Advances in Cognitive Systems 8 (2019) 53-71 Submitted 5/2019; published 12/2019

Reflecting After Learning for Understanding

Lee Martie LEE.MARTIE@IBM.COM
Mohammad Arif Ul Alam ARIF.ALAM@IBM.COM
Gaoyuan Zhang GAOYUAN.ZHANG@IBM.COM
MIT-IBM Watson AI Lab, IBM Research, 75 Binney Street, Cambridge, MA 02142 USA

Ryan R. Anderson RRANDERS@US.IBM.COM
IBM Cloud and Cognitive Software, 505 Howard St, San Francisco, CA 94105 USA

Abstract

Today, image classification is a common way for systems to process visual content. Although neural network approaches to classification have seen great progress in reducing error rates, it is not clear what this means for a cognitive system that needs to make sense of the multiple and competing predictions from its own classifiers. As a step to address this, we present a novel framework that uses meta-reasoning and meta-operations to unify predictions into abstractions, properties, or relationships. Using the framework on images from ImageNet, we demonstrate systems that unify 41% to 46% of predictions in general and unify 67% to 75% of predictions when the systems can explain their conceptual differences. We also demonstrate a system in "the wild" by feeding live video images through it and show it unifying 51% of predictions in general and 69% of predictions when their differences can be explained conceptually by the system. In a survey given to 24 participants, we found that 87% of the unified predictions describe their corresponding images.

1. Introduction

Learning in artificial intelligence is often framed as training one or more flavors of deep neural networks (Goodfellow et al., 2016b). Advances in image classifiers, for example, rely mainly on convolutional neural networks or generative adversarial networks (Goodfellow et al., 2014). Learning identity functions through autoencoders often involve training a multilayer perceptron or similar techniques (Goodfellow et al., 2016a). While reinforcement learning can use tables (Sutton & Barto, 2018), scalable solutions often use deep neural networks, as in Mnih et al.'s (2013) work on playing Atari video games.

In brief, a deep neural network for image classification is often created by tuning weights in a neural network structure to optimize a loss function, such that one label from a given set is predicted as most likely for an input. One consequence is that the understanding of images has been narrowed to predicting a most likely label, given some input from a data distribution. Some well-known examples of this are ImageNet classifiers, such as AlexNet (Krizhevsky et al., 2012), ResNet (He et al., 2016), and SqueezeNet (Iandola et al., 2016), which have been trained on millions of images over millions of parameters to predict the "correct" label from a thousand alternatives. Indeed,

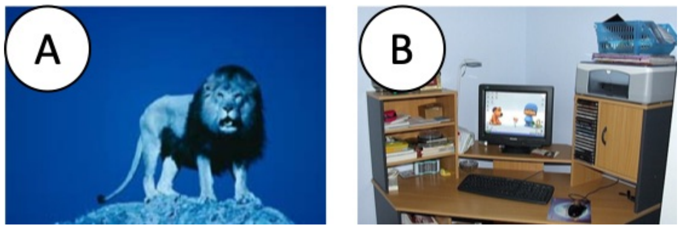


Figure 1. Image A and B from ImageNet, where A is classified as "arctic fox" by AlexNet and "ox" by ResNet, and B is classified as "desk" by AlexNet and "desktop computer" by ResNet.

ImageNet held a competition where classifiers were compared with each other according to their error rate of predicting the correct label (top–1 error rate) and the error rate of predicting the correct label in the top five predictions (top–5 error rate).

While ImageNet classifiers are reducing their error rates, it is not clear what error rate reduction says about making sense of visual input. In particular, it remains unclear how a cognitive system might know what it is looking at, especially after deployment when it cannot check "correctness" against a test set. Further, while classifiers are traditionally pitted against one another in order to find the best (Kaggle, Inc., 2019), we observe they often can output different "correct" labels after learning, regardless of what the testing data suggests. For example, we took images A and B in Figure 1 from ImageNet (Stanford Vision Lab et al., 2019) and asked AlexNet and ResNet to classify it. For Image A, AlexNet classifies this image as "arctic fox," and ResNet classifies this image as "ox". Strictly speaking, both classifiers are incorrect, and neither would know in production. However, in some sense they are also correct (i.e., both an arctic fox and ox are mammals and a mammal does appear in the image). For Image B, AlexNet classifies this image as "desk," and ResNet classifies this image as "desktop computer," but the correct label, according to the ImageNet challenge, is "desk" (Stanford Vision Lab, 2012). Again, both classifiers are correct in some sense (i.e., both a desk and desktop computer occur in the image). As such, labels are more like views of what is being seen and point to some encoded knowledge rather than a "correct" description. In this paper, we adopt the word *view* to mean the encoded knowledge in a classifier that the predicted label loosely describes.

Recent research acknowledges one label for a picture is limiting and multi-label classifiers, such as YOLO (Redmon et al., 2016), provide more labels per image. However, a critical problem remains. How can complementary or competing views, after they have already been learned, be combined into some kind of higher-level knowledge that can be understood and reasoned over? Such concept combinations appears to be a key component in how humans think (e.g., creating abstractions and relationships) (Bayne & Chalmers, 2003; Shivhare & Kumar, 2016) and is critical for using visual input to establish correct conditions for planning and verifying achievement of goals. As such, this is a key question for understanding and building cognitive systems that include frameworks for sense making.

In this paper, we investigate the research question *How can a cognitive system's different views be unified into higher-level knowledge after learning?* We address this question and the value of such a system with four main contributions:

- We present a novel approach and framework for creating cognitive systems that can reconcile views through a process of convergence, where multiple views are collected and unified into abstractions, relationships, or properties by leveraging reflection. We call our overall approach *symbolic mirroring* and refer to the paradigm as the *symbolic mirroring framework* (SMF).
- Through a detailed planning example, we demonstrate the value of our approach by showing how a system can unify its views (correct or not) to find a level of abstraction that identifies conditions when executing a plan.
- We evaluate how well our approach can unify views in an exploratory laboratory study, where we evaluate three different systems, in our framework, over 950 images from ImageNet. We find that the systems can unify a substantial number of views when their differences can be explained by differences in their abstractions. In particular, we find the programs were able to unify views for 41% to 46% of the images and, among the views the programs could explain, they could unify 67% to 75% (all $p \leq .001$). In an exploratory field study, we find a system in our framework can unify 51% of the views created from live video images and, among the views that it could explain, the system could unify 69% ($p = .041$).
- In a survey given to 24 participants, we find they agreed that 87% ($p \leq .001$) of the unified views made by a system in our framework describe their corresponding images.

