博士論文（要約）

Empirical Orthogonal Decomposition and Euclidean Embeddings for Image Feature Extraction

(経験的直交分解とユークリッド埋め込みに基づく画像特徴抽出)

椋田 悠介

This thesis is dedicated to my beloved family.

Abstract

For the practical use of image recognition technology, it is necessary that not only the recognition method demonstrates high accuracy but also the method is efficient and the theoretical property of the method is well understood.

As a feature that satisfies such property, we propose a novel framework that applies the orthogonal decomposition of the kernel with the distribution inferred from input data as a feature vector and Euclidean embedding of bag of local features in low-dimensional euclidean space. Then we combine our framework with the existing effective approach that first extracts local features densely, then summarizes the local features into one global feature considering the feature value and position in the image, and finally outputs the category of the image from the global feature. Since we can get the feature vector as an approximation of the kernel function, we can analyze the classification performance with the kernel approximation performance. Also, we can effectively approximate the kernel function using data information. Thus we can obtain informative feature even when the dimension is small. Our euclidean embedding enables us to exploit prior knowledge in feature extraction and handle the feature vector easily.

In this work, we apply our feature extraction method based on orthogonal decomposition to local feature extraction, feature encoding, and feature pooling and propose a novel method in each module. For local feature extraction, we apply our method to convolutional kernel between local image patches that summarizes the nonlinear similarity between pixels considering the pixel position in the patch and propose a patch feature that is effective even with the small feature dimension. For feature encoding, we apply our method for covariance encoding and propose a feature method with small feature dimension even when the input feature dimension is large. For feature pooling, we propose a framework that regards bag of local features from one image as a function from image plane to feature space and regards feature pooling as an orthogonal projection in function space. With this idea, we

can see bag of local features as a point in function space and thus we can apply our method to construct a novel feature pooling method.

Experimental results using standard image recognition datasets show that the proposed method demonstrates compact and effective image feature. Thus, the proposed method not only has good theoretical property but also is efficient for practical use.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Image recognition is a task to make the computer understand what is drawn in the image. As the computer resource and data size grow, image recognition becomes more and more popular, and the task becomes more and more complex such as outputting the description of the image or detecting the position of the objects in the image. However, even for such advanced task, the fundamental feature extraction technique that extracts one global feature vector from the input image affects most on the recognition performance. Since these problems are solved as a regression from image to the output labels such as the category of objects, description, detection window, we can not conduct accurate recognition if we do not extract enough discriminative information from the input image.

Now we define the condition that feature extraction method should satisfy. As for recognition accuracy, first is to exploit the prior knowledge about the input image. Second is to extract the information from training data. The prior knowledge means the knowledge about what information from the input image is informative or meaningless for the recognition. This knowledge has been used for Otsu's invariant feature extraction theory and hand-crafted local features such as SIFT, HOG, and GIST. The success of Convolutional Neural Networks (CNN) arises from the invention of Convolutional layer that exploits the prior knowledge of local translation invariance. Also, considering the effectiveness of multivariate analysis methods such as principal component analysis and canonical correlation analysis and the success of CNN method that learns most of the feature from the image, it is necessary for the performance improvement to compensate for the knowledge hard to design in prior with the statistical information obtained from the training data.

From the viewpoint of application, the first requirement is that the cost for training the model and calculation of the feature is small. We can not always

Method	Requirement			
	Performance	Prior Knowledge	Unsupervised	Analysis
hand-crafted feature	×	✓	✓	✓
CNN	✓	Translation invariance	×	Partial
Our method	Good with appropriate kernel	✓	✓	✓

Table 1.1: Comparison of existing feature extraction methods and the proposed method.

get large-scale labeled training data for recognition tasks. Since the cost of collecting the image and assigning labels is large, we often learn the classifier from small-sample or unlabeled training data. Also, when we want to conduct image recognition with small computation resource such as smart device, we need to make the training and classify cost small. The second requirement is the easiness of the analysis of the performance of the recognition method. For the application that misclassification causes crucial result such as automatic driving and medical image analysis, the estimation of classification performance for test data is important. From a machine learning viewpoint, statistical learning theory provides a method to evaluate the test performance using the performance of training data and the complexity measure of the space of recognition model to be learned. It is desirable to obtain the similar estimation of the performance of the model.

When we summarize the existing models from this viewpoint, there are two major existing models. The first model constructs the global feature using hand-crafted local features such as SIFT and encoding such as principal component analysis and Fisher Vector and then train linear classifier. The second is based on Convolutional Neural Network (CNN). The first model can exploit the prior knowledge and modify the feature dimension by adjusting the model parameter, and we can analyze the performance if we try because the parts that learn the model from the training data is relatively simple. However, the classification performance is weaker than the CNN that learns most of the parameter from the training data for large-scale classification. On the other hand, CNN shows good performance if there are enough labeled training data, but it is hard to analyze the classification performance and the method for training the model with unlabeled or small-sample data is not yet well understood. Also, it is difficult to exploit the prior knowledge without time-consuming data augmentation.

1.2 Objective

In this research we propose a novel feature extraction framework that satisfies the requirements we stated in the previous section. To this end, we introduce two functions, Empirical Orthogonal Decomposition of the kernel function and

Euclidean Embedding to low-dimensional space.

Since the information of the image is not linear to the pixel value, image feature extractor is essentially a nonlinear function. Kernel method that uses nonlinear similarity function between data as a feature is effective in handling such nonlinearity. Kernel method has several good properties for image recognition. First is that several image feature extraction methods such as fisher vector are derived from kernel method. Second is that we can incorporate the prior knowledge about invariance well by using shift-invariant kernel, the function value of which is calculated only by the difference between two inputs. Third is that the generalization performance of the kernel function is well understood. Thus, we use kernel method as a source for nonlinearity. However, since kernel method requires much computation complexity when the number of samples is large, we need to approximate the kernel function with the linear inner product of nonlinear functions. To this end, we propose a novel data-driven approximation method called Empirical Orthogonal Decomposition with good approximation performance that can be calculated fastly.

Euclidean space is the most fundamental space for feature space. Most of the classifier such as linear support vector machine assumes that the input vector lies in Euclidean space. Also, as we describe in Chapter 3, we fit the Gaussian Distribution to the input data. Thus, we require that the input data are the low dimensional Euclidean vectors. However, the input of feature extraction module is not always the vector, but often it is the bag of local features. To this end, we construct a Euclidean Embedding from bag of local features to one low-dimensional vector.

Since local features have information about both the values and the positions in the image, exploiting both information is necessary for the informative feature extraction. Our strategy is to apply nonlinear transformation to both value part and position part, and then use the summation of tensor products of transformed value vectors and transformed position vectors as embedded features. We use the summation because it is the most simple statistics that is invariant to the change of the order of the local features. As a principle for each value and position embedding, we first construct large-dimensional space with geometry designed from prior knowledge, and then extract low-dimensional low-frequency parts with respect to the geometry.

In this framework, we encode the prior knowledge in the kernel design and construction of Euclidean Embedding, and calculate the feature function with the distribution from training data. Thus the method satisfies the two condition for performance. Also, we can modify the dimension of the feature by adjusting the number of eigenvectors we use. We can evaluate the classification performance with the performance of the kernel function itself, the approximation performance of the kernel function, and estimation performance of the distribution. Thus the

method also satisfies the two condition for application.

In the previous section, we reviewed that existing approaches are roughly divided into the method that adopts manually-designed local descriptors from prior knowledge and trains the classifier from the feature, and the method that learns the end-to-end model with CNN. These approaches look different at first glance, but they share the same framework that first extracts the local feature from image patches and then summarizes the local features into one global feature and outputs the image category. Thus this framework is considered to be effective for image recognition and we follow this framework.

Then, we derive the novel method for feature extraction that extracts feature from the image subregion, feature encoding that encodes the statistics of the extracted local feature into high-dimensional feature space, and feature pooling that summarizes the local features in one global feature. We mainly handle image feature extraction, but we can apply our framework to different modal such as movie and sound recognition.

In summary, our contributions are as follows:

- We proposed a novel image feature extraction framework based on the orthogonal decomposition of the kernel function with distribution estimated from input data and Euclidean embedding of bag of vectors.
- We derive the main module of image feature extraction from our framework.
- We apply the method derived in our framework to image recognition benchmark and showed that our method can show good classification accuracy with small feature dimension.

1.3 Structure of the Thesis

In this work, we propose a feature extraction framework based on the orthogonal decomposition of kernel function using data information and euclidean embedding of bag of vectors. We first describe our proposed framework and its analysis. Then we propose a novel method for local feature extraction, feature encoding and feature pooling and experimentally evaluate the performance.

This thesis is organized as follows: we have already described the background and the objective of this thesis in Chapter 1. In Chapter 2, we describe the related work of image feature extraction and the analysis. In Chapter 3, we describe the detail of the proposed framework. This framework enables us to extracts image feature that satisfies the above 4 conditions. Thereafter, we derive the feature extraction method that corresponds to each module of image feature extraction from our framework. In Chapter 4, we apply our framework to local

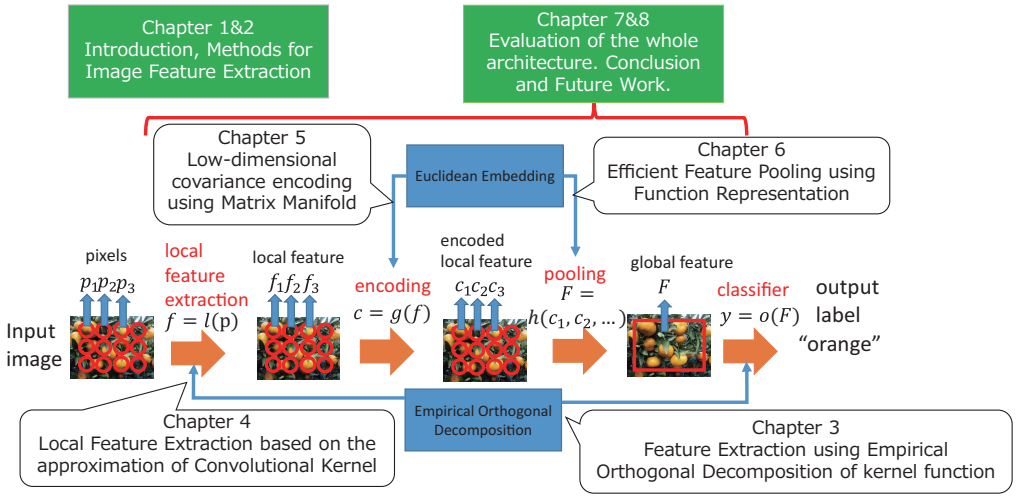


Figure 1.1: Overview of the proposed feature extraction system and the construction of the thesis.

feature extraction and combine our method to Convolutional Kernel Networks to demonstrate that we can extract informative local descriptor. In Chapter 5, we apply our framework to the covariance encoding and derive a method that suppresses the growth of feature dimension. In Chapter 6, we apply our framework to feature pooling method and derive a smooth and independent feature pooling weight. In Chapter 7, we combine the method proposed in Chapter 4, 5, 6 and evaluate the classification performance. Finally, Chapter 8 concludes this thesis and describes future works. Figure 1.1 shows the overview of our system.

Chapter 2

Methods for Image Feature Extraction

In this section, we describe the existing research about image feature extraction method for object recognition. We review existing two major approach, manually-designed image features and CNN. We describe that these approaches share the idea that first extracts the local feature from image patches and then summarizes the local features into one global feature and outputs the image category.

2.1 Manually-Designed Image Features

In the early stage, we first manually design local descriptors based on prior knowledge, then encode the local descriptors using the statistical model, and finally apply linear classifier on the global feature to recognize the image category. We introduce the research for each module.

2.1.1 Invariant Features

In image recognition, the idea of constructing global image features from invariants has mainly been applied to geometric transformations that preserve the image information, such as rotation, expansion, translation, and photometric invariants.

Hand-crafted local descriptors used the information we want to extract from the image as a prior and are mainly derived from geometric invariance. The popular local descriptors are SIFT [1], HOG [2], LBP [3], and GIST [4]. The SIFT feature [1] was constructed as a local feature robust to rotation, scale, and photometric change, PCA-SIFT [5] uses principal components of the gradient information histogram, GLOH [6] is a more robust extension of SIFT that consider

a log-polar location grid, while ASIFT [7] also considers affine transformation.

There are also works that directly derive global feature from invariance. Hu [8] used moment invariants as a translation-invariant feature and constructed more complex invariant features from these. Kondor [9] constructed translation- and rotation-invariant features by mapping the image onto a sphere and using harmonic analysis on a three-dimensional special orthogonal group. Mairal et al. [10] proposed a hierarchical model in which image region similarity is the sum of the kernel value between features of the subregions times the kernel value between 2-d positions of the subregions. Its robustness to the local translation can be adjusted by changing the kernel parameters. Anselmi *et al.* [11] proposed a patch feature that uses the mean of a nonlinear function on the image patch under some group action and constructed a hierarchical convolutional architecture that is robust to group action. Mallat [12] constructed an operator on image space that is translation-invariant and robust to diffeomorphisms by using the scattering operator that repeatedly calculates the absolute values of wavelet coefficients.

2.1.2 Feature Encoding

Bag of Visual Words [13] is the basis of the statistics-based approach, which clusters the local descriptor space and codes the number of local descriptors assigned to each cluster. It is known that a bag-of-words based approach that discards the position of local descriptors is robust to translation. The FV [14] models the distribution of the local descriptors of all images as a Gaussian mixture model (GMM) and uses the difference between the distribution of each image and the model as the feature and the Fisher kernel as the similarity measure. Let N be the number of local descriptors, f_n be the n-th local descriptor, D be the dimensionality of the local descriptors, K be the number of clusters, ω_k be the mixture weight of the k-th component, and μ_k^d, σ_k^d be the mean and standard deviation, respectively, of the d-th element in the k-th component. The FV is a $2KD$-dimensional vector organized as follows for $d \in 1..D, k \in 1..K$:

$$F_{kd} = \begin{pmatrix} \frac{\lambda_{\mu_k^d}^{-\frac{1}{2}}}{N\sqrt{\omega_k}} \sum_{n=1}^{N} \gamma_n(k) \left(\frac{f_n - \mu_k^d}{\sigma_k^d} \right) \\ \frac{\lambda_{\sigma_k^d}^{-\frac{1}{2}}}{N\sqrt{2\omega_k}} \sum_{n=1}^{N} \gamma_n(k) \left(\frac{(f_n - \mu_k^d)^2}{(\sigma_k^d)^2} - 1 \right) \end{pmatrix}, \tag{2.1}$$

where $\gamma_n(k)$ is the posterior probability of the n-th component, $\gamma_n(k) = \frac{\omega_k \mathcal{N}_k(x_n)}{\sum_{j=1}^{K} \omega_j \mathcal{N}_j(x_n)}$, where $\mathcal{N}_k$ is the probability density of the k-th Gaussian distribution and $\lambda_{\mu_k^d}, \lambda_{\sigma_k^d}$ are approximated diagonal elements of the Fisher information matrix set to $\frac{N\omega_k}{(\sigma_k^d)^2}, \frac{2N\omega_k}{(\sigma_k^d)^2}$, respectively.

The vector of locally aggregated descriptors (VLAD) [15] was introduced to simplify the FV and accelerate its computation. The VLAD is a KD-dimensional vector that uses k-means clustering and consists of the sum of the difference from each local descriptor to the cluster centroid μ_k to which it is assigned.

$$F_k = \sum_{f_n \in S_k} (f_n - \mu_k), \tag{2.2}$$

where S_k is the set of local descriptors that are assigned to the k-th cluster.

The vector of locally aggregated tensors (VLAT) [16] is an extension of the VLAD that uses the sum of tensor products of the difference from each local descriptor to the cluster centroid μ_k.

$$F_k = \text{upper}\left(\sum_{f_n \in S_k} (f_n - \mu_k)(f_n - \mu_k)^t - \mathcal{T}_k\right), \tag{2.3}$$

where upper (A) is the vector that stores the upper triangular elements of A and $\mathcal{T}_k$ is the mean of $(f_n - \mu_k)(f_n - \mu_k)^t$ of all the local descriptors assigned to the k-th cluster. The VLAT contains information that is similar to the full covariance of the GMM.

The GLC [17] uses the mean and covariance of the local descriptors as the feature. The global Gaussian [18] regards the GLC as a sufficient statistic of the Gaussian distribution and uses an information geometric metric on the statistical manifold of the Gaussian distribution as the similarity measure. The GLC is defined as:

$$F = \begin{pmatrix} \frac{1}{N}\sum_{n=1}^{N} f_n \\ \frac{1}{N}\text{upper}\left(\sum_{n=1}^{N} f_n f_n^t\right) \end{pmatrix}. \tag{2.4}$$

Moreover, the matrix logarithm for second-order statistics is considered for the region feature [19, 20]. Serra *et al.* [21] considered the covariance of pixel features as well as the covariance of patch features.

The graphical Gaussian vector (GGV) [22] uses local convolutional information of adjacent local descriptors in the image by modeling the image as a Gauss-Markov random field. The GGV combines the GLC and the following:

$$F_{\mathcal{E}_i} = \frac{1}{N}\text{vec}\left(\sum_{(j,k)\in\mathcal{E}_i} f_j f_k^t\right), \tag{2.5}$$

where vec (A) is the vector that stores the elements of A and $\mathcal{E}_i$ is the set of pairs of local descriptors with some adjacency relation.

2.1.3 Classifier

In large-scale image recognition, often the feature dimension and the number of data become large. Thus in most case, we train linear classifier with optimization method based on stochastic gradient descent. Linear classifier is a classifier that given input data $x \in \mathbb{R}^d$, outputs the category of the data from the inner metric to the weight $w \in \mathbb{R}^d$ written as $w^t x$. The most used method is soft-margin SVM. Given training data $\{(x_i, y_i)\}_{i=1}^n$ where $x_i \in \mathbb{R}^d, y_i \in \{-1, 1\}$, the loss function of w to be optimized is

$$\frac{\|w\|^2}{2} + C \sum_{i=1}^{n} \max(0, 1 - y_i w^t x_i), \tag{2.6}$$

where C is a hyper parameter.

In this case, the update rule of stochastic gradient descent is

$$w \leftarrow w - \eta \begin{cases} w & y_i w^t x_i > 1 \\ w - nCy_i x_i & \text{otherwise} \end{cases}. \tag{2.7}$$

In practice, we use faster optimization method such as stochastic averaged gradient [23], stochastic variance reduced gradient [24], and stochastic dual coordinate ascent [25].

There is an analysis of the performance of SVM based on Rademacher Complexity. When we denote the model space of w as $\|w\| \leq \Lambda$ and assume that the input data x satisfies $\|x\| \leq r$, with probability $1 - \delta$, the difference of expected ρ-margin loss and empirical ρ-margin loss is bounded by

$$2\sqrt{\frac{r^2\Lambda^2/\rho^2}{m}} + \sqrt{\frac{\log 1/\delta}{2m}}. \tag{2.8}$$

Thus, the difference is small if the model size is small and the number of training sample is large. There also exists the finer bound using stability analysis and PAC-Bayes analysis.

In some case, we apply online learning method such as Passive Aggressive [26] or Confident-Weighted Learning [27] if the training data is not fixed. These methods are analyzed with regret analysis.

2.2 Convolutional Neural Networks

In the previous section, we described the framework based on the combination of the hand-crafted image feature and linear classifier. In this section, we introduce CNN that train such image feature and classifier at once. Since the model to

be trained is more complex than the previous approach, we need much training data. The success of CNN research depends much on the large-scale data such as MNIST [28], CIFAR-10 [29], and ImageNet [30].

Deep Neural Network is a method that repeatedly applies learned linear transformation and predefined nonlinear transformation to the input and directly outputs the category. Convolutional Neural Network is a method that incorporates convolutional layer into Deep Neural Network. This convolutional layer corresponds to the local feature extraction part and contributes to the robustness to the local image translation.

Convolutional Neural Networks become popular after the success of AlexNet [31] in ILSVRC2012. Then the architecture becomes more and more deep such as VGG-Net [32], GoogLeNet [33] and ResNet [34].

Like the case of manually-designed feature, researchers have constructed a global feature by encoding the outputs of DNNs as local descriptors. Cimpoi *et al.* [35] proposed to use the outputs of the convolutional layers as local descriptors and apply a Fisher Vector [36] to encode them into a global feature. They showed that their global feature showed improved performance compared to the outputs of the fully connected layer in a texture recognition task. Babenko & Lempitsky showed that simple sum pooling showed good performance in an image-retrieving task. Lin *et al.* proposed a coding method based on bilinear CNN for a fine-grained recognition task. To construct the sentence feature, Klein *et al.* [37] proposed to regard the word vector [38] in each word as local descriptors and use the Laplacian Mixture Model (LMM) instead of the Gaussian Mixture Model (GMM) for extracting the Fisher Vector. They also proposed the Hybrid Gaussian Laplacian Mixture Model (HGLMM) that mixes GMM and LMM in each dimension. These methods were shown to outperform RNN-based methods for image-sentence retrieval tasks. This feature vector is used for more recent work such as Deep Structure-Preserving Embedding [39]. For coding a movie feature, Jain *et al.* [40] applied average pooling to the output of the fully connected layer per each frame to obtain a global feature. Xu *et al.* [41] used the output of the pool5 layer of the VGG Network and applied VLAD [42]. This method is named the Latent Concept Descriptor (LCD). Wang *et al.* [43] proposed a method named the trajectory-pooled deep-convolutional descriptor (TDD) that used the average of the output of the pool5 layer around each trajectory as a local descriptor. Gao *et al.* [44] proposed Compact Bilinear Pooling (CBP) that encodes covariance information effectively by randomization.

Also, there exists pooling layer that considers spatial information. He *et al.* [45] proposed to apply spatial pyramid pooling on the last convolutional layer to encode spatial information of the output of convolutional layer.

Thus, roughly CNN also follows the framework consisting of local feature extraction, feature encoding, and feature pooling.

There exist works that try to analyze the generalization performance of Deep Neural Networks. Neyshabur *et al.* [46] defines a norm based on the model parameters and proposed amethod to compute a Rademacher Complexity of the space with the norm. They also propose an optimization method based on the norm called Path-SGD. Sokolic *et al.* [47] proposed a method to upper bound the margin of DNN between the data point and classification margin with the function of model parameters, and then use the upper bound as a regularizer. Keskar *et al.* [48] argue that the trained model is robust to perturbation and show good generalization performance when the sharpness of the model is small. However, Zhang *et al.* [49] argues that existing generalization analysis cannot explain the performance of DNN because DNN completely overfits the training image with the random label. Also, there exists adversarial example [50] that are manually designed to cause misclassification to DNN. Thus, the analysis of DNN is not well enough for the application.

Chapter 3

Feature extraction based on empirical orthogonal decomposition

In this section, we describe the detail of the framework.

