Chebyshev Attention Depth Permutation Texture Network with Latent Texture Attribute Loss (Supplementary Material)

Ravishankar Evani, Deepu Rajan, Shangbo Mao
College of Computing and Data Science, Nanyang Technological University, Singapore
S220007@e.ntu.edu.sg, ASDRajan@ntu.edu.sg, MAOS0003@e.ntu.edu.sg

1. Acronyms Descriptions

Table 1. Descriptions of Acronyms used in the Paper

Acronym	Description
CAPTN	Chebyshev Attention Depth Permutation Texture Network
LCP	Learnable Chebyshev Polynomial
SLAR	Spatial Latent Attribute Representation
LTA	Latent Texture Attribute
SLTM	Stochastic Local Texture Masking
TFA	Texture Frequency Attention
D^2P	Dual Depth Permutation
OL	Orderless Latent Texture Attribute
SL	Spatial Latent Texture Attribute
FC	Fully Connected Layer

2. Hyperparameter Settings and Experimental Details

Tab. 2 presents the hyperparameters used for each dataset. A batch size of 32 (training and testing) and dropout of 0.9 (training) is used consistently across all datasets. The weight decay parameter is for the Adam optimizer with decoupled weight decay [3].

Table 2. Training hyperparameters for each dataset

Dataset	Epochs	Weight Decay	LCP Degree θ
DTD	300	0.5	2
FMD	100	0.01	1
KTH	30	0.5	1
GTOS	30	0.01	2
GTOS-M	30	0.01	3

Data Augmentation. All datasets undergo common augmentations including *Random Horizontal and Vertical Flip*, *Random Auto-contrast*, and *Color Jitter* to enhance variability and robustness. Additionally, the DTD dataset employs *Resize + Random Resized Crop*, while FMD applies *Resize + Center Crop* along with *Random Equalize*. KTH-TIPS2-b, GTOS, and GTOS-M undergo *Resize + Center Crop*, *Random Rotation of 5°*, and *Random Equalize*, in addition to the common transformations. For the computation of Latent Texture Attribute (LTA) loss, *Stochastic Local Texture Masking (SLTM)* is applied using a mask size of 16×16 pixels, resulting in extracted image patches of the same dimensions.

2.1. Loss Function

The weights associated with each term in the Latent Texture Attribute (LTA) loss function—λ, μ, and ν—were set to 1, assigning equal importance to all constituent loss components. Label smoothing [5] of 0.5 was applied to the cross-entropy component of the LTA loss when training on the GTOS dataset. For all other datasets, label smoothing was set to 0.

2.2. Effect of Chebyshev Polynomial Degree and Crop Location

We investigate the impact of crop location in the Spatial Latent Attribute Representation (SLAR) along with different degrees of the Learnable Chebyshev Polynomial (LCP), denoted as θ. A crop of size 5×5 is used. By default (when no spatial offset δ is applied), the crop spans from pixel location $(2, 2)$ (bottom-left) to $(6, 6)$ (top-right). To study the sensitivity to spatial shifts, an offset of $\delta = -1$ is introduced, shifting the crop one pixel diagonally (along both spatial dimensions).

Table 3. Impact of LCP degree θ and SLAR cropping offset δ on classification accuracy (%). ConvNeXt-T is used as the backbone.

θ	δ	DTD	KTH	GTOS	GTOS-M
1	0	78.6 $\pm$1.0	**92.2** $\pm$4.4	**85.1** $\pm$1.4	93.0
1	-1	78.6 $\pm$0.9	92.1 $\pm$4.4	**85.1** $\pm$1.5	93.0
2	0	**78.9** $\pm$0.8	91.8 $\pm$4.5	**85.1** $\pm$1.7	93.5
2	-1	78.8 $\pm$0.8	**91.9** $\pm$4.5	**85.1** $\pm$1.7	**93.7**
3	0	77.8 $\pm$1.0	90.9 $\pm$5.5	83.8 $\pm$2.4	**94.2**
3	-1	**77.9** $\pm$1.0	**91.0** $\pm$5.5	**83.9** $\pm$2.4	**94.2**