The construction of the cognitive systems we evaluate utilize several novel concepts introduced in our symbolic mirroring framework (SMF). In particular, our framework introduces *meta-points*, which support reflection on executed and unexecuted portions of a cognitive system by passing parts of the system to higher-level processes called *meta-operations*. The meta-operation *explain* provides an explanation when multiple views are not the same and the meta-operation *converge* unifies these views into higher-level concepts through an algorithm based on domain knowledge specified in the OWL description logic (Horrocks, 2005) and using the OWL reasoner, Pellet (Sirin et al., 2007). In this way, meta-points and meta-operations act to bridge views from trained neural networks to higher-level knowledge in symbolic approaches.

The rest of the paper provides the details of our research. Section 2 introduces our approach and framework. Section 3 demonstrates why symbolic mirroring is useful for planning. Section 4 describes the experiment design for evaluating our approach, presents the results, and discusses threats to validity. Section 5 discusses the results, Section 6 contrasts our work with related work, and Section 7 concludes with a view on future work.

2. Symbolic Mirroring Approach and Framework

We designed our framework for building cognitive systems that can unify different views among their classifiers into higher-level knowledge through a process of convergence called *symbolic mirroring*. The conjecture behind our approach is that convolutional neural networks contain hidden knowledge describing input images, even when their predictions are incorrect. While knowledge in neural networks suffers from an opacity problem, the symbolic mirroring technique attempts to make this knowledge explicit for an input. The approach involves creating a knowledge base that expresses abstractions, properties, and relationships in the domain that the classifiers' labels reside in order to "mirror" some of the knowledge inside the classifiers – hence the name symbolic mir-

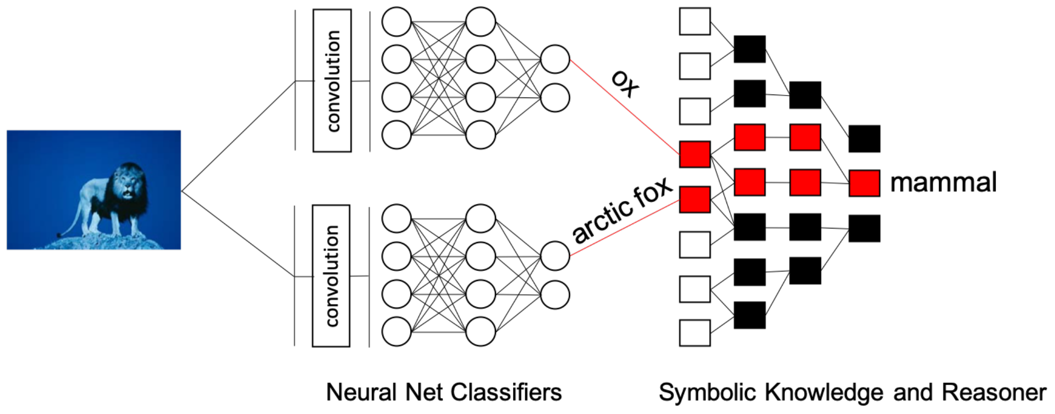


Figure 2. Example of our symbolic mirroring architecture, where knowledge in classifiers converges into a higher level of abstraction.

roring. The top predictions from two classifiers are then mapped into the knowledge base to find common abstractions, properties, and relationships. Our approach looks for common higher-level knowledge in order to converge on concepts that the predictions of the classifiers suggest exists in the image. Figure 2 presents an example of a symbolic mirroring architecture that our framework supports. In this architecture, the program attempts to unify what each classifier understands about the image by mapping their top labels (even if both are strictly wrong) onto instances (white boxes) in a knowledge base, where a reasoner finds the closest ancestor (abstraction) of both instances. In this example, the system decides it is looking at a mammal rather than an ox or an arctic fox.

The design of programs in our framework follows a composable graph design pattern, similar to the composite and command pattern (Gamma et al., 1995), but where a node in the graph has a function and annotations map the node into meta-knowledge (so it can be reasoned over at run time). A directed edge from one node to the next specifies the execution order. When a node is executed, its function is run and the return value of the function is assigned to the node. The value of one node (or any assigned to it in the past) can be retrieved by another node's function downstream as an argument. The final specified graph is executed by the framework's runtime system.

Two key node types in the framework are utilized in our programs to unify knowledge. In particular, these are meta-points and meta-operations. Meta-points are nodes that take the entire graph as input in order to perform some function (e.g., finding nodes of a particular type). They support extending programs in a modular way, because they simply can be added to any program and perform higher-level reasoning about it without the programmer needing to parameterize the node for it to work with the rest of the program. Meta-operations perform some operations on parts of the graph, but do not take the graph as input. For example, a meta-operation might make changes to the program's knowledge and/or graph structure but relies on arguments from meta-points or other nodes. We next walk through the features of the framework illustrated with an example.

