Cross-Domain Reuse of Extracted Knowledge in Genetic Programming for Image Classification

Muhammad Iqbal, *Member, IEEE,* Bing Xue, *Member, IEEE,* Harith Al-Sahaf, *Member, IEEE* and Mengjie Zhang, *Senior Member, IEEE*

***Abstract*—Genetic programming (GP) is a well-known evolutionary computation technique, which has been successfully used to solve various problems, such as optimisation, image analysis and classification. Transfer learning is a type of machine learning approach that can be used to solve complex tasks. Transfer learning has been introduced to genetic programming to solve complex Boolean and symbolic regression problems with some promise. However, the use of transfer learning with genetic programming has not been investigated to address complex image classification tasks with noise and rotations, where GP cannot achieve satisfactory performance, but GP with transfer learning may improve the performance. In this paper, we propose a novel approach based on transfer learning and genetic programming to solve complex image classification problems by extracting and reusing blocks of knowledge/information, which are automatically discovered from similar as well as different image classification tasks during the evolutionary process. The proposed approach is evaluated on three texture data sets and three office data sets of image classification benchmarks, and achieves better classification performance than the state-of-the-art image classification algorithm. Further analysis on the evolved solutions/trees shows that the proposed approach with transfer learning can successfully discover and reuse knowledge/information extracted from similar or different problems to improve its performance on complex image classification problems.**

***Index Terms*—Genetic Programming, Image Classification, Knowledge Extraction, Building Blocks, Code Fragments.**

I. Introduction

MACHINE learning algorithms have been successfully used to solve a variety of problems, such as classification and regression. Many traditional (supervised) machine learning models are learnt on the training data (source domain) and applied to test data (target domain), with the assumption that the source and target data are in the same feature space and follow the same underlying distribution. However, data in many real-world problems may not meet this assumption. When the feature space or the underlying distribution changes, the algorithm needs to re-train from scratch. In such situations, *knowledge transfer* or *transfer learning*, which aims to utilise the previously-acquired knowledge to solve similar problems, is very useful for increasing the effectiveness and efficiency in target problem solving, and has gained increasing attention in recent years [1]. Layered learning introduced by Stone and Veloso [2] is a subclass of transfer learning, which aims to address a complex learning task by decomposing it into a hierarchy of subtask layers. The knowledge learned from the lower layer (i.e. a simpler subtask) is used to learn the task at higher layers (i.e. a more complex subtask).

Genetic programming (GP) is an evolutionary computation (EC) approach in which computer programs are automatically evolved to solve a target problem. Since GP is problem-independent, and has a flexible representation and powerful search ability, it has been used to solve a variety of machine learning tasks including symbolic regression and classification [3]. In particular, the flexible representation in GP, which is usually a variable-length (tree) representation, provides a natural way of automatically finding useful knowledge/information during the learning/evolutionary process. These are often called building blocks, extracted knowledge or blocks of knowledge [4]. Therefore, among all EC algorithms, GP is one of the earliest and most frequently investigated algorithms with transfer learning [5].

Image classification, which is to assign one of the predefined class labels to each image, is a key task in computer vision with a wide range of real-world applications. Image classification often requires feature extraction, feature selection, and feature construction to form a set of (high-level) features from the raw pixels, which can then be fed to a classification/learning algorithm to perform classification. GP using a tree-based representation can automatically achieve feature extraction, feature construction, feature selection, and classification simultaneously, so GP has been used for image classification with good success [6]–[9]. However, image classification is a challenging problem, and many existing methods suffer from the issues of low classification accuracy and a long training time. Particularly when the images are complex, such as with different kinds of noise and different degrees of rotation, many algorithms fail to achieve satisfactory performance. It is worth noting that the classification of images all of which have been rotated at the same angle, e.g., 30° or 60°, usually does not differ from the classification of the unrotated images. However, when the rotated images are combined with the unrotaetd images, the classification task becomes difficult especially when the features extracted from the images are not robust or rotation-invariant. Transfer learning, particularly layered learning, might be a promising approach to solving complex image classification tasks by utilising the knowledge learnt from simpler tasks, e.g., unrotated images without noise, to help learn a good classification model on complex images, e.g., images with noise and rotations.

Transfer learning in GP has been investigated to solve

The authors are with the Evolutionary Computation Research Group (ECRG) at the School of Engineering and Computer Science, Victoria University of Wellington, PO Box 600, Wellington 6140, New Zealand.

complex Boolean and symbolic regression problems with success [10]–[13], but there are only a few works on GP with transfer learning for image classification [14]. Furthermore, "when to transfer" and "how to transfer" are two key parts in GP with transfer learning. However, regarding "when to transfer", most existing GP [15] with transfer learning methods focus on transferring knowledge only at the initialization of the GP learning/evolution process, which does not fully utilise the extracted useful knowledge. Regarding "how to transfer", existing methods [15] often transfer an evolved tree from the source domain to the target domain as a whole tree, which may not be promising, but the evolved tree might be very useful if it is used as subtree to form a new tree/solution for the target problem. Recently, Iqbal et al. [16] presented a transfer learning method in GP that successfully utilizes the extracted knowledge at the initialization process as well as the mutation process in GP to learn image classification tasks in the *same* domain.

A. Goal

The main aim of this study is to further investigate transfer learning mechanisms in GP to learn image classification problems from *similar* as well as *different* target domains. To achieve this goal, the following objectives have been set.

- identify and extract potentially useful blocks of knowledge in learning image classification tasks in a source domain,
- reuse the extracted knowledge to solve complex image classification tasks from similar as well as different target domains,
- investigate the effect of different amounts of the transferred knowledge on the classification accuracy and size of the evolved GP programs as well as on the required training time, and
- analyse the evolved GP programs to get insights about the reuse of extracted knowledge.

B. Organization

The rest of the paper is organized as follows. Section II describes the necessary background in transfer learning, and image analysis using genetic programming. In Section III, the proposed GP with transfer learning approach is detailed. Section IV introduces the data sets and parameter settings used in the experimentation. In Section V, experimental results obtained using the knowledge transfer within the same domain are presented and compared with the baseline GP system. Section VI describes the experimental results obtained by reusing the extracted knowledge from a source domain to learn image classification problems from other *related* and *different* target domains. Section VII provides an analysis of the newly introduced parameters in the proposed transfer learning based GP method. Section VIII explains in detail the reuse of extracted knowledge in the proposed approach. In Section IX, this work is concluded and the future work is outlined in Section IX-A.

II. Background

In this section, a brief review of the related work on reusing the extracted knowledge in evolutionary computation is provided. Then, the specific related work on transfer learning and image analysis using GP is described.

A. Transfer Learning in Evolutionary Computation

Transfer learning has been widely used in machine learning techniques [17]–[19]. However, it is a relatively less explored research area in the evolutionary computation community.

Santana et al. [20] utilized different statistical measures to extract the potentially useful information in evolutionary algorithms. They reported that the computed measures can be used to analyse the algorithm behavior and predict the problem difficulty.

In [21], transfer learning was introduced into a genetic algorithm (GA) to solve a complex problem. A proportion of the initial population in GA for the target domain was formed by the final solution of the GA population from the source domain. The results have shown that GA with transfer learning achieved better performance than without transfer learning. Moshaiov and Tal [22] presented a transfer learning based family bootstrapping paradigm. The evolved solutions to a common source task were reused to create initial populations for a family of six related target tasks. The reported results showed that the transfer learning based initial populations successfully evolved solutions for all the six target tasks, which were previously not solvable by using the randomly generated initial populations.

The reuse of learnt knowledge for an enhanced search performance in dynamic optimization problems has been investigated by using different memory schemes [23], [24]. Feng et al. [25] proposed a new memetic computation paradigm [26] to transfer the learned knowledge from previously solved problems in order to improve future evolutionary searches. The reported results on routing problems indicated the usefulness of knowledge transfer.

Gupta et al. [27] introduced the paradigm of evolutionary multitasking to solve multiple optimisation problems simultaneously using a single population of evolving individuals. They proposed a cross-domain optimisation technique to implicitly transfer the created genetic material among different problems. The reported results showed that the proposed technique often converged rapidly for a number of complex optimisation tasks. Ong and Gupta [28] described in detail the evolutionary multitasking approach to handling various real-world multitasking problems from different domains.

Recently, Lim et al. [29] investigated the reuse of extracted knowledge in a novel 'simultaneously problem learning and evolutionary optimization' approach. The reported results on a composites manufacturing task showed that the reuse of the learnt knowledge can significantly reduce the engineering design time.

In this study, we propose a GP-based transfer learning technique for image classification. The proposed technique utilizes the automatic knowledge construction ability of GP to extract useful knowledge from only two images in each class in the source domain, and reuse the extracted knowledge to facilitate the learning of GP on solving more difficult image classification tasks in various similar as well as different target domains. The rest of this section describes the related work on transfer learning and image analysis using GP.

B. Transfer Learning in Genetic Programming

The idea of transfer learning in GP has been investigated to solve complex Boolean and symbolic regression problems, where the standard monolithic GP may not find a solution due to the large search space. Most of the existing work is layered learning in GP to solve complex problems, which decomposed a complex problem into subtasks and each subtask is learned in a bottom-up fashion [2].

Jackson and Gibbons [5] used a two-layered GP approach to solving Boolean logic problems of the even-parity and the majority-on problem, and obtained better performance than standard GP. Later, Gustafson and Hsu [30] introduced the idea of layered learning in GP to solve a complex multi-agent system task, i.e., to learn the keep-away soccer game. The task was decomposed into two subtasks, where the final population from the bottom layer task learning was used as the initial population for the top layer task learning. The results showed that GP with layered learning outperformed the conventional GP without layered learning.

Hien et al. [11] investigated layered learning with incremental sampling in GP for symbolic regression. The results on twelve benchmark problems showed that the proposed GP approach outperformed standard GP in terms of reducing the training time and the complexity of the solutions. Further, to overcome the ad-hoc parameter setting issue in [11], Hien and Hoai [31] incorporated parameter setting techniques derived from progressive sampling for GP with the incremental sampling based layered learning for symbolic regression.

The relationship between evolution, development, and layered learning was investigated in [12] based on tree adjoining grammar guided GP (TAG3P) [32]. The proposed system named DTAG3P was examined on symbolic regression problems, Boolean even-parity problems, and ORDERTREE problems, where the results showed that the layered learning DTAG3P system obtained more structured and scalable solutions to the problems than two single-shot learning GP systems: standard tree-based GP [4] and TAG3P [32]. However, the DTAG3P system introduced a number of new parameters into the TAG3P system.

Iqbal et al. [33] used transfer learning in a GP-based learning classifier system to solve complex Boolean problems, where the results showed that by using layered learning, the proposed system achieved better performance than the existing learning classifier and GP systems and solved the 135-bit multiplexer problem for the first time. Later on, Alvarez et al. [34] enhanced the transfer learning based classifier system to solve the n-bit multiplexer problem.

Recently, Chen et al. [13] investigated the combination of GP with gradient descent for transfer learning in symbolic regression. Further, Dinh et al. [15] proposed transfer learning in GP by using the learnt knowledge from the source domain to initialize the population of GP on the target domain in three different ways, which are copying the full trees, using the sub-trees, and copying the best trees in each generation. The results on symbolic regression problems showed that the sub-tree transferring strategy outperformed the other two.

C. Image Analysis using Genetic Programming

Since GP has a flexible representation, GP can automatically extract, construct and select features based on row pixels in images [8]. GP has been widely used for image analysis [35]–[38], where typical works are reviewed in this section. A detailed description of this work is beyond the scope of this article, readers are refereed to [39]–[45] for more work on EC or GP for image analysis.

In 1996, Poli [46] showed how GP can be used for image analysis by automatically extracting and constructing image features. Smart and Zhang [47] developed a GP based multi-class image recognition approach by dynamically determining the class boundaries between different classes, which increased the performance of GP for difficult multi-class image classification problems. Ryan et al. [48] applied a general GP based image classification approach to the task of Stage 1 cancer detection in digital mammograms, which showed that GP can evolve good classifiers to detect as many cancer tissue as possible without being overly conservative (i.e. involving too many callbacks which increase the radiologists' workload and patients' stress and worry), and GP was also performed as part of a feedback loop, to both select and help generate better features.

Hindmarsh et al. [49] used GP to perform classification based on a list of pre-extracted Scale-invariant Feature Transform (SIFT) [50] features in order to improve the performance over the use of SIFT features directly for object recognition. Recently, in [51], a multi-objective GP approached was used to automatically generate domain-adaptive global feature descriptors for image classification, where a set of primitive 2-D operators were combined to construct feature descriptors shown by GP trees, and each tree was then evaluated based on the classification accuracy and the tree complexity. The results show that the proposed approach outperformed many state-of-the-art hand-designed features and two other feature learning techniques in terms of the classification accuracy.

Al-Sahaf et al. [8] proposed a GP-based image classification approach to automatically evolving descriptors, i.e., a feature vector, to perform classification, where the proposed method was called *GP-criptor*. The feature vector generated in GP-criptor is similar to local binary patterns (LBP) [52] and the main difference is that in LBP the formulae are designed by a domain-expert, whereas those formulae are automatically evolved by GP in GP-criptor. Experiments on two widely-used texture image data sets showed that the GP-criptor achieved better performance than two recent GP-based methods, i.e., the static range selection method and the dynamic range selection method, and nine well-known non-GP methods, i.e., support vector machines, naive Bayes, naive Bayes/decision trees, KStar, multilayer perceptron, adaptive boosting M1, decision trees (J48), and random forest [53]. The GP-based methods operate directly on pixel values whereas the non-GP methods use pre-computed image features, e.g. local binary patterns, gray-level co-occurrence matrices, and domain-independent features.

Lensen et al. [54] showed that GP can automatically select key regions, extract informative features, and perform classification to achieve better classification than other state-of-the-art methods. Furthermore, the features extracted by GP is also general to other classification methods, i.e, to maintain or improve the performance of these methods for the tested image classification tasks.

D. Image Analysis and Transfer Learning

The main aim of this study is to investigate the use of knowledge transfer in GP for image classification rather than designing a general competitor for image classification tasks. However, for the sake of completeness, related work from the machine learning domain is briefly discussed here.

Quattoni et al. [55] learned a sparse prototype representation using unlabeled images and a kernel function; and reused the learnt representation for image classification. The reported results on a news-topic prediction task showed that the transferred representation significantly improved the image prediction accuracy. Zhu et al. [56] proposed a heterogeneous transfer learning framework for image classification. They successfully utilized textual information from tagged images to classify the Caltech-256 image data set [57].

Cireşan et al. [58] analyzed transfer learning in deep neural networks to transfer knowledge among various character recognition tasks. They successfully transferred knowledge from digits to letters, and from Chinese characters to Latin letters. Kandaswamy et al. [59] trained a convolutional neural network to classify Latin digits and reused the trained model to classify lowercase letters. They also transferred knowledge from Arabic digits to Latin digits, and from English handwritten digits to English lowercase letters. Donahue et al. [60] extracted deep convolutional activation features (DeCAF) from a neural network trained on a large set of object recognition tasks. The extracted features were successfully reused to learn various generic visual recognition tasks.

