

ANALYSIS AND APPLICATIONS OF DEEP  
LEARNING FEATURES ON VISUAL TASKS

ANALYSIS AND APPLICATIONS OF DEEP LEARNING  
FEATURES ON VISUAL TASKS

BY  
KANGDI SHI, B.A.SC

A THESIS  
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING  
AND THE SCHOOL OF GRADUATE STUDIES  
OF MCMASTER UNIVERSITY  
IN PARTIAL FULFILMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF  
DOCTOR OF PHILOSOPHY

© Copyright by Kangdi Shi, September 2022

All Rights Reserved

Doctor of Philosophy (2022)  
(Electrical and Computer Engineering)

McMaster University  
Hamilton, Ontario

TITLE: Analysis and Applications of Deep Learning Features on  
Visual Tasks

AUTHOR: Kangdi Shi  
B.A.Sc (Electrical and Computer Engineering),  
University of Manitoba, Winnipeg, MB, Canada

SUPERVISOR: Dr. Jun Chen

NUMBER OF PAGES: cxv, 115

# Abstract

Benefiting from hardware development, deep learning (DL) has become a popular research area in recent decades. Convolutional neural network (CNN) is a critical deep learning tool that has been utilized in many computer vision problems. Moreover, the data-driven approach has unleashed CNN's potential in acquiring impressive learning ability with minimum human supervision. Therefore, many computer vision problems are brought into the spotlight again. In this thesis, we investigate the application of deep-learning-based methods, particularly the role of deep learning features, in two representative visual tasks: image retrieval and image inpainting.

Image retrieval aims to find in a dataset images similar to a query image. In the proposed image retrieval method, we use canonical correlation analysis to explore the relationship between matching and non-matching features from pre-trained CNN, and generate compact transformed features. The level of similarity between two images is determined by a hypothesis test regarding the joint distribution of transformed image feature pairs. The proposed approach is benchmarked against three popular statistical analysis methods, Linear Discriminant Analysis (LDA), Principal Component Analysis with whitening (PCAw), and Supervised Principal Component Analysis (SPCA). Our approach is shown to achieve competitive retrieval performances on Oxford5k, Paris6k,



$\mathcal{R}$ Oxford, and  $\mathcal{R}$ Paris datasets.

Moreover, an image inpainting framework is proposed to reconstruct the corrupted region in an image progressively. Specifically, we design a feature extraction network inspired by Gaussian and Laplacian pyramid, which is usually used to decompose the image into different frequency components. Furthermore, we use a two-branch iterative inpainting network to progressively recover the corrupted region on high and low-frequency features respectively and fuse both high and low-frequency features from each iteration. Moreover, an enhancement model is introduced to employ neighbouring iterations' features to further improve intermediate iterations' features. The proposed network is evaluated on popular image inpainting datasets such as Paris Streetview, Celeba, and Place2. Extensive experiments prove the validity of the proposed method in this thesis, and demonstrate the competitive performance against the state-of-the-art.

*To all love I have received during this greatest journey in my life.*

# Acknowledgements

Firstly, I would like to express sincere gratitude to my supervisor Dr. Jun Chen. Without him, I cannot accomplish any of the presented works in this thesis. His knowledgeable guidance and kind character have become a beacon in my academic journey. He is not only the best professor I have ever met but also a role model I will always look up to in my life.

To my committee member, I would like to thank Dr. Sorina Dumitrescu, Dr. Ratnasingham Tharmarasa and Dr. Jian-Kang Zhang who have provided me with academic suggestions for my work all these years. Sadly, Dr. Jian-Kang Zhang passed away last year, but I always remember his kindness during my committee meetings.

To my friends who are beside me during this journey, I would like to thank Xiaohong, Huan, Zhihao, Siyao, Jingjing, Wei, and Liangyan who support me academically and emotionally. I think our friendship is precious because it has accompanied us through a meaningful time.

Last but not least, I would like to thank my family for their continuous love and encouragement. Their unconditional and endless support help me survive the most difficult period while completing this thesis.

# Contents

<b>Abstract</b>	<b>ii</b>
<b>Acknowledgements</b>	<b>v</b>
<b>Abbreviations</b>	<b>x</b>
<b>1 Introductions</b>	<b>1</b>
1.1 Feature Engineering in Machine Learning . . . . .	1
1.2 Deep Learning-Based Computer Vision . . . . .	3
1.3 Feature Extraction for Deep Learning-Based Computer Vision . . . . .	6
1.4 Feature Transformation for Deep Learning-Based Computer Vision . . . . .	7
1.5 Contributions and Thesis Organization . . . . .	10
<b>2 Image Retrieval via Canonical Correlation Analysis and Binary Hypothesis</b>	
<b>Testing</b>	<b>21</b>
2.1 Abstract . . . . .	21
2.2 Introduction . . . . .	22
2.3 Proposed Method . . . . .	25
2.4 Experimental Results . . . . .	38
2.5 Conclusions . . . . .	57

2.A	Chernoff Information Between Two 2-Dimensional Gaussian Distributions .	58
<b>3</b>	<b>Progressive with Purpose: Guiding Progressive Inpainting DNNs through Context and Structure</b>	<b>70</b>
3.1	Abstract . . . . .	70
3.2	Introduction . . . . .	71
3.3	Related Work . . . . .	74
3.4	Proposed Inpainting Algorithm . . . . .	78
3.5	Experimental Setup . . . . .	91
3.6	Evaluation Results . . . . .	98
3.7	Conclusion . . . . .	105
<b>4</b>	<b>Conclusion and Future Work</b>	<b>112</b>
4.1	Conclusion . . . . .	112
4.2	Future Work . . . . .	113

# List of Figures

2.1	Modified VGG16 for feature extraction. . . . .	26
2.2	Block diagram of the proposed method. . . . .	28
2.3	Profile of the diagonal elements of $\Lambda$ and $\Pi$ (i.e., $c_t^{(M)}$ and $c_t^{(N)}$ , where $t \in \{1, 2, \dots, 512\}$ ) using AVE features. The CCA training was performed on the 120k-SfM dataset. . . . .	44
2.4	Profile of sorted diagonal elements of $\Lambda$ and $\Pi$ (i.e., $\tilde{c}_t^{(M)}$ and $\tilde{c}_t^{(N)}$ , where $t \in \{1, 2, \dots, 512\}$ ) using AVE features. The CCA training was performed on the 120k-SfM dataset. . . . .	45
2.5	2D visualization of matrix $\Pi$ . . . . .	48
2.6	3D visualization of matrix $\Pi$ . . . . .	48
3.7	A graphical description of the proposed solution. It shows all three stages and detail their main components and elements. . . . .	81
3.8	Visual results on Paris StreetView. . . . .	95
3.9	Visual results on CelebA. . . . .	96
3.10	Visual results on Place2. . . . .	97

# List of Tables

2.1	Examples of matching/non-matching pairs. . . . .	40
2.2	Performance comparison of the baseline, S-CCA, and G-CCA on Oxford5k, $\mathcal{R}$ Oxford, Paris6k, and $\mathcal{R}$ Paris without dimension reduction. . .	49
2.3	Evaluation results from 30k-SfM on Oxford5k, $\mathcal{R}$ Oxford, Paris6k, and $\mathcal{R}$ Paris. . . . .	52
2.4	Evaluation results from 120k-SfM on Oxford5k, $\mathcal{R}$ Oxford, Paris6k, and $\mathcal{R}$ Paris. . . . .	53
2.5	Image retrieval comparison of PCAw, SPCA, and G-CCA. . . . .	54
2.6	Image retrieval comparison of PCAw, SPCA, and G-CCA. . . . .	55
2.7	Image retrieval comparison of PCAw, MLDA, and G-CCA. . . . .	56
3.8	Numerical comparisons on three datasets. . . . .	93
3.9	Ablation study results on the Paris Streetview Dataset based on network structure . . . . .	100
3.10	Ablation study results on the Paris Streetview Dataset based on loss function	101

# Abbreviations

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
CV	Computer Vision
GT	Ground Truth
DNN	Deep Neural Network
CNN	Convolution Neutral Network
PSNR	Peak Signal-to-Noise Ratio
SSIM	Structural Similarity Index Measure
mAP	mean Average Precision
SOTA	State of the Art



mAP	mean Average Precision
AP	Average Precision
SOTA	State of the Art
CCA	Canonical Correlation Analysis
PCA	Principle Correlation Analysis
LDA	Linear Discriminative Analysis
SPCA	Supervised Principle Correlation Analysis
G-CCA	Gaussian similarity - Canonical Correlation Analysis
S-CCA	Scalar similarity - Canonical Correlation Analysis
SIFT	Scale-invariant feature detection
RNN	Recurrent Neutral Network
BN	Batch Normalization
CI	Chernoff Information
SD	Standard Deviation
GLE	Gaussian-Laplacian feature Extractor

$\mathbb{R}$	Real number
GAN	Generative Adversarial Network
CBIR	Content-Based Image Retrieval

# Chapter 1

## Introductions

### 1.1 Feature Engineering in Machine Learning

Feature engineering [42] focuses on extracting useful information from raw data and transforming extracted information to a favourable domain to improve models' learning ability. In traditional machine learning [40], the algorithms' performance largely depends on the training data. However, the collected raw data cannot be directly utilized by the machine learning algorithms due to information redundancy, inconsistent scaling, or other problems existing in features. The above challenging problems are essentially the focus of feature engineering. Therefore, it is clear that feature engineering plays an essential role in the pipeline of machine learning.

To be specific, we illustrate the effectiveness of feature engineering in machine learning from three aspects: (1) Feature cleaning [45] is a process that filters out redundant or harmful information to the performance of the algorithm. In image processing and voice recognition tasks, frequency-based filter is one of the commonly used techniques to remove noise from raw data. In particular, Gaussian kernel-based filter is an image processing

tool to smooth the image and remove noise. Furthermore, image cropping and resizing is another approach to remove irrelative object in image retrieval tasks. Speech recognition tasks generally use noise reduction algorithms to eliminate background noise. Besides, text retrieval tasks do not need a profound understanding of the sentences, and clean data by using stopwords [49] which indicate meaningless words to the content of the sentence, like article. (2) Feature extraction [47] is the first bridge between cleaned data and machine learning algorithms. Because of the original form of cleaned data, the machine learning algorithm can not process them directly. Therefore, the feature extraction plays an important role in connecting cleaned data and machine learning algorithms. Here we illustrate several feature extraction techniques used in deep learning tasks. The One hot coding is a very basic feature extraction process used for categorical variables data which can represent information in text format. Similar to text type of data, the voice recognition task firstly converts audio data into the frequency domain, and maps each vocabulary from each character into unique character labels. In image processing tasks, there are a large number of methods to detect features from images. For instance, Harris Corner Detection [50] uses Gaussian window function to extract features from image corners and edges. Scale-Invariant Feature Detection (SIFT) [33] is a classic feature extraction method to obtain key points which can express robust features from images. More recently, deep CNN is used to extract images features on both low and high dimensions. (3) Feature transformation [47] further maps extracted features into a domain suitable for machine learning algorithms. For example, feature values may vary over a wide range, which causes performance degradation. Feature scaling is a typical feature transformation that normalizes independent features, including standardization, scaling to unit length and more. On the other hand, feature dimension [48] is another variable that

can affect performance. Because of information redundancy, it is often unnecessary to employ features of full dimensions. Otherwise, the performance may get harmed, or the algorithm may become inefficient. Here we simply list three classic dimensional reduction tools used in machine learning Principle Component Analysis (PCA) [39] [31], selecting most significant components based on eigenvalues, is a dimensional reduction tool used for many machine learning algorithms. Linear Discriminant Analysis (LDA) [51] is another popular dimensional reduction based on maximizing the distance between means of the each classes. Canonical Correlation Analysis (CCA) [52] can find a projection pair that reduces the vector dimension and maximizes the correlation between two sets of random variables.

In this thesis, we present several feature extraction and feature transformation methods based on deep learning, and verify the performance of the proposed methods on image processing tasks.

## **1.2 Deep Learning-Based Computer Vision**

Deep learning [38] is a subfield of machine learning that employs techniques founded on artificial neural networks and representation learning. The Convolutional Neural Network (CNN) is the deep learning's most fundamental techniques that largely develop the deep learning-based computer vision in many areas. In the following of this section, we discuss the working process of CNN and its applications in deep learning-based computer vision.

Convolutional Neural Network (CNN) [37] is a very popular and powerful deep learning tool in image processing, and consists of pooling layers, fully connected layers and convolutional layers, which play the main role. The convolutional layers apply

convolution operation with trainable kernels on RGB images. After going through a succession of convolutional layers, the image is converted into a representation of 3D array, so-called features which are classified into low and high-dimensional features. In deep-learning-based computer vision, low dimensional features refer to contextual information, while high dimensional features refer to image structure. The CNN has been used as a feature extractor in many deep learning methods for computer vision tasks like image classification and object detection.

Benefiting from the CNN's feature extraction process with end-to-end training strategy, deep learning-based computer vision has achieved significant progress in many topics. Specifically for image classification, a fundamental task among computer vision, have developed rapidly during the period that CNN initially get explored. Innumerable works [5, 10, 36, 46] emerged, and propose many deep learning network architectures which have profound influence on other related topics. For instance, encoder-like CNN structure is firstly used for image classification's feature extraction process, and turns out to be effective in feature extraction for almost all deep learning-based computer vision tasks [24] [17] [24] [21]. Next, we go through the remarkable progress in deep learning-based image retrieval and image inpainting.

Nowadays, the deep learning-based image retrieval has made significant contribution for content-based image retrieval (CBIR) which is a very important subfield in image retrieval. Here we summarize the contribution from three aspects: network architectures, feature extraction, and feature enhancement. The notabilities of network architectures mainly reflect on depth and width. The network with more layers [35] have been verified that have better learning ability. The network's width refers to number of channels in kernel. In [22], it shows that widening the residual block can provider better performance

and shorten training time. The innovation about feature extraction focus on three sides: extracting local features from intermediate convolutional layers [20], extracting global features from last several fully-connected layers [19] and fusions of local and global features [12]. The feature enhancement can improve image retrieval's performance and efficiency, and enrich the representative information from extracted feature. For example, some approaches [15] [18] prove that the local features can be embed into global features, and enhance the extracted features. Besides, feature dimension reduction approaches [28] can further refine the extracted features, and improve the image retrieval efficiency.

Besides, deep learning-based image inpainting techniques [14] [16] also develop rapidly under the influence of deep learning. Here we summarize the contribution from two aspects: additional input information, feature fusion. Additional inputs include edge maps, structure maps, and segmentation maps. In some works [11] [10] [11], the proposed approaches use edge and structure maps as inputs which provide extra information, or guide the intermediate output layers with multi-scale segmentation maps. On the other hand, the feature fusion also play a important role in image inpainting. For example, the progressive inpainting works [17] [12] [6] [16] [12] propose several progressive feature fusion network that fuse feature during multi-stages in the inpainting process. The proposed feature fusion framework shows competitive performance against one or two stage inpainting method [24] [10].

In the next two sections, we discuss the importance of feature extraction and feature transformation, and their applications in deep learning-based computer vision tasks.

## 1.3 Feature Extraction for Deep Learning-Based Computer Vision

Most deep learning techniques utilize deep convolutional neural network (CNN) to generate desirable outputs based on back propagation algorithms. This characteristic allows deep CNN to learn the mapping from input to ground truth with minimum supervision, and makes the feature extraction process become trainable along with the network. Especially in deep learning-based computer vision, the feature extraction process can be directly applied on three dimensional image, and generates feature vectors for many computer vision tasks.

Here we briefly mention several types of deep CNN-based feature extraction networks for computer vision tasks: (1) VGG-16 [46] is a famous feature extraction network for image classification. The autoencoder-inspired network structure includes 13 convolutional layers, 5 max-pooling layers and 3 fully connected layers. The sub-network that excludes fully connected layers is usually used as a feature extraction network. (2) MobileNet-v2 [2] is a light-weight feature extraction network that runs on mobile devices for object detection. It employs depthwise convolution and pointwise convolution. In depthwise convolution, one kernel is only used for convolutional operation on one channel of the feature map. Consequently, there are two disadvantages: the channel number of the output feature map is the same as that of the input, and the features at the same location but from different channels do not get integrated effectively. Pointwise convolution solves these problems by using multiple kernels with the size of  $1 \times 1$  so that the features from depthwise convolution are combined via weighted sum. (3) GoogLeNet [1] is one of the most used feature extraction networks for computer vision tasks. It consists of multiple inception blocks



with multiple kernels of different sizes. This enables feature extraction on multiple scales, which provides better performance. Above networks are used in many computer vision tasks like image classification, object detection and video understanding.

Transformer [44] [29] is a new kind of deep learning network, which is built based on self-attention mechanism. Specifically, the transformer commonly includes two parts: encoder and decoder. Unlike the deep CNN, position encoding vector are usually provided to the input of encoder. Therefore, the transformer [53] not only take the image patches as input but also the information about pixels' coordinates in computer vision. Compared with CNN, transformer have a stronger ability to extract features from whole images. This is because the multi-head attention mechanism uses scaled dot-product attention to investigate each pixel's relationship with surrounding pixels in a larger receptive field than deep CNN. The receptive field of CNN is limited by kernel size. However, the transformer is harder to train compared with CNN due to the gradient vanishing issue.

In this thesis, we build all proposed methods based on deep CNN. Specifically, we develop an image retrieval method using features from VGG-16 in Chapter 2, and propose a feature extraction network for image inpainting tasks in Chapter 3.

## **1.4 Feature Transformation for Deep Learning-Based Computer Vision**

In deep learning, feature transformation is another important factor in the performance of deep learning models. Especially for deep CNN, the feature produced by initial network weights and learning algorithm settings can significantly impact the training process of very deep network. To address this problem, batch normalization [43] is used during

the network training. On the other hand, the attention mechanism has an effect on the deep learning feature transformation which enhances the valuable features among the original feature maps. Besides, the CNN architecture plays an important role in feature transformation because of end-to-end training strategy. Last, the classic dimension reduction algorithms have been proved to be useful in many machine learning models, and can be embedded in the deep learning networks, and provide better performance because of deep learning's end-to-end training strategy. In this section, we discuss the above three feature transformation techniques used in deep learning tasks.

Firstly, batch normalization (BN) is a proven technology for training deep neural networks. During training, the varying distribution of inputs to each layer makes the training process slow and hard to converge. To stabilize each layer's input, batch normalization is applied right after the convolutional layer; it calculates the running mean and standard deviation of the layer's output based on each batch. Besides, batch normalization has additional parameters to scale and shift the normalized output. After training, the calculated parameters are fixed for further testing. Batch normalization can significantly reduce the training time and stabilize the training process.

Secondly, attention [44] is another popular technique for transforming feature. There are many methods to obtain feature attention in deep-learning-based computer vision. Specifically, there are two popular attention mechanism in computer vision: soft attention and hard attention [27]. The soft attention [17] is derived based on the global attention where all image patches are given the attention scores, and varies between 0 and 1; however, hard attention only consider one image patch at once, and generate the attention scores as 0 or 1. The most common method is calculating the similarity between each pair of feature points in the same channel, so the obtained similarity reflects the relationship between

feature points. This similarity is the so-called attention score which can be used as the weight of feature points. The attention scores are embedded in the transformed features by applying the weighted sum to the original features.

Next, the characteristics of CNN enables the arising of various model architectures which play an important role in feature transformation. In theory, deep CNN with more layers usually have better learning ability to transform features to desirable manifold. However, the deep CNN was hard to train because of vanishing gradients which caused by feature transforming during the training process. Therefore, ResNet [35] solves this problem by employing the residual learning so that the usage of deeper network is largely expanded. Besides, generative adversarial network (GAN) [34] also introduce a novel deep learning model to the deep learning-based image generation tasks [23] [3]. The GAN is usually consisted of two sub-modules: generator learns to generate more similar images with ground truth while discriminator learn to distinguish the ground truth from lots of generated images. This learning strategy helps to find suitable domain for feature transformation.

Moreover, the deep learning networks bring the classic algorithms back to spotlight again. Here we discuss several feature transformation techniques [28] [30] [31] that combines with deep learning networks. The LDA aims to find a projection that maximizes the between-class distance and minimizes the within-class distance. In deep LDA [28], it simplifies the LDA algorithm into a eigenvalue maximization problem which can be treated as the target function in deep learning. Compared with DNN that is trained with categorical cross entropy, the deep LDA has better performance in image classification tasks. For deep CCA [30], it utilizes two deep networks to learn from two different inputs. The goal of deep CCA is to jointly learn parameters from both inputs, and maximizing the

correlation between two inputs. The experimental results show better performance than the classic CCA, and proves again that traditional methods can be further improved under deep learning techniques.

In this thesis, we present a CCA-based feature dimension reduction technique for image retrieval in Chapter 2, and a feature transformation framework with attention and batch normalization for image inpainting in Chapter 3.

## 1.5 Contributions and Thesis Organization

This *sandwich thesis* follows the terms and regulations of McMaster University, including three published/unpublished works that present several feature extraction and feature transformation techniques. The contributions to each work are outlined in the preface of Chapter 2 and Chapter 3. The reference information for these two works is listed below:

- Kangdi Shi, Xiaohong Liu, Muhammad Alrabeiah, Xintong Guo, Jie Lin, Huan Liu, and Jun Chen. "Image Retrieval via Canonical Correlation Analysis and Binary Hypothesis Testing." *Information* 13, no. 3 (2022): 106.
- Kangdi Shi, Muhammad Alrabeiah and Jun Chen. "Progressive with Purpose: Guiding Progressive Inpainting DNNs through Context and Structure." Submitted to IEEE Access.

The rest of this thesis is organized as follows:

**Chapter 2:** The details of the proposed CCA-based dimension reduction method for deep learning features and its application to image retrieval.

**Chapter 3:** The details of the proposed progressive image inpainting method based on designed feature extraction network and image inpainting framework.

**Chapter 4:** The conclusion of this thesis and the discussion of future work.

# Bibliography

- [1] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.
- [2] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.
- [3] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, “Stargan: Unified generative adversarial networks for multi-domain image-to-image translation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8789–8797.
- [4] Z. Guo, Z. Chen, T. Yu, J. Chen, and S. Liu, “Progressive image inpainting with full-resolution residual network,” in *Proceedings of the 27th acm international conference on multimedia*, 2019, pp. 2496–2504.
- [5] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE conference on computer vision*

*and pattern recognition*, 2017, pp. 4700–4708.

- [6] L. Liao, J. Xiao, Z. Wang, C.-W. Lin, and S. Satoh, “Uncertainty-aware semantic guidance and estimation for image inpainting,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 2, pp. 310–323, 2020.
- [7] H. Zhang, Z. Hu, C. Luo, W. Zuo, and M. Wang, “Semantic image inpainting with progressive generative networks,” in *Proceedings of the 26th ACM international conference on Multimedia*, 2018, pp. 1939–1947.
- [8] Z. Guo, Z. Chen, T. Yu, J. Chen, and S. Liu, “Progressive image inpainting with full-resolution residual network,” in *Proceedings of the 27th acm international conference on multimedia*, 2019, pp. 2496–2504.
- [9] Y. Ren, X. Yu, R. Zhang, T. H. Li, S. Liu, and G. Li, “Structureflow: Image inpainting via structure-aware appearance flow,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 181–190.
- [10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: <https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>
- [11] Y. Song, C. Yang, Y. Shen, P. Wang, Q. Huang, and C.-C. J. Kuo, “Spg-net: Segmentation prediction and guidance network for image inpainting,” *arXiv preprint arXiv:1805.03356*, 2018.