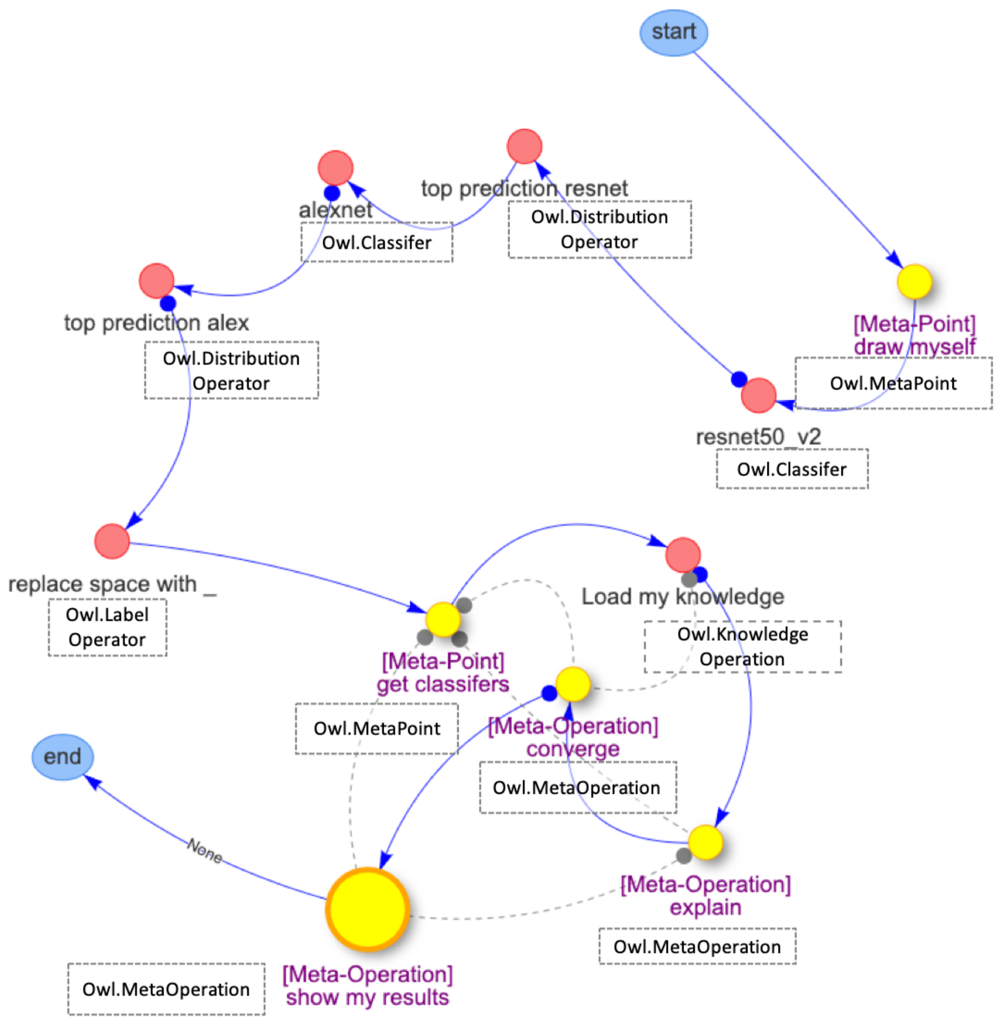
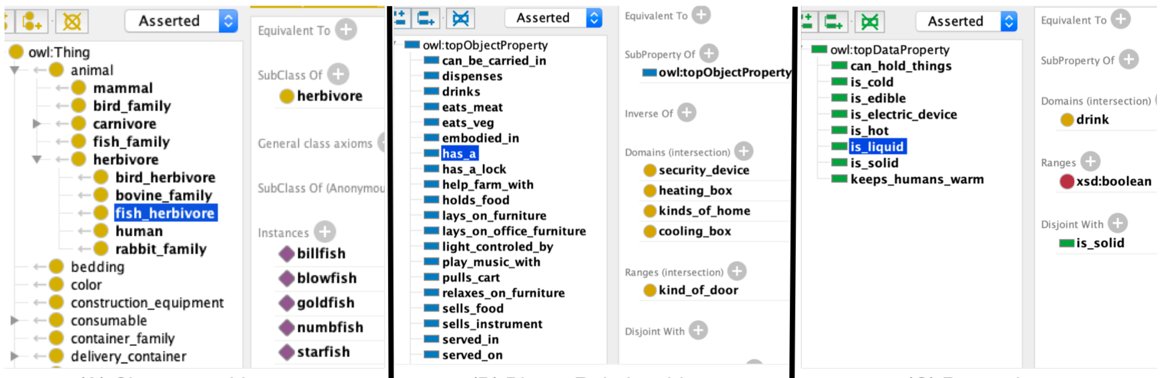


Figure 3. Automatically created graph of executing SMF program, where views of ResNet and AlexNet converge into higher-level knowledge through reflection.

2.1 Features of the Symbolic Mirroring Framework

We designed an API that supports building SMF programs as first-class entities in Python so the program itself can be reasoned over by the other meta-reasoning components in the program. Consider the graph visualization of an executed program shown Figure 3, where we annotated the nodes with their OWL class in the meta-knowledge (these appear in white boxes). The SMF runtime system executes the program in Figure 3 starting at the start node, which simply transitions to the next node, *draw_self* (a meta-point), indicated by an arrow pointing to the next node. *Draw_self* takes the entire program graph as input (as do all meta-points) and then calls a function to draw the entire program (including the node *draw_self*) to a Web application (Figure 3 is the result). The next node



owl:Thing
animal
mammal
bird_family
carnivore
fish_family
herbivore
bird_herbivore
bovine_family
fish_herbivore
human
rabbit_family
bedding
color
construction_equipment
consumable
container_family
delivery_container

SubClass Of: herbivore

Instances: billfish, blowfish, goldfish, numbfish, starfish

owl:topObjectProperty
can_be_carried_in
dispenses
drinks
eats_meat
eats_veg
embodied_in
has_a
has_a_lock
help_farm_with
holds_food
lays_on_furniture
lays_on_office_furniture
light_controled_by
play_music_with
pulls_cart
relaxes_on_furniture
sells_food
sells_instrument
served_in
served_on

SubProperty Of: owl:topObjectProperty

Domains (intersection): security_device, heating_box, kinds_of_home, cooling_box

Ranges (intersection): kind_of_door

owl:topDataProperty
can_hold_things
is_cold
is_edible
is_electric_device
is_hot
is_liquid
is_solid
keeps_humans_warm

Domains (intersection): drink

Ranges: xsd:boolean

Disjoint With: is_solid

(A) Classes and Instances (B) Binary Relationships (C) Properties

Figure 4. Sample of ImageNet domain knowledge in OWL as displayed in Protégé.

to execute is the *resnet50_v2* node, which feeds the image B shown in Figure 1 to a trained ResNet classifier via a REST request. The resulting distribution is assigned to this node. The *top prediction resnet* node retrieves the distribution of predictions from *resnet50_v2* (indicated by the edge ending in a circle) and executes a function to get the top prediction (in this case “desktop computer”). The next node to be executed is *alexnet*, which also feeds image B in Figure 1 to a trained AlexNet classifier via a REST request. The value of the *alexnet* node is another distribution of predictions that the *top prediction alex* node retrieves and, in turn, produces the top prediction (in this case “desk”). The node *replace space with _* retrieves this prediction and syntactically modifies it.