Ghifary et al. [61] proposed a domain-adaptive neural networks model that utilizes the maximum mean discrepancy measure [62] to learn domain-invariant features in a feed-forward neural net. Later on, Ghifary et al. [63] proposed a multi-task auto-encoder (MTAE) algorithm to learn domain-invariant image features from multiple data sets in an attempt to increase domain generalization. The reported results show that MTAE produced state-of-the-art performance in learning various object recognition tasks.

It is worth highlighting that the typical fore-mentioned transfer learning based methods often operate in a sparse fashion and require either a large amount of labeled data from source domains or unlabeled data from target domains. However, the proposed transfer learning based GP method operates in a pixel-by-pixel (dense) fashion, with only using two-instance-per-class data from the source domain. This makes it hard to fairly compare the proposed method with some existing algorithms such as [55], [56]. In another view, this could be potentially the strength of the proposed method in a situation when people cannot collect many unlabeled data.

Although transfer learning in GP has shown improvements to solve Boolean and regression problems, it has not been widely used in image analysis. Jaśkowski et al. [14], [64] recently proposed a GP based transfer learning method for enabling effective code reuse to solve a set of related visual learning tasks in a handwritten character recognition problem. The results show that code reuse leads to better results in terms of fitness and recognition accuracy. This work shows the potential of transfer learning in GP for image classification tasks.

Recently, Iqbal et al. [16], [65] introduced transfer learning in GP-criptor [8] to extract blocks of useful knowledge from simple texture image classification problems, and then reused the extracted knowledge to learn complex problems. The obtained results indicate that reusing the extracted knowledge improves classification accuracy in learning various texture image classification tasks in the *same* domain. This study aims to further investigate transfer learning mechanisms to learn complex image classification tasks from *similar* as well as *different* image classification domains.

III. The Proposed Method

This section first briefly describes the baseline method, GP-criptor [8], which is the state-of-the-art GP method for texture classification. In GP-criptor, each GP individual is represented in a special tree form that facilitates the extraction and reuse of knowledge. Following that, the proposed method of incorporating knowledge reuse in GP-criptor, named *transfer learning* GP-criptor (TLGP-criptor), is explained in detail.

A. The Baseline Method

In the baseline method [8], the given data set is divided into two sets containing an equal number of images. One set is used for training and the other for testing. From the training set, two images per class are randomly selected to be used to train/evolve a GP program. After evolving a GP program, the images being used in the training process are passed to the evolved GP program to generate a set of knowledge base vectors of size equal to twice the number of classes in the given data set. The set of knowledge base vectors is used to compute the fitness values of GP individuals in the training process and also serves as a knowledge base in the testing process to help in classifying the unseen images. In short, the training process produced two outputs: an evolved program and a set of knowledge base vectors. During the testing process, each image from the test set is passed to the evolved GP program and a feature vector is generated. Finally, *the-Nearest-Neighbor* (1NN) [66] classifier is applied to classify the image by using the generated feature vector and the set of knowledge base vectors.

It is to be noted that GP-criptor uses only two images per class in the training process to save computational time. Therefore, to evolve better GP individuals, during the training, the typically used fitness measure of just counting the

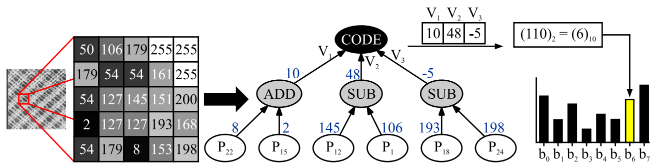


Fig. 1. The process of feature vector generation using GP-criptor [8].

correct classified images is not suitable. In order to better differentiate images from different classes in GP-criptor, the fitness measure has been designed to include a component of distance-based measure in addition to the counting of correctly classified images. The distance measure used is *Czekanowski Coefficient* [67], which is one of the widely used measures in computer vision. Czekanowski Coefficient determines the distance between two vectors [68], and is formally defined as:

$$\text{Dis}\left(\vec{u},\vec{v}\right) = 1 - \frac{2\left(\sum_{i=1}^{E}\min\left(\vec{u}_i,\vec{v}_i\right)\right)}{\sum_{i=1}^{E}\vec{u}_i + \sum_{i=1}^{E}\vec{v}_i} \tag{1}$$

where $\vec{u}$ and $\vec{v}$ are the two vectors, the number of elements in a vector is E (the two vectors are of identical length, i.e., $E = |\vec{u}| = |\vec{v}|$), and the i^{th} element of $\vec{u}$ and $\vec{v}$ is, respectively, $\vec{u}_i$ and $\vec{v}_i$. For details of the fitness function used, interested readers are referred to the original paper [8].

GP-criptor operates directly on raw pixel values of an image. A sliding window, of a specific size, is used to scan the input image, pixel by pixel, from left to right and from top to bottom. The pixel values (denoted by P_i, where $0 \leq i < window\ size$) falling within the window are used as terminal values in a GP program. For example, a sliding window of size 5×5 is being used to scan an image shown in Fig. 1. Therefore, the GP terminals in this example are denoted as P_0, P_1, P_2, ..., P_{24}; where P_0 corresponds to the top left pixel value in the sliding window (i.e. 50 in the shown scenario) and P_{24} to the right bottom pixel value (i.e. 198 in the shown scenario).

The function set used is {ADD, SUB, MUL, DIV, CODE}, where CODE (described below) is always the root node of a tree generated in GP-criptor. The length of a feature vector generated in GP-criptor is always equal to 2^n where n is the number of children of the CODE node. The process to generate a feature vector using GP-criptor is shown in Fig. 1. The system computes values V_1, V_2, ..., V_n. Next the CODE operator generates a binary code by converting each computed value to either 0 or 1 using the following rule: if the computed value is greater than zero then set it to 1, otherwise set it to 0. Then the generated binary code is converted to decimal, which is used to increment the corresponding bin (denoted as b_0, b_1, ..., b_{2^n-1}) in the feature vector, e.g., in the shown scenario, bin b_6 will be incremented.

B. The New Method

This section describes the motivation and novelty of knowledge extraction and reuse/transfer in the new method called TLGP-criptor. In this work, we will identify and extract useful knowledge, in the form of *code fragments* [33], [34], from learning simpler image classification problems, and then reuse the extracted knowledge to learn complex, e.g., rotated and noisy, problems of the *same* domain. We will also investigate the effectiveness of reusing the extracted knowledge to solve image classification problems from other *similar* as well as very *different* domains.

The overall proposed approach for extracting and reusing knowledge to solve image classification problems is depicted in Fig. 2. First of all, GP-criptor is applied to learn an initial image classification task. When the training process is completed in GP-criptor, potentially useful code fragments are extracted (described below) from the evolved GP population. These extracted code fragments are used in TLGP-criptor to learn the next image classification task. Similarly, when the training process is completed in TLGP-criptor, potentially useful code fragments are extracted from the evolved GP population to be used to learn other image classification tasks using TLGP-criptor.

TLGP-criptor will use the same fitness function and evaluation process as used in the baseline method (i.e. GP-criptor). However, the generation and evolution of GP programs will be modified by incorporating the ability of knowledge transfer in the GP process. The flowchart shown in Fig. 3 describes the modifications incorporated in the GP process to extract and reuse knowledge in TLGP-criptor, which are the main difference between GP-criptor and TLGP-criptor. Specifically, the extracted code fragments are used to generate GP individuals in the initial GP population, and to mutate a GP individual in the mutation process. The rest of this section explains these new mechanisms in detail.

1) Identification and extraction of useful knowledge: As depicted in Fig. 1, each child c of the root node in a GP program evolved in GP-criptor is actually a mathematical expression, which is evaluated to a value V_c by applying to pixel values of an input image falling within the sliding window. Each computed value V_c is converted to 0 or 1 and used in the construction of a feature vector. For example, an evolved GP program using GP-criptor is shown in Fig. 4. This GP program consists of the following eight expressions:

- `(DIV P4 P6)`,
- `(ADD P5 P5)`,
- `(DIV P1 (MUL P8 P4))`,
- `(SUB P6 (MUL P3 (MUL P1 (DIV P7 P1))))`,
- `(ADD P5 P2)`,
- `(SUB P6 P1)`,
- `(DIV P7 P8)`, and
- `(ADD P0 P0)`

Therefore, we consider each child of the root node a potentially useful code fragment or block of knowledge. Once the training process is completed, we calculate average fitness of the whole population of GP programs in the last generation. Then we select GP programs that have fitness value better than or equal to the average fitness value of the whole population. Code fragments of the selected GP programs are extracted and stored to be reused to learn complex problems of the same as well as different domains. It is to be noted that in this study the extracted code fragments are not unique.

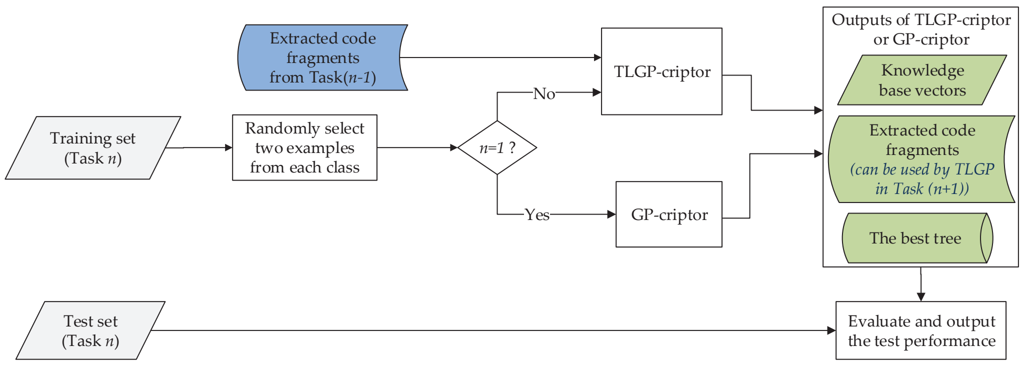

Extracted code fragments from Task(n-1)
Training set (Task n)
Randomly select two examples from each class
n=1 ?
No
Yes
TLGP-criptor
GP-criptor
Outputs of TLGP-criptor or GP-criptor
Knowledge base vectors
Extracted code fragments (can be used by TLGP in Task (n+1))
The best tree
Test set (Task n)
Evaluate and output the test performance

Fig. 2. The proposed approach for extracting and reusing knowledge to solve image classification problems. Here n denotes the number of classification tasks.

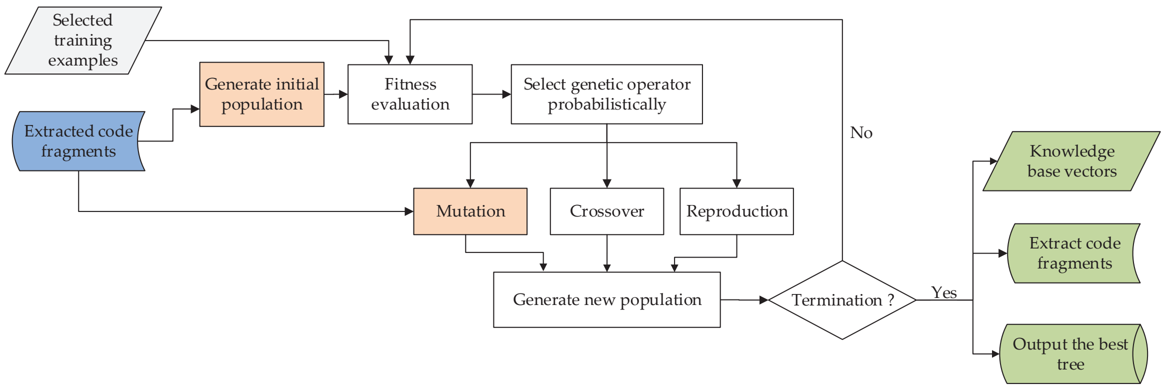


Fig. 3. A flowchart shows how transfer learning is integrated into the GP process.

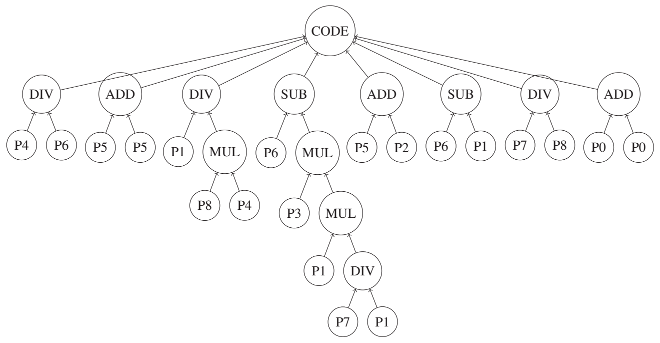


Fig. 4. An example of a GP program generated by GP-criptor.

Algorithm 1: TLGP-criptor_Initial_Program

```
Data: The number of children $n$ of the root node in the GP
      program $p$ to be generated.
Result: A newly generated GP program $p$.
1  initialize GP program $p$
2  initialize root node's children $p.children$ with length $n$
3  for $i$ = 1 to $n$ do
4      if RandomNumber[0, 1) $< \mu_I$ then
5          $cf \leftarrow$ randomly select an extracted code fragment
6      else
7          $cf \leftarrow$ randomly generate a new subtree
8      end
9      $p.children[i] \leftarrow cf$
10 end
11 return $p$
```

2) Reusing the extracted knowledge: The extracted code fragments are used to generate the initial GP populations as well as to mutate GP programs in learning complex problems of the domain. The reuse of extracted knowledge is explained below.

Program generation: In GP-criptor, each child of the root node in a GP program in the initial population is a randomly generated subtree. However, in TLGP-criptor it is either an extracted code fragment with probability μ_I, or a randomly generated subtree with probability $1-\mu_I$. This is described in Algorithm 1. It is to be noted that the code fragments being reused may be modified in a crossover operation to evolve, hopefully, better GP individuals.

New mutation operator: The baseline GP-criptor method uses the standard GP mutation operation. In GP-criptor, the mutation operation randomly selects a *subtree* in a GP program and replaces it with a randomly generated *new subtree*. In TLGP-criptor, a GP program is mutated either by randomly selecting a *child of the root node* and replacing it with an *extracted code fragment* with probability μ_M, or using the standard GP mutation operation (as used in GP-criptor) with probability $1-\mu_M$. The new mutation operation is described in Algorithm 2.

It is to be noted that Algorithm 2 is used to mutate a GP individual, which has been (pre)selected from the GP population, for mutation, with a certain probability that is

Algorithm 2: TLGP-criptor_Mutation_Operation

```
Data: A GP program $p$ to be mutated.
Result: The mutated GP program $p$.
1  $n \leftarrow$ number of children of the root node in $p$
2  if RandomNumber[0, 1) $< \mu_M$ then
3      $i \leftarrow$ RandomNumber[1, n]
4      $cf \leftarrow$ randomly select an extracted code fragment
5      $p.children[i] \leftarrow cf$
6  else
7      $t1 \leftarrow$ randomly select a subtree in $p$
8      $t2 \leftarrow$ randomly generate a new subtree
9      replace $t1$ in $p$ with $t2$
10 end
11 return $p$
```

usually set to 0.19. It is also worth noting that the new mutation operation still maintains the diversity, to a certain extent, by using the standard mutation mechanism (as used in GP-criptor) with probability $1 - \mu_M$.

