รายงานฉบับสมบูรณ์

โครงการ การใช้เทคนิคการเรียนรู้ของเครื่องจักรเพื่อทำนายการมองของมนุษย์บนรูปภาพ

โดย ผู้ช่วยศาสตราจารย์ ดร. กิติ์สุชาต พสุภา

พฤศจิกายน 2559

สัญญาเลขที่ TRG5680090

รายงานวิจัยฉบับสมบูรณ์

การใช้เทคนิคการเรียนรู้ของเครื่องจักรเพื่อทำนายการมองของมนุษย์บนรูปภาพ

Predicting where Humans Look at in Images by Machine Learning Technique

โดย
ผู้ช่วยศาสตราจารย์ ดร. กิติ์สุชาต พสุภา
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

สนับสนุนโดยสำนักงานกองทุนสนับสนุนการวิจัย สำนักงานคณะกรรมการการอุดมศึกษา และ
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
(ความเห็นในรายงานนี้เป็นของผู้วิจัย สกว. และ สกอ. ไม่จำเป็นต้องเห็นด้วยเสมอไป)

กิตติกรรมประกาศ

ผู้ดำเนินการวิจัยขอขอบคุณ ศาสตราจารย์ ดร. อิสระชัย งามหรู นักวิจัยที่ปรึกษา ที่ได้กรุณาให้คำแนะนำตั้งแต่เริ่มเขียนข้อเสนอโครงการจนสามารถปิดโครงการให้เป็นไปด้วยดี

ที่สำคัญ ผู้ดำเนินโครงการวิจัยขอขอบคุณ สำนักงานคณะกรรมการการอุดมศึกษา (สกอ.) สำนักงานกองทุนสนับสนุนการวิจัย (สกว.) และสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง (สจล.) สำหรับทุนสนับสนุนโครงการวิจัยนี้

ขอขอบคุณบิดา มารดา ภรรยา และ ลูกสาวทั้งสอง (อลิส และ อลิน) ที่คอยเป็นกำลังใจให้กับข้าพเจ้าในการทำงานเสมอมา

ผู้ช่วยศาสตราจารย์ ดร. กิติ์สุชาต พสุภา
หัวหน้าโครงการวิจัย
พฤศจิกายน 2559

รหัสโครงการ:	TRG5680090
ชื่อโครงการ:	การใช้เทคนิคการเรียนรู้ของเครื่องจักรเพื่อทำนายการมองของมนุษย์บนรูปภาพ
หัวหน้าโครงการ:	ผู้ช่วยศาสตราจารย์ ดร. กิติ์สุชาต พสุภา สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
อีเมล:	kitsuchart@it.kmitl.ac.th
ระยะเวลาโครงการ:	3 มิถุนายน 2556 ถึง 2 พฤศจิกายน 2559

บทคัดย่อ

การเก็บข้อมูลการเคลื่อนไหวของตานั้นมีค่าใช้จ่ายสูงและลำบาก อีกทั้งมีโอกาสบ่อยครั้งที่จะเกิดข้อมูลสูญหาย สมมติว่าเรากำลังจะเก็บข้อมูลการเคลื่อนไหวของตาจากผู้ใช้ต่าง ๆ ที่มองที่รูปภาพชุดหนึ่ง มีโอกาสเป็นไปได้ที่เราจะไม่สามารถเก็บข้อมูลได้ทุก ๆ ผู้ใช้ในแต่ละรูปภาพในชุดนั้น (หนึ่งหรือมากกว่าหนึ่งมุมมอง (View) อาจจะไม่ได้ถูกเก็บเพื่อเป็นตัวแทนของภาพ) เราสมมติว่าความสัมพันธ์ระหว่างมุมมองต่าง ๆ สามารถเรียนรู้ได้การเก็บมุมมองทั้งหมดหรือรายการ (Item) ดังนั้นงานนี้จะเป็นการสร้างข้อมูลขึ้นมาแทนในมุมมองที่ขาดหายไปของแต่ละรายการที่ไม่สมบูรณ์ โดยสร้างจากความสัมพันธ์ที่ได้มาจากรายการที่สมบูรณ์โดยใช้คุณสมบัติพีชคณิตของเทนเซอร์ (Tensor Algebra) เราได้แสดงว่า ปัญหานี้สามารถพิจารณาให้เป็นการเรียนรู้แบบถดถอย ยิ่งไปกว่านั้น เราใช้กรอบคิดของการหาค่าที่เหมาะสมโดยใช้พื้นฐานของการเรียนรู้ที่มีมาร์จินสูงสุด (Maximum Margin Learning) มาแก้ไขปัญหา ซึ่งวิธีนี้สามารถควบคุมได้ง่าย ปัญหานี้จะคล้ายกับการเรียนรู้ที่จะทำนายการมองของมนุษย์บนรูปภาพ โดยเราได้นำเสนออัลกอริทึมใหม่ที่เรียกว่า Tensor-based Multi-View Learning (TMVL) อีกทั้งได้นำเสนอเทคนิคในการเพิ่มประสิทธิภาพของการทำนายโดยใช้ลักษณะเด่นที่ได้มาจากการแยกองค์ประกอบโดยวิธีโครเนคเกอร์ (Kronecker Decomposition) ของรูปภาพรวมกับข้อมูลการเคลื่อนไหวของตา โดยการใช้ลักษณะเด่นที่นำเสนอนี้สามารถเพิ่มประสิทธิภาพในการทำนายได้อย่างมาก วิธีที่นำเสนอนี้ได้รับการพิสูจน์แล้วว่ามีประสิทธิภาพมากขึ้นกว่าเทคนิคอื่น ๆ

คำหลัก: การเรียนรู้หลายมุมมอง; ข้อมูลสูญหาย; พีชคณิตของเทนเซอร์; การเรียนรู้ที่มีมาร์จินสูงสุด; การเคลื่อนไหวของตา; การแยกองค์ประกอบโดยวิธีโครเนคเกอร์.

Project Code: TRG5680090

Project Title: Predicting where Humans Look at in Images by Machine Learning Technique

Investigator: Asst. Prof. Dr. Kitsuchart Pasupa

King Mongkut's Institute of Technology Ladkrabang

Email address: kitsuchart@it.kmitl.ac.th

Project period: 3 June 2013 – 2 November 2016

Abstract

Eye movement data collection is very expensive and laborious. Moreover, there are usually missing values. Assuming that we are collecting eye movement data from a set of images viewed by different users, there is a possibility that we will not able to collect the data of every user from every image–one or more views may not be represented in the image. We assume that the relationships among the views can be learnt from the whole collection of views (or items). The task is then to reproduce the missing part of the incomplete items from the relationships derived from the complete items and the known part of these items. Using certain properties of tensor algebra, we showed that this problem can be formulated consistently as a regression type learning task. Furthermore, there is a maximum margin based optimisation framework in which this problem can be solved in a tractable way. This problem is similar to learning to predict where a person is looking in an image. Therefore, we proposed an algorithm called "Tensor-based Multi-View Learning" (TMVL) in this report. Furthermore, we also proposed a technique for improving prediction by introducing a new feature set obtained from Kronecker decomposition of the image fused with user's eye movement data. Using this new feature can improve prediction performance markedly. The proposed approach was proven to be more effective than two well-known saliency detection techniques.

Keyword: multi-view learning; missing data; tensor algebra; maximum margin learning; eye movements; Kronecker decomposition.

Exclusive Summary

รหัสโครงการ: TRG5680090

โครงการ: การใช้เทคนิคการเรียนรู้ของเครื่องจักรเพื่อทำนายการมองของมนุษย์บนรูปภาพ

งานวิจัยชิ้นนี้ศึกษาเกี่ยวกับการมองของมนุษย์ โดยพยายามที่จะทำนายว่าผู้ใช้นั้นจะมองตรงส่วนไหนรูปภาพต่าง ๆ ในชุดข้อมูล ซึ่งมีแรงจูงใจมาจากปัญหาของการเก็บข้อมูลการเคลื่อนไหวของตา (Eye Movement Data) นั้นมีค่าใช้จ่ายสูงและลำบาก อีกทั้งมีโอกาสบ่อยครั้งที่จะเกิดข้อมูลสูญหาย (Missing Data) ดังนั้น เราจึงพยายามที่จะอนุมานข้อมูลในส่วนของข้อมูลที่หายไป โดยที่เราสมมติว่า เรามีข้อมูลการมองที่สมบูรณ์ของผู้ใช้ทุกคน บนรูปภาพทุกรูป ซึ่งเราสามารถหาความสัมพันธ์ของพฤติกรรมการมองของผู้ใช้เหล่านี้ได้ จากนั้นเราสามารถที่จะอนุมานข้อมูลที่สูญหายได้โดยใช้ความสัมพันธ์ที่ได้มา

งานวิจัยนี้ได้นำเสนออัลกอริทึมใหม่ที่เรียกว่า Tensor-based Multi-View Learning (TMVL) ซึ่งเรียนรู้จากหลายมุมมอง (ในที่นี้คือจำนวนผู้ใช้) อีกทั้งได้นำเสนอเทคนิคในการเพิ่มประสิทธิภาพของการทำนายโดยใช้ลักษณะเด่นที่ได้มาจากการแยกองค์ประกอบโดยวิธีโครเนคเกอร์ (Kronecker Decomposition) ของรูปภาพรวมกับข้อมูลการเคลื่อนไหวของตา ซึ่งวิธีการที่นำเสนอนี้ได้ถูกนำมาผ่านการทดสอบและการพิสูจน์แล้วว่ามีประสิทธิภาพมากขึ้นกว่าเทคนิคพื้นฐานอื่น ๆ จากผลการทดลองเราได้แสดงให้เห็นว่า การสร้างโมเดลที่มีการพิจารณาการปรับตัวของผู้ใช้ (User Adaptation) นั้นอาจจะมีประโยชน์

Contents

Chapter 1

Introduction

Many researchers have paid attention to image understanding that allows computer to capture the meanings of images in the same way as humans do. One of the challenging tasks in image understanding is saliency prediction. Visual saliency is a property of locations or object in the visual world, e.g., in images. If an object is salient, it stands out from its neighbours, with a high probability of being able to draw humans' attention to it.

It is very important to learn which parts humans tend to look at in scenes or images. Saliency prediction is useful in many applications–such as graphic design, web design, and human computer interaction–because it enables designers to evaluate their visual design quality. Many methods of saliency modelling have been proposed [1, 2]. More methods can be found in a recent survey paper that covered 256 publications related to saliency object detection [3]. Saliency models can be divided into two categories: supervised and unsupervised learning based models.

Itti et al. (1988) and Harel et al. (2006) investigated bottom-up visual saliency (i.e., low level image feature that does not involve supervised information) [1, 2]; unfortunately, human gazes do not usually match the map [4] because they are highly influenced by any image related tasks. If users are requested to view images without having been given a particular task, their gazes will be automatically directed by low-level image feature. In the case that users have been given a clear and specific task, their eye movements will be controlled by the content of the image. Consequently, top-down visual features are the features that should be considered [5]. Another approach is supervised learning based

saliency model. It utilises eye movement data which are mapped with image features [6, 7, 8, 9].

In order to learn where humans tend to look at in images, an eye tracker is required to collect eye movement data. In real-world scenarios, eye movement data collection is tedious, laborious, and expensive. Moreover, data loss is inevitable as (i) an eye tracker may temporarily lose track of a subject because he or she is moving during the experiment, and (ii) a subject may fail to respond to all of the tasks. Consequently, we aimed to estimate missing eye movement data from the available data on the same task. It is similar to learning to predict where users tend to look at based on their previous eye movement data on other images and on other available users' eye movement data on the considered image. This leads to the learning scenario introduced in this report. It is built with a general assumption that multiple views of a problem are available. It is not always possible to observe all of the views in a realistic scenario; therefore, this problem can be cast as a multi-view learning problem with missing data. In this scenario, we assume that initially there is a subset of training samples in which a complete set of views can be observed, but later on probably only some random subsets of views can be collected.

The goal of the learning task is to estimate the values of missing views in each sample. This scenario can occur in a real experiment. This type of problem is a generalisation of classical supervised learning problems such as regression. Face recognition is one of the applications that can be considered under this framework (when some parts or views of the faces are unknown because of occlusion, for example). In developing the learning framework, we made two mild assumptions: (i) there is a reasonable large number of observations (samples) where all of the views are known, therefore, a learning procedure can be realised; and (ii) from incomplete observations, at least one view is available. We made no assumption about the distribution of the missing views, but any prior knowledge about the missing data can be exploited to improve the estimation of their values.

In this report, we introduced a formulation that can be considered as a generalised regression problem in which missing values are estimated from available views and their relationships are extracted from training samples. Assuming that the missing views of a sample are output y and the known parts are input x, then we have $y \Leftarrow Wx$, where W is a linear operator that learns from complete data and describes the relationships be-

tween different views. The difficulty of this kind of regression arises from the fact that the output and input may vary among the sample items. We proposed a "Tensor-based Multi-view Learning (TMVL)" algorithm to handle the problem of incomplete view. Providing a tractable learning algorithm, TMVL is based on properties of tensor algebra and maximum margin-based optimisation framework. Tensor decomposition has already been used in some missing data problems, e.g., [10, 11], but their settings are different from ours. Liu et al. (2013) investigated a low-rank tensor technique based on tensor-trace norm minimisation problem in image reconstruction [10], while Chen and Grauman (2014) proposed a probabilistic tensor model for inferring human appearance from unseen viewpoints [11]. In this report, we show that our proposed method can estimate missing eye movements, which can be exploited for predicting where humans are likely to look at in images.

We also propose a novel approach for fusing eye movement information with image features in order to enhance prediction performance. There are many pieces of evidence suggesting that fusing low-level image features with eye movement improves prediction accuracy [12, 13, 14, 15]. Here, we employed factors derived from Kronecker decomposition of image fused with eye movement data to represent each view in TMVL.

The outline of the report is as follows: Chapter 2 describes all methods proposed in this work, including TMVL algorithm, and tensor decomposition. Chapter 3, shows the performance results of our proposed methods on a real-world dataset. Finally, the conclusion of our study is presented in Chapter 4.

Chapter 2

Methodology

This section explains the methods proposed in this work: Subsection 2.1 describes the TMVL algorithm, its algebraic framework, and the corresponding optimisation problem. Subsection 2.2 explains the procedure for decomposing images as tensors, followed by an approach for combining images with eye movement data in subsection 2.3. A model-training process framework, including data processing pipeline for two sets of features, is visually summarised in Figure 2.1.

2.1 Tensor-based Multi-view Learning Algorithm

As previously mentioned, we had a set of complete views of our samples that we used as training set, and a test set in which the views were not complete (missing randomly). We aimed to fill in the missing views for each sample. An example of multi-view learning problem with missing data is shown in Figure 2.2.

2.1.1 Algebraic Framework

Let us denote $\mathcal{R} = \{1, \ldots, n_R\}$ as the set of indices of the views considered. In our model, each of these views has a corresponding linear vector space $\mathcal{Z}_r,\ r \in \mathcal{R}$ over real numbers. The dimensions of these spaces are denoted by $\text{Dim}(\mathcal{Z}_r) = d_r,\ r \in \mathcal{R}$. The set $\mathcal{J}_R = \{j_1, \ldots, j_{n_R}\}$ comprises the indices of the samples within each of the spaces corresponding to the views, enumerating the components of the vectors chosen from the

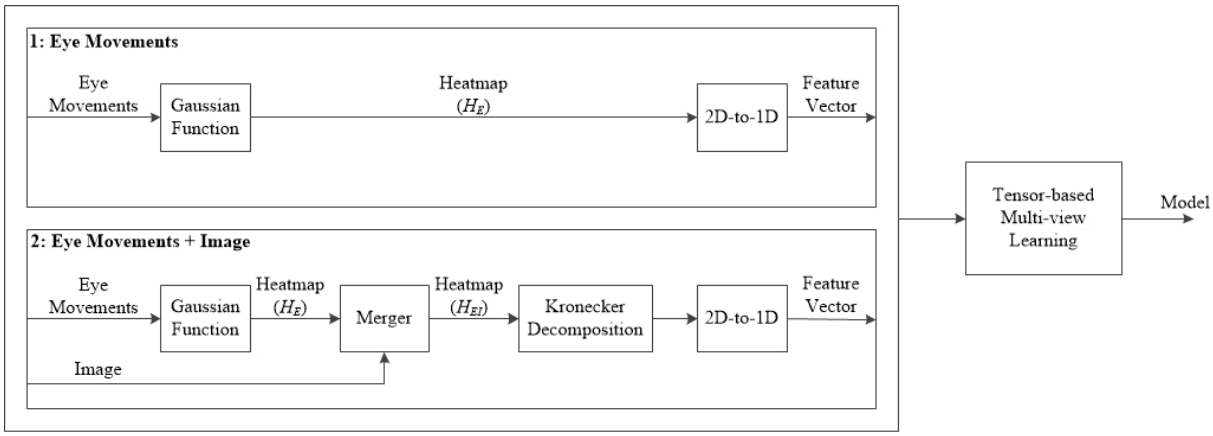


Figure 2.1: Main components and data flow in this work. There are two sets of features which are eye movement information only and image fused with eye movement information.

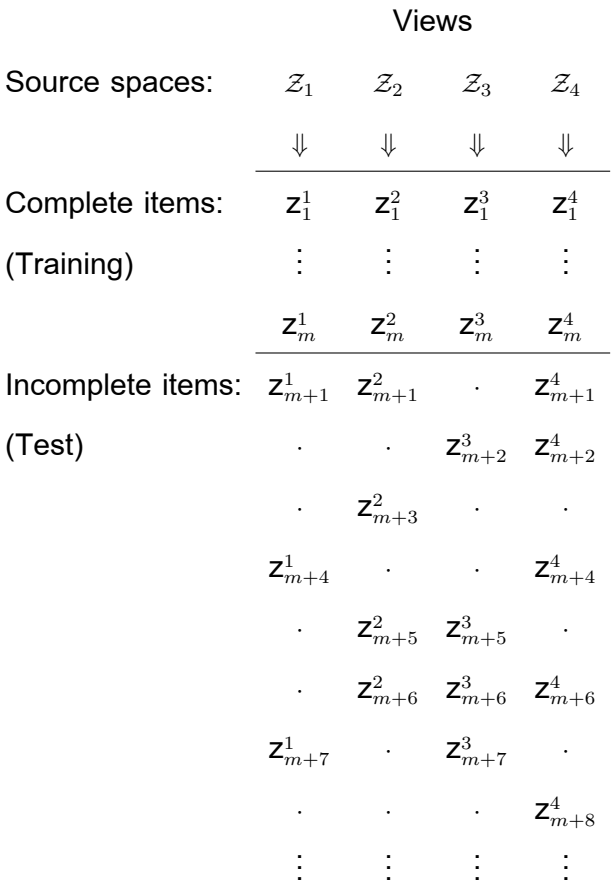


Figure 2.2: Graphical representation of the multi-view learning framework for a four-view learning problem.

space corresponding to the views. The range of these indices is equal to the number of dimensions of the corresponding space.

A sample is chosen out of direct products of these spaces, and each sample item consists of as many vectors as the number of views,

$$\begin{array}{lccccc} & & \text{Views:} & & & \\ \text{Linear vector spaces:} & \mathcal{Z}_1 & \dots & \mathcal{Z}_{n_R} & & \\ & \Downarrow & \dots & \Downarrow & & \\ \text{Sample:} & \mathsf{z}_i^1 & \dots & \mathsf{z}_i^{n_R} & i = 1, \dots, m. \end{array}$$

The product space of the views is given by the tensor product of the spaces, $\mathcal{Z} = \bigotimes_{r \in \mathcal{R}} \mathcal{Z}_r$. This construction forms the algebraic framework of our solution, see [16, 17] and the references therein for more details.

If we are given two tensor products of vectors then the following contraction operator $[.,.]$ can be defined over them as

$$[\bigotimes_{q \in \mathcal{Q}} \mathsf{u}^q, \bigotimes_{r \in \mathcal{R}} \mathsf{v}^r] = \prod_{q \in \mathcal{Q} \cap \mathcal{R}} \langle \mathsf{u}^q, \mathsf{v}^q \rangle \bigotimes_{q \in \mathcal{Q} \setminus \mathcal{R}} \mathsf{u}^q \bigotimes_{r \in \mathcal{R} \setminus \mathcal{Q}} \mathsf{v}^r, \tag{2.1}$$

where the inner product is computed for all common indices. When the two index sets are coincident then the following well known identity can be used to unfold the inner products of the tensor products as

$$\langle \bigotimes_{q \in \mathcal{Q}} \mathsf{z}_i^r, \bigotimes_{q \in \mathcal{Q}} \mathsf{z}_j^q \rangle = \prod_{q \in \mathcal{Q}} \langle \mathsf{z}_i^q, \mathsf{z}_j^q \rangle. \tag{2.2}$$

This identity states that the inner product of tensor products of vectors is equal to the product of the inner product of these vectors.

This interpretation of the indices is compatible with the notations commonly used in tensor algebra, namely, with the so-called “Einstein summation convention”. The symbol of summation $\sum$ is omitted and summation has to be carried out over all indices which are denoted by the same symbol. Since we use this strategy to denote views and algorithmic iterations which are not tensor indices, we choose to handle summations with special care by making them explicit via the contraction operator $[.,.]$. Furthermore, we assume an orthogonal representation of the indices, hence in turn, there is no need to make distinction between covariant and contravariant indices.

In the learning problem, we use a linear operator, a tensor, that is an element of the

dual space of $\mathcal{Z}$, the space of linear functionals defined on $\mathcal{Z}$, namely

$$\mathsf{W} \in \mathcal{Z}^*,\ \mathsf{W} = [W_{\mathcal{J}_R}] = [W_{j_1,\ldots,j_{n_R}}],$$

where $\mathcal{Z}^*$ denotes the dual space of all possible linear functionals defined on $\mathcal{Z}$.

We can write up Frobenius type inner products between the linear operator W and the tensor product of vectors of the views as

$$\langle \mathsf{W}, \bigotimes_{r\in\mathcal{R}} \mathsf{z}_i^r \rangle_F = \sum_{j_1,\ldots,j_{n_R}} W_{j_1,\ldots,j_{n_R}} \prod_{r\in\mathcal{R}} \mathsf{z}_{ij_r}^r. \tag{2.3}$$

In similar fashion, we can compute the Frobenius norm of W by

$$\|\mathsf{W}\|_F = \big(\sum_{j_1,\ldots,j_{n_R}} W_{j_1,\ldots,j_{n_R}}^2\big)^{\frac{1}{2}}. \tag{2.4}$$

In the next step, the set of views is partitioned into two arbitrary parts,

$$\mathcal{R}_X \subset \mathcal{R},\ \mathcal{R}_Y = \mathcal{R} \setminus \mathcal{R}_X,$$

we term the views occurring in $\mathcal{R}_X$ as inputs, and the views in $\mathcal{R}_Y$ can be handled as outputs. Corresponding to either $\mathcal{R}_X$ or $\mathcal{R}_Y$, the set of indices belonging to each view has to be split apart,

$$\mathcal{J}_X \subset \mathcal{J}_R,\ \mathcal{J}_X = \{j_r,\ r \in \mathcal{R}_X\},\ \mathcal{J}_Y = \mathcal{J}_R \setminus \mathcal{J}_X.$$

Fixing a partition, a contraction of W can be defined as

$$W_{J_Y} = W_{J_R \setminus J_X} = \mathsf{W} \bigotimes_{r\in\mathcal{R}_X} \mathsf{z}_i^r \stackrel{\text{def}}{=} \sum_{j_r\in\mathcal{J}_X} W_{J_R} \prod_{r\in\mathcal{R}_X} \mathsf{z}_{ij_r}^r, \tag{2.5}$$

where the components of W are summed only over the input views.

Consequently, the relationship between the inputs and the outputs can be described by the following inner product

$$\langle \bigotimes_{s\in\mathcal{R}_Y} \mathsf{z}_i^s, \mathsf{W} \bigotimes_{r\in\mathcal{R}_X} \mathsf{z}_i^r \rangle_F \stackrel{\text{def}}{=} \sum_{J_Y} \prod_{s\in\mathcal{R}_Y} \mathsf{z}_{ij_s}^s \sum_{J_X} W_{J_R} \prod_{r\in\mathcal{R}_X} \mathsf{z}_{ij_r}^r. \tag{2.6}$$

It provides a similarity measure between the outputs and the projection of the inputs by the linear operator W. If the norm of W is fixed, then this inner product takes a greater value if the angle between the direction of the outputs and the projection of the inputs is smaller, thus the correlation between them is greater. If both the inputs and the outputs are normalised to the same length then this similarity measure implies small distance as well.

Based on these definitions, we can derive a simple but fundamental Lemma:

Lemma 1. For all partitions $\mathcal{R}_X, \mathcal{R}_Y$ of $\mathcal{R}$ the inner products

$$\langle \bigotimes_{s\in\mathcal{R}_Y} \mathsf{z}_i^s, \mathsf{W} \bigotimes_{r\in\mathcal{R}_X} \mathsf{z}_i^r \rangle_F$$

have the same value, namely

$$\langle \mathsf{W}, \bigotimes_{r\in\mathcal{R}} \mathsf{z}_i^r \rangle_F.$$

Proof. We need to unfold only the corresponding definitions of the inner products that give the next chain of equalities

$$\begin{aligned}\langle \bigotimes_{s\in\mathcal{R}_Y} \mathsf{z}_i^s, \mathsf{W} \bigotimes_{r\in\mathcal{R}_X} \mathsf{z}_i^r \rangle_F &= \sum_{J_Y} \prod_{s\in\mathcal{R}_Y} \mathsf{z}_{ij_s}^s \sum_{J_X} W_{J_R} \prod_{r\in\mathcal{R}_X} \mathsf{z}_{ij_r}^r \\ &= \sum_{j_1,\ldots,j_{n_R}} W_{j_1,\ldots,j_{n_R}} \prod_{r\in\mathcal{R}} \mathsf{z}_{ij_r}^r \\ &= \langle \mathsf{W}, \bigotimes_{r\in\mathcal{R}} \mathsf{z}_i^r \rangle_F.\end{aligned}$$

Q.E.D.

This Lemma shows that the value of the inner product of the tensor products is invariant on the partitioning of the views into inputs and outputs.

2.1.2 The Optimisation Problem

To force a high similarity between the projected inputs and the outputs taken out of a fixed partition of the views, a "Support Vector Machine"-style, maximum-margin-based optimisation problem is formulated for the regression task. Please note the earlier application of the framework [18, 19]:

$$\begin{aligned}\text{min} \quad & \tfrac{1}{2}\|\mathsf{W}\|_F^2 + C\sum_{i=1}^m \xi_i \\ \text{w.r.t.} \quad & \mathsf{W} \text{ tensor} \in \mathcal{Z}^*,\ \boldsymbol{\xi} \in \mathbb{R}^m, \\ \text{s.t.} \quad & \langle \underbrace{\bigotimes_{s\in\mathcal{R}_Y} \mathsf{z}_i^s}_{\text{Outputs}}, \mathsf{W} \underbrace{\bigotimes_{r\in\mathcal{R}_X} \mathsf{z}_i^r}_{\text{Inputs}} \rangle_F \geq 1 - \xi_i, \\ & \xi_i \geq 0,\ i = 1, \ldots, m,\end{aligned} \tag{2.7}$$

where $C > 0$ is penalty constant.

The form is similar to the Support Vector Machine case with two notable exceptions: (i) the outputs are no longer binary labels, $\{-1, +1\}$, but vectors of an arbitrary linear vector space, and (ii) the normal vector of the separating hyperplane is reinterpreted as a linear operator projecting the inputs into the space of the outputs.

The regularisation term in the objective function forces the projections of the inputs and the outputs to be similar with respect to their inner products. When the inputs and the outputs are normalised, they live on a sphere in both corresponding spaces, hence we solve the problem by using the structure of Spherical rather than Euclidean geometry.

Based on Lemma 1, we state the next theorem:

Theorem 2. For all partitions, $\mathcal{R}_X, \mathcal{R}_Y$ of $\mathcal{R}$ the optimisation problem (2.7) is equivalent to the following one:

$$\begin{aligned} \text{min} \quad & \tfrac{1}{2}\|\mathsf{W}\|_F^2 + C\sum_{i=1}^m \xi_i \\ \text{w.r.t.} \quad & \mathsf{W} \text{ tensor} \in \mathcal{Z}^*,\ \boldsymbol{\xi} \in \mathbb{R}^m, \\ \text{s.t.} \quad & \langle \mathsf{W}, \bigotimes_{r\in\mathcal{R}} \mathsf{z}_i^r \rangle_F \geq 1 - \xi_i,\ i = 1, \ldots, m, \\ & \xi_i \geq 0,\ i = 1, \ldots, m. \end{aligned} \tag{2.8}$$

This equivalence holds true if the inputs and the outputs are partitioned independently for every sample item.

Proof. We can reformulate the constraints by following Lemma 1 that proves the statement. Q.E.D.

This fact guarantees that the linear operator W has a universal property that it is independent of the way how the views are grouped into inputs and outputs, thus it consistently characterises the underlying multi-view learning problem.

The seemingly complex problem represented by (2.8) can be solved via a simple Lagrangian dual:

$$\begin{aligned} \text{min} \quad & \tfrac{1}{2}\boldsymbol{\alpha}'\big(\mathsf{K}_1 \bullet \cdots \bullet \mathsf{K}_{n_R}\big)\boldsymbol{\alpha} - \mathsf{1}'\boldsymbol{\alpha} \\ \text{w.r.t.} \quad & \boldsymbol{\alpha} \in \mathbb{R}^m \\ \text{s.t.} \quad & \mathsf{0} \leq \boldsymbol{\alpha} \leq C\mathsf{1}, \end{aligned} \tag{2.9}$$

where

$$(\mathsf{K}_r)_{ij} \quad = \langle \mathsf{z}_i^r, \mathsf{z}_j^r \rangle,\ r \in \mathcal{R},\ i, j \in \{1, \ldots, m\} \tag{2.10}$$

are kernels corresponding to each of the views. In the formulation of the Lagrangian dual, we exploited the identity given by (2.2).

The $\bullet$ operator expresses the element-wise, Hadamard, product of matrices. This dual can be solved in a straightforward way for very large scale applications[1]. After the dual

[1]The website of the authors provides an open source implementation to this problem.

variables were computed, the optimum solution for the universal linear operator becomes

$$\mathsf{W} = \sum_{i=1}^{m} \alpha_i \bigotimes_{r\in\mathcal{R}} \mathsf{z}_i^r. \tag{2.11}$$

In the test phase, known and unknown views are considered as inputs and outputs, respectively. The output can be estimated in the following way:

$$\begin{aligned}\left(\bigotimes_{s\in\mathcal{R}_y} \mathsf{z}^s\right) \sim \mathsf{W} \bigotimes_{r\in\mathcal{R}_X} \mathsf{z}^r &= \sum_{i=1}^{m} \alpha_i [\bigotimes_{r\in\mathcal{R}} \mathsf{z}_i^r, \bigotimes_{r\in\mathcal{R}_X} \mathsf{z}^r] \\ &= \sum_{i=1}^{m} \alpha_i \prod_{r\in\mathcal{R}_X} \langle \mathsf{z}_i^r, \mathsf{z}^r \rangle \bigotimes_{s\in\mathcal{R}_Y} \mathsf{z}_i^s \\ &= \sum_{i=1}^{m} \beta_i \bigotimes_{s\in\mathcal{R}_Y} \mathsf{z}_i^s,\end{aligned} \tag{2.12}$$

where

$$\beta_i = \alpha_i \prod_{r\in\mathcal{R}_X} \langle \mathsf{z}_i^r, \mathsf{z}^r \rangle, \ i = 1\ldots, m. \tag{2.13}$$

Thus, the prediction is a linear combination of the corresponding known outputs.