The next node to execute is another meta-point, *get classifiers*, that is fed the entire graph. This finds the values of executed classifiers by looking at the preceding executed graph and locating nodes that are annotated with the *OWL.Classifier* class. Once the classifiers are identified, the relevant parts of the executed program are retrieved downstream by the *converge*, *explain*, and *show my results* nodes (indicated by edges ending with a circle, where dashed edges indicate only a dependency exists). But the next node is *Load my knowledge*, which inputs domain knowledge for ImageNet classes that is later used to reason about the classifiers’ views. Since ImageNet labels describe common-sense concepts (e.g., computer, dog, and jacket), one author was able to create domain knowledge specifying the abstractions, properties, and relationships of the ImageNet domain in OWL using Protégé (Musen, 2015). Figure 4 shows a portion of this knowledge in the Protégé interface. In total, we created 139 classes (abstractions), 25 relationships, and nine properties. Each individual in the ontology can be assigned to a class or have inferred classes and can have one or more relationships with another instance (called *object properties* in OWL) and have one or more properties (called *data properties* in OWL).

Once the ontology is loaded, the *explain* node is the next to be executed. This meta-operation explains why different views do not match in terms of their OWL classes. First, it maps the views to individuals in the ontology by linking their labels to individuals. That is, it treats views as instances of concepts that it knows about. If no concept is found, no explanation can be made. If the mapping

Table 1. Procedure for unifying views into higher-level knowledge.

```
Converge (Label: V1, Label: V2, Domain_Knowledge: DK, Reasoner: R)
# initialize to empty set
1: Higher_Level_Knowledge = {}
# matches the labels of the classifier with an individual in the ontology
2: OWL_Individual_1 = Match_View_With_Individual (V1, DK, R)
3: OWL_Individual_2 = Match_View_With_Individual (V2, DK, R)
# find the common properties between individuals (Properties)
4: Common_Properties =
Get_Properties (OWL_Individual_1, DK, R) ∩ Get_Properties (OWL_Individual_2, DK, R)
# find binary relationships between these individuals (Relationships)
5: Common_Relationships = Get_Relationships (OWL_Individual_1, OWL_Individual_2, DK, R)
# find common ancestors (Abstractions)
6: Common_Lowest_Common_Ancestor = Lowest_Level_Ancestor (
Ancestors (OWL_Individual_1, DK, R), Ancestors (OWL_Individual_2, DK, R))
# add to knowledge
7: Higher_Level_Knowledge =
Common Properties ∪ Common Relationships ∪ Common Lowest Common Ancestor
# return
8: Return Higher_Level_Knowledge
```

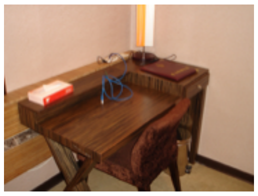
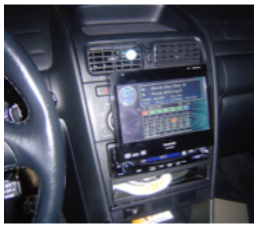
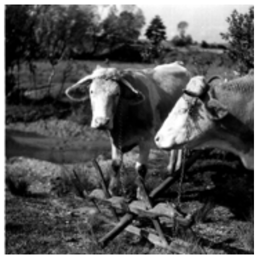
is made, the explain node looks to see if the instances are the same or not. If not, it returns an explanation about how they differ by class. In our example, the explain node outputs "desk is a kind of furniture and desktop computer is a kind of computing device." Next, the *converge* node takes in the parts of the program from the meta-point *get classifiers* and the domain knowledge from the *Load my knowledge* node. Here, the *converge* node attempts to unify the views from the classifiers into higher-level knowledge. Because the converge algorithm reasons about the system's inferences about what it is seeing (i.e., the classifiers' views), the converge node involves a type of meta-operation. We describe this general algorithm next.

2.2 Converge Algorithm

The converge algorithm is based on three principles: (1) multiple image classifiers provide multiple views of an image, where each is encoded knowledge in the classifier and can be treated as an instance of a concept; (2) views relate to each other in terms of common abstractions, relationships, or properties that unify them into higher-level knowledge; and (3) labels loosely describe a view and so views relate to each other in terms of how their labels relate to each other.

Table 1 gives a high-level walk through of how the converge algorithm unifies views by common properties, abstractions, and relationships. Given two labels *V1* and *V2*, domain knowledge *DK*, and an OWL Reasoner *R*, the algorithm begins at line 1 by instantiating an empty set of knowledge that will ultimately contain the convergences made. Lines 2 and 3 call the *Match_View_With_Individual* function, which maps the views from classifiers onto instances that exist in domain knowledge, *DK* using the views' labels. For example, if *V1* is the string "banana" and *V2* is the string "chim-

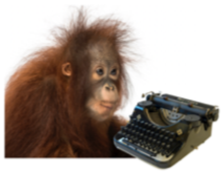
Table 2. SMF program output running on three ImageNet pictures.

Image	ResNet	AlexNet	Explain Output	Converge Output
	Table lamp	Dining table	table lamp is a **kind of furniture** and dining table is a kind of **kind of table**	**Furniture**
	CD player	Radio	CD player is a **kind of listening device** and radio is a **kind of listening device**	**Listening device**
	Ox	Plow	ox is a **kind of bovine family** and plow is a **kind of farming device**	Ox **help farm with** Plow

panzee", then each call to *Match_View_With_Individual* uses the reasoner *R* and knowledge *DK* to issue a SPARQL (SPARQL Working Group, 2013) query to find an individual named "banana" or "chimpanzee" and returns the found OWL individual (*OWL_Individual_1* and *OWL_Individual_2*, respectively).