It is worth noting that typical machine learning methods for knowledge transfer are trained in a semi-supervised fashion, i.e., involving unlabeled data from the target domain or from both the source and the target domains [18], [60], [69]–[72]. TLGP-criptor, however, is fully-supervised, with using only two instances from each class for training. This makes it hard to compare TLGP-criptor with many existing algorithms. In another view, this could be potentially the strength of TLGP-criptor in a situation when we even cannot collect many unlabeled data.

IV. Experiment Design

The proposed transfer learning based method, TLGP-criptor, is tested using three texture image data sets and an office data set. Results are compared with the baseline method, GP-criptor. The effectiveness of transfer learning in TLGP-criptor is measured using the mean classification accuracy in various source and target domain configurations. The rest of this section describes the data sets, parameter settings, and different experimental configurations used in this study.

A. Data Sets

In this study, experiments have been conducted using three widely used texture image classification data sets, i.e., Kylberg [73], Brodatz [74], and Outex [75]; and a more challenging office data set that consists of three different image domains, i.e., amazon, dslr, and webcam [76].

In the Kylberg data set, each instance is a gray-scale image of size 576×576 pixels. To reduce the computational cost, each image is resized to 115×115 pixels in this work. There are total 28 classes in the Kylberg data set, as shown in Fig. 5. The instances of the Kylberg data set are divided into two groups based on the orientation, i.e., with rotation and without rotation. There are 160 image instances in each class of the unrotated group. On the other hand, the rotated group consists of 1920 instances where each image is rotated at successive $30°$ angles. Fig. 6 shows the unrotated *cushion1* image from the Kylberg data set and its 11 rotated variations.

Each class of the Brodatz data set has a single gray-scale instance of size 640×640 pixels and there are total 112 classes. We use 20 randomly selected classes and re-sample the single instance of each class to 84 sub-images (each of size 64×64 pixels). Each image in the rotated version is rotated at successive $30°$ angles, resulting in total 1008 instances.

The Outex data set without rotation, OutexTC00000, contains 24 classes where each class consists of 20 gray-scale images of size 128×128 pixels. On the other hand, the rotated version of the Outex data set, OutexTC00010, consists of 180 instances in each class, where each image is rotated at $5°$, $10°$, $15°$, $30°$, $45°$, $60°$, $75°$ and $90°$ angles.

For the noise experiments, the additive white Gaussian noise was added in each unrotated data set in order to examine the effectiveness of transfer learning in noisy problems. For Gaussian distribution, the standard deviation σ was increased from 10^{-4} to 10^{1} in 20 steps with linearly spaced exponents, i.e., the 20 noise levels are equally spaced in a $\log_{10}$ scale. Fig. 7 shows the original *cushion1* image from the Kylberg data set and its 20 noisy variations. It is worth noting that in the last six noisy images in Fig. 7, the texture of the *cushion1* image has totally disappeared due to a severe level of noise. For the sake of completeness, we will investigate the effect of knowledge reuse at that severe level of noise, however, we do not expect any major improvement in classification accuracy.

The office data set consists of three domains, i.e., amazon, dslr and webcam, where each domain contains 31 categories of office images. In the original office data set, images of different categories have different sizes. In this study, all the images have been resized to 200×200. The amazon domain consists of medium resolution images of products downloaded from www.amazon.com. The dslr domain consists of high resolution images captured with a digital single-lens reflex (dslr) camera in realistic environments. The webcam domain consists of low resolution images recorded with a simple webcam. Fig. 8 shows some exemplary images of *back_pack* from each domain of the office data set. It is worth noting that images in the amazon domain have more variations in shape and size than the other two domains. These variations can make it difficult for GP-criptor to generate a general GP individual in the amazon domain using only two image instances per class.

B. Parameter Settings

The GP platform provided by the Evolutionary Computation Java-based software (ECJ) [77] is used to implement the baseline method, i.e., GP-criptor as well as the proposed transfer learning based method, i.e., TLGP-criptor.

In this paper, parameter settings are similar to the settings used in the baseline method [8], which are described here. As processing images at pixel level is a time consuming task in general, the number of GP individuals in a GP population is set to only 100 to save computational time. The initial GP population is generated using the ramped half-and-half method. In order to maintain the population diversity, we use the tournament selection strategy with a tournament size set to 7. The probabilities of using mutation, crossover, and

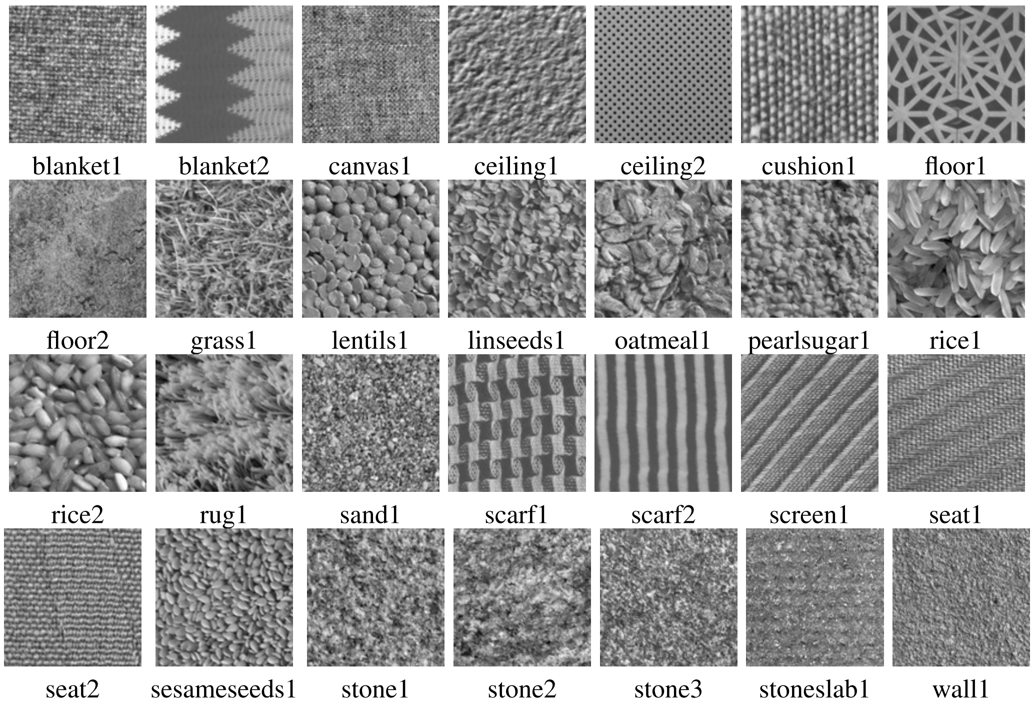


Fig. 5. Examples of images from the Kylberg data set.

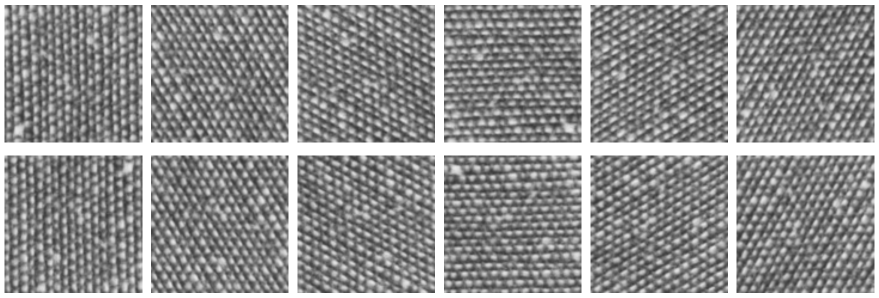


Fig. 6. Original *cushion1* image from the Kylberg data set (top left) and its 11 variations, each with a successive 30° rotation, from left to right and top to bottom.

reproduction operations are 0.19, 0.80, and 0.01 respectively. The elitism is used with ratio 0.01, i.e., the best one individual from the GP population of 100 individuals is passed to the next generation. The tree depth of an evolved program is confined between the levels 2 and 10. The evolutionary process stops under two conditions: (a) when an ideal individual is found, or (b) when the maximum number of generations is reached. The size of sliding window throughout the various experiments is set equal to 3×3 pixels and the number of children of the CODE node is set equal to 8. The probability μ_I of reusing the extracted code fragments to generate an initial GP tree, as described in Algorithm 1, is initially set to 0.5 in order to use the learned and the new genetic material with the same probability. Similarly, the probability μ_M of reusing the extracted code fragments to mutate a GP tree, as described in Algorithm 2, is initially set to 0.5. Subsequently, we analyse these parameters using different values in the experimental study.

C. Experiment Configurations

In this study, the *unrotated* Kylberg, Brodatz, and Outex data sets are denoted with Ky, Br and Ou; and the *rotated* versions with KyR (which comprises images of 12 different rotation angles), BrR (12 different rotation angles), and OuR (9 different rotation angles), respectively.

The effectiveness of transfer learning within the *same domain* is measured using the original texture data sets (without rotation and noise) as source domains and various rotated and noisy versions of the original texture data sets as target domains. We used various combinations of rotated versions of the texture data sets: (1) standalone single-rotation, data set_r, where r denotes a rotation angle, e.g., KyR_120 denotes the KyR data set that contains all images rotated at 120°, (2) unrotated plus single-rotation, data set_000_r, e.g., KyR_000_120 denotes the KyR data set that contains all images rotated at 120° as well as the unrotated images, and (3) unrotated plus incremental rotations, data set_upto_r, e.g., KyR_upto_120 denotes the KyR data set that contains all images rotated at 30°, 60°, 90°, and 120° as well as the unrotated images. Note

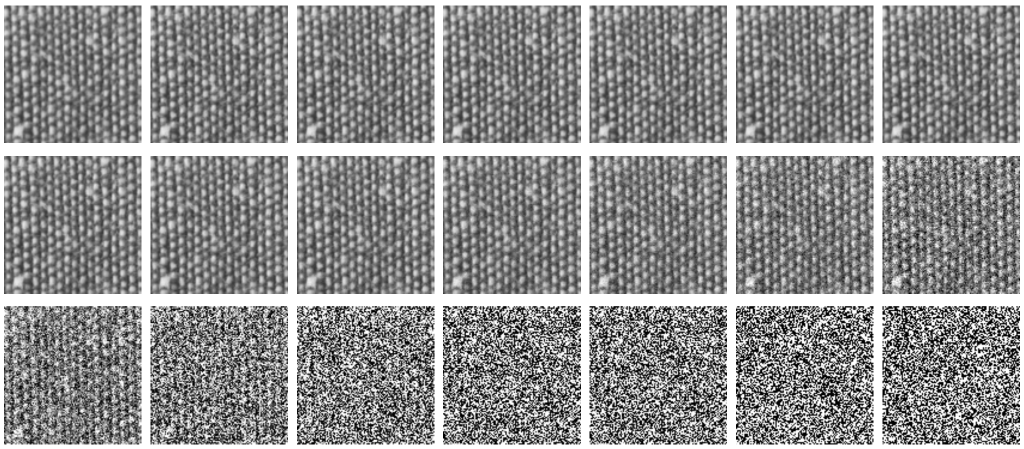


Fig. 7. Original *cushion1* image from the Kylberg data set (top left) and its 20 noisy variations, from left to right and top to bottom. Each noisy version has an increased level of noise with σ from 10^{-4} to 10^{1}.

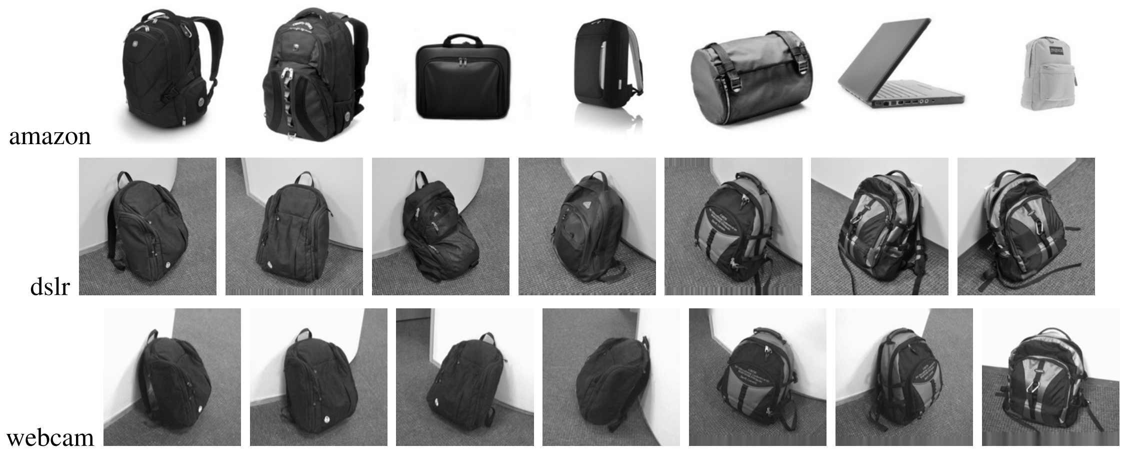


Fig. 8. Some exemplary *back_pack* images from each domain of the office data set, i.e., amazon, dslr, and webcam.

that KyR_upto_330, BrR_upto_330 and OuR_upto_90 are the same as KyR, BrR and OuR, respectively.

The usefulness of knowledge transfer in *cross-domain* image classification problems is also thoroughly investigated. To this end, knowledge is extracted from one texture/office domain and reused to learn another texture/office domain. In addition, reuse of knowledge from a texture domain to an office domain is also investigated.

Further, various experiments have been conducted to analyse the *amount of knowledge* to be transferred relative to the classification performance and size of the evolved GP programs.

As used in the baseline method [8], for each experiment in this study, we divide the given data set into two (i.e. training and test) pools containing an equal number of images. The training set is formed by randomly selecting only two images from each class of the training pool. Consequently, the typically used process of 30 independent runs for a stochastic method is repeated 10 times using different images in the training set. Both mean and standard deviations statistics are computed and reported for the test set.

To analyze the results, the Wilcoxon signed rank test is applied on the classification accuracy obtained in the test set to determine whether there was any statistically significant difference with a confidence interval of 95%, which is denoted by **bold face** in the result tables presented in the following sections.

V. Results I - Knowledge Transfer within the Same Problem Domain

This section presents results of transferring knowledge within the same problem domain, e.g., extracting knowledge from the unrotated Kylberg data set and reusing the extracted knowledge to learn a rotated version of the Kylberg data set.

It is to be noted that the mean classification accuracy in learning the unrotated data sets is the same for both GP-criptor and TLGP-criptor (i.e. 87.26 ± 0.48 in Ky, 94.72 ± 0.56 in Br, and 94.97 ± 0.47 in Ou) as there is no extracted knowledge to be used by TLGP-criptor. The initial population and mutation (which are the main differences) are identical in both methods for this case. However, after learning the unrotated data sets, potentially useful code fragments have been extracted that are to be used by TLGP-criptor to learn various rotated and noisy versions of image classification problems.