- [12] X.-S. Wei, J.-H. Luo, J. Wu, and Z.-H. Zhou, “Selective convolutional descriptor aggregation for fine-grained image retrieval,” *IEEE Transactions on Image Processing*, vol. 26, no. 6, pp. 2868–2881, 2017.
- [13] K. Nazeri, E. Ng, T. Joseph, F. Qureshi, and M. Ebrahimi, “Edgeconnect: Structure guided image inpainting using edge prediction,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0.
- [14] H. Liu, B. Jiang, Y. Xiao, and C. Yang, “Coherent semantic attention for image inpainting,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4170–4179.
- [15] H. Jégou, M. Douze, C. Schmid, and P. Pérez, “Aggregating local descriptors into a compact image representation,” in *2010 IEEE computer society conference on computer vision and pattern recognition*. IEEE, 2010, pp. 3304–3311.
- [16] Y. Zeng, J. Fu, H. Chao, and B. Guo, “Learning pyramid-context encoder network for high-quality image inpainting,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1486–1494.
- [17] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.
- [18] T. Jaakkola and D. Haussler, “Exploiting generative models in discriminative classifiers,” *Advances in neural information processing systems*, vol. 11, 1998.



- [19] K. Yan, Y. Wang, D. Liang, T. Huang, and Y. Tian, “Cnn vs. sift for image retrieval: Alternative or complementary?” in *Proceedings of the 24th ACM international conference on Multimedia*, 2016, pp. 407–411.
- [20] A. Babenko and V. Lempitsky, “Aggregating local deep features for image retrieval,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1269–1277.
- [21] Z. Shi, X. Liu, K. Shi, L. Dai, and J. Chen, “Video frame interpolation via generalized deformable convolution,” *IEEE Transactions on Multimedia*, vol. 24, pp. 426–439, 2021.
- [22] S. Zagoruyko and N. Komodakis, “Wide residual networks,” *arXiv preprint arXiv:1605.07146*, 2016.
- [23] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.
- [24] X. Liu, K. Shi, Z. Wang, and J. Chen, “Exploit camera raw data for video super-resolution via hidden markov model inference,” *IEEE Transactions on Image Processing*, vol. 30, pp. 2127–2140, 2021.
- [25] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, “Image inpainting for irregular holes using partial convolutions,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 85–100.

- [26] J. Li, N. Wang, L. Zhang, B. Du, and D. Tao, “Recurrent feature reasoning for image inpainting,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7760–7768.
- [27] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, “Show, attend and tell: Neural image caption generation with visual attention,” in *International conference on machine learning*. PMLR, 2015, pp. 2048–2057.
- [28] M. Dorfer, R. Kelz, and G. Widmer, “Deep linear discriminant analysis,” *arXiv preprint arXiv:1511.04707*, 2015.
- [29] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid, “Vivit: A video vision transformer,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6836–6846.
- [30] G. Andrew, R. Arora, J. Bilmes, and K. Livescu, “Deep canonical correlation analysis,” in *International conference on machine learning*. PMLR, 2013, pp. 1247–1255.
- [31] E. J. Candès, X. Li, Y. Ma, and J. Wright, “Robust principal component analysis?” *Journal of the ACM (JACM)*, vol. 58, no. 3, pp. 1–37, 2011.
- [32] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [33] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

- [34] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.
- [35] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [36] M. Lin, Q. Chen, and S. Yan, “Network in network,” *arXiv preprint arXiv:1312.4400*, 2013.
- [37] LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proceedings Of The IEEE*. **86**, 2278-2324 (1998)
- [38] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [39] S. Wold, K. Esbensen, and P. Geladi, “Principal component analysis,” *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.
- [40] T. Mitchell, B. Buchanan, G. DeJong, T. Dietterich, P. Rosenbloom, and A. Waibel, “Machine learning,” *Annual review of computer science*, vol. 4, no. 1, pp. 417–433, 1990.
- [41] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *International conference on machine learning*. PMLR, 2015, pp. 448–456.
- [42] A. Zheng and A. Casari, *Feature engineering for machine learning: principles and techniques for data scientists*. ” O’Reilly Media, Inc.”, 2018.

- [43] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *International conference on machine learning*. PMLR, 2015, pp. 448–456.
- [44] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.
- [45] J. Brownlee, *Data preparation for machine learning: data cleaning, feature selection, and data transforms in Python*. Machine Learning Mastery, 2020.
- [46] Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv*, **2014**, arXiv:1409.1556.
- [47] G. Dong and H. Liu, *Feature engineering for machine learning and data analytics*. CRC Press, 2018.
- [48] S. Petscharnig, M. Lux, and S. Chatzichristofis, “Dimensionality reduction for image features using deep learning and autoencoders,” in *Proceedings of the 15th international workshop on content-based multimedia indexing*, 2017, pp. 1–6.
- [49] S. Sarica and J. Luo, “Stopwords in technical language processing,” *Plos one*, vol. 16, no. 8, p. e0254937, 2021.
- [50] C. Harris, M. Stephens *et al.*, “A combined corner and edge detector,” in *Alvey vision conference*, vol. 15, no. 50. Manchester, UK, 1988, pp. 10–5244.
- [51] Fisher, R.A. The use of multiple measurements in taxonomic problems. *Ann. Eugen.* **1936**, 7, 179–188.

- [52] Johnson, R.A.; Wichern, D.W. Canonical correlation analysis. In *Applied Multivariate Statistical Analysis*, 6th ed.; Pearson: Upper Saddle River, New Jersey 2018; pp. 539–574.
- [53] N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, A. Ku, and D. Tran, “Image transformer,” in *International conference on machine learning*. PMLR, 2018, pp. 4055–4064.

The following chapter is a reproduction of an Multidisciplinary Digital Publishing Institute (MDPI) copyrighted, published paper:

Shi, Kangdi, Xiaohong Liu, Muhammad Alrabeiah, Xintong Guo, Jie Lin, Huan Liu, and Jun Chen. "Image Retrieval via Canonical Correlation Analysis and Binary Hypothesis Testing." *Information* 13, no. 3 (2022): 106.

In reference to MDPI copyrighted material which is used with permission in this thesis, no special permission is required to reuse all or part of article published by MDPI, including figures and tables. For articles published under an open access Creative Common CC BY license, any part of the article may be reused without permission provided that the original article is clearly cited. Reuse of an article does not imply endorsement by the authors or MDPI.

Contribution Declaration: Kangdi Shi (the author of this thesis) is the first author and main contributor of this article. He conducted all experiments and composed the paper. He works with Prof. Jun Chen on the proposed method. Prof. Jun Chen is the supervisor of Kangdi Shi.

## **Chapter 2**

# **Image Retrieval via Canonical Correlation Analysis and Binary Hypothesis Testing**

### **2.1 Abstract**

Canonical Correlation Analysis (CCA) is a classic multivariate statistical technique, which can be used to find a projection pair that maximally captures the correlation between two sets of random variables. The present paper introduces a CCA-based approach for image retrieval. It capitalizes on feature maps induced by two images under comparison through a pre-trained Convolutional Neural Network (CNN) and leverages basis vectors identified through CCA, together with an element-wise selection method based on a Chernoff-information-related criterion, to produce compact transformed image features; a binary hypothesis test regarding the joint distribution of transformed feature pair is

then employed to measure the similarity between two images. The proposed approach is benchmarked against two alternative statistical methods, Linear Discriminant Analysis (LDA) and Principal Component Analysis with whitening (PCAw). Our CCA-based approach is shown to achieve highly competitive retrieval performances on standard datasets, which include, among others, Oxford5k and Paris6k.

## 2.2 Introduction

The past two decades have witnessed an explosive growth of online image databases. This growth paves the way for the development of visual-data-driven applications, but at the same time poses a major challenge to the Content-Based Image Retrieval (CBIR) technology [1].

Traditional approaches to CBIR mostly rely on the exploitation of handcrafted scale- and orientation-invariant image features [2–6], which have achieved varying degrees of success. Recent advances [7, 8] in Deep Learning (DL) for image classification and object detection have generated significant interests in bringing Convolutional Neural Networks (CNNs) to bear upon CBIR. Although CNN models are usually trained for purposes different from CBIR, it is known [9] that features extracted from modern deep CNNs, commonly referred to as DL features, have great potential in this respect as well. Retrieval methods utilizing DL features can generally be divided into two categories: without/with fine-tuning the CNN model [10]. The early application of CNN to CBIR almost exclusively resorts to methods in the first category, which use Off-The-Shelf (OTS) CNNs (i.e., popular pre-trained CNNs) for feature extraction (see, e.g., [11–14]). A main advantage of such methods is the low implementation cost [15, 16], which is largely attributed to the direct adoption of pre-trained CNNs. Performance-wise, they are comparable to the



state-of-the-art traditional methods that rely on handcrafted features. In contrast, many recent methods, such as [17–19], belong to the second category, which take advantage of the fine-tuning gain to enhance the discriminatory power of the extracted DL features. A top representative from this category is the end-to-end learning framework proposed in [20]. It outperforms most existing traditional and OTS-CNN-based methods on standard testing datasets; however, this performance improvement comes at the cost of training a complex triple-branched CNN using a large dataset, which might not always be affordable in practice.

Many preprocessing methods have been developed with the goal of better utilizing DL features for image retrieval, among which Principal Component Analysis with whitening [21] (PCAw) and Linear Discriminant Analysis [22] (LDA) are arguably most well known. Despite being extremely popular, PCA and LDA have their respective weaknesses: the dimensionality reduction in PCA often leads to the elimination of critical principal components with a small contribution rate while the performance of LDA tends to suffer from decreasing differences between mismatched features. As such, there is great need for a preprocessing method with improved robustness against dimensionality reduction and enhanced sensitivity to feature mismatch. In this work, we aim to put forward a potential solution with desired properties by bringing Canonical Correlation Analysis (CCA) [23] into the picture.

CCA is a multivariate technique for elucidating the associations among two sets of variables. It can be used to identify a projection pair of a given dimension that maximally captures the correlation between the two sets. The applications of CCA are too numerous to list. In cross-modality matching/retrieval alone, extensive investigations have been carried out as evidenced by a growing body of literature, from those based on handcrafted

features [24] to the more recent ones that make use of DL features [25–27]. There is also some related development on the theoretical front (see, e.g., [28, 29]).

Motivated by the consideration of computational efficiency and affordability as well as the weaknesses inherent in the existing preprocessing methods, we develop and present in this paper a new image retrieval method based on OTS deep CNNs. Our method is built primarily upon CCA, but has several notable differences from the related works. For the purpose of dimensionality reduction (i.e., feature compression), the proposed method employs a basis-vector selection technique, which invokes a Chernoff-information-based criterion to rank how discriminative the basis vectors are. Both the basis vectors and their ranking are learned from a training set, which consists of features extracted from a pre-trained CNN—the neural network itself is not retrained/finetuned in our work. Given a new pair of features, the ranked basis vectors are used to perform transformation and compression. This is followed by a binary hypothesis test on the joint distribution of pairs of transformed features, which yields a matching score that can be leveraged to identify top candidates for retrieval. We show via extensive experimental results that the proposed CCA-based method is able to deliver highly competitive results on standard datasets, which include, among others, Oxford5k and Parise6k.

This paper is organized as follows. The proposed CCA-based preprocessing method along with the associated matching procedure is detailed in Section 2.3. Section 2.4 includes the experimental results and the relevant discussions. We close the paper in Section 2.5 with some concluding remarks.

## 2.3 Proposed Method

The proposed image retrieval method utilizes CCA in an essential way. It leverages a training dataset of features extracted from a pre-trained CNN model (see Figure 2.1) to learn a set of canonical vectors, which serve as the basis vectors of the feature space. These vectors are used to project the features of a pair of images into a new space, in which a Chernoff-information-based selection method is applied to identify the most discriminative elements of the transformed features. Such elements then undergo a binary hypothesis test to measure the similarity between the features and, consequently, the two images. This process is expounded in the following four subsections (see also Figure 2.2).

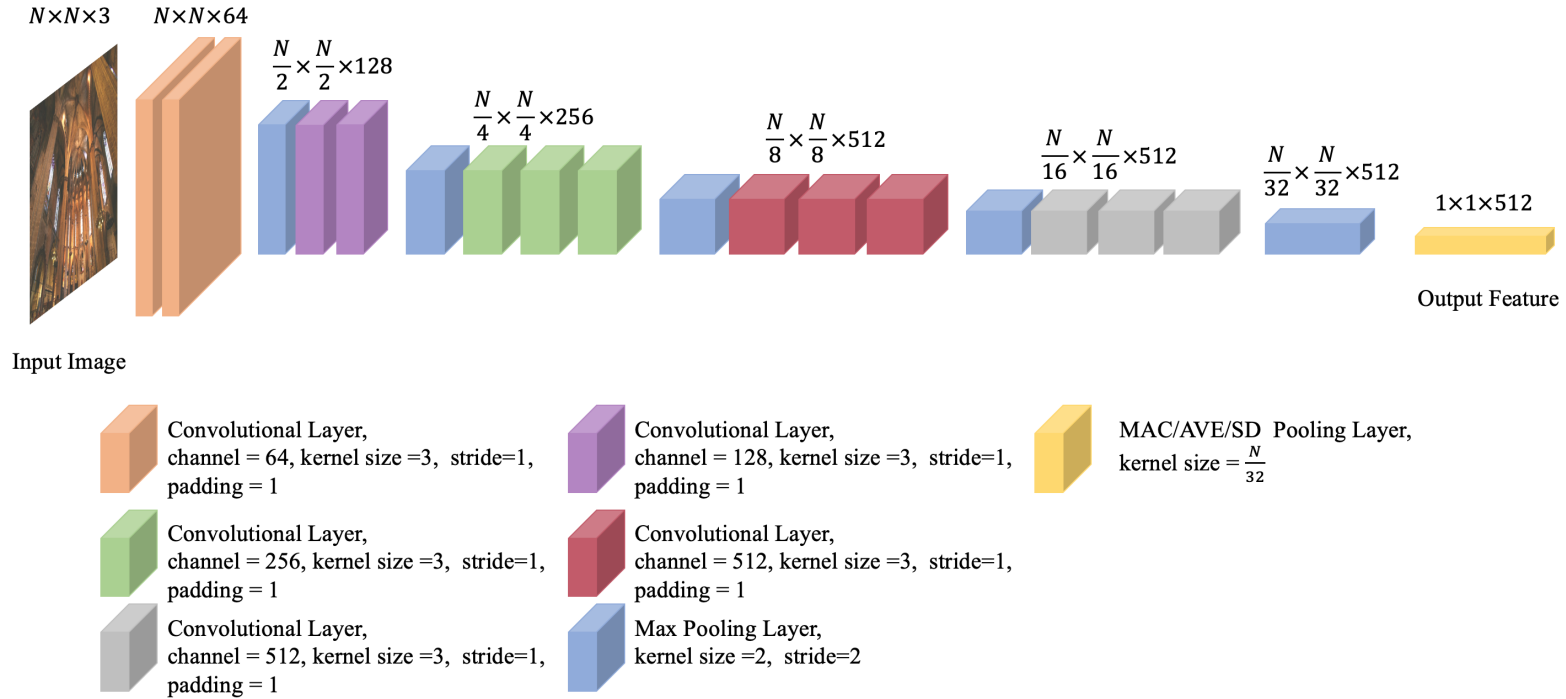


Figure 2.1: Modified VGG16 for feature extraction.

### **2.3.1 Image Pre-Processing and Feature Extraction**

The CNN model adopted in this work for feature extraction is VGG16 [30]. It takes an input image of maximum size  $1024 \times 1024$  and produces 512 feature maps of maximum size  $32 \times 32$  from its very last pooling layer. A single feature element is extracted from each feature map via pooling. A 512-dimensional vector, which resulted from the concatenation of these elements, is converted, through centralization and normalization (here centralization is performed by subtracting the mean (computed based on the training set) while normalization yields a unit-length vector), to a global feature vector, which serves as a compact representation of the image.

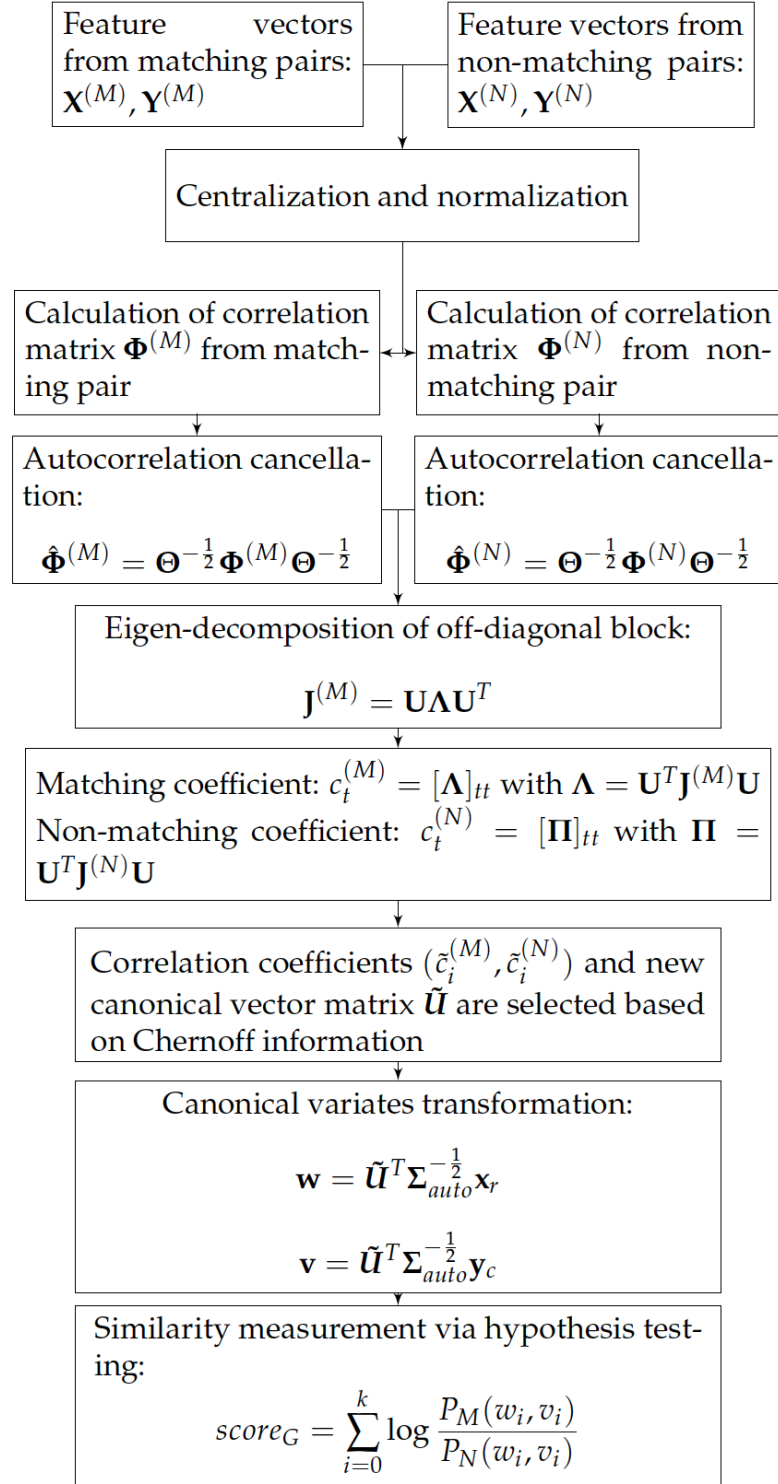


Figure 2.2: Block diagram of the proposed method.

### 2.3.2 Correlation Analysis and Canonical Vectors

At the heart of the proposed method lies so-called canonical vectors, which are learned from a large training set of matching and non-matching image features in a manner inspired by CCA. The learning process consists of the following steps.

*Step 1* : Construct two raw matching matrices

$$\mathbf{X}^{(RM)} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_L],$$

$$\mathbf{Y}^{(RM)} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_L],$$

where  $L$  is the number of raw matching pairs,  $\mathbf{x}_l$  and  $\mathbf{y}_l$  for  $l \in \{1, 2, \dots, L\}$  are a pair of global feature vectors representing two matching images (here “matching images” means images from the same class while “non-matching images” means images from different classes).

Using the raw matching pairs  $\mathbf{X}^{(RM)}$  and  $\mathbf{Y}^{(RM)}$ , a pair of matching-feature matrices is formed:

$$\mathbf{X}^{(M)} = [\mathbf{x}_1, \mathbf{y}_1, \mathbf{x}_2, \mathbf{y}_2, \dots, \mathbf{y}_L, \mathbf{x}_L],$$

$$\mathbf{Y}^{(M)} = [\mathbf{y}_1, \mathbf{x}_1, \mathbf{y}_2, \mathbf{x}_2, \dots, \mathbf{x}_L, \mathbf{y}_L].$$

The total number of training pairs is  $2L$  after feature order flipped. This is performed to ensure that in Equation (2.3.2.1) below, the diagonal blocks are identical and symmetric, so are the off-diagonal blocks. The size of both  $\mathbf{X}^{(M)}$  and  $\mathbf{Y}^{(M)}$  is  $512 \times 2L$ . The training data matrix of matching features  $\mathbf{H}^{(M)}$  is constructed by stacking  $\mathbf{X}^{(M)}$  on  $\mathbf{Y}^{(M)}$ :

$$\mathbf{H}^{(M)} = \begin{bmatrix} \mathbf{X}^{(M)} \\ \mathbf{Y}^{(M)} \end{bmatrix}_{(1024 \times 2L)}.$$

The estimated covariance matrix of matching features is given by

$$\begin{aligned} \Phi^{(M)} &= \frac{1}{2L-1} \mathbf{H}^{(M)} (\mathbf{H}^{(M)})^T \\ &= \frac{1}{2L-1} \begin{bmatrix} \mathbf{X}^{(M)} \\ \mathbf{Y}^{(M)} \end{bmatrix} \begin{bmatrix} \mathbf{X}^{(M)} \\ \mathbf{Y}^{(M)} \end{bmatrix}^T \\ &= \begin{bmatrix} \Sigma_{XX}^{(M)} & \Sigma_{XY}^{(M)} \\ \Sigma_{YX}^{(M)} & \Sigma_{YY}^{(M)} \end{bmatrix}, \end{aligned} \quad (2.3.2.1)$$

where

$$\begin{aligned} \Sigma_{XX}^{(M)} &= \frac{\mathbf{X}^{(M)} (\mathbf{X}^{(M)})^T}{2L-1}, & \Sigma_{YY}^{(M)} &= \frac{\mathbf{Y}^{(M)} (\mathbf{Y}^{(M)})^T}{2L-1}, \\ \Sigma_{XY}^{(M)} &= \Sigma_{YX}^{(M)} = \frac{\mathbf{X}^{(M)} (\mathbf{Y}^{(M)})^T}{2L-1} = \frac{\mathbf{Y}^{(M)} (\mathbf{X}^{(M)})^T}{2L-1}. \end{aligned}$$

*Step 2:* Randomly permuting the columns of one of the raw feature matrices, say from  $\mathbf{Y}^{(RM)}$  to  $\mathbf{Y}^{(RN)}$ , yields two raw non-matching matrices. More specifically, we construct two raw non-matching matrices by successively associating each column of  $\mathbf{X}^{(RM)}$  with a randomly selected (without replacement) non-matching column from  $\mathbf{Y}^{(RM)}$ . For example,

$$\mathbf{X}^{(RN)} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_L],$$

$$\mathbf{Y}^{(RN)} = [\mathbf{y}_3, \mathbf{y}_7, \dots, \mathbf{y}_{L-4}].$$



Based on these two raw non-matching matrices, the feature order flipping is performed to generate  $\mathbf{X}^{(N)}$  and  $\mathbf{Y}^{(N)}$ :

$$\mathbf{X}^{(N)} = [\mathbf{x}_1, \mathbf{y}_3, \mathbf{x}_2, \mathbf{y}_7, \dots, \mathbf{x}_L, \mathbf{y}_{L-4}],$$

$$\mathbf{Y}^{(N)} = [\mathbf{y}_3, \mathbf{x}_1, \mathbf{y}_7, \mathbf{x}_2, \dots, \mathbf{y}_{L-4}, \mathbf{x}_L].$$

With a procedure similar to that of *step 1*, we can estimate the covariance matrix  $\Phi^{(N)}$  for non-matching features  $\mathbf{H}^{(N)}$ :

$$\begin{aligned} \Phi^{(N)} &= \frac{1}{2L-1} \mathbf{H}^{(N)} (\mathbf{H}^{(N)})^T \\ &= \frac{1}{2L-1} \begin{bmatrix} \mathbf{X}^{(N)} \\ \mathbf{Y}^{(N)} \end{bmatrix} \begin{bmatrix} \mathbf{X}^{(N)} \\ \mathbf{Y}^{(N)} \end{bmatrix}^T \\ &= \begin{bmatrix} \Sigma_{XX}^{(N)} & \Sigma_{XY}^{(N)} \\ \Sigma_{YX}^{(N)} & \Sigma_{YY}^{(N)} \end{bmatrix}. \end{aligned} \tag{2.3.2.2}$$

Note that

$$\Sigma_{XX}^{(N)} = \Sigma_{YY}^{(N)} = \Sigma_{XX}^{(M)} = \Sigma_{YY}^{(M)} = \Sigma_{auto},$$

for they are the covariances of sets of random image features. As in Equation (2.3.2.1), the diagonal blocks in Equation (2.3.2.2) are also identical and symmetric, so are the off-diagonal blocks.