For both OWL individuals found, Line 4 calls the *Get_Properties* for *OWL_Individual_1* and *OWL_Individual_2* and returns their intersection. Each *Get_Properties* function issues a SPARQL query, using *R* on *DK*, to find any properties that exist in *DK* (either existing in the ontology or inferred by *R*) for an individual and returns the properties. Next, Line 5 calls the *Get_Relationships* function which finds relationships between the individuals by issuing SPARQL queries for such relationships (either existing in the ontology or inferred by *R*). Continuing with our example, if the relationship *chimpanzee eats banana* is in the ontology, then it will be returned. Next, line 6 finds the lowest common ancestor between two individuals. It does this first by getting the ancestors (either existing in the ontology or inferred by *R*) for each individual by calling the Ancestors function, which, through a recursive call of SPARQL queries (each getting the ancestors of the last ancestors), returns an ordered list of closest to furthest ancestor for the individual. Both lists of ancestors are then passed to the *Lowest_Level_Ancestor* function which scans both lists to find the closest shared ancestor. Finally, on lines 7 and 8, the union of properties, relationships, and abstractions is returned.

Table 3. Images mapped to state in the world by classifier and SMF program. We obtained the typewriter from ImageNet and orangutan from Depositphotos (2019).

	Initial State P1	Initial State P2	Goal State P2
Images			
ResNet Output	orangutan	typewriter	typewriter
AlexNet Output	langur	typewriter	spider monkey
SMF Output	**primate**	**typewriter**	spider monkey has typewriter ⇒ **primate has typewriter**

To give an understanding of the knowledge produced from this algorithm, we ran three images from ImageNet through the example program in Figure 3 and provide the output in Table 2, where the rightmost column has the results from running the converge algorithm.

3. An Example of SMF Program Usefulness

The ability to unify different views of trained image classifiers into higher-level knowledge should improve the ability to formulate and execute plans. In particular, symbolic mirroring finds the level of abstraction on which multiple views agree so the system can avoid asserting incorrect conditions in the world. Establishing correct conditions controls for erroneously executing plans on false preconditions and verifying achievement of postconditions. Combining the SMF program described in Figure 3 with Graphplan (Blum & Furst, 1997), we demonstrate how symbolic mirroring can take opposing views from classifiers and unify them into a description that correctly describes conditions for planning. This is a crucial ability for a system that has no ground truth to reference, as occurs in novel data sets.

Suppose, for simplicity, that the system's goal is to instruct a primate to retrieve a nearby typewriter. The first action it must take is to perceive the world for any primates and typewriters. In this scenario, the system looks at locations P1 and P2 and perceives the images as shown in Table 3 (Initial State Image P1 and P2). Further, suppose the system has the preconditions and operations as shown in Table 4's first two columns for finding a plan using the Graphplan syntax. We now walk through several reasons why symbolic mirroring is important in this scenario.

SMF Programs Can Generate Plans on General Conditions. First, suppose our SMF program did not unify different views from classifiers, but rather used just one classifier (ResNet or AlexNet)

Table 4. World facts and planning operators specified in Graphplan.

World Facts	Operators	Plan Found
	(**operator** GOTO (**params** (<x>) (<y>)) (**preconds** (**at** primate <y>)) (**effect** (del at primate <y>) (**at** primate <x>)))	
(P1) (P2) (**preconds** (**at** primate P1) (**at** typewriter P2)) (**effects** (has-typewriter))	(**operator** GRAB-TYPEWRITER (**params** (<y>)) (**preconds** (**at** typewriter <y>) (**at** primate <y>)) (**effects** (has-typewriter)))	# go to P2 from P1 1 GOTO_P2_P1 # grab typewriter at P2 2 GRAB-TYPEWRITER_P2

with Graphplan. In the case of ResNet, the program would perceive an orangutan and a typewriter (as indicated in the *ResNet* row in Table 3). As such, it could not use a plan that works on primates in general, but rather would need a separate plan for orangutans and for each other type of primate it could possibly predict (at least nine). The same is true for a program that includes only AlexNet, since it is also an ImageNet classifier. We would need to specify many more operators, conditions, and goals for these kinds of programs to generate plans. That is, such programs are unable to generate plans on more general conditions. However, the full SMF program, with symbolic mirroring, will be able to establish the preconditions (as shown in Table 4) by unifying orangutan and langur into primate.

SMF Programs Avoid Incorrect Plans. Second, suppose we added domain knowledge back into the simplified program discussed above. That is, the SMF program is now composed of one classifier (ResNet or AlexNet), Graphplan, and the ImageNet domain knowledge. For the program that uses ResNet, it could use a reasoner to infer that an orangutan is a primate to establish the preconditions and use the operators in Table 4 in order to generate a plan. The same is true for the program that uses AlexNet. However, if the domain knowledge is very detailed and complete (as desired), then many other conditions would also become true. For the ResNet example, plans contingent on the existence of an animal, primate, orangutan, or omnivore all could be tried. For the AlexNet example, plans contingent on the existence of an animal, primate, langur, or herbivore could be attempted. This could be disastrous if the classifier does not have an accurate view. For example, a system relying only on AlexNet might execute a feeding plan for langurs (an herbivore) for what is actually an orangutan (an omnivore), which could lead to health problems. However, the full SMF program, with symbolic mirroring, finds the closest abstraction (primate) that fits both orangutan and langur, so it would only attempt a plan contingent on primate or an ancestor.

SMF Programs Can Establish Goal Conditions. Third, suppose we again have a program composed of one classifier, Graphplan, and domain knowledge. In this scenario, suppose that it produces the plan shown in the *Plan Found* column of Table 4. Upon executing this plan, the program cannot visually confirm the goal has been obtained. After executing line 2 of the plan (GRAB-TYPEWRITER_P2), the program using ResNet can only confirm that a typewriter is at P2, as shown in Table 3. The program using AlexNet will assert that a spider monkey is at P2, as shown in Table 3. However, the SMF program can unify both classifier's views into the assertion that a spider monkey has a typewriter through the relationship *has* in its domain knowledge. The system could then use this relationship to infer that a primate has a typewriter, because spider monkey has the ancestor class primate.