Tab. 3 demonstrates the robustness of CAPTN to variations in cropping location from the concatenated representation of enhanced latent texture attributes, $\mathbf{X}_e^{M-1}$, and the feature maps from the backbone's final layer, $\mathbf{X}^{o_{M-1}}$, used to compute the Latent Texture Attribute loss. This robustness suggests that CAPTN effectively learns and transfers spatial context from the surrounding texture to the masked or cropped regions, regardless of the crop's precise location. An offset of $\delta = 0$ was used for all datasets, except for GTOS-M where $\delta = -1$. We also evaluated the optimal degree θ of the learnable Chebyshev polynomial used to represent the orderless latent texture attributes with a ConvNeXt-T backbone. On the KTH dataset, CAPTN achieved optimal performance with $\theta = 1$. For GTOS, degrees 1 and 2 yielded comparable performance. However, when using a ConvNeXt-N backbone, degree 2 (83.9 ± 1.7) outperformed degree 1 (83.5 ± 1.7). On DTD, CAPTN performed best with $\theta = 2$, and on GTOS-M, performance peaked at $\theta = 3$.

The polynomial order θ is closely tied to the complexity of texture and material characteristics in each dataset. Orderless aggregation inherently discards spatial information. To recover some of this, we model the representations as learnable Chebyshev polynomials, introducing a controllable degree of non-linearity. In datasets like KTH—captured in controlled environments with minimal intra-class variation—a low-degree polynomial ($\theta = 1$) is sufficient. In contrast, DTD contains perceptually similar categories (e.g., knitted vs. woven), demanding higher polynomial complexity. GTOS and GTOS-M, comprising terrain images, show low inter-class variation, further benefiting from higher polynomial degrees to capture nuanced material information.

2.3. Effect of Mask Size

Tab. 4 presents the classification performance of CAPTN when varying the mask size applied in the Stochastic Local Texture Masking (SLTM) module. A mask size of 16×16 pixels consistently yields strong results across multiple texture and material datasets. If the mask is too large, there is insufficient surrounding context for CAPTN to learn meaningful spatial dependencies. Conversely, if the mask is too small, the patch extracted from the image may lack enough structural content to produce a useful comparison with the SLAR features, making the Spatial Latent Texture Attribute Loss less informative.

Table 4. Impact of mask size on classification accuracy (%). ConvNeXt-N is used as the backbone.

Mask Size	DTD	KTH	GTOS	GTOS-M
8	77.8 $\pm$1.0	**90.6** $\pm$4.8	**83.9** $\pm$1.7	91.1
16	**77.9** $\pm$1.0	**90.6** $\pm$4.6	**83.9** $\pm$1.7	**91.6**
32	**77.9** $\pm$1.0	90.5 $\pm$4.7	**83.9** $\pm$1.7	90.7
64	77.7 $\pm$0.9	90.3 $\pm$4.4	**83.9** $\pm$1.7	89.8