*Step 3:* Since  $\Sigma_{auto}$  is positive definite, it follows that  $\Theta^{-\frac{1}{2}}$  is well defined, where

$$\Theta = \begin{bmatrix} \Sigma_{auto} & \mathbf{0} \\ \mathbf{0} & \Sigma_{auto} \end{bmatrix}.$$

We can multiply both covariance matrices,  $\Phi^{(M)}$  and  $\Phi^{(N)}$ , on the left and right by  $\Theta^{-\frac{1}{2}}$  to de-correlate their diagonal blocks:

$$\hat{\Phi}^{(M)} = \Theta^{-\frac{1}{2}} \Phi^{(M)} \Theta^{-\frac{1}{2}} = \begin{bmatrix} \mathbf{I} & \mathbf{J}^{(M)} \\ \mathbf{J}^{(M)} & \mathbf{I} \end{bmatrix},$$

$$\hat{\Phi}^{(N)} = \Theta^{-\frac{1}{2}} \Phi^{(N)} \Theta^{-\frac{1}{2}} = \begin{bmatrix} \mathbf{I} & \mathbf{J}^{(N)} \\ \mathbf{J}^{(N)} & \mathbf{I} \end{bmatrix},$$

where

$$\mathbf{J}^{(M)} = \Sigma_{auto}^{-\frac{1}{2}} \Sigma_{XY}^{(M)} \Sigma_{auto}^{-\frac{1}{2}} = \Sigma_{auto}^{-\frac{1}{2}} \Sigma_{YX}^{(M)} \Sigma_{auto}^{-\frac{1}{2}},$$

$$\mathbf{J}^{(N)} = \Sigma_{auto}^{-\frac{1}{2}} \Sigma_{XY}^{(N)} \Sigma_{auto}^{-\frac{1}{2}} = \Sigma_{auto}^{-\frac{1}{2}} \Sigma_{YX}^{(N)} \Sigma_{auto}^{-\frac{1}{2}}.$$

*Step 4:* Apply eigen-decomposition [31] on  $\mathbf{J}^{(M)}$ :

$$\mathbf{J}^{(M)} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^T,$$

where  $\mathbf{U}$  is a unitary matrix, and  $\mathbf{\Lambda}$  is a diagonal matrix with the diagonal entries being the eigenvalues of  $\mathbf{J}^{(M)}$ . The columns of  $\mathbf{U}$  are exactly the sought-after canonical vectors. The blockwise left- and right-multiplication of both  $\hat{\Phi}^{(M)}$  and  $\hat{\Phi}^{(N)}$  by  $\mathbf{U}^T$  and  $\mathbf{U}$ , respectively, gives the following pair of matrices:

$$\begin{bmatrix} \mathbf{U}^T \mathbf{U} & \mathbf{U}^T \mathbf{J}^{(M)} \mathbf{U} \\ \mathbf{U}^T \mathbf{J}^{(M)} \mathbf{U} & \mathbf{U}^T \mathbf{U} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{\Lambda} \\ \mathbf{\Lambda} & \mathbf{I} \end{bmatrix}, \quad (2.3.2.3)$$

$$\begin{bmatrix} \mathbf{U}^T \mathbf{U} & \mathbf{U}^T \mathbf{J}^{(N)} \mathbf{U} \\ \mathbf{U}^T \mathbf{J}^{(N)} \mathbf{U} & \mathbf{U}^T \mathbf{U} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{\Pi} \\ \mathbf{\Pi} & \mathbf{I} \end{bmatrix}, \quad (2.3.2.4)$$

where  $\mathbf{\Pi} = \mathbf{U}^T \mathbf{J}^{(N)} \mathbf{U}$ . The off-diagonal block  $\mathbf{\Lambda}$  in Equation (2.3.2.3) is a diagonal matrix whereas  $\mathbf{\Pi}$  in Equation (2.3.2.4) is not necessarily so. Nevertheless, it will be seen that in practice  $\mathbf{\Pi}$  is often close to a zero matrix (as two non-matching image features tend to be uncorrelated) and thus is approximately diagonal as well.

### 2.3.3 Chernoff Information for Canonical Vector Selection

Chernoff information (*CI*) was originally introduced for finding the upper bound for error probability of Bayesian 2-class classification problem, and has [32, 33] been proven to be effective for measuring the distance between any two distributions from the same exponential family. Specifically, the Chernoff information is equivalent to calculating the KL divergence between middle distribution and anyone of given distributions. In this section, we illustrate the approach of selecting discriminative canonical vectors by using chernoff information.

Note that the learned canonical vectors of matching image features form an orthonormal basis of  $\mathbb{R}^{512}$ . These vectors are not necessarily equally useful for the purpose of measuring the similarity between two feature vectors of an unknown pair of images; therefore, it is of considerable interest to quantify how *discriminative* each canonical vector is. To this end, the off-diagonal blocks of the covariance matrix of non-matching image features can be brought into play. Evaluating Chernoff information with respect to the diagonal elements

of both  $\Lambda$  and  $\Pi$  yields a ranking of the most different diagonal element pairs, which can be used to guide the selection of canonical vectors.

Define the following set of  $2 \times 2$  matrices

$$\mathbf{S}_t^{(M)} = \begin{bmatrix} 1 & c_t^{(M)} \\ c_t^{(M)} & 1 \end{bmatrix},$$

$$\mathbf{S}_t^{(N)} = \begin{bmatrix} 1 & c_t^{(N)} \\ c_t^{(N)} & 1 \end{bmatrix},$$

using matching coefficient  $c_t^{(M)} = [\Lambda]_{tt}$  and non-matching coefficient  $c_t^{(N)} = [\Pi]_{tt}$ ,  $t \in \{1, 2, \dots, 512\}$ , determined by the diagonal elements of  $\Lambda$  and  $\Pi$ :

$$\Lambda = \begin{bmatrix} c_1^{(M)} & 0 & \dots & 0 \\ 0 & c_2^{(M)} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & c_{512}^{(M)} \end{bmatrix},$$

$$\Pi = \begin{bmatrix} c_1^{(N)} & \pi_{1,2} & \dots & \pi_{1,512} \\ \pi_{2,1} & c_2^{(N)} & \dots & \pi_{2,512} \\ \vdots & \vdots & \ddots & \vdots \\ \pi_{512,1} & \pi_{512,2} & \dots & c_{512}^{(N)} \end{bmatrix}.$$

Now let  $\mathbf{S}_t^{(\lambda_t)} = (\lambda_t(\mathbf{S}_t^{(M)})^{-1} + (1 - \lambda_t)(\mathbf{S}_t^{(N)})^{-1})^{-1}$ ,  $\lambda_t \in [0, 1]$  and define

$$D(\mathbf{S}_t^{(\lambda_t)} \parallel \mathbf{S}_t^{(M)}) = \frac{1}{2} \log_e \frac{|\mathbf{S}_t^{(M)}|}{|\mathbf{S}_t^{(\lambda_t)}|} + \frac{1}{2} \text{tr}((\mathbf{S}_t^{(M)})^{-1} \mathbf{S}_t^{(\lambda_t)}) - 1,$$

$$D(\mathbf{S}_t^{(\lambda_t)} \parallel \mathbf{S}_t^{(N)}) = \frac{1}{2} \log_e \frac{|\mathbf{S}_t^{(N)}|}{|\mathbf{S}_t^{(\lambda_t)}|} + \frac{1}{2} \text{tr}((\mathbf{S}_t^{(N)})^{-1} \mathbf{S}_t^{(\lambda_t)}) - 1,$$

where  $\text{tr}(\cdot)$  is the trace operator. Kullback–Leibler divergence is a special case of Bregman divergence, and can be used to calculate the statistical distance between any two distributions. Let  $\lambda_t = \lambda_t^*$  be the solution of  $D(\mathbf{S}_t^{(\lambda_t)} \parallel \mathbf{S}_t^{(M)}) = D(\mathbf{S}_t^{(\lambda_t)} \parallel \mathbf{S}_t^{(N)})$ . The Chernoff information  $CI(\mathbf{S}_t^{(M)} \parallel \mathbf{S}_t^{(N)})$  is defined as

$$CI(\mathbf{S}_t^{(M)} \parallel \mathbf{S}_t^{(N)}) = D(\mathbf{S}_t^{(\lambda_t^*)} \parallel \mathbf{S}_t^{(M)}) = D(\mathbf{S}_t^{(\lambda_t^*)} \parallel \mathbf{S}_t^{(N)}).$$

An expression for individual  $\lambda_t^*$  is derived in Appendix 2.A.

Given  $\lambda_t^*$ ,  $CI$  of all pairs  $(\mathbf{S}_t^{(M)}, \mathbf{S}_t^{(N)})$  can be evaluated, leading to a ranking (greater  $CI$  corresponds to higher rank) of the most different pairs of diagonal elements  $(c_t^{(M)}, c_t^{(N)})$  and, consequently, the most discriminative canonical vectors of  $\mathbf{U}$ . Let the  $k$  most discriminative vectors serve as the columns of the new canonical vector matrix  $\tilde{\mathbf{U}}$ . Moreover, select the top  $k$  different pairs of diagonal elements  $(\tilde{c}_i^{(M)}, \tilde{c}_i^{(N)})$  and the corresponding  $(\tilde{\mathbf{S}}_i^{(M)}, \tilde{\mathbf{S}}_i^{(N)})$ , where  $i \in \{1, 2, \dots, k\}$ .

### 2.3.4 Similarity Measurement

The selected canonical vectors can be leveraged to measure the similarity between an arbitrary pair of images through a binary hypothesis test. Let  $(\mathbf{x}_r, \mathbf{y}_c)$  be an arbitrary pair of global feature vectors. The exact joint distribution of  $(\mathbf{x}_r, \mathbf{y}_c)$  likely varies from

one dataset to another and does not admit an explicit characterization. Here we make the simplifying assumption that  $\mathbf{x}_r$  and  $\mathbf{y}_c$  are jointly Gaussian. Specifically, we assume that  $(\mathbf{x}_r, \mathbf{y}_c) \sim \mathcal{N}(\mathbf{0}, \Phi^{(M)})$  if they come from two matching images, and  $(\mathbf{x}_r, \mathbf{y}_c) \sim \mathcal{N}(\mathbf{0}, \Phi^{(N)})$  otherwise, where  $\mathcal{N}(\mathbf{0}, \Sigma)$  denotes a multivariate Gaussian distribution [34] with mean  $\mathbf{0}$  and covariance matrix  $\Sigma$ . Given  $(\mathbf{x}_r, \mathbf{y}_c)$ , the transformed feature vectors are computed as follows:

$$\begin{aligned}\mathbf{w} &= [w_1, w_2, \dots, w_k]^T = \tilde{\mathbf{U}}^T \Sigma_{auto}^{-\frac{1}{2}} \mathbf{x}_r, \\ \mathbf{v} &= [v_1, v_2, \dots, v_k]^T = \tilde{\mathbf{U}}^T \Sigma_{auto}^{-\frac{1}{2}} \mathbf{y}_c.\end{aligned}$$

Since  $\Lambda$  is a diagonal matrix, it follows that  $(w_1, v_1), (w_2, v_2), \dots, (w_k, v_k)$  are mutually independent with  $(w_i, v_i) \sim \mathcal{N}(\mathbf{0}, \tilde{\mathbf{S}}_i^{(M)})$  for  $i \in \{1, 2, \dots, k\}$  in the case where  $(\mathbf{x}_r, \mathbf{y}_c)$  is a matching pair. We shall further assume that  $\Pi$  is also a diagonal matrix, which is justified by the fact that in practice  $\Pi$  is often very close to a zero matrix (see Figure 2.5 and 2.6 for some empirical evidences). As a consequence,  $(w_1, v_1), (w_2, v_2), \dots, (w_k, v_k)$  are mutually independent with  $(w_i, v_i) \sim \mathcal{N}(\mathbf{0}, \tilde{\mathbf{S}}_i^{(N)})$  for  $i \in \{1, 2, \dots, k\}$  in the case where  $(\mathbf{x}_r, \mathbf{y}_c)$  is a non-matching pair. To check whether the given two images match or not, one can perform a binary hypothesis test regarding the underlying distribution of  $(\mathbf{w}, \mathbf{v})$ :  $\otimes_{i=1}^k \mathcal{N}(\mathbf{0}, \tilde{\mathbf{S}}_i^{(M)})$  vs.  $\otimes_{i=1}^k \mathcal{N}(\mathbf{0}, \tilde{\mathbf{S}}_i^{(N)})$ .

Note that  $\mathcal{N}(\mathbf{0}, \tilde{\mathbf{S}}_i^{(M)})$  has probability density

$$P_M(w_i, v_i) = \frac{e^{-\frac{1}{2} \begin{bmatrix} w_i & v_i \end{bmatrix} \begin{bmatrix} 1 & \tilde{c}_i^{(M)} \\ \tilde{c}_i^{(M)} & 1 \end{bmatrix}^{-1} \begin{bmatrix} w_i \\ v_i \end{bmatrix}}{\sqrt{(2\pi)^2 \begin{vmatrix} 1 & \tilde{c}_i^{(M)} \\ \tilde{c}_i^{(M)} & 1 \end{vmatrix}}} \quad (2.3.4.1)$$

while  $\mathcal{N}(\mathbf{0}, \tilde{\mathbf{S}}_i^{(N)})$  has probability density

$$P_N(w_i, v_i) = \frac{e^{-\frac{1}{2} \begin{bmatrix} w_i & v_i \end{bmatrix} \begin{bmatrix} 1 & \tilde{c}_i^{(N)} \\ \tilde{c}_i^{(N)} & 1 \end{bmatrix}^{-1} \begin{bmatrix} w_i \\ v_i \end{bmatrix}}{\sqrt{(2\pi)^2 \begin{vmatrix} 1 & \tilde{c}_i^{(N)} \\ \tilde{c}_i^{(N)} & 1 \end{vmatrix}}}. \quad (2.3.4.2)$$

We are now in a position to conduct a binary hypothesis test based on the confidence score given below:

$$score_G = \log \frac{\otimes_{i=1}^n P_M(w_i, v_i)}{\otimes_{i=1}^n P_N(w_i, v_i)} = \sum_{i=1}^k \log \frac{P_M(w_i, v_i)}{P_N(w_i, v_i)}. \quad (2.3.4.3)$$

Substituting Equations (2.3.4.1) and (2.3.4.2) into Equation (2.3.4.3) gives

$$\begin{aligned} score_G &= \sum_{i=1}^k (\log P_M(w_i, v_i) - \log P_N(w_i, v_i)) \\ &= \sum_{i=1}^k \left( -\frac{w_i^2 - 2w_i v_i \tilde{c}_i^{(M)} + v_i^2}{2\pi \sqrt{(1 - (\tilde{c}_i^{(M)})^2)}} + \frac{w_i^2 - 2w_i v_i \tilde{c}_i^{(N)} + v_i^2}{2\pi \sqrt{(1 - (\tilde{c}_i^{(N)})^2)}} + \log \frac{\sqrt{1 - (\tilde{c}_i^{(N)})^2}}{\sqrt{1 - (\tilde{c}_i^{(M)})^2}} \right), \end{aligned}$$

which is equivalent to

$$\sum_{i=1}^k \left( -\frac{w_i^2 - 2w_i v_i \tilde{c}_i^{(M)} + v_i^2}{\sqrt{(1 - (\tilde{c}_i^{(M)})^2)}} + \frac{w_i^2 - 2w_i v_i \tilde{c}_i^{(N)} + v_i^2}{\sqrt{(1 - (\tilde{c}_i^{(N)})^2)}} \right) \quad (2.3.4.4)$$

as the log term and the scalar  $2\pi$  have no effect on rankings. This confidence score reflects the degree of similarity between the two given images. The higher the score is, the more likely the images match each other.

## 2.4 Experimental Results

### 2.4.1 Training Datasets

We resort to two datasets for training, namely, 120k-Structure from Motion (120k-SfM) and 30k-Structure from Motion (30k-SfM) [35]. Both are preprocessed to eliminate overlaps with the evaluation datasets. A succinct description of these two datasets can be found below:

**120k-Structure from Motion** 120k-Structure from Motion (120k-SfM) dataset is constructed from the one used in the work of Schonberger et al. [36], which contains 713 3D models with nearly 120k images. The maximum size of each image is  $1024 \times 1024$ . The original dataset includes all image from Oxford5k and Paris6k. Those images are removed to avoid overlaps (in total 98 clusters are eliminated).

**30k-Structure from Motion** 30k-Structure from Motion (30k-SfM) dataset is a subset of 120k-SfM, which contains approximately 30k images and 551 classes. The maximum size of images are resized to  $362 \times 362$ .

Each dataset serves its own purpose; 30k-SfM is a small dataset while 120k-SfM is a



























big one. This enables us to investigate the pros and cons of different datasets in terms of their sizes. Compared to 30k-SfM, 120k-SfM supplies richer features to be explored by the methods being tested.

### **2.4.2 Training Details**

Using each dataset, two lists of matching and non-matching pairs of images are created for training—feature space analysis not CNN training. Table 2.1 shows some examples of matching and non-matching pairs. Specifically, we randomly select 10,960 raw pairs from 30k-SfM and 58,502 raw pairs from 120k-SfM. We double the number matching and non-matching pairs by simultaneously using each raw pair and its flipped version to ensure that the diagonal/off-diagonal blocks of the data covariances in Equations (2.3.2.1) and (2.3.2.2) are identical and symmetric. This could also be seen from Table 2.1: each pair is used twice but with its image order flipped.

Table 2.1: Examples of matching/non-matching pairs.

Matching Pair						
						
Non-Matching Pair						
						

The feature vector of a given image is extracted from the very last pooling layer of a pre-trained VGG16 via one of the following three pooling strategies: Global Max (MAC) pooling, Global Average (AVE) pooling, and global Standard Deviation (SD) pooling (global Max (MAC) pooling, Global Average (AVE) pooling, and global Standard Deviation (SD) pooling compute, respectively, the maximum value, the average value, and the standard deviation of the feature map in each channel). We conducted separate training for each of these strategies in order to compare performances.

For benchmarking, the proposed method (G-CCA) and its variant (S-CCA) were trained along with three alternative feature-space analysis methods, i.e., PCAw [21], Supervised PCA (SPCA) [37] and Multiclass LDA (MLDA) [38]. G-CCA is depicted in Figure 2.2 while S-CCA is the same as G-CCA except that in the final step the scalar similarity measure is used instead (namely, in the last block of Figure 2.2,  $score_G$  is replaced with  $score_S = \mathbf{w}^T \cdot \mathbf{v}$ ). PCAw infers a basis matrix of the feature space from the covariance matrix of the training image features. This basis matrix is used to whiten and compress new image features, which are then leveraged to make a matching/non-matching decision based on the scalar similarity measure. See [12] for a detailed description of the PCAw method and its performance. Furthermore, we compared the proposed method with SPCA, which is a weighted PCA method. It uses a Laplacian matrix to characterize the relationship among the classes in the dataset. We implement SPCA by following the steps in [39]. As to LDA [40], its application to image retrieval has also been thoroughly investigated [41], which is hardly surprising given its popularity in statistical analysis. Here we use its variant MLDA [38] as a competing feature-space analysis method. MLDA is trained using the classes provided by both training datasets. It derives a set of projection vectors that offer the best linear separation of the classes (full separation is achievable if the classes are

linearly separable, otherwise, MLDA produces some overlaps between the classes). These projection vectors are employed to transform and compress (in the sense of dimensionality reduction) new feature vectors. Scalar similarity is then evaluated for the transformed features to determine whether or not they match.

### 2.4.3 Implementation Details

In the experiment, we compare G-CCA, S-CCA with PCAw, SPCA, and MLDA. The G-CCA and S-CCA are presented in this paper while the PCAw, SPCA, and MLDA are implemented by following procedures in [21,38,39]. Here, we discuss some detailed issues in the implementation.

Firstly, S-CCA, PCAw, SPCA, and MLDA use scalar similarity score to calculate the confidence score while G-CCA uses the proposed score in Equation (2.3.4.4). Secondly, for all these methods, the feature vectors are obtained via MAC, AVE, and SD pooling, and centralization and normalization are performed. Thirdly, the performance comparisons are conducted for eight dimensions: 512, 450, 400, 300, 200, 100, 50, and 25. Lastly, we calculate the scores between the query image and each image in the test dataset, and obtain the image retrieval results by ranking scores from high to low. All the methods are evaluated by the mean Average Precision (mAP) (we calculate the mAP without enforcing the monotonicity for Precision (Recall) relationship). which can be formulated as follows:

$$\text{mAP} = \frac{\sum_{i=1}^m \text{AP}_i}{m} \quad \text{with } \text{AP}_i = \sum_{k=1}^n P(k) \Delta r(k),$$

where  $\text{AP}_i$  is the average precision for the  $i$ -th query image,  $m$  is the total number of query images, and  $n$  is the total number of images in the testing dataset,  $P(k)$  is the precision of

top  $k$  results, and  $\Delta r(k) = R(k) - R(k-1)$  with  $R(k)$  being the recall of top  $k$  results. For calculating the precision  $P(K)$  and recall  $R(k)$ , the positive labels for each query image are provided by the test datasets.

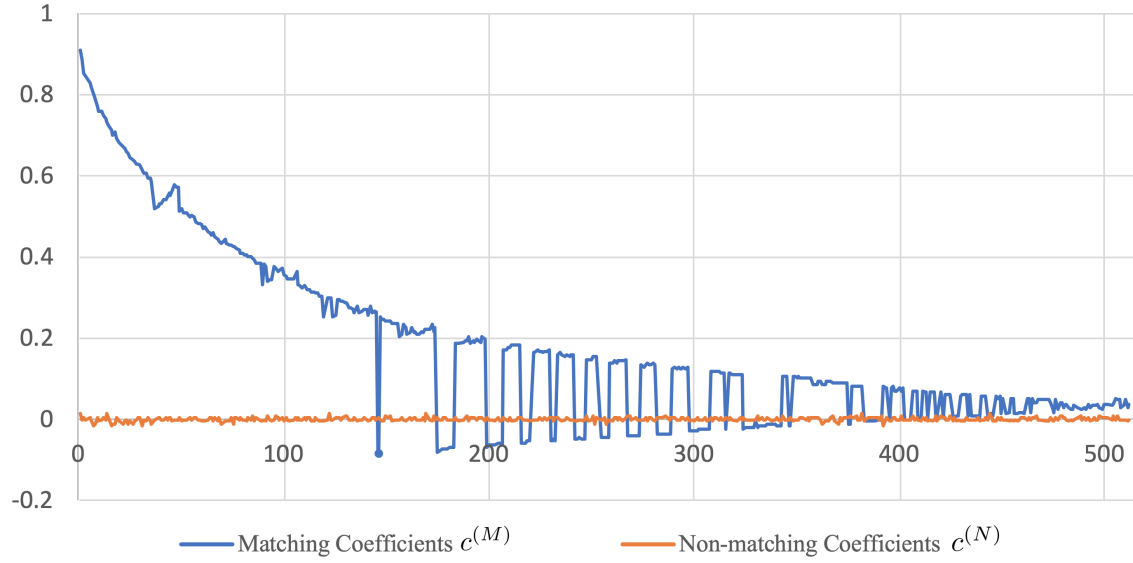


Figure 2.3: Profile of the diagonal elements of  $\Lambda$  and  $\Pi$  (i.e.,  $c_t^{(M)}$  and  $c_t^{(N)}$ , where  $t \in \{1, 2, \dots, 512\}$ ) using AVE features. The CCA training was performed on the 120k-SfM dataset.