4. Exploratory Studies

As a first step in understanding the process of developing systems in our SMF, how they perform on ImageNet data sets, and the generality of symbolic mirroring in them, we ran an exploratory study on three different programs. They are all variations on Figure 3, where program RA uses classifiers ResNet and AlexNet, RS uses ResNet and SqueezeNet, and AS uses AlexNet and SqueezeNet. While there are many classifiers to choose from, we selected ResNet, AlexNet, and SqueezeNet to see if larger or smaller differences in the top–1 accuracies (77.11, 54.92, 56.11) had an effect on the number of different views. Further, they are some of the best known and evaluated image classifiers in research. We provided each program with the meta-knowledge and domain knowledge described in Figure 4. We also performed a brief field study in which we fed live images to an SMF program while walking around. This tested the system on a data distribution that differed from training and provided an idea of the practicality of using SMF in real environments.

4.1 Study Designs

In order to mimic some of the situations in which a SMF system might be created, we created RA, RS, and AS under five design conditions:

- The author who created the domain knowledge (as shown in Figure 4) was given the thousand ImageNet labels beforehand and approximately three days to create domain knowledge in OWL using Protégé. This placed a realistic constraint on how complete the knowledge could be before the program was run.
- Another author chose 950 pictures distributed by ImageNet into the categories of animals (250), electronics (250), food (250), and furniture (200).
- The categories were shown to the author of the domain knowledge (but not the pictures) so that he might focus some of his attention here, but there was no strict requirement. The author of the knowledge also received 20 example images for each category to help in creating relationships, properties, or abstractions.
- While the abstractions and properties were based on possible labels from a classifier, the relationships were inferred by the author who created the domain knowledge by looking at example images. These relationships might not actually exist in the test set, a concern that we address with an evaluation in Section 4.4.
- The classifiers used in these programs were "off the shelf" from Gluon (Apache Software Foundation, 2019) and were not specially trained for this experiment. This helped us evaluate symbolic mirroring orthogonal to learning.

Table 5. Explained and unified results (percentages rounded) in laboratory study.

ID	Top–1 Acc. Diff.	Different Views Total	Unified Total	Disunited Total	Explained Total	Not Explained Total	Unified Total in Explained	Disunited Total in Explained
RA	22.2	369 (39%)	159 (43%)	210 (57%)	239 (65%)	130 (35%)	159 (67%)	80 (33%)
RS	21	384 (40%)	158 (41%)	226 (59%)	220 (57%)	164 (43%)	158 (72%)	62 (28%)
AS	1.2	349 (37%)	161 (46%)	188 (54%)	215 (62%)	134 (38%)	161 (75%)	54 (25%)

For the field study, we extended RA in Figure 3 so that the images fed through it were from a live video camera and we added a speech synthesizer so that it would speak aloud the views it unified. We did this by adding a few more nodes to the program graph of RA. We walked around in an office setting and recorded the number of unified views.

4.2 Study Results

We next present the results from the laboratory and field studies. We measured the performance of the programs by the percentage of unified views and explanations they created. We also include results from a survey conducted to measure if the unified views made by a SMF program actually describe something in the images fed to it (a type of "accuracy"). We include significant ($\alpha = .05$) p values by applying Chi-squared on 2×1 contingency tables.

Table 5 presents the results for programs RA, RS, and AS. The *Top–1 Acc. Diff.* column provides the differences between the reported top–1 accuracies for the classifiers used in the programs. *Different Views Total* provides the total amount of views that differ by label created from 950 different images, *Unified Total* gives the number of different views unified, and *Disunited Total* reports the total number of differing views that were not unified. *Explained Total* gives the number of explained differences between views, while *Unexplained Total* provides the number of views that could not be explained because they had no mapping into knowledge (i.e., there are missing classes in the knowledge base). *Unified Total in Explained* gives the number of views unified among the explained and *Disunited Total in Explained* gives the number of views not unified among the explained. We do not present *Unified Total in Unexplained* because they were 0% for each row, which means that, if differing views were unexplained, they could not be unified.

We found that between 41% to 46% of views were unified and between 54% to 59% remained disunited (p $\leq$ .001 for RS and p $\leq$ .008 for RA). We also found that 57% to 65% of different views were explained for each program (all p $\leq$ .004), and that among these 67% to 75% were unified for each program (all p $\leq$.001). In addition, we found that the accuracy differences between the classifiers used in each program have only a small impact on the number of different views produced (the maximum difference between the quantity of different views was 3%).

Table 6 gives the breakdown among the unified views (159, 158, 161) in terms of the kinds of convergences that happened by abstraction, properties, and relationships. In some cases, views were unified by more than one method, as given in the *Multiple Unified Total* column: this is why the

Table 6. Frequency of unified views by type results (percentages rounded) from laboratory study.

ID	Abstractions Total	Properties Total	Relationships Total	Multiple Unified Total	Unified Total
RA	149 (79%)	30 (16%)	10 (5%)	30 (16%)	189
RS	143 (78%)	26 (14%)	15 (8%)	25 (14%)	184
AS	149 (78%)	31 (16%)	12 (6%)	33 (17%)	192

Unified Total column has numbers greater than 159, 158, and 161, respectively. We found the most common way to unify views was through abstraction (i.e., lowest common ancestors), followed by unifying through common properties and relationships. We present a few results from each category in Table 7 from RA as illustrations.

For the field study, a person held a camera that fed images from an office environment into the RA program (as described in Figure 3) but modified so that it took live video images. In total, the system processed 55 images from the camera. Figure 5 presents the ordering of these results, where "S" means no differing views were generated, "U" means differing views were unified, "D" means differing views were not unified, and "D*" means differing views that had an explanation were not unified. We found 39 (71%) differing views occurred. Of the 39 differences, 20 were unified (51%) and 19 (49%) were not. Some 29 (74%) of the differing views had explanations (counting "U" and "D*"). Of these, 20 views (69%) were unified and nine (31%) remained disunited ($p = .041$).

4.3 Survey Results on Unified Views

In order to understand if the unified views actually make sense as descriptions of images, we conducted a survey to evaluate a kind of "accuracy" for the program RA. Traditionally, accuracy of a classifier is measured before deployment with a test set, but an SMF program uses classifiers that have already been deployed and the system is being used regularly.

