

One-Shot Rule Learning for Challenging Character Recognition

Dany Varghese and Alireza Tamaddoni-Nezhad

Department of Computer Science, University of Surrey, Guildford, UK
{dany.varghese, a.tamaddoni-nezhad}@surrey.ac.uk

Abstract. Unlike most of computer vision approaches which depend on hundreds or thousands of training images, humans can typically learn from a single visual example. Humans achieve this ability using background knowledge. Rule-based machine learning approaches such as Inductive Logic Programming (ILP) provide a framework for incorporating domain specific background knowledge. These approaches have the potential for human-like learning from small data or even one-shot learning, i.e. learning from a single positive example. By contrast, statistics based computer vision algorithms, including Deep Learning, have no general mechanisms for incorporating background knowledge. In this paper, we present an approach for one-shot rule learning called One-Shot Hypothesis Derivation (OSHD) which is based on using a logic program declarative bias. We apply this approach to the challenging task of Malayalam character recognition. This is a challenging task due to spherical and complex structure of Malayalam hand-written language. Unlike for other languages, there is currently no efficient algorithm for Malayalam hand-written recognition. We compare our results with a state-of-the-art Deep Learning approach, called Siamese Network, which has been developed for one-shot learning. The results suggest that our approach can generate human-understandable rules and also outperforms the deep learning approach with a significantly higher average predictive accuracy.

Keywords: One-Shot Learning · Rule-Based Machine Learning · Inductive Logic Programming (ILP) · Malayalam Character Recognition · Computer Vision

1 Introduction

Deep Neural Networks (DNNs) [7, 2, 15, 3] have demonstrated state-of-the-art results on many pattern recognition tasks, especially in image classification problems [14, 9, 23, 6]. However, recent studies [24, 26] revealed major differences between human visual cognition and DNNs, and in general most of statistics-based

Copyright © 2020 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

computer vision learning algorithms. For example, it is easy to produce images that are completely unrecognizable to humans, though DNN visual learning algorithms believe them to be recognizable objects with over 99% confidence [24].

Another major difference is related to the number of required training examples. Humans can typically learn from a single visual example [10], unlike statistical learning which depends on hundreds or thousands of images. Humans achieve this ability using background knowledge, which plays a critical role. By contrast, statistics based computer vision algorithms have no general mechanisms for incorporating background knowledge.

Computer vision is a multidisciplinary field that aims to create high-level understanding from digital images or videos. The key intention of image analysis is to bridge the semantic gap between low-level descriptions of an image and the high level concept within the image. The main objective of structural pattern analysis is to present the visual data using natural descriptions. Traditionally, this is achieved by extracting low-level visual cues from the data provided, then applying some grouping algorithm to express relationships that are then transformed into more and more complex and convoluted features that generate higher level rules [16].

Document image analysis approaches such as Optical Character Recognition (OCR) are an important part of visual artificial intelligence with many real-world applications. The main objective of these approaches is to identify significant graphical properties from images. In this context, Symbol Recognition has a long history dating back to the 70's. In the current state-of-the-art, symbol recognition involves identifying isolated symbols, however this is not enough for some more challenging real-world application. As an example, consider an application where the visual data is represented as a combination of isolated symbols as well as composite symbols that are connected with other graphical elements. Then the statistical approaches which represent shapes only as low level features will have limited success.

In this paper, we present an approach for one-shot rule learning called One-Shot Hypothesis Derivation (OSHD) which is based on using a logic program declarative bias. We apply this approach to the challenging task of Malayalam character recognition. This is a challenging task due to spherical and complex structure of Malayalam hand-written language. Unlike for other languages, there is currently no efficient algorithm for Malayalam hand-written recognition. The language scripts are mainly based on circular geometrical properties. We have created a dataset for Malayalam hand-written characters which includes high level properties of the language based on 'Omniglot' dataset designed for developing human-level concept learning algorithms [11]. We compare our results with a state-of-the-art Deep Learning approach, called Siamese Network [8], which has been developed for one-shot learning.

2 Inductive Logic Programming (ILP) and One-Shot Hypothesis Derivation (OSHD)

Inductive Logic Programming (ILP) has been defined as the intersection of inductive learning and logic programming [21]. Thus, ILP employs techniques from both machine learning and logic programming.

The main objective of ILP, in its simplest form, is to discover the definition of a predicate by observing positive and negative examples of that predicate. Together with positive and negative examples of the target predicate, other background information may also be provided containing further information relevant to learning the target predicate. This background information is represented as a logic program and is called background knowledge (BK). ILP systems develop predicate descriptions from examples and background knowledge. The examples, background knowledge and final descriptions are all described as logic programs.

The logical notations and foundations of ILP can be found in [21, 25]. The following definition, adapted from [25], defines the learning problem setting for ILP.

Definition 1 (ILP problem setting).

Input : Given $\langle B, E \rangle$, where B is a set of clauses representing the background knowledge and E is the set of positive (E^+) and negative (E^-) examples such that $B \not\models E^+$.

Output : find a theory \mathcal{H} such that \mathcal{H} is complete and (weakly) consistent with respect to B and E . \mathcal{H} is complete with respect to B and E^+ if $B \wedge \mathcal{H} \models E^+$. \mathcal{H} is consistent with respect to B and E^- if $B \wedge \mathcal{H} \wedge E^- \not\models \square$. \mathcal{H} is weakly consistent with respect to B if $B \wedge \mathcal{H} \not\models \square$.