TABLE I
THE ACCURACIES (%) ON THE TEST SETS FOR THE ROTATED DATA SETS (MEAN ± STANDARD DEVIATION).

Data Set	GP-criptor	TLGP-criptor
KyR	48.81 ± 2.09	**56.01 ± 1.39**
BrR	55.27 ± 2.15	**61.52 ± 1.32**
OuR	65.10 ± 1.39	**69.13 ± 0.89**

TABLE II
THE ACCURACIES (%) ON THE TEST SETS FOR THE UNROTATED DATA SETS (MEAN ± STANDARD DEVIATION).

Method	Ky	Br	Ou
GP-criptor	87.26 ± 0.48	94.72 ± 0.56	**94.97 ± 0.47**
TLGP-criptor	87.26 ± 0.48	94.72 ± 0.56	**94.97 ± 0.47**
CLBC	76.67 ± 4.05	63.62 ± 2.52	72.79 ± 2.87
DRLBP	86.26 ± 1.14	83.17 ± 2.49	84.96 ± 1.79
DeCAF	**94.01 ± 1.14**	**97.14 ± 1.09**	93.58 ± 1.37

TABLE III
THE ACCURACIES (%) ON THE TEST SETS FOR THE ROTATED DATA SETS (MEAN ± STANDARD DEVIATION).

Method	KyR	BrR	OuR
GP-criptor	48.81 ± 2.09	55.27 ± 2.15	65.10 ± 1.39
TLGP-criptor	56.01 ± 1.39	61.52 ± 1.32	69.13 ± 0.89
CLBC	76.58 ± 3.77	70.84 ± 3.19	75.53 ± 2.24
DRLBP	74.05 ± 2.07	69.65 ± 2.40	63.97 ± 2.57
DeCAF	85.05 ± 2.02	90.37 ± 1.35	76.91 ± 2.67
GP-criptor^{ri}	**88.5 ± 1.4**	**92.5 ± 1.1**	**86.8 ± 1.9**

A. *Overall Results*

Table I shows the mean classification accuracy and standard deviation obtained in learning the rotated data sets using GP-criptor and TLGP-criptor. It has shown that transfer learning has significantly improved the classification performance of TLGP-criptor over GP-criptor in learning the three rotated data sets.

1) Knowledge Transfer in Initialisation vs Mutation: In the proposed method, the knowledge transfer is done in two steps, i.e., the initialization and the mutation. In order to investigate whether the knowledge transfer in the two steps is necessary, we applied TLGP-criptor without knowledge transfer in the mutation on KyR, BrR, and OuR. The obtained results (i.e. 54.84 ± 1.30 in KyR, 60.64 ± 1.47 in BrR, and 68.53 ± 0.80 in OuR) show that the performance of TLGP-criptor is degraded. However, it is still statistically significantly better than the performance of GP-criptor. Subsequently, TLGP-criptor's results shown in rest of this article have been obtained using the knowledge transfer in both the initialization and the mutation steps.

2) Comparison with Other Methods: A main goal of this paper is to investigate whether introducing knowledge transfer into GP-criptor can significantly improve the classification performance, and this goal has been achieved. In this sub section, we also "compare" GP-criptor and TLGP-criptor with the state-of-the-art descriptors for image classification. The state-of-the-art image descriptors are: completed local binary count (CLBC) [78], dominant rotated local binary patterns (DRLBP) [79], and deep convolutional activation features (DeCAF) [60]. Since these methods are all designed to be robust to rotation, we also include a new version of GP-criptor considering rotation invariance, GP-criptor^{ri} [80], in the comparison on the rotated images. As the design goals of these image descriptors are different, a direct comparison of the absolute classification results of different methods in Table II and Table III might not be totally fair and it is not the goal of this paper, but such a comparison can provide some indicator on which method should be used in different scenarios.

As can be seen from the results shown in Table II, for non-rotated images on Ky, Br and Ou, GP-criptor (or TLGP-criptor) did very well, outperformed CLBC and DRLBP in all cases, and performed slightly better than DeCAF on Ou. However, DeCAF performed better than all on Ky and Br. These results indicate that for classifying non-rotated images DeCAF is a better choice.

For rotated images, the GP-criptor^{ri} method performed much better than all other state-of-the-art methods as shown in Table III. Accordingly, when classifying rotated texture images, GP-criptor^{ri} should be chosen as a priority. In the future, we will investigate how knowledge transfer can further improve the performance of GP-criptor^{ri}. This study is focused on investigating the proposed knowledge transfer mechanisms in GP-criptor.

B. *Kylberg Data Set with Single and Multiple Rotations*

The left part of Table IV shows the mean classification accuracy and standard deviation obtained in learning the rotated version of KyR having only one rotation at a time. It is observed that the classification performance of GP-criptor in learning the standalone single-rotation version is similar to the original unrotated version, as expected. However, reusing the extracted code fragments from Ky data set, transfer learning has significantly improved the classification performance of TLGP-criptor in learning the standalone single-rotation version of KyR.

The middle part of Table IV shows the mean classification accuracy and standard deviation obtained in learning the rotated version containing single-rotation plus the original unrotated images. As this rotated version is more complex than the standalone single-rotation version, the classification performance of GP-criptor decreased from approximately 87% to approximately 63%. In this rotated version, transfer learning again significantly improves the classification performance of TLGP-criptor over the baseline GP-criptor method for all cases.

The right part of Table IV shows the mean classification accuracy and standard deviation obtained in learning the Kylberg data set with incremental rotations. In this data set, code fragments are extracted from Ky to learn KyR_upto_030, then code fragments are extracted from KyR_upto_030 to learn KyR_upto_060, and so on. The incremental reuse of extracted knowledge in TLGP-criptor has shown noticeable improvement in learning Kylberg rotated data set. The performance of TLGP-criptor is improved from 56.01 ± 1.39 to 62.07 ± 0.98 in learning the whole rotated data set. The statistical analysis shows that the improvement is statistically significant in all of the rotated versions.

TABLE IV
THE ACCURACIES (%) ON THE TEST SETS FOR THE KYR DATA SET WITH SINGLE AND MULTIPLE ROTATIONS (MEAN ± STANDARD DEVIATION). THE CLASSIFICATION ACCURACY IN LEARNING THE KY DATA SET WAS 87.26 ± 0.48.

Data Set	GP-criptor	TLGP-criptor	Data Set	GP-criptor	TLGP-criptor	Data Set	GP-criptor	TLGP-criptor
KyR_030	87.19 ± 0.60	**89.07 ± 0.38**	KyR_000_030	63.87 ± 1.50	**68.10 ± 0.76**	KyR_upto_030	63.87 ± 1.50	**68.10 ± 0.76**
KyR_060	87.99 ± 0.72	**89.30 ± 0.40**	KyR_000_060	59.41 ± 1.96	**64.04 ± 0.88**	KyR_upto_060	56.64 ± 1.44	**61.21 ± 0.75**
KyR_090	87.73 ± 0.62	**89.15 ± 0.38**	KyR_000_090	60.09 ± 1.23	**63.52 ± 1.12**	KyR_upto_090	51.87 ± 1.89	**58.64 ± 0.72**
KyR_120	87.51 ± 0.73	**89.28 ± 0.46**	KyR_000_120	58.35 ± 1.51	**63.19 ± 1.10**	KyR_upto_120	47.92 ± 1.37	**56.21 ± 0.98**
KyR_150	87.88 ± 0.74	**89.43 ± 0.34**	KyR_000_150	64.47 ± 1.55	**67.80 ± 0.97**	KyR_upto_150	46.59 ± 1.69	**54.53 ± 1.09**
KyR_180	87.93 ± 0.54	**89.12 ± 0.34**	KyR_000_180	75.97 ± 0.86	**78.69 ± 0.88**	KyR_upto_180	45.07 ± 1.88	**54.75 ± 0.98**
KyR_210	87.51 ± 0.63	**89.19 ± 0.46**	KyR_000_210	62.83 ± 1.42	**66.93 ± 1.03**	KyR_upto_210	48.34 ± 1.75	**57.05 ± 1.01**
KyR_240	87.81 ± 0.73	**89.35 ± 0.32**	KyR_000_240	58.70 ± 1.51	**62.98 ± 1.27**	KyR_upto_240	46.92 ± 2.15	**57.03 ± 0.76**
KyR_270	87.78 ± 0.77	**89.18 ± 0.34**	KyR_000_270	59.33 ± 1.77	**63.71 ± 1.03**	KyR_upto_270	44.85 ± 1.86	**56.84 ± 1.05**
KyR_300	87.47 ± 0.62	**89.22 ± 0.37**	KyR_000_300	59.53 ± 1.41	**63.46 ± 0.90**	KyR_upto_300	47.07 ± 2.24	**57.24 ± 0.86**
KyR_330	87.91 ± 0.58	**89.30 ± 0.42**	KyR_000_330	66.89 ± 1.49	**70.02 ± 0.64**	KyR_upto_330	48.81 ± 2.09	**62.07 ± 0.98**

C. Brodatz Data Set with Single and Multiple Rotations

The left part of Table V shows the mean classification accuracy and standard deviation obtained in learning the rotated version of Br (BrR) having only one rotation at a time. It is observed that reusing the extracted code fragments from Br data set, transfer learning has improved the classification performance of TLGP-criptor over GP-criptor.

The middle part of Table V shows the mean classification accuracy and standard deviation obtained in learning the rotated version containing single-rotation plus the original unrotated images. In this rotated version, the classification performance of GP-criptor decreased from approximately 96% to approximately 78%.

The right part of Table V shows the mean classification accuracy and standard deviation obtained in learning the BrR data set with incremental rotations. The incremental reuse of extracted knowledge in TLGP-criptor has shown noticeable improvement in learning the BrR data set. The performance of TLGP-criptor is improved from 61.52 ± 1.32 to 73.88 ± 0.87 in learning the whole rotated data set. The statistical analysis shows that the improvement is significant in all of the rotated versions.

D. Outex Data Set with Single and Multiple Rotations

The left part of Table VI shows the mean classification accuracy obtained in learning the rotated version of OuR having only one rotation at a time. It is observed that reusing the extracted code fragments from Ou, transfer learning has improved the classification performance of TLGP-criptor in learning the standalone single-rotation version of OuR. The statistical analysis shows that this improvement is significant in all of the rotated versions, except OuR_010 and OuR_015.

The middle part of Table VI shows the mean classification accuracy obtained in learning the rotated version containing single-rotation plus the original unrotated images. It is observed that in this data set the classification accuracy of GP-criptor noticeably decreased when the rotation was greater or equal to $15°$. In this rotated version, transfer learning has statistically significantly improved the classification performance of TLGP-criptor over the baseline GP-criptor method.

The right part of Table VI shows the mean classification accuracy and standard deviation obtained in learning the rotated version of the Outex data set with incremental rotations. The incremental reuse of extracted knowledge in TLGP-criptor has shown noticeable improvement in learning the OuR data set. The performance of TLGP-criptor is improved from 69.13 ± 0.89 to 72.27 ± 0.79 in learning the whole rotated data set. The statistical analysis shows that the improvement is statistically significant.

E. Kylberg, Brodatz, and Outex Data Sets with Noise

After successful application of knowledge reuse in various rotated versions of three texture data sets, we investigated the effectiveness of knowledge reuse in presence of different levels of noise in a data set. To this end, we added the additive white Gaussian noise in the unrotated texture data sets, i.e., Ky, Br, and Ou.

Table VII shows the mean classification accuracy and standard deviation obtained in learning the unrotated data sets with various noise levels. In the first 10 noise levels, GP-criptor showed robustness to noise and maintained the classification performance to a non-noisy level. After that, its performance gradually decreased. At the last six noise levels, the performance is noticeably decreased (as expected) because the severe noise destroyed the textures as shown in the last six noisy images of the *cushion1* image in Fig. 7. Similar to the rotated versions, TLGP-criptor showed improvement in classification accuracy in noisy data sets. The statistical analysis shows that the improvement is statistically significant for a reasonable amount of noise that maintains the texture of the input image.

F. Summary

In summary, the proposed technique of transfer learning in GP has shown significant improvement in learning various complex texture image classification problems. It is observed that in learning rotated versions that contain previously seen images, TLGP-criptor has shown substantial improvement. Specifically, in the incremental learning of rotated images, the performance improvement TLGP-criptor was noticeably significant over the baseline GP-criptor method. Further, for small amount of noise, transfer learning has also shown improvement.

VI. RESULTS II - CROSS DOMAIN KNOWLEDGE TRANSFER

In the previous section, extracted knowledge was used within the same image data sets with different rotations and different levels of noise, which improved the classification

TABLE V
THE ACCURACIES (%) ON THE TEST SETS FOR THE BRR DATA SET WITH SINGLE AND MULTIPLE ROTATIONS (MEAN $\pm$ STANDARD DEVIATION). THE CLASSIFICATION ACCURACY IN LEARNING THE BR DATA SET WAS 94.72 $\pm$ 0.56.

Data Set	GP-criptor	TLGP-criptor	Data Set	GP-criptor	TLGP-criptor	Data Set	GP-criptor	TLGP-criptor
BrR_030	97.02 $\pm$ 0.35	**97.66 $\pm$ 0.29**	BrR_000_030	81.39 $\pm$ 1.38	**84.16 $\pm$ 0.73**	BrR_upto_030	81.39 $\pm$ 1.38	**84.16 $\pm$ 0.73**
BrR_060	96.06 $\pm$ 0.39	**96.63 $\pm$ 0.22**	BrR_000_060	74.44 $\pm$ 1.09	**76.73 $\pm$ 0.89**	BrR_upto_060	71.98 $\pm$ 1.33	**77.93 $\pm$ 1.07**
BrR_090	95.01 $\pm$ 0.42	**96.07 $\pm$ 0.36**	BrR_000_090	76.30 $\pm$ 1.27	**79.02 $\pm$ 0.87**	BrR_upto_090	64.19 $\pm$ 1.93	**74.97 $\pm$ 0.82**
BrR_120	96.75 $\pm$ 0.33	**97.18 $\pm$ 0.23**	BrR_000_120	72.17 $\pm$ 1.28	**74.67 $\pm$ 0.85**	BrR_upto_120	59.30 $\pm$ 2.46	**72.58 $\pm$ 0.70**
BrR_150	97.05 $\pm$ 0.37	**97.45 $\pm$ 0.24**	BrR_000_150	78.62 $\pm$ 1.46	**81.34 $\pm$ 0.71**	BrR_upto_150	58.50 $\pm$ 1.99	**71.72 $\pm$ 1.22**
BrR_180	94.62 $\pm$ 0.48	**95.31 $\pm$ 0.31**	BrR_000_180	90.47 $\pm$ 0.70	**92.28 $\pm$ 0.43**	BrR_upto_180	58.10 $\pm$ 2.14	**72.86 $\pm$ 0.93**
BrR_210	97.58 $\pm$ 0.29	**97.94 $\pm$ 0.20**	BrR_000_210	77.87 $\pm$ 1.21	**81.01 $\pm$ 0.88**	BrR_upto_210	57.91 $\pm$ 1.85	**73.99 $\pm$ 0.86**
BrR_240	97.39 $\pm$ 0.37	**97.85 $\pm$ 0.19**	BrR_000_240	74.03 $\pm$ 1.28	**76.60 $\pm$ 0.71**	BrR_upto_240	56.18 $\pm$ 1.61	**73.33 $\pm$ 0.98**
BrR_270	94.25 $\pm$ 0.47	**95.40 $\pm$ 0.40**	BrR_000_270	76.30 $\pm$ 0.91	**78.65 $\pm$ 0.98**	BrR_upto_270	54.93 $\pm$ 2.35	**73.95 $\pm$ 0.89**
BrR_300	96.49 $\pm$ 0.38	**97.06 $\pm$ 0.31**	BrR_000_300	73.80 $\pm$ 1.12	**76.88 $\pm$ 0.81**	BrR_upto_300	54.20 $\pm$ 2.44	**72.69 $\pm$ 0.93**
BrR_330	96.88 $\pm$ 0.33	**97.32 $\pm$ 0.19**	BrR_000_330	82.75 $\pm$ 1.24	**84.74 $\pm$ 0.95**	BrR_upto_330	55.27 $\pm$ 2.15	**73.88 $\pm$ 0.87**

TABLE VI
THE ACCURACIES (%) ON THE TEST SETS FOR THE OUR DATA SET WITH SINGLE AND MULTIPLE ROTATIONS (MEAN $\pm$ STANDARD DEVIATION). THE CLASSIFICATION ACCURACY IN LEARNING THE OU DATA SET WAS 94.97 $\pm$ 0.47.