3. Benefits of Learnable Chebyshev Polynomial

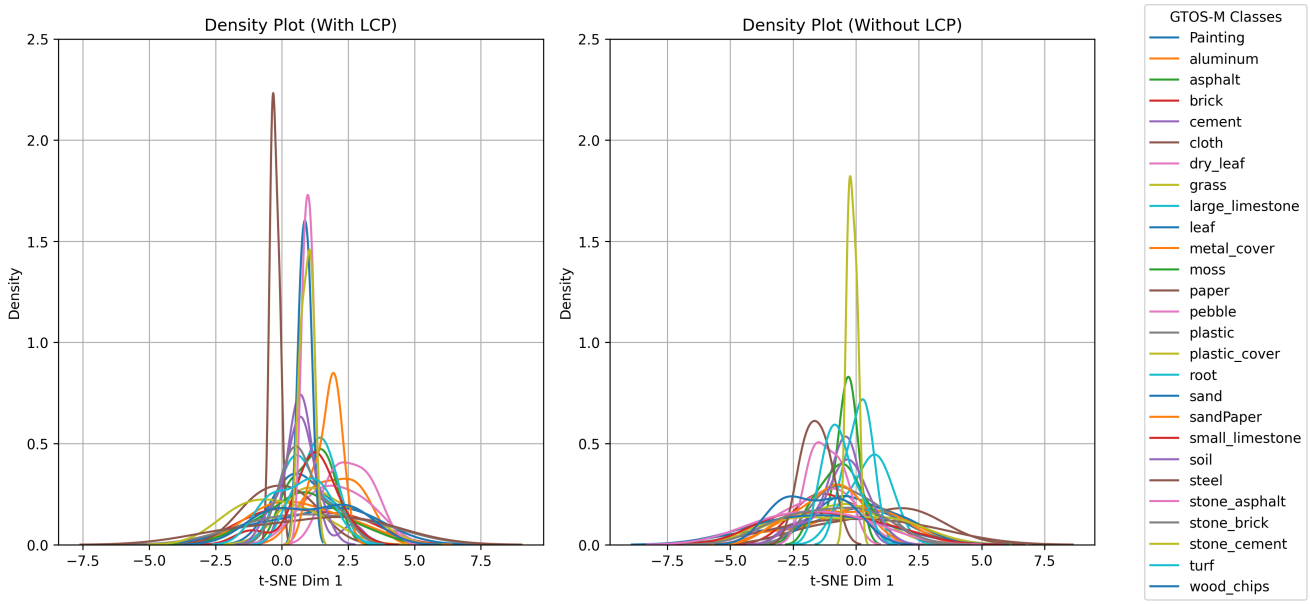
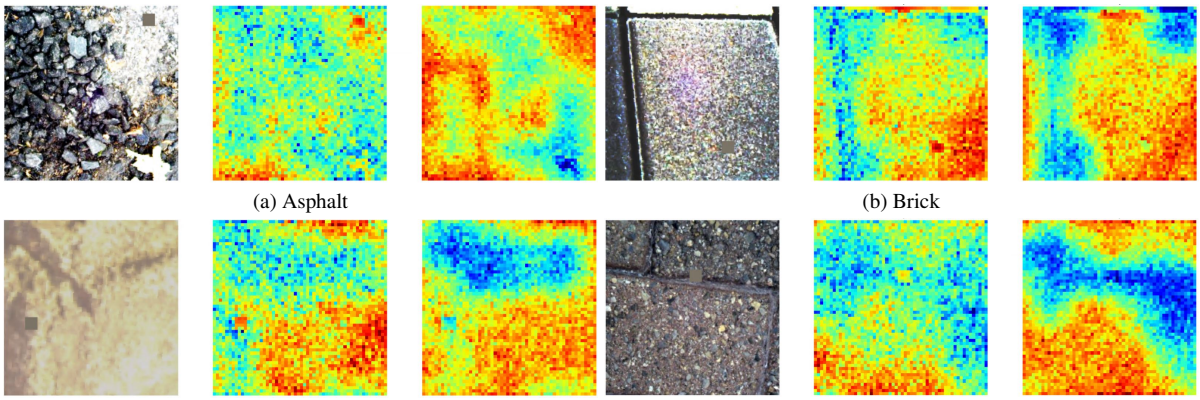


Figure 1. Density plots comparing embeddings with and without LCP.

We use the GTOS-M dataset with ConvNeXt-T as the backbone to generate the results shown in Fig. 1. This figure illustrates the distribution of class-specific embeddings along the first t-SNE dimension, comparing representations learned with and without the Learnable Chebyshev Polynomial (LCP). The Kernel Density Estimation (KDE) plots are derived from the first dimension of the t-SNE embeddings and visualize how embeddings for each class are distributed. The density plot *with LCP* exhibits more pronounced and numerous peaks compared to the plot *without LCP*. This indicates that LCP enhances intra-class compactness by clustering embeddings of the same class more tightly, resulting in sharper distributions. Moreover, the increased number of distinct peaks suggests that LCP captures finer-grained patterns or substructures within classes, making the feature representations more discriminative. These improvements likely stem from LCP's capacity to model higher-order relationships while suppressing noise in the embedding space. Overall, LCP provides a more robust representation, enhancing both compactness and separability. As such, this orderless representation using a 3^{rd}-order LCP improves the mapping from latent features to visual texture attributes.



(a) Asphalt (b) Brick

(c) Large limestone (d) Stone brick

Figure 2. Visualization from CAPTN with ConvNeXt-T backbone. Left: GTOS-M image with mask. Middle: Attention map from the TFA module without LCP. Right: Attention map from the TFA module with LCP (3^{rd} order) enabled.

Table 5. Number of trainable and non-trainable parameters (in millions) for CAPTN. GTOS-M Dataset.

Parameter Type	ConvNeXt-N	ConvNeXt-T	ConvNeXt-B	ConvNeXt-L
Trainable (CAPTN with 3^{rd} order LCP)	1.72	2.45	4.31	9.60
Trainable (CAPTN with RBF)	2.88	3.98	6.69	14.22
Non-trainable	14.95	27.82	87.56	196.23

Figure 2 further demonstrates how the LCP influences the attention mechanism during backpropagation. Specifically, it enhances the representational power of the upstream Texture Frequency Attention (TFA) module by modulating the gradient flow. The resulting spatial attention maps are more uniformly distributed over relevant regions of the material surface. When LCP is activated, the CAPTN module learns to downweight masked or irrelevant regions, as indicated by reduced attention intensity in those areas. This highlights LCP's effectiveness in guiding attention toward semantically meaningful regions, thereby improving spatial focus and interpretability.