In this definition, \models represents logical entailment and \square represents an empty clause or logical refutation. Note that in practice, due to the noise in the training examples, the completeness and consistency conditions are usually relaxed. For example, weak consistency is usually used and a noise threshold is considered which allows \mathcal{H} to be inconsistent with respect to a certain proportion (or number) of negative examples.

The following example is adapted from [13].

Example 1. In Definition 1, let E^+ , E^- and B be defined as follows:

$$\begin{aligned}
 E^+ &= \{daughter(mary, ann), daughter(eve, tom)\} \\
 E^- &= \{daughter(tom, ann), daughter(eve, ann)\} \\
 B &= \{mother(ann, mary), mother(ann, tom), father(tom, eve), \\
 &\quad father(tom, ian), female(ann), female(mary), \\
 &\quad female(eve), male(pat), male(tom), \\
 &\quad parent(X, Y) \leftarrow mother(X, Y), \\
 &\quad parent(X, Y) \leftarrow father(X, Y)\}
 \end{aligned}$$

Then both theories \mathcal{H}_1 and \mathcal{H}_2 defined as follows:

$$\begin{aligned}\mathcal{H}_1 &= \{daughter(X, Y) \leftarrow female(X), parent(Y, X)\} \\ \mathcal{H}_2 &= \{daughter(X, Y) \leftarrow female(X), mother(Y, X), \\ &\quad daughter(X, Y) \leftarrow female(X), father(Y, X)\}\end{aligned}$$

are complete and consistent with respect to B and E .

2.1 One-Shot Hypothesis Derivation (OSHD)

In this paper we adopt a form of ILP which is suitable for one-shot learning and is based on using a logic program declarative bias, i.e. using a logic program to represent the declarative bias over the hypothesis space. Using a logic program declarative bias has several advantages. Firstly, a declarative bias logic program allows us to easily port bias from one problem to another similar problem (e.g. for transfer learning). Secondly, it is possible to reason about the bias at the meta-level. Declarative bias will also help to reduce the size of the search space for the target concept or hypothesis derivation [1, 22]. We refer to this approach as One-Shot Hypothesis Derivation (OSHD) which is a special case of Top-Directed Hypothesis Derivation (TDHD) as described in [18].

Definition 2 (One-Shot Hypothesis Derivation). *The input to an OSHD system is the vector $S_{TDHD} = \langle NT, \top, B, E, e \rangle$ where NT is a set of “non-terminal” predicate symbols, \top is a logic program representing the declarative bias over the hypothesis space, B is a logic program representing the background knowledge and E is a set of examples and e is a positive example in E . The following three conditions hold for clauses in \top : (a) each clause in \top must contain at least one occurrence of an element of NT while clauses in B and E must not contain any occurrences of elements of NT , (b) any predicate appearing in the head of some clause in \top must not occur in the body of any clause in B and (c) the head of the first clause in \top is the target predicate and the head predicates for other clauses in \top must be in NT . The aim of a OSHD learning system is to find a set of consistent hypothesised clauses H , containing no occurrence of NT , such that for each clause $h \in H$ the following two conditions hold:*

$$\top \models h \tag{1}$$

$$B, h \models e \tag{2}$$

The following theorem is a special case of Theorem 1 in [18].

Theorem 1. *Given $S_{OSHD} = \langle NT, \top, B, E, e \rangle$ assumptions (1) and (2) hold only if there exists an SLD refutation R of $\neg e$ from \top, B , such that R can be re-ordered to give $R' = D_h R_e$ where D_h is an SLD derivation of a hypothesis h for which (1) and (2) hold.*

According to Theorem 1, implicit hypotheses can be extracted from the refutations of e . Let us now consider a simple example.

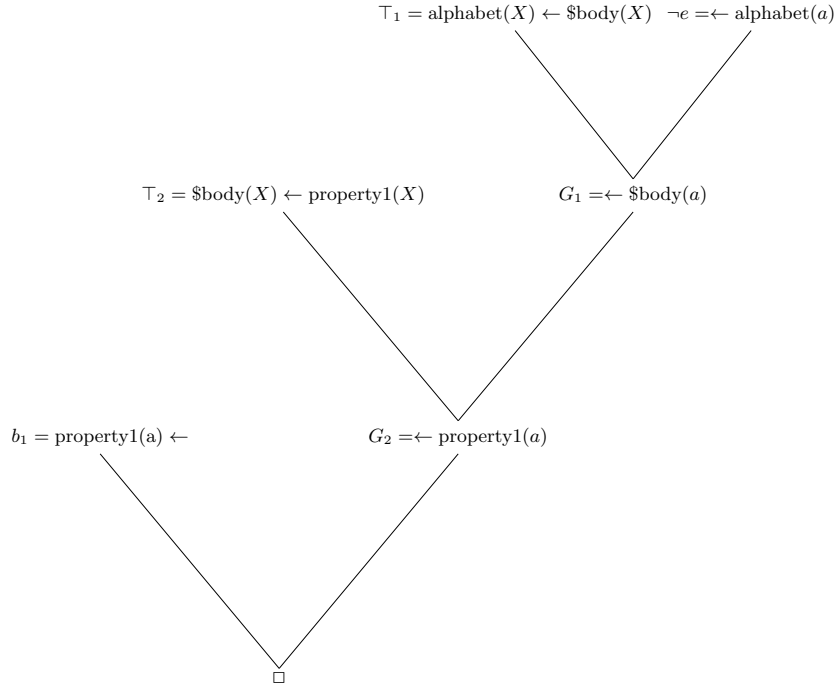


Fig. 1: SLD-refutation of $\neg e$

Example 2. Let $S_{OSHD} = \langle NT, \top, B, E, e \rangle$ where NT , B , e and \top are as follows:

$$\begin{array}{l}
 NT = \{\$body\} \\
 B = b_1 = \text{property1}(a) \leftarrow \\
 e = \text{alphabet}(a) \leftarrow
 \end{array}
 \quad
 \top = \begin{cases}
 \top_1 : \text{alphabet}(X) \leftarrow \$body(X) \\
 \top_2 : \$body(X) \leftarrow \text{property1}(X) \\
 \top_3 : \$body(X) \leftarrow \text{property2}(X)
 \end{cases}$$

Given the linear refutation, $R = \langle \neg e, \top_1, \top_2, b_1 \rangle$, as shown in Figure 1, we now construct the re-ordered refutation $R' = D_h R_e$ where $D_h = \langle \top_1, \top_2 \rangle$ derives the clause $h = \text{alphabet}(X) \leftarrow \text{property1}(X)$ for which (1) and (2) hold.

The user of OSHD can specify a declarative bias \top in the form of a logic program. A general \top theory can be also generated from user specified mode declarations. Below is a simplified example of user specified mode declarations and the automatically constructed \top theory.

2.2 The OSHD Learning Algorithm

The OSHD Learning algorithm can be described in 3 main steps:

1. Generate all hypotheses, H_e that are generalizations of e

$$\begin{array}{l}
\text{modeh(alphabet(+image)).} \\
\text{modeb(has_prop1(+image)).} \\
\text{modeb(has_prop2(+image)).}
\end{array}
\quad
\mathbb{T} = \begin{cases}
\top_1 : \text{alphabet}(X) \leftarrow \text{\$body}(X). \\
\top_2 : \text{\$body}(X) \leftarrow \text{\%emptybody} \\
\top_3 : \text{\$body}(X) \leftarrow \text{has_prop1}(X), \text{\$body}(X). \\
\top_4 : \text{\$body}(X) \leftarrow \text{has_prop2}(X), \text{\$body}(X).
\end{cases}$$

Fig. 2: Mode declarations and a \mathbb{T} theory automatically constructed from it

2. Compute the coverage of each hypothesis in H_e
3. Build final theory, T , by choosing a subset of hypothesis in H_e that maximises a given score function (e.g. compression)

In step 1, H_e is generated using the OSHD hypothesis derivation described earlier in this section.

The second step of the algorithm, computing the coverage of each hypothesis, is not needed if the user program is a pure logic program (i.e. all relationships in the background knowledge are self contained and do not rely on Prolog built in predicates). This is because, by construction, the OSHD hypothesis derivation generates all hypotheses that entail a given example with respect to the user supplied mode declarations. This implies that the coverage of an hypothesis is exactly the set of examples that have it as their generalization. However, this coverage computation step is needed for the negative examples, as they were not used to build the hypothesis set.

For step 3, the compression-based evaluation function used for the experiments in this paper is:

$$\sum Covered_Examples_Weight - Total_Literals \quad (3)$$

The weight associated to an example may be defined by the user but by default, positive examples have weight 1 and negative examples weight -1. In general, negative examples are defined with a weight smaller than 0 and positive examples with a weight greater than 0.

3 Siamese Neural Networks

In this paper, we use a state-of-the-art Deep Learning approach, called Siamese Network [8], which has been developed for one-shot learning. The original Siamese Networks were first introduced in the early 1990s by Bromley and LeCun to solve signature verification as an image matching problem [5]. A Siamese network is a Deep Learning architecture with two parallel neural networks with the same properties in terms of weight, layers etc. Each network takes a different input, and whose outputs are combined using energy function at the top to provide some prediction. The energy function computes some metric between the highest level feature representation on each side (Figure 3). Weight tying guarantees that two extremely similar images could not possibly be mapped by their respective networks to very different locations in feature space because each network

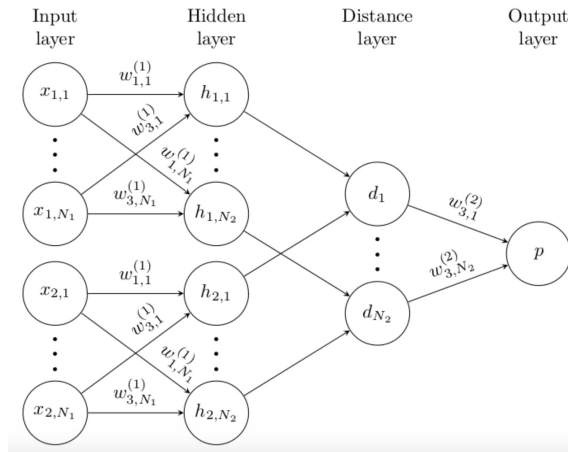


Fig. 3: A simple 2 hidden layer Siamese Neural Network [8]

computes the same function. Also, the network is symmetric, so that whenever we present two distinct images to the twin networks, the top conjoning layer will compute the same metric as if we were to present the same two images but to the opposite twin.