2.1.3 Computational Complexity of the Proposed Method

For estimation of the computational complexity of the problem presented in (2.9), one can recognise that (2.9) is equivalent to the dual problem of an unbiased Support Vector Machine (SVM). Consequently, the problem in (2.9) can be solved by applying the same type of methods that should have the same complexity depending on the sparsity of the included kernels, see a discussion about this in [20]. The basic task of the SVM is to separate two classes of output data by a hyperplane. The settings of the SVM contain a sample $\{y_i, \mathsf{x}_i\},\ y_i \in \{-1, +1\},\ \mathsf{x}_i \in \mathcal{X}(= \mathbb{R}^n)$ and the separating hyperplane which is defined by its normal vector w. Furthermore, input vectors might be embedded into a feature space, $\mathcal{H}_X$, via a function $\phi : \mathcal{X} \to \mathcal{H}_X$, where it is assumed that $\mathcal{H}_X$ is a Hilbert space.

The corresponding primal optimisation problem of the SVM is formulated as

$$\begin{aligned} \text{min} \quad & \tfrac{1}{2}\|\mathsf{w}\|_2^2 + C\mathsf{1}'\boldsymbol{\xi} \\ \text{w.r.t.} \quad & \mathsf{w} \in \mathbb{R}^n, \boldsymbol{\xi} \in \mathbb{R}^m, \\ \text{s.t.} \quad & y_i \mathsf{w}'\phi x_i \geq \mathsf{1} - \xi_i, \\ & \boldsymbol{\xi} \geq \mathsf{0}, i = 1, \ldots, m, \end{aligned} \tag{2.14}$$

while the dual problem of the SVM has the form (see for example in [21]),

$$\begin{aligned} \text{min} \quad & \tfrac{1}{2}\boldsymbol{\alpha}'(\mathsf{y}\mathsf{y}' \bullet \mathsf{K}_X)\boldsymbol{\alpha} - \mathsf{1}'\boldsymbol{\alpha} \\ \text{w.r.t.} \quad & \boldsymbol{\alpha} \in \mathbb{R}^m, \\ \text{s.t.} \quad & \mathsf{0} \leq \boldsymbol{\alpha} \leq C\mathsf{1}. \end{aligned} \tag{2.15}$$

After introducing the notation $\mathsf{K} = (\mathsf{yy}' \bullet \mathsf{K}_X)$ in the SVM dual, and similarly $\mathsf{K} = (\mathsf{K}_1 \bullet \cdots \bullet \mathsf{K}_{n_R})$ in the dual of the multi-view learning problem (2.9), we arrive at the same optimisation problem, thus the computational complexity of the proposed learning problem is the same as the complexity of the SVM.

2.1.4 Non-linear Relations

Another consequence of the optimisation problem of the proposed learning method being equivalent to that of the SVM is that the kernel trick can be applied here as well (see [21] for more details). Since both the dual problem (2.9) and the prediction (2.12) contain all input feature vectors as components of the corresponding inner products, a knowledge of the positive semi-definite kernel matrix suffices for carrying out computations in the training and in the prediction as well. More concretely, let all input features–the representation of the image parts in this application–be implicitly embedded into a feature space by these given functions,

$$\phi_r : \mathcal{Z}_r \to \mathcal{H}_r, \ r \in \mathcal{R}_X, \tag{2.16}$$

where $\mathcal{H}_r$ is a Hilbert space for each r. Then we can write up the elements of the kernels as,

$$(\mathsf{K}_\mathsf{r})_{ij} = \langle \phi_r(\mathsf{z}_i^r, \phi_r(\mathsf{z}_j^r) \rangle \,, \ r \in \mathcal{R}_X, \ i,j = \{1,\ldots,m\}. \tag{2.17}$$

For example, Gaussian or Polynomial kernels can be chosen on top of the available linear features.

2.2 Decomposing Images as Tensors

Images expressed by matrices can be represented as a product of the factors computed by Kronecker decomposition. This Kronecker decomposition, after a reordering of the elements of the image matrix, can be carried out by singular value decomposition (SVD). That kind of transformation of images can reveal the structure of the images, e.g., edges, and corners, and can yield a high level compression of the image matrices. In the case of colour images, the representing matrices can be replaced with tensors in order to capture the third dimension of the colours.

2.2.1 Kronecker decomposition of matrices

Let us consider a real 2-dimensional (2D) image decomposition of which we can expect the points nearby within continuous 2D blocks to relate stronger to each other than to the points in the 1-dimensional rows or columns. A question is, “can the SVD decomposition provide 2D blocks instead of vectors?” To achieve this end, a Kronecker product of matrices is introduced.

The Kronecker product of a matrix X can be expressed as

$$\mathsf{X} = \mathsf{A} \otimes \mathsf{B} = \begin{bmatrix} A_{1,1}\mathsf{B} & A_{1,2}\mathsf{B} & \cdots & A_{1,n_A}\mathsf{B} \\ A_{2,1}\mathsf{B} & A_{2,2}\mathsf{B} & \cdots & A_{2,n_A}\mathsf{B} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m_A,1}\mathsf{B} & A_{m_A,2}\mathsf{B} & \cdots & A_{m_A,n_A}\mathsf{B} \end{bmatrix} \tag{2.18}$$

where $\mathsf{A} \in \mathbb{R}^{m_A \times n_A}$, $\mathsf{B} \in \mathbb{R}^{m_B \times n_B}$, $m_X = m_A \times m_B$, and $n_X = n_A \times n_B$.

In the Kronecker decomposition, the second component, B, might be interpreted as a 2D filter of the image represented by the matrix X. We can try to find a sequence of filters by using the procedure presented in Algorithm 1.

Algorithm 1 Calculate the Kronecker decomposition of a matrix

---

Require: matrix X, number of iteration n,

Require: size of $\mathsf{A} \in \mathbb{R}^{m_A \times n_A}$, size of $\mathsf{B} \in \mathbb{R}^{m_B \times n_B}$

Ensure: $(\mathsf{A}^{(1)}, \mathsf{B}^{(1)}), \ldots, (\mathsf{A}^{(n)}, \mathsf{B}^{(n)})$

1: $\mathsf{X}^{(1)} = \mathsf{X}$

2: for $k = 1$ to n do

3: $\mathsf{A}^{(k)}, \mathsf{B}^{(k)} = \arg\min_{\mathsf{A},\mathsf{B}} \|\mathsf{X}^{(k)} - \mathsf{A} \otimes \mathsf{B}\|^2_{Frobenius}$

4: $\mathsf{X}^{(k)} = \mathsf{X}^{(k)} - \mathsf{A}^{(k)} \otimes \mathsf{B}^{(k)}$, ## deflation

5: end for

The question can be stated now as, “if X is given, how can we compute the optimum solution A and B for the problem $\min_{\mathsf{A},\mathsf{B}} \|\mathsf{X}^{(k)} - \mathsf{A} \otimes \mathsf{B}\|^2_{Frobenius}$?

2.2.2 Kronecker Decomposition as SVD

We can make an important observation that the algorithm solving the tensor decomposition problem does not depend directly on the order of the elements of the matrix, thus a permutation of the indexes, i.e. reordering of the columns and/or the rows, preserves the same solution. Based on this observation, the Kronecker decomposition can be computed via the SVD (see [22] for more details). An example that illuminates how the reordering of the matrix X can solve the Kronecker decomposition problem is demonstrated here:

The matrices in the Kronecker product

$$\underset{\mathbf{X}}{\begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} \\ x_{21} & x_{22} & x_{23} & x_{24} & x_{25} & x_{26} \\ x_{31} & x_{32} & x_{33} & x_{34} & x_{35} & x_{36} \\ x_{41} & x_{42} & x_{43} & x_{44} & x_{45} & x_{46} \\ x_{51} & x_{52} & x_{53} & x_{54} & x_{55} & x_{56} \\ x_{61} & x_{62} & x_{63} & x_{64} & x_{65} & x_{66} \end{bmatrix}} = \underset{\mathbf{A}\otimes\mathbf{B}}{\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \otimes \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}},$$

can be reordered into

$$\underset{\tilde{\mathbf{X}}=\tilde{\mathbf{A}}\otimes\tilde{\mathbf{B}}}{\begin{bmatrix} x_{11} & x_{13} & x_{15} & x_{31} & x_{33} & x_{35} & x_{51} & x_{53} & x_{55} \\ x_{12} & x_{14} & x_{16} & x_{32} & x_{34} & x_{36} & x_{52} & x_{54} & x_{56} \\ x_{21} & x_{23} & x_{25} & x_{41} & x_{43} & x_{45} & x_{61} & x_{63} & x_{65} \\ x_{22} & x_{24} & x_{26} & x_{42} & x_{44} & x_{46} & x_{62} & x_{64} & x_{66} \end{bmatrix}},$$

$$= \begin{bmatrix} b_{11} \\ b_{12} \\ b_{21} \\ b_{22} \end{bmatrix} \otimes \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{21} & a_{22} & a_{23} & a_{31} & a_{32} & a_{33} \end{bmatrix}$$

where the blocks of X and the matrices A and B are vectorised in a row-wise order.

We recognise that $\tilde{\mathbf{X}} = \tilde{\mathbf{A}} \otimes \tilde{\mathbf{B}}$ can be interpreted as the first step in the SVD algorithm where we might apply substitutions $\sqrt{s}\mathbf{u} = \tilde{\mathbf{A}}$ and $\sqrt{s}\mathbf{v} = \tilde{\mathbf{B}}$. The proof that this reordering provides a correct solution to the Kronecker decomposition can be found in [22].

The main steps of the Kronecker decomposition can be summarised as follows:

1. Reorder (reshape) the matrix,
2. Compute the SVD decomposition,
3. Compute the approximation of $\tilde{X}$ by $\tilde{A} \otimes \tilde{B}$
4. Invert the reordering.

This kind of Kronecker decomposition is often referred as the Nearest Orthogonal Kronecker Product as well [22].

We can extend this procedure to higher order tensors represented by higher order arrays (see the review paper of [23]). We can apply this method on the following objects:

- colour images, where three matrices express the RGB layers, a tensor of order 3, e.g. $1024 \times 1024 \times 3$,
- video stream of grey-scale images, where the third dimension is time,
- video stream of colour images, where the third dimension is colour, and the fourth dimension is time.

The Kronecker decomposition presented above can be extended further to include more than two factors (more details, alternative approaches and applications can be found in [23]). The Kronecker decomposition algorithm, as shown in Algorithm 1), implements a non-linear polynomial approximation of the target matrix or a higher order tensor. The degree of the applied polynomial is equal to the number of included factors.

2.2.3 A Set Theoretic Approach to Reordering

A matrix representation as an ordered collection of elements can be reinterpreted by using the language of set theory. Let X be a matrix with size $m \times n$. For the sake of simplicity, we assume that the elements $[X_{ij}]$ of X are real numbers. The structure of the matrix X can be described as a set $\mathscr{X}$ of the elements $\{X_{ij}\}$ with cardinality mn on which two equivalence relations, $\mathscr{R}_r$ and $\mathscr{R}_c$, are imposed, one based on the rows and the other based on the columns, i.e. two elements are equivalent with respect to $\mathscr{R}_r$ if they are in the same row, and equivalent with respect to $\mathscr{R}_c$ if they are in the same column.

Now, the reordering of the matrix elements $\mathcal{X}$ is expressible by imposing another kind of equivalence relations that classify the elements into different classes. These equivalence relations have to satisfy the following rules: all of the equivalence classes within the relation $\mathscr{R}$ have to have the same cardinality. The equal cardinality rule implies that the product of the number of the equivalence classes $N_{\mathscr{R}}$ and the common cardinality of the classes, $\text{card}_{\mathscr{R}}$, is equal to the cardinality of $\mathcal{X}$, namely to mn. Then, the reordering of the matrix elements can be carried out by sorting the equivalence classes into columns and rows of a new matrix.

Note that the equivalence relation $\mathcal{R}$ can be decomposed further into a series of equivalence relations $(\mathcal{R}_1, \ldots, \mathcal{R}_N)$. This decomposition can be realised in a recursive way.

1. Let the set of classes of $\mathcal{R}_1$ be given by $\mathcal{C}(\mathscr{R})_1, \ldots, \mathcal{C}(\mathscr{R})_{N_{R_1}}$, with common cardinality card_{R_1}.

2. For every class of relations, $\mathscr{R}_k$ apply the same type of equivalence relations, $\mathscr{R}_{k+1}$. Since the cardinality of the classes of $\mathscr{R}_k$ is the same, this relation can be applied uniformly. Consequently, the cardinality of the classes and the number of the classes in $\mathscr{R}_{k+1}$ are the same for all classes of $\mathscr{R}_k$.

By reversing the recursive classification of the elements, we can build a tensor T of order $N+1$ by starting on the classes of the $\mathscr{R}_N$.

Since the internal structure of the matrix X is not directly exploited, only the set of its elements are classified. The reordering procedure described above can be applied on any tensor of arbitrary order. Thus, any tensor can be reordered into another tensor of arbitrary order. An example of order one tensor, vectorisation, can be given by an equivalence relation where all elements fall into the same equivalent class.

This interpretation of the reordering leads us to the realm of Algebraic Statistics, and within that realm, to the Combinatorial Design Theory [24, 25]. The Combinatorial Design Theory addresses the problem of how to build systems of finite sets that satisfy certain requirements of symmetries as stated in the classical theory of Latin squares. These theories play a very important role in creating balanced statistical experimental designs, especially for medical tests.

2.2.4 Compression

Furthermore, the tensor decomposition can provide a very high level compression of images. Some examples are presented here. The compression ratio is computed by dividing the number of elements of the image matrix with the total number of elements of the components in the decomposition.

The original image matrix is given by integers in the range of $0, \ldots, 255$ for both grey-scale and RGB colour images. The decomposition happens in the space of real numbers, but after having been decomposed, the real numbers in the components can be rescaled and transformed into the original integer interval.

Let the size of a grey-scale image be equal to $(1024, 1024)$, then we can have various patterns of decompositions as shown in Table 2.1.

Component Size	Singular Values	Full Size	Compression Ratio
$(32, 32), (32, 32)$	10	$s = 10 * 2 * 32^2$	$\frac{1024^2}{s} = 51.2$
$(16, 16), (16, 16), (4, 4)$	20	$s = 20 * (2 * 16^2 + 4^2)$	$\frac{1024^2}{s} = 99.29$

Table 2.1: Examples of possible patterns of decomposition of a grey-scale image with 1024×1024.

The components provide a tightly-compressed signature for the image as illustrated in Figure 2.3.

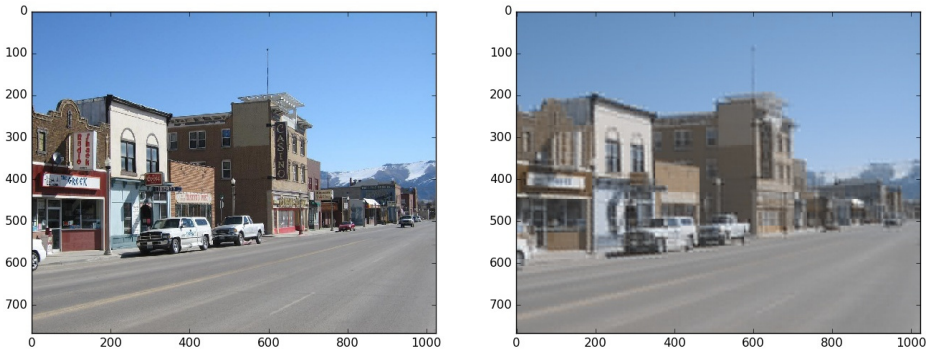


Figure 2.3: The right image is a recovered image from the Kronecker decomposition of the original image on the left. The factors were $(48, 64, 1), (16, 16, 3)$ with 6 singular values. The compression ratio was 102.4.

(a) Image without compression

(b) Image compressed by Kronecker decomposition

(c) Image compressed by SVD

Figure 2.4: An example of image compressed by the proposed and a conventional technique when the compression ratio was 100.

The Kronecker-decomposition-based features can be compared with those from a conventional compression technique with SVD and the well-known SIFT features [26]. Let the size of an RGB colour image be $(640, 960, 3)$ and the sizes of the components in the tensor decomposition be $(10, 15, 1), (8, 8, 1), (8, 8, 3)$, and let 45 singular values be computed, then $45 * (10 * 15 + 8^2 + 3 * 8^2)$ 8-bit data items can be obtained with a high compression ratio of ~ 100. This is equivalent to (i) using image compression with SVD with 4 singular values which gives $((640 * 4) + 4 + (940 * 4)) * 3$ data items and (ii) ~ 36 SIFT feature vectors with 128 real valued components, where we assume that these real numbers can be represented by 32 bits. The tensor decomposition can recover a good approximation of the original image and the colour information while the conventional technique cannot do so, as shown in Figure 2.4. Although the 36 SIFT points may be able to capture some particular characteristic points, they cannot recover the main structure of the entire image at all.

2.2.5 Interpretation of Image Components

The components provided by the tensor decomposition can be endowed with a practical interpretation. Let us discuss the two-component case where the image matrix X is expressed as a Kronecker product of two other matrices A and B. Since the second factor B in the product is forced to be the same for all positions, it has to be equal to a nonlinearly aggregated matrix. This factor is shifted around within the image and it is only scaled by elements of the matrix A. Thus, B can be interpreted as an image filter.

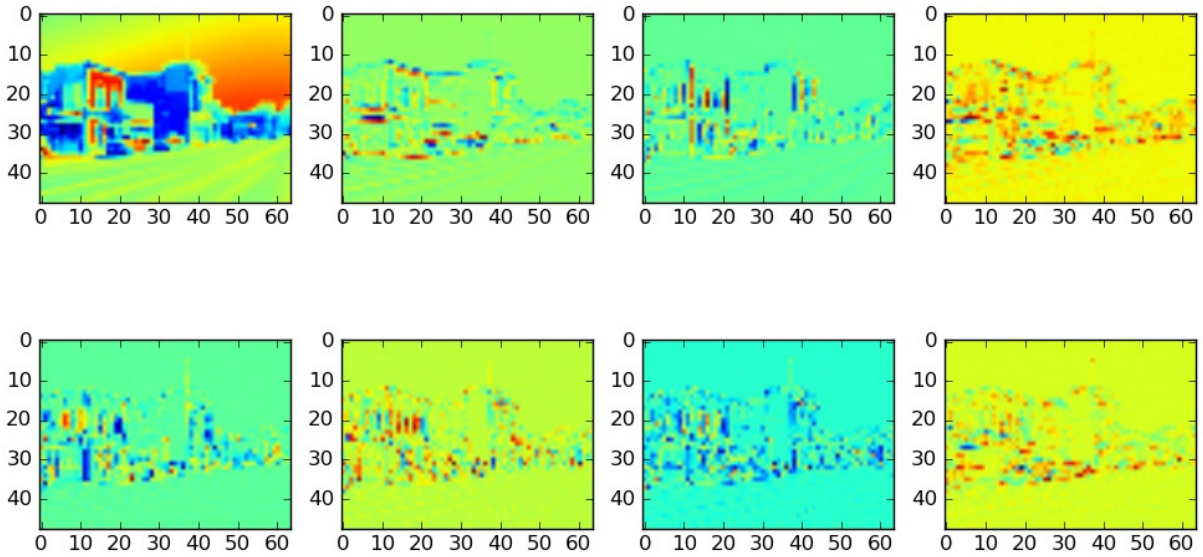


Figure 2.5: First level of the Kronecker decomposition in case of the first 8 largest singular values.

In Figure 2.5, the first level components belonging to the sequence of decreasing singular values are of the compressed image shown in Figure 2.3. This image is factorised into two levels and the corresponding low level components are presented in Figure 2.6. Some characteristic patterns can be recognised from these factors. The factor belonging to the second singular vector represents a horizontal edge filter and the factor belonging to the third singular value yields a vertical line filter.

In this way, the image decomposition finds the boundaries of the critical regions, edges, and corners in which most of the structural information concentrates, and as mentioned earlier, it also produces a very highly compressed skeleton of the data.

Since in every step the decomposition processes the residue of the previous step, it predicts those parts that have hardly been approximated earlier, thus in every step a new layer of the structure is discovered. In images, these layers are first the flat areas, then edges of different directions–vertical, horizontal, and slant–corners of different kind, and the higher order singularities of the intensity surface. This kind of incremental approach resembles a boost in which hardly predictable sample items receive larger weights.

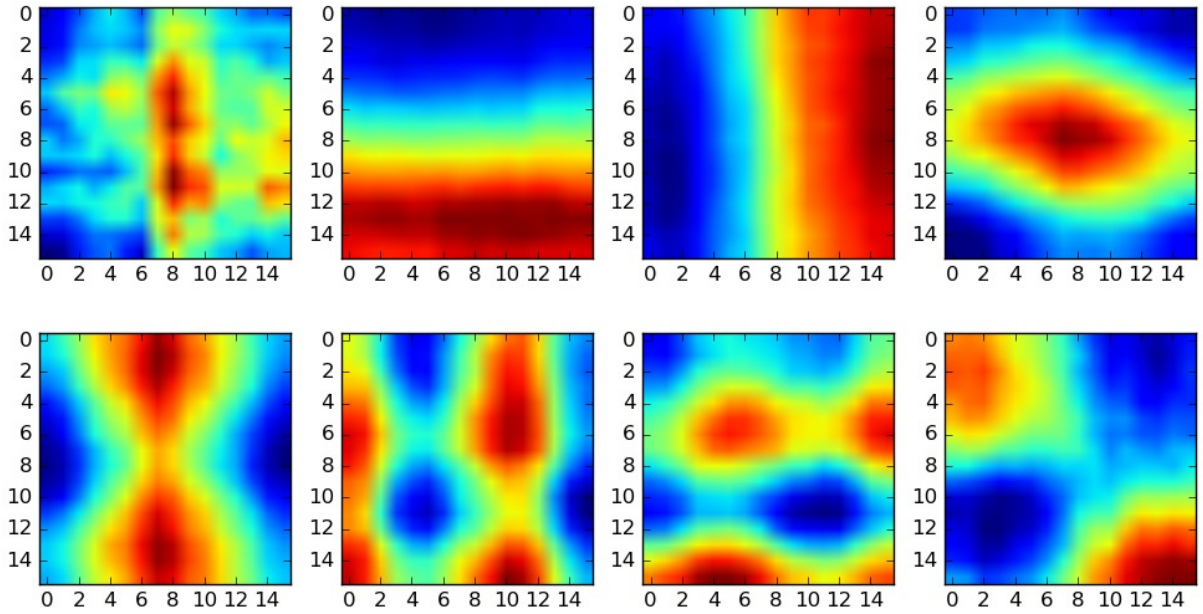


Figure 2.6: Second level of the Kronecker decomposition in case of the first 8 largest singular values.

2.2.6 Relation to Known Saliency Detection Approaches

Generally applied saliency detectors are built on estimated derivatives of the intensity function, I, of an image (see for example [27]). These derivatives can provide information about the locations that the intensity changes significantly faster than a given threshold. The general first order edge and corner detection algorithms consist of two phases: the first phase uses a Gaussian filter to smooth out the noise caused by non-differentiable representation of the image, and the second phase estimates the gradients of the image intensity. The higher order methods also exploit the second derivative, the Hessian, of the intensity image. Estimation of the derivatives can be performed by applying image filters on the image intensity, e.g. Laplacian, Sobel and Prewitt.

For example Harris-Laplace detector relies on the first order partial derivatives, I_x, I_y, of the intensity measured in the local neighbourhood of a given image point

$$\mathsf{A}(x) = \sum_{i,j} w_{ij} \begin{bmatrix} I_x^2(\mathsf{x}) & I_x I_y(\mathsf{x}) \\ I_x I_y(\mathsf{x}) & I_y^2(\mathsf{x}) \end{bmatrix}, \quad R = \det(\mathsf{A}) - \alpha \, \mathrm{trace}^2(\mathsf{A}).$$

Similarly the Hessian affine region detector exploits the second partial derivatives of the

Gaussian-filtered intensity, the Laplacian of the Gaussian (LoG) representation,

$$\mathsf{H}(x) = \begin{bmatrix} L_{xx}(\mathsf{x}) & L_{xy}(\mathsf{x}) \\ L_{xy}(\mathsf{x}) & L_{yy}(\mathsf{x}) \end{bmatrix}, \; L(x) = g(\cdot|0, \sigma_I) \otimes I(x), \; LoG = \mathsf{trace}(\mathsf{H}),$$

where L is the Gaussian-filtered intensity, and g is a Gaussian filter with a parameter σ_I.

Our Kronecker decomposition based algorithm iteratively reproduces the partial derivatives of the intensity function via higher order polynomial approximation of successive error terms. During this iterative procedure, the algorithm automatically selects, in the least-square sense, the best fitting filters as the last factor of the Kronecker product. In this way, it gives an approximation of the multivariate Taylor series of the intensity image that provides an estimation of the full series of those partial derivatives that are exploited in well-known saliency detectors. It is also important to mention that the global optimality of the Kronecker decomposition allows us to select only regions where the derivatives change the most and eliminate flat, redundant ones. This selective property leads to high level compression of the information needed to describe the variation of the intensity. For example, Harris detector based methods focus only on local features without forcing globally optimal extraction.

2.3 Combining Images with Eye Movements

The images processed in our eye movement experiments were assumed to be RGB colour images represented by 3 parallel matrices and indexed by row and column coordinates of the pixels in the images. Since the sizes of the images varied, all of them were transformed into the same common size in the learning procedure. That common size was 50×50.

The eye movements in all of the experiments were given as a list of coordinate pairs consisting of the image related row and column indices. When the eyes moved beyond the image, those eye movement coordinates were represented by invalid, e.g., negative values. Therefore, the coordinate pairs of the eye movement had to be filtered to extract only those points that corresponded to valid image pixels. When the sizes of the images were transformed to the same size, that transformation applied to the eye movement coordinates as well.

When eye movement coordinates were combined with an image, we needed to deal

with the uncertainty of eye positions caused by measurement errors and movements of the eyes and the head that were probably not related to the image. That uncertainty was handled by applying Gaussian smoothing, a spatial filter, to each point of eye movement. The Gaussian filter was centred at a point u with a width parameter σ assigns to each image point x,

$$g(\mathsf{x}|\mathsf{u},\sigma) = e^{-\frac{\|\mathsf{x}-\mathsf{u}\|^2}{2\sigma}}. \tag{2.19}$$

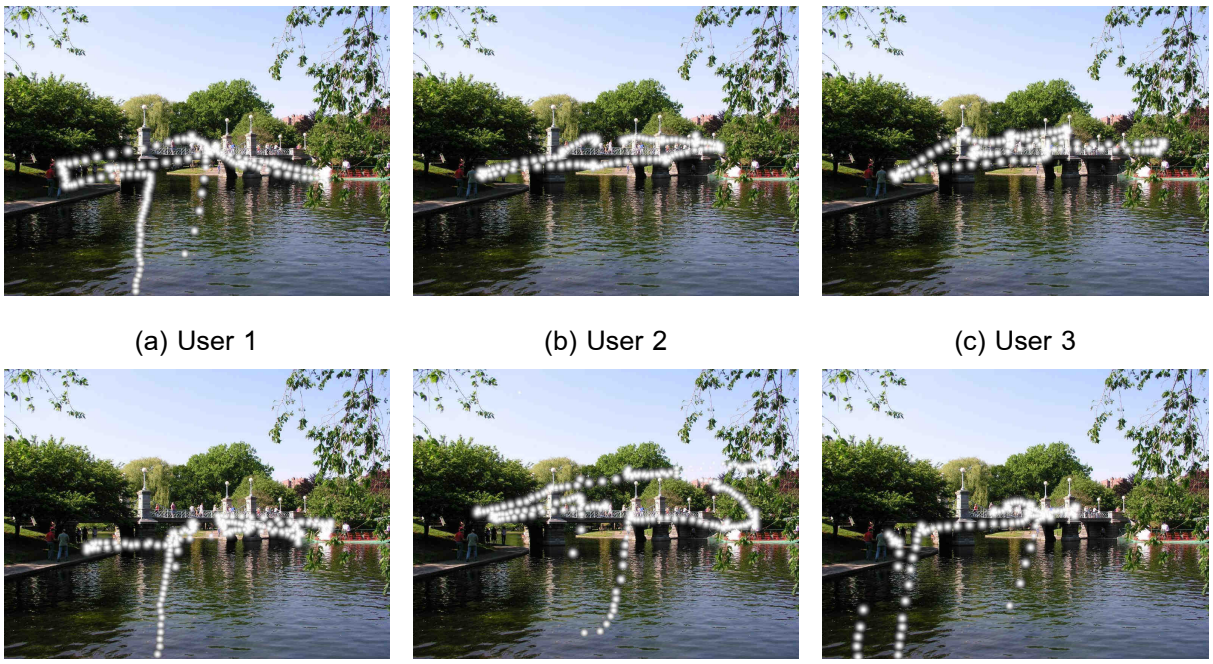
The centres of the Gaussian filters were localised at the observed points of the eye movements. Since all image points could be connected by the filter to all eye movement points, as many filter values were assigned to each image point as the number of observed eye movement points. To aggregate the filter values, we applied the following function on all of the image points x:

$$g_A(\mathsf{x}) = \max_{\mathsf{u}} g(\mathsf{x}|\mathsf{u},\sigma). \tag{2.20}$$

The function g_A assigned to each point of the image, x, a Gaussian filter value that belonged to the closest point of the eye movement. That Gaussian filter can well represent foveation in human vision [28].

In Figure 2.7, the eye movement trajectories on a stimuli image of six contributing users are presented. We were able to have some impression on the similarity and the variation of the responds of the users looking at the same image and trying to perform the same task.

Figure 2.8 shows images that were already merged with eye movements data. These images would subsequently be decomposed by a Kronecker product. The decomposition of the Kronecker product on the image and the combined images with eye movement data are shown in Figure 2.9.