4. Computation Time to Generate Texture Embeddings

Understanding the computational efficiency of texture recognition methods is critical for selecting models suitable for real-time or resource-constrained environments. By measuring the inference time required to generate texture embeddings, we aim to evaluate the practicality of **RADAM**, **GTN**, and **CAPTN** across different backbone architectures. Since the RADAM module and the classifier (e.g., Support Vector Machines, k-Nearest Neighbors) are separate, we removed the final classification layers from GTN and CAPTN to ensure a fair comparison. The timing measurements focus solely on the time taken to process an image and generate the final texture embeddings, before they are then fed into the classification layer or model. All experiments were conducted on a single NVIDIA RTX A6000 GPU. To mitigate initialization overheads, the GPU was warmed up, and a single RGB image of size 224×224 was used as input for each model. Before measuring inference time, a warm-up phase was performed by running 10 forward passes through each model. This step ensures that initialization overheads, such as GPU memory allocation and CUDA kernel compilation, are excluded from the timing results. After warming up, inference timing was measured by synchronizing the GPU to ensure that all pending operations were completed. Each model performed 100 forward passes in a loop, with each iteration representing a single inference. The end time was recorded after the loop, and GPU synchronization before and after the timing ensured accurate measurements, as CUDA operations are asynchronous by default. Finally, the average inference time per forward pass was computed. Fig. 3 presents the average time required for a forward pass to generate texture embeddings (measured in milliseconds) for **RADAM**, **GTN**, and **CAPTN**, evaluated across different backbones. CAPTN has the lowest and stable average time for forward pass. Hence, in addition to achieving state-of-the-art performance for most of the challenging benchmark texture and material datasets, CAPTN has also shown to be more efficient.

5. Further Analysis of Quantitative Results

We provide a deeper analysis of the results presented in Table 1 of our paper. CAPTN demonstrates strong performance on the DTD, KTH, GTOS, and GTOS-M datasets. However, RADAM [4] generally outperforms CAPTN on the FMD dataset. This can be attributed to the nature of the FMD dataset, which contains a higher proportion of object-level and shape-related information compared to the other datasets. RADAM leverages positional encoding that can effectively incorporate spatial structural information, enabling it to better capture shape-centric features. In contrast, CAPTN is specifically designed to generate and enhance latent texture attribute information, which is essential for texture and material representation. It is not explicitly optimized for modeling shape-related information. Furthermore, the Latent Texture Attribute Loss in CAPTN is crafted to reinforce texture representation learning rather than emphasize shape characteristics. Despite this, CAPTN generally surpasses GTN [2] on the FMD dataset. This improvement can be partially attributed to the Stochastic Local Texture Masking (SLTM) mechanism, which allows CAPTN to balance global and contextual features with fine-grained local details. Additionally, unlike GTN, CAPTN benefits more from scaling up the backbone architecture. When using a larger backbone such as ConvNeXt-L, CAPTN achieves performance on par with RADAM.

6. Reproducibility

We have re-run some experiments to ensure their reproducibility by other researchers in the future. Some differences in decimal values were observed, as shown in Tab. 6.