4 One-Shot learning for Malayalam character recognition

We apply One-Shot Hypothesis Derivation (OSHD) as well as Deep Learning (i.e. Siamese Network) to the challenging task of one-shot Malayalam character recognition. This is a challenging task due to spherical and complex structure of Malayalam hand-written language.

4.1 Character recognition and human-like background knowledge

Malayalam is one of the four major languages of the Dravidian language family and originated from the ancient Brahmi script. Malayalam is the official language of Kerala, a state of India with roughly forty-five million people. Unlike for other languages, there is currently no efficient algorithm for Malayalam hand-written recognition. The basic Malayalam characters along with International Phonetic Alphabet (IPA) are shown in Figure 4¹.

The handwriting recognition for Malayalam script is a major challenge compared to the recognition of other scripts because of the following reasons:

- Presence of large number of alphabets
- Different writing styles
- Spherical features of alphabets

¹ From: <https://sites.google.com/site/personaltesting1211/malayalam-alphabet>.

Vowel	Vowel Sign	IPA
അ	-	/a/ A
ആ	ഃ	/a:/ AA
ഇ	ി	/i/ I
ഈ	ീ	/i:/ II
ഉ	ു	/u/ U
ഊ	ൂ	/u:/ UU
ഋ	ൃ	/r:/ R
എ	െ	/e/ E
ഏ	േ	/e:/ EE
ഐ	ൈ	/aj/ AI
ഒ	ൊ	/o/ O
ഓ	ോ	/o:/ OO
ഔ	ൌ	/au/ AU
അം	ം	/am/ am
അഃ	ഃ	/ah/

(a) Malayalam Vowels

Alphabet	IPA
ല്	/l/
ല്ല്	/LL/
ര്	/RR/
ണ്	/NN/
ന്	/N/

(b) Special Consonants (Chill)

Alphabet	IPA	Alphabet	IPA	Alphabet	IPA	Alphabet	IPA
ക	/ka/ KA	ബ	/ba/ BA	ന്റ	/nta/	ത	/tʃa/
ഖ	/kʰa/ KHA	ഭ	/bʰa/ BHA	റ	/ra/ RA	ശ്ശ	/ʃʃ/
ഗ	/ga/ GA	മ	/ma/ MA	ന്ത	/nta/	ര	/rma/
ഘ	/gʰa/ GHHA	യ	/ja/ YA	ല്ല	/lla/	ത	/tma/
ങ	/ŋa/ NGA	ര	/ra/ RA	ക്ഷ	/kʃa/	വ്വ	/vva/
ച	/tʃa/ CA	റ	/ra, ʃa	ത	/tʃa/	ജ	/jʃa/
ഛ	/tʃʰa/ CHHA	ല	/la/ LA	ള	/lla/	ശ്ശ	/ʃʃ/
ജ	/dʒa/ JA	ള	/lla/ LLA	മ്പ	/mpa/	ച്ഛ	/tʃʃa/
ഝ	/dʒʰa/ JHA	ഴ	/ʒa/ ZHA	മ്മ	/mma/	ബ്ബ	/bba/
ഞ	/ɲa/ NYA	വ	/va/ VA	ക	/ka/	ഗ്ഗ	/gga/
ട	/ta/ TA	ശ	/ʃa/ SHA	ന്ത	/nta/	പ	/pa/
ത്	/tʰa/ TTHA	ഷ	/ʃa/ SSA	ന്ദ	/nda/	ന്ത	/nta/
ഡ	/da/ DDA	സ	/sa/ SA	ന്ത	/nta/	ഗ	/ga/
ഡ	/dʰa/ DDHA	ഹ	/ha/ HA	ഞ്ച	/tʃa/	ത	/ta/
ണ	/ɲa/ NNA	ക	/ka/	യ്യ	/jja/	ഹ	/ha/
ത	/ta/ TA	ന്ന	/nna/	ദ്ധ	/dʰa/	ബ	/ba/
ഥ	/tʰa/ THA	ത്ത	/tta/	ന്ത	/nta/	ഗ	/ga/
ദ	/da/ DA	ട്ട	/tta/	സ്സ	/ssa/	ദ്ധ	/dʰa/
ധ	/dʰa/ DHA	പ്പ	/ppa/	ഞ്ച	/tʃa/		
ന	/na, na/	ച്ഛ	/tʃa/	ട്ട	/tta/		
പ	/pa/ PA	ങ്ങ	/ŋa/	ക	/ka/		
ഫ	/pʰa/ PHA	ങ്ങ	/ŋa/	ത	/ta/		

(c) Consonants and Consonant Clusters

Fig. 4: Sample Malayalam characters

– Similarity in character shapes

We selected the hand-written characters from 'Omniglot' dataset [11]. Sample Malayalam alphabets from our dataset are shown in Figure 5. Feature extraction is conducted utilizing a set of advanced geometrical features [27] and directional features.

Geometrical Features Every character may be identified by its geometric designations such as loops, junctions, arcs, and terminals. Geometrically, loop means a closed path. Malayalam characters contain more intricate loops which



(a) Character 'Aha' (b) Character 'Tha'

Fig. 5: Sample Malayalam characters from our dataset

may contain some up and downs within the loops itself. So we follow a concept as shown in Figure 6(b). If the figure has a continuous closed curve then we will identify it as a loop. Junctions may be defined as a meeting point of two or more curves or line. It is easy for human to identify the junction from an image as shown in Figure 6(c). As per dictionary definitions, an arc is a component of a curve. So in our case, a path with semi opening will be considered as an arc. Please refer to Figure 6(d) for more details. Terminals may be classified as points where the character stroke ends, i.e. no more connection beyond that point. Figure 6(e) is a self-explanatory example for the definition.