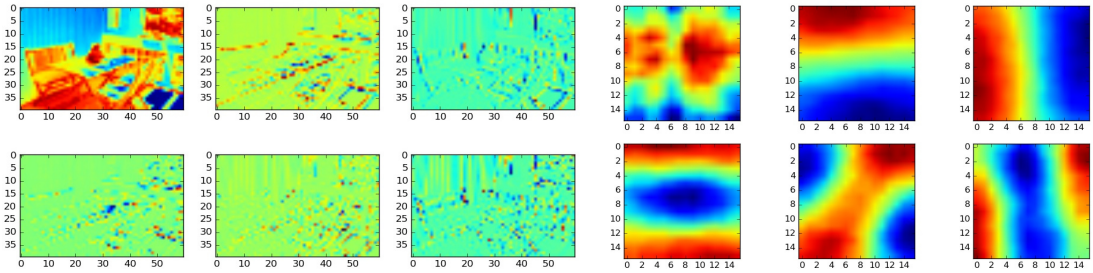


(a) User 1 (b) User 2 (c) User 3

(d) User 4 (e) User 5 (f) User 6

Figure 2.7: The eye movements of six different users on a sample image.

Figure 2.8: Eye movement data overlaid on the image (left) and images merged with eye movement data (right) of two users.



Stimuli image

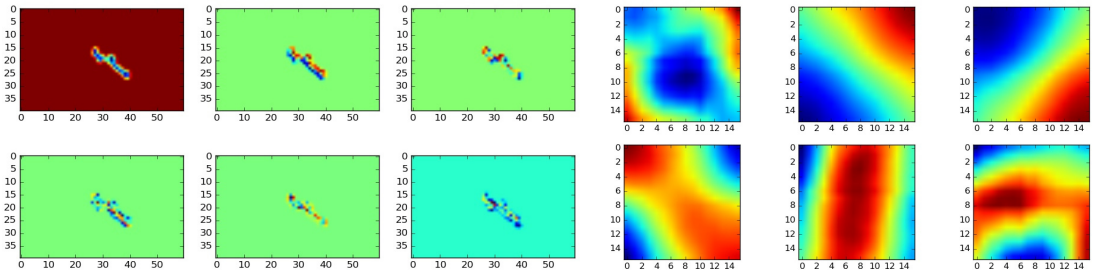


Image merged with eye movement data of User 1

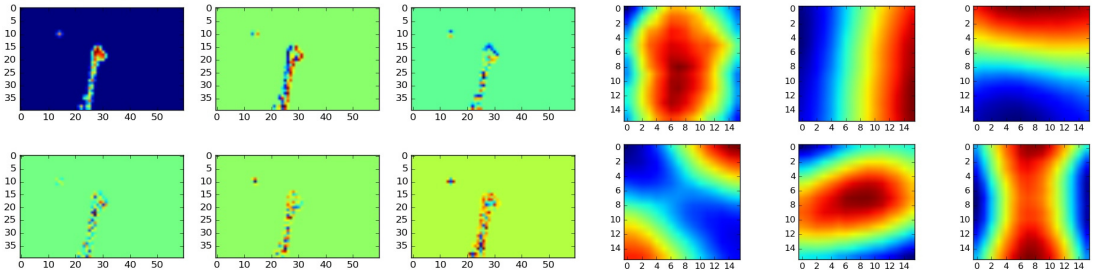


Image merged with eye movement data of User 2

Figure 2.9: Kronecker decomposition of the original stimuli image and the corresponding eye movement data of two users; two Kronecker factors and six singular values were computed

Chapter 3

Performance Evaluations and Discussions

We evaluated our TMVL algorithm on a publicly available eye tracking dataset [4]. The dataset contains eye tracking data of 15 different users on 1003 images. Each image consists of three-second free-view trajectories of different users. In order to encourage users to pay attention to the task, they were memory-tested at the end of the data collection of 100 images.

In our experiment, only eight users were randomly selected; hence, there were eight views in this setting. Each view was represented by a heatmap of each user. Heatmap quantifies the degree of importance of parts of image; a higher probability of an important part is implied by a higher density of eye movements on that part. We investigated two sets of heatmap input features.

1. Heatmap generated from eye movement data alone (H_E): A user's heatmap was created by convolving a Gaussian kernel with each eye fixation point.

2. Heatmap generated from eye movement data and the image (H_{EI}): This set of features was generated by a Kronecker decomposition of an image combined with eye movement data as described in section 2.3 with σ of 2. We extracted features with two components of Kronecker product from a grey scale processed image. The sizes of the high (first) (A) and low (second) (B) level component were

(10,10), and (5,5), respectively with 20 singular values. The feature vector became $\begin{bmatrix} \mathsf{A}_1 \otimes \mathsf{B}_1 & \mathsf{A}_2 \otimes \mathsf{B}_2 & \dots & \mathsf{A}_{20} \otimes \mathsf{B}_{20} \end{bmatrix}$.

All heatmaps were normalised to unit norm. To compare prediction performances, we used three performance matrices as follows:

1. Area under the receiver operating characteristic (AUROC) is one of commonly used performance metrics for comparing heatmaps and eye fixations[1]. It is based on an ROC curve that can be computed by varying the threshold of the predicted heatmap. A pixel is predicted as a target when its heatmap value is greater than the threshold, while it is classified as a background when the value is below the threshold. AUROC ranges from 0 (complete mismatch) to 1 (perfect match).

2. Correlation indicates the degree of linear relationship between two maps. It ranges from -1 (perfect correlation but in the opposite direction) to $+1$ (perfect correlation). Zero indicates no correlation between the two maps.

3. Jensen-–Shannon divergence (JSD) was used to identify the dissimilarity between two distributions. It is based on Kullback-Leibler divergence (KLD) which can capture a certain kind of non-linear, entropy type and dependency. KLD is defined as,

$$D_{KL}(P||Q) = \sum_{i=1} \ln\left(\frac{P_i}{Q_i}\right) P_i \tag{3.1}$$

where P, Q are the probability distributions. JSD is symmetric while KLD is not [29]. Square root of JSD has matrix properties. The more similar the two objects are, the smaller the value of JSD is, and vice versa. JSD is defined as,

$$D_{JS}(P||Q) = \frac{1}{2} D_{KL}(P||M) + \frac{1}{2} D_{KL}(Q||M) \tag{3.2}$$

where $M = \frac{1}{2}(P + Q)$.

Here, linear kernel function was used. Model selection was performed with five-fold cross validation based on AUROC. We examined two scenarios of the test sets: (i) {1-7} missing views were randomly selected and (ii) one fixed view was missing for each run. The experiments were run 10 times with different random data splits.

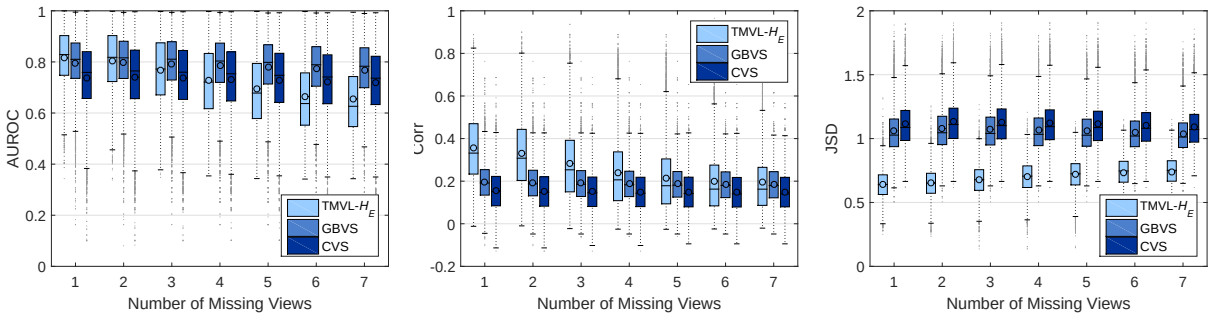
[1]Available for download at http://www.vision.caltech.edu/∼harel/share/gbvs.php.

Although our scenario is of a supervised learning approach, but its setting is different from those mentioned earlier in the introduction. All previous works used a saliency map of average locations fixated by all users as a global model. In our scenario, we instead used individual fixation maps and focused on user models. Our assumption was that there were some correlations between users' eye movement behaviours. We aim to predict the individual maps. There was no available technique designed for this kind of scenario, so we compared our proposed method with baseline methods in a similar way that we did in one of our previous works [30]. The baseline methods were two well-known saliency map models: Conventional Visual Saliency (CVS) [1], and Graph-Based Visual Saliency (GBVS) [2].

3.1 Randomly Select Missing Views

When the number of missing views increased, AUROC and correlation decreased for all algorithms as shown in Figure 3.1a and 3.1b, respectively. On the other hand, JSD increased when there was an increase in the number of missing views as shown in Figure 3.1c. When the number of missing views increased, TMVL-H_E performance was dramatically reduced while GBVS and CVS's performances were slightly decrease. TMVL-H_E only used eye movement data, therefore, the prediction performances highly depended on number of available view, while GBVS and CVS used image information. TMVL-H_E's performances on AUROC were better than those on GBVS when there were 1–2 missing views and better than those on CVS when there were 1–3 missing views. Unfortunately, TMVL-H_E performances on AUROC were worse than those on GBVS and CVS in other cases as shown in Figure 3.1a. However, TMVL-H_E was still able to achieve better average correlation and JSD than GBVS and CVS were in all cases as shown in Figure 3.1b and 3.1c, respectively.

Figure 3.2 shows an example of prediction by GBVS, CVS, and our proposed algorithm when two views were missing. It can be seen that GBVS and CVS failed to predict where users were looking at. Both algorithms put attention on the woman's arm and the windows while users actually focused on her face. Clearly, TMVL was more effective than GBVS and CVS for all three performance matrices as shown in Table 3.1,



(a) AUROC (b) Correlation (c) JSD

Figure 3.1: A boxplot comparison of all methods when randomly selected {1-7} missing views were considered.

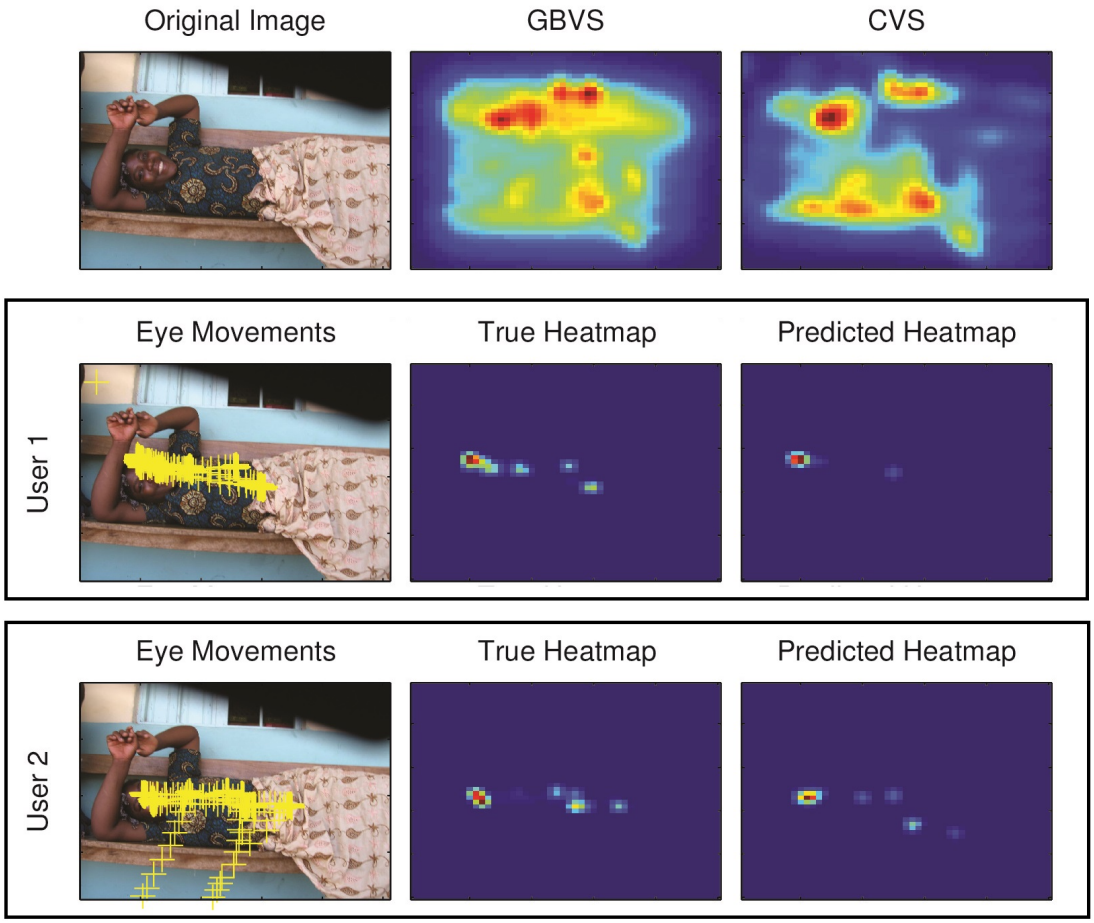


Figure 3.2: True and TMVL-predicted heatmaps of an eight-view problem with two views missing (User 1 and 2) compared to those predicted by baseline methods–GBVS and CVS.

User	Method	AUROC	Corr	JSD
1	TMVL	0.8065	0.7160	0.3726
	GBVS	0.6543	0.0783	1.2755
	CVS	0.7358	0.0792	1.2454
2	TMVL	0.7812	0.6900	0.4064
	GBVS	0.6707	0.0985	1.1863
	CVS	0.6983	0.0981	1.1627

Table 3.1: Performance matrices of all of the methods on the image in figure 3.2. Bold values indicate the best performance achieved in each user.

We improved prediction performance by using H_{EI} as feature vectors and compared the prediction results with those from other methods, as shown in Figure 3.3. TMVL-H_{EI} outperformed CVS in all cases but was only better than GBVS in {1–4}-missing-view cases. TMVL was comparable to GBVS when five views were missing but was worse than GBVS in the case of {6–7} missing views. It should be noted that we only evaluated the performances on AUROC as it compared the predicted heatmaps directly with eye movement data for all of the methods. Correlation and JSD compare between predicted heatmaps to target heatmaps but target heatmaps for TMVL with H_E and with H_{EI} were different. Hence, we were not able to directly compare these matrices. It is clear that using H_{EI} that combined images with eye movement information could dramatically enrich the performance on AUROC over using H_E as shown in Figure 3.4. AUROC was improved at 2.82% in 1-missing-view case and up to 15.20% in seven-missing-view case.

Figure 3.5 shows an example of predictions by all of the methods when two views were missing. Again, GBVS and CVS failed to predict that the cat on the table would be favourably looked at rather than chairs with light shone on them. Both TMVL with H_E and H_{EI} were able to detect the cat; however, TMVL-H_{EI} was able to capture more eye movements than TMVL-H_E did, especially for User 2.

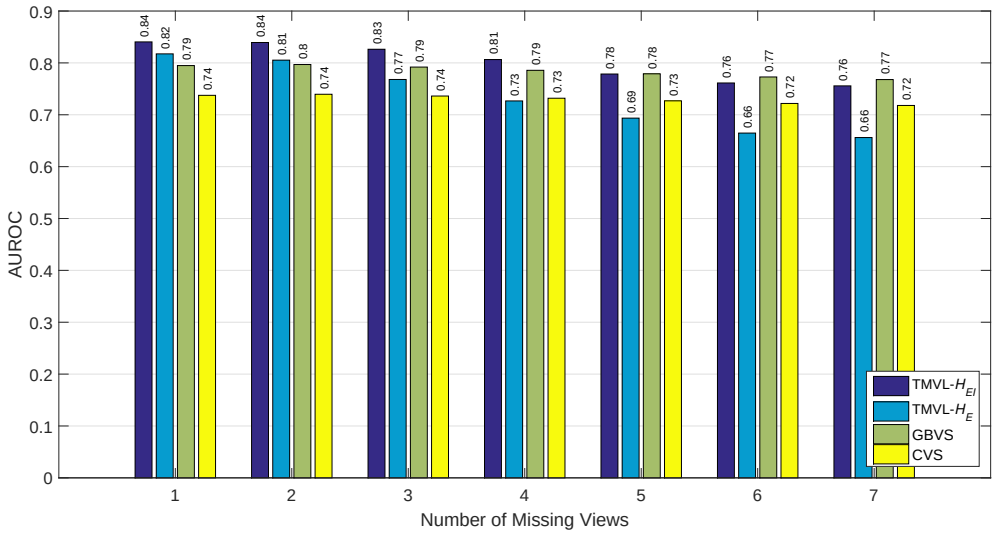


Figure 3.3: A comparison between the proposed method and two existing saliency prediction methods in the case of randomly selected {1–7} missing views

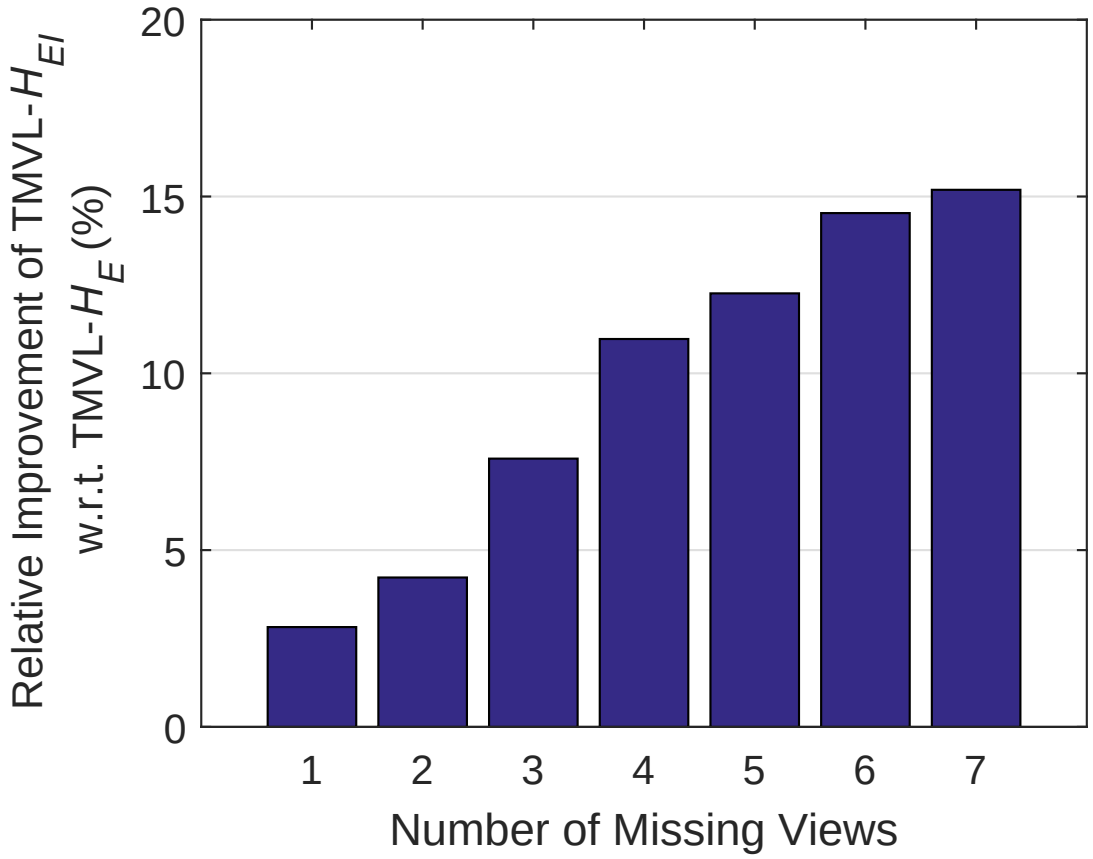


Figure 3.4: Relative improvement of AUROC using H_{EI} compared to using H_E in TMVL

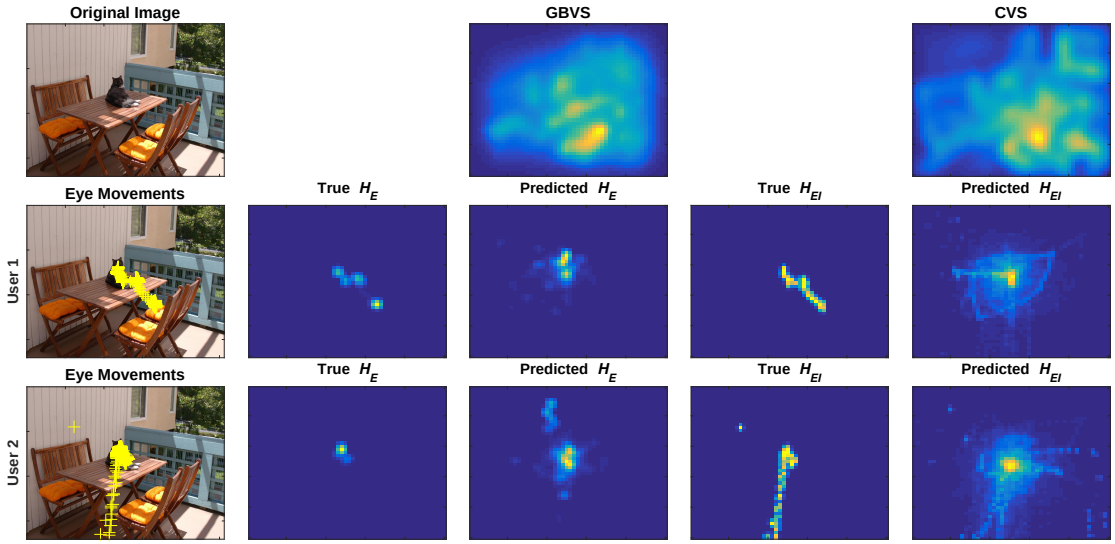


Figure 3.5: A comparison of predicted heatmaps by all of the considered methods.

Method	User Index							
	1	2	3	4	5	6	7	8
TMVL-H_{EI}	0.8550	0.8631	0.7939	0.8300	0.9010	0.8468	0.8418	0.7901
TMVL-H_E	0.8349	0.8508	0.7690	0.8079	0.8711	0.8246	0.8184	0.7670
GBVS	0.8040	0.7848	0.7638	0.8028	0.8410	0.7888	0.8025	0.7636
CVS	0.7426	0.7164	0.7146	0.7517	0.7708	0.7257	0.7457	0.7171

Table 3.2: A comparison between all of the methods with AUROC when only one fixed view/user are missing. Bold values indicate the best AUROC achieved in each user.

3.2 Fixing A Missing View

In this scenario, we wanted to predict where a user would be looking at in a (test) set of images based on where the other seven users were looking at in the same set. According to Table 3.2–Table 3.4, TMVL was the best contender for all users, followed by GBVS and CVS in that order. Using H_{EI} as input features could improve prediction performance significantly for all cases and on all matrices.

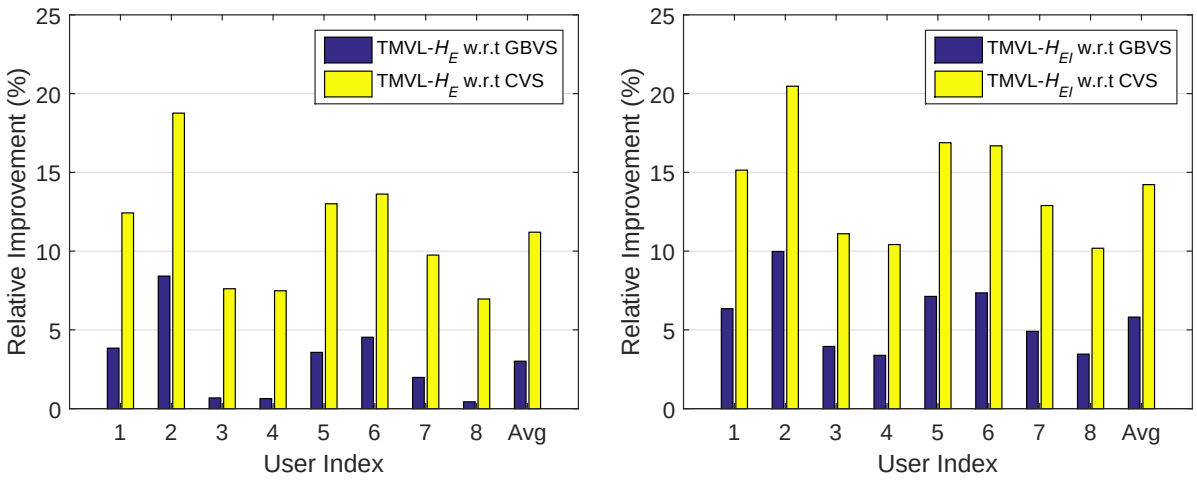
Compared to GBVS and CVS, TMVL-H_E showed an improvement with AUROC for all users at 3.01% and 11.20% on the average, respectively, as shown in Figure 3.6a. TMVL-H_{EI} performance was improved much more, 5.81% against GBVS and 14.22% against

Feature	Method	User Index							
		1	2	3	4	5	6	7	8
H_{EI}	TMVL	0.4147	0.4747	0.3568	0.4046	0.4569	0.4266	0.4154	0.3657
	GBVS	0.2310	0.2319	0.2356	0.2736	0.2259	0.2432	0.2660	0.2645
	CVS	0.1762	0.1757	0.1829	0.2114	0.1727	0.1829	0.2045	0.2097
H_E	TMVL	0.3276	0.4433	0.2951	0.3437	0.4013	0.3874	0.3706	0.2861
	GBVS	0.1835	0.1932	0.1856	0.2083	0.1928	0.1981	0.2135	0.1976
	CVS	0.1425	0.1465	0.1476	0.1662	0.1501	0.1526	0.1675	0.1604

Table 3.3: A comparison between all of the methods with correlation when only one fixed view/user are missing.

Feature	Method	User Index							
		1	2	3	4	5	6	7	8
H_{EI}	TMVL	0.6324	0.5279	0.5152	0.4866	0.6234	0.5364	0.4998	0.4095
	GBVS	0.7442	0.7046	0.5912	0.5893	0.8102	0.6735	0.6193	0.4921
	CVS	0.7625	0.7171	0.5927	0.5968	0.8343	0.6844	0.6277	0.4833
H_E	TMVL	0.6792	0.5463	0.7011	0.6606	0.6055	0.6083	0.6269	0.7003
	GBVS	1.1585	1.1138	1.0224	1.0024	1.1760	1.0451	0.9996	0.9582
	CVS	1.2184	1.1753	1.0757	1.0595	1.2376	1.1056	1.0587	1.0103

Table 3.4: A comparison between all of the methods with JSD when only one fixed view/user are missing.



(a) H_E (b) H_{EI}

Figure 3.6: Relative improvement of TMVL compared to GBVS and CVS when only one fixed view/user was missing.

CVS on the average, as shown in Figure 3.6b. Our proposed algorithm achieved the highest improvement compared to both baselines for User 2. This means that other users' eye movements behaviours were very useful to User 2. However, the least improvement was found for User 8. This indicates that user adaptation can be useful as the performance was improved even though not so much for the case of User 8. Using TMVL-H_{EI} yielded better AUROC, at 2.73% on the average than TMVL-H_E did, as shown in Figure 3.7.

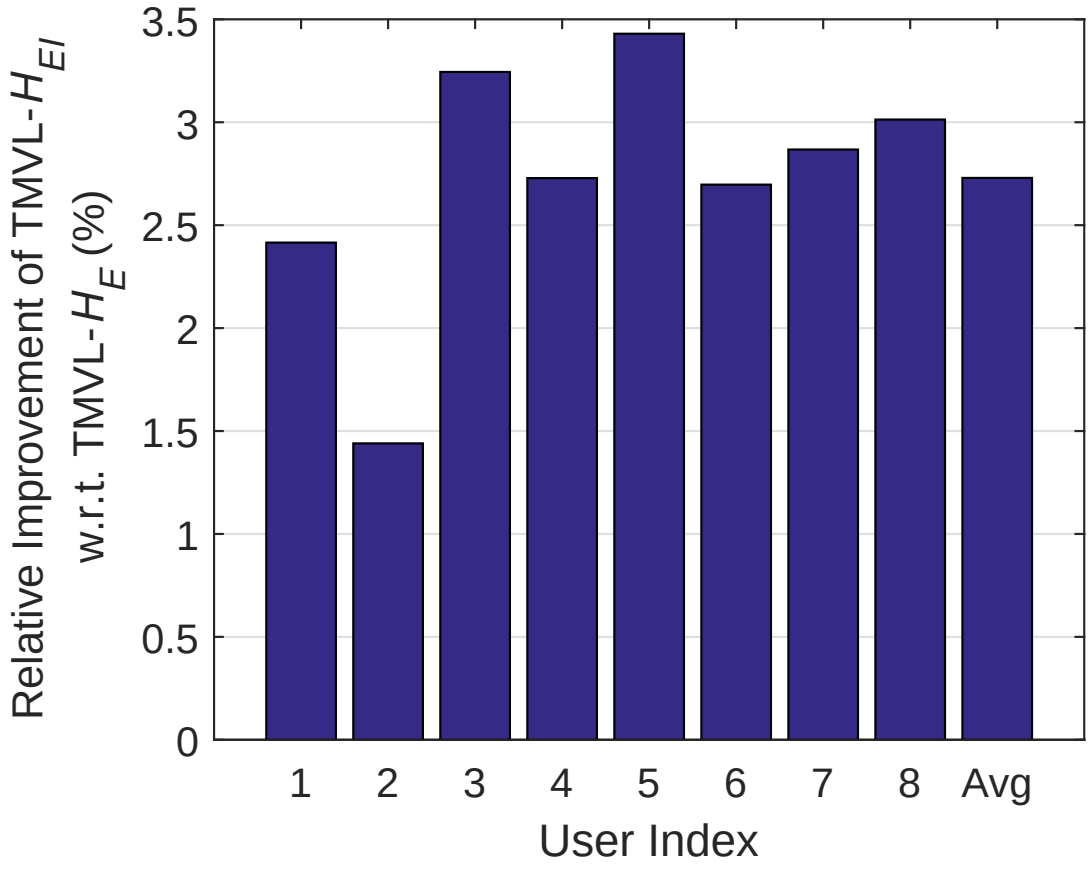


Figure 3.7: Relative improvement of TMVL-H_{EI} compared to TMVL-H_E when only one fixed view/user was missing.

Chapter 4

Conclusion

In this report, we introduced a missing-value prediction schema built upon maximum-margin-based learning and invariances of tensor algebra. Our proposed algorithm was tested on an eye movement dataset in order to identify where users were looking at in images. We also proposed a new technique that decomposes images called “Kronecker Decomposition”. This technique can be used in image compression applications. We also proposed an approach to combine image with eye movement data. The processed image is then decomposed by using Kronecker decomposition. We have demonstrated that our framework was able to perform better than two well-known saliency detection techniques. Several initial results show that user adaptation may be useful; thus, user information should be investigated further in future research.

References

[1] L. Itti, C. Koch, and E. Niebur, “A model of saliency-based visual attention for rapid scene analysis,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 11, pp. 1254–1259, 1998.

[2] J. Harel, C. Koch, and P. Perona, “Graph-based visual saliency,” in Advances in Neural Information Processing Systems, pp. 545–552, 2006.