3.1 Recognition based on kernel function

As a classification method that uses prior knowledge, there exist two approaches. First is to construct a feature vector directly from the prior knowledge. Second is to construct a kernel function that represents the similarity of two input data considering the prior knowledge. These are equivalent from the view of representation ability, but often the approach that uses kernel method is fundamental for feature extraction. For example, Murata *et al.* [51] showed that three-layer neural network is the approximation of kernel function. Training with Fisher Vector can be regarded as multiple kernel learning. We can incorporate invariance information effectively using characteristic shift-invariant kernel [52]. Thus, we construct our framework based on kernel method.

3.2 Existing Kernel Approximation Method

However, the complexity of kernel methods grows quadratically or cubically with the amount of the training data, which makes it difficult to scale directly for large-scale datasets. A method that approximates the kernel function using the inner product of the nonlinear feature functions, which map data into a relatively low-dimensional feature space, is useful because it is compatible with fast linear classifiers.

There are two major methods for approximating the kernel function: the Nyström method [53, 54] and the random features method [55].

The Nyström method approximates the true Gram matrix using kernel similarity to randomly sample data from training examples. Let $\{x_1, x_2, \cdots, x_D\}$ denote the subset of samples and let $K_D = U\Lambda U^t$ denote the eigen-decomposition of the Gram matrix generated by the subset of samples, then the Nyström method maps input x in the following way:

$$\Lambda^{-1/2}U^t(k(x, x_1), k(x, x_2), \cdots, k(x, x_D))^t \tag{3.1}$$

To analyze Nyström methods, the bound of the spectral norm of the approximation error is usually calculated. In [53], the authors showed that the approximation error is $O(D^{-1/2})$. Because Bartlett *et al.* [56] showed that the generalization error of the kernel method is $O(N^{-1/2})$, where N is the size of training data, the required number of samples D should be $O(N)$ to achieve a small approximation error. According to the analysis in [57], the number of samples D is reduced to $O(N^{-1/2})$ by assuming that there is a large gap between the eigenvalues. Kumar *et al.* [58] provided a detailed comparison of various fixed and adaptive sampling techniques. However, an $O(D^3)$ calculation of $K^{-1/2}$ of the sample Gram matrix is required, and $O(D^2)$ post-processing for each datum is required, which is time-consuming when D is large.

The random features method approximates the kernel function using an inner product of randomly sampled feature functions.

Definition 3.2.1. *For kernel k on domain X, if there are functions ψ_ω parameterized by ω and parameter distribution $p(\omega)$ that fulfill the equality*

$$k(x, y) = E_\omega[\psi_\omega(x)^*\psi_\omega(y)] = \int d\omega p(\omega)\psi_\omega(x)^*\psi_\omega(y), \tag{3.2}$$

then a random feature is a method that samples D ω_ds i.i.d from $p(\omega)$ and maps $x \to \frac{1}{\sqrt{D}}(\psi_{\omega_1}(x), ..., \psi_{\omega_D}(x))$.

If ψ_ω is uniformly bounded, then we can show that we can approximate the original kernel with high probability using a sufficiently large dimension D by applying Hoeffding's inequality.

Rahimi and Recht [55] proposed a random feature using trigonometric functions for a shift-invariant kernel in Euclidean space $\mathbb{R}^d$. A shift-invariant kernel is a kernel that can be calculated using only the difference of two inputs such as $k(x, y) = \phi(x - y)$. Rahimi and Recht [55] constructed a random Fourier feature using Bochner's theorem, which connects shift-invariant kernels with probability distributions in Fourier space.

Theorem 3.2.1 (Bochner [59]). *For ϕ corresponding to a shift-invariant kernel, there is a probability $p(\omega)$ on $\mathbb{R}^d$ that*

$$k(x, y) = \phi(x - y) = \int d\omega p(\omega) e^{i\omega(x-y)} \tag{3.3}$$

holds.

According to Bocher's theorem, the shift-invariant kernel is a Fourier transform of some distribution. By sampling ω_d from this distribution $p(\omega)$, the mapping

$$x \to \frac{1}{\sqrt{D}} \left(e^{i\omega_1 x}, ..., e^{i\omega_D x} \right) \tag{3.4}$$

approximates the original kernel. In addition, the method that uniformly samples b_d from $[0, 2\pi]$ and maps

$$x \to \frac{1}{\sqrt{D}} \left(\sqrt{2}\cos\left(\omega_1 x + b_1\right), ..., \sqrt{2}\cos\left(\omega_D x + b_D\right) \right) \tag{3.5}$$

also becomes a random feature, which is used to make the feature value real. We use this form for the experiments. This feature function is uniformly bounded, so it fulfills the condition for Hoeffding's inequality.

The framework of Equation 3.4 is simple and versatile, but because the feature is random, the feature tends to be verbose. To solve this problem, Hamid *et al.* [60] proposed a method that oversamples ω and projects it in a lower-dimensional space, where the projection matrix is also randomly sampled. Yang *et al.* [61] proposed a method to use quasi-Monte Carlo instead of i.i.d random variables. To decrease the complexity, Le *et al.* [62] proposed a method to approximate a feature function with complexity $O(D \log d)$.

As an application for data mining, Lopez-Paz *et al.* [63] combined a random feature with PCA and CCA and showed that they approximate kernel PCA and kernel CCA. Lu *et al.* [64] reported performance comparable with deep learning by combining multiple kernel learning and the composition of kernels. Dai *et al.* [65] and Xie *et al.* [66] proposed a method that combined a random feature with stochastic gradient descent to construct an online learning method.

Yang *et al.* [67] proposed the random Laplace feature for the kernel $k(x, y) = \phi(x + y)$ on a semi-group $(\mathbb{R}^m > 0, +)$ and applied it to kernels on histogram data, such as Bag of Visual Words.

However, Rahimi and Recht [68] reported the generalization performance using a random feature is $O(N^{-1/2} + D^{-1/2})$. Thus we need to sample $O(N)$ random features to gain sufficient generalization performance, so the complexity does not decrease.

In this paper, we propose a method to closely approximate the kernel function via empirical orthogonal decomposition without post-processing for the features. In the proposed method, the kernel function is decomposed using the probability distribution estimated from training data, which enables it to have a high approximation ability. As the proposed method directly approximates the kernel function, post-processing for the features becomes unnecessary. We show that the spectral norm of the approximation error of the Gram matrix is bounded using eigenvalues and the distance between the true and approximate distributions. We also present the calculation method of the proposed kernel approximation using the Gaussian kernel.

3.3 Kernel Approximation based on Empirical Orthogonal Decomposition

The Nyström method uses information from input data as the feature function and provides good generalization performance, but requires post-processing of the feature. The random features method approximates the kernel function and does not require post-processing of the feature. The information required to obtain the feature function $p(\omega), e^{i\omega x}$ requires only the kernel function, and hence, it provides lower generalization performance. In this section, we propose a method that approximates the kernel function using information from input data to overcome the limitations of both methods.

First, from Mercer's theorem [69], we can represent the kernel k on domain X with finite measure μ as

$$k(x, y) = \sum_{i=0}^{\infty} \lambda_i \psi_i(x) \psi_i^*(y), \tag{3.6}$$

using eigenvalues λ_i and the normalized eigenfunctions ψ_i of the positive definite operator T_k on $L_2(X)$ such that

$$(T_k \psi)(\cdot) = \int_X k(\cdot, x) \psi(x) d\mu(x). \tag{3.7}$$

We can regard the Nyström method as approximating this distribution μ using the histogram of randomly sampled input data. Additionally, using a shift-invariant kernel and a Lebesgue measure, the feature corresponds to a random Fourier feature. Because the Lebesgue measure is not finite, the decomposition is an integral instead of a discrete sum; therefore we need to randomly sample the feature function.

In this paper, we propose an intermediate approach that approximates the input distribution μ using a distribution for which its eigenfunction decomposition can be solved, and use the eigenfunctions as feature functions. The algorithm is as follows:

1. Estimate the parameter of some distribution $p(x;\theta)$ using training data.
2. Solve the eigenfunction decomposition $(T_k\psi)(\cdot) = \int_X k(\cdot,x)\psi(x)p(x;\theta)dx$ using the estimated distribution $p(x;\theta)$.
3. Use $\lambda_i^{1/2}\psi_i$ corresponding to the D largest eigenvalues as feature functions.

3.3.1 Analysis of the approximation error

In this section, we evaluate the expectation and high-probability bound for the spectral norm of the approximation error corresponding to the Gram matrix, which is important for the efficiency of the kernel approximation method. For example, Cortes *et al.* [70] proposed a method to bound the estimation error of test data for kernel ridge regression and kernel support vector machine using the spectral norm error of Gram matrix.

We denote the Gram matrix using N data $\{x_1, x_2, ..., x_N\}$ by K_{true}, the Gram matrix using the proposed approximation method by K_{app}, and assume that the kernel function is upper bounded by some κ such that $k(x,x) \leq \kappa$ for $\forall x \in X$. The following holds when we use the D-dimensional feature:

Theorem 3.3.1. *Given the true probability density $p_{\text{true}}(x)$ and the approximated density as p_{app}, then*

$$\begin{aligned} &E_{x_i\sim p_{\text{true}}}[\|K_{\text{true}} - K_{\text{app}}\|_2] \\ &\leq N\left(\sum_{n=D}^{\infty}\lambda_n + \kappa\int_X |p_{\text{true}}(x) - p_{\text{app}}(x)|dx\right), \end{aligned} \tag{3.8}$$

holds. Additionally, for a probability larger than $1-\delta$,

$$\begin{aligned} &\|K_{\text{true}} - K_{\text{app}}\|_2 \\ \leq\ & N\left(\sum_{n=D}^{\infty}\lambda_n + \kappa\int_X |p_{\text{true}}(x) - p_{\text{app}}(x)|dx\right) \\ +\ & \sqrt{\frac{N\kappa^2}{2}\log\frac{1}{\delta}}, \end{aligned} \tag{3.9}$$

holds.

Proof. It holds that $K_{\text{diff}} = K_{\text{true}} - K_{\text{app}}$ is also a Gram matrix using kernel $k(x,y) = \sum_{i=D}^{\infty} \lambda_i \psi_i(x)\psi_i^*(y)$, so K_{diff} is a symmetric positive semidefinite matrix. Note that this does not hold for a random Fourier feature, which uses an integral instead of discrete sum and does not use its eigenvalues directly. Hence,

$$\begin{aligned} \|K_{\text{true}} - K_{\text{app}}\|_2 &= \lambda_{\max}(\|K_{\text{diff}}\|) \\ &\leq \text{trace}\|K_{\text{diff}}\| = \sum_{i=1}^{N} k_{\text{diff}}(x_i, x_i), \end{aligned} \tag{3.10}$$

holds. Hence, $E_{x_i \sim p_{\text{true}}}[\|K_{\text{true}} - K_{\text{app}}\|_2] \leq N E_{x \sim p_{\text{true}}}[k_{\text{diff}}(x,x)]$. Moreover,

$$\begin{aligned} & E_{x \sim p_{\text{true}}}[k_{\text{diff}}(x,x)] \\ = & \int_X (p_{\text{true}}(x) - p_{\text{app}}(x)) k_{\text{diff}}(x,x) dx \\ + & E_{x \sim p_{\text{app}}}[k_{\text{diff}}(x,x)], \end{aligned} \tag{3.11}$$

The former is bounded by $\int_X |(p_{\text{true}}(x) - p_{\text{app}}(x))||k_{\text{diff}}(x,x)|dx \leq \kappa \int_X |(p_{\text{true}}(x) - p_{\text{app}}(x))|dx$, and using the property of eigenfunction decomposition,

$$E_{x \sim p_{\text{app}}}[\psi_i(x)\psi_i^*(y)] = 1, \tag{3.12}$$

the latter becomes $\sum_{n=D}^{\infty} \lambda_n$. Thus, the inequality for the expectation Equation 3.8 holds.

Because $0 \leq k_{\text{diff}}(x,x) \leq \kappa$, applying Hoeffding's inequality to Equation 3.11, we obtain

$$P(\sum_{i=1}^{N} k_{\text{diff}}(x_i, x_i) - N E_{x \sim p_{\text{true}}}[k_{\text{diffx}}] \geq Nt) \leq \exp\left(-\frac{2Nt^2}{\kappa^2}\right). \tag{3.13}$$

Thus, a high probability bound Equation 3.9 is obtained. □

From the discussion in the work by Yang *et al.* [57], we require that $\|K_{\text{true}} - K_{\text{app}}\|_2 = O(N^{1/2})$ holds for good generalization performance; that is, we require $\sum_{n=D}^{\infty} \lambda_n, \int_X |p_{\text{true}}(x) - p_{\text{app}}(x)|dx$ to be $O(N^{-1/2})$ for sufficient performance.

3.3.2 Linear kernel with Gaussian Distribution

First, we apply our method to the linear kernel using Gaussian distribution. Since the feature vector corresponding to the linear kernel is already known, we use the technique in this section for the dimensionality reduction.

When we assume the d-dimensional space and distribution $p = \mathcal{N}(0, \Sigma)$, the equation of the eigenfunction will be

$$\lambda\psi(y) = \int_{\mathbb{R}^d} y^t x \psi(x) p(x) dx \tag{3.14}$$

When Σ is diagonalized by $\Sigma = U^t S U$, we denote $v = Uy$, $w = Ux$, $g = \psi \cdot U^t$, this can be written as

$$\lambda g(v) = \int_{\mathbb{R}^d} v^t w g(w) \prod_{j=1}^{d} \frac{1}{\sqrt{2s_j}} e^{\frac{-w_j^2}{2s_j}} dx. \tag{3.15}$$

Since $E[w_i w_j] = \delta_{i,j} s_j$, $g(x) = x_j/\sqrt{s_j}$ and $\lambda = s_j$ satisfies the equation. Thus, the derived the feature is top k eigenvector of covariance, which is principal component analysis. Thus, we can derive principal component analysis in our framework.

3.3.3 Gaussian kernel with Gaussian Distribution

As an example of more complex analytic solution for eigenfunction decomposition, we consider the Gaussian kernel and a Gaussian distribution as an approximate distribution. If the dimension $d = 1$, setting $k(x, y) = \exp(-b(x-y)^2), p(x) = N(0, \frac{1}{4a})$, and using $c = \sqrt{a^2 + 2ab}$, $A = a + b + c$, and $B = b/A$, the eigensystem is as presented by Zhu *et al.* [71]:

$$\lambda_n = \sqrt{\frac{2a}{A}} B^n \tag{3.16}$$

$$\psi_n(x) = \exp(-(c-a)x^2) \mathrm{H}_n(\sqrt{2c}x), \tag{3.17}$$

where H_n denotes a Hermite polynomial of integer order n and is defined as $\mathrm{H}_n(x) = (-1)^n \exp(x^2) \frac{d^n}{dx^n} \exp(-x^2)$. The feature function is localized and better reflects the properties of the Gaussian kernel, for which the similarity diminishes if the data are distant, than a random Fourier feature, which does not attenuate. Additionally, as n increases higher resolution information can be obtained. There are studies that use this solution for kernel learning [71, 72], but to the best of our knowledge, the present research is the first to learn the distribution from data and apply it to unsupervised feature learning.

When the input dimension d is larger than 1 and the covariance matrix is diagonal, the eigensystem is a product of the above Hermite solution. Even if the covariance is non-diagonal, the solution reduces to the case in which covariance is diagonal by rotating the axis. To evaluate the approximation error, we denote the feature dimension by D and simplify the calculation by assuming a is the

same for each dimension. This bounds the general case. Thus, $\sum_{n=D}^{\infty} \lambda_n = (\frac{2a}{A})^{d/2}((\frac{1}{1-B})^d - (\frac{1-B^{D/d}}{1-B})^d) \simeq (\frac{2a}{A})^{d/2}(\frac{1}{1-B})^d dB^{D/d} = dB^{D/d}$ and we can see that the error decreases exponentially with D.

3.3.4 Gaussian kernel with Gaussian mixture Distribution

To approximate a more complex distribution, we consider a Gaussian mixture. The analytic solution using this Gaussian mixture is not known, so we consider approximating it using the result for a Gaussian distribution. We denote the number of components by K and set $p(x) = \sum_{k=1}^{K} \gamma_k \mathcal{N}(\mu_k, \Sigma_k)$. Let (λ_n^k, ψ_n^k) be the eigensystem for $\mathcal{N}(\mu_k, \Sigma_k)$. Because the kernel can be decomposed as

$$k(x,y) = \sum_{k=1}^{K} \omega_k k(x,y) = \sum_{k=1}^{K} \sum_{n=0}^{\infty} \omega_k \lambda_n^k \psi_n^k(x) \psi_n^{k*}(y), \tag{3.18}$$

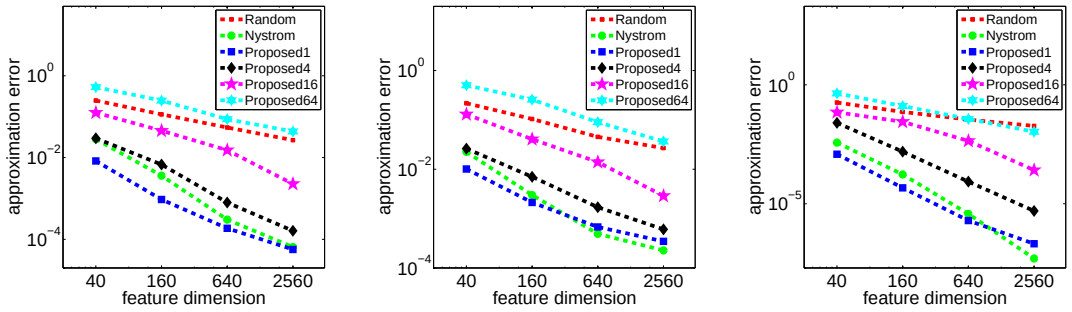
we consider using $(\omega_k \lambda_n^k)^{1/2} \psi_n^k(x)$ for larger $\omega_k \lambda_n^k$ as feature functions.

Next, we analyze the performance of this method. As the feature function is not the true eigenfunction, the above discussion does not hold in its current form. However, we can see that K_{diff} is a symmetric positive semidefinite matrix, and we have only to bound $E_{x\sim p}[k_{\text{diff}}(x,x)]$. To simplify this, we assume that $\omega_k = \frac{1}{K}$ and a, b are the same for each distribution and dimension, and each μ_k is well separated such that $E_{x\sim\mathcal{N}(\mu_k,\Sigma_k)}[\psi^{k'}(x)\psi^{k*}(y)] < R$ for some R. In this case, using the result from the previous section, $E_{x\sim p}[k_{\text{diff}}(x,x)] < (1+(k-1)R)dB^{\frac{D}{dk}}$ is obtained. Thus, we infer that this method has an exponential gain in performance with D.

3.3.5 Relation to kernel PCA

When we apply kernel methods, we often use PCA in the projected high-dimensional space to obtain uncorrelated useful features. The proposed methods, which use eigenfunctions, are automatically projected in the feature spaces, so correlations between features are small. Thus, it is expected that our methods demonstrate a similar effect to PCA.

When we use the proposed method with the Gaussian distribution, we rotate the input space so that each input element is uncorrelated. Then, the axis with a large variance has large B, so even high-order eigenfunctions with a high resolution are used. Conversely, the axis with a small variance makes a small contribution to the feature vector. In particular, the axis on which only the 0-th order eigenfunction is used only contributes to the norm of the feature. Thus we can ignore it when, for example, the features are normalized. Therefore, we



(a) Gaussian (b) Laplace (c) Uniform

Figure 3.1: Comparison of the approximation error for the Gram matrix on synthesized data sampled from (a) Gaussian distribution, (b) Laplace distribution, and (c) Uniform distribution.

can say that the proposed method also applies dimensionality reduction in the input space. When the input space is d dimensional, the number of substantial dimensions is d', and we extract the D dimensional feature, the complexity of the proposed method is $O(dd')$ for rotation plus an $O(d'D)$ calculation of Hermite polynomials. We replace d' with d when we use the full input vector. In all cases, it is much smaller than the Nyström method, which requires an $O(D)$ calculation of kernel values between d dimension vectors plus an $O(D^2)$ whitening step when D is large.

3.4 Experiment

To test the efficiency of our methods, we compared the approximation error of the Gram matrices, the classification accuracy, and performance for unsupervised feature learning.

3.4.1 Approximation error of the Gram matrices

First, we evaluated the approximation performance using synthesized data. We set the dimensionality of the input data to $d = 10$, number of samples to $N = 5000$, and kernel parameter to $b = \frac{1}{2d}$, and compared the Nyström method (Nyström), random Fourier feature (Random), and proposed methods (Proposed) using data sampled from the Gaussian distribution with mean equal to 0 and a covariance identity matrix, from the Laplace distribution with location parameter equal to 0 and scale parameter equal to 1 as a super-Gaussian distribution,

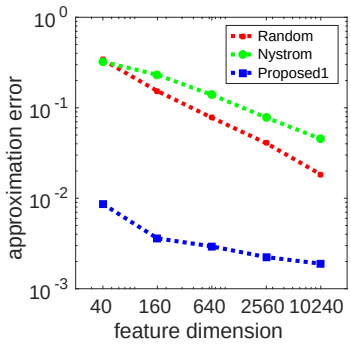


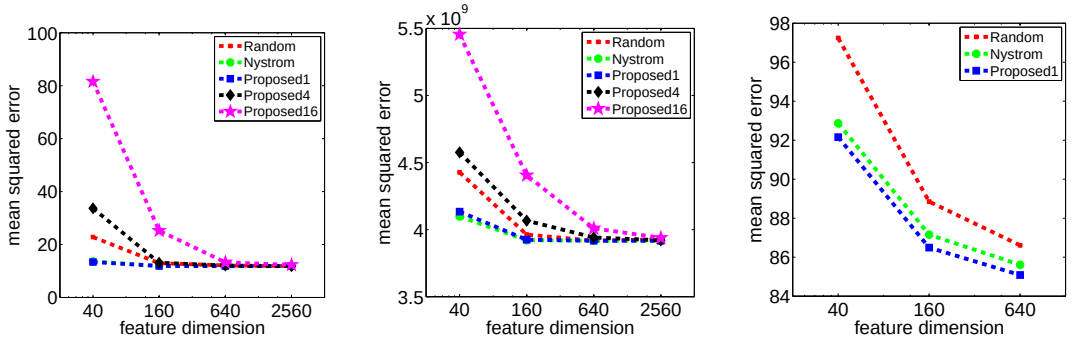
Figure 3.2: Comparison of the approximation error for the Gram matrix on GoogLeNet features extracted from the ILSVRC2015 dataset.