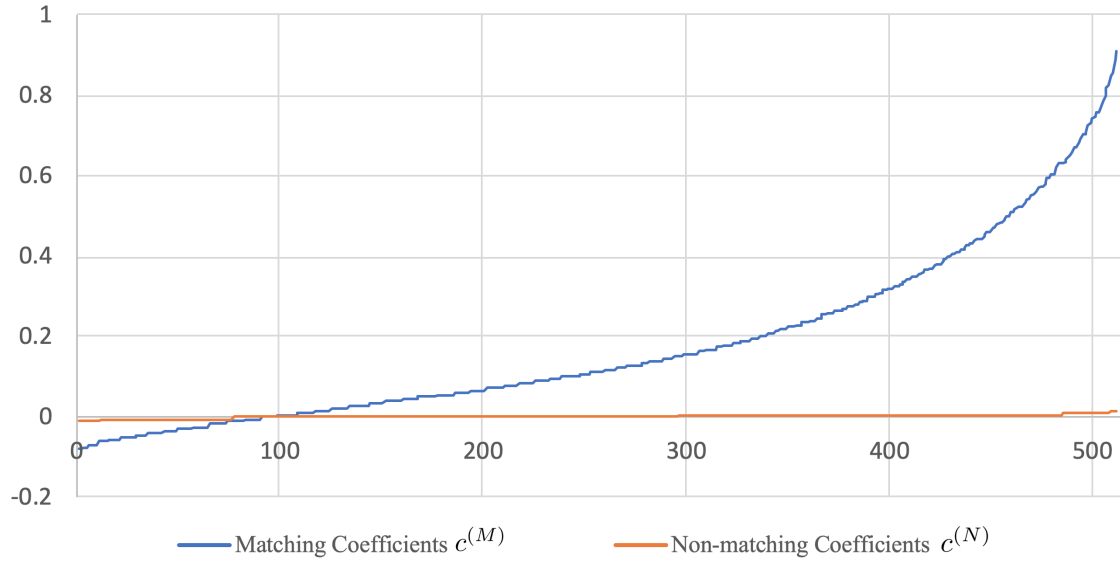


Figure 2.4: Profile of sorted diagonal elements of  $\Lambda$  and  $\Pi$  (i.e.,  $\tilde{c}_t^{(M)}$  and  $\tilde{c}_t^{(N)}$ , where  $t \in \{1, 2, \dots, 512\}$ ) using AVE features. The CCA training was performed on the 120k-SfM dataset.

## 2.4.4 Evaluation Datasets and Details

Four datasets, namely, Oxford5k [42], Paris6k [44],  $\mathcal{R}$ Oxford [43], and  $\mathcal{R}$ Paris [43], are used to assess the performance of each retrieval method. As the first two datasets are contained in the large raw 120k-SfM dataset, they are excluded from the training dataset via preprocessing. The last two datasets contain new annotations and more difficult query images, and consequently create more challenges for image retrieval; therefore, they can help test the reliability of our approach. A short description of each dataset is given below.

**Oxford5k** Oxford5k dataset contains 5063 images and 55 query images for 11 different buildings. It is annotated with bounding boxes for the main objects.

**$\mathcal{R}$ Oxford**  $\mathcal{R}$ Oxford dataset contains 4993 images and 70 query images for 11 different buildings. Query images are excluded from the retrieval images. Same as Oxford5k, it is annotated with bounding boxes for the main objects.

**Paris6k** Paris6k dataset contains 6412 images and 55 query images for 11 different buildings. It is also annotated with bounding boxes.

**$\mathcal{R}$ Paris**  $\mathcal{R}$ Paris dataset contains 6322 images and 70 query images for 11 different buildings. Query images are excluded from the retrieval images. Same as Paris6k, it is annotated with bounding boxes.

The performance of each retrieval method is evaluated using mean Average Precision (mAP) [42]. The positive labels of each query image are provided by the datasets. The standard evaluation protocol is followed for Oxford5k and Paris6k. As for the  $\mathcal{R}$ Oxford and  $\mathcal{R}$ Paris datasets, the medium protocol setups in [43] are adopted. We crop all the query images with the provided bounding boxes before feeding them to VGG16. Each method undergoes training and evaluation twice. The first training used the small dataset, 30k-SfM, followed by evaluation. Then it was trained with the large dataset, 120k-SfM,



before evaluation. This enables us to study the effect of dataset size and diversity on the methods under comparison.

### 2.4.5 Performance Evaluation and Analysis

Before getting into the performance evaluation of the proposed method, it is useful to have some insights about how discriminative the canonical vectors are. Figures 2.3 and 2.4 show the profile of the diagonal elements of the off-diagonal blocks in Equations (2.3.2.3) and (2.3.2.4). It can be seen that the values of  $c_t^{(N)}$  fluctuate around zero whereas those of  $c_t^{(M)}$  range between  $-0.1$  and  $0.9$ . This observation suggests that there exists a set of canonical vectors that can effectively tell apart matching from non-matching pairs of images. This is shown in the rest of this subsection.

Table 2.2 reports the baseline performances of MAC, AVE, and SD without dimensionality reduction. Specifically, for these baselines, we directly calculate the scalar similarity between the pooling features (after centralization and normalization) of the query image and each image in the testing dataset. In the evaluation, we consider the proposed method (G-CCA) and its variant with Gaussian-distribution-based hypothesis testing replaced by scalar similarity (S-CCA). From Table 2.2, we observe that G-CCA achieves better performance than S-CCA in most cases except for Paris6k and AVE on  $\mathcal{R}$ Paris.

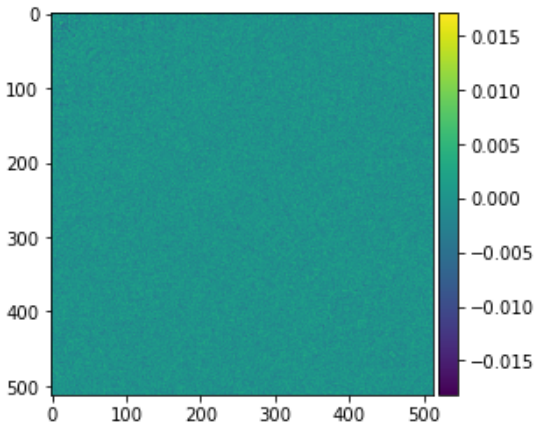


Figure 2.5: 2D visualization of matrix  $\Pi$ .

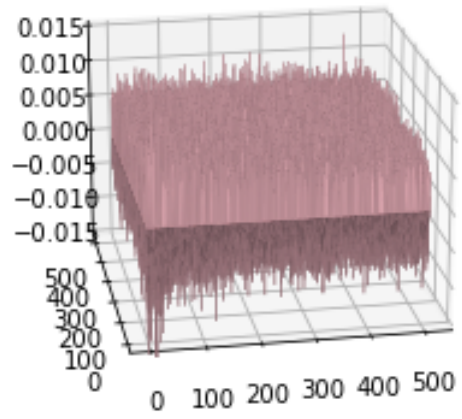


Figure 2.6: 3D visualization of matrix  $\Pi$ .

Table 2.2: Performance comparison of the baseline, S-CCA, and G-CCA on Oxford5k,  $\mathcal{R}$ Oxford, Paris6k, and  $\mathcal{R}$ Paris without dimension reduction.

Method	Oxford5k	$\mathcal{R}$ Oxford	Paris6k	$\mathcal{R}$ Paris
MAC	0.5296	0.3295	0.7455	0.5122
S-CCA + MAC	0.5800	0.3575	<b>0.7726</b>	0.5408
G-CCA + MAC	<b>0.6275</b>	<b>0.3996</b>	0.7455	<b>0.5939</b>
AVE	0.5312	0.2884	0.6467	0.4653
S-CCA + AVE	0.6845	0.4303	<b>0.7845</b>	<b>0.5936</b>
G-CCA + AVE	<b>0.7146</b>	<b>0.4444</b>	0.7507	0.5812
SD	0.6095	0.3834	0.7355	0.5311
S-CCA + SD	0.6943	0.4503	<b>0.8191</b>	0.6199
G-CCA + SD	<b>0.7419</b>	<b>0.4806</b>	0.8164	<b>0.6403</b>

<sup>1</sup> The evaluation results are based on 120k-SfM. <sup>2</sup> For the same type of features, the best performances are highlighted in **bold**.

By considering three different pooling strategies, three image retrieval methods are trained on the 30k-SfM dataset and evaluated on all four test sets. Table 2.3 provides a comprehensive depiction of the experimental results for each retrieval method with different pooling strategies and feature dimensionality choices (compression levels). The results for MLDA are not reported there, for MLDA cannot be trained on the 30k-SfM dataset, which is a consequence of the fact that the difference between classes is too small as far as MLDA training is concerned. From Table 2.3, four observations can be made. The first is regarding the effect of the pooling strategy. Specifically, SD pooling appears to result in the most competitive performance for all methods at every choice of feature dimensionality. The reason behind this phenomenon is that the standard deviation can be considered as square root of the power of signal in discrete time domain, and provide more representative information compared with MAC and AVE. while MAC renders G-CCA superior to SPCA and PCAw at low dimensions over all test sets. The second observation is that for MAC, AVE, and SD pooling strategies, the proposed method outperforms PCAw at low feature dimensionality. As such, the proposed method is a better choice for producing compact features than PCAw regardless of the pooling strategy. The last observation is that G-CCA is more robust against dimensionality reduction than S-CCA.

The performance of the proposed method can be improved by replacing 30k-SfM with 120k-SfM, which is a larger training set. Table 2.4 shows the corresponding evaluation results for all the methods with different pooling strategies and dimensionality choices (the only exception is SPCA for which the training on 120k-SfM is computationally infeasible as its Laplacian matrix is too large to be stored on our computer). It is clear that the increased-size training set leads to an improved mAP performance on all test sets and for all pooling strategies. It is also interesting to note that the proposed method outperforms

all others on Oxford5k. This uniform superiority across all dimensions is only attained on Paris6k using SD pooling. Although AVE and MAC improve mAP, they cause G-CCA to lose its edge at high dimensions on Paris6k. In contrast, with SD pooling, the proposed method retains its dominating performance at all feature dimensions. On  $\mathcal{R}$ Oxford and  $\mathcal{R}$ Paris, the performance of G-CCA is better than MLDA at almost all dimensions with MAC. G-CCA almost outperforms PCAw in every dimensions with all three pooling strategies.

Based on Tables 2.3 and 2.4, there are three notable advantages of G-CCA over MLDA, PCAw, and SPCA. The first is that the CCA-based methods can be trained using datasets with small differences between classes whereas MLDA cannot be trained on such datasets. The second advantage is that G-CCA typically shows a more graceful performance degradation than PCAw after dimensionality reduction. The last is that SPCA can not be trained on large datasets as compared with G-CCA.

Tables 2.5–2.7 present some retrieval results for visual illustration. In Table 2.5, a query image from the Oxford5k set is presented to PCAw, SPCA, and G-CCA, trained on the 30k-SfM set, while in Tables 2.6 and 2.7, a query image from the Oxford5k set is presented to PCAw, MLDA, and G-CCA, trained on the 120k-SfM set. We list top 10 matches for each method with each list ranked using the matching score associated with the corresponding method. Tables 2.5 and 2.6 show the top 10 retrieved images for different methods with SD pooling while Table 2.7 gives examples for G-CCA with different pooling strategies.

Table 2.3: Evaluation results from 30k-SfM on Oxford5k,  $\mathcal{R}$ Oxford, Paris6k, and  $\mathcal{R}$ Paris.

	Dim	MAC				AVE				SD			
		SPCA	PCAw	S-CCA	G-CCA	SPCA	PCAw	S-CCA	G-CCA	SPCA	PCAw	S-CCA	G-CCA
Oxford5k	25	0.3589	0.3555	0.2431	<b>0.3873</b>	0.4474	0.4443	0.3091	<b>0.4873</b>	0.4757	0.4838	0.3439	<b>0.4979</b>
	50	0.4412	0.4258	0.3174	<b>0.4487</b>	0.4930	0.4933	0.3782	<b>0.5127</b>	0.5086	0.5074	0.4403	<b>0.5690</b>
	100	0.5016	0.5027	0.4122	<b>0.5043</b>	0.5599	0.5697	0.5447	<b>0.6034</b>	0.6002	0.6041	0.5191	<b>0.6164</b>
	200	<b>0.5628</b>	0.5583	0.4818	0.5501	0.6083	0.6086	0.6157	<b>0.6445</b>	0.6635	0.6619	0.6280	<b>0.6772</b>
	300	<b>0.5723</b>	0.5672	0.5280	0.5379	0.6416	0.6307	0.6428	<b>0.6552</b>	0.6753	0.6736	0.6513	<b>0.6830</b>
	400	<b>0.5728</b>	0.5715	0.5505	0.5405	0.6517	0.6385	0.6373	<b>0.6525</b>	<b>0.6811</b>	<b>0.6811</b>	0.6703	0.6745
	450	<b>0.5670</b>	0.5654	0.5609	0.5364	<b>0.6544</b>	0.6422	0.6393	0.6538	0.6839	<b>0.6849</b>	0.6740	0.6746
	512	<b>0.5615</b>	0.5601	0.5580	0.5363	0.6506	0.6388	0.6493	<b>0.6537</b>	<b>0.6766</b>	0.6763	0.6764	0.6743
$\mathcal{R}$ Oxford	25	0.2070	0.2226	0.1495	<b>0.2276</b>	0.2702	<b>0.2709</b>	0.1939	0.2590	0.2883	0.2856	0.2116	<b>0.3031</b>
	50	0.2823	0.2771	0.1886	<b>0.2914</b>	0.2731	0.2757	0.2206	<b>0.2876</b>	0.3117	0.3123	0.2590	<b>0.3485</b>
	100	0.3259	0.3281	0.2484	<b>0.3282</b>	0.3304	0.3197	0.3083	<b>0.3372</b>	<b>0.3885</b>	0.3795	0.3007	0.3848
	200	0.3462	0.3545	0.3071	<b>0.3569</b>	0.3569	0.3531	0.3759	<b>0.4002</b>	0.4399	0.4368	0.4021	<b>0.4417</b>
	300	<b>0.3595</b>	0.3593	0.3290	0.3413	0.3901	0.3771	0.3911	<b>0.4057</b>	<b>0.4507</b>	0.4420	0.4173	0.4484
	400	<b>0.3576</b>	0.3568	0.3424	0.3400	0.3905	0.3796	0.3798	<b>0.4065</b>	0.4526	0.4381	0.4454	<b>0.4538</b>
	450	<b>0.3551</b>	0.3544	0.3466	0.3398	0.4002	0.3772	0.3876	<b>0.4052</b>	0.4498	0.4382	0.4435	<b>0.4499</b>
	512	0.3442	<b>0.3469</b>	0.3444	0.3396	0.4042	0.3767	0.3963	<b>0.4077</b>	0.4417	0.4383	0.4412	<b>0.4419</b>
Paris	25	0.4878	0.5084	0.4133	<b>0.5464</b>	0.4944	0.4330	0.4182	<b>0.4990</b>	0.5633	0.5858	0.4758	<b>0.5969</b>
	50	0.6027	0.6208	0.5391	<b>0.6347</b>	0.5692	0.5893	0.5898	<b>0.6153</b>	0.6415	0.6555	0.6084	<b>0.6746</b>
	100	0.6691	0.6750	0.5848	<b>0.6808</b>	0.6441	0.6736	0.6559	<b>0.6790</b>	0.7290	0.7267	0.6988	<b>0.7426</b>
	200	0.7035	0.6942	0.6384	<b>0.7166</b>	0.6931	0.6994	0.7049	<b>0.7106</b>	0.7719	0.7620	0.7501	<b>0.7811</b>
	300	0.7004	0.6980	0.6701	<b>0.7067</b>	0.7109	<b>0.7328</b>	0.7297	0.7118	0.7834	0.7819	0.7739	<b>0.7892</b>
	400	<b>0.7076</b>	0.7057	0.6893	0.7052	0.7375	<b>0.7586</b>	0.7418	0.7120	<b>0.8010</b>	0.7970	0.7885	0.7867
	450	<b>0.7091</b>	0.7073	0.6964	0.7027	0.7482	<b>0.7679</b>	0.7472	0.7130	<b>0.8067</b>	0.8066	0.7969	0.7871
	512	<b>0.7032</b>	0.7060	0.7039	0.7029	0.7508	<b>0.7732</b>	0.7520	0.7133	0.8020	0.8031	<b>0.8036</b>	0.7874
$\mathcal{R}$ Paris	25	0.3966	0.3944	0.3225	<b>0.4212</b>	0.3877	0.3981	0.3085	<b>0.4212</b>	0.4361	0.4410	0.3602	<b>0.4433</b>
	50	0.4725	0.4738	0.4063	<b>0.4781</b>	0.4354	0.4442	0.4385	<b>0.4538</b>	0.5006	0.5015	0.4524	<b>0.5056</b>
	100	0.5007	0.5021	0.4311	<b>0.5106</b>	0.4820	0.4886	0.4946	<b>0.5082</b>	0.5457	0.5501	0.5258	<b>0.5653</b>
	200	0.5183	0.5182	0.4668	<b>0.5370</b>	0.5118	0.5129	0.5302	<b>0.5355</b>	0.5822	0.5827	0.5635	<b>0.5985</b>
	300	0.5206	0.5200	0.4894	<b>0.5285</b>	0.5281	0.5306	<b>0.5507</b>	0.5377	0.5966	0.5964	0.5829	<b>0.6045</b>
	400	0.5224	0.5219	0.5040	<b>0.5272</b>	0.5504	0.5507	<b>0.5577</b>	0.5379	0.6064	<b>0.6070</b>	0.5958	0.6024
	450	0.5222	0.5200	0.5109	<b>0.5255</b>	0.5587	0.5590	<b>0.5620</b>	0.5383	0.6119	<b>0.6121</b>	0.6013	0.6027
	512	0.5169	0.5168	0.5154	<b>0.5256</b>	0.5579	0.5588	<b>0.5646</b>	0.5384	0.6051	<b>0.6067</b>	0.6048	0.6028







<sup>1</sup> The best performances in each dimension are highlighted in bold.

Table 2.4: Evaluation results from 120k-SfM on Oxford5k,  $\mathcal{R}$ Oxford, Paris6k, and  $\mathcal{R}$ Paris.

	Dim	MAC				AVE				SD			
		MLDA	PCAw	S-CCA	G-CCA	MLDA	PCAw	S-CCA	G-CCA	MLDA	PCAw	S-CCA	G-CCA
Oxford5k	25	0.3603	0.3906	0.2677	<b>0.4019</b>	0.4758	0.4266	0.2644	<b>0.4821</b>	0.4759	0.4790	0.3400	<b>0.5212</b>
	50	0.4760	0.4319	0.3802	<b>0.4987</b>	<b>0.5612</b>	0.5033	0.4293	0.5572	0.5375	0.5355	0.4667	<b>0.5956</b>
	100	0.5157	0.5275	0.4537	<b>0.5481</b>	0.6017	0.5756	0.5529	<b>0.6402</b>	0.6429	0.6240	0.5593	<b>0.6688</b>
	200	0.5887	0.5453	0.5562	<b>0.6231</b>	0.6571	0.6437	0.6498	<b>0.6964</b>	0.6861	0.6410	0.6620	<b>0.7244</b>
	300	0.6028	0.5669	0.5697	<b>0.6306</b>	0.6643	0.6474	0.6658	<b>0.7102</b>	0.7030	0.6711	0.6754	<b>0.7382</b>
	400	0.5974	0.5810	0.5768	<b>0.6275</b>	0.6688	0.6681	0.6758	<b>0.7139</b>	0.7020	0.6970	0.6864	<b>0.7422</b>
	450	0.5939	0.5840	0.5820	<b>0.6279</b>	0.6678	0.6728	0.6781	<b>0.7144</b>	0.6972	0.6986	0.6939	<b>0.7412</b>
	512	0.5868	0.5799	0.5800	<b>0.6275</b>	0.6613	0.6711	0.6845	<b>0.7146</b>	0.6958	0.6946	0.6943	<b>0.7419</b>
$\mathcal{R}$ Oxford	Dim	MAC				AVE				SD			
	25	0.2330	<b>0.2503</b>	0.1543	0.2459	<b>0.2712</b>	0.2533	0.1422	0.2441	0.2666	<b>0.3037</b>	0.1782	0.2853
	50	0.2989	0.2664	0.2366	<b>0.3025</b>	<b>0.3522</b>	0.2802	0.2337	0.3366	<b>0.3418</b>	0.3357	0.2636	0.3254
	100	<b>0.3470</b>	0.3521	0.2724	0.3437	0.3981	0.3318	0.3412	<b>0.4290</b>	0.4002	0.4075	0.3269	<b>0.4073</b>
	200	0.3924	0.3510	0.3482	<b>0.3991</b>	0.4324	0.3911	0.3913	<b>0.4411</b>	0.4497	0.4192	0.4085	<b>0.4622</b>
	300	<b>0.4006</b>	0.3557	0.3497	0.3986	0.4404	0.3920	0.4056	<b>0.4430</b>	0.4645	0.4454	0.4335	<b>0.4796</b>
	400	0.3964	0.3625	0.3526	<b>0.4001</b>	0.4412	0.4106	0.4215	<b>0.4462</b>	0.4673	0.4609	0.4429	<b>0.4812</b>
	450	0.3941	0.3613	0.3587	<b>0.3998</b>	0.4363	0.4159	0.4215	<b>0.4443</b>	0.4624	0.4604	0.4394	<b>0.4807</b>
512	0.3881	0.3570	0.3575	<b>0.3996</b>	0.4267	0.4136	0.4303	<b>0.4444</b>	0.4597	0.4501	0.4503	<b>0.4806</b>	
Paris6k	Dim	MAC				AVE				SD			
	25	0.5781	0.4878	0.5109	<b>0.6270</b>	0.5553	0.5013	0.4442	<b>0.5693</b>	0.6204	0.5543	0.5269	<b>0.6611</b>
	50	0.6384	0.6153	0.5416	<b>0.6679</b>	0.6362	0.5893	0.5467	<b>0.6314</b>	0.6900	0.6575	0.5935	<b>0.6968</b>
	100	0.6916	0.6788	0.6226	<b>0.7339</b>	0.6994	0.6736	0.6657	<b>0.6910</b>	0.7502	0.7313	0.7105	<b>0.7641</b>
	200	0.7244	0.7124	0.6765	<b>0.7674</b>	0.7162	0.6994	0.7220	<b>0.7491</b>	0.7845	0.7842	0.7760	<b>0.8043</b>
	300	0.7493	0.7214	0.6900	<b>0.7719</b>	0.7299	0.7328	<b>0.7538</b>	0.7491	0.8030	0.8046	0.7973	<b>0.8160</b>
	400	0.7548	0.7230	0.7146	<b>0.7729</b>	0.7247	0.7586	<b>0.7729</b>	0.7507	0.8042	0.8143	0.8067	<b>0.8164</b>
	450	0.7540	0.7222	<b>0.7729</b>	0.7455	0.7197	0.7679	<b>0.7775</b>	0.7508	0.8003	0.8144	0.8096	<b>0.8161</b>
512	0.7549	0.7288	<b>0.7726</b>	0.7455	0.7111	0.7732	<b>0.7845</b>	0.7507	0.7971	0.8159	0.8164	<b>0.8191</b>	
$\mathcal{R}$ Paris	Dim	MAC				AVE				SD			
	25	0.4321	0.3728	0.3956	<b>0.4787</b>	0.4524	0.3745	0.3607	<b>0.4455</b>	0.4817	0.4136	0.4075	<b>0.5032</b>
	50	0.4910	0.4685	0.4214	<b>0.5156</b>	<b>0.4944</b>	0.4495	0.4229	0.4877	0.5373	0.4998	0.4611	<b>0.5415</b>
	100	0.5339	0.5096	0.4681	<b>0.5631</b>	0.5003	0.5101	0.5052	<b>0.5340</b>	0.5796	0.5596	0.5472	<b>0.5970</b>
	200	0.5526	0.5346	0.5066	<b>0.5910</b>	0.5656	0.5310	0.5437	<b>0.5678</b>	0.6066	0.6002	0.5928	<b>0.6317</b>
	300	0.5520	0.5425	0.5124	<b>0.5942</b>	0.5809	0.5566	0.5639	<b>0.5799</b>	0.6195	0.6156	0.6021	<b>0.6408</b>
	400	0.5437	0.5406	0.5282	<b>0.5941</b>	0.5843	0.5738	0.5824	<b>0.5857</b>	0.6165	0.6228	0.6072	<b>0.6401</b>
	450	0.5399	0.5369	0.5313	<b>0.5941</b>	0.5829	0.5796	<b>0.5844</b>	0.5813	0.6136	0.6187	0.6118	<b>0.6401</b>
512	0.5333	0.5387	0.5408	<b>0.5939</b>	0.5830	0.5828	<b>0.5936</b>	0.5812	0.6093	0.6178	0.6199	<b>0.6403</b>	

<sup>1</sup> The best performances in each dimension are highlighted in bold.