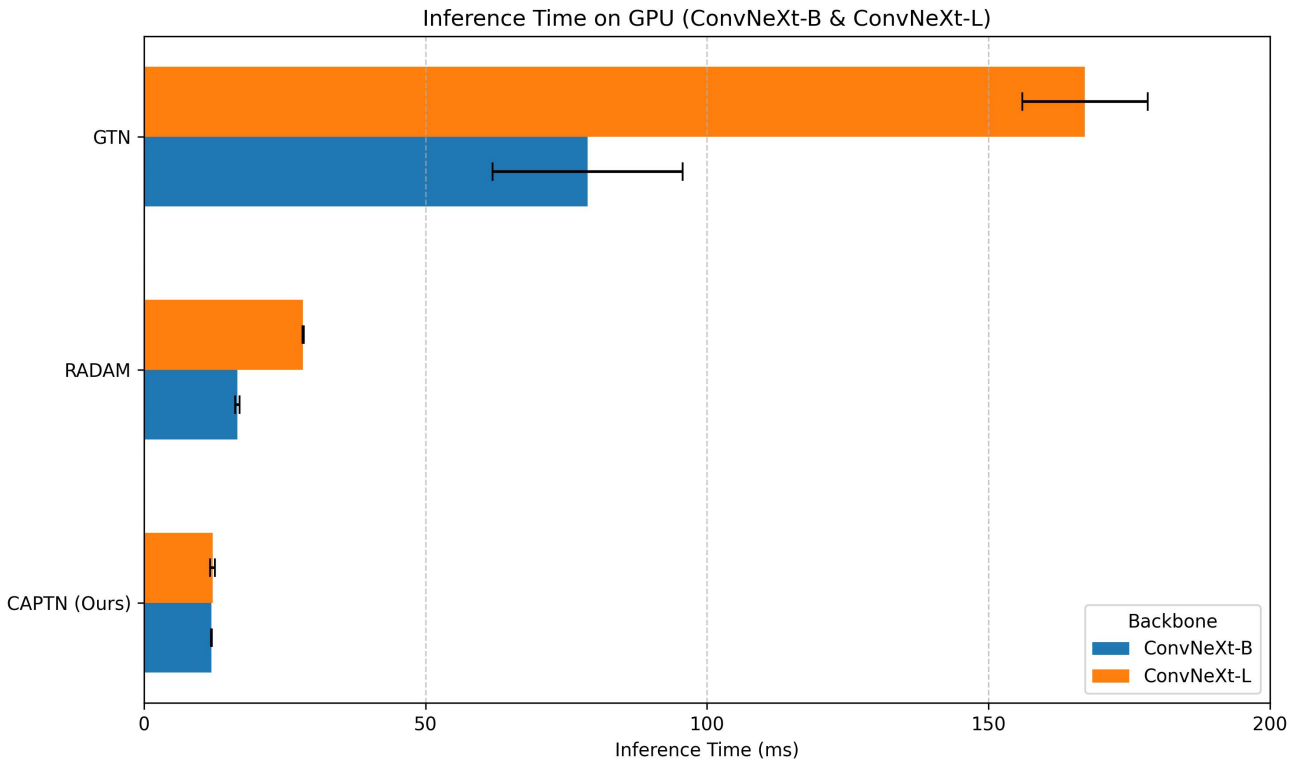


Figure 3. Average Time (ms) for forward pass for different Models and backbones

Table 6. GTOS-M Classification Accuracy (%) for different models and backbones.

Backbone	RADAM	GTN	CAPTN (ours)
ConvNeXt-N	81.8	90.1	**91.6**
ConvNeXt-T	85.3	91.9	**94.2**
ConvNeXt-B	82.2	**94.8**	94.0
ConvNeXt-L	85.8	93.9	**94.9**

Table 7. Comparison of rescaled $[-1, 1]$ vs. unbounded $(-\infty, \infty)$ domains for LCP.

Domain	DTD	GTOS
$[-1, 1]$	77.5 ± 1.0	83.8 ± 1.8
$(-\infty, \infty)$	$\mathbf{77.9} \pm 1.0$	$\mathbf{83.9} \pm 1.7$

7. Impact of Rescaling on LCP Transformation

Motivated by [1], we explored the operational domain for LCP. Rescaling inputs to the Chebyshev orthogonal domain $[-1, 1]$ may suppress texture-sensitive features that are critical for learning discriminative representations. Instead, by skipping this rescaling step and directly applying the LCP transformation to raw LTAs, the model preserves essential information about activation magnitudes and inter-channel dependencies. This allows the network to exploit a broader and more expressive range of polynomial outputs. The benefit is particularly pronounced with higher-degree Chebyshev terms, which exhibit greater variation when evaluated on unbounded input domains. As shown in Table 7, this approach yields improved performance for texture and material recognition.

References

[1] Zeyuan Allen-Zhu and Yuanzhi Li. Faster principal component regression and stable matrix chebyshev approximation. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia*, pages 107–115. PMLR, 2017. 5

[2] Ravishankar Evani, Deepu Rajan, and Shangbo Mao. Multiscale graph texture network. In *Computer Vision – ECCV 2024*, pages 218–235, Cham, 2025. Springer Nature Switzerland. 4

[3] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA*, 2019. 1

[4] Leonardo Scabini, Kallil M. Zielinski, Lucas C. Ribas, Wesley N. Gonçalves, Bernard De Baets, and Odemir M. Bruno. Radam: Texture recognition through randomized aggregated encoding of deep activation maps. *Pattern Recognition*, 143:109802, 2023. 4

[5] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 2