TASK	DATA	# TRAIN	# TEST	# Attr.
Reg.	CPUSMALL	7392	800	12
Reg.	CADATA	18640	2000	8
Reg.	YEARMSD	463715	51630	90
Class.	ADULT	32561	16281	123
Class.	IJCNN1	49990	91701	22
Class.	COVTYPE	522910	58102	54

Table 3.1: Statistics for the datasets.

and from a uniform distribution from [-1,1] as a sub-Gaussian distribution. For each method, we evaluated the normalized spectral norm of the error matrix $\frac{\|K_{\text{true}}-K_{\text{app}}\|_2}{\|K_{\text{true}}\|_2}$. We set the feature dimension $D = 40, 160, 640, 2560$, and the number of mixture components to $1, 4, 16, 64$. Note that if the number of mixture components is 1, the situation is equivalent to the Gaussian case. To estimate the parameter of the data distribution, we used another set of N data sampled from the same distribution. For preprocessing, we rotated the data so that the estimated covariance was diagonal and used the diagonal Gaussian mixture. This rotation did not change the kernel value. We performed the experiment 10 times for each setting and calculated the mean value.

Figure 3.1 shows the results. The number in the "Proposed" label indicates the number of mixture components. The figure shows that for each distribution, the proposed method that assumed a Gaussian distribution yielded the best approximation performance if the dimension was low. Even if the dimension was high, the proposed method yielded a performance comparable with the Nyström method. Better performance for low feature dimensions occurred because the



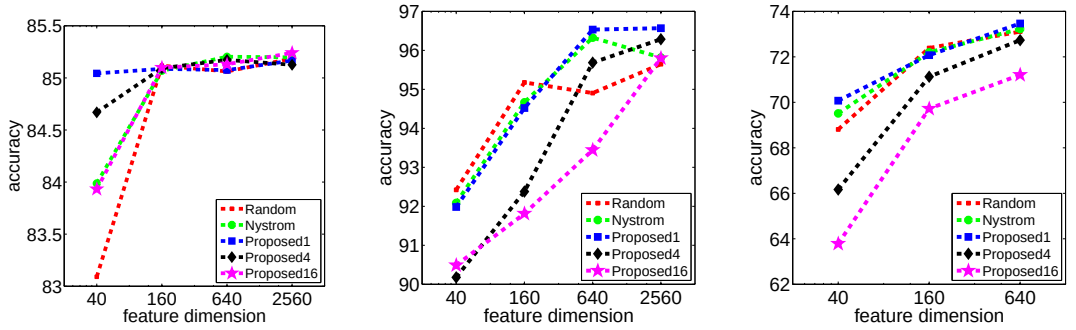
(a) CPUSMALL (b) CADATA (c) YEARMSD

Figure 3.3: Comparison of the mean squared error for the datasets (a) CPUSMALL, (b) CADATA, and (c) YEARMSD.

rough estimation of the proposed method approximated the true distribution better than the estimation using a small sample histogram from the Nyström method. The random Fourier feature demonstrated similar performance for each distribution, which agrees with the fact that random features method does not use distribution information. By contrast, when we assumed a Gaussian mixture, the performance was lower and the performance gain was also smaller than the case that assumed a Gaussian distribution in each case. The uncertainty associated with the parameter estimation and the decrease of decay speed of eigenvalues influenced the performance more than the approximation accuracy of the true distribution.

We then compared the performance of the proposed kernel approximation method assuming Gaussian distribution using the ILSVRC2015 classification dataset. The dataset contained approximately 1,200,000 images and we used the output of the global average pooling layer of GoogLeNet as input features, which are 1,024 dimensional per image. We used 100,000 samples for model inference and evaluated the normalized spectral norm of the error matrix $\frac{\|K_{\text{true}} - K_{\text{app}}\|_2}{\|K_{\text{true}}\|_2}$ using 5,000 randomly chosen samples. We plotted the mean of five trials. We set the feature dimension $D = 40, 160, 640, 2560, 10240$

Figure 3.2 shows that the proposed approximation method demonstrates better performance even for a real image dataset.



(a) ADULT (b) IJCNN1 (c) COVTYPE

Figure 3.4: Comparison of the classification accuracy for the datasets (a) ADULT, (b) IJCNN1, and (c) COVTYPE.

3.4.2 Classification Accuracy

We compared regression performance and classification accuracy using real data. We used the data from the LIBSVM site [1]. We scaled each element of the input data to [0,1] for the classification task and [-1,1] for the regression task. Table 3.1 shows the statistics for the datasets. We set the kernel parameter $b = \frac{1}{2d}$ and used LIBLINEAR[2] with $C = 100$ to compare the classification accuracy of the test data for classification tasks and ridge regression $\min_w \|\Psi^t w - t\|_2^2 + \lambda\|w\|_2^2$ with $\lambda = 0.01$ to compare the mean squared error of the test data $\frac{1}{n}\sum_{i=1}^{n} \|t_i - w^t\psi(x_i)\|_2^2$ for regression tasks. We set the feature dimension $D = 40, 160, 640, 2560$ for CPUSMALL, CADATA, ADULT and IJCNN1, and $40, 160, 640$ for the larger datasets YEARMSD and COVTYPE. We set the number of mixture components to $1, 4, 16$. For parameter estimation, we sampled 1000 data for CPUSMALL, CADATA, and 10000 data for ADULT, IJCNN1, YEARMSD and COVTYPE. For each setting, we conducted 10 experiments and calculated the mean.

Figure 3.3 and Figure 3.4 show the results. The result for Nyström is overlapped by that for Proposed1 in CPUSMALL. We omitted the results for Proposed4 and Proposed16 in YEARMSD because, in some cases, they had a mean squared error that was too large. The figures show that the proposed method assuming a Gaussian distribution demonstrated better performance than the random Fourier feature, especially when the dimension was small. Additionally, they demonstrated comparable performance with the Nyström method for each dimension. Because the Nyström method requires $O(D^2)$ post-processing for each feature, our method is more efficient considering the computation complexity. Gen-

[1] http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/
[2] https://www.csie.ntu.edu.tw/ cjlin/liblinear/

Method	1,024	2,048	4,096	8,192	16,384
Random	4.1e-2	8.0e-2	2.8e-1	1.07	4.36
Nyström	3.8e-2	1.0e-1	3.9e-1	2.52	4.9e1
Proposed	3.9e-2	9.0e-2	2.3e-1	1.2e-1	5.6

Table 3.2: Computation time (second) on synthesized data.

erally, the proposed method assuming a Gaussian mixture demonstrated poorer performance than the other methods. However, the differences between each performance were small when the dimension was high, and Proposed16 demonstrated the best performance in ADULT. The proposed methods assuming the mixture model work well if the feature dimension is not small and the model fits the data distribution. Generally, when assuming a Gaussian distribution, the data distribution was sufficiently approximated and our method demonstrated comparable performance with the Nystöm method.

3.4.3 Computation Time

We also compared the computation time using synthesized data used in Section 3.4.1. We varied the output feature dimension D to 40, 160, 640, 2560 and 10240. We compared the computation time required to encode 5,000 input vector using an Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz. We implemented each method using Matlab with the "-singleCompThread" option. Table 3.2 shows that our method shows performance similar to Random, while Nyströ method requires large computation time as D grows. Thus, our method is as efficient as random features method while as effective as Nyström method.

3.5 Application to Image Recognition

In this chapter, we propose an overview of the method to apply our framework to the image recognition.

As for local feature extraction, it is important to represent the relationships between the positions of pixels in the image patches. Also, since the information of local region is not linear to the values of pixels, we need to express this nonlinear similarity between pixels. To this end, we apply our method to the convolutional kernel between image patches proposed in Convolutional Kernel Networks [10] in Chapter 4. In Convolutional Kernel Networks, we transform both feature values of the pixels and feature positions so that they approximate the Gaussian kernel between feature values and between feature positions respec-

tively. Then, we use the summation of tensor products of transformed feature values and transformed feature positions as a feature of the patch of the next layer. We apply the proposed approximation method to construct the transformation of the feature values. Also, we can apply our non-linear embedding to the global feature after feature pooling.

The same strategy works well for constructing one global feature from the bag of local features by associating the feature values with pixel values and feature positions in the image with pixel positions in the patch. The difference is that we need to extract lower-order information from feature values because the dimension of the input feature is large and the interpretability of the encoding method is important. Conversely, as for transforming the feature position, we need to extract more complex position information effectively because the change of position is larger in the image than in the patch.

As a principle for constructing each transformation, we first construct large-dimensional space that perserve information of input vectors, and then extract low-dimensional low-frequency parts with respect to the geometry using the prior knowledge.

For feature encoding that transforms the value of the feature, we first map second order information into large-dimensional covariance space and then reduce the dimension using the prior knowledge of low-rankness and the idea of matrix manifold in Chapter 5.

For feature pooling that transform the position part of local features, we first map the feature positions into the large-dimensional function space from image space to feature space and then apply the linear dimensionality reduction using the prior knowledge that the background function is not too complex in Chapter 6.

Chapter 4

Application to Local Feature Extraction

Kernel approximations are used to construct a hierarchical nonlinear local feature by iteratively building kernels between image patches [10, 73, 74, 75, 76]. Thus, we can apply the framework proposed in the previous chapter to construct the local image feature.

We now combine our approximation method with convolutional kernel networks (CKN) architecture [10] and propose a novel method for unsupervised feature learning. The CKN hierarchically defines the kernel between image patches as the summation of kernels between 2-d positions of points in the patches multiplied by the kernels between feature vectors of points to both consider the nonlinear relationships of pixel values and the position information of pixels in the patches. The kernel value between patches Ω, Ω' can be represented as follows:

$$K(\Omega, \Omega') = \sum_{p \in \Omega} \sum_{p' \in \Omega'} \|\phi(p)\| \|\phi'(p')\| e^{-\frac{1}{2\beta^2}\|\delta p\|^2} e^{-\frac{1}{2\sigma^2}\|\delta \phi\|^2}, \tag{4.1}$$

where p and p' are positions of the points, $\phi(p)$ and $\phi'(p')$ denote feature vectors of the points. We use RGB value or the gradient of RGB that means edge information as the input of the first layer. δp and $\delta \phi$ denote $p - p'$ and $\phi(p) - \phi'(p')$ respectively. When we approximate the kernel between positions as $\xi_{\text{pos}}(p)^T \xi_{\text{pos}}(p')$ and the kernel between feature vectors as $\xi_{\text{feat}}(\phi(p)^T \xi_{\text{feat}}(\phi'(p'))$, the convolutional kernel is approximated as the linear inner product of

$$\sum_{p \in \Omega} \|\phi(p)\| \xi_{\text{pos}}(p) \otimes \xi_{\text{feat}}(\phi(p)), \tag{4.2}$$

where $\otimes$ denotes the Kronecker product. CKN hierarchically applies this mapping and uses the feature vector of the final layer for recognition.

The original CKN uses the approximated feature for the kernel between positions as $\xi_{\text{pos}}(p) = e^{\frac{1}{\beta^2}\|p\|^2}$, and uses the approximated feature for the kernel between features in the form of $\eta_d^{1/2} e^{\frac{1}{\sigma^2}\|\phi(p)-w_d\|^2}$, and then learns η_d, w_d so that the reconstruction error of kernel values

$$\sum_{i=1}^{n}\left(e^{-\frac{1}{2\sigma^2}\|\delta\phi\|^2}-\sum_{d=1}^{D}\eta_d e^{\frac{1}{\sigma^2}\|\phi(p_i)-w_d\|^2}e^{\frac{1}{\sigma^2}\|\phi(p_i')-w_d\|^2}\right)^2, \tag{4.3}$$

is minimized, where $(p_i, p_i')_{i=1}^n$ are patch pairs sampled from training data and D is the dimension of the feature.

Instead of learning feature with a gradient descent, we can use other kernel approximation methods. We propose using eigenfunctions with distribution learned from p_i, p_i' as an approximation function. The proposed method does not experience a long optimization time and local minima.

4.1 Experiments

Next, we used the random features method, Nyström method, and proposed method with the Gaussian distribution, which demonstrated good performance in previous experiments as kernel approximation methods for CKN architecture, and compared the accuracy with the original CKN.

We used MNIST [28], CIFAR-10 [29], CIFAR-100 [29], and SVHN [77] as datasets and adopted similar network architectures to those of Mairal *et al.* [10]. We used 300,000 patch pairs for feature learning, LIBLINIEAR as a linear classifier and determined the regularization parameter using 5-fold cross validation from $2^i, i = -15, ..., 15$.

We show the results in Table 4.1. In most cases, the proposed method demonstrated better performance than the original CKN. This is because the proposed method did not experience local minima of the optimization and could use the input information more efficiently. The proposed method did not require a time-consuming optimization phase. Thus, it was a good choice to adopt the proposed kernel approximation method for CKN architecture. Additionally, the proposed method demonstrated better performance than the random features method and comparable or better performance than the Nyström method in most settings, which demonstrated a similar tendency to that of previous experiments. This indicates the effectiveness of hierarchically approximating the kernel, and the proposed method is reasonable and effective.

Additionally, we showed the covariance of the rescaled first 400-dimensional feature in the final layer learned from CIFAR-10 with setting 1 in Figure 4.1. The figure shows that while CKN had relatively large non-diagonal covariance, the

Setting	CKN	Random	Nyström	Proposed
MNIST				
1	99.34	**99.49**	99.38	99.36
2	99.28	99.46	**99.48**	99.43
3	99.42	99.45	99.47	**99.51**
CIFAR-10				
1	74.59	75.93	75.72	**76.00**
2	79.19	80.73	**81.52**	81.27
3	77.41	77.68	**78.57**	78.29
CIFAR-100				
1	43.25	43.64	43.36	**43.66**
2	53.37	**55.20**	54.44	55.01
3	50.41	51.02	50.53	**51.04**
SVHN				
1	91.80	91.54	91.96	**91.98**
2	90.79	90.75	91.18	**91.36**
3	85.52	85.60	85.88	**86.05**

Table 4.1: Classification accuracy for MNIST, CIFAR-10, CIFAR-100, and SVHN.

proposed method demonstrated uniformly small non-diagonal covariance, which agrees with the argument that the proposed method demonstrated a similar effect to PCA.

Additionally, we varied the number of feature maps in the final layer to $200, 400, 600, 800$ and evaluated the performance using CIFAR-10 with setting 1. We show the result in Figure 4.2. As the dimension decreased, the method using random features demonstrated poorer performance than the proposed method and Nyström method, which illustrates the importance of using input information for approximation. Conversely, these three methods demonstrated similar performance when the dimension was 200. This suggests that when the dimension was very small, the input information was not sufficient and we needed to include discriminative information in learning. In any case, the proposed method demonstrated much better performance than the original CKN.

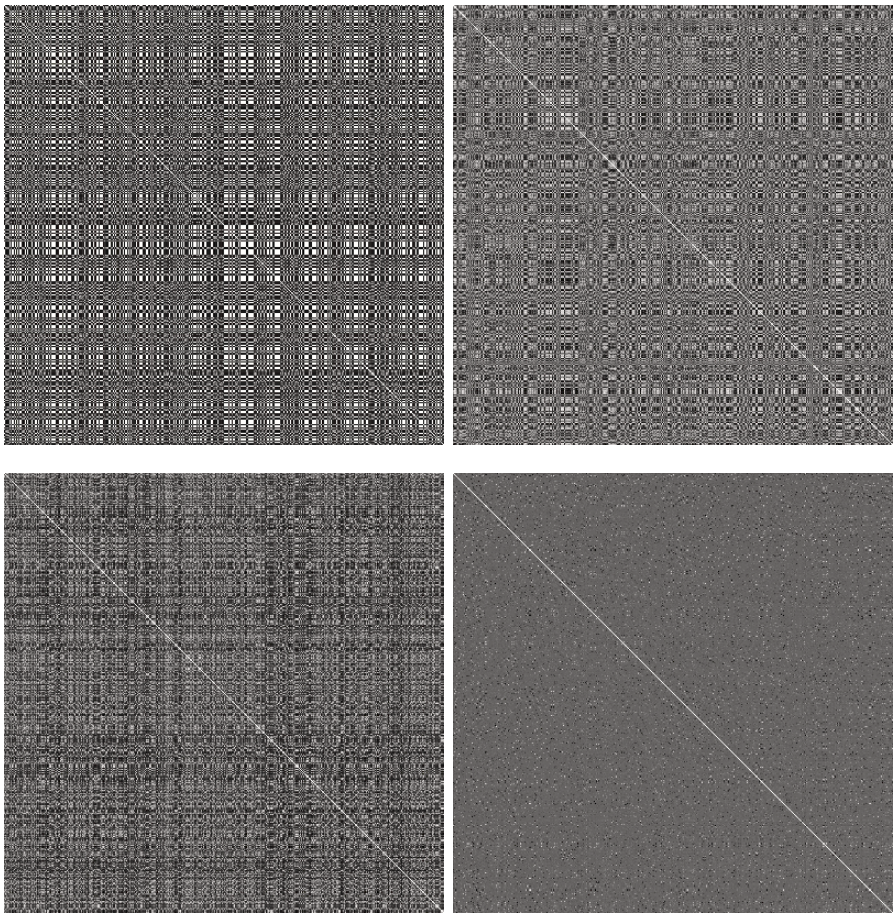


Figure 4.1: Covariance of learned feature for (**top left**) CKN, (**top right**) Random, (**bottom left**) Nyström, and (**bottom right**) Proposed. The proposed method shows less non-diagonal covariance than the other methods.

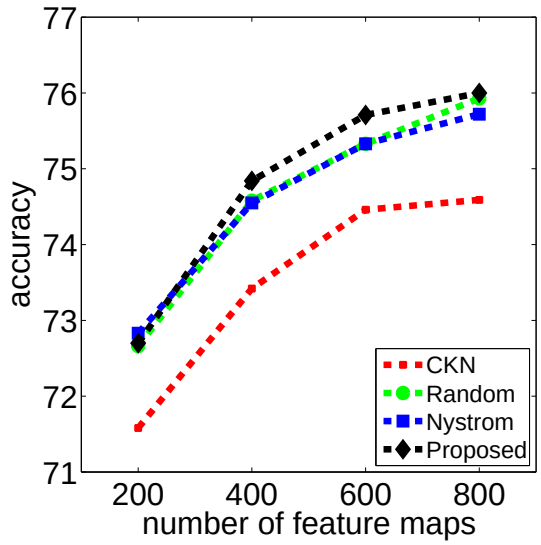


Figure 4.2: Accuracy of CKN with fewer feature maps.

Chapter 5

Application to Feature Encoding

The relevant content in this chapter is scheduled to be published within 5 years.

Chapter 6

Application to Feature Pooling

In this chapter, we apply our method to feature pooling, which summarizes local features in one image into one global feature. To this end, we first introduce a function representation for the local features in one image. Then we apply our kernel approximation method in this function space.

When designing feature pooling, it is important for the global feature to contain rich information and be robust to small image translations. Spatial pyramid matching is the feature-pooling method that is most commonly used. It divides an image into subregions according to various resolutions and uses statistics of local features in each subregion, e.g., the mean and maximum values, as global features. However, there is no theoretical guideline for determining the pooling region. In addition, the global feature value changes discontinuously when the local feature strides over the edge of subregions. Also, the spatial pyramid matching representation is verbose because the different spatial pooling regions overlap. Moreover, we cannot obtain useful features when the resolution is too high because robustness to small translations is lost. Thus, we need a large pyramid size to extract spatial information.

To overcome these problems, we propose a novel feature-pooling method that uses the weighted averages of local features based on the position of the local features in an image. To determine the weights, we propose a novel viewpoint that regards local features in one image as a function. Local features have their own feature values associated with positions in the image. Thus, we can see a set of local features as a function from the image space to the local feature space whose output is the value of the local feature at the input position. With this interpretation, we can regard spatial pyramid matching as a projection into the space of piecewise constant functions based on the standard inner product.

From this viewpoint, we first consider linear kernel with prior isometric Gaussian distribution to derive novel pooling weights as orthogonal projections of this function form into the spaces of low-degree polynomials with certain inner prod-

ucts. We obtain this pooling weight by first calculating orthonormal basis of the spaces of low-degree polynomials with the inner products and then calculating the inner product of the delta functions with the basis. Since the pooling weights are polynomials of the position and thus smooth, the proposed global feature is robust to small image translations. Also, since spatial pooling weights are orthogonal with respect to the given metric, it is expected that we can extract spatial information effectively. The feature dimension and the amount of spatial information can be controlled by the degree of the polynomial space. Then we consider Gaussian kernel with Gaussian distribution learned from training data to get more complex spatial statistics.

As a metric of the polynomial space, we first derive the spatial pooling weights of the spaces of low-degree polynomials with the standard inner product, which consist of the products of Legendre polynomials. To derive the pooling weights more robust to local translations than Legendre polynomials, we then propose a weighted pooling method that considers the function space with weighted inner products, which are more robust to local translations than the standard inner product.

6.1 Existing work on feature pooling

Feature pooling is a method that combines local descriptors in an image into one global feature. The simplest strategy is average pooling, which uses the means of local descriptors as a global feature. Max pooling [78] is a method that is inspired by the human visual cortex and is used for coding methods using histograms such as Bag of Visual Words [13] and Locality-constrained Linear Coding [79]. Max pooling uses element-wise maximum values instead of the average of local descriptors as a global feature and has been shown to be more robust to noise. A theoretical analysis of these pooling methods was conducted in [80]. In [80], the method that uses the L_p norm of each dimension is proposed as a method that bridges between average pooling and max pooling. These pooling methods are compared exhaustively via experiments in [81].

Lazebnik *et al.* [82] highlighted the importance of using spatial information of local features in image recognition. As an approximation for the pyramid match kernel, Lazebnik *et al.* proposed spatial pyramid matching, which divides the input image into subregions with various resolutions and concatenates Bag of Visual Words [13] in each subregion to obtain the global feature. Spatial pyramid matching is also applied to global features with richer information, such as the Fisher vector (FV) [14], the vector of locally aggregated descriptors (VLAD) [42]. Though other methods can be combined with spatial pyramid matching, spatial pyramid matching using average pooling is standard in feature pooling.

Thus, we consider average pooling in the next section. In addition, spatial pyramid matching is combined with convolutional neural networks (CNNs) [31] and demonstrates good performance [45].

As extensions of original spatial pyramid matching, Perronnin *et al.* [36] proposed the non-regular spatial pyramid matching that uses different spatial resolutions for x-axis and y-axis. Shahiduzzoman *et al.* [83] proposed to apply Gaussian blur to the input image before extracting local features. Koniusz & Mikolajczyk [84], Sanchez *et al.* [85] proposed a method that simply concatenates the normalized two-dimensional (2D) position of local features to the feature value and then applies feature coding methods to obtain accuracy comparable to spatial pyramid matching with a smaller global feature dimension. Boureau *et al.* [86] apply pooling based on both image space and local feature space. Krapac *et al.* [87] derived a global feature that models both the local descriptor space and image space using the Gaussian mixture model. Similarly, Cinbis *et al.* [88] assumed a hierarchical probabilistic model that includes the feature position and uses the differential of log-probability with respect to hyper-parameters as the global feature.