Table 2.5: Image retrieval comparison of PCAw, SPCA, and G-CCA.





Query	TOP 10 Retrieved Images
A	 
B	 
C	 

<sup>1</sup> Top 10 retrieved images from Oxford5k. (A) SD + PCAw. (B) SD + SPCA (C) SD + G-CCA.

<sup>2</sup> Correct images are bounded with green boxes, wrong images are bounded with red boxes.































Table 2.6: Image retrieval comparison of PCAw, SPCA, and G-CCA.

Query	TOP 10 Retrieved Images
A 	
B 	
C 	

<sup>1</sup> Top 10 retrieved images from Oxford5k. (A) SD + PCAw. (B) SD + SPCA (C) SD + G-CCA.

<sup>2</sup> Correct images are bounded with green boxes, wrong images are bounded with red boxes.

Table 2.7: Image retrieval comparison of PCAw, MLDA, and G-CCA.

Query	TOP 10 Retrieved Images
A	          
B	          
C	          

<sup>1</sup> Top 10 retrieved images from Oxford5k. (A) SD + PCAw. (B) SD + MLDA (C) SD + G-CCA.

<sup>2</sup> Correct images are bounded with green boxes, wrong images are bounded with red boxes.

## 2.5 Conclusions

In view of the success of DL in image classification, a CCA-based method is proposed to exploit DL features for image retrieval applications. By adopting an OTS CNN without fine-tuning, it achieves good retrieval accuracy with a minimal computational overhead. As shown by the experimental results on standard evaluation datasets, the proposed method is performance-wise competitive against traditional and other OTS-CNN-based methods. Moreover, it exhibits improved robustness against dimensionality reduction and enhanced sensitivity to feature mismatch.

## 2.A Chernoff Information Between Two 2-Dimensional Gaussian Distributions

For notational simplicity, we suppress subscript  $t$  in the following derivation. Consider

$$\mathbf{S}^{(M)} = \begin{bmatrix} 1 & c^{(M)} \\ c^{(M)} & 1 \end{bmatrix}, \quad \mathbf{S}^{(N)} = \begin{bmatrix} 1 & c^{(N)} \\ c^{(N)} & 1 \end{bmatrix},$$

where  $c^{(M)}$  and  $c^{(N)}$  are two corresponding coefficients. Let  $\mathbf{S}^{(\lambda)} = (\lambda(\mathbf{S}^{(M)})^{-1} + (1 - \lambda)(\mathbf{S}^{(N)})^{-1})^{-1}$ ,  $\lambda \in [0, 1]$ . Now we proceed to find the solution  $\lambda = \lambda^*$  of the equation  $D(\mathbf{S}^{(\lambda)}||\mathbf{S}^{(M)}) = D(\mathbf{S}^{(\lambda)}||\mathbf{S}^{(N)})$ .

Note that

$$\begin{aligned} D(\mathbf{S}^{(\lambda)}||\mathbf{S}^{(M)}) &= D(\mathbf{S}^{(\lambda)}||\mathbf{S}^{(N)}) \\ \Leftrightarrow \log_e \frac{|\mathbf{S}^{(M)}|}{|\mathbf{S}^{(N)}|} &= \text{tr}(((\mathbf{S}^{(N)})^{-1} - (\mathbf{S}^{(M)})^{-1})\mathbf{S}^{(\lambda)}). \end{aligned}$$

We have

$$\begin{aligned} &((\mathbf{S}^{(N)})^{-1} - (\mathbf{S}^{(M)})^{-1})\mathbf{S}^{(\lambda)} \\ &= \frac{1}{\lambda}((\lambda(\mathbf{S}^{(M)})^{-1}\mathbf{S}^{(N)} + (1 - \lambda)\mathbf{I})^{-1} - \mathbf{I}). \end{aligned}$$

It can be verified that

$$\begin{aligned}
& (\lambda(\mathbf{S}^{(M)})^{-1}\mathbf{S}^{(N)} + (1 - \lambda)\mathbf{I})^{-1} \\
&= \frac{1}{\theta} \begin{bmatrix} \frac{\lambda(1-c^{(M)}c^{(N)})}{1-(c^{(M)})^2} + 1 - \lambda & \frac{\lambda(c^{(M)}-c^{(N)})}{1-(c^{(M)})^2} \\ \frac{\lambda(c^{(M)}-c^{(N)})}{1-(c^{(M)})^2} & \frac{\lambda(1-c^{(M)}c^{(N)})}{1-(c^{(M)})^2} + 1 - \lambda \end{bmatrix},
\end{aligned}$$

where

$$\theta = -\frac{(c^{(N)} - c^{(M)})^2}{1 - (c^{(M)})^2}\lambda^2 + \frac{2c^{(M)}(c^{(M)} - c^{(N)})}{1 - (c^{(M)})^2}\lambda + 1.$$

As a consequence,

$$\begin{aligned}
& \text{tr}(((\mathbf{S}^{(N)})^{-1} - (\mathbf{S}^{(M)})^{-1})\mathbf{S}^{(\lambda)}) \\
&= \frac{2}{\theta} \left( \frac{(c^{(N)} - c^{(M)})^2}{1 - (c^{(M)})^2}\lambda + \frac{c^{(M)}(c^{(N)} - c^{(M)})}{1 - (c^{(M)})^2} \right).
\end{aligned}$$

Therefore,  $\lambda = \lambda^*$  is a root in  $[0, 1]$  of the following quadratic equation:

$$\alpha\lambda^2 + \beta\lambda + \gamma = 0, \tag{2.A.0.1}$$

where

$$\begin{aligned}\alpha &= \frac{(c^{(N)} - c^{(M)})^2}{1 - (c^{(M)})^2} \log_e \frac{|\mathbf{S}^{(M)}|}{|\mathbf{S}^{(N)}|}, \\ \beta &= \frac{2(c^{(N)} - c^{(M)})^2}{1 - (c^{(M)})^2} - \frac{2c^{(M)}(c^{(M)} - c^{(N)})}{1 - (c^{(M)})^2} \log_e \frac{|\mathbf{S}^{(M)}|}{|\mathbf{S}^{(N)}|}, \\ \gamma &= \frac{2c^{(M)}(c^{(N)} - c^{(M)})}{1 - (c^{(M)})^2} - \log_e \frac{|\mathbf{S}^{(M)}|}{|\mathbf{S}^{(N)}|}.\end{aligned}$$

We shall show that Equation (2.A.0.1) has a unique root in  $[0, 1]$ , which is given by

$$\lambda^* = \frac{-\beta + \sqrt{\beta^2 - 4\alpha\gamma}}{2\alpha}. \quad (2.A.0.2)$$

Clearly, Equation (2.A.0.1) must have a root in  $[0, 1]$  since  $D(\mathbf{S}^{(\lambda)}||\mathbf{S}^{(M)})|_{\lambda=0} > 0$ ,  $D(\mathbf{S}^{(\lambda)}||\mathbf{S}^{(N)})|_{\lambda=1} > 0$ , and  $D(\mathbf{S}^{(\lambda)}||\mathbf{S}^{(M)})|_{\lambda=1} = D(\mathbf{S}^{(\lambda)}||\mathbf{S}^{(N)})|_{\lambda=0} = 0$ . So it remains to prove the uniqueness of this root.

First consider the case  $(c^{(N)})^2 > (c^{(M)})^2$ . It is clear that  $\alpha > 0$  and

$$\begin{aligned}\gamma &= \frac{2c^{(M)}(c^{(N)} - c^{(M)})}{1 - (c^{(M)})^2} - \log_e \frac{|\mathbf{S}^{(M)}|}{|\mathbf{S}^{(N)}|} \\ &= \frac{2c^{(M)}(c^{(N)} - c^{(M)})}{1 - (c^{(M)})^2} - \log_e \frac{1 - (c^{(M)})^2}{1 - (c^{(N)})^2} \\ &\leq \frac{2c^{(M)}(c^{(N)} - c^{(M)})}{1 - (c^{(M)})^2} - \frac{(c^{(N)})^2 - (c^{(M)})^2}{1 - (c^{(M)})^2} \\ &= -\frac{(c^{(N)} - c^{(M)})^2}{1 - (c^{(M)})^2} \\ &< 0,\end{aligned}$$

where the first inequality is due to  $\log_e x \geq \frac{x-1}{x}$ . Therefore, the two roots of Equation (2.A.0.1) must be of different signs, which implies that there exists a unique root in  $[0, 1]$

with the expression given by Equation (2.A.0.2).

Next consider the case  $(c^{(N)})^2 < (c^{(M)})^2$ . Define  $\bar{\lambda} = 1 - \lambda$ . Equation (2.A.0.1) can be written equivalently as

$$\alpha(1 - \bar{\lambda})^2 + \beta(1 - \bar{\lambda}) + \gamma = 0,$$

i.e.,

$$\alpha\bar{\lambda}^2 - (2\alpha + \beta)\bar{\lambda} + (\alpha + \beta + \gamma) = 0. \quad (2.A.0.3)$$

Note that

$$\begin{aligned} & 2\alpha + \beta \\ &= \frac{2(c^{(N)} - c^{(M)})^2}{1 - (c^{(M)})^2} - \frac{2c^{(N)}(c^{(M)} - c^{(N)})}{1 - (c^{(M)})^2} \log_e \frac{|\mathbf{S}^{(M)}|}{|\mathbf{S}^{(N)}|} \end{aligned}$$

and

$$\begin{aligned} & \alpha + \beta + \gamma \\ &= \frac{2c^{(N)}(c^{(N)} - c^{(M)})}{1 - (c^{(M)})^2} - \frac{1 - (c^{(N)})^2}{1 - (c^{(M)})^2} \log_e \frac{|\mathbf{S}^{(M)}|}{|\mathbf{S}^{(N)}|}. \end{aligned}$$

Therefore, Equation (2.A.0.3) can be rewritten as

$$\bar{\alpha}\bar{\lambda}^2 + \bar{\beta}\bar{\lambda} + \bar{\gamma} = 0, \quad (2.A.0.4)$$

where

$$\begin{aligned} \bar{\alpha} &= \frac{(c^{(M)} - c^{(N)})^2}{1 - (c^{(N)})^2} \log_e \frac{|\mathbf{S}^{(N)}|}{|\mathbf{S}^{(M)}|}, \\ \bar{\beta} &= \frac{2(c^{(M)} - c^{(N)})^2}{1 - (c^{(N)})^2} - \frac{2c^{(N)}(c^{(N)} - c^{(M)})}{1 - (c^{(N)})^2} \log_e \frac{|\mathbf{S}^{(N)}|}{|\mathbf{S}^{(M)}|}, \\ \bar{\gamma} &= \frac{2c^{(N)}(c^{(M)} - c^{(N)})}{1 - (c^{(N)})^2} - \log_e \frac{|\mathbf{S}^{(N)}|}{|\mathbf{S}^{(M)}|}. \end{aligned}$$

A similar argument to that for the case  $(c^{(N)})^2 > (c^{(M)})^2$  can be used to prove that Equation (2.A.0.4) has one root in  $[0, 1]$  and the other root in  $(-\infty, 0)$ . This implies that Equation (2.A.0.1) must have one root in  $[0, 1]$  and the other root in  $(1, \infty)$ ; the one in  $[0, 1]$  must be given by Equation (2.A.0.2) (note that  $\alpha < 0$  when  $(c^{(N)})^2 < (c^{(M)})^2$ ). ...



# Bibliography

- [1] Zhou, W.; Li, H.; Tian, Q. Recent advance in content-based image retrieval: A literature survey. *arXiv* **2017**, arXiv:1706.06064.
- [2] Lowe, D.G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110.
- [3] Ojala, T.; Pietikainen, M.; Maenpaa, T. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. Pattern Anal. Mach. Intell.* **2002**, *24*, 971–987.
- [4] Tan, X.; Triggs, B. Enhanced local texture feature sets for face recognition under difficult lighting conditions. *IEEE Trans. Image Process.* **2010**, *19*, 1635–1650.
- [5] Ojansivu, V.; Heikkilä, J. Blur insensitive texture classification using local phase quantization. In *International Conference on Image and Signal Processing*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 236–243.
- [6] Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. *IEEE Conf. Comput. Vis. Pattern Recognit.* **2005**, *1*, 886–893.
- [7] Batool, A.; Nisar, M.W.; Shah, J.H.; Khan, M.A.; El-Latif, A.A.A. iELMNet: integrating novel improved extreme learning machine and convolutional neural

network model for traffic sign detection. *Big Data* 2022, *ahead of print*, <http://doi.org/10.1089/big.2021.0279>

- [8] Nawaz, M.; Nazir, T.; Javed, A.; Tariq, U.; Yong, H.-S.; Khan, M.A.; Cha, J. An efficient deep learning approach to automatic glaucoma detection using optic disc and optic cup localization. *Sensors* **2022**, *22*, 434.
- [9] Razavian, A.S.; Azizpour, H.; Sullivan, J.; Carlsson, S. CNN features off-the-shelf: an astounding baseline for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, Institute of Electrical and Electronics Engineers: Columbus, OH, USA, 23–28 June, 2014 ; pp. 806–813.
- [10] Khan, M.A.; Muhammad, K.; Sharif, M.; Akram, T.; Kadry, S. Intelligent fusion-assisted skin lesion localization and classification for smart healthcare. *Neural Comput. Appl.* **2021**, 1–16, doi:10.1007/s00521-021-06490-w.
- [11] Tolias, G.; Sivic, R.; Jégou, H. Particular object retrieval with integral max-pooling of CNN activations. *arXiv* **2015**, arXiv:1511.05879.
- [12] Babenko, A.; Lempitsky, V. Aggregating local deep features for image retrieval. In *Proceedings of the IEEE International Conference on Computer Vision*, Santiago, Chile, 7–13 December 2015; pp. 1269–1277.
- [13] Lin, J.; Duan, L.-Y.; Wang, S.; Bai, Y.; Lou, Y.; Chandrasekhar, V.; Huang, T.; Kot, A.; Gao, W. Hnlp: Compact deep invariant representations for video matching, localization, and retrieval. *IEEE Trans. Multimed.* **2017**, *19*, 1968–1983.

- [14] Kalantidis, Y.; Mellina, C.; Osindero, S. Cross-dimensional weighting for aggregated deep convolutional features. In *European Conference on Computer Vision*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 685–701.
- [15] Azhar, I.; Sharif, M.; Raza, M.; Khan, M.A.; Yong, H.-S. A decision support system for face sketch synthesis using deep learning and artificial intelligence. *Sensors* **2021**, *21*, 8178.
- [16] Khan, S.; Khan, M.A.; Alhaisoni, M.; Tariq, U.; Yong, H.-S.; Armghan, A.; Alenezi, F. Human action recognition: A paradigm of best deep learning features selection and serial based extended fusion. *Sensors* **2021**, *21*, 7941.
- [17] Noh, H.; Araujo, A.; Sim, J.; Weyand, T.; Han, B. Largescale image retrieval with attentive deep local features. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 3456–3465.
- [18] Arandjelovic, R.; Gronat, P.; Torii, A.; Pajdla, T.; Sivic, J. NetVLAD: CNN architecture for weakly supervised place recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26–30 June 2016; pp. 5297–5307.
- [19] Radenović, F.; Tolias, G.; Chum, O. CNN image retrieval learns from bow: Unsupervised fine-tuning with hard examples. In *European Conference on Computer Vision*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 3–20.
- [20] Gordo, A.; Almazan, J.; Revaud, J.; Larlus, D. End-to-end learning of deep visual representations for image retrieval. *Int. J. Comput. Vis.* **2017**, *124*, 237–254.

- [21] Hyvärinen, A.; Hurri, J.; Hoyer, P.O. Principal components and whitening. In *Natural Image Statistics*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 93–130.
- [22] Izenman, A.J. Linear discriminant analysis. In *Modern Multivariate Statistical Techniques*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 237–280.
- [23] Johnson, R.A.; Wichern, D.W. Canonical correlation analysis. In *Applied Multivariate Statistical Analysis*, 6th ed.; Pearson: Upper Saddle River, New Jersey 2018; pp. 539–574.
- [24] Gong, Y.; Ke, Q.; Isard, M.; Lazebnik, S. A multi-view embedding space for modeling internet images, tags, and their semantics. *Int. J. Comput. Vis.* **2014**, *106*, 210–233.
- [25] Yan, F.; Mikolajczyk, K. Deep correlation for matching images and text. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3441–3450.
- [26] Dorfer, M.; Schlüter, J.; Vall, A.; Korzeniowski, F.; Widmer, G.; “End-to-end cross-modality retrieval with cca projections and pairwise ranking loss. *Int. J. Multimed. Inf. Retr.* **2018**, *7*, 117–128.
- [27] Yu, Y.; Tang, S.; Aizawa, K.; Aizawa, A. Category-based deep cca for fine-grained venue discovery from multimodal data. *arXiv* **2018**, arXiv:1805.02997.
- [28] Lin, Z.; Peltonen, J. An information retrieval approach for finding dependent subspaces of multiple views. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 1–16.

- [29] Yair, O.; Talmon, R. Local canonical correlation analysis for nonlinear common variables discovery. *IEEE Trans. Signal Process.* **2017**, *65*, 1101–1115.
- [30] Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv*, **2014**, arXiv:1409.1556.
- [31] Abdi, H. The eigen-decomposition: eigenvalues and eigenvectors. In *Encyclopedia of Measurement and Statistics*; SAGE Publications, Inc: Thousand Oaks, CA, USA 2007; pp. 304–308.
- [32] Nielsen, F. An information-geometric characterization of chernoff information. *IEEE Signal Process. Lett.* **2013**, *20*, 269–272.
- [33] Nielsen, F. Chernoff information of exponential families. *arXiv* **2011**, arXiv:1102.2684.
- [34] Prince, S.J. Common probability distribution. In *Computer Vision: Models, Learning and Inference*; Cambridge University Press: Cambridge, England 2012; pp. 35–42.
- [35] Radenović, F.; Toliás, G.; Chum, O. Fine-tuning CNN image retrieval with no human annotation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*; Institute of Electrical and Electronics Engineers: Manhattan, New York, USA, 2018.
- [36] Schonberger, J.L.; Radenovic, F.; Chum, O.; Frahm, J.-M. From single image query to detailed 3d reconstruction. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 5126–5134.
- [37] Koren, Y.; Carmel, L. Robust linear dimensionality reduction. *IEEE Trans. Vis. Comput. Graph.* **2004**, *10*, 459–470.

- [38] Li, T.; Zhu, S.; Ogihara, M. Using discriminant analysis for multi-class classification. In *Third IEEE International Conference on Data Mining*; IEEE Computer Society: Los Alamitos, CA, USA, 2003; pp. 589–589.
- [39] Mirkes, E.M.; Gorban, A.N.; Zinoviev, A. A Supervised PCA. 2016. Available online: 2016 <https://github.com/Mirkes/SupervisedPCA>.
- [40] Fisher, R.A. The use of multiple measurements in taxonomic problems. *Ann. Eugen.* **1936**, 7, 179–188.
- [41] Swets, D.L.; Weng, J.J. Using discriminant eigenfeatures for image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.* **1996**, 8, 831–836.
- [42] Philbin, J.; Chum, O.; Isard, M.; Sivic, J.; Zisserman, A. Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Minneapolis, MN, USA, 17–22 June 2007; pp. 1–8.
- [43] Radenovic, F.; Iscen, A.; Tolias, G.; Avrithis, Y.; Chum, O. Revisiting oxford and paris: Large-scale image retrieval benchmarking. In *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, Salt Lake City, UT, USA, 18–23 June 2018.
- [44] Philbin, J.; Chum, O.; Isard, M.; Sivic, J.; Zisserman, A. Lost in quantization: Improving particular object retrieval in large scale image databases. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, AK, USA, 23–28 June 2008.

The following chapter is a reproduction of a submitted paper to IEEE Access:

Shi Kangdi, Muhammad Alrabeiah and Jun Chen. "Progressive with Purpose: Guiding Progressive Inpainting DNNs through Context and Structure."

Contribution Declaration: Kangdi Shi (the author of this thesis) is the first author and main contributor of this article. He conducted all experiments and composed the paper. Muhammad Alrabeiah helped in polishing the paper. Jun Chen is the supervisor of Kangdi Shi.

## **Chapter 3**

# **Progressive with Purpose: Guiding Progressive Inpainting DNNs through Context and Structure**

### **3.1 Abstract**

The advent of deep learning in the past decade has significantly helped advance image inpainting. Although achieving promising performance, deep learning-based inpainting algorithms still struggle from the distortion caused by the fusion of structural and contextual features, which are commonly obtained from, respectively, deep and shallow layers of a convolutional encoder. Motivated by this observation, we propose a novel progressive inpainting network that maintains the structural and contextual integrity of a processed image. More specifically, inspired by the Gaussian and Laplacian pyramids, the core of the proposed network is a feature extraction module named GLE. Stacking GLE



modules enables the network to extract image features from different image frequency components. This ability is important to maintain structural and contextual integrity, for high frequency components correspond to structural information while low frequency components correspond to contextual information. The proposed network utilizes the GLE features to progressively fill in missing regions in a corrupted image in an iterative manner. Our benchmarking experiments demonstrate that the proposed method achieves clear improvement in performance over many state-of-the-art inpainting algorithms.

## **3.2 Introduction**

Image inpainting is the task of restoring missing patches of pixels in an image [1, 2]. As the name suggests, inpainting targets filling in missing parts of an image (i.e., image holes) with contextually meaningful information so that the image can be restored to its original form. This could be a quite difficult task for machines, for it is an ill-posed inverse problem [8]. It requires not only the ability to predict what is missing, but also whether it fits within the context of the image or not. Thus, a key to attaining satisfying inpainting results is to ensure that the reconstructed pixels are consistent with the uncorrupted region and exhibit coherence in both structure and texture.

As a main remedy for restoring image quality, inpainting is of great importance nowadays, for modern societies are increasingly reliant on visual content with images as its building block, from surveillance systems to autonomous vehicles, media streaming, and conference calls. Storing, displaying, and exchanging huge amount of images make them prone to damages, one of which is missing pixels (image holes). It is thus unsurprising to see increasing research interest in inpainting within the computer vision community.