[3] A. Borji, M.-M. Cheng, H. Jiang, and J. Li, “Salient object detection: A survey,” arXiv, vol. 1411.5878, 2014.

[4] T. Judd, K. Ehinger, F. Durand, and A. Torralba, “Learning to predict where humans look,” in IEEE 12th International Conference on Computer Vision, pp. 2106–2113, 2009.

[5] J. M. Henderson, J. R. Brockmole, M. S. Castelhano, and M. Mack, “Visual saliency does not account for eye movements during visual search in real-world scenes,” Eye Movements: A Window on Mind and Brain, pp. 537–562, 2007.

[6] Q. Zhao and C. Koch, “Learning a saliency map using fixated locations in natural scenes,” Journal of Vision, vol. 11, no. 3, pp. 1–15, 2011.

[7] M. Liang and X. Hu, “Feature selection in supervised saliency prediction,” IEEE Transactions on Cybernetics, vol. 45, no. 5, pp. 914–926, 2015.

[8] J. Wang, A. Borji, C. J. Kuo, and L. Itti, “Learning a combined model of visual saliency for fixation prediction,” IEEE Transactions on Image Processing, vol. 25, no. 4, pp. 1566–1579, 2016.

[9] L. Zhang, X. Li, L. Nie, Y. Yang, and Y. Xia, “Weakly supervised human fixations prediction,” IEEE Transactions on Cybernetics, vol. 46, no. 1, pp. 258–269, 2016.

[10] J. Liu, P. Musialski, P. Wonka, and J. Ye, “Tensor completion for estimating missing values in visual data,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 1, pp. 208–220, 2013.

[11] C.-Y. Chen and K. Grauman, “Inferring unseen views of people,” in IEEE Conference on Computer Vision and Pattern Recognition, pp. 2011–2018, 2014.

[12] D. R. Hardoon, K. Pasupa, and J. Shawe-Taylor, “Image ranking with implicit feedback from eye movements,” in Proceeding of the 6th Biennial Symposium on Eye Tracking Research & Applications (ETRA 2010), 22-24 March 2010, Austin, USA (C. H. Morimoto, H. O. Istance, A. Hyrskykari, and Q. Ji, eds.), pp. 291–298, 2010.

[13] P. Auer, Z. Hussain, S. Kaski, A. Klami, J. Kujala, J. Laaksonen, A. P. Leung, K. Pasupa, and J. Shawe-Taylor, “Pinview: Implicit feedback in content-based image retrieval,” in Proceeding of the Workshop on Applications of Pattern Analysis (WAPA 2010), 1-2 September 2010, Cumberland Lodge, UK (T. Diethe, N. Cristianini, and J. Shawe-Taylor, eds.), vol. 11 of Journal of Machine Learning Research - Proceedings Track, pp. 51–57, 2010.

[14] Z. Hussain, A. P. Leung, K. Pasupa, D. R. Hardoon, P. Auer, and J. Shawe-Taylor, “Exploration-exploitation of eye movement enriched multiple feature spaces for content-based image retrieval,” in Proceeding of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD 2010), Part I, 20-24 September 2010, Barcelona, Spain (J. L. Balcázar, F. Bonchi, A. Gionis, and M. Sebag, eds.), vol. 6321 of Lecture Notes in Computer Science, pp. 554–569, 2010.

[15] K. Pasupa, P. Chatkamjuncharoen, C. Wuttilertdeshar, and M. Sugimoto, “Using image features and eye tracking device to predict human emotions toward abstract images,” in Proceeding of the 7th Pacific Rim Symposium on Image and Video Technology (PSIVT 2015), 23-27 Nov 2015, Auckland, New Zealand (T. Bräunl, B. Mc-

Cane, M. Rivers, and X. Yu, eds.), vol. 9431 of Lecture Notes in Computer Science, pp. 419—430, 2016.

[16] M. Itskov, Tensor Algebra and Tensor Analysis for Engineers With Applications to Continuum Mechanics. Springer, 2nd ed., 2009.

[17] J. Synge and A. Schild, Tensor Calculus. Dover, 1978.

[18] K. Astikainen, L. Holm, E. Pitkänen, S. Szedmak, and J. Rousu, "Towards structured output prediction of enzyme function," in BMC Proceedings, vol. 2, Suppl 4, p. S2, 2008.

[19] S. Szedmak, T. De Bie, and D. R. Hardoon, "A metamorphosis of canonical correlation analysis into multivariate maximum margin learning," in The 15th European Symposium on Artificial Neural Networks, pp. 211–216, 2007.

[20] T. Joachims, "Training linear SVMs in linear time," in Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2006), 20-23 August 2006, Philadelphia, USA, pp. 217–226, 2006.

[21] N. Cristianini and J. Shawe-Taylor, An introduction to Support Vector Machines and other kernel-based learning methods. Cambridge University Press, 2000.

[22] C. V. Loan, "The ubiquitous kronecker product," Journal of Computational and Applied Mathematics, vol. 123, pp. 85–100, 2000.

[23] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," SIAM Review, vol. 51, no. 3, pp. 455–500, 2009.

[24] L. Pachter and B. Sturmfels, Algebraic Statistics for Computational Biology. Cambridge University Press, 2005.

[25] M. Drton, B. Sturmfels, and S. Sullivant, Lectures on Algebraic Statistics, vol. Oberwolfach Seminars, Vol 40. Birkhauser, 2009.

[26] D. G. Lowe, "Object recognition from local scale-invariant features," in Proceedings of the Seventh IEEE International Conference on Computer Vision, vol. 2, pp. 1150–1157, 1999.

[27] C. Schmid, R. Mohr, and C. Bauckhage, “Evaluation of interest point detectors,” International Journal of Computer Vision, vol. 37, no. 2, pp. 151–172, 2000.

[28] E.-C. Chang, S. Mallat, and C. Yap, “Wavelet foveation,” Applied and Computational Harmonic Analysis, vol. 9, no. 3, pp. 312–335, 2000.

[29] J. Briët and P. Harremoës, “Properties of classical and quantum Jensen-Shannon divergence,” Physical Review A, vol. 79, no. 5, p. 052311, 2009.

[30] K. Pasupa and S. Szedmak, “Learning to predict where people look with tensor-based multiview learning,” in Proceeding of the 22nd International Conference on Neural Information Processing (ICONIP 2015), 9-12 Nov 2015, Istanbul, Turkey (S. Arik, T. Huang, W. K. Lai, and Q. Liu, eds.), vol. 9489 of Lecture Notes in Computer Science, pp. 432—441, 2015.

Research Outputs

Project Code: TRG5680090

Project Title: Predicting where Humans Look at in Images by Machine Learning Technique

---

International Journal

1. Kitsuchart Pasupa, Sandor Szedmak, “Utilising Kronnecker Decomposition and Tensor-based Multi-view Learning to Predict Where People are Looking in Images”, Neurocomputing, accepted. (Scopus, ISI) (5-Year Impact Factor: 2.471)

International Conference Proceeding

1. Kitsuchart Pasupa, Siripen Jungjareantrat, “Water Levels Forecast In Thailand: A Case Study Of Chao Phraya River”, In Proceeding of the 14th International Conference on Control, Automation, Robotics and Vision (ICARCV 2016), 13–15 November 2016, Phuket, Thailand, pp. 1–6, 2016. (Scopus, ISI-CPCI-S)
2. Kitsuchart Pasupa, Wisuwat Sunhem, “A Comparison between Shallow and Deep Architecture Classifiers on Small Dataset”, In Proceeding of the 8th International Conference on Information Technology and Electrical Engineering (ICITEE 2016), 5–6 October 2016, Yogyakarta, Indonesia, pp. 390–395, 2016. (Scopus, ISI-CPCI-S)

Appendices

Appendix A

International Journal

1. Kitsuchart Pasupa, Sandor Szedmak, “Utilising Kronnecker Decomposition and Tensor-based Multi-view Learning to Predict Where People are Looking in Images”, Neurocomputing, accepted. (Scopus, ISI) (5-Year Impact Factor: 2.471)

Submissions with an Editorial Office Decision for Author Kitsuchart Pasupa, PhD

Page: 1 of 1 (1 total completed submissions) Display 10 ▼ results per page.

Action	Manuscript Number	Title	Initial Date Submitted	Status Date	Current Status	Date Final Disposition Set	Final Disposition
View Submission R1 View Decision Letter Send E-mail	NEUCOM-D-16-02114	Utilising Kronecker Decomposition and Tensor-based Multi-view Learning to Predict Where People are Looking in Images	06/27/2016	11/15/2016	Accept		

Page: 1 of 1 (1 total completed submissions) Display 10 ▼ results per page.

<< Author Main Menu

You should use the free Adobe Acrobat Reader 6 or later for best PDF Viewing results.

Get Acrobat Reader

Utilising Kronecker Decomposition and Tensor-based Multi-view Learning to Predict Where People are Looking in Images

Kitsuchart Pasupa[a,*], Sandor Szedmak[b]

[a] *Faculty of Information Technology, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand*
[b] *Department of Computer Science, Aalto University, Aalto FI-00076, Finland*

Abstract

Eye movement data collection is very expensive and laborious. Moreover, there are usually missing values. Assuming that we are collecting eye movement data from a set of images viewed by different users, there is a possibility that we will not able to collect the data of every user from every image–one or more views may not be represented in the image. We assume that the relationships among the views can be learnt from the whole collection of views (or items). The task is then to reproduce the missing part of the incomplete items from the relationships derived from the complete items and the known part of these items. Using certain properties of tensor algebra, we showed that this problem can be formulated consistently as a regression type learning task. Furthermore, there is a maximum margin based optimisation framework in which this problem can be solved in a tractable way. This problem is similar to learning to predict where a person is looking in an image. Therefore, we proposed an algorithm called "Tensor-based Multi-View Learning" (TMVL) in this paper. Furthermore, we also proposed a technique for improving prediction by introducing a new feature set obtained from Kronecker decomposition of the image fused with user's eye movement data. Using this new feature can improve prediction performance markedly. The proposed approach was proven to be more effective than two

*Corresponding author
Email address: kitsuchart@it.kmitl.ac.th (Kitsuchart Pasupa)

well-known saliency detection techniques.

Keywords: multi-view learning, missing data, tensor algebra, maximum margin learning, eye movements, Kronecker decomposition

1. Introduction

Many researchers have paid attention to image understanding that allows computer to capture the meanings of images in the same way as humans do. One of the challenging tasks in image understanding is saliency prediction. Visual saliency is a property of locations or object in the visual world, e.g., in images. If an object is salient, it stands out from its neighbours, with a high probability of being able to draw humans' attention to it.

It is very important to learn which parts humans tend to look at in scenes or images. Saliency prediction is useful in many applications–such as graphic design, web design, and human computer interaction–because it enables designers to evaluate their visual design quality. Many methods of saliency modelling have been proposed [1, 2]. More methods can be found in a recent survey paper that covered 256 publications related to saliency object detection [3]. Saliency models can be divided into two categories: supervised and unsupervised learning based models.

Itti *et al.* (1988) and Harel *et al.* (2006) investigated bottom-up visual saliency (i.e., low level image feature that does not involve supervised information) [1, 2]; unfortunately, human gazes do not usually match the map [4] because they are highly influenced by any image related tasks. If users are requested to view images without having been given a particular task, their gazes will be automatically directed by low-level image feature. In the case that users have been given a clear and specific task, their eye movements will be controlled by the content of the image. Consequently, top-down visual features are the features that should be considered [5]. Another approach is supervised learning based saliency model. It utilises eye movement data which are mapped with image features [6, 7, 8, 9].

In order to learn where humans tend to look at in images, an eye tracker is required to collect eye movement data. In real-world scenarios, eye movement data collection is tedious, laborious, and expensive. Moreover, data loss is inevitable as (i) an eye tracker may temporarily lose track of a subject because he or she is moving during the experiment, and (ii) a subject may fail to respond to all of the tasks. Consequently, we aimed to estimate missing eye movement data from the available data on the same task. It is similar to learning to predict where users tend to look at based on their previous eye movement data on other images and on other available users' eye movement data on the considered image. This leads to the learning scenario introduced in this paper. It is built with a general assumption that multiple views of a problem are available. It is not always possible to observe all of the views in a realistic scenario; therefore, this problem can be cast as a multi-view learning problem with missing data. In this scenario, we assume that initially there is a subset of training samples in which a complete set of views can be observed, but later on probably only some random subsets of views can be collected.

The goal of the learning task is to estimate the values of missing views in each sample. This scenario can occur in a real experiment. This type of problem is a generalisation of classical supervised learning problems such as regression. Face recognition is one of the applications that can be considered under this framework (when some parts or views of the faces are unknown because of occlusion, for example). In developing the learning framework, we made two mild assumptions: (i) there is a reasonable large number of observations (samples) where all of the views are known, therefore, a learning procedure can be realised; and (ii) from incomplete observations, at least one view is available. We made no assumption about the distribution of the missing views, but any prior knowledge about the missing data can be exploited to improve the estimation of their values.

In this paper, we introduced a formulation that can be considered as a generalised regression problem in which missing values are estimated from available views and their relationships are extracted from training samples. Assuming

that the missing views of a sample are output $\mathbf{y}$ and the known parts are input $\mathbf{x}$, then we have $\mathbf{y} \Leftarrow \mathbf{W}\mathbf{x}$, where $\mathbf{W}$ is a linear operator that learns from complete data and describes the relationships between different views. The difficulty of this kind of regression arises from the fact that the output and input may vary among the sample items. We proposed a "Tensor-based Multi-view Learning (TMVL)" algorithm to handle the problem of incomplete view. Providing a tractable learning algorithm, TMVL is based on properties of tensor algebra and maximum margin-based optimisation framework. Tensor decomposition has already been used in some missing data problems, e.g., [10, 11], but their settings are different from ours. Liu *et al.* (2013) investigated a low-rank tensor technique based on tensor-trace norm minimisation problem in image reconstruction [10], while Chen and Grauman (2014) proposed a probabilistic tensor model for inferring human appearance from unseen viewpoints [11]. In this paper, we show that our proposed method can estimate missing eye movements, which can be exploited for predicting where humans are likely to look at in images.

We also propose a novel approach for fusing eye movement information with image features in order to enhance prediction performance. There are many pieces of evidence suggesting that fusing low-level image features with eye movement improves prediction accuracy [12, 13, 14, 15]. Here, we employed factors derived from Kronecker decomposition of image fused with eye movement data to represent each view in TMVL.

The outline of the paper is as follows: Section 2 describes all methods proposed in this work, including TMVL algorithm, and tensor decomposition. Section 3, shows the performance results of our proposed methods on a real-world dataset. Finally, the conclusion of our study is presented in section 4.

2. Methodology

This section explains the methods proposed in this work: Subsection 2.1 describes the TMVL algorithm, its algebraic framework, and the corresponding

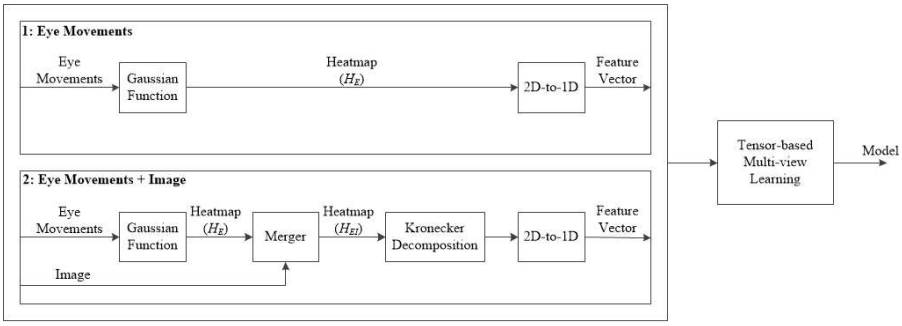


Figure 1: Main components and data flow in this work. There are two sets of features which are eye movement information only and image fused with eye movement information.

optimisation problem. Subsection 2.2 explains the procedure for decomposing images as tensors, followed by an approach for combining images with eye movement data in subsection 2.3. A model-training process framework, including a data processing pipeline for two sets of features, is visually summarised in Figure 1.

2.1. Tensor-based Multi-view Learning Algorithm

As previously mentioned, we had a set of complete views of our samples that we used as training set, and a test set in which the views were not complete (missing randomly). We aimed to fill in the missing views for each sample. An example of multi-view learning problem with missing data is shown in Figure 2.

2.1.1. Algebraic Framework

Let us denote $\mathcal{R} = \{1, \ldots, n_R\}$ as the set of indices of the views considered. In our model, each of these views has a corresponding linear vector space $\mathcal{Z}_r,\ r \in \mathcal{R}$ over real numbers. The dimensions of these spaces are denoted by $\mathrm{Dim}(\mathcal{Z}_r) = d_r,\ r \in \mathcal{R}$. The set $\mathcal{J}_R = \{j_1, \ldots, j_{n_R}\}$ comprises the indices of the samples within each of the spaces corresponding to the views, enumerating the components of the vectors chosen from the space corresponding to the views. The range of these indices is equal to the number of dimensions of the corresponding space.

	Views			
Source spaces:	$\mathcal{Z}_1$	$\mathcal{Z}_2$	$\mathcal{Z}_3$	$\mathcal{Z}_4$
	$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$
Complete items:	$\mathbf{z}_1^1$	$\mathbf{z}_1^2$	$\mathbf{z}_1^3$	$\mathbf{z}_1^4$
(Training)	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$\mathbf{z}_m^1$	$\mathbf{z}_m^2$	$\mathbf{z}_m^3$	$\mathbf{z}_m^4$
Incomplete items:	$\mathbf{z}_{m+1}^1$	$\mathbf{z}_{m+1}^2$	$\cdot$	$\mathbf{z}_{m+1}^4$
(Test)	$\cdot$	$\cdot$	$\mathbf{z}_{m+2}^3$	$\mathbf{z}_{m+2}^4$
	$\cdot$	$\mathbf{z}_{m+3}^2$	$\cdot$	$\cdot$
	$\mathbf{z}_{m+4}^1$	$\cdot$	$\cdot$	$\mathbf{z}_{m+4}^4$
	$\cdot$	$\mathbf{z}_{m+5}^2$	$\mathbf{z}_{m+5}^3$	$\cdot$
	$\cdot$	$\mathbf{z}_{m+6}^2$	$\mathbf{z}_{m+6}^3$	$\mathbf{z}_{m+6}^4$
	$\mathbf{z}_{m+7}^1$	$\cdot$	$\mathbf{z}_{m+7}^3$	$\cdot$
	$\cdot$	$\cdot$	$\cdot$	$\mathbf{z}_{m+8}^4$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Figure 2: Graphical representation of the multi-view learning framework for a four-view learning problem.

A sample is chosen out of direct products of these spaces, and each sample item consists of as many vectors as the number of views,

$$\begin{array}{lcccc} & & \text{Views:} & & \\ \text{Linear vector spaces:} & \mathcal{Z}_1 & \dots & \mathcal{Z}_{n_R} & \\ & \Downarrow & \dots & \Downarrow & \\ \text{Sample:} & \mathbf{z}_i^1 & \dots & \mathbf{z}_i^{n_R} & i = 1, \dots, m. \end{array}$$

The product space of the views is given by the tensor product of the spaces, $\mathcal{Z} = \bigotimes_{r \in \mathcal{R}} \mathcal{Z}_r$. This construction forms the algebraic framework of our solution, see [16, 17] and the references therein for more details.

If we are given two tensor products of vectors then the following contraction operator $[.,.]$ can be defined over them as

$$[\bigotimes_{q \in \mathcal{Q}} \mathbf{u}^q, \bigotimes_{r \in \mathcal{R}} \mathbf{v}^r] = \prod_{q \in \mathcal{Q} \cap \mathcal{R}} \langle \mathbf{u}^q, \mathbf{v}^q \rangle \bigotimes_{q \in \mathcal{Q} \setminus \mathcal{R}} \mathbf{u}^q \bigotimes_{r \in \mathcal{R} \setminus \mathcal{Q}} \mathbf{v}^r, \qquad (1)$$

where the inner product is computed for all common indices. When the two index sets are coincident then the following well known identity can be used to unfold the inner products of the tensor products as

$$\langle \bigotimes_{q \in \mathcal{Q}} \mathbf{z}_i^r, \bigotimes_{q \in \mathcal{Q}} \mathbf{z}_j^q \rangle = \prod_{q \in \mathcal{Q}} \langle \mathbf{z}_i^q, \mathbf{z}_j^q \rangle. \qquad (2)$$

This identity states that the inner product of tensor products of vectors is equal to the product of the inner product of these vectors.

This interpretation of the indices is compatible with the notations commonly used in tensor algebra, namely, with the so-called "Einstein summation convention". The symbol of summation $\sum$ is omitted and summation has to be carried out over all indices which are denoted by the same symbol. Since we use this strategy to denote views and algorithmic iterations which are not tensor indices, we choose to handle summations with special care by making them explicit via the contraction operator $[.,.]$. Furthermore, we assume an orthogonal representation of the indices, hence in turn, there is no need to make distinction between covariant and contravariant indices.

In the learning problem, we use a linear operator, a tensor, that is an element

of the dual space of $\mathcal{Z}$, the space of linear functionals defined on $\mathcal{Z}$, namely

$$\mathbf{W} \in \mathcal{Z}^*,\ \mathbf{W} = [W_{\mathcal{J}_R}] = [W_{j_1,\ldots,j_{n_R}}],$$

where $\mathcal{Z}^*$ denotes the dual space of all possible linear functionals defined on $\mathcal{Z}$.

We can write up Frobenius type inner products between the linear operator $\mathbf{W}$ and the tensor product of vectors of the views as

$$\langle \mathbf{W}, \bigotimes_{r\in\mathcal{R}} \mathbf{z}_i^r \rangle_F = \sum_{j_1,\ldots,j_{n_R}} W_{j_1,\ldots,j_{n_R}} \prod_{r\in\mathcal{R}} \mathbf{z}_{ij_r}^r. \tag{3}$$

In similar fashion, we can compute the Frobenius norm of $\mathbf{W}$ by

$$\|\mathbf{W}\|_F = \big(\sum_{j_1,\ldots,j_{n_R}} W^2_{j_1,\ldots,j_{n_R}}\big)^{\frac{1}{2}}. \tag{4}$$

In the next step, the set of views is partitioned into two arbitrary parts,

$$\mathcal{R}_X \subset \mathcal{R},\ \mathcal{R}_Y = \mathcal{R} \setminus \mathcal{R}_X,$$

we term the views occurring in $\mathcal{R}_X$ as inputs, and the views in $\mathcal{R}_Y$ can be handled as outputs. Corresponding to either $\mathcal{R}_X$ or $\mathcal{R}_Y$, the set of indices belonging to each view has to be split apart,

$$\mathcal{J}_X \subset \mathcal{J}_R,\ \mathcal{J}_X = \{j_r,\ r \in \mathcal{R}_X\},\ \mathcal{J}_Y = \mathcal{J}_R \setminus \mathcal{J}_X.$$

Fixing a partition, a contraction of $\mathbf{W}$ can be defined as

$$W_{J_Y} = W_{J_R \setminus J_X} = \mathbf{W} \bigotimes_{r\in\mathcal{R}_X} \mathbf{z}_i^r \stackrel{\text{def}}{=} \sum_{j_r\in\mathcal{J}_X} W_{J_R} \prod_{r\in\mathcal{R}_X} \mathbf{z}_{ij_r}^r, \tag{5}$$

where the components of $\mathbf{W}$ are summed only over the input views.

Consequently, the relationship between the inputs and the outputs can be described by the following inner product

$$\langle \bigotimes_{s\in\mathcal{R}_Y} \mathbf{z}_i^s, \mathbf{W} \bigotimes_{r\in\mathcal{R}_X} \mathbf{z}_i^r \rangle_F \stackrel{\text{def}}{=} \sum_{J_Y} \prod_{s\in\mathcal{R}_Y} \mathbf{z}_{ij_s}^s \sum_{J_X} W_{J_R} \prod_{r\in\mathcal{R}_X} \mathbf{z}_{ij_r}^r. \tag{6}$$

It provides a similarity measure between the outputs and the projection of the inputs by the linear operator $\mathbf{W}$. If the norm of $\mathbf{W}$ is fixed, then this inner product takes a greater value if the angle between the direction of the outputs

and the projection of the inputs is smaller, thus the correlation between them is greater. If both the inputs and the outputs are normalised to the same length then this similarity measure implies small distance as well.

Based on these definitions, we can derive a simple but fundamental Lemma:

Lemma 1. *For all partitions $\mathcal{R}_X, \mathcal{R}_Y$ of $\mathcal{R}$ the inner products*

$$\langle \bigotimes_{s\in\mathcal{R}_Y} \mathbf{z}_i^s, \mathbf{W} \bigotimes_{r\in\mathcal{R}_X} \mathbf{z}_i^r \rangle_F$$

have the same value, namely

$$\langle \mathbf{W}, \bigotimes_{r\in\mathcal{R}} \mathbf{z}_i^r \rangle_F .$$

Proof. We need to unfold only the corresponding definitions of the inner products that give the next chain of equalities

$$\begin{aligned} \langle \bigotimes_{s\in\mathcal{R}_Y} \mathbf{z}_i^s, \mathbf{W} \bigotimes_{r\in\mathcal{R}_X} \mathbf{z}_i^r \rangle_F &= \sum_{J_Y} \prod_{s\in\mathcal{R}_Y} \mathbf{z}_{ij_s}^s \sum_{J_X} W_{J_R} \prod_{r\in\mathcal{R}_X} \mathbf{z}_{ij_r}^r \\ &= \sum_{j_1,\ldots,j_{n_R}} W_{j_1,\ldots,j_{n_R}} \prod_{r\in\mathcal{R}} \mathbf{z}_{ij_r}^r \\ &= \langle \mathbf{W}, \bigotimes_{r\in\mathcal{R}} \mathbf{z}_i^r \rangle_F . \end{aligned}$$

Q.E.D.

This Lemma shows that the value of the inner product of the tensor products is invariant on the partitioning of the views into inputs and outputs.

2.1.2. The Optimisation Problem

To force a high similarity between the projected inputs and the outputs taken out of a fixed partition of the views, a "Support Vector Machine"-style, maximum-margin-based optimisation problem is formulated for the regression task. Please note the earlier application of the framework [18, 19]:

$$\begin{aligned} \min \quad & \tfrac{1}{2}\|\mathbf{W}\|_F^2 + C\sum_{i=1}^m \xi_i \\ \text{w.r.t.} \quad & \mathbf{W} \text{ tensor} \in \mathcal{Z}^*,\ \boldsymbol{\xi} \in \mathbb{R}^m, \\ \text{s.t.} \quad & \langle \underbrace{\bigotimes_{s\in\mathcal{R}_Y} \mathbf{z}_i^s}_{\text{Outputs}}, \mathbf{W} \underbrace{\bigotimes_{r\in\mathcal{R}_X} \mathbf{z}_i^r}_{\text{Inputs}} \rangle_F \geq 1 - \xi_i, \\ & \xi_i \geq 0,\ i = 1, \ldots, m, \end{aligned} \tag{7}$$

where $C > 0$ is penalty constant.

The form is similar to the Support Vector Machine case with two notable exceptions: (i) the outputs are no longer binary labels, $\{-1, +1\}$, but vectors of an arbitrary linear vector space, and (ii) the normal vector of the separating hyperplane is reinterpreted as a linear operator projecting the inputs into the space of the outputs.

The regularisation term in the objective function forces the projections of the inputs and the outputs to be similar with respect to their inner products. When the inputs and the outputs are normalised, they live on a sphere in both corresponding spaces, hence we solve the problem by using the structure of Spherical rather than Euclidean geometry.

Based on Lemma 1, we state the next theorem:

Theorem 2. *For all partitions, $\mathcal{R}_X, \mathcal{R}_Y$ of $\mathcal{R}$ the optimisation problem (7) is equivalent to the following one:*

$$\begin{array}{ll} \min & \frac{1}{2}\|\mathbf{W}\|_F^2 + C\sum_{i=1}^m \xi_i \\ \text{w.r.t.} & \mathbf{W} \text{ tensor} \in \mathcal{Z}^*,\ \boldsymbol{\xi} \in \mathbb{R}^m, \\ \text{s.t.} & \langle \mathbf{W}, \bigotimes_{r\in\mathcal{R}} \mathbf{z}_i^r \rangle_F \geq 1 - \xi_i,\ i = 1, \ldots, m, \\ & \xi_i \geq 0,\ i = 1, \ldots, m. \end{array} \tag{8}$$

This equivalence holds true if the inputs and the outputs are partitioned independently for every sample item.

Proof. We can reformulate the constraints by following Lemma 1 that proves the statement. Q.E.D.

This fact guarantees that the linear operator $\mathbf{W}$ has a universal property that *it is independent of the way how the views are grouped into inputs and outputs*, thus it consistently characterises the underlying multi-view learning problem.

The seemingly complex problem represented by (8) can be solved via a simple

Lagrangian dual:

$$\begin{aligned} \min \quad & \tfrac{1}{2}\boldsymbol{\alpha}'\big(\mathbf{K}_1 \bullet \cdots \bullet \mathbf{K}_{n_R}\big)\boldsymbol{\alpha} - \mathbf{1}'\boldsymbol{\alpha} \\ \text{w.r.t.} \quad & \boldsymbol{\alpha} \in \mathbb{R}^m \\ \text{s.t.} \quad & \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}, \end{aligned} \tag{9}$$

where

$$(\mathbf{K_r})_{ij} \quad = \langle \mathbf{z_i^r}, \mathbf{z_j^r} \rangle,\ r \in \mathcal{R},\ i,j \in \{1,\ldots,m\} \tag{10}$$

are kernels corresponding to each of the views. In the formulation of the Lagrangian dual, we exploited the identity given by (2).

The $\bullet$ operator expresses the element-wise, Hadamard, product of matrices. This dual can be solved in a straightforward way for very large scale applications[1]. After the dual variables were computed, the optimum solution for the universal linear operator becomes

$$\mathbf{W} = \textstyle\sum_{i=1}^{m} \alpha_i \bigotimes_{r\in\mathcal{R}} \mathbf{z}_i^r. \tag{11}$$

In the test phase, known and unknown views are considered as inputs and outputs, respectively. The output can be estimated in the following way:

$$\begin{aligned} \big(\bigotimes_{s\in\mathcal{R}_y} \mathbf{z}^s\big) \sim \mathbf{W} \bigotimes_{r\in\mathcal{R}_X} \mathbf{z}^r \quad &= \textstyle\sum_{i=1}^{m} \alpha_i [\bigotimes_{r\in\mathcal{R}} \mathbf{z}_i^r, \bigotimes_{r\in\mathcal{R}_X} \mathbf{z}^r] \\ &= \textstyle\sum_{i=1}^{m} \alpha_i \prod_{r\in\mathcal{R}_X} \langle \mathbf{z}_i^r, \mathbf{z}^r \rangle \bigotimes_{s\in\mathcal{R}_Y} \mathbf{z}_i^s \\ &= \textstyle\sum_{i=1}^{m} \beta_i \bigotimes_{s\in\mathcal{R}_Y} \mathbf{z}_i^s, \end{aligned} \tag{12}$$

where

$$\beta_i = \alpha_i \textstyle\prod_{r\in\mathcal{R}_X} \langle \mathbf{z}_i^r, \mathbf{z}^r \rangle,\ i = 1\ldots,m. \tag{13}$$

Thus, the prediction is a linear combination of the corresponding known outputs.