Some researchers have considered pooling methods that use the weighted average. In [89], a weight based on saliency is proposed. Generalized Max Pooling [90] calculates the weight using the feature value to suppress the effect of frequent but meaningless local features. Some works [10, 73] adopted Gaussian Weighted average instead of original average pooling. We can regard our method as some extensions of these works because the proposed methods can derive similar weight as the pooling weight that corresponds to 0-th degree polynomial, and also derive the weight with higher order information as higher degree polynomials. Geometric L_p norm Feature Pooling (GLFP) [91] also considers the weighted average with respect to the local feature position. However, while we can apply our method even when the image sizes differ because our method considers the normalized position of the local features, we cannot apply GLFP directly for this situation because GLFP considers the adjacent relation between local descriptors. Also, our method is faster than GLFP because GLFP requires the calculation of cubic order of the number of local descriptors to calculate the weight, while our methods require linear order.

Though our method computes the weight in an unsupervised manner, we can calculate the discriminative weight by combining our method with the methods that learn the weight of spatial pyramid discriminatively [92, 93].

In this paper, we focus on an extension of spatial pyramid matching with average pooling because this method is general and can be easily combined with coding methods.

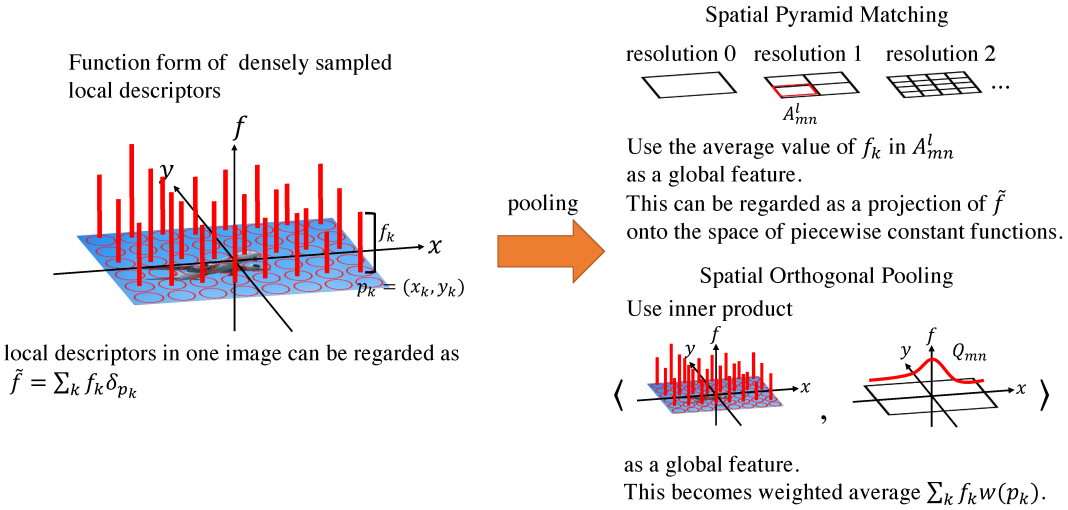


Figure 6.1: Overview of spatial pyramid matching and the proposed pooling method.

6.2 Spatial Pooling as a Projection

In Section 6.2 and 6.3, we propose an interpretation that regards local descriptors in one image as a function and pooling as a projection in the function space. Figure 6.1 shows an overview.

We assume that local features $\{(f_k, p_k)\}_{k=1}^N$ in one image are densely extracted, where N denotes the number of local features from an image, $f_k \in \mathbb{R}^d$ denotes the local features of each point after feature coding, such as the FV, and $p_k = (x_k, y_k) \in (-1, 1)^2$ is the normalized 2D position of each local feature with the image center $(0, 0)$. The goal of feature pooling is to construct one global feature $F \in \mathbb{R}^D$ from $\{(f_k, p_k)\}_{k=1}^N$. Since feature pooling is applied element-wise, we also assume that $d = 1$ for simplicity. In the general case, we concatenate the output of feature pooling for each dimension to obtain the global feature.

Average pooling is a method that simply ignores the feature position and uses the mean as the global feature as follows:

$$F = \frac{1}{N} \sum_{k=1}^{N} f_k. \tag{6.1}$$

Notice from this equation that average pooling completely disregards spatial information, which significantly affects recognition performance.

To include spatial information, spatial pyramid matching divides the image

space using various resolutions and uses the feature mean in each subregion A^l_{mn} as the global feature as follows:

$$F^l_{mn} = \frac{1}{N^l_{mn}} \sum_{p_k \in A^l_{mn}} f_k, \tag{6.2}$$

where N^l_{mn} is the number of local features in A^l_{mn}. We select the image subregion A^l_{mn} such as $\left(\frac{m-1}{l}, \frac{m}{l}\right) \times \left(\frac{n-1}{l}, \frac{n}{l}\right)$, $(-l < m, n \leqq, l)$, where l corresponds to the resolution.

In the following, we propose the interpretation of feature pooling as a projection in the function space to analyze the property of spatial pyramid matching and the proposed spatial weighted pyramid uniformly using the property of the projected function space. Thus, we provide a function representation for both the input local features and the output of feature pooling.

First, as a function representation of local features that includes both feature values and spatial information, we consider a hyper-function in the image space that connects the feature position to the feature value as follows:

$$\tilde{f} = \sum_{k=1}^{N} f_k \delta_{p_k}, \tag{6.3}$$

where δ_p denotes the delta function that satisfies

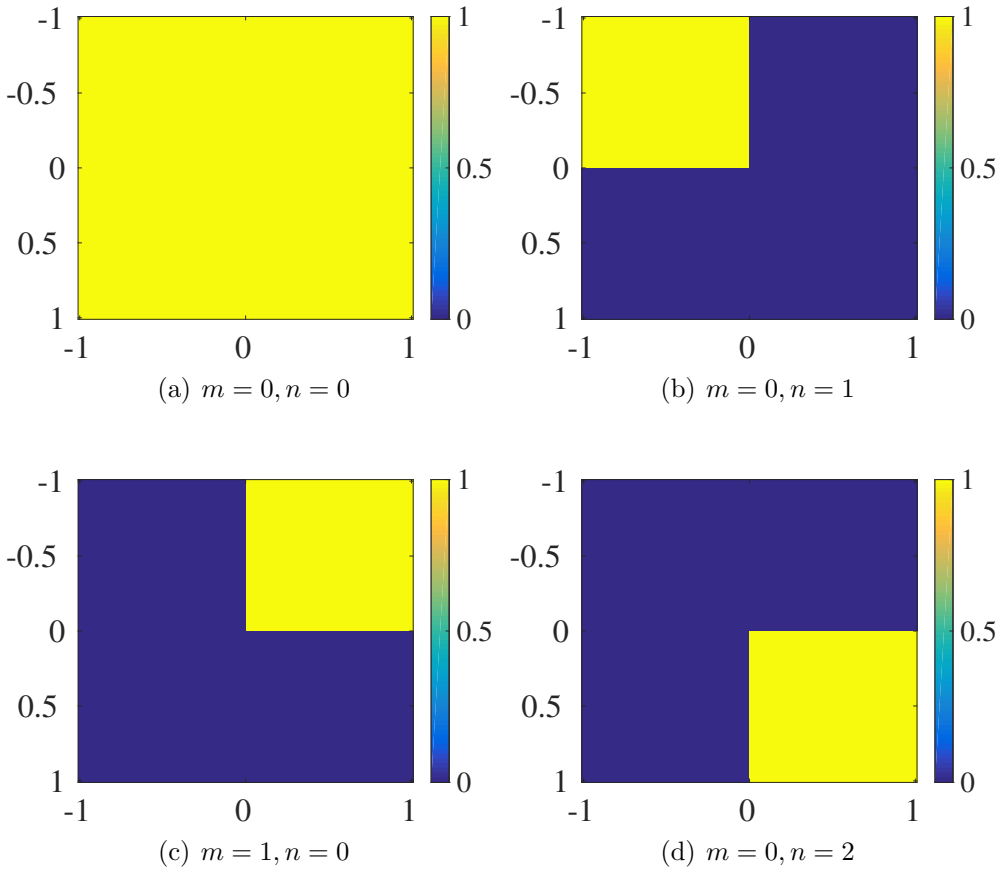
$$\langle \delta_p, g \rangle \equiv \int_{-1}^{1} \int_{-1}^{1} dxdy \delta_p(x, y) g(x, y) = g(p), \tag{6.4}$$

for a function g that is smooth and bounded near p.

Next, we consider a function space that consists of functions that are constant in each A^l_{mn}: $\mathcal{F}^l_{\text{const}} = \{f | f = \sum_{m,n} c^l_{mn} 1_{A^l_{mn}}, c^l_{mn} \in \mathbb{R}\}$, where $1_{A^l_{mn}}$ is a function that outputs 1 in A^l_{mn} and 0 otherwise and c^l_{mn} is a coefficient. When l is fixed, the set $\{1_{A^l_{mn}}\}_{mn}$ is a base for this space; hence, the orthogonal projection is

$$\tilde{f} \to \sum_{m,n} \langle \tilde{f}, 1_{A^l_{mn}} \rangle 1_{A^l_{mn}}, \tag{6.5}$$

where each coefficient $\langle \tilde{f}, 1_{A^l_{mn}} \rangle = F^l_{mn} N^l_{mn}$. When we sample local features densely, we assume that N^l_{mn} is approximately equal for each m and n. This implies that the coefficients have almost equal information to F^l_{mn}. Thus, spatial pyramid matching is an orthogonal projection of the function representation of local features $\tilde{f}$ into a space of piecewise constant functions $\mathcal{F}_{\text{const}}$.



(a) $m = 0, n = 0$ (b) $m = 0, n = 1$

(c) $m = 1, n = 0$ (d) $m = 0, n = 2$

Figure 6.2: Values of Weights for Spatial Pyramid Matching

6.3 Spatial Orthogonal Pooling

In the previous section, we showed that spatial pyramid matching can be regarded as an orthogonal projection into a space of piecewise constant functions. The limitations of spatial pyramid matching that we stated in the introduction originate from the properties of the projected function space. Thus, we attempt to consider a different function space with better properties. To decide the function space we use, we need to decide the function space itself and the metric in the space that calculates the similarity between two functions.

Among all, metric is more important than the function space as long as the function space is large enough because metric directly encodes the value of position relation.

The requirement for the metric is that first it is easy to compute the metric and second is that we can encode the prior knowledge that the metric is robust

to the local translation. The candidates are standard inner product

$$\langle g, h \rangle = \int_{-1}^{1} \int_{-1}^{1} dxdyg(x, y)h(x, y), \tag{6.6}$$

and weighted inner product

$$\langle g, h \rangle = \int_{(-1,1)^4} dp_1 dp_2 g(p_1)h(p_2)w(p_1, p_2), \tag{6.7}$$

where w is the weight of the position p_1 and p_2. From the second requirement, the weighted inner product is preferred because standard inner product does not consider the product between the function value of different position. Also, from the requirement of easiness of computation of metric, we set that the Gaussain weight $w(p_1, p_2) = e^{-\frac{\|p_1 - p_2\|^2}{2a}}$ where a decides the bandwidth. As a function space, we require that there are nested function spaces that converges to L_2 space with respect to the metric to ensure that we can make the function space enough large as the dimension grows and we can choose the dimension of the space according to the computation resources. Also, the function space should be consists of smooth functions so that the pooling weight is robust to the local translation. Finally, we require that it is easy to figure out the meaning of the function space. Now, we consider the space of o-degree polynomials $\mathcal{F}^o_{\text{poly}}$. The space converges to L_2 when $o \to \infty$ and polynomials are small. We can regard the meaning of the basis as the orthogonal polynomial with respect to the metric. We call the proposed method spatial orthogonal pooling (SOP).

We calculate the weight of the case using standard inner product as a baseline in Section 6.3.1 and the case using weighted inner product in Section 6.3.2.

6.3.1 Spatial Orthogonal Pooling Using the Standard Inner Product

First, we consider the standard inner product of $L_2(-1, 1)^2$,

$$\langle g, h \rangle = \int_{-1}^{1} \int_{-1}^{1} dxdyg(x, y)h(x, y). \tag{6.8}$$

The orthogonal polynomials for this inner product are the products of the orthogonal polynomials for the one-dimensional (1D) inner product

$$\int_{-1}^{1} dxg(x)h(x), \tag{6.9}$$

for each element x and y, which is the definition of Legendre polynomials. m-th Legendre polynomial P_m, which can be written as

$$P_m(x) = \sqrt{\frac{m+1}{2}} \frac{1}{2^m m!} \frac{d^m}{dx^m}[(x^2-1)^m], \tag{6.10}$$

is a m-degree polynomial and satisfies the following property:

$$\int_{-1}^{1} dx P_m(x) P_n(x) = \delta_{m,n}. \tag{6.11}$$

Thus, P_ms for $0 \leq m \leq o$ compose a basis of $\mathcal{F}^o_{\text{poly}}$.

Thus, when we denote $Q_{mn}(x,y)$ as $P_m(x)P_n(y)$, then the proposed method concatenates the weighted average

$$\begin{aligned} F_{mn} &= \langle \tilde{f}, Q_{mn} \rangle & (6.12) \\ &= \sum_{k=1}^{N} f_k P_m(x_k) P_n(y_k), & (6.13) \end{aligned}$$

for $0 \leq m, n$ and $m+n \leq o$ to obtain the global feature. The pyramid size is $\frac{(o+1)(o+2)}{2}$. Note that the computational cost of calculating these weights is negligible compared to the computational cost of calculating the feature value.

6.3.2 Spatial Orthogonal Pooling Using a Weighted Inner Product

Next, we consider the inner product

$$\langle g, h \rangle_a = \int_{(-1,1)^4} dp_1 dp_2 g(p_1) h(p_2) e^{-\frac{\|p_1-p_2\|^2}{2a}}. \tag{6.14}$$

This inner product also summarizes the product of function values for a different position with the Gaussian weight $e^{-\frac{\|p_1-p_2\|^2}{2a}}$ and is thus more robust to image translations than the standard inner product. We can balance the robustness and spatial information by adjusting a. When $a \to +0$, this method converges to average pooling. When $a \to \infty$, this method converges to the pooling method proposed in Section 6.3.1. Note that we do not use Gaussian as a pooling weight directly. Instead, we calculate smooth weight function including both 0-th order information similar to Gaussian and high frequency information so as to approximate the Gaussian-weighted inner product.

Similar to the previous subsection, orthogonal polynomials for this inner product are products of orthogonal polynomials in the 1D case P_n^a. Let $Q_{mn}^a(x,y) = P_m^a(x)P_n^a(y)$. We concatenate the weighted average

$$F_{mn}^a = \langle \tilde{f}, Q_{mn}^a \rangle_a \tag{6.15}$$

$$= \sum_{k=1}^{N} f_k \langle \delta_{p_k}, Q_{mn}^a \rangle_a \tag{6.16}$$

for $0 \leq m, n$ and $m + n \leq o$ as a global feature. The pyramid size is $\frac{(o+1)(o+2)}{2}$. When the Gaussian weight $e^{-\frac{\|p_1 - p_2\|^2}{2a}}$ is used, we can calculate inner products $\langle x^{d_1}, x^{d_2} \rangle_a$ analytically using the error function by applying a variable transformation. Thus, P_n^a can be calculated using Gram-Schmidt orthonormalization. Furthermore, the inner products of orthogonal polynomials and delta functions

$$\langle \delta_p, Q_{mn}^a \rangle_a = \int_{(-1,1)^2} dp_1 Q_{mn}^a(p_1) e^{-\frac{\|p - p_1\|^2}{2a}} \tag{6.17}$$

can be written as functions of p and a. Thus, the complexity of calculating the weight is approximately the same as when the standard inner product is used and can be ignored. Figure 6.3 shows an example of the weights used in the experiment. This figure shows that the weights have similar information to those of original spatial pyramid matching. For example, Figure 6.3 (a) is a smoother version of the weight of layer 0 of spatial pyramid matching. Figure 6.3 (b), (c), and (d) construct the weights of layer 1. Since the weight is smooth, the proposed weights are both robust to local translation and have the spatial information comparable to spatial pyramid matching.

6.3.3 Analysis of the Robustness of the Proposed Methods

In this subsection, we analyze how the global feature changes when the positions of local features are slightly changed. We denote the position change as $\tau = (\tau_x, \tau_y)$ and assume that $\tilde{f}$ has a nonzero value only in $(-1 + \|\tau\|, 1 - \|\tau\|)^2$ for simplicity. In this case, the change of the global feature F_{mn}^a, δF_{mn}^a, can be bounded by

$$|\delta F_{mn}^a| = |\sum_{k=1}^{N} f_k \left(\langle \delta_p, Q_{mn} \rangle - \langle \delta_{p+\tau}, Q_{mn} \rangle \right)| \tag{6.18}$$

$$\leq \sum_{k=1}^{N} |f_k| \max_k |\langle \delta_{p_k}, Q_{mn} \rangle - \langle \delta_{p_k+\tau}, Q_{mn} \rangle|. \tag{6.19}$$

Thus, we evaluate the bound for

$$|\langle \delta_p, Q_{mn} \rangle - \langle \delta_{p+\tau}, Q_{mn} \rangle|. \tag{6.20}$$

When the standard inner product is used, by applying the mean value theorem, this value can be written as

$$\begin{aligned} &|Q_{mn}(p) - Q_{mn}(p+\tau)| \\ &= |\tau_x P'_m(x+\theta\tau_x) P_n(y+\theta\tau_y) + \tau_y P_m(x+\theta\tau_x) P'_n(y+\theta\tau_y)|, \end{aligned} \tag{6.21}$$

for some $0 < \theta < 1$. Because P_m are polynomials, the $P'P$ terms on the right-hand side are bounded. Thus, for some constant c, the right-hand side is bounded by $c\|\tau\|$. Finally, the change of the global feature $|\delta F^a_{mn}|$ can be bounded using $\sum_{k=1}^{N} |f_k|, \tau$.

Similarly, by applying the mean value theorem to $e^{-\frac{\|p-p_1\|^2}{2a}}$ in Eq. (13), a bound can be derived for the case when the weighted inner product is used. This analysis is based on the smoothness of the basis function, so robustness is not necessarily ensured for spatial pyramid matching, which uses non-smooth piecewise constant functions as a basis.

6.4 Experiments

We tested our pooling methods on standard object recognition datasets and an action recognition dataset. In Section 6.4.1, we applied our methods on the image recognition datasets. In Section 6.4.2, we applied our methods on the action recognition dataset.

6.4.1 Image Recognition

First, we evaluated our methods with SIFT + FV on the image recognition datasets. We tested our methods on three object recognition datasets, including fine-grained datasets (Caltech UCSD Birds 200 (CUB-200), Stanford Dogs, and Caltech256 dataset (Caltech256)).

CUB-200 [94] is a standard fine-grained object recognition dataset that consists of 200 bird species with 60 images per class. The Stanford Dogs dataset [95] consists of approximately 20,000 images of 120 dog classes. The Caltech256 dataset [96] consists of approximately 30,600 images of 256 object classes. We used given train/test split for the CUB-200 dataset and Stanford Dogs dataset and evaluated the accuracy. For the Caltech256 dataset, we randomly sampled 25 images per class as training data and 30 images per class as test data 10 times and evaluated the average of the accuracy.

For all datasets, we extracted 128-dimensional SIFT features densely with a step size of two and scales 4, 6, 8, and 10. We used 'vl_phow' implemented in VLFeat [97] for extraction. We used PCA to reduce the dimensionality of the features to 80. Then, each local descriptor was encoded using the FV with 128 clusters. We used 250,000 local features to learn the codebooks. For each dataset, we applied spatial pyramid matching with scales $[1 \times 1]$, $[1 \times 1, 2 \times 2]$, $[1 \times 1, 2 \times 2, 1 \times 3]$, $[1 \times 1, 2 \times 2, 3 \times 3]$, $[1 \times 1, 2 \times 2, 4 \times 4]$, which had pyramid sizes of 1, 5, 8, 14, and 21, respectively. We also compared the method that applied gaussian weight on the local feature according to the position in each pyramid as a baseline. We evaluated the proposed weighted average pooling using Legendre polynomials of degree 0, 1, 2, 3, 4, and 5 had pyramid sizes of 1, 3, 6, 10, 15, and 21, respectively and the proposed weighted average pooling using a weighted inner product with kernel parameter $a = 0.25$, 0.5, 1.0 and degree 0, 1, 2, 3, 4, 5 and had pyramid sizes that were the same as those using Legendre polynomials. We did not compare max pooling because this pooling does not work well on Fisher Vector. For post-processing, we applied power normalization plus L_2 normalization on each pyramid for spatial pyramid matching and power normalization plus L_2 normalization on the entire vector for the proposed methods.

For the linear classifier, we used a one-vs-rest SVM. We used the SVM implemented in LIBLINEAR [98] with $C = 100$ for training and plotted the accuracy.

Figure 6.4 shows the results for the original methods, where SOP indicates the results for Spatial Orthogonal Pooling using standard inner product, SOP with numbers indicate the results for Spatial Orthogonal Pooling using weighted inner product with the number meaning kernel parameter. The figures show that the performance is ranked as follows: *spatial orthogonal pooling with the standard inner product* < *spatial pyramid matching* < *gaussian-weighted spatial pyramid matching* < *spatial orthogonal pooling with an appropriately weighted inner product*. The poor performance of the standard inner product may be because Legendre polynomials are not sufficiently robust to small translations. The appropriate choice of the kernel parameter contributes to the performance. In each dataset, the case for the kernel parameter $a = 0.25, 0.5$ demonstrates good performance. This is close to $1/3$, which is the variance of the uniform distribution in $(-1, 1)$. In addition, the performance of spatial pyramid matching saturates around $[1 \times 1, 2 \times 2, 1 \times 3]$, but the performance of the proposed methods rapidly improves until the degree is two, and the performance gradually increases until the degree is five. Thus, the proposed methods demonstrate good performance, even when the pyramid size is small; moreover, better accuracy can be achieved using higher degree polynomials.

6.4.2 Action Recognition

Next, we applied our method to the action recognition task on two movie datasets, HMDB51 dataset [99] that consists of about 7,000 movie clips with 51 action categories, and UCF101 dataset [100] that consists of 13,320 movie clips with 101 action categories.

As a local descriptor, we extracted TDD features [43] that are the mean of output of convolutional layer around each improved dense trajectory [101]. As a CNN, we used VGG16 [32] network pretrained with spatial (RGB) and temporal (opticalflow) images and extracted the output of conv3, 4, and 5 layer as local features. Then we coded TDD feature using FV with dimension of the local descriptor reduced to 64 and the number of clusters 256 and then applied the linear SVM with $C = 100$.