### 3.2.1 Motivation

Many image inpainting techniques have been proposed over the past two decades. They could loosely be grouped into two major categories: traditional and modern. The main defining difference between the two categories is the use of deep learning. The traditional category of techniques could collectively be divided into two sub-categories [8]: exemplar-based and diffusion-based. The former approach [9] searches for the best matching patches from known regions and pastes them into missing regions. Such techniques have high computational cost for patch searching and generate unrealistic results due to the lack of perspective transformation. The diffusion-based techniques, on the other hand, recreate a missing region with features from its surrounding known region. Although diffusion-based techniques are more efficient than their exemplar-based counterparts, they result in over-smoothed inpainting results because of regularization based on partial-differential equations.

The advent of deep learning in computer vision has created a surge of inpainting techniques that utilize Deep Neural Networks (DNNs). Although those techniques exhibit some overlap with the traditional diffusion-based techniques, they define the state-of-the-art in inpainting, and, therefore, merit a category of their own. Most of the early works on inpainting with deep learning, like [10] [11], follow a two-stage approach, which firstly learns the image structure from a given edge/structure map of corrupted image, then refines the missing region with a texture generator. However, two-stage image inpainting methods usually cause artifacts due to their limited ability to recover both structure and texture.

To deal with those artifacts, progressive inpainting techniques have been explored. They rely on the idea that not all predicted pixels in a region plagued with artifacts are

defective; some are good predictions that could be utilized to improve the re-generated region and weed out the artifacts. Hence, those techniques fill in the missing holes by iterating over the image and learning from previously-predicted pixels. Examples of such technique are the full-resolution residual network proposed by Guo *et al.*, [12] and the iterative confidence feedback network proposed by Zheng *et al.*, [13]. Despite the improvement they provide over two-stage techniques, the performance of progressive techniques is still prone to artifacts. This could be traced back to the inexplicit modeling of structure and texture in those techniques.

Fusing structure and texture awareness with progressive inpainting is arguably the most promising approach to overcoming visual artifacts, which we shall follow in this paper. Developing structure and texture-aware algorithms has been explored recently in Guo *et al.*, [14]. Two different but coupled autoencoders are trained with structure and texture constraints to fill in holes in corrupted images. The results are encouraging, but the algorithm can cause distortion in deep parts of the hole due to one-stage feature fusion. This could be overcome by incorporating progressive inpainting into the learning process.

### 3.2.2 Contribution

In an attempt to bring together progressive learning and fusion of texture and structure, this paper presents a Gaussian-Laplacian feature Extraction (GLE) module. The main contributions of the proposed architecture are summarized below:

- **GLE Module:** Inspired by image pyramid, we propose a GLE module to obtain features from high and low image frequency components. Those components provide texture information (low-frequency components) and structure information (high-frequency components). The GLE module leverages those multi-frequency

components to learn textural and structural features.

- **Iterative Repainting Component:** A progressive repainting component is developed such that it gradually fills in the corrupted regions of an image. It utilizes features learned by the GLE modules from different frequency components to fill in the outer edge of the corrupted regions iteratively until the region is restored.
- **Benchmarking and Evaluation Experiments:** Various experiments are designed to evaluate the performance of the proposed architecture and show the benefits of the GLE module and the repainting component. The experiments also compare the proposed inpainting algorithm to some state-of-the-art algorithms to situate its contribution to the inpainting problem.

### 3.2.3 Paper organization

The organization of this paper is shown as follows. Section II reviews works that are related to our method. Section III details the architecture of the proposed progressive image inpainting network. Sections IV and V illustrate the experimental setup and the experimental results. Section VI concludes this paper.

## 3.3 Related Work

The proposed solution is developed on top of a rich literature of image inpainting with deep learning. To facilitate the discussion, the following two subsections will review some concepts related to the proposed solution. They should lay the necessary groundwork for the detailed description in Section 3.4

### 3.3.1 Variants of Texture and Structure Inpainting

Inpainting based on texture and structure has been attempted in various forms in the literature. The concepts of *style* and *content* have been introduced in [3], which could be viewed as derivatives of texture and structures, respectively. They are used to build a two-stage inpainting DNN. In the first, two encoders extract style and content latent information separately, and the second stage synthesizes a full image from that information. Semantic segmentation masks are another alternative that helps capture structure information. They have been utilized in [4–6] as a way to guide texture generation. In all those papers, an encoder network learns to generate a latent representation of the corrupted image that captures the structure. It does so by pushing the decoder network to recover not only the inpainted image but also its segmentation mask. The three papers differ in the details of how to encode a corrupted image and generate a structurally consistent image.

### 3.3.2 Progressive Image Inpainting

Progressive image inpainting, as the name suggests, aims to recover images gradually by utilizing features from undamaged and recently recovered regions. Overall, algorithms following this approach could be grouped in two broad categories: (i) contextual information-based algorithms, and (ii) structural constraints-based algorithm. Both are briefly reviewed below.

Contextual information-based algorithms rely mainly on CNN features extracted from input images to restore damaged regions. As a pioneer of the contextual features-based algorithms, Hsu *et al.*, [15] propose using several deep convolutional networks to learn progressive inpainting from multiple image scales, from low to high resolution images.

Zhang *et al.*, [16] recognize how inpainting lends itself to recurrent modelling; they propose to use several generative networks inter-connected with LSTM module, which progressively fills in the missing region of an image. More recently, Li *et al.*, [17] extend that analogy further. They propose a recurrent feature reasoning module with knowledge-consistent attention, which can progressively enhance the details in masked regions.

Compared with their contextual information-based counterparts, the structural constraints-based algorithms take advantage of additional external structural constraints provided by edge detection algorithms. The algorithms in [18] [19] utilize contour or edge maps as a guide for image completion. To progressively complete the image, Li *et al.*, [20] propose a U-net that recovers the edge maps while inpainting images progressively. These approaches, collectively, seek to tackle image inpainting by introducing structural constraints, yet their performance remains limited by a lack of information for recovering deeper pixels in the missing regions.

### 3.3.3 Gaussian and Laplacian Pyramid

A classical approach to image inpainting is centered around the idea of building multi-scale image pyramids, in which inpainting is done progressively from one scale to another—commonly from smallest to largest scale. Those pyramids are usually called Gaussian or Laplacian pyramids based on the type of filters used to generate them. Specifically, let  $\mathcal{G}$  denote the Gaussian smooth operator,  $\mathbf{I}_\tau$  express the input image to the  $\tau_{th}$  level of Gaussian pyramid.  $\mathcal{Q}$  denotes upsampling operation,  $\mathcal{D}$  denotes downsampling operation. The formulas for the output images  $\mathbf{G}_\tau$ ,  $\mathbf{F}_\tau$  of  $\tau_{th}$  Gaussian pyramid and

Laplacian pyramid are

$$\mathbf{G}_\tau = \mathcal{D}(\mathcal{G}(\mathbf{I}_\tau)) \quad (3.3.3.1)$$

$$\mathbf{F}_\tau = \mathbf{I}_\tau - \mathcal{Q}(G_\tau) \quad (3.3.3.2)$$

Inpainting algorithms using Gaussian and Laplacian pyramids could roughly be clustered into three groups: inpainting on multiple Gaussian pyramids [21], inpainting on multiple Laplacian pyramids [22] [23], and inpainting on multiple Gaussian and Laplacian [23].

The difference between the first and the second group is in how the inpainting algorithm is applied on different image pyramids. For instance, Farid *et al.*, [21] first generate multiple Gaussian pyramids until most missing pixels are eliminated by the smoothing operation. Then, their algorithm copies and pastes the missing pixels from the small-scale image (top of the pyramid) to the large-scale images (bottom of the pyramid). In contrast, [22] utilizes Laplacian pyramid with patch search to recover missing pixels from small to large scale images in the pyramid. Because of the limitation of exemplar-based methods, both kinds of methods suffer from unrealistic inpainting results.

Benefiting from the combination of structure and texture, the third inpainting group (i.e., algorithms relying on multiple Gaussian and Laplacian pyramids) usually achieve better performance than their counterparts relying only on one of the two pyramids. However, the additional cost from inpainting both Pyramids is relatively high compared to that of the former two groups.

### 3.4 Proposed Inpainting Algorithm

Like a person solving a jigsaw puzzle, an inpainting algorithm should fill in the missing regions by gradually piecing pixels together while keeping an eye on context and structure. Progressive algorithms, as mentioned earlier, restore missing pixels gradually using undamaged and recently recovered pixels, yet they do not jointly maintain contextual and structural information. This observation fuels the work in this paper; a Deep neural Network (DNN) is designed such that it progressively inpaints with the purpose of maintaining structure and context information. Hence, it is described as being progressive with purpose.

The idea behind the proposed algorithm is to break down the inpainting task into three main stages, namely feature extraction (first stage), iterative inpainting (second stage), and enhancing and reconstruction (third stage). The first stage is aimed to extract multi-level features from the corrupted image, which mimics, to some extent, feature extraction from image pyramids used in classical inpainting algorithms such as [21, 22]. The multi-level features capture contextual and structural information. They are fed to the iterative inpainting stage, which attempts to recover some of the missing information gradually over several iterations. Each one generates a pair of feature volumes. The pairs are passed to the enhancement and reconstruction stage to enhance the recovered information, fuse them into one feature volume, and reconstruct the complete image. The architecture of the proposed algorithm is depicted in Figure 3.7.

The architecture is detailed in the following four subsections. The first one presents a formal description of how progressive inpainting restores missing pixels. The following three are a deep-dive into the three stages of the proposed architecture, describing the inner workings of each stage. Finally, the last subsection presents the loss function used to train



the architecture.

### 3.4.1 Rationale Behind the Proposed Algorithm

Let  $\mathbf{a}$  be the original image, and  $\mathbf{b}$  be the corrupted image. We denote the conditional probability distribution of the original image given the corrupted image by  $p_{\mathbf{A}|\mathbf{B}}$ . Image inpainting can be formulated as a maximum a posterior (MAP) estimation problem:

$$\hat{\mathbf{a}}_{map} = \arg \max_{\hat{\mathbf{a}}} p_{\mathbf{A}|\mathbf{B}}(\hat{\mathbf{a}}|\mathbf{b}).$$

Let  $\mathbf{m}$  be the ground truth of the corrupted region,  $\mathbf{n}$  and  $\mathbf{c}$  be the valid region and corrupted region of the corrupted image, then  $\mathbf{a} = \mathbf{m} \cup \mathbf{n}$ ,  $\mathbf{b} = \mathbf{c} \cup \mathbf{n}$ . Note that the conditional distribution of  $\mathbf{m}$  given  $\mathbf{n}$ , denoted by  $p_{\mathbf{M}|\mathbf{N}}$ , is a projected version of  $p_{\mathbf{A}|\mathbf{B}}$  and can be learned from the training dataset. The MAP estimation of  $\mathbf{a}$  based on  $\mathbf{b}$  can be reduced to the MAP estimation of  $\mathbf{m}$  based on  $\mathbf{n}$ :

$$\hat{\mathbf{m}}_{map} = \arg \max_{\hat{\mathbf{m}}} p_{\mathbf{M}|\mathbf{N}}(\hat{\mathbf{m}}|\mathbf{n}).$$

Clearly, we have

$$\hat{\mathbf{a}}_{map} = \hat{\mathbf{m}}_{map} \cup \mathbf{n}.$$

Our algorithm aims to produce an approximate version of  $\hat{\mathbf{m}}_{map}$ .

We divide the corrupted region into  $T$  concentric regions, and progressively recover the corrupted region in an inward manner from the  $1_{th}$  to the  $T_{th}$  concentric region.

Let  $\tilde{\mathbf{m}}_r^{(\tau)}$  denote the inpainted  $r_{th}$  concentric region at the  $\tau_{th}$  step. The process proceeds as follows. At the  $\tau_{th}$  step (with  $\tau$  from 1 to  $T$ ), we generate  $\tilde{\mathbf{m}}_\tau^{(\tau)}$  based on the valid region

$\mathbf{n}$  and the  $\tau - 1$  inpainted concentric regions  $\tilde{\mathbf{m}}_r^{(\tau-1)} (1 \leq r \leq \tau - 1)$  from the  $\tau - 1_{th}$  step\*, then refine  $\tilde{\mathbf{m}}_r^{(\tau-1)} (1 \leq r \leq \tau - 1)$  to  $\tilde{\mathbf{m}}_r^{(\tau)} (1 \leq r \leq \tau - 1)$ , respectively.

At the end of the  $T_{th}$  step, we collected  $\tilde{\mathbf{m}}_r^{(\tau)} (1 \leq \tau \leq T, 1 \leq r \leq \tau)$  generated throughout the process and perform an enhancement. Specifically, for  $\tau$  from 1 to  $T - 1$ , we leverage  $\cup_{r=1}^{\tau-1} \tilde{\mathbf{m}}_r^{(\tau-1)}$  (which is void when  $\tau = 1$ ) and  $\cup_{r=1}^{\tau} \tilde{\mathbf{m}}_r^{(\tau+1)}$ , together with the valid region  $\mathbf{n}$ , to enhance  $\cup_{r=1}^{\tau} \tilde{\mathbf{m}}_r^{(\tau)}$ . More precisely, for each  $\tau$ , the enhancement is carried out in two parts separately:  $\cup_{r=1}^{\tau-1} \tilde{\mathbf{m}}_r^{(\tau)}$  is enhanced based on  $\cup_{r=1}^{\tau-1} \tilde{\mathbf{m}}_r^{(\tau-1)}$  and  $\cup_{r=1}^{\tau-1} \tilde{\mathbf{m}}_r^{(\tau+1)}$  while  $\tilde{\mathbf{m}}_{\tau}^{(\tau)}$  is enhanced based on  $\tilde{\mathbf{m}}_{\tau}^{(\tau+1)}$ . It is also worth mentioning that in our implementation, we decompose  $\mathbf{n}$  into low-level information  $\mathbf{l}$  and high-level information  $\mathbf{h}$  using a feature extraction network. Let  $\hat{\mathbf{m}}_{\tau}$  denote the enhanced version<sup>†</sup> of  $\cup_{r=1}^{\tau} \tilde{\mathbf{m}}_r^{(\tau)} (1 \leq \tau \leq T)$ . Our algorithm produces  $\cup_{\tau=1}^T \hat{\mathbf{m}}_{\tau}$  as an approximation of  $\hat{\mathbf{m}}_{map}$ .

---

\*when  $\tau = 1$ , there is no inpainted region available, but only the valid region  $\mathbf{n}$ .

<sup>†</sup>Note that  $\hat{\mathbf{m}}_T = \cup_{r=1}^T \tilde{\mathbf{m}}_r^{(T)}$  since no enhancement is performed when  $\tau = T$ .

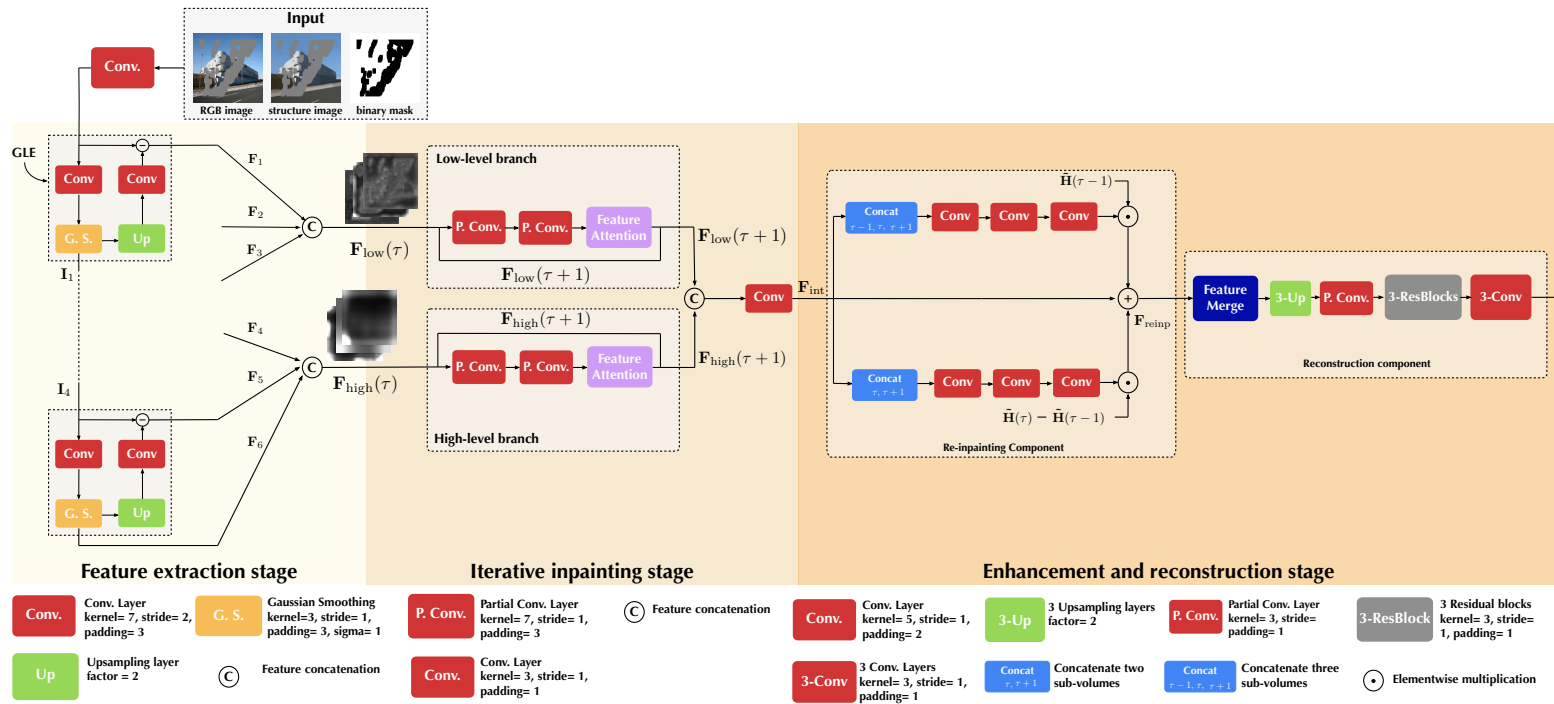


Figure 3.7: A graphical description of the proposed solution. It shows all three stages and detail their main components and elements.

### 3.4.2 Feature Extraction Stage

The feature extraction stage is inspired by Gaussian and Laplacian pyramids. The main component of this stage is a sequence of convolution, Gaussian smoothing, upsampling, another convolution, and subtraction. This sequence will be henceforth referred to as the GLE module. As shown in the first column of Figure 3.7, the input is first passed through a convolutional layer with 64 kernels, and a ReLU activation function. Next, the generated feature continue passing through a convolutional layer with a number of kernels that is double the number of input channels. Each kernel has a  $7 \times 7$  height and width,  $2 \times 2$  stride, and  $3 \times 3$  padding. It results in a reduced size feature that is then blurred using a  $3 \times 3$  Gaussian kernel moving with a stride of 1 and implementing a padding of 1 to maintain the spatial dimensions fixed. The smoothed feature is passed to the next feature module and to the upsampling layer, as well. It is upsampled using nearest neighbor to recover the original input size before it is passed through the second convolutional layer. This convolution is characterized with the same hyper-parameters as those of the first one, but it has half the number of kernels recovering the same number of channels as that of the input tensor. The output feature map is produced by subtracting the original input from the feature map coming from the second convolutional layer. Let  $I_{\tau-1}$  denotes the input feature maps of  $\tau_{th}$  GLE module,  $\mathcal{G}$  denotes the gaussian smoothing operation,  $Up$  denotes the upsample operation,  $\Lambda_{Gs}$  denotes the weights of the convolutional layer before the Gaussian smoothing operation, and  $\Lambda_{Up}$  denotes the weights of convolution layer after the upsample operation. The GLE module can be expressed as

$$\mathbf{F}_{\tau} = \mathbf{I}_{\tau-1} - \Lambda_{\mathcal{Q}}(\mathcal{Q}(\mathcal{G}(\Lambda_{\mathcal{G}}(\mathbf{I}_{\tau-1})))), \quad (3.4.2.1)$$

where  $\tau \in \{1, 2, \dots, 5\}$ .  $\mathbf{F}_\tau$  denotes output feature maps from  $\tau_{th}$  GLE module. The  $\mathbf{F}_6$  is generated right after the gaussian smooth layer of  $5_{th}$  GLE module.

To produce various levels of features, this stage is designed to have 5 GLE modules stacked consecutively, each one feeds into the next. A corrupted image, one with missing pixels and denoted by  $\mathbf{I}_{in}$  in Figure 3.7, a corrupted structural image  $\mathbf{I}_{struc}$  used in [10] and a binary mask  $\mathbf{M}_{in}$  are the input to the first module, and the output is the blurred feature volume  $\mathbf{I}_1$  as well as the difference feature volume  $\mathbf{F}_1$ .  $\mathbf{I}_1$  has half the height and width of the input image and double the number of channels, and it is passed to the next GLE module.  $\mathbf{F}_1$ , on the other hand, is buffered to construct the feature pyramid output that represents the output of the feature extraction stage. The pyramid is formed by stacking the difference-feature volumes generated by each GLE module, namely  $\mathbf{F}_1, \dots, \mathbf{F}_6$ .

### 3.4.3 Iterative Inpainting Stage

This is the second stage of the proposed solution, which is based on the concept of progressive inpainting. The main elements of this stage are partial convolution, regular convolution, and feature attention. These elements make up two parallel branches, in which features are processed iteratively. The following three subsections detail the inner workings of this stage.

#### Partial Convolution

Partial convolution is a fundamental tool to fill the irregular holes in deep learning-based image inpainting and keep track of the unfilled regions of the image. To see how a partial convolution layer accomplishes this, let  $\mathbf{W}_k \in \mathbb{R}^{C \times H' \times W'}$  denote the weight tensor of the  $k$ -th kernel in a partial convolution layer,  $\mathbf{X}_{i,j} \in \mathbb{R}^{C \times H' \times W'}$  denote the input feature patch

extracted from the input tensor  $\mathbf{X}_{\text{in}} \in \mathbf{R}^{C \times H \times W}$  centered around  $(i, j)$ -th pixel, where  $C$ ,  $H'$  and  $W'$  are, respectively, the number of channels, height, and width of the patch and  $C$ ,  $H$  and  $W$  are, respectively, the number of channels, height, and width of the input tensor. Also, let  $\tilde{\mathbf{H}}_{i,j}$  denote a  $H' \times W'$  binary patch centered around the  $(i, j)$ -th pixel, and  $\mathbf{H}_{i,j}$  is a  $C \times H' \times W'$  binary tensor formed by stacking  $C$  copies of the matrix  $\tilde{\mathbf{H}}_{i,j}$ . Then, the  $(i, j)$ -th value of the  $k$ -th output feature map, i.e.,  $y_{i,j,k}$ , produced by a partial convolution layer—before activation—is given by

$$y_{i,j,k} = \begin{cases} g(\mathbf{W}_k, \mathbf{X}_{i,j}, \mathbf{H}_{i,j}) + b, & \sum_{C,H',W'} \mathbf{H}_{i,j} > 0, \\ 0, & \text{otherwise,} \end{cases} \quad (3.4.3.1)$$

where  $g(\mathbf{W}_k, \mathbf{X}_{i,j}, \mathbf{H}_{i,j})$  is defined as

$$\sum_{C,H',W'} \mathbf{W}_k \odot (\mathbf{X}_{i,j} \odot \mathbf{H}_{i,j}) \frac{\sum_{C,H',W'} \mathbf{1}}{\sum_{C,H',W'} \mathbf{H}_{i,j}}, \quad (3.4.3.2)$$

$\mathbf{1}$  is a  $C \times H' \times W'$  tensor of all ones, and  $b \in \mathbb{R}$  is the bias associated with the  $k$ -th kernel. Following a partial convolution is a mask update to make sure that the mask is keeping up with the updated feature map coming out of the partial convolution. Let the full mask be given by

$$\tilde{\mathbf{H}} = \begin{bmatrix} h_{11} & \dots & h_{1W} \\ \vdots & \ddots & \vdots \\ h_{H1} & \dots & h_{HW} \end{bmatrix}, \quad (3.4.3.3)$$

where  $H$  and  $W$  are, respectively, the height and width of the mask such that  $H \gg H'$  and  $W \gg W'$ , and  $\tilde{\mathbf{H}}_{i,j}$  is a sub-matrix forming a block in  $\tilde{\mathbf{H}}$  centered around  $(i, j)$ -th pixel. This mask is updated by convolving an all one kernel with the mask. Let  $\mathbf{U}$  be a  $H' \times W'$

kernel of all ones. Then, the updated mask is given by

$$\tilde{\mathbf{H}}_{\text{new}} = \tilde{\mathbf{H}}_{\text{previous}} * \mathbf{U}, \quad (3.4.3.4)$$

where  $*$  is the convolution operation with a stride equal to that of the partial convolution kernel. More about partial convolution could be found in [24].