2.1.3. Computational Complexity of the Proposed Method

For estimation of the computational complexity of the problem presented in (9), one can recognise that (9) is equivalent to the dual problem of an unbiased Support Vector Machine (SVM). Consequently, the problem in (9) can be solved

[1] The website of the authors provides an open source implementation to this problem.

by applying the same type of methods that should have the same complexity depending on the sparsity of the included kernels, see a discussion about this in [20]. The basic task of the SVM is to separate two classes of output data by a hyperplane. The settings of the SVM contain a sample $\{y_i, \mathbf{x}_i\}$, $y_i \in \{-1, +1\}$, $\mathbf{x}_i \in \mathcal{X}(= \mathbb{R}^n)$ and the separating hyperplane which is defined by its normal vector $\mathbf{w}$. Furthermore, input vectors might be embedded into a feature space, $\mathcal{H}_X$, via a function $\phi : \mathcal{X} \to \mathcal{H}_X$, where it is assumed that $\mathcal{H}_X$ is a Hilbert space.

The corresponding primal optimisation problem of the SVM is formulated as

$$\begin{array}{ll} \min & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\mathbf{1}'\boldsymbol{\xi} \\ \text{w.r.t.} & \mathbf{w} \in \mathbb{R}^n, \boldsymbol{\xi} \in \mathbb{R}^m, \\ \text{s.t.} & y_i \mathbf{w}' \phi x_i \geq \mathbf{1} - \xi_i, \\ & \boldsymbol{\xi} \geq \mathbf{0}, i = 1, \ldots, m, \end{array} \tag{14}$$

while the dual problem of the SVM has the form (see for example in [21]),

$$\begin{array}{ll} \min & \frac{1}{2}\boldsymbol{\alpha}'(\mathbf{y}\mathbf{y}' \bullet \mathbf{K}_X)\boldsymbol{\alpha} - \mathbf{1}'\boldsymbol{\alpha} \\ \text{w.r.t.} & \boldsymbol{\alpha} \in \mathbb{R}^m, \\ \text{s.t.} & \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}. \end{array} \tag{15}$$

After introducing the notation $\mathbf{K} = (\mathbf{y}\mathbf{y}' \bullet \mathbf{K}_X)$ in the SVM dual, and similarly $\mathbf{K} = \left(\mathbf{K}_1 \bullet \cdots \bullet \mathbf{K}_{n_R}\right)$ in the dual of the multi-view learning problem (9), we arrive at the same optimisation problem, thus the computational complexity of the proposed learning problem is the same as the complexity of the SVM.

2.1.4. Non-linear Relations

Another consequence of the equivalence of the dual optimisation problems of the proposed method and the SVM is that the kernel trick can be applied here as well (see [21] for more details on background of kernel trick). Since both the dual problem (9) and the prediction (12) contain all input feature vectors as components of the corresponding inner products knowing only a positive semi-definite kernel matrix suffices for carrying out computations in the training and in the prediction as well. More concretely, let all input features – the

representation of the image parts in this application – be implicitly embedded into a feature space by these functions,

$$\boldsymbol{\phi}_r : \mathcal{Z}_r \to \mathcal{H}_r, \ r \in \mathcal{R}_X, \tag{16}$$

where $\mathcal{H}_r$ is a Hilbert space for each r. Then we can write up the elements of the kernels as,

$$(\mathbf{K_r})_{ij} = \langle \boldsymbol{\phi}_r(\mathbf{z}_i^r, \boldsymbol{\phi}_r(\mathbf{z}_j^r) \rangle \,, \ r \in \mathcal{R}_X \,, \ i,j = \{1,\ldots,m\}. \tag{17}$$

For example, Gaussian or Polynomial kernels can be chosen on top of the available linear features.

2.2. Decomposing Images as Tensors

Images expressed by matrices can be represented as a product of the factors computed by Kronecker decomposition. This Kronecker decomposition, after a reordering of the elements of the image matrix, can be carried out by singular value decomposition (SVD). That kind of transformation of images can reveal the structure of the images, e.g., edges, and corners, and can yield a high level compression of the image matrices as well. In case of colour images, tensors can replace the matrices in order to capture the third dimension of the colours.

2.2.1. Kronecker decomposition of matrices

Let us consider a real 2-dimensional (2D) image decomposition of which we can expect that the points nearby within continuous 2D blocks can relate stronger to each other than points in the 1-dimensional rows or columns. A question is, "can the SVD decomposition provide 2D blocks instead of vectors?", achieve this end, a Kronecker decomposition is applied.

The Kronecker product of a matrix $\mathbf{X}$ can be expressed as

$$\mathbf{X} = \mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} A_{1,1}\mathbf{B} & A_{1,2}\mathbf{B} & \cdots & A_{1,n_A}\mathbf{B} \\ A_{2,1}\mathbf{B} & A_{2,2}\mathbf{B} & \cdots & A_{2,n_A}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m_A,1}\mathbf{B} & A_{m_A,2}\mathbf{B} & \cdots & A_{m_A,n_A}\mathbf{B} \end{bmatrix} \tag{18}$$

where $\mathbf{A} \in \mathbb{R}^{m_A \times n_A}$, $\mathbf{B} \in \mathbb{R}^{m_B \times n_B}$, $m_X = m_A \times m_B$, and $n_X = n_A \times n_B$.

In the Kronecker decomposition, the second component, $\mathbf{B}$, might be interpreted as a 2D filter of the image represented by the matrix $\mathbf{X}$. We can try to find a sequence of those filters by applying the procedure presented in Algorithm 1.

Algorithm 1 Calculate the Kronecker decomposition of a matrix

Require: matrix $\mathbf{X}$, number of iteration n,

Require: size of $\mathbf{A} \in \mathbb{R}^{m_A \times n_A}$, size of $\mathbf{B} \in \mathbb{R}^{m_B \times n_B}$

Ensure: $(\mathbf{A}^{(1)}, \mathbf{B}^{(1)}), \ldots, (\mathbf{A}^{(n)}, \mathbf{B}^{(n)})$

1: $\mathbf{X}^{(1)} = \mathbf{X}$

2: **for** $k = 1$ to n **do**

3: $\quad \mathbf{A}^{(k)}, \mathbf{B}^{(k)} = \arg\min_{\mathbf{A},\mathbf{B}} \|\mathbf{X}^{(k)} - \mathbf{A} \otimes \mathbf{B}\|^2_{Frobenius}$

4: $\quad \mathbf{X}^{(k)} = \mathbf{X}^{(k)} - \mathbf{A}^{(k)} \otimes \mathbf{B}^{(k)}$, ## deflation

5: **end for**

The question is: given $\mathbf{X}$, how can we compute the optimum solution $\mathbf{A}$ and $\mathbf{B}$ for the problem,

$$\min_{\mathbf{A},\mathbf{B}} \|\mathbf{X}^{(k)} - \mathbf{A} \otimes \mathbf{B}\|^2_{Frobenius}? \tag{19}$$

2.2.2. Kronecker Decomposition as SVD

We can make an important observation that the algorithm solving the tensor decomposition problem does not depend directly on the order of the elements of the matrix, thus a permutation of the indexes, i.e. reordering of the columns and/or the rows, preserves the same solution. Based on this observation, the Kronecker decomposition can be computed via the SVD (see [22] for more details). An example that illuminates how the reordering of the matrix $\mathbf{X}$ can solve the Kronecker decomposition problem is demonstrated here:

The matrices in the Kronecker product

$$\underbrace{\left[\begin{array}{cc|cc|cc} x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} \\ x_{21} & x_{22} & x_{23} & x_{24} & x_{25} & x_{26} \\ \hline x_{31} & x_{32} & x_{33} & x_{34} & x_{35} & x_{36} \\ x_{41} & x_{42} & x_{43} & x_{44} & x_{45} & x_{46} \\ \hline x_{51} & x_{52} & x_{53} & x_{54} & x_{55} & x_{56} \\ x_{61} & x_{62} & x_{63} & x_{64} & x_{65} & x_{66} \end{array}\right]}_{\mathbf{X}} = \underbrace{\left[\begin{array}{ccc} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{array}\right] \otimes \left[\begin{array}{cc} b_{11} & b_{12} \\ b_{21} & b_{22} \end{array}\right]}_{\mathbf{A} \otimes \mathbf{B}},$$

can be reordered into

$$\underbrace{\left[\begin{array}{ccccccccc} x_{11} & x_{13} & x_{15} & x_{31} & x_{33} & x_{35} & x_{51} & x_{53} & x_{55} \\ x_{12} & x_{14} & x_{16} & x_{32} & x_{34} & x_{36} & x_{52} & x_{54} & x_{56} \\ x_{21} & x_{23} & x_{25} & x_{41} & x_{43} & x_{45} & x_{61} & x_{63} & x_{65} \\ x_{22} & x_{24} & x_{26} & x_{42} & x_{44} & x_{46} & x_{62} & x_{64} & x_{66} \end{array}\right]}_{\tilde{\mathbf{X}} = \tilde{\mathbf{A}} \otimes \tilde{\mathbf{B}}},$$

$$= \left[\begin{array}{c} b_{11} \\ b_{12} \\ b_{21} \\ b_{22} \end{array}\right] \otimes \left[\begin{array}{ccccccccc} a_{11} & a_{12} & a_{13} & a_{21} & a_{22} & a_{23} & a_{31} & a_{32} & a_{33} \end{array}\right]$$

where the blocks of $\mathbf{X}$ and the matrices $\mathbf{A}$ and $\mathbf{B}$ are vectorised in a row-wise order.

We observe that $\tilde{\mathbf{X}} = \tilde{\mathbf{A}} \otimes \tilde{\mathbf{B}}$ can be interpreted as the first step in the SVD algorithm where we might apply the substitutions $\sqrt{s}\mathbf{u} = \tilde{\mathbf{A}}$ and $\sqrt{s}\mathbf{v} = \tilde{\mathbf{B}}$. The proof that this reordering provides a correct solution to the Kronecker decomposition can be found in [22].

The main steps of the Kronecker decomposition can be summarised as follows:

1. Reorder (reshape) the matrix,
2. Compute the SVD decomposition,
3. Compute the approximation of $\tilde{\mathbf{X}}$ by $\tilde{\mathbf{A}} \otimes \tilde{\mathbf{B}}$

4. Invert the reordering.

This kind of Kronecker decomposition is often referred as the Nearest Orthogonal Kronecker Product as well [22].

We can extend this procedure to higher order tensors represented by higher order arrays (see the review paper of [23]). Then, this method can be applied on the following objects:

- colour images, where three matrices express the RGB layers, a tensor of order 3, e.g. $1024 \times 1024 \times 3$,
- video stream of grey-scale images, where the third dimension is time,
- video stream of colour images, where the third dimension is colour, and the fourth dimension is time.

The Kronecker decomposition presented above can be extended further to include more than two factors (more details, alternative approaches and applications can be found in [23]). The Kronecker decomposition algorithm, as shown in Algorithm 1), implements a non-linear polynomial approximation of the target matrix or a higher order tensor. The degree of the applied polynomial is equal to the number of included factors.

2.2.3. A Set Theoretic Approach to Reordering

A matrix representation as an ordered collection of elements can be reinterpreted by using the language of set theory. Let $\mathbf{X}$ be a matrix with size $m \times n$. For the sake of simplicity, we assume that the elements $[X_{ij}]$ of $\mathbf{X}$ are real numbers. The structure of the matrix $\mathbf{X}$ can be described as a set $\mathcal{X}$ of the elements $\{X_{ij}\}$ with cardinality mn on which two equivalence relations, $\mathscr{R}_r$ and $\mathscr{R}_c$, are imposed, one based on the rows and the other based on the columns, i.e. two elements are equivalent with respect to $\mathscr{R}_r$ if they are in the same row, and equivalent with respect to $\mathscr{R}_c$ if they are in the same column.

Now, the reordering of the matrix elements $\mathcal{X}$ is expressible by imposing another kind of equivalence relations that classify the elements into different

classes. These equivalence relations have to satisfy the following rules: all of the equivalence classes within the relation $\mathscr{R}$ have to have the same cardinality. The equal cardinality rule implies that the product of the number of the equivalence classes $N_{\mathscr{R}}$ and the common cardinality of the classes, $\text{card}_{\mathscr{R}}$, is equal to the cardinality of $\mathcal{X}$, namely to mn. Then, the reordering of the matrix elements can be carried out by sorting the equivalence classes into columns and rows of a new matrix.

Note that the equivalence relation $\mathcal{R}$ can be decomposed further into a series of equivalence relations $(\mathcal{R}_1, \ldots, \mathcal{R}_N)$. This decomposition can be realised in a recursive way.

1. Let the set of classes of $\mathcal{R}_1$ be given by $\mathcal{C}(\mathscr{R})_1, \ldots, \mathcal{C}(\mathscr{R})_{N_{R_1}}$, with common cardinality card_{R_1}.

2. For every class of relations, $\mathscr{R}_k$ apply the same type of equivalence relations, $\mathscr{R}_{k+1}$. Since the cardinality of the classes of $\mathscr{R}_k$ is the same, this relation can be applied uniformly. Consequently, the cardinality of the classes and the number of the classes in $\mathscr{R}_{k+1}$ are the same for all classes of $\mathscr{R}_k$.

By reversing the recursive classification of the elements, we can build a tensor $\mathbf{T}$ of order $N+1$ by starting on the classes of the $\mathscr{R}_N$.

Since the internal structure of the matrix $\mathbf{X}$ is not directly exploited, only the set of its elements are classified. The reordering procedure described above can be applied on any tensor of arbitrary order. Thus, any tensor can be reordered into another tensor of arbitrary order. An example of order one tensor, vectorisation, can be given by an equivalence relation where all elements fall into the same equivalent class.

This interpretation of the reordering leads us to the realm of *Algebraic Statistics*, and within that realm, to the *Combinatorial Design Theory* [24, 25]. The Combinatorial Design Theory addresses the problem of how to build systems of finite sets that satisfy certain requirements of symmetries as stated in the

classical theory of *Latin squares*. These theories play a very important role in creating balanced statistical experimental designs, especially for medical tests.

2.2.4. Compression

Furthermore, the tensor decomposition can provide a very high level compression of images. Some examples are presented here. The compression ratio is computed by dividing the number of elements of the image matrix with the total number of elements of the components in the decomposition.

The original image matrix is given by integers in the range of $0, \ldots, 255$ for both grey-scale and RGB colour images. The decomposition happens in the space of real numbers, but after having been decomposed, the real numbers in the components can be rescaled and transformed into the original integer interval.

Let the size of a grey-scale image be equal to $(1024, 1024)$, then we can have various patterns of decompositions as shown in Table 1. The components

Component Size	Singular Values	Full Size	Compression Ratio
$(32, 32), (32, 32)$	10	$s = 10 * 2 * 32^2$	$\frac{1024^2}{s} = 51.2$
$(16, 16), (16, 16), (4, 4)$	20	$s = 20 * (2 * 16^2 + 4^2)$	$\frac{1024^2}{s} = 99.29$

Table 1: Examples of possible patterns of decomposition of a grey-scale image with 1024×1024.

provide a tightly-compressed signature for the image as illustrated in Figure 3.

The Kronecker-decomposition-based features can be compared with those from a conventional compression technique with SVD and the well-known SIFT features [26]. Let the size of an RGB colour image be $(640, 960, 3)$ and the sizes of the components in the tensor decomposition be $(10, 15, 1), (8, 8, 1), (8, 8, 3)$, and let 45 singular values be computed, then $45*(10*15+8^2+3*8^2)$ 8-bit data items can be obtained with a high compression ratio of ~ 100. This is equivalent to (i) using image compression with SVD with 4 singular values which gives $((640*4)+4+(940*4))*3$ data items and (ii) ~ 36 SIFT feature vectors with 128 real valued

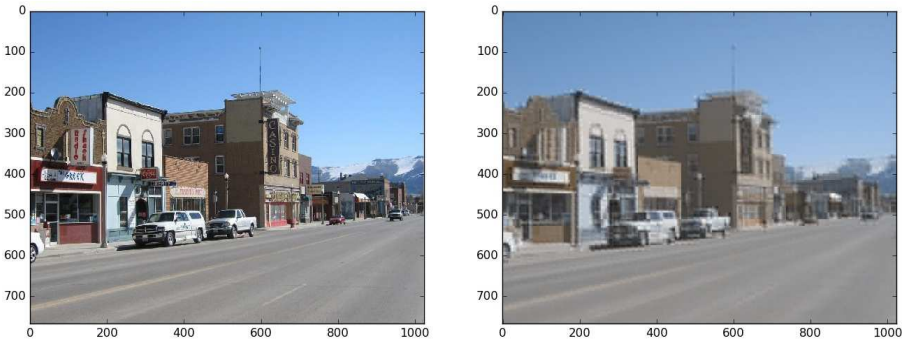


Figure 3: The right image is a recovered image from the Kronecker decomposition of the original image on the left. The factors were $(48, 64, 1), (16, 16, 3)$ with 6 singular values. The compression ratio was 102.4.

(a) Image without compression

(b) Image compressed by Kronecker decomposition

(c) Image compressed by SVD

Figure 4: An example of image compressed by the proposed and a conventional technique when the compression ratio was 100.

components, where we assume that these real numbers can be represented by 32 bits. The tensor decomposition can recover a good approximation of the original image and the colour information while the conventional technique cannot do so, as shown in Figure 4. Although the 36 SIFT points may be able to capture some particular characteristic points, they cannot recover the main structure of the entire image at all.

2.2.5. Interpretation of Image Components

The components provided by the tensor decomposition can be endowed with a practical interpretation. Let us discuss the two-component case where the

image matrix $\mathbf{X}$ is expressed as a Kronecker product of two other matrices $\mathbf{A}$ and $\mathbf{B}$. Since the second factor $\mathbf{B}$ in the product is forced to be the same for all positions, it has to be equal to a nonlinearly aggregated matrix. This factor is shifted around within the image and it is only scaled by elements of the matrix $\mathbf{A}$. Thus, $\mathbf{B}$ can be interpreted as an image filter.

In Figure 5, the first level components belonging to the sequence of decreasing singular values are of the compressed image shown in Figure 3. This image is factorised into two levels and the corresponding low level components are presented in Figure 6. Some characteristic patterns can be recognised from these factors. The factor belonging to the second singular vector represents a horizontal edge filter and the factor belonging to the third singular value yields a vertical line filter. In this way, the image decomposition finds the boundaries of the critical regions, edges, and corners in which most of the structural information concentrates, and as mentioned earlier, it also produces a very highly compressed skeleton of the data.

Since in every step the decomposition processes the residue of the previous step, it predicts those parts that have hardly been approximated earlier, thus in every step a new layer of the structure is discovered. In images, these layers are first the flat areas, then edges of different directions–vertical, horizontal, and slant–corners of different kind, and the higher order singularities of the intensity surface. This kind of incremental approach resembles a boost in which hardly predictable sample items receive larger weights.

2.2.6. Relation to Known Saliency Detection Approaches

The generally applied saliency detectors are built on estimated derivatives of the intensity function, I, of an image (see for example [27]). These derivatives can provide information about the locations that the intensity changes significantly faster than a given threshold. The general first order edge and corner detection algorithms consist of two phases: the first phase uses a Gaussian filter to smooth out the noise caused by non-differentiable representation of the image, and the second phase estimates the gradients of the image intensity. The

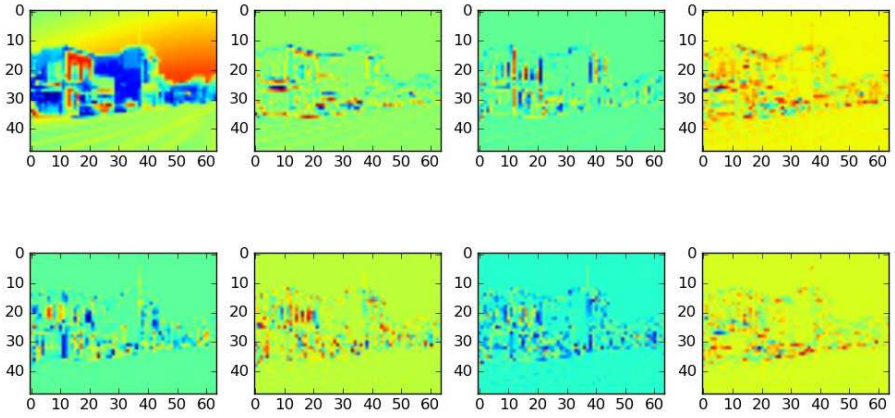


Figure 5: First level of the Kronecker decomposition in case of the first 8 largest singular values.

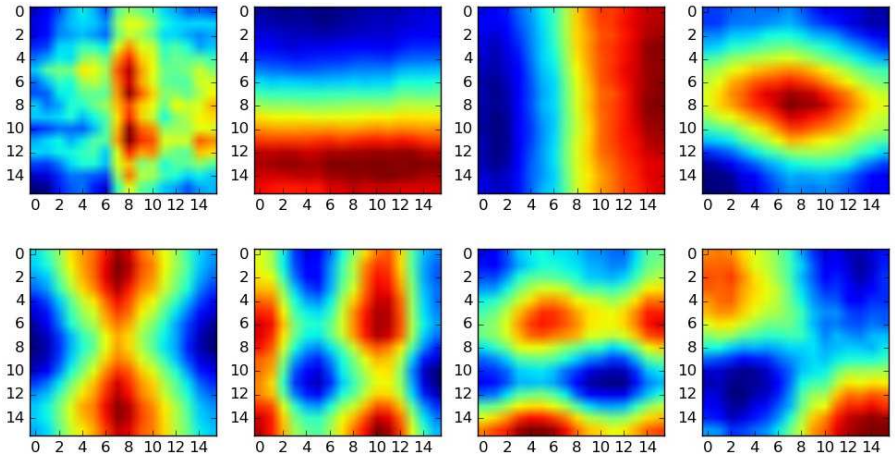


Figure 6: Second level of the Kronecker decomposition in case of the first 8 largest singular values.

higher order methods also exploit the second derivative, the Hessian, of the intensity image. Estimation of the derivatives can be performed by applying image filters on the image intensity, e.g. Laplacian, Sobel and Prewitt.

For example *Harris-Laplace detector* relies on the first order partial derivatives, I_x, I_y, of the intensity measured in the local neighbourhood of a given image point

$$\mathbf{A}(x) = \sum_{i,j} w_{ij} \begin{bmatrix} I_x^2(\mathbf{x}) & I_x I_y(\mathbf{x}) \\ I_x I_y(\mathbf{x}) & I_y^2(\mathbf{x}) \end{bmatrix}, \; R = \det(\mathbf{A}) - \alpha \, \mathrm{trace}^2(\mathbf{A}).$$

Similarly the *Hessian affine region detector* exploits the second partial derivatives of the Gaussian-filtered intensity, the Laplacian of the Gaussian (LoG) representation,

$$\mathbf{H}(x) = \begin{bmatrix} L_{xx}(\mathbf{x}) & L_{xy}(\mathbf{x}) \\ L_{xy}(\mathbf{x}) & L_{yy}(\mathbf{x}) \end{bmatrix}, \; L(x) = g(\cdot|0, \sigma_I) \otimes I(x), \; LoG = \mathrm{trace}(\mathbf{H}),$$

where L is the Gaussian-filtered intensity, and g is a Gaussian filter with a parameter σ_I.

The Kronecker decomposition based algorithm iteratively reproduces the partial derivatives of the intensity function via higher order polynomial approximation of successive error terms. During this iterative procedure, the algorithm automatically selects, in the least-square sense, the best fitting filters as the last factor of the Kronecker product. In this way, it gives an approximation of the *multivariate Taylor series* of the intensity image that provides an estimation of the full series of those partial derivatives that are exploited in well-known saliency detectors. It is also important to mention that the global optimality of the Kronecker decomposition allows us to select only regions where the derivatives change the most and eliminate flat, redundant ones. This selective property leads to high level compression of the information needed to describe the variation of the intensity. For example, Harris detector based methods focus only on local features without forcing globally optimal extraction.

2.3. Combining Images with Eye Movements

The images processed in our eye movement experiments were assumed to be RGB colour images represented by 3 parallel matrices and indexed by row and column coordinates of the pixels in the images. Since the sizes of the images varied, all of them were transformed into the same common size in the learning procedure. That common size was 50×50.

The eye movements in all of the experiments were given as a list of coordinate pairs consisting of the image related row and column indices. When the eyes moved beyond the image, those eye movement coordinates were represented by invalid, e.g., negative values. Therefore, the coordinate pairs of the eye movement had to be filtered to extract only those points that corresponded to valid image pixels. When the sizes of the images were transformed to the same size, that transformation applied to the eye movement coordinates as well.

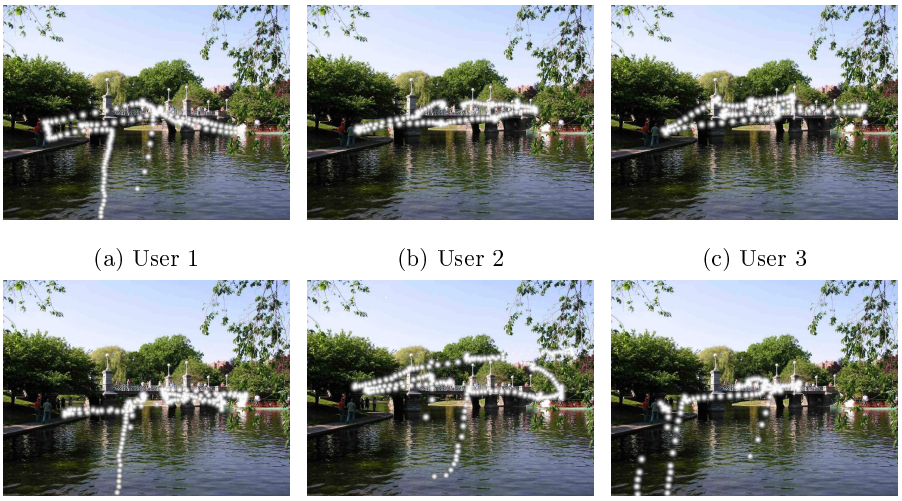
When eye movement coordinates were combined with an image, we needed to deal with the uncertainty of eye positions caused by measurement errors and movements of the eyes and the head that were probably not related to the image. That uncertainty was handled by applying Gaussian smoothing, a spatial filter, to each point of eye movement. The Gaussian filter was centred at a point $\mathbf{u}$ with a width parameter σ assigns to each image point $\mathbf{x}$,

$$g(\mathbf{x}|\mathbf{u}, \sigma) = e^{-\frac{\|\mathbf{x}-\mathbf{u}\|^2}{2\sigma}}. \tag{20}$$

The centres of the Gaussian filters were localised at the observed points of the eye movements. Since all image points could be connected by the filter to all eye movement points, as many filter values were assigned to each image point as the number of observed eye movement points. To aggregate the filter values, we applied the following function on all of the image points $\mathbf{x}$:

$$g_A(\mathbf{x}) = \max_{\mathbf{u}} g(\mathbf{x}|\mathbf{u}, \sigma). \tag{21}$$

The function g_A assigned to each point of the image, $\mathbf{x}$, a Gaussian filter value that belonged to the closest point of the eye movement. That Gaussian filter can well represent foveation in human vision [28].



(a) User 1 (b) User 2 (c) User 3

(d) User 4 (e) User 5 (f) User 6

Figure 7: The eye movements of six different users on a sample image.

In Figure 7, the eye movement trajectories on a stimuli image of six contributing users are presented. We were able to have some impression on the similarity and the variation of the responds of the users looking at the same image and trying to perform the same task.

Figure 8 shows images that were already merged with eye movements data. These images would subsequently be decomposed by a Kronecker product. The decomposition of the Kronecker product on the image and the combined images with eye movement data are shown in Figure 9.

3. Performance Evaluations and Discussions

We evaluated our TMVL algorithm on a publicly available eye tracking dataset [4]. The dataset contains eye tracking data of 15 different users on 1003 images. Each image consists of three-second free-view trajectories of different users. In order to encourage users to pay attention to the task, they were memory-tested at the end of the data collection of 100 images.

In our experiment, only eight users were randomly selected; hence, there

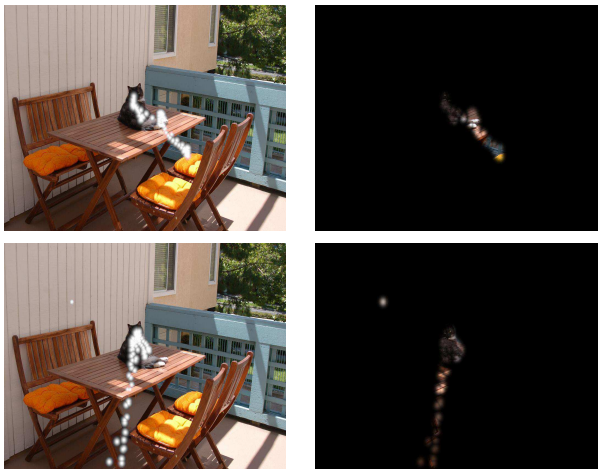


Figure 8: Eye movement data overlaid on the image (left) and images merged with eye movement data (right) of two users.

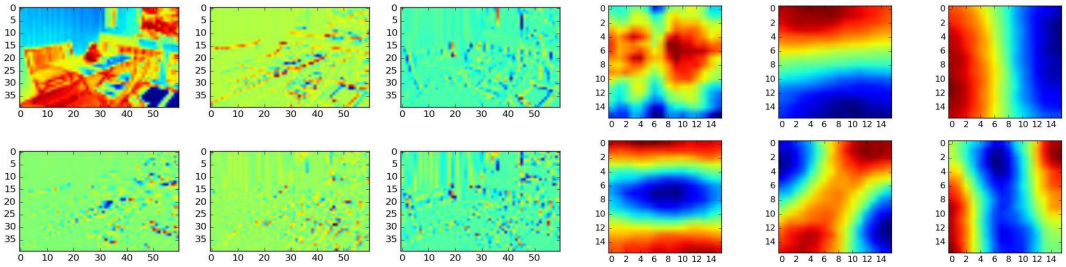
were eight views in this setting. Each view was represented by a heatmap of each user. Heatmap quantifies the degree of importance of parts of image; a higher probability of an important part is implied by a higher density of eye movements on that part. We investigated two sets of heatmap input features.

1. Heatmap generated from eye movement data alone (H_E): A user's heatmap was created by convolving a Gaussian kernel with each eye fixation point.