Data Set	GP-criptor	TLGP-criptor	Data Set	GP-criptor	TLGP-criptor	Data Set	GP-criptor	TLGP-criptor
OuR_005	94.97 $\pm$ 0.48	**95.39 $\pm$ 0.29**	OuR_000_005	94.59 $\pm$ 0.44	**95.49 $\pm$ 0.31**	OuR_upto_005	94.59 $\pm$ 0.44	**95.49 $\pm$ 0.31**
OuR_010	95.75 $\pm$ 0.43	95.74 $\pm$ 0.33	OuR_000_010	92.37 $\pm$ 0.53	**93.83 $\pm$ 0.39**	OuR_upto_010	94.02 $\pm$ 0.42	**95.55 $\pm$ 0.38**
OuR_015	95.83 $\pm$ 0.43	95.84 $\pm$ 0.37	OuR_000_015	88.39 $\pm$ 0.94	**89.85 $\pm$ 0.59**	OuR_upto_015	92.89 $\pm$ 0.66	**94.65 $\pm$ 0.38**
OuR_030	96.19 $\pm$ 0.57	**96.75 $\pm$ 0.37**	OuR_000_030	77.86 $\pm$ 1.17	**80.03 $\pm$ 0.80**	OuR_upto_030	85.21 $\pm$ 0.98	**88.74 $\pm$ 0.38**
OuR_045	96.24 $\pm$ 0.55	**97.09 $\pm$ 0.35**	OuR_000_045	73.78 $\pm$ 1.18	**75.90 $\pm$ 0.79**	OuR_upto_045	80.03 $\pm$ 0.97	**84.78 $\pm$ 0.65**
OuR_060	95.05 $\pm$ 0.46	**95.64 $\pm$ 0.38**	OuR_000_060	72.07 $\pm$ 0.99	**74.20 $\pm$ 0.83**	OuR_upto_060	75.52 $\pm$ 1.29	**80.60 $\pm$ 0.71**
OuR_075	95.16 $\pm$ 0.38	**95.53 $\pm$ 0.30**	OuR_000_075	71.20 $\pm$ 1.04	**73.98 $\pm$ 0.58**	OuR_upto_075	70.37 $\pm$ 1.63	**76.39 $\pm$ 0.76**
OuR_090	95.07 $\pm$ 0.42	**95.45 $\pm$ 0.40**	OuR_000_090	72.37 $\pm$ 1.21	**75.45 $\pm$ 0.70**	OuR_upto_090	65.10 $\pm$ 1.39	**72.27 $\pm$ 0.79**

TABLE VII
THE ACCURACIES (%) ON THE TEST SETS FOR THE UNROTATED NOISY DATA SETS (MEAN $\pm$ STANDARD DEVIATION).

Noise Level	Ky		Br		Ou	
(σ)	GP-criptor	TLGP-criptor	GP-criptor	TLGP-criptor	GP-criptor	TLGP-criptor
0.0001	87.26 $\pm$ 0.48	**88.60 $\pm$ 0.44**	94.72 $\pm$ 0.56	**95.92 $\pm$ 0.23**	94.97 $\pm$ 0.47	**95.52 $\pm$ 0.31**
0.0002	87.26 $\pm$ 0.48	**88.60 $\pm$ 0.44**	94.72 $\pm$ 0.56	**95.92 $\pm$ 0.23**	94.97 $\pm$ 0.47	**95.52 $\pm$ 0.31**
0.0003	87.26 $\pm$ 0.48	**88.60 $\pm$ 0.44**	94.72 $\pm$ 0.56	**95.92 $\pm$ 0.23**	94.97 $\pm$ 0.47	**95.52 $\pm$ 0.31**
0.0006	87.20 $\pm$ 0.74	**88.57 $\pm$ 0.34**	94.78 $\pm$ 0.55	**95.85 $\pm$ 0.35**	95.15 $\pm$ 0.39	**95.55 $\pm$ 0.20**
0.0011	87.18 $\pm$ 0.62	**88.50 $\pm$ 0.47**	94.64 $\pm$ 0.46	**95.85 $\pm$ 0.30**	95.19 $\pm$ 0.43	**95.59 $\pm$ 0.32**
0.0021	87.18 $\pm$ 0.54	**88.70 $\pm$ 0.27**	94.85 $\pm$ 0.55	**95.95 $\pm$ 0.34**	95.34 $\pm$ 0.35	**95.55 $\pm$ 0.31**
0.0038	87.27 $\pm$ 0.61	**88.59 $\pm$ 0.45**	94.79 $\pm$ 0.46	**95.93 $\pm$ 0.25**	95.18 $\pm$ 0.41	**95.61 $\pm$ 0.21**
0.0070	87.48 $\pm$ 0.54	**88.73 $\pm$ 0.35**	94.89 $\pm$ 0.61	**95.87 $\pm$ 0.32**	94.74 $\pm$ 0.40	**95.20 $\pm$ 0.30**
0.0127	87.46 $\pm$ 0.54	**88.80 $\pm$ 0.37**	94.94 $\pm$ 0.35	**95.75 $\pm$ 0.34**	93.64 $\pm$ 0.47	**94.25 $\pm$ 0.32**
0.0234	87.35 $\pm$ 0.66	**88.60 $\pm$ 0.40**	94.59 $\pm$ 0.47	**95.41 $\pm$ 0.35**	91.44 $\pm$ 0.49	**92.33 $\pm$ 0.60**
0.0428	86.92 $\pm$ 0.49	**88.15 $\pm$ 0.43**	93.30 $\pm$ 0.62	**94.37 $\pm$ 0.35**	85.96 $\pm$ 0.84	**86.96 $\pm$ 0.80**
0.0785	83.15 $\pm$ 0.77	**84.29 $\pm$ 0.52**	91.25 $\pm$ 0.64	**92.28 $\pm$ 0.50**	**68.04 $\pm$ 1.55**	67.05 $\pm$ 1.12
0.1438	71.68 $\pm$ 1.35	**73.60 $\pm$ 1.03**	86.07 $\pm$ 1.02	**86.75 $\pm$ 0.84**	35.52 $\pm$ 2.02	34.74 $\pm$ 1.66
0.2637	43.04 $\pm$ 1.70	**44.22 $\pm$ 1.29**	67.66 $\pm$ 1.05	**69.53 $\pm$ 0.73**	17.18 $\pm$ 0.71	17.22 $\pm$ 0.62
0.4833	11.74 $\pm$ 0.87	**12.84 $\pm$ 0.61**	44.94 $\pm$ 0.81	**47.00 $\pm$ 0.84**	14.66 $\pm$ 0.38	14.72 $\pm$ 0.45
0.8859	5.68 $\pm$ 0.19	5.76 $\pm$ 0.20	30.42 $\pm$ 0.94	**31.07 $\pm$ 0.73**	14.47 $\pm$ 0.45	14.50 $\pm$ 0.55
1.6238	4.81 $\pm$ 0.12	4.82 $\pm$ 0.14	19.89 $\pm$ 0.59	19.74 $\pm$ 0.60	14.41 $\pm$ 0.49	14.52 $\pm$ 0.44
2.9764	4.79 $\pm$ 0.13	4.79 $\pm$ 0.14	14.32 $\pm$ 0.61	14.35 $\pm$ 0.48	14.39 $\pm$ 0.52	14.40 $\pm$ 0.38
5.4556	4.67 $\pm$ 0.13	4.69 $\pm$ 0.16	**10.90 $\pm$ 0.44**	10.53 $\pm$ 0.46	14.35 $\pm$ 0.59	14.49 $\pm$ 0.46
10.0000	4.75 $\pm$ 0.13	4.77 $\pm$ 0.11	8.45 $\pm$ 0.46	8.37 $\pm$ 0.31	14.30 $\pm$ 0.37	14.36 $\pm$ 0.41

accuracy of the proposed method over the baseline method. However, cross-domain reuse of the extracted knowledge is not necessarily to be beneficial in all scenarios. If the source domain and the target domain are negatively related, the transferred knowledge might degrade the performance in the target domain.

In order to investigate whether the extracted knowledge is useful for learning other image data sets than the one from which it was extracted, we conducted the following three sets of experiments to reuse knowledge: (1) from a texture domain to another texture domain, (2) from an office domain to another office domain, and (3) from a texture domain to an office domain. It is anticipated that the reuse of extracted knowledge in (1) and (2) will be more beneficial than that in (3) as the source and target domains in the former are more related than that in the latter.

A. Knowledge Transfer among Texture Domains

We conducted the following six experiments to extract and reuse knowledge among different source and target domains from three texture data sets: Ky $\rightarrow$ BrR, Ky $\rightarrow$ OuR, Br $\rightarrow$ KyR, Br $\rightarrow$ OuR, Ou $\rightarrow$ KyR, and Ou $\rightarrow$ BrR. The obtained results are shown in Table VIII along with the results of transferring knowledge within the same domain (as shown in Table I).

It is observed that the cross-domain transfer of extracted knowledge in TLGP-criptor has achieved a similar improvement in the classification accuracy as within the same domain. This shows that the extracted code fragments are domain

TABLE VIII
THE ACCURACIES (%) ON THE TEST SETS FOR CROSS DOMAIN KNOWLEDGE TRANSFER USING DIFFERENT TEXTURE DATA SETS AS SOURCE AND TARGET DOMAINS (MEAN ± STANDARD DEVIATION).

KyR 48.81 ± 2.09	BrR 55.27 ± 2.15	OuR 65.10 ± 1.39
Ky → KyR **56.01 ± 1.39**	Ky → BrR **61.20 ± 1.38**	Ky → OuR **68.79 ± 1.08**
Br → KyR **55.76 ± 1.09**	Br → BrR **61.52 ± 1.32**	Br → OuR **68.33 ± 0.76**
Ou → KyR **56.73 ± 0.86**	Ou → BrR **61.52 ± 1.52**	Ou → OuR **69.13 ± 0.89**

TABLE IX
THE ACCURACIES (%) ON THE TEST SETS FOR CROSS DOMAIN KNOWLEDGE TRANSFER USING DIFFERENT SOURCE AND TARGET DOMAINS FROM THE OFFICE DATA SET (MEAN ± STANDARD DEVIATION).

amazon 07.72 ± 0.52	dslr 36.91 ± 2.04	webcam 33.00 ± 0.96
amazon → amazon N. A.	amazon → dslr 33.65 ± 1.55	amazon → webcam 28.80 ± 1.81
dslr → amazon **09.51 ± 0.70**	dslr → dslr N. A.	dslr → webcam **34.45 ± 1.32**
webcam → amazon **09.81 ± 0.51**	webcam → dslr **40.56 ± 1.04**	webcam → webcam N. A.

independent. It is worth noting that the gain in classification improvement achieved in learning KyR from Ou is more than the gain achieved by reusing the knowledge from Ky. These findings indicate not only that it is feasible to reuse/transfer knowledge among different *related* problem domains, but also that more gain could be obtained in this way than the knowledge reuse within the same domain.

B. Knowledge Transfer among Office Domains

In order to investigate the effectiveness of the proposed approach in learning more challenging image classification problems other than the texture benchmarks, we conducted a set of experiments using the office data set [76].

Table IX shows results obtained using different source and target domains, i.e., amazon → dslr, amazon → webcam, dslr → amazon, dslr → webcam, webcam → amazon, webcam → dslr.

It is observed that reusing the extracted knowledge in TLGP-criptor significantly improved the classification accuracy in learning the office data set except amazon → dslr and amazon → webcam. The amazon domain consists of images mostly having more than one object in a single image. Further, it is found that there are some discrepancies in the amazon domain, e.g., a laptop image in the *back_pack* images. Due to these difficulties and using only two images per class in the training process, the classification accuracy of the baseline GP-criptor method is very small in the amazon domain as shown in the first row in Table IX. The poor performance of GP-criptor in the amazon domain resulted in poor code-fragments to be used to learn amazon → dslr and amazon → webcam, hence no improvement in the classification accuracy.

In order to investigate the effect of using more than two images per class in GP-criptor, we conducted three additional experiments on the amazon domain for 5, 10, and 15 images per class. It improved the classification accuracy to 12.56 ± 2.49, 20.21 ± 2.36, and 28.12 ± 2.27 using 5, 10, and 15 images per class respectively. Further improvement is possible by using an increased population size of GP individuals. However, this is beyond the scope of this study.

TABLE X
THE ACCURACIES (%) ON THE TEST SETS FOR CROSS DOMAIN KNOWLEDGE TRANSFER FROM A TEXTURE DATA SET TO THE OFFICE DATA SET (MEAN ± STANDARD DEVIATION).

amazon 07.72 ± 0.52	dslr 36.91 ± 2.04	webcam 33.00 ± 0.96
KBO → amazon **10.88 ± 0.94**	KBO → dslr **44.09 ± 0.97**	KBO → webcam **37.95 ± 1.03**

C. Knowledge Transfer from Texture to Office Domains

Furthermore, we performed a set of experiments to investigate the usefulness of knowledge transfer among very *different* problem domains. To this end, we combined three texture data sets, i.e., Ky, Br, and Ou into a single large data set, denoted by KBO. Then, we conducted the following three experiments to transfer knowledge from the texture domain to the office domain: KBO → amazon, KBO → dslr, KBO → webcam. The obtained results are shown in Table X.