We have included the visual explanation for the geometrical feature extraction in Figure 6. We have selected two characters to explicate the features as shown in Figure 5 and marked each geometrical features as we discussed. Table 1 will give an abstract conception about the dataset we have developed for the experiments from 'Omniglot' dataset.

Directional Features Every character may be identified by its directional specifications such as starting and ending points of the stroke. There are certain unwritten rules for Malayalam characters, e.g. it always commences from left and moves towards the right direction. Native Malayalam users can easily identify the starting and ending point. However, we will need to consider the starting and ending point as features so that these can be easily identified without semantic knowledge of a character. The starting and ending points are determined by standard direction properties as shown in the Figure 6(a). Figure 6(f, g) will give you an idea about developing the directional features from an alphabet. Character ID:13 is the corresponding entry for the character shown in figure 6(g). As we discussed, a user can identify both starting and ending point of the character displayed in Figure 6(g) easily whereas the terminus point of Figure 6(f) is arduous to determine.

4.2 Mode declarations and background knowledge representation

In this section, we define the OSHD specific details of the declarative bias, defined by mode declaration and background knowledge representation used in our experiments.

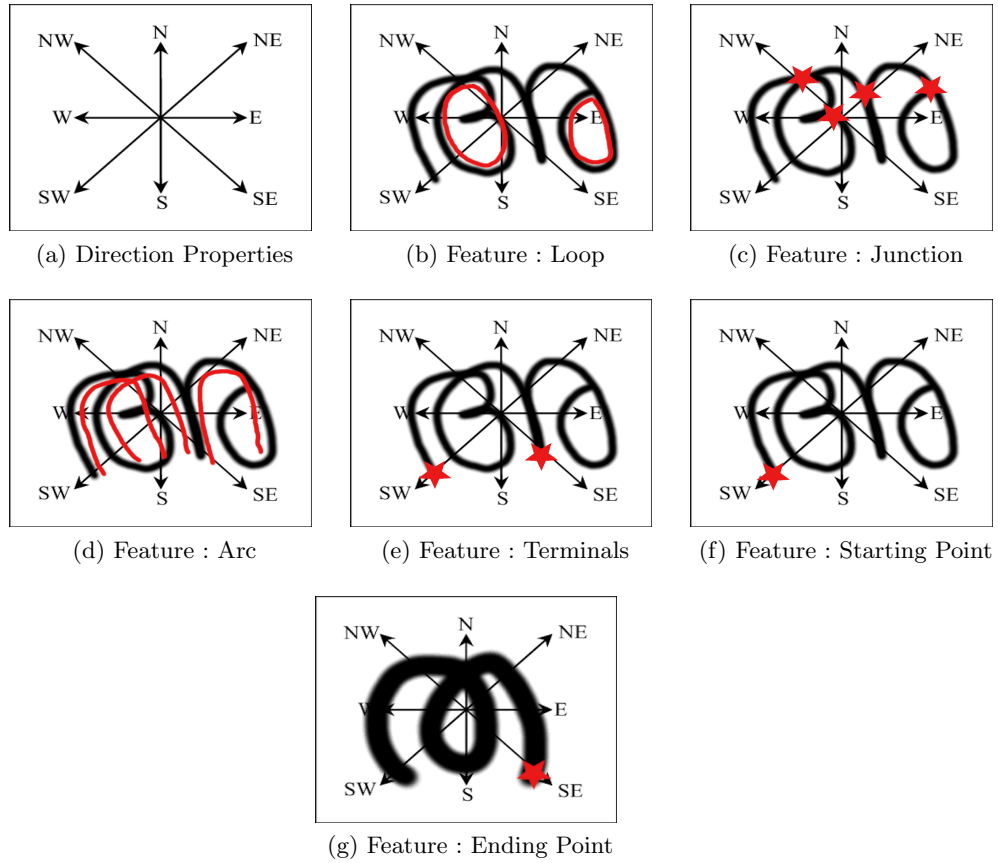


Fig. 6: Human-like feature extraction criteria

The first step was to develop and represent the background knowledge based on the concepts described in Section 4.1. Table 1 shows the geometrical and directional features of 18 characters from 5 different alphabets used in our experiments.

Mode declaration and declarative bias In this section we describe how the declarative bias for the hypothesis space was defined using mode declarations. Here, we use the same notations used in Progol [17] and Toplog [18]. There are two types of mode declarations.

1. *modeh* : defines the head of a hypothesised rule.
2. *modeb* : defines the literals (conditions) that may appear in the body of a hypothesised rule.

For example, in our experiments, *alphabet(+character)* is the head of the hypothesis, where *+character* defines the character identifier *character* as an

Table 1: Geometrical and Directional Properties

Character ID	Geometrical Properties				Directional Properties	
	No. Loops	No. Junctions	No. Arcs	No. Terminals	Starting Point	Ending Point
1	2	4	3	2	sw	null
2	3	4	3	2	sw	null
3	3	4	3	2	sw	null
4	1	2	3	2	null	se
5	1	3	3	2	nw	se
6	0	1	3	2	nw	se
7	1	1	2	1	null	se
8	1	1	2	2	nw	se
9	1	1	2	1	null	se
10	3	3	4	1	null	se
11	4	4	4	0	null	null
12	3	4	3	0	null	null
13	1	1	2	2	sw	se
14	1	1	1	2	nw	se
15	1	1	1	2	sw	ne
16	0	2	1	2	sw	ne
17	0	0	1	2	sw	ne
18	0	2	1	2	nw	ne

input argument. We are using four predicates in the body part of the hypothesis as shown in the listing 1.1. Note that +, -, indicate input, output or a constant value arguments.