In this case, since the feature dimension is larger than that used in the image recognition dataset and time-axis is finer compared to image-space with respect to the position of TDD features, we only consider spatal pyramid with respect to time-axis.

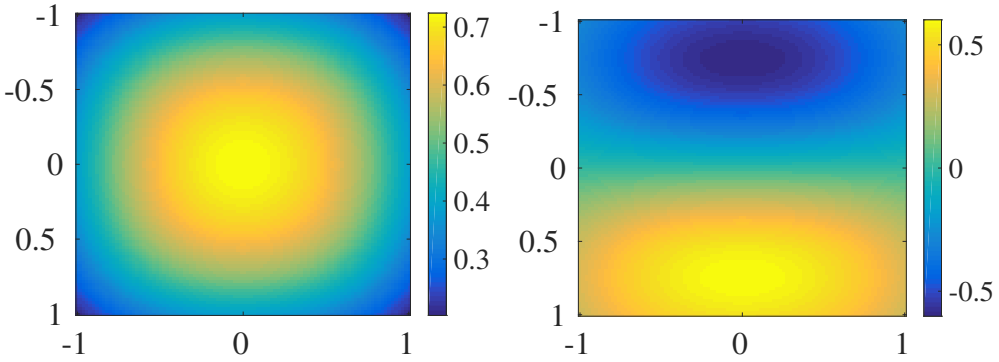
For each dataset, we compared spatial pyramid matching with scales $[1 \times 1 \times 1], [1 \times 1 \times 1, 1 \times 1 \times 2], [1 \times 1 \times 1; 1 \times 1 \times 2; 1 \times 1 \times 4]$, which had pyramid sizes of 1, 5, and 7, respectively and proposed methods with degree 0, 1, 2, 3, and 4 with pyramid sizes 1, 2, 3, 4 and 5 respectively. Each dataset gives 3 train/test splits and we evaluated the average accuracy using these 3 splits.

Figure 6.5 shows the results. The proposed methods show much better accuracy than spatial pyramid matching on HMDB51 dataset even when the pyramid size is small. Also, the proposed methods with weighted inner products show comparable performance on UCF101 dataset.

Next, we plotted the score of each layer in UCF101 dataset in Figure 6.6 to evaluate the effect of spatial pooling in detail. We can gain performance by considering spatial information on RGB input, while spatial information does not work well on flow image. This is because while we can decide the time position of each RGB image exactly, flow image used the optical flow information of 10 frames around the time. Since each movie clip has the time of the order of seconds, this time width of the frame is not negligible and time information is much noisy. In such situation, though the proposed method with standard inner product shows poor performance, the proposed method with weighted inner product is as much robust as spatial pyramid matching and show a little better score. Considering that our methods show better performance on RGB input image, the proposed methods are effective for both spatial and temporal input. These results showed that our methods can extract time-domain information much better than spatial pyramid matching. Thus our methods are also effective for movie recognition.

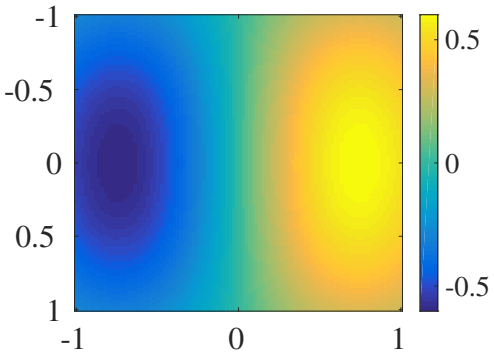
6.5 Discussion

In this chapter, we provided an interpretation of spatial pyramid matching as an orthogonal projection into the function space by considering local features in an image as a function on the image space. We also proposed a novel feature pooling methods called Spatial Orthogonal Pooling that used the weighted average as orthogonal projections into a space of low-degree polynomials and evaluated robustness to image translations of the proposed methods. Experimental evaluations using image recognition datasets and action recognition datasets demonstrated that the proposed pooling methods resulted in higher accuracy than spatial pyramid matching in both cases in which the pyramid size was small and large. These results showed that proposed methods exploit spatial information more effectively.

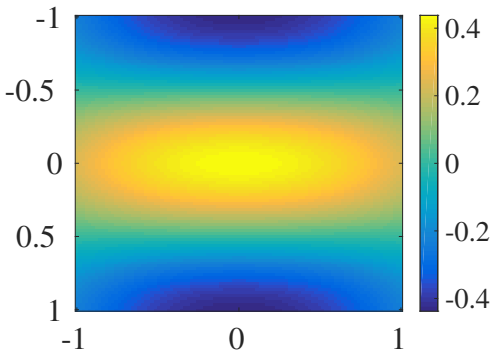


(a) $m = 0, n = 0$

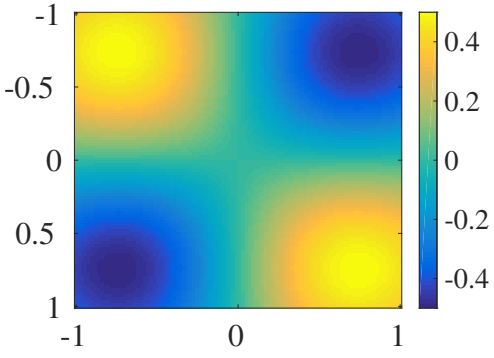
(b) $m = 0, n = 1$



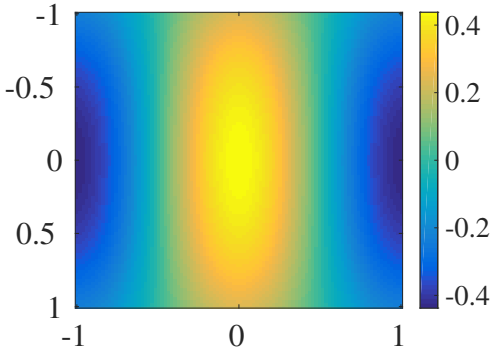
(c) $m = 1, n = 0$



(d) $m = 0, n = 2$

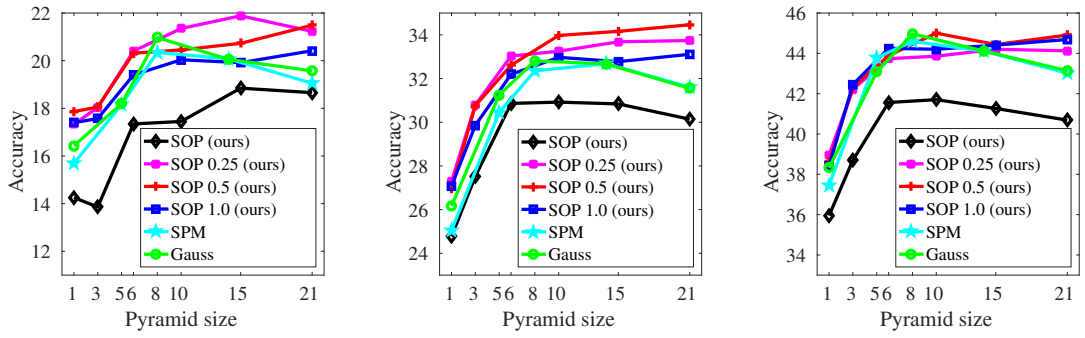


(e) $m = 1, n = 1$



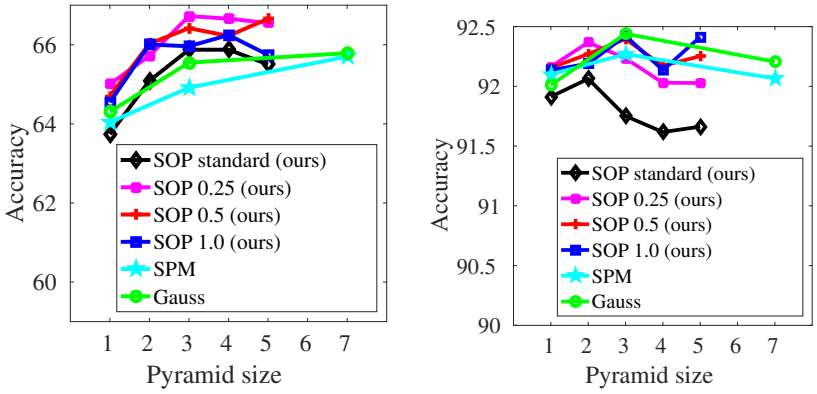
(f) $m = 2, n = 0$

Figure 6.3: Values of $\langle \delta_p, Q^a_{mn} \rangle_a$ with small m and n for $\alpha = 0.25$.



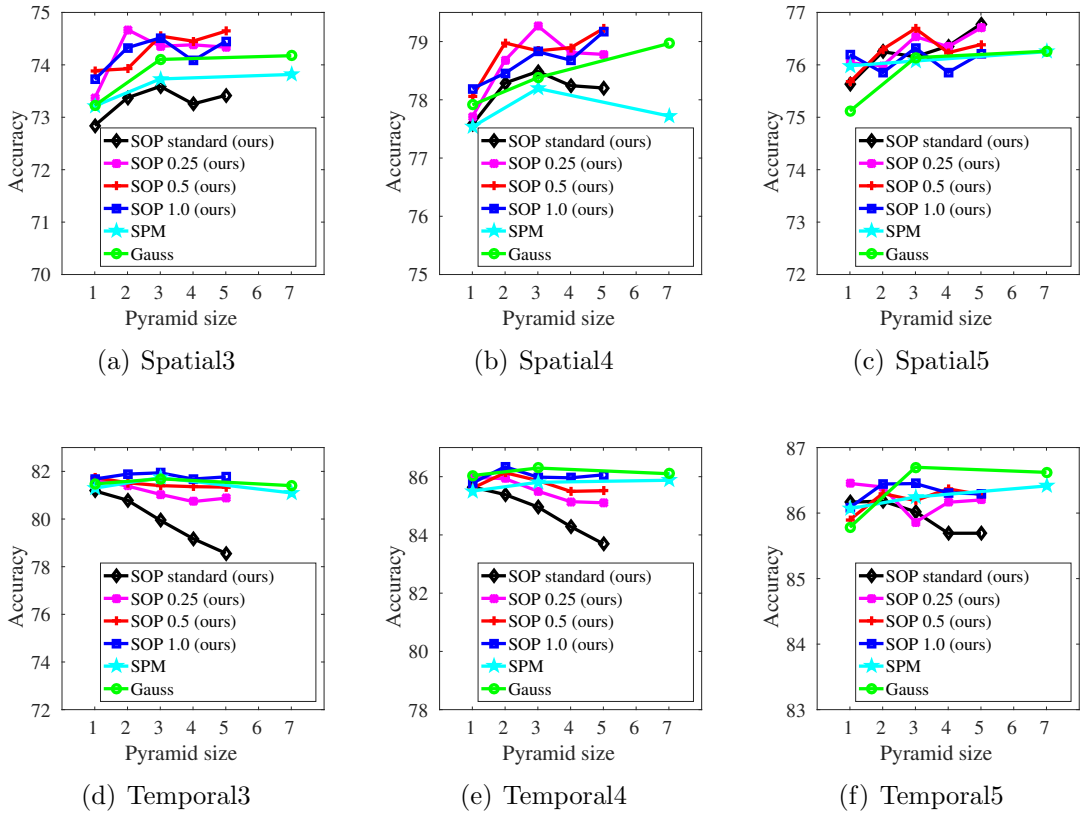
(a) CUB-200 (b) Stanford Dogs (c) Caltech256

Figure 6.4: Comparison of classification performance using SIFT + FV in (a) CUB-200 dataset, (b) Stanford Dogs dataset, and (c) Caltech256 dataset.



(a) HMDB51 (b) UCF101

Figure 6.5: Comparison of classification performance using TDD + FV in (a) HMDB51 dataset and (b) UCF101 dataset.



(a) Spatial3 (b) Spatial4 (c) Spatial5

(d) Temporal3 (e) Temporal4 (f) Temporal5

Figure 6.6: Comparison of classification performance of each layer using TDD + FV in UCF101 dataset. 'Spatial' indicates the score that we used the features extracted from RGB image and 'Temporal' indicates the score for the features extracted from flow image. The number in the name indicates the number of the layer.

Chapter 7

Evaluation of the Whole Architecture

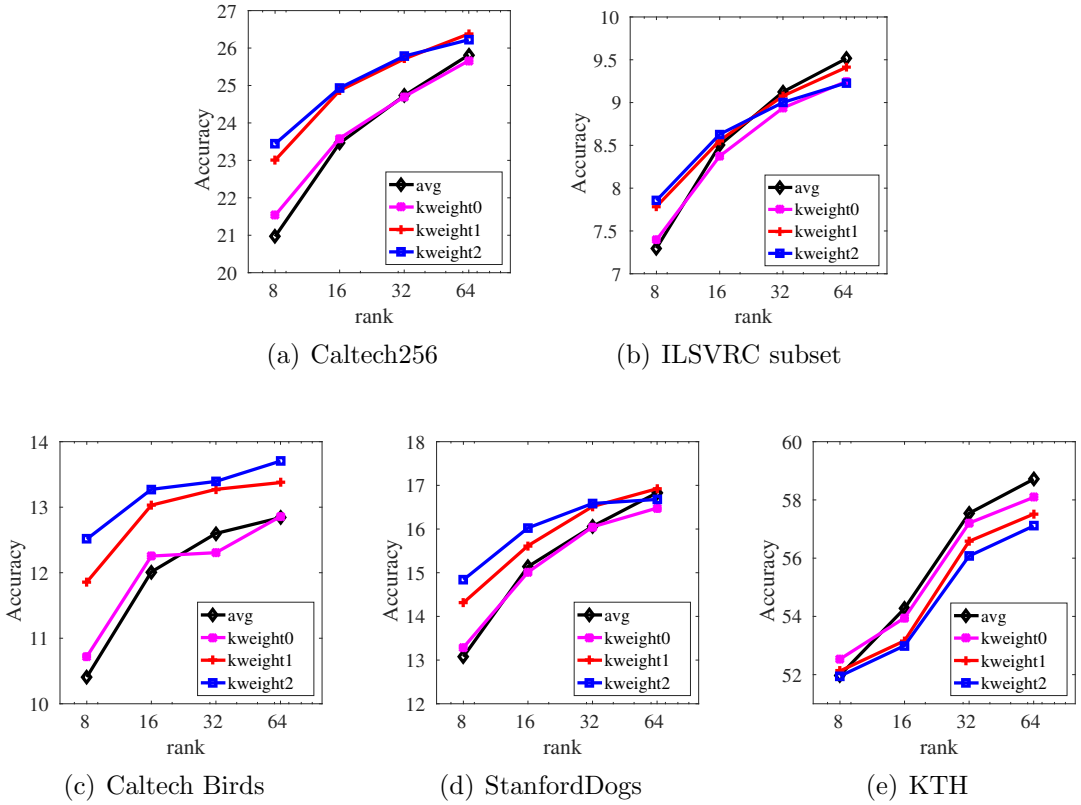
Since we have proposed a method for feature extraction, feature encoding, and feature pooling, we can now combine these three modules and extract one global feature from the raw image.

Our final framework is that first extracts raw patch feature based on the convolutional kernel network architecture combined with proposed empirical orthogonal decomposition method that is described in Chapter 4. Then apply each local patch feature the low-rank covariance encoding based on matrix manifold that we described in Chapter 5. Then we summarize each encoded local patch feature associated with the position of the feature in the image into one global feature that we explained in Chapter 6. Finally, we apply non-linear embedding using kernel approximation method that is explained in Chapter 3 and finally applies linear SVM to output the category of the image.

7.1 Experimental Setting

We now evaluate the performance of the whole framework. To evaluate the accuracy in several domain, we tested our method using two standard object recognition datasets (Caltech256 dataset [96], and ilsvrc subset [30]), two fine-grained image recognition datasets (CUB-200 [94], Stanford Dogs dataset [95]), and one texture recognition dataset (KTH-TIPS-2b [102, 103]). We omitted MNIST [28], CIFAR-10 [29], CIFAR-100 [29], and SVHN [77] because they are very easy and the size is too small to extract position information.

The Caltech256 dataset consists of approximately 30,600 images of 256 object classes. We randomly selected images with 500 classes and 60 images per class. CUB-200 is a standard fine-grained object recognition dataset that consists of



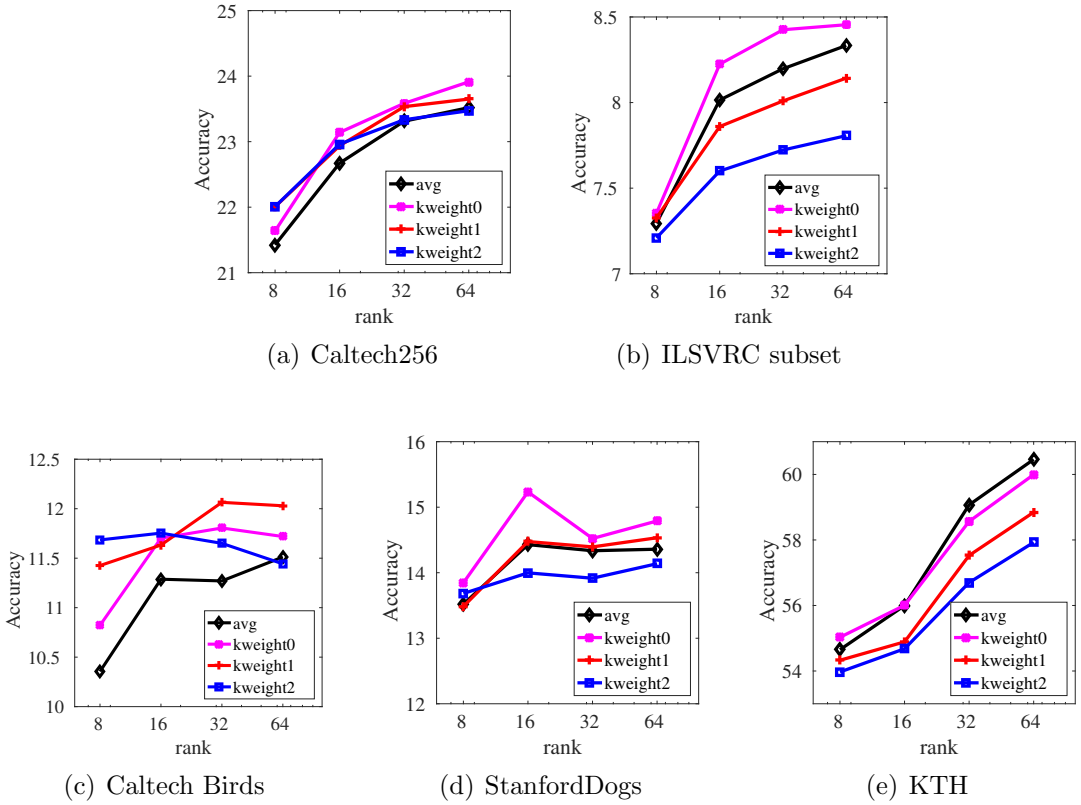
(a) Caltech256 (b) ILSVRC subset

(c) Caltech Birds (d) StanfordDogs (e) KTH

Figure 7.1: Accuracy using the architecture without non-linear embedding on global feature

200 bird species with 60 images per class. The Stanford Dogs dataset consists of approximately 20,000 images of 120 dog classes. KTH-TIPS-2b contains 11 material categories with 4,752 images. For the Caltech256 dataset, we randomly sampled 25 images per class as training data and 30 images per class as test data 5 times and evaluated the average of the accuracy. We randomly sampled 30 images per class as training and the rest as test data 5 times and evaluate the average of the accuracy. We used given train/test split for the CUB-200 dataset and Stanford Dogs dataset and evaluated the accuracy. We used the average of 4 kinds of given train/test split for KTH dataset.

As a network architecture, we modified the architecture proposed by Alex Krizhevsky [31] that consists of 11×11 convolution layer with filter size 96 and stride 4, 3×3 max pooling with stride 2, 5×5 convolution layer with filter size 256 and pad 2, 3×3 max pooling with stride 2, two 3×3 convolution layers with filter size 384 with pad 1, 3×3 convolution layer with filter size 256 with pad 1, and three fully-connected layer with number of hidden states 4,096 with



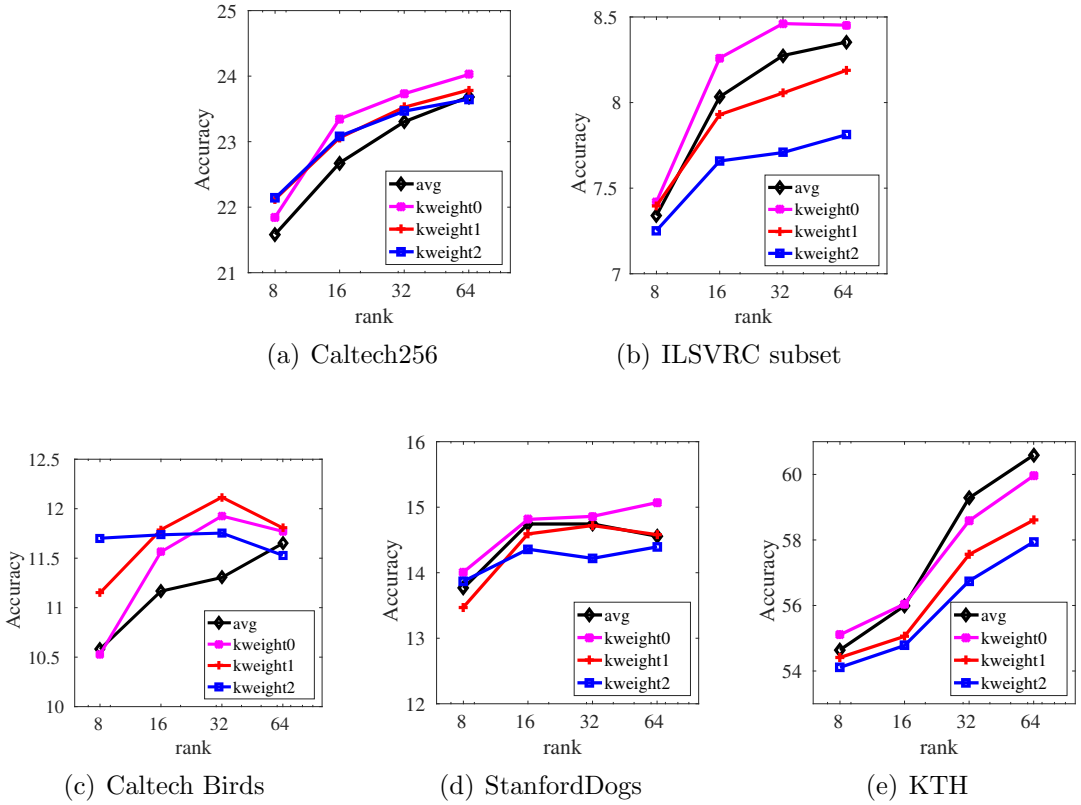
(a) Caltech256 (b) ILSVRC subset

(c) Caltech Birds (d) StanfordDogs (e) KTH

Figure 7.2: Accuracy using the architecture using non-linear embedding with 8,192 dimension on global feature.

relu activation function. As for our method, we replace convolutional layer + relu with our proposed learned feature function based on empirical orthogonal decomposition and max pooling with average pooling. We also replace fully-connected layers with our low-rank covariance encoding, proposed spatial orthogonal pooling, empirical orthogonal decomposition, and linear SVM.