It is observed that the reuse of knowledge from the combined texture data set to learn different office domains resulted in an improvement in the classification accuracy that is noticeably greater than the classification accuracy obtained by reusing the knowledge within different office domains. This might be due to the fact that the texture data sets are simpler than the office data set, so, better code fragments learned in the former improved performance in the latter. This finding suggests that it is beneficial to first learn easy tasks and then reuse the learned knowledge to solve more complex tasks.

It is worth noting that the reuse of knowledge in cross-domain experiments might cause problems when users have no prior knowledge about the negative relations between source domains and target domains. However, there was no evidence of negative transfer in all experiments conducted in this study. It is anticipated that the inherent adaptiveness of GP has automatically eliminated negatively transferred knowledge via evolutionary principles.

VII. PARAMETER ANALYSIS

In previous sections, we used the extracted knowledge in the initialization step with probability μ_I, and in the mutation step with probability μ_M where both μ_I and μ_M were empirically set equal to 0.5. It is worth noting that in this study the code fragments being reused were not kept static in order to avoid overfitting. These code fragments were allowed to be modified in the crossover operation and to be replaced in the mutation operation with new code fragments.

In this section, we experimentally analyze the effect of probability μ_I and μ_M on the classification accuracy and size of the evolved GP programs as well as on the required training time in TLGP-criptor. To this end, we conducted 100 experiments, with μ_I and μ_M ranging from 0.1 to 1.0, over the BrR data set. The obtained results are shown in Table XI, Table XII, and Table XIII.

It is observed that TLGP-criptor produced significantly better classification accuracy than GP-criptor for any value of μ_I and μ_M as shown in Table XI. Further, an increase in the value of μ_I, mostly, results in an increase in the classification accuracy of TLGP-criptor, as shown in each column of Table XI. However, the accuracy does not increase much when $\mu_I > 0.7$. The parameter μ_M exhibits almost a similar behavior in terms of classification accuracy for any value from 0.1 to 1.0, as shown in each row of Table XI.

On the other hand, an increase in the value of μ_I, mostly, results in an increase in the size of the evolved GP programs in TLGP-criptor as shown in each column of Table XII. The parameter μ_M, exhibits almost a similar behavior in terms of the size of evolved GP solutions for any value from 0.1 to 1.0, as shown in each row of Table XII.

The results shown in Table XIII indicate that the required training time for TLCP-criptor is usually independent of any value of μ_I and μ_M. The training time required by TLGP-criptor to learn BrR, mostly, lies between 4.50 to 5.00 minutes, which is slightly less than the time required by GP-criptor, i.e., 5.02 ± 0.89 minutes.

From Table XI and Table XII, it seems that the value of μ_I in the range of 0.5 to 0.7 is an acceptable tradeoff between the classification accuracy and the size of the evolved program. Further, μ_M can be set equal to μ_I (such as 0.5) so that in TLGP-criptor only a single new parameter (say, μ) is required instead of two.

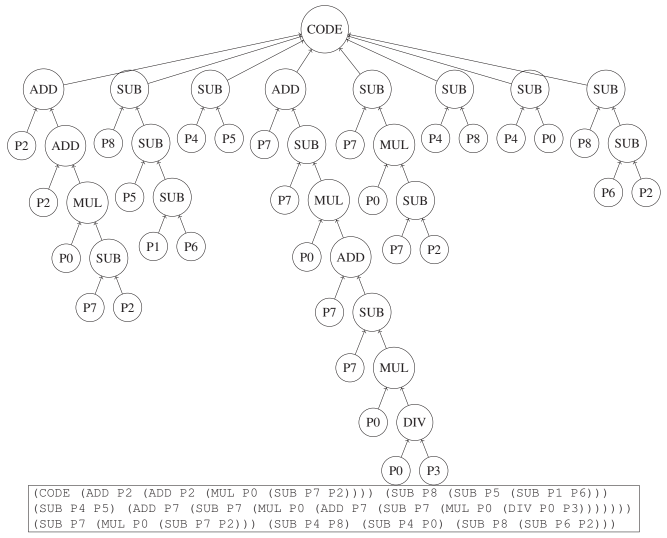
VIII. Analysis of Evolved GP Programs

In this section, we discuss the effectiveness of reusing the extracted knowledge in the proposed approach. To this end, we first analyze an exemplary GP program evolved using the baseline GP-criptor method, and then another GP program evolved using the proposed TLGP-criptor method.

A. An Exemplary Program Evolved using GP-criptor

An exemplary GP program evolved on the KBO data set using the baseline GP-criptor method is shown in Fig. 9. This program has achieved 87.44% accuracy on the test set. It is observed that the third, sixth, and seventh bits of the code are generated by applying the subtraction operator on values of two pixels; whereas to generate the other bits more operators are required along with the subtraction operator. This finding about more frequent appearance of the subtraction operator in an evolved GP program is consistent with observations reported in the baseline paper that this operator has the potential to flip the code value from positive to negative and vice versa [8].

A closer inspection of children of the CODE node in Fig. 9 reveals that the baseline GP-criptor method *implicitly* reused different code fragments to evolve a good GP individual. For example, the third, sixth, and seventh children, i.e., `(SUB P4 P5)`, `(SUB P4 P8)`, and `(SUB P4 P0)`, have the same operator (i.e. SUB) and the same first operand (i.e. P4); they only differ in the second operand. Similarly, the first



```
(CODE (ADD P2 (ADD P2 (MUL P0 (SUB P7 P2)))) (SUB P8 (SUB P5 (SUB P1 P6)))
(SUB P4 P5) (ADD P7 (SUB P7 (MUL P0 (ADD P7 (SUB P7 (MUL P0 (DIV P0 P3)))))))
(SUB P7 (MUL P0 (SUB P7 P2))) (SUB P4 P8) (SUB P4 P0) (SUB P8 (SUB P6 P2)))
```

Fig. 9. A program evolved by GP-criptor on the KBO data set (test accuracy is 87.44%).

and fifth children share the code fragment `(MUL P0 (SUB P7 P2))`. The third and eighth children share the code fragments `(SUB P8 (SUB ... ...))`; and the fourth and fifth children share the code fragment `(SUB P7 (MUL P0 ...))`. Further, it is observed that a code fragment is being reused within the same child, e.g., the code fragment `(SUB P7 (MUL P0 ...))` in the fourth child.

This *implicit* reuse of different code fragments can be due to the following two reasons: (1) Such code fragments are occurring frequently in multiple GP individuals and the crossover operator combined them together in a single GP individual like the one shown in Fig. 9, and (2) Such code fragments are being generated, frequently, by the mutation operator in the same GP individual. The possibility of the second reason is less likely to occur than the first reason. In either case, these findings support our intuition that to extract the potentially useful knowledge (in the form of code fragments) in learning simple tasks and reuse the extracted knowledge to learn more complex tasks can (potentially) improve the classification performance.

B. An Exemplary Program Evolved using TLGP-criptor

In TLGP-criptor, we reuse the extracted code fragments (i.e. children of the CODE node) from the evolved GP programs that have fitness value better than or equal to the average fitness of the final population. Some exemplary code fragments extracted in learning the KBO data set by GP-criptor are shown in Fig. 10. Fig. 11 shows a program evolved by TLGP-criptor on the webcam data set by reusing the extracted code fragments from the KBO data set. This program has achieved 45.93% accuracy on the test set. The frequent appearance of the subtraction operator `SUB` is observed again in the extracted code fragments shown in Fig. 10 as well as in the evolved program shown in Fig. 11.

A closer inspection of the evolved program using TLGP-criptor reveals two important aspects of the reuse of the extracted knowledge: (1) The extracted code fragments are

TABLE XI
THE EFFECT ON CLASSIFICATION ACCURACY FOR USING DIFFERENT VALUES OF μ_I AND μ_M IN TLGP-CRIPTOR TO LEARN BRR. THE CLASSIFICATION ACCURACY OBTAINED USING GP-CRIPTOR TO LEARN BRR IS 55.27 ± 2.15.

μ_I	μ_M									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	58.01 ± 1.84	58.31 ± 1.80	58.77 ± 1.70	58.99 ± 1.41	59.27 ± 1.28	59.71 ± 1.71	59.66 ± 1.64	59.83 ± 1.40	59.60 ± 1.84	59.82 ± 1.45
0.2	59.24 ± 1.62	60.38 ± 1.53	59.59 ± 1.35	60.17 ± 1.35	59.98 ± 1.18	60.34 ± 1.30	60.72 ± 1.30	61.04 ± 1.28	60.55 ± 1.33	61.01 ± 1.36
0.3	59.54 ± 1.49	60.18 ± 1.57	59.90 ± 1.77	60.21 ± 1.50	60.80 ± 1.53	60.67 ± 1.60	60.76 ± 1.63	61.06 ± 1.28	61.11 ± 1.42	61.60 ± 1.61
0.4	60.66 ± 1.54	60.83 ± 1.35	61.26 ± 1.45	61.69 ± 1.10	61.42 ± 1.59	61.57 ± 1.53	61.40 ± 1.24	61.43 ± 1.39	61.50 ± 1.37	61.60 ± 1.23
0.5	60.98 ± 1.31	61.19 ± 1.32	61.56 ± 1.24	61.54 ± 1.66	61.52 ± 1.32	62.26 ± 1.13	61.59 ± 1.23	61.34 ± 1.31	61.68 ± 1.16	61.69 ± 1.33
0.6	61.22 ± 1.08	61.45 ± 1.34	61.45 ± 1.07	61.58 ± 1.33	61.78 ± 1.18	61.58 ± 0.88	61.80 ± 1.05	61.93 ± 1.10	61.86 ± 1.23	61.83 ± 1.02
0.7	61.36 ± 1.26	61.26 ± 1.25	61.84 ± 1.08	62.17 ± 1.14	61.77 ± 1.12	62.09 ± 1.54	62.01 ± 1.18	61.80 ± 1.07	61.87 ± 1.21	62.55 ± 1.11
0.8	61.89 ± 1.39	61.67 ± 1.24	62.28 ± 1.28	62.12 ± 1.48	62.06 ± 1.05	62.21 ± 1.31	62.42 ± 1.42	62.15 ± 1.32	62.20 ± 1.37	62.65 ± 1.42
0.9	62.15 ± 1.22	62.11 ± 1.18	62.19 ± 1.32	61.93 ± 1.50	61.97 ± 1.21	62.53 ± 1.53	62.62 ± 1.43	62.33 ± 1.32	62.60 ± 1.57	62.65 ± 1.38
1.0	62.20 ± 1.24	62.56 ± 0.94	62.50 ± 1.19	62.59 ± 1.11	62.61 ± 1.28	62.69 ± 1.36	62.86 ± 0.93	62.56 ± 1.06	62.94 ± 1.31	62.97 ± 1.60

TABLE XII
THE EFFECT ON SIZE OF THE EVOLVED GP SOLUTION FOR USING DIFFERENT AMOUNT OF EXTRACTED KNOWLEDGE IN TLGP-CRIPTOR TO LEARN BRR. THE SIZE OF THE EVOLVED GP SOLUTION OBTAINED USING GP-CRIPTOR TO LEARN BRR IS 61.37 ± 4.66.

μ_I	μ_M									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	64.69 ± 2.96	64.27 ± 3.10	65.05 ± 3.35	65.16 ± 2.99	65.57 ± 3.55	65.81 ± 3.13	66.73 ± 3.06	66.85 ± 3.00	65.77 ± 2.91	65.40 ± 3.48
0.2	65.51 ± 3.83	66.27 ± 2.78	66.59 ± 3.54	66.19 ± 3.36	65.87 ± 3.65	66.55 ± 3.48	66.05 ± 3.42	67.11 ± 3.35	67.30 ± 3.05	67.77 ± 3.26
0.3	67.62 ± 3.60	66.30 ± 3.41	66.78 ± 3.40	67.11 ± 2.81	67.33 ± 3.37	66.49 ± 4.15	67.11 ± 3.19	66.92 ± 2.79	67.19 ± 3.17	67.32 ± 2.82
0.4	66.86 ± 3.38	68.53 ± 3.27	67.11 ± 2.96	67.57 ± 3.30	67.83 ± 3.39	67.39 ± 2.94	67.03 ± 3.99	67.33 ± 3.01	67.98 ± 3.06	68.55 ± 2.58
0.5	67.95 ± 2.40	66.66 ± 3.17	68.13 ± 2.89	67.77 ± 3.29	67.96 ± 3.97	68.31 ± 3.77	67.76 ± 2.79	67.39 ± 3.10	67.95 ± 3.25	67.39 ± 3.07
0.6	68.17 ± 3.28	67.73 ± 3.49	67.79 ± 3.68	67.87 ± 4.16	68.42 ± 3.68	67.16 ± 3.14	68.70 ± 3.21	67.01 ± 2.99	68.91 ± 2.91	68.47 ± 3.53
0.7	68.93 ± 3.19	68.41 ± 2.60	68.30 ± 3.29	68.40 ± 3.11	68.10 ± 2.43	68.94 ± 3.42	70.64 ± 3.07	68.37 ± 2.68	68.37 ± 3.30	69.29 ± 4.30
0.8	68.73 ± 3.72	68.04 ± 2.85	68.33 ± 2.46	69.63 ± 3.14	68.56 ± 3.30	68.87 ± 2.64	69.63 ± 3.84	69.23 ± 4.01	68.50 ± 3.64	68.76 ± 3.23
0.9	69.36 ± 3.48	68.85 ± 3.72	69.01 ± 3.56	69.32 ± 2.74	68.82 ± 3.23	69.18 ± 2.94	69.05 ± 3.33	69.15 ± 2.77	69.25 ± 3.13	69.49 ± 3.45
1.0	67.77 ± 3.20	69.01 ± 3.20	69.11 ± 3.91	69.84 ± 3.52	69.71 ± 2.85	69.65 ± 3.00	69.30 ± 3.13	68.49 ± 2.86	69.65 ± 2.88	69.37 ± 3.56

TABLE XIII
THE EFFECT ON TRAINING TIME TO LEARN BRR USING DIFFERENT AMOUNT OF EXTRACTED KNOWLEDGE IN TLGP-CRIPTOR. THE TRAINING TIME REQUIRED BY GP-CRIPTOR TO LEARN BRR IS 5.02 ± 0.89 MINUTES.