Listing 1.1: Mode declarations

```

:- modeh(1, alphabet(+character)).
:- modeb(*, has_gemproperties(+character, -properties)).
:- modeb(*, has_gemproperties_count(+properties,
                                     #geo_feature_name, #int)).
:- modeb(*, has_dirproperties(+character, -properties)).
:- modeb(*, has_dirproperties_feature(+properties,
                                     #dir_feature_name, #featurevalue)).

```

The meaning of each *modeb* condition is defined as follows:

has_gemproperties/2 predicate was used to represent the geometrical features as defined in Table 1. The input argument *character* is the unique identifier for an alphabet, *properties* refers to the property name.

has_gemproperties_count/3 predicate outlines the count of the particular feature associated with the alphabet. The *properties* is the unique identifier for a particular geometrical property of a particular alphabet, *geo_feature_name* refers to the property name and *int* stands for the feature count.

has_dirproperties/2 predicate used to represent the directional features mentioned in table 1. The *character* is the unique identifier for the alphabet, *properties* refers to the property name.

has_dirproperties_count/3 predicate outlines the count of a particular directional feature associated with the alphabet. The *properties* is a unique identifier for a particular property of a particular alphabet, *dir_feature_name* refers to the property name and *featurevalue* stands for the feature vale.

Background knowledge representation As defined in Definition 1, background knowledge is a set of clauses representing the background knowledge about a problem. In general, background knowledge can be represented as a logic program and could include general first-order rules. However, in this paper we only consider ground fact background knowledge. In the listing 1.2 we have a sample background knowledge for an alphabet.

Listing 1.2: Sample background knowledge for alphabet 'Aha'

```

%% Geometrical Feature 'Loops' with feature count
has_gemproperties(character_0, loops_0).
has_gemproperties_count(loops_0, loops, 2).
%% Geometrical Feature 'Arcs' with feature count
has_gemproperties(character_0, arcs_0).
has_gemproperties_count(arcs_0, arcs, 3).
%% Geometrical Feature 'Junctions' with feature count
has_gemproperties(character_0, junctions_0).
has_gemproperties_count(junctions_0, junctions, 4).
%% Geometrical Feature 'Terminals' with feature count
has_gemproperties(character_0, terminals_0).
has_gemproperties_count(terminals_0, terminals, 2).
%% Directional Feature 'Starting Point' with feature
has_dirproperties(character_0, starts_0).
has_dirproperties_feature(starts_0, startsat, sw).

```

5 Experiments

In this section we evaluate the OSHD approach for complex character recognition as described in this paper. We also compare the performance of OSHD with a state-of-the-art Deep Learning architecture for one-shot learning, i.e. the Siamese

Network approach described in Section 4. In particular we test the following null hypotheses:

Null Hypothesis H1 OSHD cannot outperform Siamese Networks in one-shot learning for complex character recognition.

Null Hypothesis H2 OSHD cannot learn human comprehensible rules for complex character recognition.

5.1 Materials and Methods

The OSHD algorithm in this experiment is based on Top-Directed Hypothesis Derivation implemented in Toplog [18], and uses mode declarations and background knowledge which defined earlier in this paper. The Siamese Network used in the experiment is based on the implementation described in [8].

The challenging Malayalam character recognition dataset and the machine learning codes, configurations and input files are available from:

<https://github.com/danyvarghese/One-Shot-ILP>

We have selected 5 complex alphabets from the 'Omniglot' dataset [11]. Example characters used in our experiment and their visual properties (developed using the geometrical and directional concepts, as discussed in Section 4) are listed in Table 2. We have endeavoured to reiterate the same concept of working in both architectures and repeated the experiments for different number of folds and each fold consists of single positive example and n negative examples, where n varies from 1 to 4 and the negative examples are selected from other alphabets. In our experiment we are using the term 'number of classes' in different aspect. The number of classes is defined by the total number of examples (i.e. 1 positive and n negative) used for the cross-validation.

In the following we define specific parameter settings for each algorithm.

OSHD parameter settings The following Toplog parameter settings were used in this experiment.

clause_length (value = 15) defines the maximum number of literals (including the head) of a hypothesis.




weight the weight of negative example is taken always as the default value. The weight of positive example is the number of negative examples for that class.

During the cross-validation test, we add one more positive example of the same alphabet for each fold. The weight of newly added example will not be greater than the previous one included in the same fold.

positive_example_inflation (value = 10) multiplies the weights of all positive examples by this factor.

negative_example_inflation (value = 5) multiplies the weights of all negative examples by this factor.