### Feature Attention

For any feature volume  $\mathbf{F} \in \mathbb{R}^{C \times H \times W}$ , an attention tensor could be generated using cosine similarity and softmax. Let  $\mathbf{f}_{i,j}$  and  $\mathbf{f}_{i',j'}$  denote pair of feature values at location  $i, j$  and  $i', j'$ . Then, their cosine similarity is computed as follows:

$$z_{i,j,i',j'} = \left\langle \frac{\mathbf{f}_{i,j}}{\|\mathbf{f}_{i,j}\|}, \frac{\mathbf{f}_{i',j'}}{\|\mathbf{f}_{i',j'}\|} \right\rangle, \quad (3.4.3.5)$$

where  $z_{i,j,i',j'}$  denotes the cosine similarity score between the  $\mathbf{f}_{i,j}$  and  $\mathbf{f}_{i',j'}$ <sup>‡</sup>. Let  $\mathbf{Z}_{i,j} \in \mathbb{R}^{H \times W}$  denotes the score matrix of a feature vector at location  $i, j$  and all  $C$ -dimensional feature vectors in  $\mathbf{F}$ . The softmax function is applied across the height and width to generate the attention score of location  $i, j$  in  $\mathbf{F}$ . Formally, this is expressed as follows:

$$\hat{\mathbf{Z}}_{i,j} = \text{sfm}(\mathbf{Z}_{i,j}), \quad (3.4.3.6)$$

where  $\hat{\mathbf{Z}}_{i,j} \in \mathbb{R}^{H \times W}$ . The final feature volume  $\mathbf{F}$  has an attention tensor  $\hat{\mathbf{Z}} \in \mathbb{R}^{HW \times H \times W}$  formed by stacking  $HW$  score maps  $\hat{\mathbf{Z}}_{i,j}$ . Based on the calculated score map, we reuse the feature patches from the input of feature attention module as de-convolutional filters to

---

<sup>‡</sup> $\mathbf{f}_{i,j}$  is a feature vector with size  $1 \times 1 \times C$  at position  $i, j$  from a feature volume with size  $H \times W \times C$ .  $\mathbf{f}_{i,j}$  and  $\mathbf{f}_{i',j'}$  have the same size, and  $\|\mathbf{f}_{i,j}\|$  and  $\|\mathbf{f}_{i',j'}\|$  are the second norms of those vectors.

reconstruct the new feature map.

### Putting It All Together

Iterative inpainting is built on top of the feature extraction stage with the feature pyramid as its input. This is illustrated in the middle column of Figure 3.7. The pyramid is first split into two halves; feature maps coming from the first three feature modules (i.e., the first three from the input side) are concatenated to form the feature volume  $\mathbf{F}_{\text{low}} \in \mathbb{R}^{C_{in} \times H_{in} \times W_{in}}$  with  $C_{in}$  channels,  $H_{in}$  height, and  $W_{in}$  width. Feature maps coming from the last three feature modules form another feature volume denoted  $\mathbf{F}_{\text{high}} \in \mathbb{R}^{C_{in} \times H_{in} \times W_{in}}$ . Those two volumes are sent down two different but parallel iterative branches that have the same composition of layers. Both start with two partial convolutions with leaky ReLU activations, followed by a feature attention module. The specifications of each layer are detailed at the bottom of the middle column of Figure 3.7.

Each branch processes the input volume iteratively, which is done as follows. Let  $\tau$  represent a time index for the iterative process. Both  $\mathbf{F}_{\text{low}}(0)$  or  $\mathbf{F}_{\text{high}}(0)$  goes through the partial convolutions and the attention module, making up the first iteration ( $\tau = 1$ ). The outputs, denoted  $\mathbf{F}_{\text{low}}(\tau + 1)$  and  $\mathbf{F}_{\text{high}}(\tau + 1)$ , are used to initialize the next iteration as well as construct a new feature volume. A copy of  $\mathbf{F}_{\text{low}}(\tau + 1)$  and  $\mathbf{F}_{\text{high}}(\tau + 1)$  is sent back to the input to undergo the next iteration. Another copy is sent forward to a concatenation operation to form part of a new feature volume denoted  $\mathbf{F}_{\text{cat}}$ . This keeps on going for  $T$  iterations ( $\tau \in \{1, 2, \dots, T\}$ ) until  $\mathbf{F}_{\text{cat}}$  is complete, i.e., a tensor of dimensions  $C_{\text{cat}} \times H_{\text{cat}} \times W_{\text{cat}}$  where  $C_{\text{cat}} = 2TC_{in}$  is formed. This tensor is, finally, passed to a convolution layer with leaky ReLU activation, which generates the intermediate feature volume  $\mathbf{F}_{\text{int}}$ .



**Remark:** Please note that the  $\mathbf{F}_{\text{int}}$  feature volume comprises  $C_{\text{int}} = C_{\text{cat}} = 2TC_{in}$  feature maps, which could be split into  $T$  sub-volumes. This is important for the sake of the third and final stage of the proposed architecture.

### 3.4.4 Enhancement and Reconstruction Stage

#### Reinpainting Component

The main idea behind the repainting component is to re-enhance the fused feature sub-volumes in  $\mathbf{F}_{\text{int}}$ . This is done along two branches that process two different concatenations of feature sub-volumes from  $\mathbf{F}_{\text{int}}$ . See Figure 3.7. Let  $\mathbf{F}_{\text{int}}(\tau)$  represent the  $\tau$ -th sub-volume in  $\mathbf{F}_{\text{int}}$ , where  $\tau \in \{1, \dots, T - 1\}$ . The first branch concatenates  $\mathbf{F}_{\text{int}}(\tau - 1)$ ,  $\mathbf{F}_{\text{int}}(\tau)$ , and  $\mathbf{F}_{\text{int}}(\tau + 1)$  and passes them into three convolutional layers with ReLU activations. The result is multiplied with the updated mask of iteration  $\tau - 1$  from the second stage, i.e.,  $\tilde{\mathbf{H}}(\tau - 1)$ , to eliminate the negative effect of the unfilled region in each iteration.

The second branch is symmetric with the first, but focuses on different sub-volumes. It concatenates  $\mathbf{F}_{\text{int}}(\tau)$  and  $\mathbf{F}_{\text{int}}(\tau + 1)$  and passes them through three convolutional layers with ReLU activations. The result here is multiplied with the difference of two updated masks from iterations  $\tau$  and  $\tau - 1$ , i.e.,  $\tilde{\mathbf{H}}(\tau) - \tilde{\mathbf{H}}(\tau - 1)$ , which only contains information from the intersection region between  $\mathbf{F}_{\text{int}}(\tau)$  and  $\mathbf{F}_{\text{int}}(\tau - 1)$ . The results of the two branches are combined with the sub-volume  $\mathbf{F}_{\text{int}}(\tau)$  to produce a new sub-volume  $\mathbf{F}_{\text{reinp}}(\tau)$ .

#### Reconstruction Component

The repainting model outputs a feature volume  $\mathbf{F}_{\text{reinp}}$  that is fed to the reconstruction component. This is the final component of the proposed architecture, and it is responsible

for producing the complete image. A visualization of the reconstruction component is illustrated in the left panel of Fig. 3.7. This component adopts the feature merging module from [17] which fuses the feature group based on the filled locations in each iteration. The merge module feeds into three upsampling layers followed by a partial convolution layer, three residual blocks, and a sequence of three convolutional layers. The complete architecture is summarized in Algorithm 1 <sup>§</sup>.

---

**Algorithm 1** The Proposed Inpainting Network
 

---

**Input** Input image  $I_{in}$ ,  
 Input structural  $I_{struc}$ , Input mask  $M_{in}$ ,  
 Total number of iteration  $T$

**Output** Recovered image  $I_{out}$

- 1:  $\mathbf{F}_{high}(0), \mathbf{F}_{low}(0), \tilde{\mathbf{H}}(0) \leftarrow GLE(I_{in}, I_{struc}, M_{in})$
- 2:  $FeaturePool \leftarrow \{\mathbf{F}_{high}(0), \mathbf{F}_{low}(0)\}$
- 3:  $\tau \leftarrow 0$
- 4: **if**  $\tau \leq T$  **then**
- 5:    $\mathbf{F}_{low}(\tau + 1), \tilde{\mathbf{H}}(\tau + 1) \leftarrow \mathcal{I}(\mathbf{F}_{low}(\tau), \tilde{\mathbf{H}}(\tau))$
- 6:    $\mathbf{F}_{high}(\tau + 1), \tilde{\mathbf{H}}(\tau + 1) \leftarrow \mathcal{I}(\mathbf{F}_{high}(\tau), \tilde{\mathbf{H}}(\tau))$
- 7:    $\mathbf{F}_{int}(\tau + 1) \leftarrow FeatureFuse(\mathbf{F}_{high}(\tau + 1), \mathbf{F}_{low}(\tau + 1))$
- 8:    $\tau \leftarrow \tau + 1$
- 9: **end if**
- 10:  $\tau \leftarrow 1$
- 11: **if**  $\tau \leq T - 1$  **then**
- 12:    $F = \{\mathbf{F}_{int}(\tau - 1), \mathbf{F}_{int}(\tau), \mathbf{F}_{int}(\tau + 1)\}$
- 13:    $H = \{\tilde{\mathbf{H}}(\tau - 1), \tilde{\mathbf{H}}(\tau), \tilde{\mathbf{H}}(\tau + 1)\}$
- 14:    $\mathbf{F}_{reinp}(\tau) \leftarrow \mathcal{RI}(F, H) + \mathbf{F}_{int}(\tau)$
- 15:    $FeaturePool \leftarrow FeaturePool + \{\mathbf{F}_{reinp}(\tau)\}$
- 16:    $\tau \leftarrow \tau + 1$
- 17: **end if**
- 18:  $F_{merged} \leftarrow FeatureMerge(FeaturePool)$
- 19:  $I_{out} \leftarrow Reconstruction(F_{merged})$
- 20: **return**  $I_{out}$

---

<sup>§</sup>To simplify the notation in Algorithm 1, We use  $\mathcal{I}$  to express Iterative Inpainting Module, use  $\mathcal{RI}$  to express Re-Inpainting Module

### 3.4.5 Loss Functions

This section describes the loss functions for training the proposed inpainting network. It is a composite loss with multiple terms accounting for different aspects that the proposed algorithm needs to maintain. Perceptual loss and style loss are two of those terms that are popular for solving image generation problems. They are calculated using groundtruth and output feature maps obtained from a pretrained VGG model [25]. *Groundtruth features* are those produced by the max-pooling layers of the VGG network when the input is the complete groundtruth image whereas *output features* are those obtained from the same pooling layers but with the restored image as an input. Formally, the perceptual loss is given by

$$L_{\text{perc}} = \sum_{\theta=1}^N \frac{1}{H_{\theta}W_{\theta}C_{\theta}} |\phi_{\theta}^{\text{gt}} - \phi_{\theta}^{\text{out}}|_1, \quad (3.4.5.1)$$

and the style loss is given by

$$L_{\text{sty}} = \sum_{\theta=1}^N \frac{1}{C_{\theta} \times C_{\theta}} \left| \frac{1}{H_{\theta}W_{\theta}C_{\theta}} (\phi_{\theta}^{\text{gt}} (\phi_{\theta}^{\text{gt}})^T - \phi_{\theta}^{\text{out}} (\phi_{\theta}^{\text{out}})^T) \right|_1, \quad (3.4.5.2)$$

where  $\phi_{\theta}^{\text{gt}}$  denotes the vectorized groundtruth feature map from the  $\theta_{th}$  pooling layer of VGG-16,  $\phi_{\theta}^{\text{out}}$  denotes the vectorized output feature map from the  $\theta_{th}$  pooling layer of pretrained VGG-16, and  $C_{\theta}$ ,  $H_{\theta}$ , and  $W_{\theta}$  are, respectively, the number of channels, height, and width of the  $\theta_{th}$  feature map.

The third term of the composite loss is the total variation loss, which enforces smoothness in the region of predicted pixels (i.e., the holes) [24, 26]. Formally, this term

is formulated as follows. Let  $I_{\text{out}}^{i,j}$  denote the pixel value of output image at location  $i, j$ ,  $N$  denote the total number of elements in the output image,  $\mathcal{R}$  denote the set of pixels surrounding a corrupted pixel  $I_{\text{out}}^{i,j}$ . The total variation loss is given by

$$L_{tv} = \sum_{(i,j) \in \mathcal{R}, (i,j+1) \in \mathcal{R}} \frac{|I^{i,j+1} - I^{i,j}|_1}{N} + \sum_{(i,j) \in \mathcal{R}, (i+1,j) \in \mathcal{R}} \frac{|I^{i+1,j} - I^{i,j}|_1}{N}. \quad (3.4.5.3)$$

The last two terms in the composite loss are first norms of the difference between the output and groundtruth images. Let  $I_{\text{out}}$  denote the output image from the proposed algorithm,  $I_{\text{gt}}$  denote the ground truth image, and  $\tilde{\mathbf{H}}_{\text{gt}}$  denote the groundtruth mask of the image. The two terms are, then, given by

$$L_{\text{valid}} = |I_{\text{out}} \odot \tilde{\mathbf{H}}_{\text{gt}} - I_{\text{gt}} \odot \tilde{\mathbf{H}}_{\text{gt}}|_1, \quad (3.4.5.4)$$

$$L_{\text{hole}} = |I_{\text{out}} \odot (1 - \tilde{\mathbf{H}}_{\text{gt}}) - I_{\text{gt}} \odot (1 - \tilde{\mathbf{H}}_{\text{gt}})|_1, \quad (3.4.5.5)$$

where  $L_{\text{valid}}$  expresses the first norm loss between undamaged region of the output image and the ground truth image, and  $L_{\text{hole}}$  expresses the first norm loss between filled region of the output image and the ground truth image. The composite loss, as the name suggests, is a weighted sum of all the above terms

$$\mathcal{L} = \lambda_{\text{valid}} L_{\text{valid}} + \lambda_{\text{hole}} L_{\text{hole}} + \lambda_{\text{perc}} L_{\text{perc}} + \lambda_{\text{style}} L_{\text{style}} + \lambda_{tv} L_{tv}, \quad (3.4.5.6)$$

where  $\lambda_{\text{valid}}$ ,  $\lambda_{\text{hole}}$ ,  $\lambda_{\text{perc}}$ ,  $\lambda_{\text{style}}$ , and  $\lambda_{tv}$  are all hyper-parameters scaling the contribution of each of their respective terms to the composite loss.

## 3.5 Experimental Setup

The proposed algorithm needs to be put to test in order to demonstrate its performance. This section presents the experimental setup adopted to evaluate its performance. It describes the development datasets, the implementation details, and the benchmark algorithms.

### 3.5.1 Datasets

Four development datasets are adopted here:

- **Paris Streetview Dataset** [27] is collected from Google StreetView, a large-scale dataset that includes street images for 12 cities across the world. This dataset contains 15000 images, 14900 images for training, and 100 for testing.
- **Large-scale CelebFaces Attributes (CelebA) Dataset** [28] is a well-known and publicly available face recognition dataset. It includes around 200K celebrity images representing 10000 different identities, all of which have a wide range of posture variations. This dataset contains 202599 images, 162770 images for training, 19867 images for validation, and 19962 images for testing.
- **Place2 Dataset** [28] This dataset contains 8 million images which are collected from 365 scene categories, like streets, indoor rooms and so on.
- **NVIDIA Irregular Mask Dataset Dataset** [29] is a popular irregular mask dataset. This dataset contains 12000 irregular masks which are randomly drawn by individuals. The mask ratio of the dataset is uniformly distributed on  $0.0\sim 0.1$ ,

0.1~0.2, 0.2~0.3, 0.3~0.4, 0.4~0.5, and 0.5~0.6. Each mask ratio class has 1000 masks with and without border.

Table 3.8: Numerical comparisons on three datasets.

Mask Ratio		PSNR				SSIM				Mean $l_1$			
		10-20%	30-40%	40-50%	50-60%	10-20%	30-40%	40-50%	50-60%	10-20%	30-40%	40-50%	50-60%
PVS	PIC	29.02	24.82	23.71	19.61	0.930	0.795	0.757	0.524	0.0134	0.0327	0.0429	0.0785
	PC	30.42	25.35	23.86	21.47	0.933	0.825	0.763	0.628	0.1001	0.0253	0.0373	0.0612
	EC	30.83	25.94	24.03	21.76	0.945	0.849	0.781	0.653	0.0084	0.0220	0.0297	0.0578
	PRVS	30.87	26.21	24.11	21.94	0.952	0.857	0.794	0.665	0.0085	0.0217	0.0288	0.0567
	RFR	31.74	26.35	24.56	22.43	0.954	0.864	0.811	0.687	0.0076	0.0211	0.0280	0.0543
	MDEFE	31.11	26.23	24.18	22.17	0.954	0.859	0.806	0.674	0.0081	0.0217	0.0287	0.0549
	Ours	<b>31.99</b>	<b>26.86</b>	<b>25.04</b>	<b>22.61</b>	<b>0.964</b>	<b>0.889</b>	<b>0.833</b>	<b>0.721</b>	<b>0.0073</b>	<b>0.0192</b>	<b>0.0268</b>	<b>0.0485</b>
Celeba	PIC	30.69	24.65	21.40	19.16	0.965	0.875	0.739	0.667	0.0094	0.0258	0.0433	0.0756
	PC	32.72	26.59	24.24	22.03	0.970	0.915	0.864	0.782	0.0062	0.0181	0.0269	0.0537
	EC	32.47	26.61	24.43	21.42	0.974	0.921	0.873	0.746	0.0065	0.0185	0.0260	0.0564
	PRVS	33.12	27.05	24.82	22.32	0.977	0.928	0.883	0.793	0.0059	0.0169	0.0252	0.0493
	RFR	33.58	27.62	25.48	22.59	0.980	0.931	0.896	0.822	0.0057	0.0161	0.0224	0.0482
	MDEFE	33.20	27.21	24.97	22.44	0.976	0.925	0.879	0.814	0.0059	0.0167	0.0251	0.0509
	Ours	<b>33.82</b>	<b>27.93</b>	<b>25.92</b>	<b>22.76</b>	<b>0.984</b>	<b>0.945</b>	<b>0.915</b>	<b>0.834</b>	<b>0.0053</b>	<b>0.0150</b>	<b>0.0213</b>	<b>0.0359</b>
Place2	PIC	26.67	21.24	19.13	17.27	0.931	0.761	0.656	0.499	0.0137	0.0342	0.0573	0.0953
	PC	26.84	22.25	20.09	18.33	0.935	0.768	0.722	0.538	0.0131	0.0312	0.0475	0.0861
	EC	27.02	22.41	20.35	18.41	0.934	0.811	0.745	0.531	0.0135	0.0309	0.0447	0.0846
	PRVS	27.35	22.73	20.51	18.72	0.937	0.825	0.773	0.569	0.0140	0.0293	0.0423	0.0769
	RFR	27.90	23.27	21.34	18.87	0.945	0.837	0.782	0.589	0.0114	0.0284	0.0402	0.0726
	MDEFE	27.55	22.92	21.01	18.65	0.941	0.832	0.775	0.582	0.0126	0.0297	0.0411	0.0737
	Ours	<b>28.89</b>	<b>23.66</b>	<b>21.94</b>	<b>19.59</b>	<b>0.957</b>	<b>0.864</b>	<b>0.800</b>	<b>0.667</b>	<b>0.0099</b>	<b>0.0266</b>	<b>0.0368</b>	<b>0.0552</b>

<sup>1</sup> The best performances are highlighted in **bold**.

### 3.5.2 Implementation Details

The proposed algorithm is trained with batch size of 4 on two NVIDIA 1080 TITANs. We use corrupted images, structural maps, and irregular holes as inputs, which are resized to  $256 \times 256$ . Adam [30] optimizer is used to train the network. The training is conducted with a learning rate of  $10^{-4}$ , and the network is fine-tuned with a learning rate of  $10^{-5}$ . The network is trained on Paris and CelebA Dataset for 40 epochs and fine-tuned for 20 epochs. On Place2 Dataset, the network is trained for 200 epochs and fine-tuned for 100 epochs. During the fine-tuning, only the weights of batch normalization layers are frozen while the rest are adjusted. The hyper-parameters of the loss function are set to  $\lambda_{\text{valid}} = 1$ ,  $\lambda_{\text{hole}} = 6$ ,  $\lambda_{\text{perc}} = 0.05$ ,  $\lambda_{\text{style}} = 120$ , and  $\lambda_{tv} = 0.1$ .



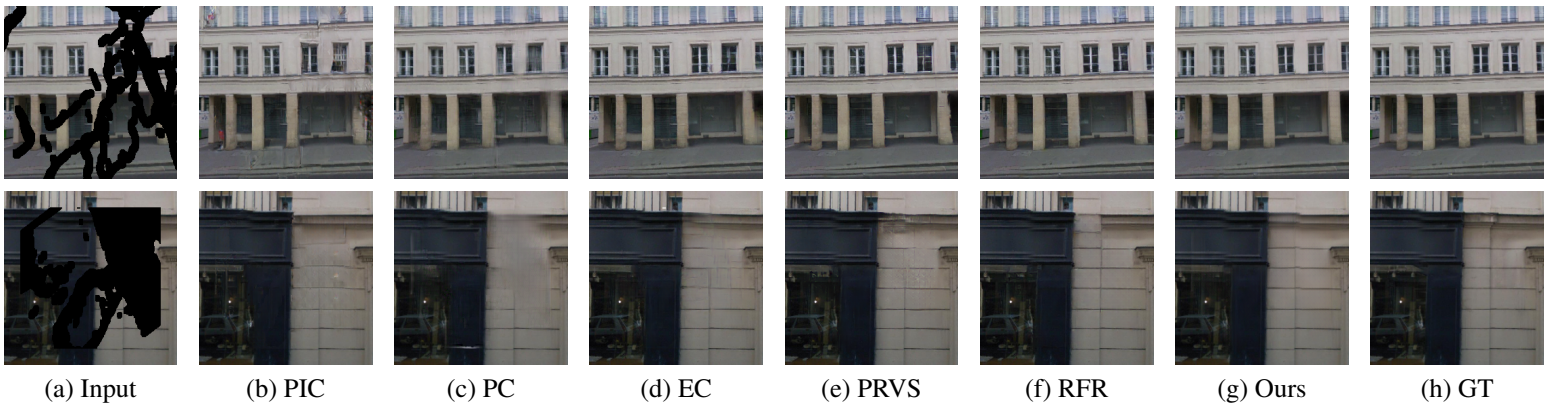


Figure 3.8: Visual results on Paris StreetView.