We varied the rank of low-rank covariance encoding as 8, 16, 32, and 64. We used the pooling method as average pooling and proposed spatial orthogonal pooling using gaussian weight with band width 0.25 and degree of polynomials as 0, 1, and 2. As a final embedding, we used the linear embedding and orthogonal embedding with final dimension 4,096, 8,192. Since the dimension of local feature is 256, the dimension of the global feature is 256 times the rank of covariance encoding times the pyramid size when we use the linear embedding and the chosen dimension of global feature when orthogonal embedding is used.



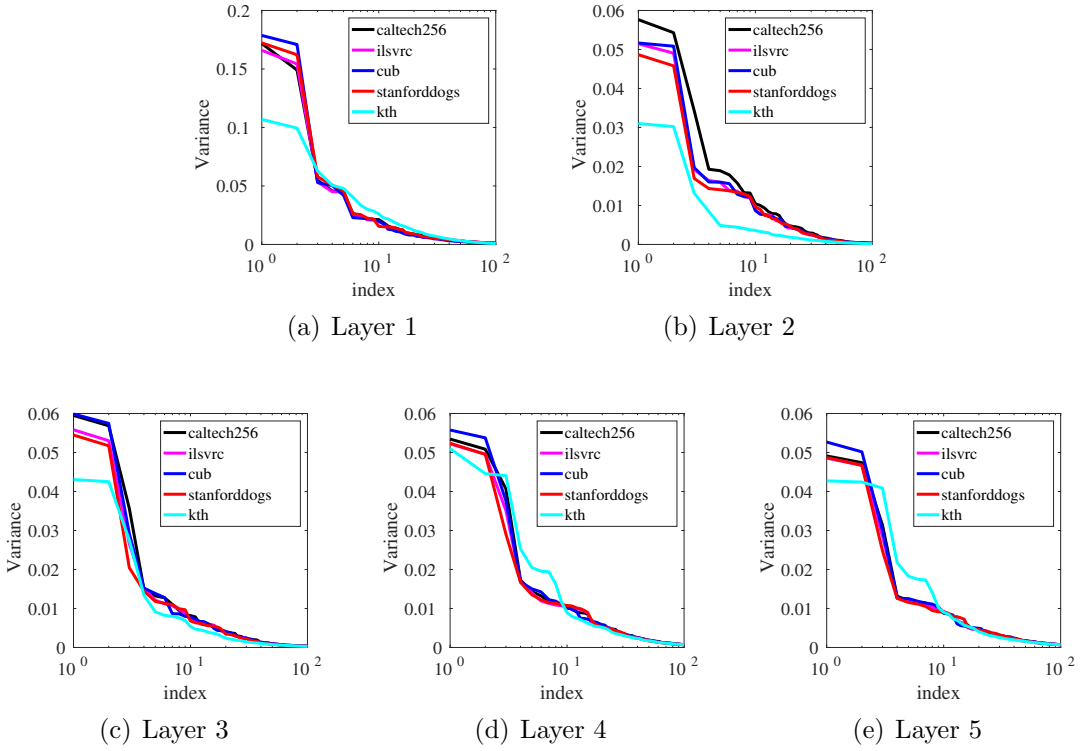
(a) Caltech256 (b) ILSVRC subset

(c) Caltech Birds (d) StanfordDogs (e) KTH

Figure 7.3: Accuracy using the architecture using non-linear embedding with 4,096 dimension on global feature.

7.2 Result

Figure 7.1 shows the results that we use linear embedding for global feature. As for pooling weight, we can gain performance by using our pooling method and increasing the pyramid size except for KTH dataset. This is consistent to fact that KTH is a texture dataset and spatial information does not contribute to classification but spatial information is useful in object recognition. Also, the accuracy increases as the rank of fixed-rank PSD increases. Our encoding using fixed-rank PSD works well till 64 rank. As the pyramid size grows, the performance gain is larger when the rank is smaller. Since the statistics correspond to higher-degree polynomials are noisier than that correspond to smaller-degree polynomials and the statistics correspond to subspace with smaller eigenvalues are noisier than that of larger eigenvalues. Thus, the statistics correspond to higher-degree polynomials and subspace with smaller eigenvalues are very noisy and have smaller impact on the performance. Thus, when the pyramid size is



(a) Layer 1 (b) Layer 2

(c) Layer 3 (d) Layer 4 (e) Layer 5

Figure 7.4: Distribution of eigenvalues of covariance of features in each layer.

large, we can gain performance even when the rank is small.

Figures 7.2 and 7.3 show the result when we use non-linear embedding on the global feature to 4,096 and 8,192 dimension respectively and apply linear SVM. These figures show that we can gain performance for KTH, but the linear method works better for the rest. KTH has the small number of categories and has relatively large number of images per each category. Thus, non-linear method works well. However, for the rest dataset, the original global feature is already highly non-linear and adding further nonlinearity does not work well. Also, the case for 4,096 dimensions and 8,192 dimensions show similar performance. Thus, 4,096 dimensional embedding is enough in our case.

Figure 7.4 shows the top-100 eigenvalues of covariance of input vectors in each layer. KTH shows relatively large eigenvalues even when the index is large. This indicates the variety of local patch patterns of texture datasets. Object recognition datasets including fine-grained datasets show similar tendency. Fine-grained StanfordDogs show a little larger decay of eigenvalues and Caltech256 dataset show a little smaller decay of eigenvalues especially in Layer 2. We can assume that the patches are similar for fine-grained datasets and thus the

Methods	Caltech256	ILSVRC subset	Caltech Birds	StanfordDogs	KTH
Fisher Vector	15.7	8.15	7.51	12.6	49.4
Auto Encoder	10.0	4.0	4.8	6.1	37.3
Ours (Linear)	**26.4**	**9.5**	**13.7**	**16.7**	58.1
Ours (Nonlinear 4,096)	23.7	8.45	11.8	14.5	60.0
Ours (Nonlinear 8,192)	23.8	8.45	11.5	14.9	**60.6**

Table 7.1: Comparison of accuracy with existing methods.

variance decays quickly. However, by applying non-linear embedding repeatedly, this difference becomes smaller as it becomes a higher layer.

7.3 Comparison with existing methods

Also, we compare the result with two existing unsupervised feature extraction methods, Fisher Vector and Auto Encoder. As for Fisher Vector, we extract RGB patch with size 25×25 with step size 16 from the image resized to 227×227 to match the number of local feature to that of our method. Then we encode the raw RGB vector using Fisher Vector with local feature dimension reduced to 80 and the number of components 128. Thus the dimension of the global feature is $2 \times 80 \times 128$. As for Auto Encoder, the architecture of autoencoder is the same as the first 7 layer of AlexNet and the decoder consists of two fully-connected layers and four deconvolution layers. We learn the model using l2 loss and apply linear SVM to the 4,096-dimensional encoded feature. As for our method, we used the score when we select the rank of the matrix manifold and pooling method using cross validation per each embedding method of global feature.

Table 7.1 shows the accuracy of Fisher Vector and Auto Encoder and our method, where Ours (Linear), Ours (Nonlinear 4,096), and Ours (Nonlinear 8,192) denote the score of the method that uses the linear embedding and orthogonal embedding with final dimension 4,096, 8,192 respectively. Our method shows better performance than both methods. Compared to Auto Encoder, while Auto Encoder is trained to model the generation of image itself, our method models the relationships between images, thus we can model more complex information and that contributes to the classification accuracy. Compared to Fisher Vector, our method has larger layers and learn the feature more than Fisher Vector. Thus, our method can extract more complex feature that reflect the information of training data. This property contributes to the performance of the proposed method.

These results show that the proposed framework can extract discriminative information well.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

In this research, we propose a novel image feature extraction method that can be learned without label and can be analyzed easily by proposing a novel framework that regards the image feature extraction modules uniformly as Euclidean Embeddings from the bag of local features to the low-dimensional vector, and by combining our image feature extraction framework with novel data-driven kernel approximation method. We analyzed the performance of our approximation method and proposed a method to apply our method to each module used in image feature extraction. Our framework is the generalization of the framework based on the hand-crafted features that exploits data information more frequently and our framework is also an extension of auto encoder that models the relationship between images instead of image itself and that uses additional encoding and pooling layers that exploit the prior knowledge. To the best of our knowledge, this paper is the first work that found out the common structure between local feature extraction and global feature extraction and proposed the feature extraction method that exploits this structure.

In Chapter 3, we proposed a kernel approximation method based on the orthogonal decomposition with distribution estimated from training data. We calculated the bound for the kernel approximation performance and experimentally demonstrated that our method can better approximate the kernel than existing methods.

In Chapter 4, we apply our framework to local feature extraction by combining our method to Convolutional Kernel Networks architecture. From the experiment, we observed that we can fastly calculate the effective local feature and our feature is more uncorrelated than existing features.

In Chapter 5, we apply our framework to the covariance encoding by assuming the distribution of the tangent space of matrix manifold of fixed-rank PSD. We

observed that our new feature demonstrates good performance for several input with small feature dimension.

In Chapter 6, we apply our framework to feature pooling method by introducing the function representation for the bag of local features in one image and considering the kernel and distribution in function space. From our method, we can derive a smooth and independent feature pooling weight and correlation between pooled value, resulting in good recognition performance with respect to the feature dimension.

In Chapter 7, we combine the methods proposed in Chapter 3, 4 and 6 into the end-to-end unsupervised image feature extractor. We applied our proposed method to standard object recognition datasets and texture recognition dataset and demonstrated that our method shows better accuracy than Fisher Vector and Auto Encoder.

Finally, we summarize the relationship between our proposed Euclidean Embedding from the bag of local features $\{(f_i, p_i)\}_{i=1}^K$ to the global feature F written as

$$F = \sum_{i=1}^{K} \xi_{\text{pos}}(p_i) \otimes \xi_{\text{feat}}(f_i), \tag{8.1}$$

and the proposed feature extraction methods in each chapter. In Chapter 4, we apply the embedding to the bag of pixel values in the local image patch to construct the local patch feature by learning ξ_{feat} unsupervisedly as an approximation of the Gaussian kernel calculated as Equation 3.17 in Chapter 3 and by using gaussian weight for ξ_{pos}. In Chapter 5 and 6, we apply the embedding to the bag of local features in the whole image to construct the global feature by applying fixed-rank covariance encoding calculated in Chapter 5 where $n(f_i f_i^t - \mu\mu^t)$ is substituted to Σ_n as ξ_{feat} and by using spatial orthogonal pooling written as Equation 6.17 in Chapter 6 as ξ_{pos}.

In summary, our findings in our research are as follows:

- We can approximate the nonlinear Gaussian kernel efficiently by estimating Gaussian distribution from training data.
- We can calculate the efficient patch feature by approximating convolutional kernel by our method.
- We can extract low-dimensional covariance feature by assuming the tangent space of matrix manifold.
- We can derive various pooling methods by regarding bag of local features as a function from image space to feature space.

- Our feature extraction framework show better accuracy than existing unsupervised feature learning methods.

8.2 Future Works

In this research, we proposed a method to derive the image feature extraction method based on kernel approximation and Euclidean embeddings. As a future work, first is to encode more general prior knowledge. We have applied our method to simple Gaussian kernel with Gaussian Distribution, but we will need to handle various kernels with respect the prior knowledge. To come down to our method, we need to apply variable transformation based on the invariance property of the kernel. This derivation is in our future work. Second is the application to the different modal. We handle image feature extraction in our study, but our framework is general and we can apply our method to the modal such as sound and video recognition. The evaluation of the performance on such modal is in our future work.

Appendix A: Fast Random Features for Semigroup Kernels

In this chapter, we describe the method for approximating semigroup kernels. The method is not included in our framework, but contributes to fast approximation of kernel function.

The random features method is an approximation method that first represents the given kernel function by the expectation of the product of the parameterized function values of two inputs and then approximates the kernel with the inner product of vectors composed of the values of randomly sampled functions. Various random features techniques have been proposed, with their construction depending on the structure of the given input feature and kernel function [55, 67, 104, 105].

When we apply the random features method on large-scale data, we need to consider the cost involved in calculating the approximate feature vectors. Naive approximation methods often require $O(Dd)$ computation, where d and D denote input and output dimensions, respectively. This is not negligible when d is large because in most cases, $D \geq d$. Thus, recent work has proposed methods to calculate the features with $O(D \log d)$. Most existing work approximates the Gaussian kernel [62, 106, 107] or polynomial kernels [108]. In this work, we propose a fast method that approximates semigroup kernels that is effective for positive vectors but where we cannot apply existing fast methods directly.

A wide range of features, including histograms, discrete probabilistic distributions, and activations of convolutional neural networks (CNNs), are associated with the space of positive vectors $\mathbb{R}_+^d$. The semigroup kernel is a kernel that exploits such positive definiteness. For two inputs $x, y \in \mathbb{R}_+^d$, the kernel value of the semigroup kernel depends only on the summation of the two inputs $x + y$. In such a case, Berg *et al.* [109] have proven that there exists a distribution $p(w)$ on $\mathbb{R}_+^d$ such that

$$k(x + y) = \int e^{-w^t(x+y)} p(w) dw. \tag{A.2}$$

Based on (A.2), Yang *et al.* [67] proposed random Laplace features, which use

$\{\sqrt{\frac{1}{D}}e^{-w_j x}\}_{j=1}^{D}$ with $\{w_j\}_{j=1}^{D}$ sampled from $p(w)$ as a D-dimensional feature vector. Random Laplace features exhibit superior performance to existing commonly used kernels such as the Gaussian and χ^2 kernels with regard to the classification of histogram data. The computational complexity is $O(Dd)$. As the dimension of the input feature grows, this linear dependency on d becomes non-negligible.

In this paper, we propose a method to calculate the feature vector with $O(D \log d)$ complexity by exploiting structured matrices to approximate the semigroup kernel quickly. Structured matrices are matrices with special structures that enable us to calculate the matrix-vector products recursively with $O(d \log d)$ time complexity and $O(d)$ memory at the expense of independence between different feature elements. Existing fast Gaussian kernel approximation methods [62, 106, 107] combine structured matrices with random sign flipping to reduce the correlation between different feature elements. For random Fourier features on a shift-invariant kernel such as Gaussian kernel, the sample distribution is symmetric. Thus, sign flipping does not change the output distribution. However, for random Laplace features, the support of the weight distribution $p(w)$ and the input x are positive. Thus, sign flipping changes the output distribution. Instead, we propose a novel method called "alternating circulant random features" that utilizes weights randomly sampled from multiple independent structured matrices to calculate the features. Because the proposed method uses structured matrices, the computation complexity depends on the log of the input dimension d. Further, the proposed method preserves the weight distribution by replacing random sign flipping with random choice from the variables sampled from the same distribution. Thus, this technique can be applied to semigroup kernels.

In addition, we demonstrate that the covariance for random circulant features without random sign flipping is as large as autocovariance, while the covariance for the proposed method becomes small with respect to the autocovariance as d increases. Thus, the proposed features exhibit comparable performance to random Laplace features.

We compare the performance of the proposed method to that of an approximate semigroup kernel for application to Bag of Visual Words features that can be regarded as a histogram, and to the CNN features. We demonstrate that the proposed method exhibits comparable classification performance to the random Laplace features method with a significantly shorter computation time.

In summary, the contributions of this paper are as follows:

- We propose novel fast random features called "alternating circulant random features"; the proposed method can compute random features with computation complexity $O(D \log d)$ and $O(D)$ memory requirement.

- We theoretically evaluate that the approximation error of the proposed

Method	Memory	Time	Gaussian	Semigroup
Random Fourier / Laplace Features	$O(Dd)$	$O(Dd)$	Yes	Yes
FastFood	$O(D)$	$O(D \log d)$	Yes	No
Circulant	$O(D)$	$O(D \log d)$	Yes	Partially Yes
Structured Orthogonal Random Features	$O(D)$	$O(D \log d)$	Yes	No
Alternating Circulant Random Features (ours)	$O(mD)$	$O(mD \log d)$	Yes	Yes

Table A.1: Comparison of kernel approximation methods. Here, d and D denote the dimensions of the input and output features, respectively, and m is the number of mixed structured matrices (2 or $\log_2 d$ in this study). "Gaussian" and "Semigroup" indicate the method applicability to Gaussian and semigroup kernels, respectively. Note that the random circulant features method can be applied to semigroup kernels if we omit random sign flipping.

method is comparable to that of the random Laplace features method as the input dimension d grows.

- Experimental results using Bag of Visual Words features and CNN features show that the proposed method achieves similar accuracy to the random Laplace features method and requires less computation time.

A.1 Background

For a given kernel function $k(\cdot, \cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ expressed as the expectation of the product of the parameterized function $\psi_w(\cdot) : \mathbb{R}^d \to \mathbb{R}$ with respect to the parameter w, denoted as $E_{w \sim p(w)}[\psi_w(\cdot)\psi_w(\cdot)]$, random features use $\{\sqrt{\frac{1}{D}}\psi_{w_j}(\cdot)\}_{j=1}^D$ with $\{w_j\}_{j=1}^D$ randomly sampled from p as a feature function to approximate the kernel value.

Semigroup kernels are the kernels whose values depend only on the summation $x + y$ of the two inputs $x, y \in \mathbb{R}_+^d$. For the semigroup kernel, there exists a distribution $p(w)$ on $\mathbb{R}_+^d$ that satisfies (A.2). Random Laplace features use $\{\sqrt{\frac{1}{D}}e^{-w_i x}\}_{i=1}^D$ with $\{w_i\}_{i=1}^D$ sampled from $p(w)$ as a feature. In this study, in addition to the assumption of semigroup kernel, we also assume that the kernel can be decomposed as the product of the kernels of each dimension k_1, as $k(x + y) = \prod_{j=1}^d k_1(x_j + y_j)$. Note that the exponential-semigroup kernel $e^{-\beta \sum_{j=1}^d \sqrt{x_j + y_j}}$ and reciprocal-semigroup kernel $\prod_{j=1}^d \frac{\lambda}{x_j + y_j + \lambda}$, to which Yang *et al.* applied random Laplace features, satisfy this assumption. When we denote the matrix $W = [w_1, w_2, \cdots w_D]^t \in \mathbb{R}^{D \times d}$, the features are expressed as $\sqrt{\frac{1}{D}}e^{-Wx}$.

We call the kernel function defined by two inputs $x, y \in \mathbb{R}^d$ a "shift-invariant kernel," if the kernel value depends on the difference in the input $x - y$ only. Bochner [59] has shown that the shift-invariant kernel k can be expressed with the measure $p(w)$ on $\mathbb{R}^d$ as

$$k(x - y) = \int e^{iw^t(x-y)} p(w) dw. \tag{A.3}$$

Random Fourier features methods [55] randomly sample $\{w_j\}_{j=1}^D$ from $p(w)$ in (A.3) and use $\sqrt{\frac{1}{D}} e^{iWx}$ or $\sqrt{\frac{2}{D}} \cos(Wx + b)$, with b uniformly sampled from $[0, 2\pi]$, as features.

Recently, fast computation methods have been proposed for the Gaussian kernel, which is one of the shift-invariant kernels. These approaches use structured matrices where the matrix-vector product can be calculated with $O(d \log d)$ and random diagonal matrices where the matrix-vector product can be calculated with $O(d)$ [62, 106, 107]. These studies have allowed calculation of d-dimensional random features with $O(d \log d)$ time complexity and we achieve D-dimensional features with $O(D \log d)$ complexity by independently sampling the features D/d times. Such fast random features exhibit comparable or superior approximation performance to the original random Fourier features.

A.2 Related Work

Various approximation techniques have been proposed depending on the kernel structures [55, 67, 104, 108]. In this section, we focus on fast random Fourier features using structured matrices. The properties of the existing fast approximation methods and the proposed method are summarized in Table A.1. In the discussion that follows, we assume that d is a power of 2. In the general case, we pad 0's to set the dimension to a power of 2.

FastFood FastFood [62] exploits the property for which the Hadamard matrix combined with diagonal Gaussian matrices exhibits similar behavior to a Gaussian matrix. For the Gaussian kernel $e^{-\frac{\|x-y\|^2}{2\sigma^2}}$, FastFood uses

$$\sqrt{2} \cos\left(\frac{1}{\sigma\sqrt{d}} SHG\Pi HBx + b\right) \tag{A.4}$$

as a global feature, where H is the Hadamard matrix whose elements are -1 or 1.This matrix is a constant multiple of orthogonal matrices, and the matrix-vector product can be calculated recursively with $O(d \log d)$ time complexity.

Here, S, G, and B are diagonal matrices representing random scaling, a random Gaussian variable, and random sign flipping, respectively. Further, Π is a random permutation. Thus, the complexity needed to calculate a d-dimensional random feature is $O(d \log d)$. We independently sample this feature D/d times to obtain the D-dimensional output feature.

Random Circulant Features Yu *et al.* [107] have proposed a random Fourier features that uses the circulant matrix instead of the Hadamard matrix. For the vector $r \in \mathbb{R}^d$, the circulant matrix $\mathrm{circ}(r) \in \mathbb{R}^{d \times d}$ is defined as

$$\mathrm{circ}(r) = \begin{bmatrix} r_1 & r_d & \cdots & r_2 \\ r_2 & r_1 & \cdots & r_3 \\ \vdots & r_2 & \ddots & \vdots \\ r_d & r_{d-1} & \cdots & r_1 \end{bmatrix}. \tag{A.5}$$

We can calculate the matrix-vector product using the fast Fourier transform as $\mathrm{circ}(r)x = F^{-1}(F(r) \circ F(x))$ with $O(d \log d)$ complexity, where $\circ$ denotes the element-wise product.

With randomly sampled w, the random circulant features approach yields

$$\sqrt{2} \cos\left(\mathrm{circ}(w)Bx + b\right) \tag{A.6}$$

as a global feature. Although Yu *et al.* did not analyze the performance of this method, in experiments, the random circulant features method exhibits equivalent performance to the random Fourier features method.

Choromanski & Sindhwani [110] generalized the fast Fourier features method to construct low-variance features via graph-theoretic concept.