In an online survey, we asked 24 people (three authors and 21 volunteers recruited over a company channel) to evaluate if unified views described their corresponding images or not. Specifically, we asked each participant *Does the word or phrase listed to the left of each picture describe something in the picture?* for 40 unified views and their corresponding images. They indicated their answer with a button labeled "agree" or "disagree." For each participant, ten unified views made by program RA, were chosen at random from each category (furniture, food, electronics, and animals), so all categories were represented. We decided to ask only 40 questions, rather than 159 for all

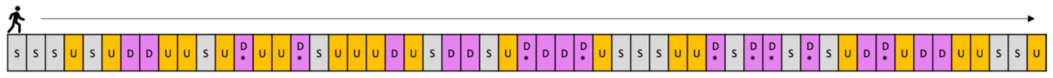

S S S U S U D D U U S U D* U U D* S U U U D U S D D S U D* D D D* U S S S U U D* S D* D* S D* S U D D* U D D U U S S U

Figure 5. Ordered encoding of unified, disunited, and explained views in field study results.

Table 7. Three ImageNet examples from laboratory study by each type of convergence.

Image	ResNet	AlexNet	Explain Output	Converge Output
	Backpack	Purse	backpack is a **kind of carrying device** and purse **is a kind of carrying device**	**Carrying device** (abstraction)
	Crib	Quilt	crib is a **kind of bed** and quilt is a **kind of bedding**	Quilt **lays on furniture** crib (relationship)
	Ice Cream	Mashed Potato	ice cream is a **kind of sweets** and mashed potato is a **kind of prepared potatoes**	**Is edible** (property) **Food family** (abstraction)

unified views made by RA, because the length of the survey would then have been prohibitively long. In total, we asked 960 questions across the 24 participants.

We found that for 839/960 (87%) of the questions, the participants agreed the unified views described something in the image ($p \leq .001$). By category, participants agreed the unified views described their corresponding images for 222/240 (92.5%) of the animal images ($p \leq .001$), 209/240 (87%) of the furniture images ($p \leq .001$), 203/240 (84.5%) of the food images ($p \leq .001$), and 205/240 (85.4%) of the electronic images ($p \leq .001$).

4.4 Threats to Study Validity

While our results are promising, there are threats to its validity. For external validity, our study is limited to a common-sense domain (ImageNet labels) and we did not seek expert opinion on the domain knowledge created. Thus, the results may not apply to expert areas, but common-sense reasoning in cognitive systems is important (Davis & Marcus, 2015). Further, while the laboratory study provided control for us to take measurements without noise, we had to give up the realism of the study by using the ImageNet data set. However, we attempted to compensate for this with a field study that demonstrated the approach in a real environment on a different data distribution. Since the laboratory and field study told similar stories, we are confident in our results.

For internal validity, the abstractions and properties in the domain knowledge were inferred from the labels of the classifiers that were known to exist in the test set. However, by the developer inferring the relationships from the few example images he had might mean they do not exist in the test set. To see if they did, we conducted a survey among seven people (four authors and three volunteer, five male and two female) who took an online survey presenting each with a relationship produced from the laboratory study, the associated image, and a choice of "relationship exists" or "relationship does not exist" (creating 37 questions total). We asked them to use their own judgement when choosing and we provided no definition of relationship. We found that, for 29 (78%) of the images, most participants agreed the relationship existed in the image. This gives us confidence that, when a relationship was used to unify views, the relationship likely existed.

5. Discussion

The results suggest our approach and framework can help build cognitive systems capable of unifying views into higher-level knowledge after learning. While 54% to 59% of results were not unified in the laboratory study, the results also showed that 41% to 46% were unified in the laboratory study and 51% unified in the field study. For systems processing thousands to millions of images, this could have a large impact. Further, the system greatly improves when differing views were explained (i.e., views could be mapped into the domain knowledge classes), where the system could unify 67% to 75% of views in the laboratory study and 69% in the field study. This suggests that a cognitive system built with our SMF can gain a unified understanding of 41% to 75% of images being processed (possibly millions).

The survey results (87% of unified views are descriptive) suggest that the symbolic mirroring approach often produced accurate descriptions. However, it also means that views with incorrect labels were unified into words or phrases people thought were descriptive of the corresponding images. This implies that the symbolic mirroring approach is a means to describe the knowledge encoded in classifiers themselves and could help reveal what is learned in a classifier rather than relying only on its predicted label.

The results also suggest two important and potentially powerful developer methods of improving cognitive systems. First, as we saw in Table 5, if the system could explain differing views, then its chance of unifying them improved greatly. This suggests that developers could continually add instances of concepts to the domain knowledge every time their program encountered a view from a classifier that it had not encountered before. Tools for recording when the cognitive system does not know how to unify views or is "confused" will be needed. Second, as discussed in Subsection 4.4, the author of the domain knowledge could infer relationships from relatively few examples and these relationships often existed in the test set. This suggests developers could add relationships to domain knowledge with relatively few image examples, an interesting contrast to the millions of examples used by machine learning today.

In the laboratory study setting, the author creating the domain knowledge was given the four categories animals, electronics, food, and furniture to help focus his development. However, many of the views in the experiment did not map into these categories, as seen by the number not explained because of missing knowledge. This suggests that differing views from classifiers can vary widely across categories, which means that "conceptually wide" domain knowledge is needed to account for conceptual differences among the different views.

6. Related Work

Our research contrasts with previous work in meta-reasoning along two dimensions. First, we have investigated meta-reasoning as a sense-making mechanism to unify the visual world by bridging neural networks with symbolic approaches. Second, we have provided a modular framework for using meta-reasoning and meta-operations in a more flexible way than the classic approach.

Meta-reasoning has been described as an essential component of general intelligence, where it is often argued that it is critical for choosing different ways to think given finite resources, as in humans (Epstein & Petrovic, 2011; Griffiths et al., 2019). One example is changing game strategies in new settings. For example, adaptation in response to the external environment has been demonstrated by agents reasoning over models of their own strategies (Goel & Jones, 2011; Rugaber et al., 2013) and altering those models to meet external demands. One aspect of these agents is their representation of themselves in the Task Method Knowledge Language (Murdock & Goel, 2008), which lets the agent reason over its goals and methods. Similarly, meta-reasoning has been applied to self-monitoring and self-repair when anomalous situations arise (Schmill et al., 2011).