2. Heatmap generated from eye movement data and the image (H_{EI}): This set of features was generated by a Kronecker decomposition of an image combined with eye movement data as described in subsection 2.3 with σ of 2. We extracted features with two components of Kronecker product from a grey scale processed image. The sizes of the high (first) ($\mathbf{A}$) and low (second) ($\mathbf{B}$) level component were (10,10), and (5,5), respectively with 20 singular values. The feature vector became $\begin{bmatrix} \mathbf{A}_1 \otimes \mathbf{B}_1 & \mathbf{A}_2 \otimes \mathbf{B}_2 & \dots & \mathbf{A}_{20} \otimes \mathbf{B}_{20} \end{bmatrix}$.

All heatmaps were normalised to unit norm. To compare prediction performances, we used three performance matrices as follows:

1. Area under the receiver operating characteristic (AUROC) is one of com-



Stimuli image

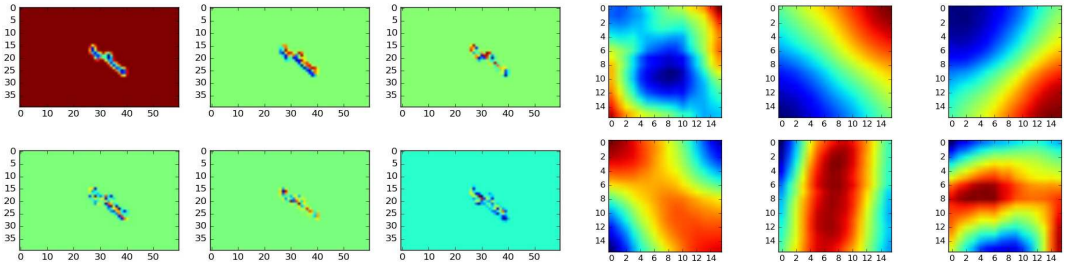


Image merged with eye movement data of User 1

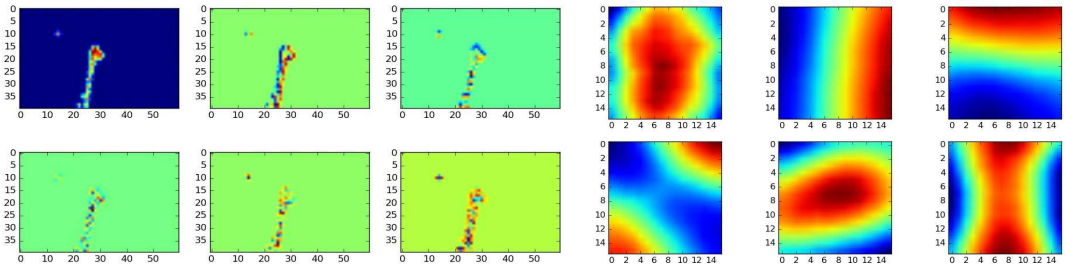


Image merged with eye movement data of User 2

Figure 9: Kronecker decomposition of the original stimuli image and the corresponding eye movement data of two users; two Kronecker factors and six singular values were computed

monly used performance metrics for comparing heatmaps and eye fixations[2]. It is based on an ROC curve that can be computed by varying the threshold of the predicted heatmap. A pixel is predicted as a target when its heatmap value is greater than the threshold, while it is classified as a background when the value is below the threshold. AUROC ranges from 0 (complete mismatch) to 1 (perfect match).

2. Correlation indicates the degree of linear relationship between two maps. It ranges from -1 (perfect correlation but in the opposite direction) to $+1$ (perfect correlation). Zero indicates no correlation between the two maps.

3. Jensen-Shannon divergence (JSD) was used to identify the dissimilarity between two distributions. It is based on Kullback-Leibler divergence (KLD) which can capture a certain kind of non-linear, entropy type and dependency. KLD is defined as,

$$D_{KL}(P||Q) = \sum_{i=1} \ln\left(\frac{P_i}{Q_i}\right) P_i \tag{22}$$

where P, Q are the probability distributions. JSD is symmetric while KLD is not [29]. Square root of JSD has matrix properties. The more similar the two objects are, the smaller the value of JSD is, and vice versa. JSD is defined as,

$$D_{JS}(P||Q) = \frac{1}{2} D_{KL}(P||M) + \frac{1}{2} D_{KL}(Q||M) \tag{23}$$

where $M = \frac{1}{2}(P + Q)$.

Here, linear kernel function was used. Model selection was performed with five-fold cross validation based on AUROC. We examined two scenarios of the test sets: (i) {1-7} missing views were randomly selected and (ii) one fixed view was missing for each run. The experiments were run 10 times with different random data splits.

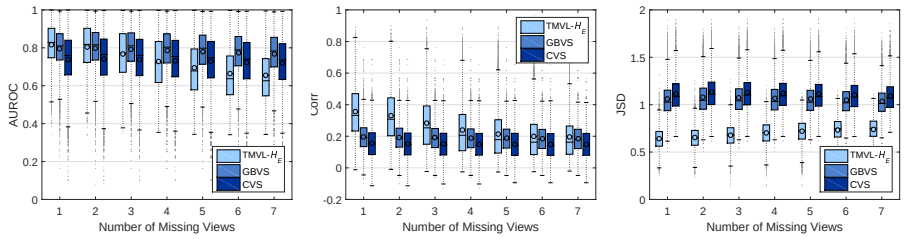
[2]Available for download at http://www.vision.caltech.edu/~harel/share/gbvs.php.

Although our scenario is of a supervised learning approach, but its setting is different from those mentioned earlier in the introduction. All previous works used a saliency map of average locations fixated by all users as a global model. In our scenario, we instead used individual fixation maps and focused on user models. Our assumption was that there were some correlations between users' eye movement behaviours. We aim to predict the individual maps. To the best of our knowledge, there was no available technique designed for this kind of scenario, so we compared our proposed method with baseline methods in a similar way that we did in one of our previous works [30]. The baseline methods were two well-known saliency map models: Conventional Visual Saliency (CVS) [1], and Graph-Based Visual Saliency (GBVS) [2].

3.1. Randomly Select Missing Views

When the number of missing views increased, AUROC and correlation decreased for all algorithms as shown in Figure 10a and 10b, respectively. On the other hand, JSD increased when there was an increase in the number of missing views as shown in Figure 10c. When the number of missing views increased, TMVL-H_E performance was dramatically reduced while GBVS and CVS's performances were slightly decrease. TMVL-H_E only used eye movement data, therefore, the prediction performances highly depended on number of available view, while GBVS and CVS used image information. TMVL-H_E's performances on AUROC were better than those on GBVS when there were 1–2 missing views and better than those on CVS when there were 1–3 missing views. Unfortunately, TMVL-H_E performances on AUROC were worse than those on GBVS and CVS in other cases as shown in Figure 10a. However, TMVL-H_E was still able to achieve better average correlation and JSD than GBVS and CVS were in all cases as shown in Figure 10b and 10c, respectively.

Figure 11 shows an example of prediction by GBVS, CVS, and our proposed algorithm when two views were missing. It can be seen that GBVS and CVS failed to predict where users were looking at. Both algorithms put attention on the woman's arm and the windows while users actually focused on her face.



(a) AUROC (b) Correlation (c) JSD

Figure 10: A boxplot comparison of all methods when randomly selected {1-7} missing views were considered.

User	Method	AUROC	Corr	JSD
1	TMVL	**0.8065**	**0.7160**	**0.3726**
	GBVS	0.6543	0.0783	1.2755
	CVS	0.7358	0.0792	1.2454
2	TMVL	**0.7812**	**0.6900**	**0.4064**
	GBVS	0.6707	0.0985	1.1863
	CVS	0.6983	0.0981	1.1627

Table 2: Performance matrices of all of the methods on the image in figure 11. Bold values indicate the best performance achieved in each user.

Clearly, TMVL was more effective than GBVS and CVS for all three performance matrices as shown in Table 2,

We improved prediction performance by using H_{EI} as feature vectors and compared the prediction results with those from other methods, as shown in Figure 12. TMVL-H_{EI} outperformed CVS in all cases but was only better than GBVS in {1–4}-missing-view cases. TMVL was comparable to GBVS when five views were missing but was worse than GBVS in the case of {6–7} missing views. It should be noted that we only evaluated the performances on AUROC as it compared the predicted heatmaps directly with eye movement data for all of the methods. Correlation and JSD compare between predicted heatmaps to target heatmaps but target heatmaps for TMVL with H_E and with H_{EI}

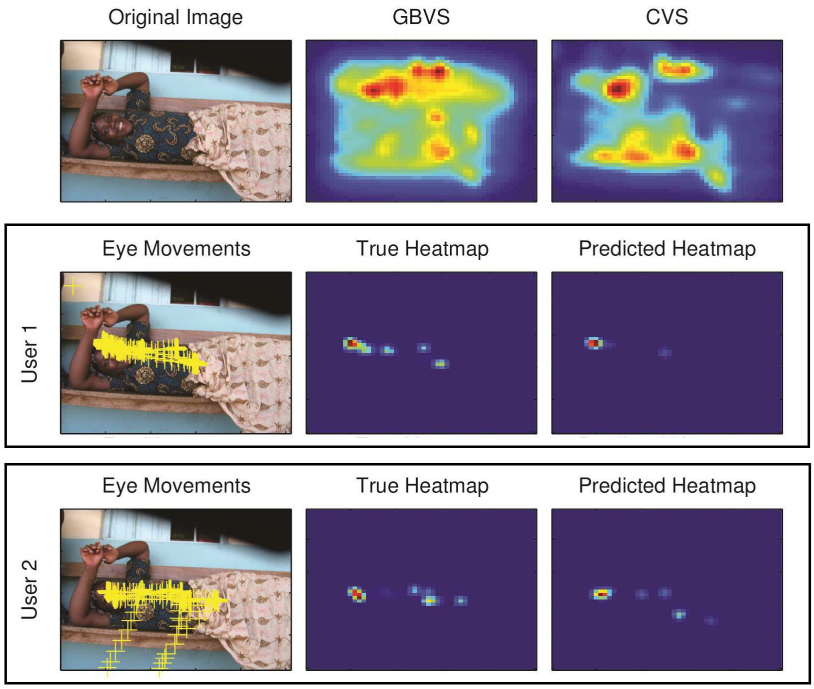


Figure 11: True and TMVL-predicted heatmaps of an eight-view problem with two views missing (User 1 and 2) compared to those predicted by baseline methods–GBVS and CVS.

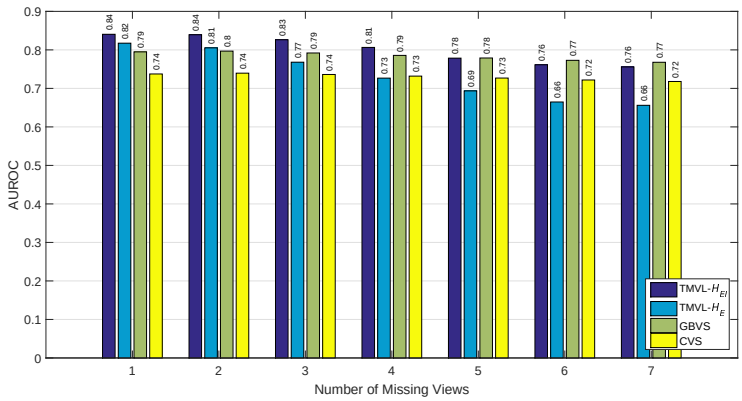


Figure 12: A comparison between the proposed method and two existing saliency prediction methods in the case of randomly selected {1–7} missing views

were different. Hence, we were not able to directly compare these matrices. It is clear that using H_{EI} that combined images with eye movement information could dramatically enrich the performance on AUROC over using H_E as shown in Figure 13. AUROC was improved at 2.82% in 1-missing-view case and up to 15.20% in seven-missing-view case.

Figure 14 shows an example of predictions by all of the methods when two views were missing. Again, GBVS and CVS failed to predict that the cat on the table would be favourably looked at rather than chairs with light shone on them. Both TMVL with H_E and H_{EI} were able to detect the cat; however, TMVL-H_{EI} was able to capture more eye movements than TMVL-H_E did, especially for User 2.

3.2. Fixing A Missing View

In this scenario, we wanted to predict where a user would be looking at in a (test) set of images based on where the other seven users were looking at in the same set. According to Table 3–Table 5, TMVL was the best contender for all users, followed by GBVS and CVS in that order. Using H_{EI} as input features could improve prediction performance significantly for all cases and on all matrices.

Method	User Index							
	1	2	3	4	5	6	7	8
TMVL-H_{EI}	**0.8550**	**0.8631**	**0.7939**	**0.8300**	**0.9010**	**0.8468**	**0.8418**	**0.7901**
TMVL-H_E	0.8349	0.8508	0.7690	0.8079	0.8711	0.8246	0.8184	0.7670
GBVS	0.8040	0.7848	0.7638	0.8028	0.8410	0.7888	0.8025	0.7636
CVS	0.7426	0.7164	0.7146	0.7517	0.7708	0.7257	0.7457	0.7171

Table 3: A comparison between all of the methods with AUROC when only one fixed view/user are missing. Bold values indicate the best AUROC achieved in each user.

Feature	Method	User Index							
		1	2	3	4	5	6	7	8
H_{EI}	TMVL	0.4147	0.4747	0.3568	0.4046	0.4569	0.4266	0.4154	0.3657
	GBVS	0.2310	0.2319	0.2356	0.2736	0.2259	0.2432	0.2660	0.2645
	CVS	0.1762	0.1757	0.1829	0.2114	0.1727	0.1829	0.2045	0.2097
H_E	TMVL	0.3276	0.4433	0.2951	0.3437	0.4013	0.3874	0.3706	0.2861
	GBVS	0.1835	0.1932	0.1856	0.2083	0.1928	0.1981	0.2135	0.1976
	CVS	0.1425	0.1465	0.1476	0.1662	0.1501	0.1526	0.1675	0.1604

Table 4: A comparison between all of the methods with correlation when only one fixed view/user are missing.

Feature	Method	User Index							
		1	2	3	4	5	6	7	8
H_{EI}	TMVL	0.6324	0.5279	0.5152	0.4866	0.6234	0.5364	0.4998	0.4095
	GBVS	0.7442	0.7046	0.5912	0.5893	0.8102	0.6735	0.6193	0.4921
	CVS	0.7625	0.7171	0.5927	0.5968	0.8343	0.6844	0.6277	0.4833
H_E	TMVL	0.6792	0.5463	0.7011	0.6606	0.6055	0.6083	0.6269	0.7003
	GBVS	1.1585	1.1138	1.0224	1.0024	1.1760	1.0451	0.9996	0.9582
	CVS	1.2184	1.1753	1.0757	1.0595	1.2376	1.1056	1.0587	1.0103

Table 5: A comparison between all of the methods with JSD when only one fixed view/user are missing.

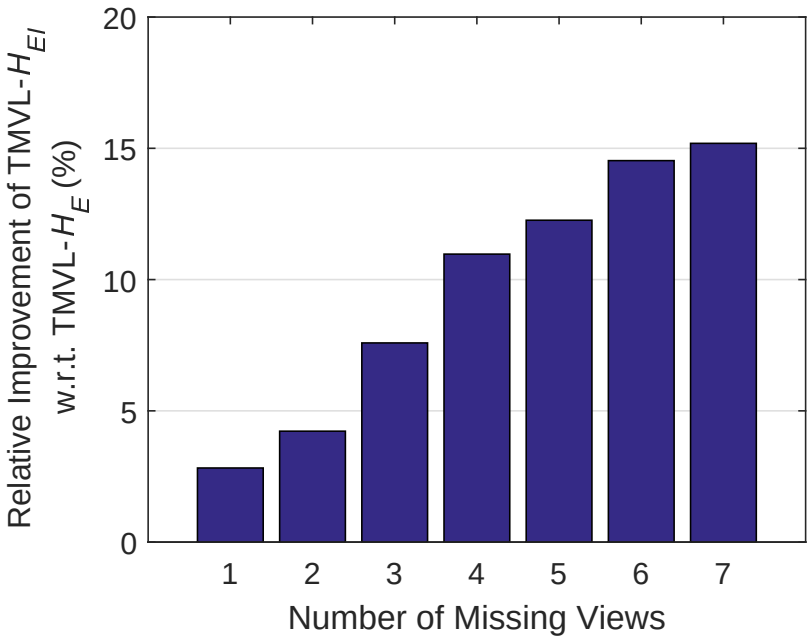


Figure 13: Relative improvement of AUROC using H_{EI} compared to using H_E in TMVL

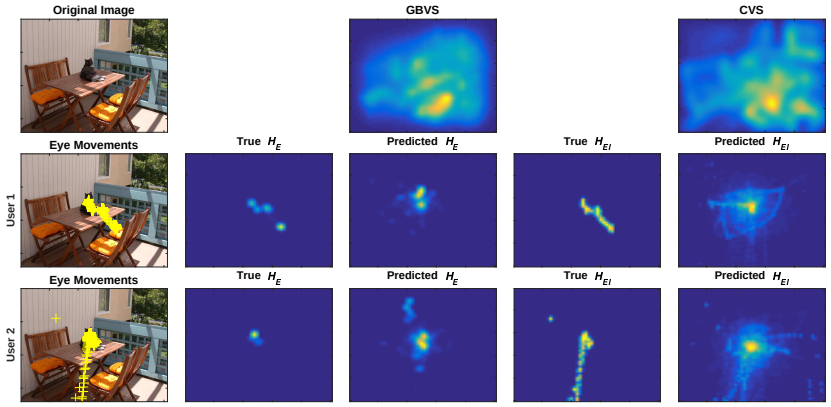
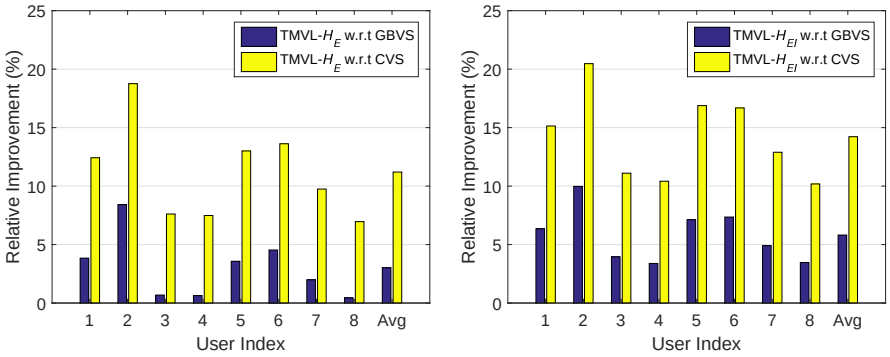


Figure 14: A comparison of predicted heatmaps by all of the considered methods.

Compared to GBVS and CVS, TMVL-H_E showed an improvement with AUROC for all users at 3.01% and 11.20% on the average, respectively, as shown in Figure 15a. TMVL-H_{EI} performance was improved much more, 5.81%



(a) H_E (b) H_{EI}

Figure 15: Relative improvement of TMVL compared to GBVS and CVS when only one fixed view/user was missing.

against GBVS and 14.22% against CVS on the average, as shown in Figure 15b. Our proposed algorithm achieved the highest improvement compared to both baselines for User 2. This means that other users' eye movements behaviours were very useful to User 2. However, the least improvement was found for User 8. This indicates that user adaptation can be useful as the performance was improved even though not so much for the case of User 8. Using TMVL-H_{EI} yielded better AUROC, at 2.73% on the average than TMVL-H_E did, as shown in Figure 16.

4. Conclusion

In this paper, we introduced a missing-value prediction schema built upon maximum-margin-based learning and invariances of tensor algebra. Our proposed algorithm was tested on an eye movement dataset in order to identify where users were looking at in images. We also proposed a new technique that decomposes images called "Kronecker Decomposition". This technique can be used in image compression applications. We also proposed an approach to combine image with eye movement data. The processed image is then decomposed by using Kronecker decomposition. We have demonstrated that our framework

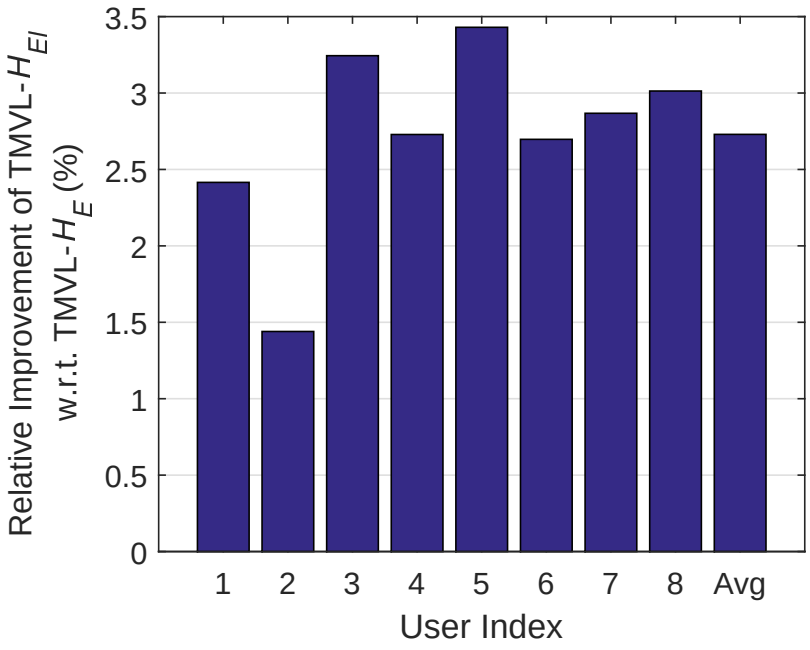


Figure 16: Relative improvement of TMVL-H_{EI} compared to TMVL-H_E when only one fixed view/user was missing.

was able to perform better than two well-known saliency detection techniques. Several initial results show that user adaptation may be useful; thus, user information should be investigated further in future research.

Acknowledgement

This work was supported by the Thailand Research Fund under grant agreement number TRG5680090.

References

[1] L. Itti, C. Koch, E. Niebur, A model of saliency-based visual attention for rapid scene analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (1998) 1254–1259.

[2] J. Harel, C. Koch, P. Perona, Graph-based visual saliency, in: Advances in Neural Information Processing Systems, 2006, pp. 545–552.

[3] A. Borji, M.-M. Cheng, H. Jiang, J. Li, Salient object detection: A survey, arXiv 1411.5878 (2014).

[4] T. Judd, K. Ehinger, F. Durand, A. Torralba, Learning to predict where humans look, in: Proceeding of the IEEE 12th International Conference on Computer Vision, 2009, pp. 2106–2113. doi:10.1109/ICCV.2009.5459462.

[5] J. M. Henderson, J. R. Brockmole, M. S. Castelhano, M. Mack, Visual saliency does not account for eye movements during visual search in real-world scenes, Eye Movements: A Window on Mind and Brain (2007) 537–562.

[6] Q. Zhao, C. Koch, Learning a saliency map using fixated locations in natural scenes, Journal of Vision 11 (2011) 1–15.

[7] M. Liang, X. Hu, Feature selection in supervised saliency prediction, IEEE Transactions on Cybernetics 45 (2015) 914–926.

[8] J. Wang, A. Borji, C. J. Kuo, L. Itti, Learning a combined model of visual saliency for fixation prediction, IEEE Transactions on Image Processing 25 (2016) 1566–1579.

[9] L. Zhang, X. Li, L. Nie, Y. Yang, Y. Xia, Weakly supervised human fixations prediction, IEEE Transactions on Cybernetics 46 (2016) 258–269.

[10] J. Liu, P. Musialski, P. Wonka, J. Ye, Tensor completion for estimating missing values in visual data, IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (2013) 208–220.

[11] C.-Y. Chen, K. Grauman, Inferring unseen views of people, in: Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 2011–2018. doi:10.1109/CVPR.2014.258.

[12] D. R. Hardoon, K. Pasupa, J. Shawe-Taylor, Image ranking with implicit feedback from eye movements, in: C. H. Morimoto, H. O. Istance, A. Hyrskykari, Q. Ji (Eds.), Proceeding of the 6th Biennial Symposium on Eye Tracking Research & Applications (ETRA 2010), 22-24 March 2010, Austin, USA, 2010, pp. 291–298. doi:10.1145/1743666.1743734.

[13] P. Auer, Z. Hussain, S. Kaski, A. Klami, J. Kujala, J. Laaksonen, A. P. Leung, K. Pasupa, J. Shawe-Taylor, Pinview: Implicit feedback in content-based image retrieval, in: T. Diethe, N. Cristianini, J. Shawe-Taylor (Eds.), Proceeding of the Workshop on Applications of Pattern Analysis (WAPA 2010), 1-2 September 2010, Cumberland Lodge, UK, volume 11 of *Journal of Machine Learning Research - Proceedings Track*, 2010, pp. 51–57. URL: http://www.jmlr.org/proceedings/papers/v11/auer10a.html.

[14] Z. Hussain, A. P. Leung, K. Pasupa, D. R. Hardoon, P. Auer, J. Shawe-Taylor, Exploration-exploitation of eye movement enriched multiple feature spaces for content-based image retrieval, in: J. L. Balcázar, F. Bonchi, A. Gionis, M. Sebag (Eds.), Proceeding of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD 2010), Part I, 20-24 September 2010, Barcelona, Spain, volume 6321 of *Lecture Notes in Computer Science*, 2010, pp. 554–569. doi:10.1007/978-3-642-15880-3_41.

[15] K. Pasupa, P. Chatkamjuncharoen, C. Wuttilertdeshar, M. Sugimoto, Using image features and eye tracking device to predict human emotions toward abstract images, in: T. Brunl, B. McCane, M. Rivers, X. Yu (Eds.), Proceeding of the 7th Pacific Rim Symposium on Image and Video Technology (PSIVT 2015), 23-27 Nov 2015, Auckland, New Zealand, volume 9431 of *Lecture Notes in Computer Science*, 2016, pp. 419–430. doi:10.1007/978-3-319-29451-3_34.

[16] M. Itskov, Tensor Algebra and Tensor Analysis for Engineers With Applications to Continuum Mechanics, 2nd ed., Springer, 2009.

[17] J. Synge, A. Schild, Tensor Calculus, Dover, 1978.

[18] K. Astikainen, L. Holm, E. Pitkänen, S. Szedmak, J. Rousu, Towards structured output prediction of enzyme function, in: BMC Proceedings, volume 2, Suppl 4, 2008, p. S2. doi:10.1186/1753-6561-2-s4-s2.

[19] S. Szedmak, T. De Bie, D. R. Hardoon, A metamorphosis of canonical correlation analysis into multivariate maximum margin learning, in: Proceeding of the 15th European Symposium on Artificial Neural Networks, 2007, pp. 211–216.

[20] T. Joachims, Training linear SVMs in linear time, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2006), 20-23 August 2006, Philadelphia, USA, 2006, pp. 217–226. doi:10.1145/1150402.1150429.

[21] N. Cristianini, J. Shawe-Taylor, An introduction to Support Vector Machines and other kernel-based learning methods, Cambridge University Press, 2000.

[22] C. V. Loan, The ubiquitous kronecker product, Journal of Computational and Applied Mathematics 123 (2000) 85–100. The nearest Kronecker product.

[23] T. G. Kolda, B. W. Bader, Tensor decompositions and applications, SIAM Review 51 (2009) 455–500.

[24] L. Pachter, B. Sturmfels, Algebraic Statistics for Computational Biology, Cambridge University Press, 2005.

[25] M. Drton, B. Sturmfels, S. Sullivant, Lectures on Algebraic Statistics, volume Oberwolfach Seminars, Vol 40, Birkhauser, 2009.

[26] D. G. Lowe, Object recognition from local scale-invariant features, in: Proceedings of the 7th IEEE International Conference on Computer Vision, volume 2, 1999, pp. 1150–1157. doi:10.1109/ICCV.1999.790410.

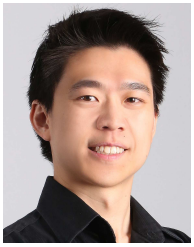
[27] C. Schmid, R. Mohr, C. Bauckhage, Evaluation of interest point detectors, International Journal of Computer Vision 37 (2000) 151–172.

[28] E.-C. Chang, S. Mallat, C. Yap, Wavelet foveation, Applied and Computational Harmonic Analysis 9 (2000) 312–335.

[29] J. Briët, P. Harremoës, Properties of classical and quantum Jensen-Shannon divergence, Physical Review A 79 (2009) 052311.

[30] K. Pasupa, S. Szedmak, Learning to predict where people look with tensor-based multiview learning, in: S. Arik, T. Huang, W. K. Lai, Q. Liu (Eds.), Proceeding of the 22nd International Conference on Neural Information Processing (ICONIP 2015), 9-12 Nov 2015, Istanbul, Turkey, volume 9489 of *Lecture Notes in Computer Science*, 2015, pp. 432–441. doi:10.1007/978-3-319-26532-2_47.

Biography



Kitsuchart Pasupa is an Assistant Professor in the Faculty of Information Technology, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand. Previously, he was a research fellow at Southampton and Sheffield Universities. He obtained a BEng in Electrical Engineering from Sirindhorn International Institute of Technology, Thammasat University, Thailand in 2003, and then went to the Department of Automatic Control and Systems Engineering, University of Sheffield where he received his MSc(Eng) and PhD in Automatic Control and Systems Engineering in 2004 and 2008, respectively. His main research interests lie in the application of machine learning techniques in the real world application.

Sandor Szedmak was born in Hungary. He obtained an MS degree in Mathematics from Lajos Kossuth University, Debrecen, Hungary. He received his PhD in Operations Research from the Rutgers, the State University of New Jersey, in the USA. He held research positions at the Computer Science Department of Royal Holloway, University of London and the University of Helsinki. Following that, he was a senior research fellow at the School of Electronics and Computer Science of University of Southampton in the UK, and in the Informatics Institute at the University of Innsbruck in Austria. Recently, he is a senior researcher at the Department of Computer Science of the Aalto University in Finland. His main research interest focuses on mathematical problems, e.g. optimisation, arising in machine learning and pattern recognition.