Structured Orthogonal Random Features Felix *et al.* [106] have proposed the use of randomly sampled orthogonal matrices as the W for a Gaussian kernel, and have also proposed structured orthogonal random features that compute matrices similar to orthogonal matrices with $O(d \log d)$ time complexity. Although structured orthogonal random features are not an unbiased estimator of the Gaussian kernel, the approximation error converges to 0 with increasing d. Further, the correlation between features is small because orthogonal matrices are used. Structured orthogonal random features are calculated as

$$\sqrt{2} \cos\left(\frac{\sqrt{d}}{\sigma} HB_1HB_2HB_3 + b\right), \tag{A.7}$$

where the B_i terms correspond to independent random sign flipping.

A.3 Alternating Circulant Random Features

As discussed in the previous section, existing fast random features methods using structured matrices depend on sign flipping to reduce the covariance between the obtained features. Thus, we cannot apply these methods to the case of random Laplace features for which the weights are sampled from $\mathbb{R}^d_+$. The random circulant features approach does not use Hadamard matrices; thus, random circulant features can be applied if random sign flipping is omitted. However, as shown in the next section, the covariance of the features becomes very large. To overcome this problem, we propose the alternating circulant random features method, in which random sign flipping is replaced with random mixture, and we analyze the approximation performance.

A.3.1 Method

In the proposed method, we employ a random column mixture of m independent circulant matrices. We sample $\{w_j^{(l)}\}_{j=1,l=1}^{j=d,l=m}$ in advance and use $\sqrt{\frac{1}{D}}e^{-Wx}$ or $\sqrt{\frac{2}{D}}\cos(Wx+b)$, with each column of W uniformly sampled from $\mathrm{circ}(w^{(l)})_{:,j}$ with l ranging from 1 to m. As each column in $\mathrm{circ}(w)$ is composed of all the elements of w, only changing the column reduces the correlation of feature elements drastically. Thus, we can rapidly calculate the proposed features, as in the case of random circulant features. We denote $s^{(l)} \in \mathbb{R}^d$ as a vector, where $s_j^{(l)} = 1$ if and only if the j-th column is sampled from $\mathrm{circ}(w^{(l)})$ and is 0 otherwise. Then, Wx can be calculated as

$$\sum_{l=1}^{m} F^{-1}\left(F\left(w^{(l)}\right) \circ F\left(s^{(l)} \circ x\right)\right). \tag{A.8}$$

The computational complexity of (A.8) is $O(mD\log d)$ and the required memory is $O(mD)$. In the experiments discussed below, we considered the case in which $m = 2$ and $m = \log_2 d$. When $m = 2$, the computational complexity is identical to that for existing fast random Fourier features methods. Further, the proposed method is an unbiased estimator of the original kernel because for each selected row, each column element is independent and identically distributed (i.i.d.) from the original distribution.

A.3.2 Analysis

In this section, we evaluate the variance of the kernel functions approximated with random Laplace features. We first evaluate the original method, which randomly

samples all the weights i.i.d. from p_1; then, we evaluate the case of random circulant features without random sign flipping. Finally, we calculate the variance for the proposed features. For two inputs x and y, we define $z = x + y \in \mathbb{R}^d$. We assume that $k(z) = \prod_{j=1}^{d} k_1(z_j)$. In this case, the corresponding distribution p can be decomposed as $p(w) = \prod_{j=1}^{d} p_1(w_j)$.

First, we evaluate the autocovariance of the approximate kernel value e^{-wz} for the original method.

Theorem A.3.1. *$Var_{w\sim p(w)}[e^{-z^t w}] = k(2z) - k(z)^2$.*

Proof. It holds that

$$E_{w\sim p(w)}\left[\left(e^{-z^t w}\right)^2\right] = E_{w\sim p(w)}[e^{-(2z)^t w}] = k(2z). \tag{A.9}$$

Thus,

$$\begin{aligned} & \mathrm{Var}_{w\sim p(w)}[e^{-z^t w}] && \text{(A.10)}\\ = \; & E_{w\sim p(w)}\left[\left(e^{-z^t w}\right)^2\right] - \left(E_{w\sim p(w)}[e^{-z^t w}]\right)^2 && \text{(A.11)}\\ = \; & k(2z) - k(z)^2. && \text{(A.12)} \end{aligned}$$

□

As there is no covariance between elements with different dimensions, the variance of the approximated kernel with output dimension D is $\frac{k(2z)-k(z)^2}{D}$.

Next, we calculate the covariance for random circulant features without random sign flipping. For the vector $v = [v_1, v_2, \cdots, v_d]^t$, we denote

$$\begin{aligned} {}^{\uparrow n}v &= [v_{n+1}, v_{n+2}, \cdots, v_d, v_1, \cdots, v_n]^t && \text{(A.13)}\\ {}^{\downarrow n}v &= {}^{\uparrow -n}v && \text{(A.14)}\\ \mathrm{rev}(v) &= [v_d, v_{d-1}, \cdots, v_2, v_1]^t. && \text{(A.15)} \end{aligned}$$

The autocovariance is identical to that for the original method. In this case, the elements for the different dimensions have non-zero covariance because the weights are shared in the different rows.

Theorem A.3.2.

$$E_{w\sim p(w)}[e^{-(Wz)_{j_1}} e^{-(Wz)_{j_2}}] = k(z + {}^{\uparrow(j_1 - j_2)}z), \tag{A.16}$$

for $1 \le j_1, j_2 \le d$.

Proof.

$$
\begin{aligned}
& E_{w\sim p(w)}\left[e^{-(Wz)_{j_1}}e^{-(Wz)_{j_2}}\right] && \text{(A.17)}\\
= \; & E_{w\sim p(w)}\left[e^{-{}^{\uparrow j_1}(\mathrm{rev}(w))^t z}e^{-{}^{\uparrow j_2}(\mathrm{rev}(w))^t z}\right] && \text{(A.18)}\\
= \; & E_{w\sim p(w)}\left[e^{-w^t \mathrm{rev}({}^{\downarrow j_1} z)}e^{-w^t \mathrm{rev}({}^{\downarrow j_2} z)}\right] && \text{(A.19)}\\
= \; & E_{w\sim p(w)}\left[e^{-w^t(\mathrm{rev}({}^{\downarrow j_1} z)+\mathrm{rev}({}^{\downarrow j_2} z))}\right] && \text{(A.20)}\\
= \; & k(\mathrm{rev}({}^{\downarrow j_1} z)+\mathrm{rev}({}^{\downarrow j_2} z)) && \text{(A.21)}\\
= \; & k({}^{\downarrow j_1} z+{}^{\downarrow j_2} z)=k(z+{}^{\uparrow(j_1-j_2)} z). && \text{(A.22)}
\end{aligned}
$$

□

Thus, the random circulant features without random sign flipping have $k(z+{}^{\uparrow(j_1-j_2)}z)-k(z)^2$ covariance. For example, when each element in z is identical, this value is the same as the autocovariance. Thus, the variance is significant.

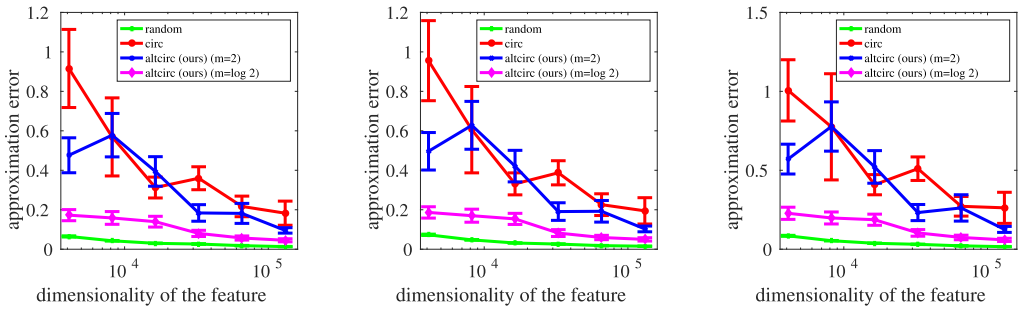
Next, we analyze the variance for the proposed method. We need to sum over all the random sampling of $\mathrm{circ}(w^{(l)})_{:,j}, l=1\text{–}m$; which makes this problem more complex than the previous cases. We first calculate the covariance between the first and d-th elements. In this case, we can calculate the exact solution.

Theorem A.3.3.

$$
\begin{aligned}
& E_{w^{(l)}\sim p(w), l_j\sim unif\{1,2,\ldots,m\}}\left[e^{-(Wz)_1}e^{-(Wz)_d}\right] && \text{(A.23)}\\
= & \left(\prod_{j=1}^{d}(k_1(z_j+z_{j+1})+(m-1)k_1(z_j)k_1(z_{j+1})) \right. && \text{(A.24)}\\
& \left. +(m-1)\prod_{j=1}^{d}(k_1(z_j+z_{j+1})-k_1(z_j)k_1(z_{j+1}))\right)/m^d, && \text{(A.25)}
\end{aligned}
$$

where we write $z_{d+1}=z_1$.

The ratio of the covariance to the autocovariance is roughly $(\frac{k_1(2z)+(m-1)k_1(z)^2}{mk_1(2z)})^d$, which exponentially converges toward 0 as d increases. The convergence speed is high if m is large. In fact, we vary k_1 as d increases; thus, the convergence speed does not exactly follow this order. However, this value is considerably smaller than that for random circulant features.



(a) CUB (b) StanfordDogs (c) Caltech256

Figure A.1: Comparison of the approximation error for the gram matrix using VGG-16 last activation.

Proof. When we denote $u^{(l)} = \mathrm{rev}(w^{(l)})$, it holds that

$$E_{w\sim p(w),l_j\sim\mathrm{unif}\{1,\dots,m\}}\left[e^{-(Wz)_1}e^{-(Wz)_d}\right] \tag{A.26}$$

$$=E_{w\sim p(w),l_j\sim\mathrm{unif}\{1,\dots,m\}}\left[e^{-\sum_{j=1}^{d}v_j^{(l_j)}z_j}e^{-\sum_{j=1}^{d}v_j^{(l_j)}z_j}\right] \tag{A.27}$$

$$=E_{w\sim p(w),l_j\sim\mathrm{unif}\{1,\dots,m\}}\left[e^{-\sum_{j=1}^{d}\left(v_j^{(l_j)}z_j+v_j^{(\uparrow 1 l_j)\uparrow 1}z_j\right)}\right] \tag{A.28}$$

$$=E_{l_j\sim\mathrm{unif}\{1,\dots,m\}}\left[\prod_{j=1}^{d}E_{w\sim p(w)}\left[e^{-\left(v_j^{(l_j)}z_j+v_j^{(l_{j+1})}z_{j+1}\right)}\right]\right]. \tag{A.29}$$

The last equality follows from the fact that $v_j^{(l)}$ corresponding to different j are independent regardless of the values of l_j. We define the matrices $C^j \in \mathbb{R}^{m\times m}$ such that $C^j_{o,p} = E_{w^{(l)}\sim p(w)}\left[e^{-\left(v_j^{(o)}z_j+v_j^{(p)}z_{j+1}\right)}\right]$. In other words, the C^j are matrices with diagonal and non-diagonal elements are $k_1(z_j + z_{j+1})$ and $k_1(z_j)k_1(z_{j+1})$, respectively. With the C^j terms, (A.29) can be expressed as

$$E_{l_j\sim\mathrm{unif}\{1,\dots,m\}}\left[\prod_{j=1}^{d}E_{w\sim p(w)}\left[e^{-\left(v_j^{(l_j)}z_j+v_j^{(l_{j+1})}z_{j+1}\right)}\right]\right] \tag{A.30}$$

$$=E_{l_j\sim\mathrm{unif}\{1,\dots,m\}}\left[\prod_{j=1}^{d}C^j_{l_j,l_{j+1}}\right] \tag{A.31}$$

$$=\frac{\mathrm{trace}(\prod_{j=1}^{d}C^j)}{m^d}, \tag{A.32}$$

Dataset	Method	$D = d$	$D = 2d$	$D = 4d$	$D = 8d$	$D = 16d$
CUB	random	11.20 ± 0.12	12.70 ± 0.11	14.06 ± 0.08	**14.81 ± 0.05**	15.12 ± 0.05
	circ	9.37 ± 0.90	12.15 ± 0.55	13.77 ± 0.35	14.57 ± 0.19	**15.36 ± 0.08**
	altcirc (ours) ($m = 2$)	10.95 ± 0.39	12.59 ± 0.14	14.03 ± 0.19	14.73 ± 0.11	15.27 ± 0.09
	altcirc (ours) ($m = \log_2 d$)	**11.26 ± 0.21**	**12.77 ± 0.11**	**14.09 ± 0.13**	14.71 ± 0.09	15.23 ± 0.07
Stanford Dogs	random	16.65 ± 0.10	19.23 ± 0.12	21.02 ± 0.05	22.23 ± 0.08	23.06 ± 0.08
	circ	12.50 ± 2.10	18.73 ± 0.89	**21.49 ± 0.28**	**22.49 ± 0.26**	**23.37 ± 0.13**
	altcirc (ours) ($m = 2$)	16.68 ± 0.79	**19.51 ± 0.29**	21.30 ± 0.22	22.18 ± 0.21	23.22 ± 0.15
	altcirc (ours) ($m = \log_2 d$)	**16.83 ± 0.20**	18.91 ± 0.21	21.22 ± 0.15	22.38 ± 0.10	23.03 ± 0.09
Caltech 256	random	**30.12 ± 0.15**	**32.72 ± 0.09**	34.50 ± 0.08	35.55 ± 0.08	36.06 ± 0.10
	circ	25.82 ± 1.88	31.42 ± 0.93	34.23 ± 0.35	**35.56 ± 0.23**	36.23 ± 0.12
	altcirc (ours) ($m = 2$)	29.28 ± 0.81	32.46 ± 0.23	34.51 ± 0.24	35.45 ± 0.23	**36.30 ± 0.16**
	altcirc (ours) ($m = \log_2 d$)	30.06 ± 0.27	32.55 ± 0.31	**34.60 ± 0.17**	**35.56 ± 0.16**	36.09 ± 0.16

Table A.2: Comparison of accuracy on image recognition datasets using Bag of Visual Words.

and the equation is derived. □

We continue to the general case. The point of the previous proof is that when we calculate the correlation by first obtaining the expectation with respect to w and then calculating the average of each l^j ranging from 1 to m, the correlation is represented as the trace of the product of d matrices that can be simultaneously diagonalized. This is because when the difference of the dimension is 1, when we bridge the dimensions that share the same $w^{(l)}$, the graph is a cyclic such as $1 \to 2 \to \cdots \to d \to 1$. When the greatest common factor between $\delta_j = j_1 - j_2$ and d is 2^g, the graph becomes an independent 2^g cyclic graph of length $d/(2^g)$ such as $1 \to 1 + \delta j \to 1 + 2\delta j \to \cdots \to 1$. The calculation for one loop is the same as the previous proof. Thus, the ratio of one loop is roughly of the order $(\frac{k_1(2z)+(m-1)k_1(z)^2}{mk_1(2z)})^{d/2^g}$. When the 2^g products are calculated, the order is $(\frac{k_1(2z)+(m-1)k_1(z)^2}{mk_1(2z)})^{d}$. Thus, the order is roughly the same in the general case.

Therefore, the covariance is small if d is sufficiently large.

A.4 Experiments

In this section, we experimentally evaluate the accuracy and computation time of the proposed method. In Section A.4.1, we evaluate the approximation error of the gram matrix. In Section A.4.2 and A.4.3, we evaluate the classification accuracy on image recognition datasets. In Section A.4.4, the computation time for feature encoding using synthesized data is compared.

Dataset	Method	$D = d$	$D = 2d$	$D = 4d$	$D = 8d$	$D = 16d$
CUB	random	**36.07 ± 0.30**	**39.48 ± 0.11**	**40.91 ± 0.12**	**42.03 ± 0.08**	**42.55 ± 0.07**
	circ	33.29 ± 2.55	35.26 ± 2.18	37.88 ± 2.03	40.35 ± 1.14	41.20 ± 0.92
	altcirc (ours) ($m = 2$)	33.76 ± 2.08	36.95 ± 1.95	39.11 ± 1.09	40.77 ± 0.89	42.10 ± 0.63
	altcirc (ours) ($m = \log_2 d$)	36.05 ± 0.68	37.50 ± 1.09	**40.91 ± 0.43**	41.86 ± 0.21	42.46 ± 0.20
	gauss	23.07 ± 0.06	23.34 ± 0.05	23.38 ± 0.06	23.39 ± 0.03	23.44 ± 0.04
	linear	22.83				
Stanford Dogs	random	78.81 ± 0.04	**79.13 ± 0.02**	**79.22 ± 0.04**	79.28 ± 0.04	79.29 ± 0.04
	circ	78.50 ± 0.30	78.93 ± 0.13	79.02 ± 0.11	79.31 ± 0.04	79.33 ± 0.03
	altcirc (ours) ($m = 2$)	78.75 ± 0.19	79.00 ± 0.12	79.21 ± 0.07	79.29 ± 0.02	**79.35 ± 0.03**
	altcirc (ours) ($m = \log_2 d$)	**78.86 ± 0.08**	79.08 ± 0.06	79.18 ± 0.05	**79.34 ± 0.04**	79.34 ± 0.02
	gauss	78.07 ± 0.03	78.03 ± 0.04	78.02 ± 0.02	78.00 ± 0.03	78.03 ± 0.01
	linear	78.12				
Caltech 256	random	**70.04 ± 0.21**	**71.52 ± 0.18**	**72.25 ± 0.21**	**72.57 ± 0.15**	**72.74 ± 0.14**
	circ	67.35 ± 1.19	69.20 ± 1.02	70.47 ± 0.96	71.84 ± 0.51	72.32 ± 0.36
	altcirc (ours) ($m = 2$)	68.54 ± 0.95	69.94 ± 0.93	71.45 ± 0.51	72.10 ± 0.36	72.69 ± 0.20
	altcirc (ours) ($m = \log_2 d$)	69.78 ± 0.35	70.73 ± 0.32	72.01 ± 0.21	72.53 ± 0.16	72.71 ± 0.18
	gauss	58.48 ± 0.20	58.72 ± 0.22	58.65 ± 0.23	58.65 ± 0.24	58.69 ± 0.24
	linear	57.90				

Table A.3: Comparison of accuracy on image recognition datasets using VGG-16 softmax output.

A.4.1 Approximation Error of Gram Matrix

We first applied the proposed method to CNN features and evaluated the approximation error of kernel values. The method was tested on three object recognition datasets (CUB-200, Stanford Dogs, and Caltech256). CUB-200 [94] is a standard fine-grained object recognition dataset that consists of 200 bird species with 60 images per class. Stanford Dogs [95] consists of approximately 20,000 images of 120 dog classes, and Caltech256 [96] consists of approximately 30,600 images of 256 object classes.

For the CNN model, we used the VGG-16 [32] network pretrained with ImageNet. We employed the l_1-normalized activation of the last fully connected layer with a dimension of 4,096. The activation was positive definite because we applied the ReLU function.

As a kernel function, we used an exponential-semigroup kernel that was reported to exhibit good performance in [67], using the kernel parameter $\beta = 0.1$. The kernel function had the form $e^{-\beta \sum_{j=1}^{d} \sqrt{x_j + y_j}}$ and the corresponding weight distribution was a Lévy distribution $p(w) = \frac{\beta}{2\sqrt{\pi}} w^{-3/2} e^{\frac{-\beta^2}{4w}}$.

We compared random Laplace features (random), random circulant features excluding random sign flipping (circ), and the proposed method (altcirc) with $m = 2$ and $\log_2 d$. Then, we varied the output feature dimension to $4,096 \times \{1, 2, 4, 8, 16, 32\}$.

We randomly sampled 2,000 data from each dataset and computed the mean and standard error of $\frac{\|K_{\text{true}} - K_{\text{approx}}\|_F}{\|K_{\text{true}}\|_F}$, where K_{true} is the true gram matrix, and

Dataset	Method	$D = d$	$D = 2d$	$D = 4d$	$D = 8d$	$D = 16d$
CUB	random	55.00 ± 0.16	56.89 ± 0.12	57.94 ± 0.07	58.43 ± 0.09	58.74 ± 0.04
	circ	54.14 ± 0.94	56.80 ± 0.36	57.68 ± 0.19	58.45 ± 0.08	58.72 ± 0.06
	altcirc (ours) ($m = 2$)	54.85 ± 0.34	56.63 ± 0.22	57.75 ± 0.16	**58.57 ± 0.07**	**58.94 ± 0.05**
	altcirc (ours) ($m = \log_2 d$)	54.92 ± 0.14	56.75 ± 0.12	**57.81 ± 0.06**	58.39 ± 0.09	58.68 ± 0.05
	gauss	**56.88 ± 0.11**	**57.60 ± 0.08**	57.75 ± 0.06	57.84 ± 0.03	57.87 ± 0.05
	linear	56.47				
Stanford Dogs	random	75.45 ± 0.06	76.59 ± 0.09	77.20 ± 0.05	77.52 ± 0.06	**77.73 ± 0.03**
	circ	75.75 ± 0.27	**76.90 ± 0.08**	**77.44 ± 0.07**	**77.65 ± 0.06**	77.72 ± 0.04
	altcirc (ours) ($m = 2$)	75.70 ± 0.13	76.75 ± 0.07	77.24 ± 0.07	77.47 ± 0.05	77.71 ± 0.05
	altcirc (ours) ($m = \log_2 d$)	75.45 ± 0.11	76.48 ± 0.07	77.20 ± 0.07	77.53 ± 0.04	77.64 ± 0.03
	gauss	76.02 ± 0.05	76.23 ± 0.06	76.37 ± 0.05	76.39 ± 0.03	76.44 ± 0.04
	linear	**77.73**				
Caltech 256	random	75.42 ± 0.17	76.48 ± 0.16	77.15 ± 0.15	**77.54 ± 0.15**	77.65 ± 0.16
	circ	73.07 ± 1.64	76.03 ± 0.57	76.81 ± 0.40	77.45 ± 0.17	77.60 ± 0.14
	altcirc (ours) ($m = 2$)	75.01 ± 0.48	76.25 ± 0.28	**77.16 ± 0.14**	77.44 ± 0.16	77.66 ± 0.16
	altcirc (ours) ($m = \log_2 d$)	75.33 ± 0.18	76.53 ± 0.15	77.11 ± 0.13	77.44 ± 0.18	**77.70 ± 0.16**
	gauss	**76.32 ± 0.20**	**76.60 ± 0.21**	76.78 ± 0.19	76.83 ± 0.18	76.85 ± 0.19
	linear	75.50				

Table A.4: Comparison of accuracy on image recognition datasets using VGG-16 last activation.

Table A.5: Computation time (second) on synthesized data.

Method	1,024	2,048	4,096	8,192	16,384
random	3.4e-3	1.5e-2	6.2e-2	2.5e-1	1.0
altcirc (ours) ($m = 2$)	**3.6e-4**	**3.7e-4**	**4.8e-4**	**9.6e-4**	**1.7e-3**
altcirc (ours) ($m = \log_2 d$)	6.3e-4	8.5e-4	1.8e-3	4.0e-3	8.6e-3

K_{approx} denotes gram matrices calculated from random features for 10 trials.