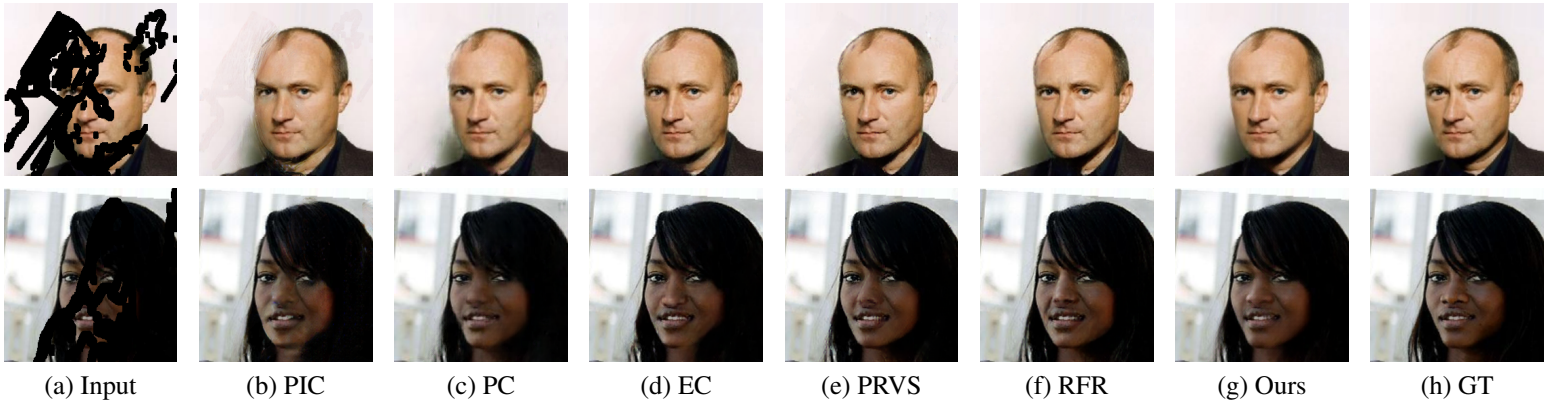


Figure 3.9: Visual results on CelebA.

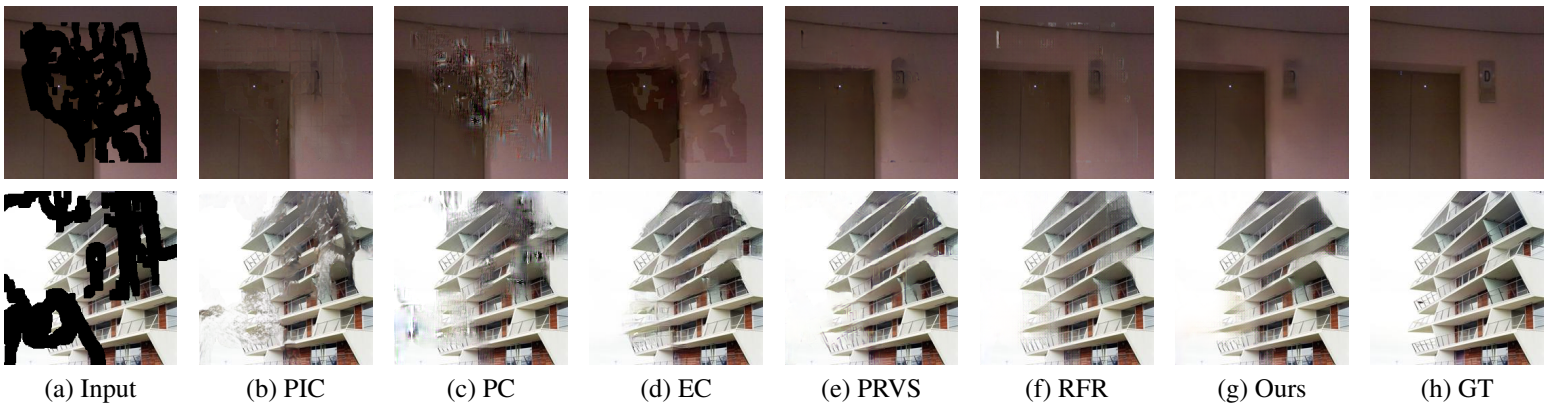


Figure 3.10: Visual results on Place2.

### 3.5.3 Benchmark Algorithms and Evaluation Metrics

The proposed algorithm is compared to five state-of-the-art methods, namely PIC [31], PC [24], PRVS [20], EC [11] and RFR [17]. We use PIC as baseline which is a probabilistically principled framework in image inpainting. PC is a fundamental technique that can be considered as another baseline in image inpainting. EC is a two-stage image inpainting method based on the edge recovering method. PRVS and RFR belong to the family of progressively image inpainting. The PRVS progressively recover the image structural information while RFR recover the image contextual information. Those five algorithms are henceforth referred to as the benchmark algorithms. The proposed algorithm is compared to all five using three metrics, which are Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM) and mean first norm loss ( $L_1$ ).

## 3.6 Evaluation Results

The performance of the proposed algorithm is evaluated in this section using the setup described in Section 3.5. The evaluation starts with quantitative analysis where the proposed algorithm is benchmarked to others. Then, a qualitative analysis follows. It presents a comparison of the quality of inpainted images between the proposed algorithm and the benchmark algorithms. Finally, this section is concluded with an ablation analysis illustrating the value of each novel component in the proposed network.

### 3.6.1 Quantitative Analysis

The proposed algorithm is compared to the benchmark algorithms on the basis of PSNR, SSIM, and  $L_1$  loss. Table 3.8 presents the comparison results on the Paris

StreetView, CelebA, and Place2 datasets. It presents the results for different choices of masking percentage (i.e., mask ratio). The performance of the proposed algorithm stands out throughout the table; despite the slim margin in some cases, its performance could be argued to best all other competing algorithms on all three datasets.

### 3.6.2 Qualitative Analysis

The above quantitative results are translated into visual analysis to demonstrate the inpainting quality of the proposed algorithm. This is done through a few examples from three datasets, namely Paris StreetView, CelebA, and Place2 datasets. Fig. 3.8, 3.9 and 3.10 show three corrupted images, their groundtruth, the inpainted images by the proposed and benchmark algorithms. The proposed algorithm can generate realistic details and structures. Specifically, in the top row of Fig. 3.8, the window produced by the proposed algorithm is clearer than those produced by other methods. Further evidence could be seen in the top row of Fig. 3.9 and both rows of Fig. 3.10. In the former, the hair strands atop the man's forehead are better defined and clearer in the image produced by the proposed algorithm compared to those produced by the benchmark algorithms; they look similar to those strands depicted in the groundtruth image. Both rows of Fig.3.10 show artifacts in the inpainted region by benchmark algorithms while the proposed algorithm does not suffer from such artifacts, producing a more pleasing image to the eye.

Table 3.9: Ablation study results on the Paris Streetview Dataset based on network structure

Ablation setting			PSNR			SSIM			Mean $l_1$		
Reinpainting-1	Reinpainting-2	GLE	10-20%	30-40%	40-50%	10-20%	30-40%	40-50%	10-20%	30-40%	40-50%
X	X	X	31.82	26.70	24.83	0.9632	0.8858	0.8276	0.0074	0.0196	0.0277
X	X	✓	31.90	26.76	24.86	0.9637	0.8859	0.8285	0.0073	0.0195	0.0275
X	✓	X	31.92	26.83	24.93	0.9638	0.8886	0.8314	0.0073	0.0193	0.0272
✓	X	X	31.93	26.83	24.92	0.9640	0.8887	0.8312	0.0073	0.0192	0.0272
✓	X	✓	31.95	26.84	24.96	0.9641	0.8888	0.8317	0.0073	0.0192	0.0271
X	✓	✓	31.99	26.86	25.04	0.9642	0.8890	0.8330	0.0073	0.0192	0.0268

Table 3.10: Ablation study results on the Paris Streetview Dataset based on loss function

Ablation setting			PSNR			SSIM			Mean $l_1$		
$L_{valid}$ & $L_{hole}$	$L_{perc}$ & $L_{style}$	$L_{tv}$	10-20%	30-40%	40-50%	10-20%	30-40%	40-50%	10-20%	30-40%	40-50%
✓	✗	✗	31.72	26.61	24.85	0.9604	0.8817	0.8237	0.0072	0.0193	0.0270
✗	✓	✗	31.69	26.72	24.96	0.9599	0.8830	0.8254	0.0079	0.0200	0.0278
✓	✓	✗	31.86	26.70	24.92	0.9616	0.8823	0.8223	0.0076	0.0199	0.0279
✓	✓	✓	31.99	26.86	25.04	0.9642	0.8890	0.8330	0.0073	0.0192	0.0268

### 3.6.3 Ablation Analysis on Proposed Architecture

The proposed architecture is closely examined to get a better understanding of the role of the novel components. More to the point, the GLE module and the repainting component are novel parts that set the proposed architecture apart from the other inpainting algorithms. Therefore, this section will focus on shedding some light on their roles in the inpainting process. The objective is to address the question: *how much of an impact do the GLE module and the repainting component have on the performance of the algorithm?* This is going to be done in three experiments. The first has the two parts removed and the performance of the remaining architecture is evaluated. This helps establish the baseline results. The other two experiments examine the impact of adding each of the two parts, i.e., GLE and repainting, separately on the inpainting performance. The results of the three experiments are shown below.

#### Removing the GLE Module and Repainting Component

The GLE module and repainting component are both removed from the proposed architecture. To avoid jeopardizing the capacity of the proposed model, the GLE module is removed by stripping away the Gaussian smoothing and upsampling layers making a direct path from the first to the second convolution layers of the module.

Removing both parts chips away from the inpainting performance of the architecture. This is evident in Table 3.9; with all three metrics, the table shows a clear degradation in performance on the Paris StreetView dataset when the architecture is trained and tested without the GLE module and the repainting component.



### **Removing the GLE Module**

Using the same removal strategy in the above section (Section 3.6.3), the GLE module is removed in this experiment while keeping the repainting component. The result of doing so is a slight improvement in the performance compared to the baseline case, i.e., no GLE and repainting, as Table 3.9 shows. However, the performance is still worse compared to having both parts plugged in. The results of this experiment could be used to argue for the value of the GLE module; it helps the proposed architecture extract expressive features from different frequency components of the image.

### **Removing the Repainting Component**

Using the same removing strategy once again, the repainting component is removed while keeping the GLE module. It is hypothesized that repainting has the ability to fill larger holes by accessing features from neighboring iterations. The results in Table 3.9 verify that hypothesis to some extent; removing the repainting component degrades the performance of the architecture despite the presence of the GLE module.

### **Value of progression for repainting**

The hypothesis about the repainting component being able to fill large holes is further examined here. More to the point, it will be argued that accessing sub-volumes from different iterations (i.e.,  $\mathbf{F}_{\text{int}}(\tau - 1)$  and  $\mathbf{F}_{\text{int}}(\tau + 1)$ ) has added value to the inpainting process. This is first done by restricting the input to the re-inpainting component to only the  $\tau$ -th feature sub-volume, i.e.,  $\mathbf{F}_{\text{int}}(\tau)$ . Then, two experiments are conducted with and without the GLE components.

- The GLE modules are removed as described in Section 3.6.3. The performance in

this case is very close to that of removing the whole inpainting component. This is indicated in Table 3.9 under Reinpainting-2. This verifies that the  $\mathbf{F}_{\text{int}}(\tau)$  and  $\tilde{\mathbf{H}}(\tau)$  can not provide more useful information for the inpainting process. The redundant information even slightly causes the performance to degrade.

- The GLE modules are put back and the experiment is repeated again. Again, the results, shown in Table 3.9, further verify that the input features from neighbouring iterations are useful for enhancing the re-inpainting results.

### 3.6.4 Ablation Analysis on Loss Function

The performance of the proposed method is further investigated based on each component of the loss function. The ablation study settings for ablation studies on loss function are into three parts:  $L_{\text{valid}}$  and  $L_{\text{hole}}$  monitors the L1 loss on reconstructed region and the inpainted region,  $L_{\text{perc}}$  and  $L_{\text{style}}$  minimize the differences on the feature level,  $L_{\text{tv}}$  penetrates the noises in the inpainting results. In the following three experiments, we present the ablation results in Table 3.10 and verify the effectiveness of each part of the loss function.

#### Using $L_{\text{valid}}$ and $L_{\text{hole}}$

Using  $L_{\text{valid}}$  and  $L_{\text{hole}}$  as the only terms of the loss functions, we observe that the mean  $l_1$  is slightly better compared to the final results; however, the PSNR and SSIM are not satisfactory especially for the case of corrupted images with large holes. The undesirable performance indicates that the loss has to account for the semantic information in the inpainted image.

### Using $L_{perc}$ and $L_{style}$

Constructing the loss function using  $L_{perc}$  and  $L_{style}$  alone, we observe an improvement in the PSNR and SSIM for large holes. The performance shows the role and effectiveness of  $L_{perc}$  and  $L_{style}$  in discovering semantic information in the valid region; however, the mean  $l_1$  loss increases significantly due to the lack of  $L_{valid}$  and  $L_{hole}$ .

### Using $L_{valid}$ , $L_{hole}$ , $L_{perc}$ and $L_{style}$

After using the  $L_{valid}$ ,  $L_{hole}$ ,  $L_{perc}$  and  $L_{style}$  as loss functions, the PSNR, SSIM and mean  $l_1$  enhance in the low mask ratio. On the other hand, the PSNR and SSIM are slightly worse than that of using  $L_{perc}$  and  $L_{style}$ . The existing noises may be generated by loss functions from the pixel level and feature level, and cause this situation. To eliminate such noise, the  $L_{tv}$  is finally added to the loss function and improves performance. After using the  $L_{valid}$ ,  $L_{hole}$ ,  $L_{perc}$  and  $L_{style}$  as loss functions, the PSNR, SSIM and mean  $l_1$  enhance in the low mask ratio. On the other hand, the PSNR and SSIM are slightly worse than that of using  $L_{perc}$  and  $L_{style}$ . The existing noises may be generated by loss functions from the pixel level and feature level, and cause this situation. To eliminate such noise, the  $L_{tv}$  is finally added to the loss function and improves performance.

## 3.7 Conclusion

This paper introduces a three-stage neural network architecture that is able to progressively inpaint corrupted images while maintaining their structural and contextual integrity. In its core is a novel Gaussian-Laplacian feature Extraction (GLE) module. Stacking GLE modules constructs the first stage of the architecture and enables the

network to build a feature pyramid of different frequency components, disintegrating structural (high frequency) and contextual (low frequency) information. The feature pyramid is the key for structurally- and contextually-aware progressive inpainting; low- and high-frequency components are iteratively but separately inpainted and fused in the second stage. The third, and final, stage enhances the fused features before it reconstructs the inpainted image. Experimental results and benchmarking show that the three-stage architecture is able to restore fine details in the corrupted region, outperforming the state-of-the-art algorithms. Ablation experiments reveal that the GLE module and the repainting component are responsible for the superior performance of the proposed architecture.

# Bibliography

- [1] J. Jam, C. Kendrick, K. Walker, V. Drouard, J. G.-S. Hsu, and M. H. Yap, “A comprehensive review of past and present image inpainting methods,” *Computer Vision and Image Understanding*, vol. 203, p. 103147, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1077314220301661>
- [2] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, “Image inpainting,” in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000, pp. 417–424.
- [3] L. Liao, R. Hu, J. Xiao, and Z. Wang, “Artist-net: Decorating the inferred content with unified style for image inpainting,” *IEEE Access*, vol. 7, pp. 36 921–36 933, 2019.
- [4] L. Liao, J. Xiao, Z. Wang, C.-W. Lin, and S. Satoh, “Guidance and evaluation: Semantic-aware image inpainting for mixed scenes,” in *European Conference on Computer Vision*. Springer, 2020, pp. 683–700.
- [5] L. Liao, J. Xiao, Z. Wang, C.-W. Lin, and S. Satoh, “Image inpainting guided by coherence priors of semantics and textures,” in *Proceedings of the IEEE/CVF*

*Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6539–6548.

- [6] L. Liao, J. Xiao, Z. Wang, C.-W. Lin, and S. Satoh, “Uncertainty-aware semantic guidance and estimation for image inpainting,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 2, pp. 310–323, 2020.
- [7] Z. Qin, Q. Zeng, Y. Zong, and F. Xu, “Image inpainting based on deep learning: A review,” *Displays*, vol. 69, p. 102028, 2021.
- [8] C. Guillemot and O. Le Meur, “Image inpainting : Overview and recent advances,” *IEEE Signal Processing Magazine*, vol. 31, no. 1, pp. 127–144, 2014.
- [9] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, “Patchmatch: A randomized correspondence algorithm for structural image editing,” *ACM Trans. Graph.*, vol. 28, no. 3, p. 24, 2009.
- [10] Y. Ren, X. Yu, R. Zhang, T. H. Li, S. Liu, and G. Li, “Structureflow: Image inpainting via structure-aware appearance flow,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 181–190.
- [11] K. Nazeri, E. Ng, T. Joseph, F. Qureshi, and M. Ebrahimi, “Edgeconnect: Structure guided image inpainting using edge prediction,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019.
- [12] Z. Guo, Z. Chen, T. Yu, J. Chen, and S. Liu, “Progressive image inpainting with full-resolution residual network,” in *Proceedings of the 27th acm international conference on multimedia*, 2019, pp. 2496–2504.

- [13] Y. Zeng, Z. Lin, J. Yang, J. Zhang, E. Shechtman, and H. Lu, “High-resolution image inpainting with iterative confidence feedback and guided upsampling,” in *European Conference on Computer Vision*. Springer, 2020, pp. 1–17.
- [14] X. Guo, H. Yang, and D. Huang, “Image inpainting via conditional texture and structure dual generation,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 134–14 143.
- [15] C. Hsu, F. Chen, and G. Wang, “High-resolution image inpainting through multiple deep networks,” in *2017 International Conference on Vision, Image and Signal Processing (ICVISIP)*. IEEE, 2017, pp. 76–81.
- [16] H. Zhang, Z. Hu, C. Luo, W. Zuo, and M. Wang, “Semantic image inpainting with progressive generative networks,” in *Proceedings of the 26th ACM international conference on Multimedia*, 2018, pp. 1939–1947.
- [17] J. Li, N. Wang, L. Zhang, B. Du, and D. Tao, “Recurrent feature reasoning for image inpainting,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7760–7768.
- [18] W. Xiong, J. Yu, Z. Lin, J. Yang, X. Lu, C. Barnes, and J. Luo, “Foreground-aware image inpainting,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5840–5848.
- [19] K. Nazeri, E. Ng, T. Joseph, F. Z. Qureshi, and M. Ebrahimi, “Edgeconnect: Generative image inpainting with adversarial edge learning,” *arXiv preprint arXiv:1901.00212*, 2019.

- [20] J. Li, F. He, L. Zhang, B. Du, and D. Tao, “Progressive reconstruction of visual structure for image inpainting,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5962–5971.
- [21] M. S. Farid, H. Khan, and A. Mahmood, “Image inpainting based on pyramids,” in *IEEE 10th INTERNATIONAL CONFERENCE ON SIGNAL PROCESSING PROCEEDINGS*. IEEE, 2010, pp. 711–715.
- [22] J. H. Lee, I. Choi, and M. H. Kim, “Laplacian patch-based image synthesis,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2727–2735.
- [23] S. Padmavathi and K. Soman, “Laplacian pyramid based hierarchical image inpainting. adv,” *Image and Video Processing (AIVP)*, vol. 2, no. 1, pp. 9–22, 2014.
- [24] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, “Image inpainting for irregular holes using partial convolutions,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 85–100.
- [25] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [26] A. Mahendran and A. Vedaldi, “Understanding deep image representations by inverting them,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5188–5196.
- [27] C. Doersch, S. Singh, A. Gupta, J. Sivic, and A. Efros, “What makes paris look like paris?” *ACM Transactions on Graphics*, vol. 31, no. 4, 2012.



- [28] Z. Liu, P. Luo, X. Wang, and X. Tang, “Deep learning face attributes in the wild,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3730–3738.
- [29] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, “Nvidia irregular mask dataset,” in <https://nv-adlr.github.io/publication/partialconv-inpainting>, 2018.
- [30] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *ICLR (Poster)*, 2015. [Online]. Available: <http://arxiv.org/abs/1412.6980>
- [31] C. Zheng, T.-J. Cham, and J. Cai, “Pluralistic image completion,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1438–1447.
- [32] H. Liu, B. Jiang, Y. Song, W. Huang, and C. Yang, “Rethinking image inpainting via a mutual encoder-decoder with feature equalizations,” in *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer, 2020, pp. 725–741.

# Chapter 4

## Conclusion and Future Work

### 4.1 Conclusion

This thesis presents several feature extraction and feature transformation techniques based on deep CNNs, and apply them to computer vision problems like image retrieval and image inpainting.

In Chapter 2, we present a CCA-based dimension reduction method for deep learning features, and apply it to image retrieval. The proposed method uses Canonical Correlation Analysis as a powerful tool to find a projection pair that maximally captures the correlation between two sets of random variables. This characteristic makes it suitable for image retrieval, which requires the evaluation of similarity between two images. The proposed method consists of three parts: 1) Correlation analysis and canonical vectors. This part is motivated by traditional CCA, and finds the projector matrix to maximize the correlation between two image feature vectors. 2) Feature element selection. This part is inspired by Chernoff information which is used to find a symmetric distance between two distributions. Benefiting from this idea, we can choose the most different feature points, and use them for

similarity comparison. 3) Similarity measurement. The multivariate Gaussian distribution is used to calculate the similarity between two images. Extensive comparison experiments and ablation studies are conducted to demonstrate the superior performance against other dimension reduction methods like PCA, LDA, and SPCA, and the effectiveness of each part of the proposed method.

In Chapter 3, we present a feature extraction network and a feature transformation framework with attention for image inpainting. Overall, the proposed method includes three novelties: 1) Gaussian and Laplacian Pyramid motivated feature extraction network. Compared with the traditional encoder-structured feature extractor, the proposed network can generate better contextual and structural information for image inpainting. 2) A two-branch inpainting network simultaneously utilizes contextual and structural information to progressively recover the image. 3) A repainting network, which is appended to the proposed two-branch inpainting network, enhances the intermediate features during the progressive inpainting process. The experimental results validate the superiority of employing contextual and structural information on progressive inpainting as well as the effectiveness of image inpainting framework design.

## 4.2 Future Work

Even though the proposed methods have achieved superior performance compared with other SOTA approaches, some improvements can still be made to increase the model's efficiency and performance.

The proposed method in the first work is built on pre-trained VGG-16 without further training. To improve the performance, we can realize the proposed method using an end-to-end trainable CCA network. Specifically, a two-branch convolutional neural

network is used to extract feature vectors from two input images. The ground truth can be made to suit network training using one-shot coding. And the feature selection based Chernoff Information can be replaced with feature attention techniques which has been widely used in many deep learning tasks. The potential difficulties include maintaining the storage space for feature vectors which are used to deriving matching and non-matching coefficients, and making all formulas differentiable in the algorithm. Theoretically, the proposed method could be converted into a deep learning model which can take advantage of end-to-end training. In theory, end-to-end training can boost the model's performance in deep learning.

In the second work, Gaussian and Laplacian Pyramid motivated feature extraction network can be re-designed to reduce the number of parameters and increase model efficiency. For example, we can use Gaussian and Laplacian Pyramid algorithms to generate multi-scale sub-images directly and remove some convolutional layers as well as all Gaussian smooth layers in the feature extraction network. These sub-images are used to guide the feature extraction process. Specifically, we use a lightweight-shared network to project extracted features into 3-dimensional images that are supervised by the multi-scale sub-images. Therefore, the extracted features can represent the sub-images in high dimensions. The idea of progressive inpainting is now applied to each sub-image. The resulting sub-images are then used to recover the original image via the Gaussian and Laplacian Pyramid. In detail, we simply use partial convolutional layers to recover each sub-images on the feature level, and rebuild the original image by passing recovered features through a reconstruction network with the same settings of the feature extraction network. Moreover, generative adversarial network is another powerful tool in many image synthesis tasks, and typically offers better image generation ability. In consideration of

model efficiency, we choose not to use it in our method. Nevertheless, the trade-off between performance and model efficiency is an interesting problem to consider.