μ_I	μ_M									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	6.12 ± 0.60	4.54 ± 0.52	4.42 ± 0.39	4.29 ± 0.08	4.61 ± 0.70	5.58 ± 0.88	4.55 ± 0.55	4.28 ± 0.08	4.58 ± 0.51	4.95 ± 0.67
0.2	4.47 ± 0.36	4.66 ± 0.61	4.33 ± 0.06	4.47 ± 0.40	4.29 ± 0.08	4.74 ± 0.65	4.52 ± 0.43	4.47 ± 0.40	4.82 ± 0.70	4.36 ± 0.08
0.3	4.35 ± 0.09	4.37 ± 0.13	4.46 ± 0.15	4.52 ± 0.54	4.37 ± 0.07	4.66 ± 0.63	4.83 ± 0.69	4.70 ± 0.54	4.86 ± 0.74	4.54 ± 0.23
0.4	4.54 ± 0.42	4.79 ± 0.75	4.73 ± 0.60	4.48 ± 0.24	4.50 ± 0.43	4.34 ± 0.06	4.72 ± 0.71	4.59 ± 0.45	4.40 ± 0.06	4.57 ± 0.59
0.5	4.38 ± 0.06	4.44 ± 0.22	4.37 ± 0.07	4.37 ± 0.07	4.64 ± 0.60	4.38 ± 0.08	4.77 ± 0.63	4.36 ± 0.07	4.50 ± 0.40	4.40 ± 0.04
0.6	4.75 ± 0.63	4.73 ± 0.59	4.73 ± 0.61	4.68 ± 0.44	4.76 ± 0.64	4.82 ± 0.79	4.55 ± 0.15	4.52 ± 0.15	4.83 ± 0.60	4.86 ± 0.79
0.7	4.65 ± 0.28	4.45 ± 0.12	4.62 ± 0.53	4.50 ± 0.25	4.57 ± 0.43	4.57 ± 0.39	4.59 ± 0.48	4.75 ± 0.68	4.89 ± 0.75	4.74 ± 0.58
0.8	5.13 ± 0.95	4.44 ± 0.06	4.44 ± 0.09	4.59 ± 0.45	4.88 ± 0.71	4.72 ± 0.59	4.45 ± 0.06	4.65 ± 0.58	4.47 ± 0.08	4.47 ± 0.08
0.9	4.61 ± 0.47	4.65 ± 0.46	4.60 ± 0.57	4.80 ± 0.74	4.42 ± 0.08	4.67 ± 0.55	4.77 ± 0.60	4.48 ± 0.09	4.65 ± 0.50	4.75 ± 0.58
1.0	5.03 ± 0.69	4.98 ± 0.72	4.88 ± 0.76	4.68 ± 0.67	4.80 ± 0.68	4.60 ± 0.44	4.63 ± 0.51	4.64 ± 0.46	4.46 ± 0.06	5.18 ± 0.78

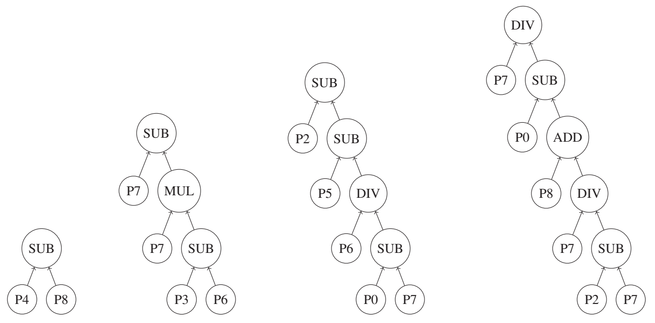


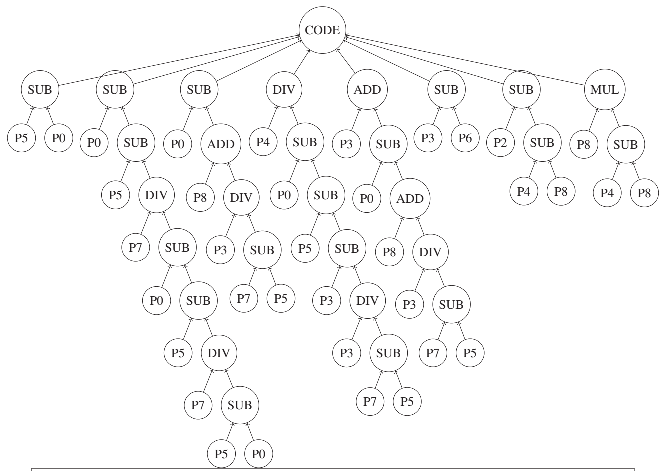
Fig. 10. Exemplary code fragments extracted in learning the KBO data set by GP-criptor.

being reused in TLGP-criptor as a whole and in parts as well to generate new code fragments. For example, the first extracted code fragment `(SUB P4 P8)` is used as a whole to generate the last two children of the CODE node in the evolved program shown in Fig. 11. Similarly, the sixth child of this program is generated using a part of the second extracted code fragment `(SUB P7 (MUL P7 (SUB P3 P6)))`, and the second, third, and fifth children are reusing parts of the fourth extracted code fragment. (2) The structure of the extracted code fragments, i.e., the order in which different operators have been used in an extracted code fragment, is observed in the newly generated code fragments in the evolved program. For example, the second and fourth children of the CODE node in Fig. 11 are reusing the structure of the third extracted code fragment, i.e., the sequence of operators `(SUB SUB DIV SUB)`. Further, nested reuse of structure `SUB DIV SUB` is observed in the second child of the CODE node.

In summary, the idea of reusing the learned code fragments in TLGP-criptor was to avoid learning the potentially good code fragments from scratch. The reuse of extracted knowledge significantly improved the classification performance of TLGP-criptor over the baseline GP-criptor method.

IX. CONCLUSIONS

The main goal of this work was to propose and implement a novel algorithm for transfer learning in GP to learn complex



```
(CODE (SUB P5 P0) (SUB P0 (SUB P5 (DIV P7 (SUB P0 (SUB P5 (DIV P7 (SUB P5
P0))))))) (SUB P0 (ADD P8 (DIV P3 (SUB P7 P5)))) (DIV P4 (SUB P0 (SUB P5 (SUB
P3 (DIV P3 (SUB P7 P5)))))) (ADD P3 (SUB P0 (ADD P8 (DIV P3 (SUB P7 P5)))))
(SUB P3 P6) (SUB P2 (SUB P4 P8)) (MUL P8 (SUB P4 P8)))
```

Fig. 11. A program evolved by TLGP-criptor on the webcam data set by reusing the extracted knowledge from the KBO data set (test accuracy is 45.93%).

image classification tasks. This goal has been successfully achieved by incorporating the idea of transfer learning in an existing state-of-the-art GP method, which works very well on image classification tasks without noise and rotations. The proposed approach was designed to use the idea of transfer learning to extract blocks of knowledge in the form of code fragments when learning simple problems, and then reuse them to learn more complex tasks, i.e., various rotated and noisy versions of the original images. Further, the extracted knowledge was also reused to learn various complex images in other related as well as different image domains. The experiments on six image classification benchmarks of varying difficulty showed that the proposed approach of transfer learning in GP has significantly improved the classification performance over the baseline method. Further analysis on the evolved GP trees/solutions showed that the extracted code fragments from simpler problems are general and useful for generating a good initial population to provide a better starting point for GP. The extracted code fragments are also helpful for constructing better GP trees during the evolutionary process, which is the main reason why the proposed approach can improve the performance of GP on complex image classification tasks.

This work confirms the ability of GP in discovering and extracting useful knowledge from a simple task to help learn a complex task. Deep analysis in this work shows that the extracted knowledge is useful either as code fragments or as a structure of code fragments when solving complex image classification problems. The obtained results indicate that the proposed method has the ability to transfer the extracted knowledge to a similar domain as well as to different domains. In all experiments conducted in this study, no negative effect has been observed, which (i.e. negative effect) is usually expected in cross-domain transfer. It is anticipated that the inherent adaptiveness of GP, like other evolutionary approaches, has automatically eliminated negatively transferred knowledge via evolutionary principles. In the future, more rigorous methods will be considered that explicitly account for transfer characteristics.

A. Future Work

This is an early, but promising, approach to incorporating transfer learning in GP for complex image classification, which opens a door for this field and many future directions can be explored. For example, the current baseline method is using raw image pixels as input while future transfer learning based approaches can utilise the ability of GP in feature extraction/construction/selection to further improve the performance.

It is also worth extending the idea of extracting and reusing blocks of knowledge in traditional GP so that it can be applied to a wide range of tasks. One such possible extension is to reuse the extracted knowledge to populate a node in a GP tree. Traditionally, a node in a GP tree is populated by creating each child of the node from a symbol in the given set of terminals T or the set of functions F. To reuse the extracted knowledge, a child should be created either by selecting an extracted code fragment (or a randomly selected subtree from that code fragment) with probability μ_P, or by selecting a symbol from T and F (as used in standard GP) with probability $1 - \mu_P$.

Similarly, the extracted knowledge can be used to mutate a GP tree in the traditional GP. In standard GP, the mutation operation randomly selects a subtree in a GP program and replaces it with a randomly generated new subtree. To reuse the extracted knowledge, a GP program can be mutated either by replacing the selected subtree with an extracted code fragment (or a randomly selected subtree from that code fragment) with probability μ_M, or by replacing with a randomly generated new subtree (as used in standard GP) with probability $1-\mu_M$.

Acknowledgment

The authors would like to thank Muhammad Ghifary for his help to collect results using DeCAF on texture image data sets.

This work was supported in part by the Marsden Fund of New Zealand Government under Contract VUW1209, VUW1509 and 16-VUW-111, Huawei Industry Fund E2880/3663, and the University Research Fund at Victoria University of Wellington 209861/3580, 209862/3580, and 213150/3662.

References

[1] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[2] P. Stone and M. Veloso, "Layered Learning," in *Proceedings of the European Conference on Machine Learning*, vol. 1810. Springer, 2000, pp. 369–381.

[3] A. H. Gandomi, A. H. Alavi, and C. Ryan, *Handbook of Genetic Programming Applications*. Springer, 2015.

[4] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. The MIT Press, 1992.

[5] D. Jackson and A. P. Gibbons, "Layered learning in boolean gp problems," in *Proceedings of the European Conference on Genetic Programming*, ser. Lecture Notes in Computer Science, vol. 4445. Springer, 2007, pp. 148–159.

[6] H. Al-Sahaf, A. Song, K. Neshatian, and M. Zhang, "Two-tier genetic programming: Towards raw pixel-based image classification," *Expert Systems with Applications*, vol. 39, no. 16, pp. 12 291–12 301, 2012.

[7] Y. Li, J. Ma, and Q. Zhao, "Two improvements in genetic programming for image classification," in *IEEE Congress on Evolutionary Computation*, 2008, pp. 2492–2497.

[8] H. Al-Sahaf, M. Zhang, M. Johnston, and B. Verma, "Image descriptor: A genetic programming approach to multiclass texture classification," in *IEEE Congress on Evolutionary Computation*. IEEE, 2015, pp. 2460–2467.

[9] A. Lensen, H. Al-Sahaf, M. Zhang, and B. Xue, "A Hybrid Genetic Programming Approach to Feature Detection and Image Classification," in *Proceedings of the 30th International Conference on Image and Vision Computing New Zealand*. IEEE, 2015, pp. 1–6.

[10] K. Krawiec and B. Wieloch, "Automatic Generation and Exploitation of Related Problems in Genetic Programming," in *IEEE Congress on Evolutionary Computation*. IEEE, 2010, pp. 1–8.

[11] N. T. Hien, N. X. Hoai, and B. McKay, "A Study on Genetic Programming with Layered Learning and Incremental Sampling," in *IEEE Congress on Evolutionary Computation*, 2011, pp. 1179–1185.

[12] T.-H. Hoang, R. I. B. McKay, D. Essam, and N. X. Hoai, "On Synergistic Interactions Between Evolution, Development and Layered Learning," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 3, pp. 287–312, 2011.

[13] Q. Chen, B. Xue, and M. Zhang, "Generalisation and Domain Adaptation in GP with Gradient Descent for Symbolic Regression," in *IEEE Congress on Evolutionary Computation*, 2015, pp. 1137–1144.

[14] W. Jaśkowski, K. Krawiec, and B. Wieloch, "Cross-task code reuse in genetic programming applied to visual learning," *International Journal of Applied Mathematics and Computer Science*, vol. 24, no. 1, pp. 183–197, 2014.

[15] T. T. H. Dinh, T. H. Chu, and Q. U. Nguyen, "Transfer Learning in Genetic Programming," in *IEEE Congress on Evolutionary Computation*, 2015, pp. 1145–1151.

[16] M. Iqbal, B. Xue, and M. Zhang, "Reusing Extracted Knowledge in Genetic Programming to Solve Complex Texture Image Classification Problems," in *Proceedings of the Pacific Asia Knowledge Discovery and Data Mining Conference, Part II*, 2016, pp. 117–129.

[17] M. Hauschild, M. Pelikan, K. Sastry, and C. Lima, "Analyzing Probabilistic Models in Hierarchical BOA," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 6, pp. 1199–1217, 2009.

[18] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain Adaptation via Transfer Component Analysis," *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–210, 2011.

[19] J. Lu, V. Behbood, P. Hao, H. Zuo, S. Xue, and G. Zhang, "Transfer Learning using Computational Intelligence: A survey," *Knowledge-Based Systems*, vol. 80, no. C, pp. 14–23, 2015.

[20] R. Santana, R. Armañanzas, C. Bielza, and P. Larrañaga, "Network Measures for Information Extraction in Evolutionary Algorithms," *International Journal of Computational Intelligence Systems*, vol. 6, no. 6, pp. 1163–1188, 2013.

[21] B. Kocer and A. Arslan, "Genetic Transfer Learning," *Expert Systems with Applications*, vol. 37, no. 10, pp. 6997–7002, 2010.

[22] A. Moshaiov and A. Tal, "Family Bootstrapping: a Genetic Transfer Learning Approach for Onsetting the Evolution for a Set of Related Robotic Tasks," in *IEEE Congress on Evolutionary Computation*. IEEE, 2014, pp. 2801–2808.

[23] S. Yang and X. Yao, "Population-Based Incremental Learning with Associative Memory for Dynamic Environments," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 5, pp. 542–561, 2008.

[24] M. Mavrovouniotis and S. Yang, *Direct Memory Schemes for Population-Based Incremental Learning in Cyclically Changing Environments*. Springer International Publishing, 2016, pp. 233–247.

[25] L. Feng, Y.-S. Ong, A.-H. Tan, and I. W. Tsang, "Memes as Building Blocks: A Case Study on Evolutionary Optimization + Transfer Learning for Routing Problems," *Memetic Computing*, vol. 7, no. 3, pp. 159–180, 2015.

[26] R. Meuth, M.-H. Lim, Y.-S. Ong, and D. C. W. II, "A Proposition on Memes and Meta-Memes in Computing for Higher-Order Learning," *Memetic Computing*, vol. 1, no. 2, pp. 85–100, 2009.

[27] A. Gupta, Y.-S. Ong, and L. Feng, "Multifactorial Evolution: Towards Evolutionary Multitasking," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 3, pp. 343–357, 2016.

[28] Y.-S. Ong and A. Gupta, "Evolutionary Multitasking: A Computer Science View of Cognitive Multitasking," *Cognitive Computation*, vol. 8, no. 2, pp. 125–142, 2016.

[29] D. Lim, Y. S. Ong, A. Gupta, C. K. Goh, and P. S. Dutta, "Towards a new Praxis in optinformatics targeting knowledge re-use in evolutionary computation: simultaneous problem learning and optimization," *Evolutionary Intelligence*, 2016, doi:10.1007/s12065-016-0146-1.

[30] S. M. Gustafson and W. H. Hsu, "Layered Learning in Genetic Programming for a Cooperative Robot Soccer Problem," in *Proceedings of the European Conference on Genetic Programming*. Springer, 2001, pp. 291–301.