Figure A.1 shows the results. The approximation error is roughly 'random' $<$ 'altcirc ($m = \log_2 d$)' $<$ 'altcirc ($m = 2$) $<$ 'circ'. Random circulant features have a much larger error and the performance is unstable. Though the error of "random" is smaller than the error of "altcirc ($m = \log_2 d$)," "altcirc ($m = \log_2 d$)" has a small error, and its performance increases in stable fashion as D increases.

A.4.2 Semigroup Kernel on Bag of Visual Words

Next, we applied the proposed method to Bag of Visual Words features and evaluated its accuracy on image recognition datasets. As in Section A.4.1, the method was tested on CUB-200, Stanford Dogs, and Caltech256.

We used the default train/test splits for the CUB-200 and Stanford Dogs datasets. For the Caltech256 dataset, we randomly sampled 25 images per class

as training data and 30 images per class as test data.

For all datasets, we extracted dense 128-dimensional scale-invariant feature transform (SIFT) features with a step size of 2 and scales of 4, 6, 8, and 10. We used "vl_phow" implemented in VLFeat [97] for extraction and then encoded the extracted SIFT to 4,096-dimensional Bag of Visual Words features. We used 250,000 local features to learn the codebooks.

As a kernel function, we used an exponential-semigroup kernel like Section A.4.1 with the kernel parameter $\beta = 0.01$. We compared random Laplace features (random), random circulant features excluding random sign flipping (circ), and the proposed method (altcirc) with $m = 2$ and $\log_2 d$. Then, we varied the output feature dimension to $4,096 \times \{1, 2, 4, 8, 16\}$. We applied linear SVM implemented in LIBLINEAR [98] with $C = 100$ and evaluated the mean and standard error of the accuracy over 10 trials.

The results are listed in Table A.2 and indicate that random circulant features without random sign flipping exhibited poor performance with a large standard error. Thus, the random circulant features are unstable. On the other hand, the proposed method with $m = 2$ exhibited a very similar performance to the random Laplace features. Further, the proposed method exhibited comparable performance to the random Laplace features method for $m = \log_2 d$, although the standard error was slightly larger for the former. Thus, the proposed method is a good choice for histogram feature.

A.4.3 Semigroup Kernel on the CNN feature

Next, we applied the proposed method to features calculated from the CNNs. We used the VGG-16 [32] network pretrained with ImageNet. We employed the l_1-normalized activation of the last fully connected layer with a dimension of 4,096 and a 1,000-dimensional output for the softmax layer, which was embedded into a 1,024-dimensional vector with the additional dimensions filled with 0's. The activation was positive. The softmax layer output was also positive. Thus, both features satisfied the requirement for positive definiteness.

We compared the random Laplace features, random circulant features, and the proposed method with $m = 2$ and $m = \log_2 d$, corresponding to the exponential-semigroup kernel. We also evaluated the results obtained using the methods for original features (linear) and by applying random Fourier features corresponding to a Gaussian kernel (gauss) so as to verify the performance of the exponential-semigroup kernel on CNN features. As kernel parameters, we used $\beta = 0.1$ and 0.01 for the outputs of the last fully-connected and softmax layers, respectively. We used the mean of the 50-th l_2 nearest-neighbor distances of 1,000 data sampled from the training data for the Gaussian kernel, following [107]. In addition, we employed the same dataset, evaluation metric, and classifier setting used in

Section A.4.2.

Tables A.3 and A.4 show the results. Table A.3 indicates that, while the Gaussian kernel did not work well for the softmax output, the exponential-semigroup kernel contributed significantly to a performance improvement. The results indicate that the semigroup kernel was effective not only for histogram data but also for probability score of the softmax output. Furthermore, the proposed method with $m = \log_2 d$ worked as well as the random Laplace features approach. In Table A.4, when the output feature dimension was small, the linear method and the Gaussian kernel worked well. However, as the dimension grew, the approximation method for the semigroup kernel exhibited better performance. Thus, we can state that the semigroup kernel is superior to the Gaussian kernel for encoding the activation output of the CNNs when the approximation is sufficiently accurate.

These results demonstrate that the proposed method is effective for encoding CNN features.

A.4.4 Computation Time

Finally, we compared the feature-encoding computation time using synthesized data, with each element sampled from a uniform distribution ranging from 0 to 1. We varied the dimension of the input feature d to 1,024, 2,048, 4,096, 8,192 and 16,384, and set the dimension of the output feature $D = d$. We compared the computation time required to encode one input vector using an Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz. We implemented each method using Matlab with the "-singleCompThread" option.

Table A.5 shows that the overall performance roughly follows the order given in Table A.1. The proposed method requires significantly less computation time than the random Laplace features method. Although the proposed method with $m = \log_2 d$ is slower than the proposed method with $m = 2$, even with $m = \log_2 d$, it is approximately 100 times as fast as the random Laplace features method when the dimension is 16,384. Thus, the proposed method is efficient.

A.5 Discussion

In this chapter, we proposed alternating circulant random features for application to semigroup kernels. The proposed method randomly mixes the circulant random features. We analyzed the covariance of the proposed features and showed that this value is small when the dimension of the global feature is large. From experiments on image recognition datasets using histogram and CNN features, we demonstrated that the semigroup kernel is effective on a wide range of positive

definite features, and that the proposed method exhibits comparable performance to, with much lower computation time than, the original random Laplace features. The proposed method enabled us to apply the corresponding semigroup kernels very efficiently.

We have applied our method to semigroup kernels. However, the concept of mixing random features can be applied to a wider domain. Thus, we will extend this concept to a broader range of kernels in future work.

References

[1] David G Lowe. Distinctive image features from scale-invariant keypoints. *ICJV*, 60:91–110, 2004. 6

[2] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 6

[3] Zhenhua Guo, Lei Zhang, and David Zhang. A completed modeling of local binary pattern operator for texture classification. *IEEE Transactions on Image Processing*, 19(6):1657–1663, 2010. 6

[4] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 42(3):145–175, 2001. 6

[5] Yan Ke and Rahul Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *CVPR*, 2004. 6

[6] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *PAMI*, 27(10):1615–1630, 2005. 6

[7] Jean-Michel Morel and Guoshen Yu. Asift: A new framework for fully affine invariant image comparison. *SIIMS*, 2:438–469, 2009. 7

[8] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *IRE Trans Inf Theory*, 8(2):179–187, 1962. 7

[9] Risi Kondor. A novel set of rotationally and translationally invariant features for images based on the non-commutative bispectrum. *arXiv preprint cs/0701127*, 2007. 7

[10] Julien Mairal, Piotr Koniusz, Zaid Harchaoui, and Cordelia Schmid. Convolutional kernel networks. In *NIPS*, 2014. 7, 24, 26, 27, 34

[11] Fabio Anselmi, Joel Z Leibo, Lorenzo Rosasco, Jim Mutch, Andrea Tacchetti, and Tomaso Poggio. Unsupervised learning of invariant representations in hierarchical architectures. *arXiv preprint arXiv:1311.4158*, 2013. 7

[12] Stéphane Mallat. Group invariant scattering. *Comm Pure Appl Math*, 65(10):1331–1398, 2012. 7

[13] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *ECCV Workshop on SLCV*, 2004. 7, 33

[14] Jorge Sánchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. Image classification with the fisher vector: Theory and practice. *IJCV*, 105:222–245, 2013. 7, 33

[15] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010. 8

[16] David Picard and Philippe-Henri Gosselin. Efficient image signatures and similarities using tensor products of local descriptors. *CVIU*, 117:680–687, 2013. 8

[17] Hideki Nakayama, Tatsuya Harada, and Yasuo Kuniyoshi. Dense sampling low-level statistics of local features. *IEICE TRANSACTIONS on Information and Systems*, 93:1727–1736, 2010. 8

[18] Hideki Nakayama, Tatsuya Harada, and Yasuo Kuniyoshi. Global gaussian approach for scene categorization using information geometry. In *CVPR*, 2010. 8

[19] Sadeep Jayasumana, Richard Hartley, Mathieu Salzmann, Hongdong Li, and Mehrtash Harandi. Kernel methods on the riemannian manifold of symmetric positive definite matrices. In *CVPR*, 2013. 8

[20] Peihua Li and Qilong Wang. Local log-euclidean covariance matrix (l2ecm) for image representation and its applications. In *ECCV*, 2012. 8

[21] Giuseppe Serra, Costantino Grana, Marco Manfredi, and Rita Cucchiara. Covariance of covariance features for image classification. In *ICMR*, 2014. 8

[22] Tatsuya Harada and Yasuo Kuniyoshi. Graphical gaussian vector for image categorization. In *NIPS*, 2012. 8

[23] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017. 9

[24] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *NIPS*, 2013. 9

[25] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *JMLR*, 14(Feb):567–599, 2013. 9

[26] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *JMLR*, 7(Mar):551–585, 2006. 9

[27] Mark Dredze, Koby Crammer, and Fernando Pereira. Confidence-weighted linear classification. In *ICML*, 2008. 9

[28] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *IEEE*, 86(11):2278–2324, 1998. 10, 27, 48

[29] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images, 2009. 10, 27, 48

[30] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 10, 48

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 10, 34, 49

[32] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2014. 10, 43, 67, 69

[33] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 10

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 10

[35] Mircea Cimpoi, Subhransu Maji, and Andrea Vedaldi. Deep filter banks for texture recognition and segmentation. In *CVPR*, 2015. 10

[36] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*. 2010. 10, 34

[37] Benjamin Klein, Guy Lev, Gil Sadeh, and Lior Wolf. Associating neural word embeddings with deep image representations using fisher vectors. In *CVPR*, 2015. 10

[38] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013. 10

[39] Liwei Wang, Yin Li, and Svetlana Lazebnik. Learning deep structure-preserving image-text embeddings. *CVPR*, 2016. 10

[40] Mihir Jain, J Gemert, Cees GM Snoek, et al. University of amsterdam at thumos challenge 2014. 2014. 10

[41] Zhongwen Xu, Yi Yang, and Alex G Hauptmann. A discriminative cnn video representation for event detection. In *CVPR*, 2015. 10

[42] Hervé Jégou, Florent Perronnin, Matthijs Douze, Cordelia Schmid, et al. Aggregating local image descriptors into compact codes. *PAMI*, 34:1704–1716, 2012. 10, 33

[43] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, 2015. 10, 43

[44] Yang Gao, Oscar Beijbom, Ning Zhang, and Trevor Darrell. Compact bilinear pooling. In *CVPR*, 2016. 10

[45] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014. 10, 34

[46] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *COLT*, 2015. 11

[47] Jure Sokolic, Raja Giryes, Guillermo Sapiro, and Miguel RD Rodrigues. Robust large margin deep neural networks. *IEEE Transactions on Signal Processing*, 2017. 11

[48] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *ICLR*, 2017. 11

[49] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017. 11

[50] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *CVPR*, 2014. 11

[51] Noboru Murata. An integral representation of functions using three-layered networks and their approximation bounds. *Neural Networks*, 9(6):947–956, 1996. 12

[52] Kenji Fukumizu, Arthur Gretton, Bernhard Schölkopf, and Bharath K Sriperumbudur. Characteristic kernels on groups and semigroups. In *NIPS*, 2009. 12

[53] Petros Drineas and Michael W Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *JMLR*, 6(Dec):2153–2175, 2005. 13

[54] Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *NIPS*, 2001. 13

[55] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *NIPS*, 2007. 13, 57, 60

[56] Peter L Bartlett, Olivier Bousquet, and Shahar Mendelson. Local rademacher complexities. *Annals of Statistics*, 33(4):1497–1537, 2005. 13

[57] Tianbao Yang, Yu-Feng Li, Mehrdad Mahdavi, Rong Jin, and Zhi-Hua Zhou. Nyström method vs random fourier features: A theoretical and empirical comparison. In *NIPS*, 2012. 13, 17

[58] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Sampling methods for the nyström method. *JMLR*, 13(Apr):981–1006, 2012. 13

[59] Walter Rudin. *Fourier analysis on groups*. John Wiley & Sons, 2011. 14, 60

[60] Raffay Hamid, Ying Xiao, Alex Gittens, and Dennis DeCoste. Compact random feature maps. In *ICML*, 2014. 14

[61] Jiyan Yang, Vikas Sindhwani, Haim Avron, and Michael Mahoney. Quasi-monte carlo feature maps for shift-invariant kernels. In *ICML*, 2014. 14

[62] Quoc Le, Tamás Sarlós, and Alex Smola. Fastfood–approximating kernel expansions in loglinear time. In *ICML*, 2013. 14, 57, 58, 60

[63] David Lopez-Paz, Suvrit Sra, Alex Smola, Zoubin Ghahramani, and Bernhard Schölkopf. Randomized nonlinear component analysis. In *ICML*, 2014. 14

[64] Zhiyun Lu, Avner May, Kuan Liu, Alireza Bagheri Garakani, Dong Guo, Aurélien Bellet, Linxi Fan, Michael Collins, Brian Kingsbury, Michael Picheny, et al. How to scale up kernel methods to be as good as deep neural nets. *arXiv preprint arXiv:1411.4000*, 2014. 14

[65] Bo Dai, Bo Xie, Niao He, Yingyu Liang, Anant Raj, Maria-Florina F Balcan, and Le Song. Scalable kernel methods via doubly stochastic gradients. In *NIPS*, 2014. 14

[66] Bo Xie, Yingyu Liang, and Le Song. Scale up nonlinear component analysis with doubly stochastic gradients. *arXiv preprint arXiv:1504.03655*, 2015. 14

[67] Jiyan Yang, Vikas Sindhwani, Quanfu Fan, Haim Avron, and Michael W Mahoney. Random laplace feature maps for semigroup kernels on histograms. In *CVPR*, 2014. 14, 57, 60, 67

[68] Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *NIPS*, 2009. 14

[69] James Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 209(441-458):415–446, 1909. 15

[70] Corinna Cortes, Mehryar Mohri, and Ameet Talwalkar. On the impact of kernel approximation on learning accuracy. In *AISTATS*, 2010. 16

[71] Huaiyu Zhu, Christopher KI Williams, Richard Rohwer, and Michal Morciniec. Gaussian regression and optimal finite dimensional linear models. 1997. 18

[72] Christopher Williams and Matthias Seeger. The effect of the input density distribution on kernel-based classifiers. In *ICML*, 2000. 18

[73] Liefeng Bo, Kevin Lai, Xiaofeng Ren, and Dieter Fox. Object recognition with hierarchical kernel descriptors. In *CVPR*, 2011. 26, 34

[74] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Kernel descriptors for visual recognition. In *NIPS*, 2010. 26

[75] Jake Bouvrie, Lorenzo Rosasco, and Tomaso Poggio. On invariance in hierarchical models. In *NIPS*, 2009. 26

[76] Youngmin Cho and Lawrence K Saul. Large-margin classification in infinite neural networks. *Neural computation*, 22(10):2678–2697, 2010. 26

[77] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, 2011. 27, 48

[78] Marc Ranzato, Y lan Boureau, and Yann L. Cun. Sparse feature learning for deep belief networks. In *NIPS*, 2007. 33

[79] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010. 33

[80] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *ICML*, 2010. 33

[81] Piotr Koniusz, Fei Yan, and Krystian Mikolajczyk. Comparison of mid-level feature coding approaches and pooling strategies in visual concept detection. *CVIU*, 117(5):479–492, 2013. 33

[82] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. 33

[83] Mohammad Shahiduzzaman, Dengsheng Zhang, and Guojun Lu. Improved spatial pyramid matching for image classification. In *ACCV*, 2010. 34

[84] Piotr Koniusz and Krystian Mikolajczyk. Spatial coordinate coding to reduce histogram representations, dominant angle and colour pyramid match. In *ICIP*, 2011. 34

[85] Jorge Sánchez, Florent Perronnin, and TeóFilo De Campos. Modeling the spatial layout of images beyond spatial pyramids. *Pattern Recognition Letters*, 33(16):2216–2223, 2012. 34

[86] Y-Lan Boureau, Nicolas Le Roux, Francis Bach, Jean Ponce, and Yann LeCun. Ask the locals: multi-way local pooling for image recognition. In *ICCV*, 2011. 34

[87] Josip Krapac, Jakob Verbeek, and Frédéric Jurie. Modeling spatial layout with fisher vectors for image categorization. In *ICCV*, 2011. 34

[88] Ramazan Gokberk Cinbis, Jakob Verbeek, and Cordelia Schmid. Image categorization using fisher kernels of non-iid image models. In *CVPR*, 2012. 34

[89] Teofilo De Campos, Gabriela Csurka, and Florent Perronnin. Images as sets of locally weighted features. *CVIU*, 116(1):68–85, 2012. 34

[90] Naila Murray and Florent Perronnin. Generalized max pooling. In *CVPR*, 2014. 34

[91] Jiashi Feng, Bingbing Ni, Qi Tian, and Shuicheng Yan. Geometric lp-norm feature pooling for image classification. In *CVPR*, 2011. 34

[92] Tatsuya Harada, Yoshitaka Ushiku, Yuya Yamashita, and Yasuo Kuniyoshi. Discriminative spatial pyramid. In *CVPR*, 2011. 34

[93] Gaurav Sharma and Frederic Jurie. Learning discriminative spatial representation for image classification. In *BMVC*, 2011. 34

[94] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-ucsd birds 200. Technical Report CNS-TR-201, Caltech, 2010. 41, 48, 67

[95] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In *CVPR workshop on FGVC*, 2011. 41, 48, 67

[96] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007. 41, 48, 67

[97] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008. 42, 69

[98] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008. 42, 69

[99] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV*, 2011. 43

[100] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. Technical report, CRCV-TR-12-01, 2012. 43

[101] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *ICCV*, 2013. 43

[102] Barbara Caputo, Eric Hayman, and P Mallikarjuna. Class-specific material categorisation. In *ICCV*, 2005. 48

[103] Eric Hayman, Barbara Caputo, Mario Fritz, and Jan-Olof Eklundh. On the significance of real-world conditions for material classification. In *ECCV*, 2004. 48

[104] Jeffrey Pennington, X Yu Felix, and Sanjiv Kumar. Spherical random features for polynomial kernels. In *NIPS*, 2015. 57, 60

[105] Purushottam Kar and Harish Karnick. Random feature maps for dot product kernels. In *AISTATS*, 2012. 57

[106] X Yu Felix, Ananda Theertha Suresh, Krzysztof M Choromanski, Daniel N Holtmann-Rice, and Sanjiv Kumar. Orthogonal random features. In *NIPS*, 2016. 57, 58, 60, 61

[107] Felix X Yu, Sanjiv Kumar, Henry Rowley, and Shih-Fu Chang. Compact nonlinear maps and circulant extensions. *arXiv preprint arXiv:1503.03893*, 2015. 57, 58, 60, 61, 69

[108] Ninh Pham and Rasmus Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *KDD*, 2013. 57, 60

[109] Christian Berg, Jens Peter Reus Christensen, and Paul Ressel. Harmonic analysis on semigroups. 1984. 57

[110] Krzysztof Choromanski and Vikas Sindhwani. Recycling randomness with structure for sublinear time kernel expansions. In *ICML*, 2016. 61

Publications

Reviewed Conference

1. Yusuke Mukuta, Tatsuya Harada. Probabilistic Partial Canonical Correlation Analysis. In Proceedings of the 31st International Conference on Machine Learning (ICML 2014), pp.1449-1457, 2014.

2. Asako Kanezaki, Yusuke Mukuta, Tatsuya Harada. Mirror Reflection Invariant HOG descriptors for Object Detection. In IEEE International Conference on Image Processing (ICIP 2014), pp.1594-1598, 2014.

3. Yoshitaka Ushiku, Masataka Yamaguchi, Yusuke Mukuta, Tatsuya Harada. Common Subspace for Model and Similarity: Phrase Learning for Caption Generation from Images. In the 14th International Conference on Computer Vision (ICCV 2015), pp.2668-2676, December, 2015.

4. Yusuke Mukuta, Tatsuya Harada. Kernel Approximation via Empirical Orthogonal Decomposition for Unsupervised Feature Learning. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016), pp.5222-5230, June, 2016.

5. Kuniaki Saito, Yusuke Mukuta, Yoshitaka Ushiku, Tatsuya Harada. Deep Modality Invariant Adversarial Network for Shared Representation Learning. In the 16th International Conference on Computer Vision Workshop on Transferring and Adapting Source Knowledge in Computer Vision (ICCV 2017 TASK-CV), pp.2623-2629, October, 2017.

6. Yusuke Mukuta, Yoshitaka Ushiku, Tatsuya Harada. Spatial-Temporal Weighted Pyramid using Spatial Orthogonal Pooling. In the 16th International Conference on Computer Vision Workshop on Compact and Efficient Feature Representation and Learning in Computer Vision (ICCV 2017 CEFRL), pp.1041-1049, October, 2017.

7. Yusuke Mukuta, Akisato Kimura, David Adrian, Zoubin Gharamani. Weakly Supervised Collective Feature Learning from Curated Media. In the 32nd AAAI Conference on Artificial Intelligence (AAAI 2018), accepted, Febrary, 2018.

8. Yusuke Mukuta, Yoshitaka Ushiku, Tatsuya Harada. Alternating Circulant Random Features for Semigroup Kernels. In the 32nd AAAI Conference on Artificial Intelligence (AAAI 2018), accepted, Febrary, 2018.

Un-reviewed Conference

1. Atsushi Kanehira, Masatoshi Hidaka, Yusuke Mukuta, Yuichiro Tsuchiya, Tetsuaki Mano, Tatsuya Harada. MIL at ImageCLEF 2014: Scalable System for Image Annotation. CLEF Evaluation Labs and Workshop, Online Working Notes (CLEF 2014), September, 2014.

Un-reviewed Domestic Conference

1. 椋田悠介, 原田達也. 確率的偏正準相関分析. 信学技報, vol. 113, no. 286, IBISML2013-58, pp.169-176, 2013 年 11 月.

2. 椋田悠介, 牛久祥孝, 原田達也. 交代巡回ランダム特徴によるセミグループカーネルの高速な近似. 信学技報, vol. 117, no. 211, IBISML2017-14, pp. 27-34, 2017 年 9 月.

Others

1. **(Invited talk)** Yusuke Mukuta and Tatsuya Harada. Probabilistic Partial Canonical Correlation Analysis. the 18th Meeting on Image Recognition and Understanding (MIRU 2015), August, 2015.

2. **(Competition)** Masataka Yamaguchi, Qishen Ha, Katsunori Ohnishi, Masataka Yamaguchi, Yusuke Mukuta, Tatsuya Harada. Got the 3rd place, ImageNet Large Scale Visual Recognition Challenge 2015 (in conjunction with ICCV 2015), December, 2015.