Table 2: Sample characters & properties

Alphabet	Properties
 Alphabet 'Aha' (ID:1)	Loops : 2 Junctions : 4 Arcs : 3 Terminals : 2 Starting Point : SW Ending Point : Null
 Alphabet 'Eh' (ID:4)	Loops : 1 Junctions : 2 Arcs : 3 Terminals : 2 Starting Point : Null Ending Point : SE
 Alphabet 'Uh' (ID:7)	Loops : 1 Junctions : 1 Arcs : 2 Terminals : 1 Starting Point : Null Ending Point : SE

Siamese Networks parameter settings For the implementation of the Siamese Net, we followed the same setups used by Koch et al [8]. Koch et al use a convolutional Siamese network to classify pairs of 'Omniglot' images, so the twin networks are both Convolutional Neural Nets (CNNs). The twins each have the following architecture:

- Convolution with 64 (10×10) filters uses 'relu' activation function.
- 'max_pooling' convolution 128 (7×7) filters with 'relu' activation function.
- 'max_pooling' convolution 128 (4×4) filters with 'relu' activation function.
- 'max_pooling' convolution 256 (4×4) filters with 'relu' activation function.

The twin networks reduce their inputs down to smaller and smaller 3D tensors. Finally, there is a fully connected layer with 4096 units.

In most of the implementations of Siamese Network, they are trying to develop the training model from a high amount of data. Also, particularly in the case of character recognition, they compare a character from a language against the characters from other languages [12, 4]. In our experiment we have only considering alphabets from a single language.

5.2 Results and Discussions

Figure 7 shows the average predictive accuracy of ILP (OSHD) vs Deep Learning (Siamese Net) in One_shot character recognition with increasing number of character classes. According to this figure, OSHD outperforms the Siamese Nets,

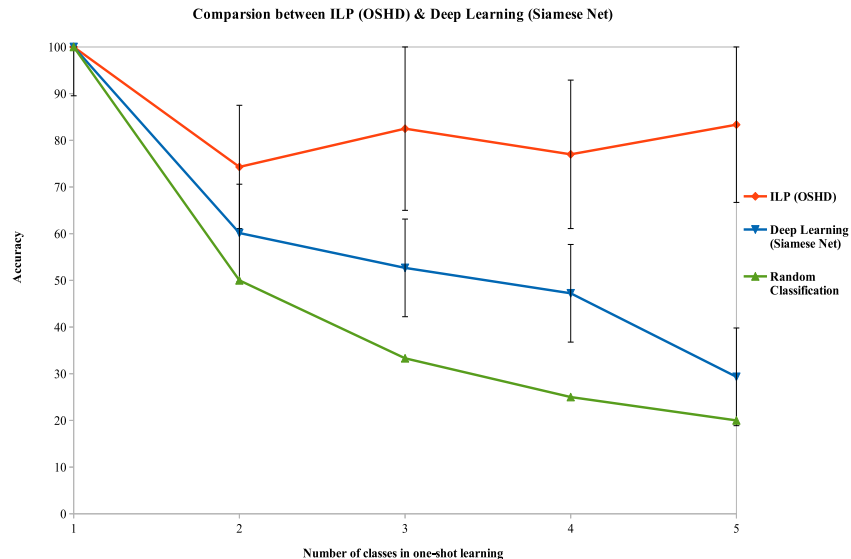


Fig. 7: Average Predictive accuracy of ILP (OSHD) vs Deep Learning (Siamese Net) in One-shot character recognition with increasing number of character classes.

with an average difference of more than 20%. In this figure the random curve represents the default accuracy of random guess. The accuracy for one class prediction is always 100%. Null hypothesis H1 is therefore refuted by this experiment. A better predictive accuracy of OSHD compared to the Siamese Net could be explained by the fact that it uses background knowledge.

Table 3 shows example of learned rules by OSHD generated from one positive and two negative examples. One can easily differentiate alphabet 'Aha' against 'Eh' & 'Uh'. The unique properties of 'Ah' from others alphabets is given in the column 'Human Interpretations', which is almost similar to the learned rule in column 4. It is also clear that the rule in column 4 is human comprehensible. Null hypothesis H2 is therefore refuted by this experiment.

6 Conclusion

In this paper, we presented a novel approach for one-shot rule learning called One-Shot Hypothesis Derivation (OSHD) which is based on using a logic program declarative bias. We applied this approach to the challenging task of Malayalam character recognition. This is a challenging task due to spherical and complex structure of Malayalam hand-written language. Unlike for other languages, there is currently no efficient algorithm for Malayalam hand-written recognition.

Table 3: Example of learned rules

+ve Example	-ve Example	Human Interpretations	Learned Rules
Alphabet 'Aha' Loops : 2 Junctions : 4 Arcs : 3 Terminals : 2 Starting Point : SW Ending Point : Null	Alphabet 'Eh' Loops : 1 Junctions : 2 Arcs : 3 Alphabet 'Uh' Terminals : 2 Starting Point : Null Ending Point : SE Alphabet 'Uh' Loops : 1 Junctions : 1 Arcs : 2 Terminals : 1 Starting Point : Null Ending Point : SE	Loops : 2 Junctions : 4 Starting Point : SW	alphabet(A) if has_gemproperties(A, B), has_gemproperties(A, C), has_gemproperties_count (B, junctions, 4), has_gemproperties(A, D)

The features used to express the background knowledge were developed in such a way that it is acceptable for human visual cognition also. We could learn rules for each character which is more natural and visually acceptable. We compared our results with a state-of-the-art Deep Learning approach, called Siamese Network, which has been developed for one-shot learning. The results suggest that our approach can generate human-understandable rules and also outperforms the deep learning approach with a significantly higher average predictive accuracy (an increase of more than 20% in average). It was clear from the results that deep learning paradigm use more data and its efficiency is less when dealing with a small amount of data. As future work we would like to further extend the background knowledge to include more semantic information. We will also explore the new framework of Meta-Interpretive Learning (MIL) [19, 20] in order to learn recursive rules.

