algorithm as an ad-hoc trick to obtain the contribution of each element to the 1-Wasserstein distance. That being said, we show here that using the absolute value between matching elements in sorted order is not arbitrary, but rather it represents the magnitude of the gradient of the squared 2-Wasserstein distance between the two distributions.

The Wasserstein distance for distributions with dimensionality $d = 1$ has the analytic solution:

$$W_p(X,Y) = \left(\int_0^1 |F^{-1}(z) - G^{-1}(z)|^p dz \right)^{1/p}, \tag{3}$$

where F and G are the cumulative distributions of X and Y, respectively. For our empirical distributions this can be written as

$$W_p(X,Y) = \left(\sum_{k=0}^{T^2} |X_{O_k} - Y_{L_k}|^p \right)^{1/p}. \tag{4}$$

The squared 2-Wasserstein distance is then simply

$$W_2^2(X,Y) = \sum_{k=0}^{T^2} (X_{O_k} - Y_{L_k})^2, \tag{5}$$

so the magnitude of the gradient of an element X_{O_k} is given by

$$\left| \frac{\partial W_2^2}{\partial X_{O_k}} \right| = 2|X_{O_k} - Y_{L_k}| = 2 \cdot \mathtt{FCA}_k. \tag{6}$$

Since multiplying all scores by a constant factor does not change the anomaly localization, it follows that QFCA can be seen as efficiently computing the gradient of the squared 2-Wasserstein distance between all patches and the global reference.

S5. Additional Details

All time measurements of our method as well as the baselines of FCA [AW24b], GRNR [YLC*24], ZvM [ATO23], and April-GAN [CHZ23] were performed using an NVIDIA RTX A5000 GPU. For the other baselines we use the runtime reported by the authors.

For our local average pooling experiment in Figure 3 we use up-to-date versions of the libraries. That is: Tensorflow version 2.18.0, Pytorch version 2.6.0, and Jax version 0.5.2 .

S6. Code

The code is available at: github.com/TArdelean/QuantizedFCA.