Appendix B

International Conference Papers

1. Kitsuchart Pasupa, Siripen Jungjareantrat, “Water Levels Forecast In Thailand: A Case Study Of Chao Phraya River”, In Proceeding of the 14th International Conference on Control, Automation, Robotics and Vision (ICARCV 2016), 13–15 November 2016, Phuket, Thailand, pp. 1–6, 2016. (Scopus, ISI-CPCI-S)
2. Kitsuchart Pasupa, Wisuwat Sunhem, “A Comparison between Shallow and Deep Architecture Classifiers on Small Dataset”, In Proceeding of the 8th International Conference on Information Technology and Electrical Engineering (ICITEE 2016), 5–6 October 2016, Yogyakarta, Indonesia, pp. 390–395, 2016. (Scopus, ISI-CPCI-S)

2016 14th International Conference on Control, Automation, Robotics & Vision
Phuket, Thailand, 13-15th November 2016 (ICARCV 2016)

Water Levels Forecast In Thailand: A Case Study Of Chao Phraya River

Kitsuchart Pasupa
Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang
Bangkok 10520, Thailand
Email: kitsuchart@it.kmitl.ac.th

Siripen Jungjareantrat
Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang
Bangkok 10520, Thailand
Email: siripenj7@gmail.com

***Abstract*—It is always desirable to be able to manage level of water in river, dam, and reservoir. Models have been constructed for predicting the level of these bodies of water, and good models can help increase the effectiveness of water management. Presently, the model that is employed by the Hydrographic Department of the Royal Thai Navy for predicting the level of water in Chao Phraya river is a harmonic method of tidal modeling. This model can predict the overall trend well but with high individual prediction error. Many machine learning algorithms for making predictions have also been introduced in recent years. Therefore, it was attempted in this study to compare the prediction performance of several machine learning models to that of the Royal Thai Navys model. These models were the following: linear regression, kernel regression, support vector regression, k-nearest neighbors, and random forest. The data input into these models were water level time series data of past 24, 48, and 72 hours measured at the Royal Thai Navy headquarters station, Phra Chulachomklao Fort, thirteen other stations along the river, and the output were predictions for the next 24 hours. It was found that all of the machine learning techniques were able to achieve better performances than that of the harmonic method of tidal modeling. The support vector regression model with Radial basis function kernel and 72-hour past time series data yielded prediction results with the least errors, at 0.117 m and 0.116 m for the water levels at the Royal Thai Navy headquarters station and Phra Chulachomklao Fort, respectively.**

I. Introduction

Thailand is an agricultural based country with a total agricultural area of 265,200 square meters [1]. A lifeblood of agriculture, water is one of the basic needs of living things. Water is everywhere but freshwater is very limited. Although 72.0% of the earth is covered with water but 97.5% of the water is in the ocean [2]. It is saltwater and not drinkable. The remaining 2.5% is fresh water which is in rivers, lakes, ground, icecaps, glaciers, etc. Therefore, water resource management for maintaining efficient agricultural production is very important to a country.

There are many approaches to water supply management: scheduled irrigation can be performed based on crop evapotranspiration [3] and forecast water level in the river or reservoir. There have been several works that focused on predicting water level in Thailand [4], [5], [6], [7], [8], [9], [10]. Chuanpongpanich *et al.* (2012) predicted water level at Phra Chulachomklao Fort by using Harmonic analysis [4]. This method represents water level at a considered location by mathematical model. Their experiment showed that using four selected tidal constituents as input variables were the best approach for predicting the water level. Ogata *et al.* (2012) simulated river discharge and water level in Chao Phraya river basin with a distributed hydrological model (DHM) [5]; unfortunately, the model overestimated the discharges in some areas. Alternatively, water level can be predicted by machine learning algorithms. Machine learning techniques have become popular in these days. Many researchers have applied machine learning algorithms to predict water level at several areas in Thailand such as Khlong U-Tapao river basin [6], at M.7 gauge station Mun river in Ubon Ratchathani [7], Chao Phraya river [8], [9], [10], and sea level at Gulf of Thailand [11]. These works can be further developed into an accurate flood warning application.

In this paper, we focused on forecasting water level at two locations along Chao Phraya river in Bangkok. Currently, the Hydrographic Department of the Royal Thai Navy makes prediction of water level in Chao Phraya river by using a harmonic method of tidal model. This model can predict the overall trend well but with high individual prediction error. Therefore, we aim to improve the prediction performance of the current approach by using various machine learning techniques with input features from two Royal Thai Navy stations at the Royal Thai Navy headquarters and Phra Chulachomklao Fort as well as thirteen telemetry stations of the Department of Drainage and Sewerage of Bangkok Metropolitan. We also investigated the importance of each used feature in terms of how well it reflected the water level in Chao Phraya river.

This paper is structured as follows: Section II describes the methodologies including harmonic analysis and machine learning algorithms used in this work; Section III describes the studied area followed by the experimental framework in section IV; Experimental result and conclusion are shown in section V and VI, respectively.

II. Methodology

A. Harmonic method of Tidal model

It is known that "tides" are the pattern of rising and falling sea level with respect to land. They occur once or twice a day depending on the location. Tides are mainly created by gravitational forces of the sun and the moon. Other factors are such as the elliptical shape of earth's orbit around the sun and

978-1-5090-3549-6/16/$31.00 ©2016 IEEE

the inclination of the lunar orbit plane. Tide periods do not happen at the same time in a day because the moon requires approximately 24 hours and 50 minutes to appear at the same point on the Earth again [12].

Tidal prediction is made all around the world and available to public. It is important for individuals who depends on the ocean for their life. Tide can be predicted by harmonic analysis which applies superposition principle to various selected sinusoidal components. Harmonic analysis can be defined as follows:

$$\mathrm{WL}(t) = \sum_{i=1}^{N} a_i \cos(\omega_i t + \alpha_i), \tag{1}$$

where WL is water level at time t, a_i is amplitude of the i^{th} tidal component, ω_i and α is angular frequency and phase of the i^{th} tidal constituent, and N is number of selected tidal components. Examples of important tidal components are principal lunar semidiurnal, principal solar semidiurnal, larger Lunar elliptic semidiurnal, Luni-solar declinational diurnal, and Lunar declinational diurnal constituent.

B. Machine Learning Techniques

Our task here is a regression task. It aims at obtaining a model that can explain the relationship between input variables and output variables. The output variables are continuous values. Many machine learning techniques have been introduced and reintroduced for regression task. In this work, we investigated on five learning algorithms.

1) Linear Regression: Consider a linear function,

$$h(x) = w_0 + x_1 w_1 + x_2 w_1 + \ldots + x_n w_n \tag{2}$$

where x is an input data consisting of n features, w_n is a weight corresponding to each input feature, and w_0 is a bias value. We aim to find a model, $h(x_i)$, that gives a good approximation output by minimizing the mean square error between the predicted and observed output defined as:

$$J(w) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h(x_i) - y_i)^2 + \lambda \sum_{j=1}^{n} w_j^2\right] \tag{3}$$

where m is number of samples, y is output, and λ is regularization parameter. The above equation can be solved by a method of gradient descent that produces the following update equations:

$$w_0 := w_0 - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}(h(x_i) - y_i)\cdot x_0^{(i)}\right]; x_0^{(i)} = 1 \tag{4}$$

$$w_j := w_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}(h(x_i) - y_i)\cdot x_j^{(i)} + \frac{\lambda}{m} w_j\right] \tag{5}$$

where $j = 1, \ldots, n$, and α is a learning rate.

2) Kernel Regression: Because data is non-linear in real world application; therefore, linear regression model can fail to generalize. Hence, sometimes it is required to find a non-linear relationship between a pair of input and output. In order to accommodate the non-linearity, "kernel trick" is introduced. It aims to project data from the original feature space to a new feature space in which linear function can be used to generalize. Gram matrix (K), used as a design matrix in linear regression, is associated with kernel $k(\cdot,\cdot)$, i.e. $k_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. In this work, we used (Gaussian) radial basis function kernel (RBF),

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}\right) \tag{6}$$

where σ is a kernel parameter.

3) Support Vector Regression: The solution of Kernel regression is not sparse in variables as it uses all features. This leads to a complex model with high computational cost. Therefore, support vector regression (SVR) is introduced that aims to solve the following convex optimization problem:

$$\begin{aligned} \min_{w,\xi,\hat{\xi}} \quad & \tfrac{1}{2}||\mathbf{w}||^2 + \tfrac{C}{2}\textstyle\sum_i(\xi_i^2 + \hat{\xi}_i^2) \\ s.t. \quad & \mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b - y_i \le \epsilon + \xi_i \\ & y_i - \mathbf{w}^{\mathrm{T}}\mathbf{x}_i - b \le \epsilon + \hat{\xi}_i \\ & \xi_i, \hat{\xi}_i \ge 0, i = 1, \ldots, m \end{aligned} \tag{7}$$

where C is SVR hyperparameter, b is a bias value, and $\xi_i, \hat{\xi}_i$ are slack variables. SVR can be used to solve quadratic programs that are convex problems that have a unique optimal solution. Again, kernel trick is utilized to enable nonlinear mapping.

4) k-Nearest Neighbors (kNN): kNN is a non-parametric and instance-based learning method. It is one of the simplest algorithms that compares a test sample to the k-closest training examples. Therefore, the output will be an average of the values of its k nearest neighbors.

5) Random Forest (RF): RF is an ensemble method that combines several decision tree models. Each model is trained on a subset of features that are randomly selected. A test sample is tested on all decision tree models. A predicted value is calculated by aggregating the predictions of these trees. This can be done by averaging all of the values.

III. Study Area

Chao Phraya River is a major river of Thailand. It begins in Nakhon Sawan province in Northern Thailand from a combination of two rivers–the Ping river and the Nan river. Chao Phraya river flows south for 372 kilometers, passes through Bangkok, and ends at the Gulf of Thailand.

This study focuses on predicting water levels at two locations along Chao Phraya river. Data of water level at the two locations were measured by the Royal Thai Navy. Within the Royal Thai Navy is a government organization that regularly makes prediction of expected sea water level and expected river water level in Chao Phraya river. We collected data from two Royal Thai Navy stations as well as 14 other stations along the river that belong to the Department of Drainage and Sewerage (DDS) of Bangkok Metropolitan. The two Royal Thai Navy stations are a station at the Hydrographic Department (HD) of the Royal Thai Navy headquarters and a station at Phra Chulachomklao Fort (PC). Currently, the Royal Thai Navy makes prediction of expected water level one year ahead by using a harmonic method of tidal model (HT). DDS has 77 telemetry stations measuring water level, rainfall, temperature, and humidity at every canal in Bangkok. However, in this

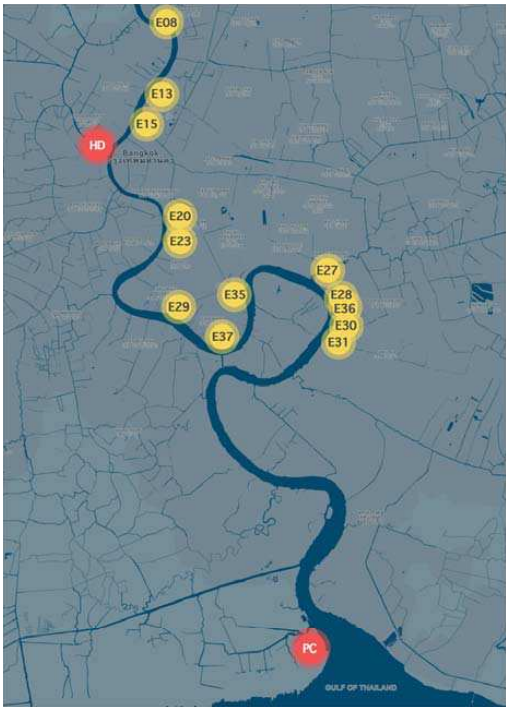


Fig. 1. Location of selected telemetry stations in Chao Phraya river.

study, we focused only on water level in Chao Phraya river; therefore, the data collected were only from stations at the canals that connect directly Chao Phraya river. Moreover, only telemetry stations where are located in the East side of Chao Phraya river were considered. Data from a total of 14 DDS telemetry stations were considered in this work. It should be noted that the water levels at the 14 stations are the levels of water outside the water gate, hence, these levels reflect the levels of water in Chao Phraya river. The locations of these stations are shown in Fig. 1.

IV. Experimental Framework

Data used in this study were collected in 2010 and 2011. Description of telemetry stations considered in the study are shown in Table I. The data was sampled and stored every hour. This led to 17,520 data points. Missing data always occurred because of some unexpected and unavoidable problems, e.g. sensors were broken. We excluded the data collected from Khlong Bang Sue Station (E10) because there were too many data points missing, 5,702 in all. Other stations also had some missing data points but they were comparatively fewer than those missing in E10. Hence, the total number of features used in this study was 17. These features were the following: (i) two water level values at HD and PC (subsequently referred to as $\mathrm{WL_{HD}}$ and $\mathrm{WL_{PC}}$, respectively; (ii) two HT-predicted water levels at HD and PC (subsequently referred to as $\mathrm{HT_{HD}}$ and $\mathrm{HT_{PC}}$, respectively); and (iii) thirteen water levels at 13 of the 14 DDS telemetry stations, excluding the one at E10 station.

TABLE I. A list of telemetry stations and their returned data used in this research.

The Royal Thai Navy		
Index	Name	Value (Unit: Metres above mean sea level)
HD	Hydrographic Department	Water level and Expected water level by HT
PC	Phra Chulachomklao Fort	
The Department of Drainage and Sewerage, Bangkok		
Index	Name	Value (Unit: Metres above mean sea level)
E08	Khlong Bang Khen Mai	Water Level
E10	Khlong Bang Sue	
E13	Khlong Sam Sen	
E15	Khlong Thewet	
E20	Khlong Krung Kasem	
E23	Khlong Sathorn	
E29	Khlong Wat Sai	
E37	Khlong Wat Dan	
E35	Khlong Chong Nonsi	
E27	Khlong Toei	
E28	Khlong Jek	
E36	Khlong Bang Chak	
E30	Khlong Bang O	
E31	Khlong Bang Na	

We pre-processed the data by introducing lagged variables (up to 72 hours lag) for these 17 features as input data matrix, $\mathbf{x}_i = [\ x_i(t) \quad x_i(t-1) \quad x_i(t-2) \quad ... \quad x_i(t-72)\ ], i = 1, ..., 17$. We aimed to make 24-hour ahead predictions; therefore, the input data matrix was paired with the output data matrix of 24-hour ahead water levels at HD and PC. The input data matrix became $17,520 \times 1,241$. Again, we eliminated samples that had missing values. This gave us a new data matrix with 15,483 data points. The data points were normalized to zero mean and unit standard deviation.

We made predictions using several different well-known machine learning algorithms mentioned in Section II, namely, linear regression, kernel regression, SVR with linear and RBF kernel, kNN, and RF. The dataset was randomly split into training and test sets with 1/3 and 2/3 of the total number of samples, respectively. We ran simulation five times with different random seeds. The adjustable parameters of each algorithm were tuned as follows: (i) Linear regression: learning rate range was $\{10^{-6}, 10^{-5}, \ldots, 10^{5}, 10^{6}\}$; (ii) Kernel regression: RBF kernel parameter range was $\{10^{-6}, 10^{-5}, \ldots, 10^{5}, 10^{6}\}$ and learning rate range was $\{10^{-6}, 10^{-5}, \ldots, 10^{5}, 10^{6}\}$; (iii) SVR: C parameter range was from 10^{-6} to 10^{6} for both linear and kernel cases (for comparison to be fair, RBF function was used with a range similar to that of the kernel regression); (iv) kNN: k range was from 1 to 500; and (v) RF: the number of trees was from 100 to 1000.

To obtain an optimal model for each algorithm and each realization, five-fold cross-validation was applied on the training set. After the optimal parameters for each model were obtained, the model was trained with the whole training set and tested with the test set. We reported performance as average root mean squared error (RMSE) in meter (m.) across five runs. RMSE was calculated by,

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2} \tag{8}$$

where y_i is a true value and $\hat{y}_i$ is a predicted value.

TABLE IV. PERCENTAGE OF SUPPORT VECTOR USED IN SVR WITH DIFFERENT TYPE OF KERNELS AT HD AND PC STATIONS.

Methods	Location	%SV Used
SVR-Linear	HD	98.07%
SVR-RBF	HD	99.08%
SVR-Linear	PC	98.03%
SVR-RBF	PC	98.07%

V. EXPERIMENTAL RESULTS AND DISCUSSION

We analyzed the following combinations of feature sets: (i) Using $\mathrm{HT_{HD}}$ to predict water level at HD station as well as using $\mathrm{HT_{PC}}$ to predict water level at PC station, (ii) Using HT with two stations' previous water level records ($\mathrm{WL_{HD}}$ and $\mathrm{WL_{PC}}$), (iii) Using only the available water level information from the Royal Thai Navy stations together with those from the DSS stations ($\mathrm{WL_{Exx}}$), and (iv) Using all of the information we gathered. The results obtained from using these combinations were compared with that of the baseline HT method.

It is clear that all of the combinations of feature sets proposed here was able to achieve lower RMSE in all cases as shown in Table II and III for HD and PC, respectively. It was found that regression with RBF kernel yielded the best performance on the average and in most of the cases of both HD and PC stations. kNN was the worse among all. SVR was slightly worse than regression but SVR model was simpler than the regression model. Moreover, the regression model used all of the training samples but SVR used support vectors that consisted of only some of the training samples as shown in Table IV. That might be the cause for the drop in performance.

Using more lag data should be able to lower RMSE. It can been seen that the SVR with RBF function in conjunction with 72-hour lag data of all features yielded the best prediction performances for both HD and PC stations as shown in Table II and III, respectively.

Fig. 2 shows 24-hour ahead predictions of water level for 5 days for HD and PC by SVM-RBF with 72-hour lag data. All of the proposed features performed better than HT–their predicted values were closer to the true values. According to Fig. 2, it was observed that tides were periodic rises and falls of water level with approximately 24-hour period. Clearly, this is the effect of gravitational attraction of the moon and the sun.

In addition, we investigated the contribution of each feature to predicting water level with linear regression and SVR with linear kernel. The contributions of 72-hour lag feature to water level predictions for HD and PC are shown in Fig. 3 and Fig. 4, respectively. The contribution of every feature was averaged across five runs. It should be noted that we reported the contribution to making accurate prediction for linear cases only, ignoring the non-linear weights of RBF to the input features. For HD station, $\mathrm{HT_{HD}}$ was the most informative feature for linear regression followed by $\mathrm{WL_{E27}}$, $\mathrm{WL_{E30}}$, $\mathrm{WL_{E13}}$, and $\mathrm{WL_{E35}}$. In the SVR case, the most dominant feature was $\mathrm{WL_{E27}}$, followed by $\mathrm{HT_{PC}}$, $\mathrm{WL_{E31}}$, $\mathrm{WL_{E37}}$, $\mathrm{WL_{E30}}$, and $\mathrm{WL_{PC}}$. This indicated the following:

1) water levels in Chao Phraya river were affected by sea water level as reflected in $\mathrm{HT_{HD}}$, $\mathrm{HT_{PC}}$, and $\mathrm{WL_{PC}}$;

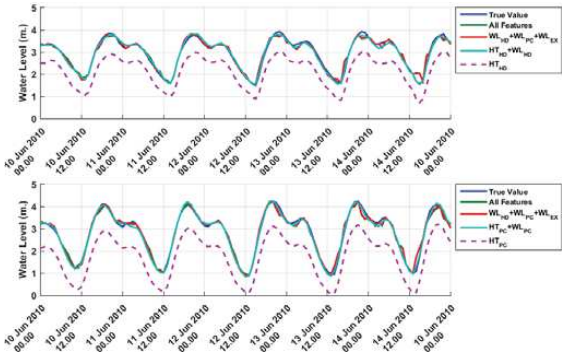


Fig. 2. Five-day prediction of water levels in June 2010 at HD (top) and PC (bottom) using SVM-RBF with different sets of features (72 hours past time series data).

2) water levels at the telemetry stations near the Gulf of Thailand–$\mathrm{WL_{E30}}$ and $\mathrm{WL_{E31}}$–strongly affected the water level at HD because of the influence of sea water level (this included $\mathrm{WL_{E37}}$, and $\mathrm{WL_{E35}}$ because they were located near Khlong Lat Pho; the water level at Khlong Lat Pho was not be included in this study because it is located in the West side of the Chao Phraya river; Khlong Lat Pho was initiated by HM King Bhumibol Adulyadej and aimed to drain flood water into the sea more quickly bypassing 18 kilometers of the river to 600 meters [13]);
3) water level at Khlong Toei station $\mathrm{WL_{E27}}$ also strongly affected the water level in Chao Phraya river (this is because it was located near Phra Khanong pumping station, the biggest pumping station for preventing flood in Bangkok [14]; we did not include Khlong Phra Khanong station in our models because it was not located along Chao Phraya river, but the water level at Klong Toei station which were next to it was heavily influenced by the pumping action at Khlong Phra Khanong station);
4) water level at the upstream of the river also affected the level at $\mathrm{WL_{E13}}$.

To sum up, it is clear that sea water level strongly affected the water levels at PC station because $\mathrm{HT_{PC}}$ and $\mathrm{WL_{PC}}$ were the largest weights among all of the weights, followed by $\mathrm{HT_{HD}}$, and $\mathrm{WL_{E37}}$. This was because PC was located near the sea and $\mathrm{WL_{E37}}$ was near Khlong Lat Pho, a bypass canal to the sea.

We further investigated lag hour that was most important to accurate prediction. The results are shown in Fig. 5 and Fig. 6 for HD and PC, respectively. Both algorithms gave the highest weights to the current event $x_0(t)$. Weights decreased for every lag hour until 24-hour lag was met, then the algorithms tended to increase weight again but the increased weights were smaller than the current events weight. This happening periodically occurred every 24 hours.

Fig. 7 and Fig. 8 show a comparison of observed water levels with predicted water levels by all of the methods used in combination with 72-hour lag feature at HD and PC, respec-

TABLE II. AVERAGE RMSE ACROSS FIVE RUNS ON DIFFERENT TYPES OF INPUT, LAGGED HOURS, AND MACHINE LEARNING ALGORITHMS AT HD STATION. BOLD VALUES IN EACH ROW INDICATE THE BEST RMSE OF EACH CASE. * INDICATES THE BEST ACCURACY ACHIEVED AT HD STATION.

Input Features	Lagged Hour	Baseline	Regression		SVR		RF	kNN
			Linear	RBF	Linear	RBF		
HT_{HD}	24	0.892	0.321	**0.217**	0.327	0.233	0.273	0.459
	48		0.311	**0.184**	0.315	0.236	0.265	0.445
	72		0.310	**0.177**	0.315	0.218	0.259	0.439
$HT_{HD} + WL_{HD}$	24		0.142	**0.133**	0.143	**0.133**	0.148	0.290
	48		0.138	0.129	0.140	**0.128**	0.144	0.267
	72		0.138	0.128	0.139	**0.126**	0.141	0.266
WL_{HD}	24		0.158	**0.140**	0.166	**0.140**	0.148	0.214
$+WL_{PC}$	48		0.143	**0.129**	0.160	**0.129**	0.145	0.227
$+WL_{EX}$	72		0.140	0.123	0.171	**0.122**	0.142	0.243
$HT_{HD} + WL_{HD}$	24		0.141	**0.127**	0.143	0.129	0.146	0.214
$HT_{PC} + WL_{PC}$	48		0.135	**0.121**	0.150	**0.121**	0.142	0.227
$+WL_{EX}$	72		0.134	0.118	0.161	**0.117***	0.139	0.243
Average			0.184	**0.144**	0.194	0.153	0.174	0.295

TABLE III. AVERAGE RMSE ACROSS FIVE RUNS ON DIFFERENT TYPES OF INPUT, LAGGED HOURS, AND MACHINE LEARNING ALGORITHMS AT PC STATION. BOLD VALUES IN EACH ROW INDICATE THE BEST RMSE OF EACH CASE. * INDICATES THE BEST ACCURACY ACHIEVED AT PC STATION.

Input Features	Lagged Hour	Baseline	Regression		SVR		RF	kNN
			Linear	RBF	Linear	RBF		
HT_{PC}	24	0.976	0.148	**0.129**	0.149	0.138	0.165	0.351
	48		0.143	**0.126**	0.144	0.131	0.153	0.348
	72		0.140	**0.128**	0.141	0.130	0.148	0.361
$HT_{PC} + WL_{PC}$	24		0.129	**0.121**	0.129	0.123	0.152	0.314
	48		0.123	0.119	0.123	**0.118**	0.141	0.320
	72		0.121	**0.117**	0.122	0.119	0.136	0.336
WL_{HD}	24		0.189	**0.155**	0.200	0.161	0.172	0.295
$+WL_{PC}$	48		0.151	**0.136**	0.169	0.139	0.164	0.323
$+WL_{EX}$	72		0.144	**0.127**	0.176	0.129	0.158	0.353
$HT_{HD} + WL_{HD}$	24		0.131	**0.123**	0.135	0.131	0.155	0.284
$HT_{PC} + WL_{PC}$	48		0.127	**0.117**	0.140	0.120	0.145	0.315
$+WL_{EX}$	72		0.128	**0.116***	0.151	**0.116***	0.139	0.345
Average			0.140	**0.126**	0.148	0.130	0.152	0.329

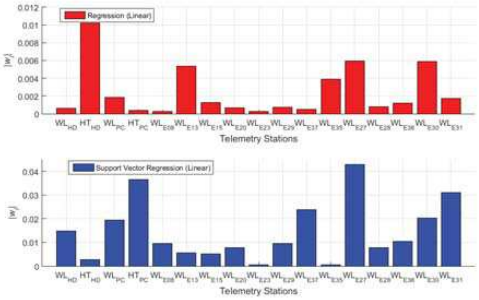


Fig. 3. An illustration of feature importance to every considered telemetry stations, the absolute value of the linear regression's (top) and SVR's (bottom) weight vectors $|w_i|$ average across five runs at HD station.

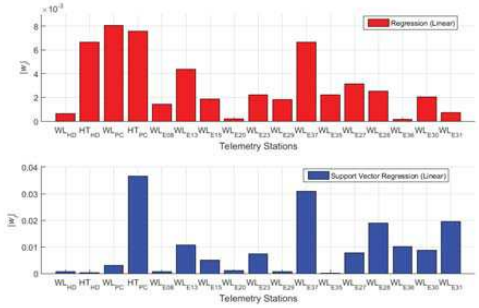


Fig. 4. An illustration of feature importance to every considered telemetry stations, the absolute value of the linear regression's (top) and SVR's (bottom) weight vectors $|w_i|$ average across five runs at PC station.

tively. It is clear that using kernel regression with RBF kernel and SVR with RBF kernel yielded less scattered estimates than using linear kernel for both considered stations. The performance of RF was comparable to Regression-RBF and SVR-RBF. Clearly, kNN was the worst algorithm compared to the other algorithms.

VI. CONCLUSION

This paper proposed approaches to forecast water levels in the Chao Phraya river by using different machine learning algorithms, i.e. linear regression, kernel regression, SVR with linear and RBF kernel, kNN, and RF. It was found that SVR with RBF kernel function in conjunction with 72-hour lag feature was the best contender. All of the proposed approaches also yielded better prediction performance than the current approach used by the Royal Thai Navy. We also analyzed the contributions of every feature used in this study and found that they reflected the near-the-sea-geography of Chao Phraya river and the pumping operation of stations in the vicinity– water levels in Chao Phraya river near Bangkok were strongly influenced by both nature and human.

REFERENCES

[1] S. Sethaputra, S. Thanopanuwat, L. Kumpa, S. Pattanee, L. Ti, T. Facon *et al.*, "Thailand's water vision: a case study." *From vision to action: A synthesis of experiences in Southeast Asia*, pp. 71–97, 2001.

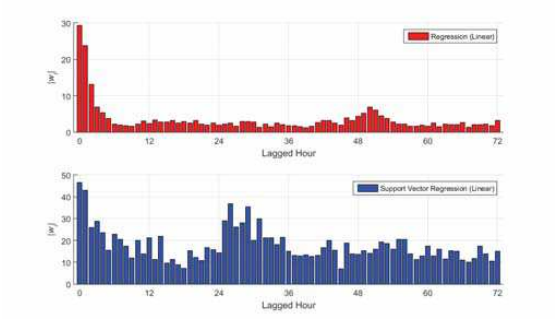


Fig. 5. An illustration of feature importance to lagged hours, the absolute value of the linear regression's (top) and SVR's (bottom) weight vectors $|w_i|$ average across five runs at HD station.

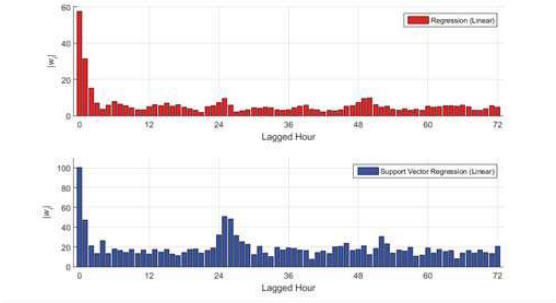


Fig. 6. An illustration of feature importance to lagged hours, the absolute value of the linear regression's (top) and SVR's (bottom) weight vectors $|w_i|$ average across five runs at PC station.

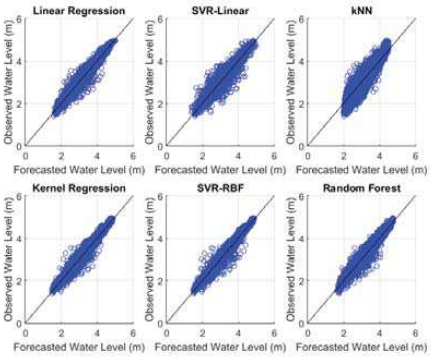


Fig. 7. Comparison of observed water level and forecasted water level at HD station using 72-hour lagged of all features.

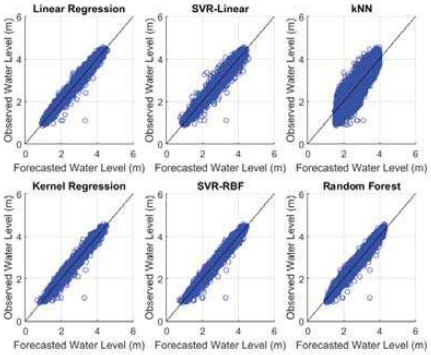


Fig. 8. Comparison of observed water level and forecasted water level at PC station using 72-hour lagged of all features.

[2] B. Israel. (2010) How much water is on earth? LiveScience. [Online]. Available: http://www.livescience.com/29673-how-much-water-on-earth.html

[3] K. Pasupa and E. Thamwiwatthana, "Prediction of reference evapotranspiration with missing data in Thailand," in *Proceeding of the 5th International Conference on Information Technology and Electrical Engineering (ICITEE 2013), 7-8 October 2013, Yogyakarta, Indonesia*, 2013, pp. 181–186.

[4] S. Chuanpongpanich, P. Arlai, M. Koch, T. Kojiri, K. Tanaka, and T. Tingsanchali, "Tide prediction for the downstream boundary condition of the Chao Phraya river integrated river model using harmonic analysis," in *Proceeding of the 10th International Conference on Hydroscience and Engineering (ICHE 2012), 4-7 November, Orlando, USA*, 2012, pp. 1–5.

[5] T. Ogata, O. C. Saavedra Valeriano, C. Yoshimura, W. Liengcharernsit, and Y. Hirabayashi, "Past and future hydrological simulations of Chao Phraya river basin," *Journal of Japan Society of Civil Engineers, Ser. B1 (Hydraulic Engineering)*, vol. 68, no. 4, pp. I_97–I_102, 2012.

[6] S. Phuphong and C. Surussavadee, "An artificial neural network based runoff forecasting model in the absence of precipitation data: A case study of Khlong U-tapao river basin, Songkhla province, Thailand," in *Proceeding of the 4th International Conference on Intelligent Systems, Modelling and Simulation (ISMS 2013), 29-31 Janurary, Bangkok, Thailand*, 2013, pp. 73–77.