Acknowledgments

We would like to acknowledge Stephen Muggleton and Jose Santos for the development of Top Directed Hypothesis Derivation and Toplog [18] which was the basis for One-Shot Hypothesis Derivation (OSHD) presented in this paper. We also acknowledge the Vice Chancellor's PhD Scholarship Award at the University of Surrey.

References

1. Adé, H., Raedt, L.D., Bruynooghe, M.: Declarative bias for specific-to-general ilp systems. *Machine Learning* **20**, 119–154 (1995)
2. Bengio, Y.: Learning Deep Architectures for AI. *Found. Trends Mach. Learn.* **2**(1), 1–127 (Jan 2009)
3. Bennett, C.H., Parmar, V., Calvet, L.E., Klein, J., Suri, M., Marinella, M.J., Querlioz, D.: Contrasting advantages of learning with random weights and backpropagation in non-volatile memory neural networks. *IEEE Access* **7**, 73938–73953 (2019)
4. Bouma, S.: One shot learning and siamese networks in keras. <https://sorenbouma.github.io/blog/oneshot/> (2017)
5. Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., Shah, R.: Signature verification using a “siamese” time delay neural network. In: *Proceedings of the 6th International Conference on Neural Information Processing Systems*. p. 737–744. NIPS’93, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1993)
6. Dany Varghese, Viju Shankar: A novel approach for single image super resolution using statistical mathematical model. *International Journal of Applied Engineering Research (IJAER)* **10**(44) (2015)
7. Hinton, G.: Learning multiple layers of representation. *Trends in cognitive sciences* **11**, 428–434 (11 2007)
8. Koch, G., Zemel, R., Salakhutdinov, R.: Siamese neural networks for one-shot image recognition. In: *Proceedings of the 32 nd International Conference on Machine Learning*. vol. 37 (2015)
9. Krig, S.: *Computer Vision Metrics Survey, Taxonomy, and Analysis*. Apress, Berkeley, CA (2014)
10. Lake, B., Salakhutdinov, R., Gross, J., Tenenbaum, J.: One shot learning of simple visual concepts. In: *Proceedings of the 33rd Annual Conference of the Cognitive Science Society*. pp. 2568–2573 (2011)
11. Lake, B.M., Salakhutdinov, R., Tenenbaum, J.B.: Human-level concept learning through probabilistic program induction. *Science* **350**(6266), 1332–1338 (2015)
12. Lamba, H.: One shot learning with siamese networks using keras. <https://towardsdatascience.com/one-shot-learning-with-siamese-networks-using-keras-17f34e75bb3d> (2019)
13. Lavrač, N., Džeroski, S.: *Inductive Logic Programming : Techniques and Applications*. Ellis Horwood (1993)
14. Le, Q.V., Zou, W.Y., Yeung, S.Y., Ng, A.Y.: Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In: *CVPR 2011*. pp. 3361–3368 (2011)
15. Liu, X., He, P., Chen, W., Gao, J.: Multi-task deep neural networks for natural language understanding. *CoRR* **1901.11504** (2019)
16. Mas, J., Sanchez, G., Lladós, J., Lamiroy, B.: An incremental on-line parsing algorithm for recognizing sketching diagrams. In: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*. vol. 1, pp. 452–456 (2007)
17. Muggleton, S.: Inverse entailment and Progol. *New Generation Computing* **13**, 245–286 (1995)
18. Muggleton, S.H., Santos, J., Tamaddoni-Nezhad, A.: TopLog: ILP using a logic program declarative bias. In: *Proceedings of the International Conference on Logic Programming 2008*. pp. 687–692. LNCS 5366, Springer-Verlag (2010)
19. Muggleton, S., Lin, D., Tamaddoni-Nezhad, A.: Meta-interpretive learning of higher-order dyadic datalog: Predicate invention revisited. *Machine Learning* **100**(1), 49–73 (2015)

20. Muggleton, S., Dai, W.Z., Sammut, C., Tamaddoni-Nezhad, A.: Meta-interpretive learning from noisy images. *Machine Learning* **107** (2018)
21. Muggleton, S., de Raedt, L.: Inductive logic programming: Theory and methods. *The Journal of Logic Programming* **19-20**, 629 – 679 (1994), special Issue: Ten Years of Logic Programming
22. Nedellec, C.: Declarative bias in ilp (1996)
23. Neethu, K.S., Varghese, D.: An incremental semi-supervised approach for visual domain adaptation. In: 2017 International Conference on Communication and Signal Processing (ICCSP). pp. 1343–1346 (2017)
24. Nguyen, A., Yosinski, J., Clune, J.: Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 427–436 (2015)
25. Nienhuys-Cheng, S.H., de Wolf, R.: *Foundations of Inductive Logic Programming*. Springer-Verlag, Berlin (1997), LNAI 1228
26. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. In: International Conference on Learning Representations (2014)
27. Usman Akram, M., Bashir, Z., Tariq, A., Khan, S.A.: Geometric feature points based optical character recognition. In: 2013 IEEE Symposium on Industrial Electronics Applications. pp. 86–89 (2013)