[31] N. T. Hien and N. X. Hoai, "Learning in stages: A layered learning approach for genetic programming," in *Proceedings of the International Conference on Computing and Communication Technologies, Research, Innovation, and Vision for the Future*, 2012, pp. 1–4.

[32] N. X. Hoai, R. I. B. McKay, and D. Essam, "Representation and Structural Difficulty in Genetic Programming," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 2, pp. 157–166, 2006.

[33] M. Iqbal, W. N. Browne, and M. Zhang, "Reusing Building Blocks of Extracted Knowledge to Solve Complex, Large-Scale Boolean Problems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 465–480, 2014.

[34] I. M. Alvarez, W. N. Browne, and M. Zhang, "Human-inspired Scaling in Learning Classifier Systems: Case Study on the N-bit Multiplexer Problem Set," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2016, pp. 429–436.

[35] P. Espejo, S. Ventura, and F. Herrera, "A survey on the application of genetic programming to classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 40, no. 2, pp. 121–144, 2010.

[36] C. B. Perez and G. Olague, "Evolutionary learning of local descriptor operators for object recognition," in *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*. ACM, 2009, pp. 1051–1058.

[37] K. Neshatian, M. Zhang, and P. Andreae, "A filter approach to multiple feature construction for symbolic learning classifiers using genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 5, pp. 645–661, 2012.

[38] U. Bhowan, M. Johnston, M. Zhang, and X. Yao, "Evolving diverse ensembles using genetic programming for classification with unbalanced data," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 3, pp. 368–386, 2013.

[39] H. Al-Sahaf, M. Zhang, and M. Johnston, "Binary image classification: A genetic programming approach to the problem of limited training instances," *Evolutionary Computation*, pp. 1–40, 2015, DOI:10.1162/EVCO_a_00146.

[40] D. L. Atkins, K. Neshatian, and M. Zhang, "A domain independent genetic programming approach to automatic feature extraction for image classification," in *IEEE Congress on Evolutionary Computation*. IEEE, 2011, pp. 238–245.

[41] D. Agnelli, A. Bollini, and L. Lombardi, "Image classification: an evolutionary approach," *Pattern Recognition Letters*, vol. 23, no. 1, pp. 303 – 309, 2002.

[42] W. Albukhanajer, J. Briffa, and Y. Jin, "Evolutionary multiobjective image feature extraction in the presence of noise," *IEEE Transactions on Cybernetics*, vol. 45, no. 9, pp. 1757–1768, 2015.

[43] K. Krawiec and B. Bhanu, "Visual learning by evolutionary and coevolutionary feature synthesis," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 5, pp. 635–650, 2007.

[44] G. Abo Smara and F. Khalefah, "Localization of license plate number using dynamic image processing techniques and genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 2, pp. 244–257, 2014.

[45] J. Santamaría, S. Damas, O. Cordón, and A. Escámez, "Self-adaptive evolution toward new parameter free image registration methods," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 4, pp. 545–557, 2013.

[46] R. Poli, "Genetic Programming for Image Analysis," in *Proceedings of the 1st Annual Conference on Genetic Programming*, 1996, pp. 363–368.

[47] W. R. Smart and M. Zhang, "Classification strategies for image classification in genetic programming," in *Proceedings of the 18th International Conference on Image and Vision Computing New Zealand*. Massey University, 2003, pp. 402–407.

[48] C. Ryan, J. Fitzgerald, K. Krawiec, and D. Medernach, "Image classification with genetic programming: Building a stage 1 computer aided detector for breast cancer," in *Handbook of Genetic Programming Applications*. Springer, 2015, pp. 245–287.

[49] S. Hindmarsh, P. Andreae, and M. Zhang, "Genetic programming for improving image descriptors generated using the scale-invariant feature transform," in *Proceedings of the 27th International Conference on Image and Vision Computing New Zealand*. ACM, 2012, pp. 85–90.

[50] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[51] L. Shao, L. Liu, and X. Li, "Feature learning for image classification via multiobjective genetic programming," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 7, pp. 1359–1371, 2014.

[52] B. Yang and S. Chen, "A comparative study on local binary pattern (LBP) based face recognition: LBP histogram versus LBP image," *Neurocomputing*, vol. 120, pp. 365–379, 2013.

[53] M. Zhang, V. Ciesielski, and P. Andreae, "A Domain-Independent Window Approach to Multiclass Object Detection Using Genetic Programming," *EURASIP Journal on Advances in Signal Processing*, vol. 2003, no. 8, pp. 841–859, 2003.

[54] A. Lensen, H. Al-Sahaf, M. Zhang, and B. Xue, "Genetic Programming for Region Detection, Feature Extraction, Feature Construction and Classification in Image Data," in *Proceedings of the 19th European Conference on Genetic Programming*, ser. Lecture Notes in Computer Science, vol. 9594. Springer, 2016, pp. 49–64.

[55] A. Quattoni, M. Collins, and T. Darrell, "Transfer Learning for Image Classification with Sparse Prototype Representations," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[56] Y. Zhu, Y. Chen, Z. Lu, S. J. Pan, G.-R. Xue, Y. Yu, and Q. Yang, "Heterogeneous Transfer Learning for Image Classification," in *AAAI Conference on Artificial Intelligence*, 2011, pp. 1304–1309.

[57] G. Griffin, A. Holub, and P. Perona, "Caltech-256 Object Category Dataset," California Institute of Technology, Tech. Rep. 7694, 2007. [Online]. Available: http://authors.library.caltech.edu/7694

[58] D. C. Cireşan, U. Meier, and J. Schmidhuber, "Transfer Learning for Latin and Chinese Characters with Deep Neural Networks," in *Proceedings of the International Joint Conference on Neural Networks*, 2012, pp. 1–6.

[59] C. Kandaswamy, L. M. Silva, L. A. Alexandre, J. M. Santos, e. S. de Sá, Joaquim Marques", C. Weber, W. Duch, T. Honkela, P. Koprinkova-Hristova, S. Magg, G. Palm, and A. E. P. Villa, "Improving Deep Neural Network Performance by Reusing Features Trained with Transductive Transference," in *International Conference on Artificial Neural Networks*. Springer International Publishing, 2014, pp. 265–272.

[60] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition," in *International Conference in Machine Learning*, 2014.

[61] M. Ghifary, W. B. Kleijn, and M. Zhang, "Domain Adaptive Neural Networks for Object Recognition," in *Trends in Artificial Intelligence - 13th Pacific Rim International Conference on Artificial Intelligence*, 2014, pp. 898–904.

[62] K. M. Borgwardt, A. Gretton, M. J. Rasch, H.-P. Kriegel, B. Schölkopf, and A. J. Smola, "Integrating Structured Biological Data by Kernel Maximum Mean Discrepancy," *Bioinformatics*, vol. 22, no. 14, pp. e49–e57, 2006.

[63] M. Ghifary, W. B. Kleijn, M. Zhang, and D. Balduzzi, "Domain Generalization for Object Recognition with Multi-task Autoencoders," in *International Conference on Computer Vision*. IEEE, 2015, pp. 2551–2559.

[64] W. Jaśkowski, K. Krawiec, and B. Wieloch, "Multitask visual learning using genetic programming," *Evolutionary computation*, vol. 16, no. 4, pp. 439–459, 2008.

[65] M. Iqbal, M. Zhang, and B. Xue, "Improving Classification on Images by Extracting and Transferring Knowledge in Genetic Programming," in *IEEE Congress on Evolutionary Computation*, 2016, pp. 3582–3589.

[66] H. Al-Sahaf, A. Song, and M. Zhang, "Hybridisation of Genetic Programming and Nearest Neighbour for Classification," in *IEEE Congress on Evolutionary Computation*. IEEE, 2013, pp. 2650–2657.

[67] D. Androutsos, K. Plataniotis, and A. Venetsanopoulos, "Distance Measures for Color Image Retrieval," in *Proceedings of the International Conference on Image Processing*, vol. 2. IEEE, 1998, pp. 770–774.

[68] S.-H. Cha, "Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions," *International Journal of Mathematical Models and Methods in Applied Sciences*, vol. 1, no. 4, pp. 300–307, 2007.

[69] A. van Opbroek, M. A. Ikram, M. W. Vernooij, and M. de Bruijne, "Transfer Learning Improves Supervised Image Segmentation Across Imaging Protocols," *IEEE Transactions on Medical Imaging*, vol. 34, no. 5, pp. 1018–1030, 2015.

[70] V. M. Patel, R. Gopalan, R. Li, and R. Chellapa, "Visual Domain Adaptation: A survey of Recent Advances," *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 53–69, 2015.

[71] M. Ghifary, W. B. Kleijn, M. Zhang, D. Balduzzi, and W. Li, "Deep Reconstruction-Classification Networks for Unsupervised Domain Adaptation," in *Proceedings of the 14th European Conference on Computer Vision*, 2016, pp. 597–613.

[72] M. Ghifary, D. Balduzzi, W. B. Kleijn, and M. Zhang, "Scatter Component Analysis: A Unified Framework for Domain Adaptation and Domain Generalization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016, doi:10.1109/TPAMI.2016.2599532.

[73] G. Kylberg, "The Kylberg texture dataset v. 1.0," Centre for Image Analysis, Swedish University of Agricultural Sciences and Uppsala University, Uppsala, Sweden, External report (Blue series) 35, 2011.

[74] P. Brodatz, *Textures: A Photographic Album for Artists and Designers*. Dover Publications, 1999.

[75] T. Ojala, T. Mäenpää, M. Pietikäinen, J. Viertola, J. Kyllonen, and S. Huovinen, "Outex - New Framework for Empirical Evaluation of Texture Analysis Algorithms," in *Proceedings of the 16th International Conference on Pattern Recognition*. IEEE, 2002, pp. 701–706.

[76] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting Visual Category Models to New Domains," in *Proceedings of 11th European Conference on Computer Vision*. Springer, 2010, pp. 213–226.

[77] S. Luke, *Essentials of Metaheuristics*, 2nd ed. Lulu, 2013, available for free at http://cs.gmu.edu/∼sean/book/metaheuristics/.

[78] Y. Zhao, D.-S. Huang, and W. Jia, "Completed Local Binary Count for Rotation Invariant Texture Classification," *IEEE Transactions on Image Processing*, vol. 21, no. 10, pp. 4492–4497, 2012.

[79] R. Mehta and K. Egiazarian, "Dominant Rotated Local Binary Patterns (DRLBP) for texture classification," *Pattern Recognition Letters*, vol. 71, no. 1, pp. 16–22, 2016.

[80] H. Al-Sahaf, A. Al-Sahaf, B. Xue, M. Johnston, and M. Zhang, "Automatically Evolving Rotation-invariant Texture Image Descriptors by Genetic Programming," *IEEE Transactions on Evolutionary Computation*, 2016, DOI:10.1109/TEVC.2016.2577548.

Muhammad Iqbal (M'11) received a Master in Computer Science degree from the National University of Sciences and Technology, Pakistan, in 2008, and a PhD in Computer Science from Victoria University of Wellington, New Zealand, in 2014.

Iqbal's main research interests are in the area of evolutionary machine learning. His research focuses on evolutionary image analysis and classification using transfer learning in genetic programming and learning classifier systems techniques. He is also interested in medical image analysis, data mining, and scalability of evolutionary techniques.

Mr Iqbal is a member of IEEE, the IEEE Computational Intelligence Society, and the ACM SIGEVO Group. He has published a couple dozen academic papers in refereed international journals and conferences.

Bing Xue (M'11) received her PhD degree in 2014 at Victoria University of Wellington, New Zealand. She is now a Senior Lecturer at Victoria University of Wellington (VUW). She is with the Evolutionary Computation Research Group at VUW, and her research focuses mainly on evolutionary computation, machine learning and data mining, particularly, evolutionary computation for feature selection, feature construction, dimension reduction, symbolic regression, transfer learning, image analysis, multi-objective optimisation, bioinformatics and big data. Bing is currently leading the strategic research direction on evolutionary feature selection and construction in Evolutionary Computation Research Group at VUW, and has been organising special sessions and special issues on evolutionary computation for feature selection and construction. She is the funding and current Chair of IEEE CIS Task Force on Evolutionary Computation for Feature Selection and Construction. Bing is a committee member of Evolutionary Computation Technical Committee, and Emergent Technologies Technical Committee, IEEE CIS. She has been serving as a guest editor, associated editor or editorial board member for international journals, and program chair, special session chair, symposium/special session organiser for a number of international conferences, and as reviewer for top international journals and conferences in the field.

Harith Al-Sahaf (M'16) received his B.Sc. degree in Computer Science from Baghdad University, Baghdad, Iraq, in 2005. He received his Master of Computer Science (MCompSc) degree in 2010 from Victoria University of Wellington, Wellington, New Zealand (VUW). Since 2010, he has joined the Evolutionary Computation Research Group (ECRG) at VUW. Currently, he is a Ph.D. Candidate in the School of Engineering and Computer Science at Victoria University of Wellington.

Harith's research interests are in evolutionary computation, computer vision, pattern recognition, machine learning, feature manipulation including feature detection, selection, extraction and construction, transfer learning, domain adaptation, one-shot learning, and image understanding.

Mr. Al-Sahaf is a member of the IEEE Computational Intelligence Society (CIS). He has been serving as a reviewer for top international journals and conferences in the field.

Mengjie Zhang (M'04–SM'10) received the B.E. and M.E. degrees from Artificial Intelligence Research Center, Agricultural University of Hebei, Hebei, China, and the Ph.D. degree in computer science from RMIT University, Melbourne, VIC, Australia, in 1989, 1992, and 2000, respectively.

Since 2000, he has been with the Victoria University of Wellington, Wellington, New Zealand, where he is currently Professor of Computer Science, Head of the Evolutionary Computation Research Group, and the Associate Dean (Research and Innovation) in the Faculty of Engineering. His current research interests include evolutionary computation, particularly genetic programming, particle swarm optimization, and learning classifier systems with application areas of image analysis, multiobjective optimization, feature selection and reduction, job shop scheduling, and transfer learning. He has published over 350 research papers in refereed international journals and conferences.

Prof. Zhang has been serving as an Associated Editor or Editorial Board Member for ten international journals (including IEEE Transactions on Evolutionary Computation, IEEE Transactions on Cybernetics, Evolutionary Computation Journal, and IEEE Transactions on Emergent Topics in CI) and as a Reviewer of over 20 international journals. He has been serving as a Steering Committee Member and a Program Committee Member for over 100 international conferences. He has supervised over 50 postgraduate research students. He is the Chair of the IEEE CIS Intelligent Systems and Applications Technical Committee, a member of the IEEE CIS Evolutionary Computation Technical Committee, a Vice-Chair of the IEEE CIS Task Force on Evolutionary Computer Vision and Image Processing, a Vice-Chair of the IEEE CIS Task Force on Evolutionary Computation for Feature Selection and Construction, a member of IEEE CIS Task Force of Hyper-heuristics, and the Founding Chair for IEEE Computational Intelligence Chapter in New Zealand.