[7] T. Vangpaisal and J. Threenat, "Factors affecting the accuracy of water level forecasting at M.7 gauge station using artificial neural network model," in *Proceeding of the 2nd STOU Graduate Research Conference, 4-5 September, Nonthaburi, Thailand*, 2012, pp. 1–14.

[8] T. Tingsanchali, "Forecasting model of Chao Phraya river flood levels at Bangkok," in *Proceeding of the International Conference on Chao Phraya Delta, Bangkok, Thailand*, 2000, pp. 1–13.

[9] S. Chuanpongpanich, P. Arlai, M. Koch, K. Tanaka, and T. Tingsanchali, "Inflow prediction for the upstream boundary condition of the Chao Phraya river model using an artificial neural network," in *Proceeding of the 10th International Conference on Hydroscience and Engineering (ICHE 2012), 4-7 November, Orlando, USA*, 2012, pp. 1–5.

[10] C. Soomlek, N. Kaewchainam, T. Simano, and C. So-In, "Using backpropagation neural networks for flood forecasting in Phra Nakhon Si Ayutthaya, Thailand," in *Proceeding of the International Computer Science and Engineering Conference (ICSEC 2015), 23-26 November, Chiang Mai, Thailand*, 2015, pp. 1–6.

[11] P. Sojisuporn, C. Sangmanee, and G. Wattayakorn, "Recent estimate of sea-level rise in the Gulf of Thailand," *Maejo International Journal of Science and Technology*, vol. 7, pp. 106–113, 2013.

[12] MoonConnection. (2016) The ocean's tides explained. MoonConnection.com. [Online]. Available: http://www.moonconnection.com/tides.phtml

[13] National News Bureau of Thailand, Government Public Relations Department. (2015) Lat Pho canal project to tackle flooding in Bangkok. HM the King's 88th Birthday Celebrations. [Online]. Available: http://thainews.prd.go.th/nnt_en/King87en/Relatearticles.html

[14] The Department of Drainage and Sewerage. (2015) Action plan for preventing flooding in Bangkok. Annual Report. [Online]. Available: http://203.155.220.119/News_dds/magazine/Plan58/Plan2558_1.pdf

A Comparison between Shallow and Deep Architecture Classifiers on Small Dataset

Kitsuchart Pasupa
Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang
Bangkok 10520, Thailand
Email: kitsuchart@it.kmitl.ac.th

Wisuwat Sunhem
Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang
Bangkok 10520, Thailand
Email: wisuwat.sun@gmail.com

***Abstract*—Many machine learning algorithms have been introduced to solve different types of problem. Recently, many of these algorithms have been applied to deep architecture model and showed very impressive performance. In general, deep architecture model suffers from over-fitting problem when there is a small number of training data. In this paper, we attempted to remedy this problem in deep architecture with regularization techniques including overlap pooling, flipped-image augmentation and dropout, and we also compared a deep structure model (convolutional neural network (CNN)) with shallow structure models (support vector machine and artificial neural network with one hidden layer) on a small dataset. It was statistically confirmed that the shallow models achieved better performance than the deep model that did not use a regularization technique. However, a deep model augmented with a regularization technique–CNN with dropout technique–was competitive to the shallow models.**

***Keywords*—machine learning; shallow learning; deep learning; small dataset.**

I. Introduction

Today, machine learning techniques are becoming widely used in real world applications [1]. They are now integrated in many modern applications, e.g., hand writing recognition, image understanding, and video suggestion. They create and improve models by learning from accumulated data. In the past, many machine learning techniques, e.g., Support Vector Machine (SVM), have shown their substantial performance and effectiveness in many real world applications [2]–[4]. However, these algorithms require good extracted hand-crafted features. Feature extraction is one of challenging tasks that needs expert knowledge to perform adequately. Features extracted from each data sample are fed into learning algorithms. We later refer to these algorithms as "shallow model" algorithms because they consist of very few layers of composition. These include Artificial Neural Network (ANN) with one hidden layer as well. In order to avoid complex feature extraction process, "deep model" was introduced. It aims to automatically learn feature hierarchy from low level to high level. It moves machine learning forward closer to the ultimate goal of artificial intelligence, that is, a machine that can think like a human does [5].

Although there are pieces of evidence that deep learning is able to achieve better performance than shallow learning [6], [7], it is well known that deep learning has limitations that should be considered. Examples of these limitations are the following:

i) High computational cost in training phase: it has higher computational complexity comparing to shallow learning and requires high performance computing unit to train–using GPU-based implementation can significantly reduce model training time comparing to using CPU-based implementation [8].
ii) Over-fitting problem when the data set is small: performance can drop dramatically due to over-fitting. Shallow learning techniques can overcome the problems of deep learning techniques when data is scarce. Many techniques have been proposed for solving over-fitting problem, such as overlap pooling, dropout and flipped-image augmentation. Another technique, transfer learning can be used when there is a small number of training sample. This can be done by training a model with related domain dataset that contains large number of samples. As a model converges, it fine-tunes all of the layers of the network by using an in-domain small sample data set. An example of usages of transfer learning technique can be found in a recent work by [9]. They attempted to use Convolutional Neural Network (CNN) with transfer learning technique for emotion recognition of human face on small datasets. However, it is not always possible to find a related domain dataset to train on first. CNN was also applied on ImageNet dataset [7]. This dataset consists of 1.2 million labeled images under 1000 different categories. Although the dataset is large, there are only roughly 1000 images in each of the 1000 categories. This is rather a small sample size compared to the number of categories. In this case, the authors applied a dropout technique to solve the over-fitting problem.

There has been a work that compared the performances of machine learning with shallow and deep architecture [10]. This paper concluded that combining them together gave the best result for text analysis task.

Our study investigated the same small face shape dataset that we used in one of our previous studies [11]. In that study, we showed that SVMs in conjunction with Radial Basis Function (RBF) outperformed other shallow machine learning

algorithms on this dataset. In this study, we used a deep learning technique–CNN–and compared it with two well-known shallow model learning algorithms–SVM and ANN with one hidden layer. Because the dataset was very small, CNN model could be over-fitting, so we employed overlap pooling, flipped-image augmentation, and dropout regularization techniques to handle it.

The paper is organized as follows: Section II presents the methodology used in this paper; we explain our experimental framework and settings in Section III; The results are discussed and concluded in Section IV and V, respectively.

II. Methodology

Many machine learning algorithms have been introduced and re-introduced. They can be categorized into two types according to their model structure: shallow and deep.

A. Shallow model learning

Shallow model learning is a type of machine learning algorithms can generate good generalized predictive model with only a few layers of composition. It requires samples with well-studied discriminative features extracted by experts. It can perform well even though only a limited number of samples is available. This work focused on two well-known shallow machine learning algorithms: ANN with one hidden layer and SVM.

1) Artificial Neural Network (ANN): ANN was inspired by human brain operation. Its complexity can be increased by adding more hidden layers into the model and/or more neurons into each layer. ANN with one hidden layer is considered a shallow structure model while ANN with more than one hidden layers is a deep model. The model can be trained with different loss functions such as cross entropy error and classification error functions. A gradient decent algorithm is usually applied to minimize a selected loss function in order to obtain optimal variables.

2) Support Vector Machine (SVM): SVM is an instance-based learning that classifies data into two classes. The model selects and utilizes proper representative instances from a training set, so-called "support vector". SVM tries to generate a maximum-margin hyper-plane between support vectors of each class. Basically, it performs linear classification. In order to enable it to be a nonlinear classifier, a kernel function is applied. Kernel function maps data from its input space to a new space that SVM can perform nonlinear classification of data. SVM can also be extended to solve regression tasks.

Both models can perform nonlinear prediction. SVM usually performs better than ANN as shown in [12]–[14]. SVM performance can drop when applied to noisy data whereas ANN tolerates noise better and is more robust in some tasks [15], [16]. These algorithms are powerful machine learning techniques that can be applied to solve a complex problem; however, they require discriminative features extracted by human.

B. Deep model learning

The first deep learning technique so-called "Convolutional Neural Network" (CNN) was first introduced by Fukushima in 1980 [15]. It was inspired by real biological learning process. A complex combination of cells in an animal's visual cortex is called a receptive field. The field processes small sub-pictures of an image and combines them to recognize the context [17]. In deep learning, a region in an input image covered by a mask is a receptive field for a neuron in hidden layer. The mask is shifted by one or more pixel throughout the input image. A sub-image has similar features to the others if their output values are close to each other. The number of masks represents the number of neurons in the hidden layers. Hidden layers are structured hierarchically as layers in ANN. The weights of each mask can be adjusted to achieve an optimal performance. This process is called "feature learning" [18]. It performs automatic feature extraction prior to the classification phase. Feature learning is an important ability of deep learning algorithms that can learn to extract features from raw data without human aid. In this work, we also investigated CNN. CNN consists of three main layers which are (i) convolutional layer, (ii) pooling layer, and (iii) fully connected layer. Convolutional layer operates as a filter–receptive field–that can be tuned to gain an optimal model with good feature extraction. Pooling layer performs a data summarizing operation such as Mean, Max, and Min in order to reduce the spatial size of the representation in the network and control over-fitting of the model. The architecture of the last type of layer, fully connected layer, is similar to that of a traditional neural network that uses extracted features from the layers before it. With all of these components, the model can be trained with any kinds of raw data, especially, image data. It is known that deep learning algorithms require a high performance computing unit to generate a model and a large dataset to obtain a good model [19]. If there is a small number of training data, the model can be easily over-fitting. The following special techniques are needed to solve this problem–overlap pooling, dropout, data flipped-image augmentation [7].

i) Overlap pooling: In a standard pooling layer, the mask is shifted (stride) so that the next position does not overlap with the current one. The technique is applied to collect more input information with the size of the mask previously mentioned. Hence, it can solve the over-fitting problem. Moreover, in order to enhance the performance of the network, the mask can also be overlapped [20].
ii) Flipped-image augmentation: CNN is definitely over-fitting if the training set is small. To cope with this problem, a new set of images is created by flipping images in the dataset horizontally and augmenting them to the training set. Therefore, the number of samples is now twice that of the samples in the old training set. It also increases the diversity of training set.
iii) Dropout: this is a popular technique that handles over-fitting by randomly removes some neurons in the hidden layer of the fully-connected architecture in the training

phase, but in the test phase, all neurons in the hidden layer are still used. A probability of retain unit, p, is required for adjustment to get a good model. [21] suggested that if n is an optimal number of nodes used in a standard ANN, a good dropout network should have $\frac{n}{p}$ neurons.

III. Experimental Framework

A. Face Shape Dataset

Images of women faces were collected from the Internet in order to build a hairstyle recommendation system [11]. They were labeled by volunteers who had passed a qualifying test. This dataset contained 500 samples of five face shape categories: heart-shape, oval-shape, oblong-shape, round-shape, and square-shape. Each class consisted of 100 samples. This dataset was later used in the developed hairstyle recommendation system for woman to analyze the face shape of a user and suggest proper hairstyles for her [22]. The system could also simulate the hairstyle she was wearing. Nineteen features were carefully extracted to obtain discriminative features for five different face shapes. It was found that SVM in conjunction with RBF was the best contender for this dataset.

B. Data pre-processing

In this work, we investigated three types of data representation.

1) Hand-crafted features: The 19 features proposed in [11] were applied to SVM and ANN.

2) Raw image: We simply utilized raw images shown in Fig. 1a. All images were resized to 48×48 pixels and converted to gray level scale because color domain is not necessary for face shape classification. This type of data representation were used in our deep model.

3) Reconstructed image: Raw images are very complex and composed of components that may not be necessary for performing classification and can lead to poor performance. Therefore, from raw images, we reconstructed new images that had only the necessary components for classification. We generated 61 feature points from the Active Appearance Model (AAM) and face color-based segmentation as described in [11]. All of these points were connected to represent the shape of a face as shown in Fig. 1b. The reconstructed images were used in the CNN.

C. Experimental Setting

We aimed to find a set of optimal parameters for each model. In order for us to be able to do that, we divided the data into three sets: 400 samples for training set, 50 samples for validation set, and 50 samples for test set. The set of optimal parameters were selected based on the prediction accuracy on the validation set. Once it was obtained, we again trained the model on the combined training and validation set. Here, we compared the prediction accuracy and area under the receiver operating characteristic (AUROC) on the test set between all of the contenders. The adjustable parameters of each algorithm were tuned as follows:

1) SVM: As SVM is a binary classifier, a one-vs-one scheme was used to enable SVM to perform multi-class classification task. RBF kernel was employed. The regularization and kernel parameter range was $\{10^{-5}, 10^{-4}, ..., 10^{4}, 10^{5}\}$.

2) ANN with one hidden layer: We varied the number of neurons in each layer from 1 to 30.

3) CNN: There was a large number of parameters to be considered for CNN that needs high computational resource. Hence, we investigated only CNN with one to five convolutional layers to get an optimal model. We considered the following two structures: (i) CNN with one convolutional layer: the 32-dimension deep convolutional layer used a mask of 5×5 size that shifted one pixel at a time. We utilized Max pooling with a 2×2 filter in the pooling layer. In this layer, the filter shifted 2 pixels at a time in order to reduce the number of variables which was the output volume. This is shown in Fig. 2. (ii) CNN with two to five convolutional layers: The structure of this one was similar to the previous structure but we inserted 1–4 more convolutional layers and one more pooling layer between the current pooling layer and the fully-connected layer. In the additional convolutional layer, its depth dimension was 64 and the size of the mask was 3×3 with 1-pixel step shift, set to preserve the context of the image. The size of the image in the additional convolutional layer was $m \times m$ where $m = 22 - 2 \times (n - 1)$ and n was the number of convolutional layers. Additional pooling layer settings were set to be the same as those of the previous one in order to reduce the number of parameters in the classification phase. This architecture is shown in Fig. 3. In both architectures, a hidden layer in a fully connected architecture contained 256 neurons with five outputs.

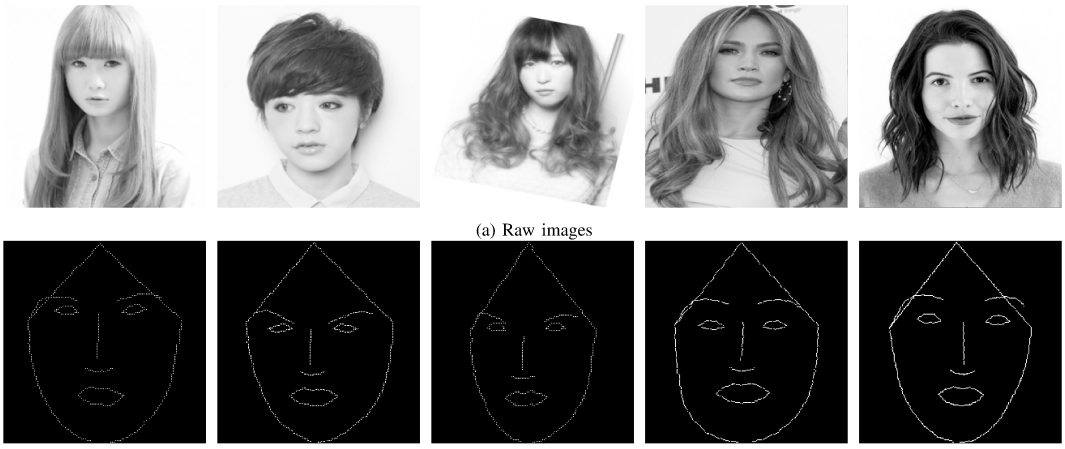
We applied all of the regularization techniques mentioned in the previous section one-by-one in this experiment as follows:

1) Overlap pooling method: the size of the mask in the first pooling layer was changed from 2×2 to 3×3 but the shift was still 2 pixels at a time, so the masks were overlapped.
2) Image augmentation technique: each image were horizontally flipped then augmented to the training set.
3) Dropout technique: We followed a good practice by [21] explained in the previous section. p range was $\{0.1, 0.2, \ldots, 1.0\}$ while n was set to 256. Therefore, the number of neurons in the hidden layer became $\frac{256}{p}$.

The experiment was run 20 times with different random seeds.

IV. Results and Discussion

Table I and II show the accuracy and AUROC of each method for 20 runs, respectively. These results were presented as violin plots that clearly show their distribution in Fig. 4a and 4b. We applied one-way analysis of variance (ANOVA) to analyze the differences between group means, and it was found that all of the means were different, confirming that there was a statistically significant difference between at least one pair of means at $p < 0.001$ for both accuracy and AUROC. Then, multiple comparisons were performed. The p-values are shown in Table III. It should be noted that we only discussed the accuracy because, overall, AUROC was very much the same



(a) Raw images

(b) Reconstructed images

Fig. 1: Examples of images trained by CNN. There were five-classes of face shapes: oval, round, oblong, square and heart (from left to right).

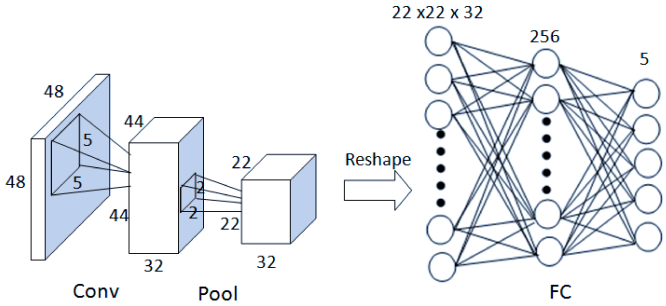


Fig. 2: CNN architecture with one convolutional layer.

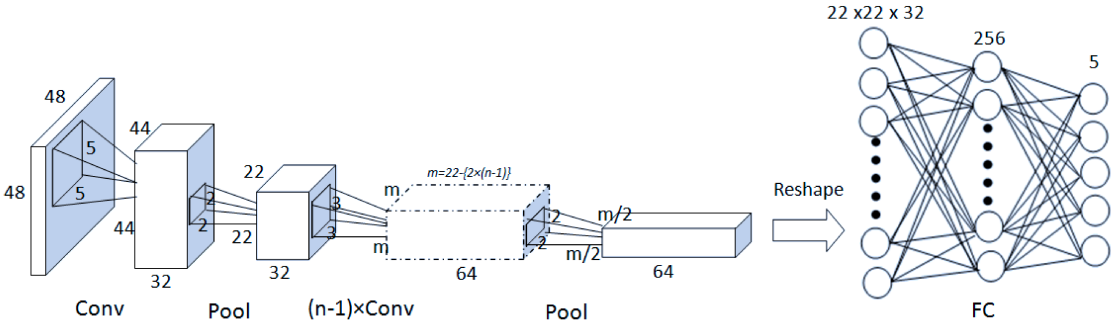


Fig. 3: CNN architecture with n convolutional layers (2–5).

as accuracy. According to the results, ANN achieved a better accuracy than SVM did at 60.2% and 58.7%, respectively, as shown in Table I, but the conclusion was statistically inconclusive ($p = 0.9863$), as shown in Table III.

We compared two types of data representation–raw and reconstructed images described in Section III-B. It is clear that using reconstructed images in CNN gave significantly better performance in accuracy than using raw images, at 53.00%, and 30.20% respectively ($p < 0.001$). Therefore, we only used reconstructed images in CNN with different types of

TABLE I: The classification accuracy of each algorithm on the test set.

Run	Shallow Model		Deep Model				
	19 Features		Raw	Reconstructed Image			
	SVM	ANN	CNN	CNN			
	RBF	1-hidden	Standard	Standard	Overlap	Aug	Dropout
1	56.00	54.00	18.00	54.00	54.00	50.00	56.00
2	60.00	60.00	26.00	46.00	52.00	54.00	52.00
3	56.00	60.00	34.00	44.00	56.00	46.00	50.00
4	62.00	56.00	32.00	57.00	57.00	56.00	66.00
5	58.00	58.00	28.00	54.00	52.00	56.00	57.00
6	62.00	60.00	36.00	44.00	44.00	44.00	50.00
7	68.00	62.00	28.00	60.00	57.00	60.00	70.00
8	56.00	60.00	22.00	56.00	60.00	46.00	46.00
9	64.00	68.00	30.00	52.00	60.00	54.00	70.00
10	58.00	58.00	38.00	52.00	50.00	57.00	57.00
11	66.00	60.00	30.00	57.00	60.00	62.00	60.00
12	38.00	62.00	24.00	48.00	46.00	52.00	50.00
13	54.00	60.00	34.00	62.00	57.00	60.00	54.00
14	56.00	58.00	34.00	50.00	54.00	50.00	56.00
15	48.00	62.00	24.00	54.00	64.00	52.00	64.00
16	66.00	68.00	42.00	56.00	66.00	66.00	70.00
17	54.00	60.00	30.00	56.00	62.00	57.00	64.00
18	62.00	64.00	34.00	52.00	56.00	64.00	66.00
19	70.00	58.00	30.00	56.00	62.00	57.00	57.00
20	60.00	56.00	30.00	50.00	46.00	52.00	48.00
Mean	58.70	60.20	30.20	53.00	55.75	54.75	58.15
STD	7.26	3.55	5.69	4.91	6.09	5.96	7.73

TABLE II: The area under the ROC curve of each algorithm on the test set.

Run	Shallow Model		Deep Model				
	19 Features		Raw	Reconstructed Image			
	SVM	ANN	CNN	CNN			
	RBF	1-hidden	Standard	Standard	Overlap	Aug	Dropout
1	75.00	72.50	56.25	68.75	66.25	70.00	67.50
2	72.50	71.25	48.75	71.25	71.25	68.75	72.50
3	75.00	75.00	53.75	66.25	70.00	71.25	70.00
4	72.50	75.00	58.75	65.00	72.50	66.25	68.75
5	76.25	72.50	57.50	73.75	73.75	72.50	78.75
6	73.75	73.75	55.00	71.25	70.00	72.50	73.75
7	76.25	75.00	60.00	65.00	65.00	65.00	68.75
8	80.00	76.25	55.00	75.00	73.75	75.00	81.25
9	72.50	75.00	51.25	72.50	75.00	66.25	66.25
10	77.50	80.00	56.25	70.00	75.00	71.25	81.25
11	73.75	73.75	61.25	70.00	68.75	73.75	73.75
12	78.75	75.00	56.25	73.75	75.00	76.25	75.00
13	61.25	76.25	52.50	67.50	66.25	70.00	68.75
14	71.25	75.00	58.75	76.25	73.75	75.00	71.25
15	72.50	73.75	58.75	68.75	71.25	68.75	72.50
16	67.50	76.25	52.50	71.25	77.50	70.00	77.50
17	78.75	80.00	63.75	72.50	78.75	78.75	81.25
18	71.25	75.00	56.25	72.50	76.25	73.75	77.50
19	76.25	77.50	58.75	70.00	72.50	77.50	78.75
20	81.25	73.75	56.25	72.50	76.25	73.75	73.75
Mean	74.19	75.13	56.38	70.69	72.44	71.81	73.94
STD	4.54	2.22	3.56	3.13	3.84	3.77	4.82

regularization and evaluated them.

The accuracy of CNN using overlap pooling, flipped-image augmentation, and dropout were 55.75%, 54.75%, and 58.15%, respectively. These results were clearly better than those obtained by conventional CNN on the average. Unfortunately, it was ambiguous to conclude that overlap pooling and flipped-image augmentation techniques were able to enhance the performance of CNN in this case, as the comparison resulted in high 0.7784 and 0.9698 p-value, respectively. Fortunately, dropout was able to improve the performance of CNN at $p = 0.0976$.

It was clear that the shallow models were able to achieve better performance than the deep models on the average. However, they were significantly better than only the deep model that did not use any regularization techniques ($p < 0.05$).

V. Conclusion

We statistically compared deep learning with shallow learning technique on a small dataset–a face shape dataset. We showed that SVM and ANN in conjunction with hand-crafted discriminative features were able to achieve better performances than deep learning techniques that used either raw or reconstructed images but no regularization technique. Moreover, it is ambiguous to say that a shallow model gave a higher accuracy than a deep model that used regularization techniques did, especially the case of CNN using dropout. Thus, we can conclude that using the dropout technique can handle overfitting in small dataset. We were able to build a comparable model of deep learning on a small dataset to a shallow model by using image transformation and regularization techniques.

Acknowledgement

We would like to thank the Information Security Advisory Group (ISAG) and Digital Image Processing Laboratory (DIP) at the Department of Computer Engineering, Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang for giving us access to their computing resources. This work was supported by the Thailand Research Fund under grant agreement number TRG5680090.

TABLE III: p-values from multiple comparisons of accuracy & AUROC.

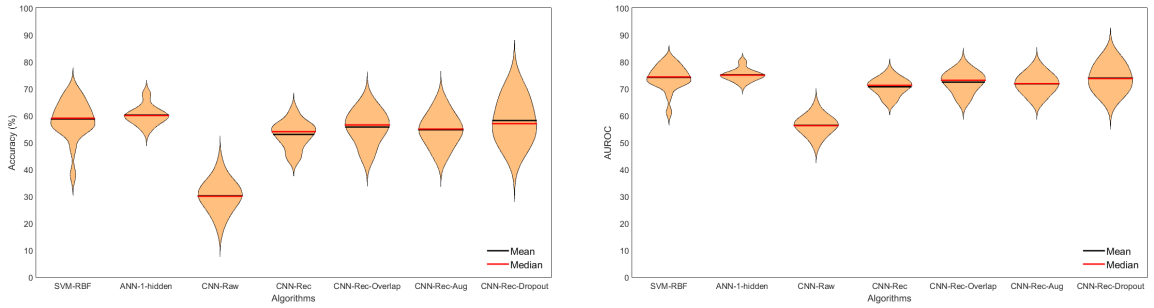
Algorithm 1	Algorithm 2	p-value	
		Accuracy	AUROC
SVM-RBF	ANN	0.9863	0.9866
SVM-RBF	CNN-Raw	< 0.001	< 0.001
SVM-RBF	CNN-Rec	< 0.050	0.0532
SVM-RBF	CNN-Rec-Overlap	0.7156	0.7667
SVM-RBF	CNN-Rec-Aug	0.3691	0.4237
SVM-RBF	CNN-Rec-Dropout	1.0000	1.0000
ANN	CNN-Raw	< 0.001	< 0.001
ANN	CNN-Rec	< 0.050	< 0.050
ANN	CNN-Rec-Overlap	0.2272	0.2703
ANN	CNN-Rec-Aug	0.0642	0.0819
ANN	CNN-Rec-Dropout	0.9353	0.9557
CNN-Raw	CNN-Rec	< 0.001	< 0.001
CNN-Raw	CNN-Rec-Overlap	< 0.001	< 0.001
CNN-Raw	CNN-Rec-Aug	< 0.001	< 0.001
CNN-Raw	CNN-Rec-Dropout	< 0.001	< 0.001
CNN-Rec	CNN-Rec-Overlap	0.7784	0.7667
CNN-Rec	CNN-Rec-Aug	0.9698	0.9660
CNN-Rec	CNN-Rec-Dropout	0.0976	0.0939
CNN-Rec-Overlap	CNN-Rec-Aug	0.9985	0.9985
CNN-Rec-Overlap	CNN-Rec-Dropout	0.8704	0.8726
CNN-Rec-Aug	CNN-Rec-Dropout	0.5589	0.5639

References

[1] R. F. Harrison and K. Pasupa, "Sparse multinomial kernel discriminant analysis (sMKDA)," *Pattern Recognition*, vol. 42, no. 9, pp. 1795–1802, 2009.

[2] D. Inman, "Machine learning applied to recognising hand-written Japanese," in *Proceeding of 4th IEEE International Workshop on Robot and Human Communication (RO-MAN 1995)*, 1995, pp. 117–122.

[3] C. Agarwal and A. Sharma, "Image understanding using decision tree based machine learning," in *Proceeding of International Conference on Information Technology and Multimedia (ICIM 2011)*, 2011, pp. 1–8.



(a) Accuracy (b) AUROC

Fig. 4: Violin plot of the performances shown by all of the methods.

[4] L. M. López-López, J. J. Castro-Schez, D. Vallejo-Fernandez, and J. Albusac, "A recommender system based on a machine learning algorithm for B2C portals," in *Proceeding of IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT 2009)*, 2009, pp. 524–531.

[5] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards, *Artificial intelligence: a modern approach*. Prentice Hall Upper Saddle River, 2003, vol. 2.

[6] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98–113, 1997.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceeding of Advances in Neural Information Processing Systems (NIPS 2012)*, 2012, pp. 1097–1105.

[8] NVIDIA. (2015) GPU-based deep learning inference: A performance and power analysis. [Online]. Available: https://www.nvidia.com/content/tegra/embedded-systems/pdf/jetson_tx1_whitepaper.pdf

[9] H.-W. Ng, V. D. Nguyen, V. Vonikakis, and S. Winkler, "Deep learning for emotion recognition on small datasets using transfer learning," in *Proceeding of ACM International Conference on Multimodal Interaction (ICMI 2015)*, 2015, pp. 443–449.

[10] J. Wagner, J. Foster, and J. van Genabith, "A comparative evaluation of deep and shallow approaches to the automatic detection of common grammatical errors," in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007, pp. 112–121.

[11] W. Sunhem and K. Pasupa, "An approach to face shape classification for hairstyle recommendation," in *Proceeding of 8th International Conference on Advanced Computational Intelligence (ICACI 2016)*, 2016, pp. 390–394.

[12] N. Naseer, K.-S. Hong, M. J. Khan, and M. R. Bhutta, "Comparison of artificial neural network and support vector machine classifications for fNIRS-based BCI," in *Proceeding of 15th International Conference on Control, Automation and Systems (ICCAS 2015)*, 2015, pp. 1817–1821.

[13] K. Kianmehr, S. Gao, J. Attari, M. M. Rahman, K. Akomeah, R. Alhajj, J. Rokne, and K. Barker, "Text summarization techniques: SVM versus neural networks," in *Proceedings of 11th International Conference on Information Integration and Web-based Applications & Services (iiWAS 2009)*, 2009, pp. 487–491.

[14] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proceedings of 23rd International Conference on Machine Learning (ICML 2006)*, 2006, pp. 161–168.

[15] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[16] S. Anwar and R. Ismal, "Robustness analysis of artificial neural networks and support vector machine in making prediction," in *Proceeding of IEEE 9th International Symposium on Parallel and Distributed Processing with Applications (ISPA 2011)*, 2011, pp. 256–261.

[17] T. D. Team. Convolutional neural networks (lenet). [Online]. Available: http://deeplearning.net/tutorial/lenet.html

[18] T. Dettmers. (2015) Deep learning in a nutshell: Core concepts. [Online]. Available: https://devblogs.nvidia.com/parallelforall/deep-learning-nutshell-core-concepts/

[19] S. Raschka. (2016) When does deep learning work better than SVMs or random forests. [Online]. Available: http://www.kdnuggets.com/2016/04/deep-learning-vs-svm-random-forest.html

[20] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *Proceeding of International Conference on Artificial Neural Networks (ICANN 2010)*, 2010, pp. 92–101.

[21] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[22] W. Sunhem, K. Pasupa, and P. Jansiripitikul, "Hairstyle recommendation for women," in *Proceeding of 5th ICT International Student Project Conference (ICT-ISPC 2016)*, 2016.