From an interactive perspective, a question and answer system has been designed to answer "common sense" kinds of questions about itself (Morbini & Schubert, 2011). This approach is particularly appealing from a debugging point of view, where explanations can provide understanding and link events (Cox, 2011). In image processing, work on pipeline selection has used meta-reasoning for choosing alternatives based on context (Robertson & Laddaga, 2011), much as in Auto ML (Feurer et al., 2015). In contrast, our work applies meta-reasoning to unify viewpoints of the visual world into higher-level knowledge structures that the system can use in a variety of applications (e.g., planning or explanation). In particular, in Section 3, we demonstrated how our approach can be used to avoid incorrect conditions in planning and to detect goals. Further, it is a step closer to building systems that think in a more unified, human-like way (Bayne & Chalmers, 2003; Shivhare & Kumar, 2016) than the disjoint predictions of image classifiers.

Gilpin et al. (2018) demonstrated a monitoring technique to analyze captions from image classifiers and determine if they are reasonable. Their method involved mapping each part of speech from the captions onto slots in frames that represent primitive actions. In contrast, our work is at the lower level of the perception. In particular, symbolic mirroring unifies different views in terms of abstractions, properties, or relationships, which can then be monitored for reasonableness with systems like those demonstrated by Gilpin et al.

While there are a variety applications for meta-reasoning, architectures are often presented in a uniform way that includes a ground level, an object level, and a meta-level (Russell & Wefald, 1991). Yet this separation has an impact in the design of these systems in two ways. First, operations in each level must be treated as one type, but common reasoning methods will often cut across layers. Indeed, some have claimed "...metareasoning and reasoning are entangled in such a way that it is impossible to separate" (Robertson & Laddaga, 2011). Second, the runtime system must have three separate modes and one intermediate mode – making it hard to seamlessly switch between levels. This also has the consequence that the execution of the ground or object levels are separate (making it harder to intervene and control execution from the meta-level). In this work, we introduced meta-points and meta-operations as nodes in a graph, where each node can be any arbitrary function that switch from any level at any execution point.

7. Conclusion

In this paper, we identified the problem of unifying different views of classifiers into high-level knowledge after learning. We developed a general approach, symbolic mirroring, and a framework that leverages meta-reasoning in order to unify different views from classifiers into higher-level symbolic knowledge. Through a planning example, we demonstrated how an SMF program avoids generating plans based on incorrect assertions, how it supports general planning operations, and how it helps identify goal conditions. In an exploratory laboratory study, we found that SMF programs unified 41% to 46% of the views of images and, among the views that could be explained, unified 67% to 75% of these. Further, in an exploratory field study, we found the SMF program unified 51% of the views from live video images and, among those that could be explained, the system could unify 69% of them, showing the approach can work on a different data distribution than ImageNet and "in the wild". Further, a survey of 24 people suggested that 87% of the unified views made by the SMF program described the images it processed.

Our future work will proceed in two directions. From the lower level, we will investigate what kinds of predictions from image classifiers work better in our approach. Rather than using image classifiers that predicate complex objects (e.g., dog or jacket), we will see if simpler predictions (e.g., edge or square) from different classifiers can be unified into more complex objects using symbolic mirroring. In a sense, we will be investigating the balance between the complexity of predictions with the complexity of knowledge and reasoning needed to unify predictions. From the higher level, we will explore how to unify abstractions, relationships, and properties into higher-level stories that have temporal components (as in the field study) or that are created from a single image. In particular, we will see how to take unified views from sequential video frames and unify them into descriptive sentences. Applications could extend to letting a system understand the ethical implications of a story it creates from images.

Acknowledgements

We would like to thank the MIT-IBM Watson AI Lab, Nicola Palmarini, Grady Booch, and the reviewers for their support and insight in shaping this research.

References

Apache Software Foundation (2019). Gluon Model Zoo. Retrieved May 21, 2019, from `https://mxnet.incubator.apache.org/api/python/gluon/model_zoo.html`.

Bayne, T., & Chalmers, D. J. (2003). What is the unity of consciousness? In A. Cleeremans (Ed.), *The unity of consciousness: Binding, integration, and dissociation*, 23–58. Oxford, UK: Oxford University Press.

Blum, A. L., & Furst, M. L. (1997). Fast planning through planning graph analysis. *Artificial Intelligence*, *90*, 281–300.

Cox, M. T. (2011). Metareasoning, monitoring, and self-explanation. In M. T. Cox & A. Raja (Eds.), *Metareasoning: Thinking about thinking*, 131–149. Cambridge, MA: MIT Press.

Davis, E., & Marcus, G. (2015). Commonsense reasoning and commonsense knowledge in artificial intelligence. *Communications of the ACM*, *58*, 92–103.

Depositphotos, Inc. (2019). Stock Photos, Royalty Free Images, Vectors, Footage. Retrieved July 6, 2019, from `https://depositphotos.com/`.

Epstein, S. L., & Petrovic, S. (2011). Learning expertise with bounded rationality and self-awareness. In M. T. Cox & A. Raja (Eds.), *Metareasoning: Thinking about thinking*, 43–57. Cambridge, MA: MIT Press.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., & Hutter, F. (2015). Efficient and robust automated machine learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems 28*, 2962–2970. Montreal, Canada: Curran Associates.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design patterns: Elements of reusable object-oriented software*. Boston, MA: Addison-Wesley Longman Publishing Co.

Gilpin, L. H., Macbeth, J. C., & Florentine, E. (2018). Monitoring scene understanders with conceptual primitive decomposition and commonsense knowledge. *Advances in Cognitive Systems*, *6*, 45–63.

Goel, A. K., & Jones, J. (2011). Metareasoning for self-adaptation in intelligent agents. In M. T. Cox & A. Raja (Eds.), *Metareasoning: Thinking about thinking*, 151–165. Cambridge, MA: MIT Press.

Goodfellow, I., Bengio, Y., & Courville, A. (2016a). Autoencoders. In I. Goodfellow, Y. Bengio, & A. Courville (Eds.), *Deep learning*, 493–516. Cambridge, MA: MIT Press.

Goodfellow, I., Bengio, Y., & Courville, A. (2016b). *Deep learning*. Cambridge, MA: MIT Press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., et al. (2014). Generative adversarial nets. *Proceedings of the Twenty-Seventh International Conference on Neural Information Processing Systems* (pp. 2672–2680). Cambridge, MA: MIT Press.

Griffiths, T. L., Callaway, F., Chang, M., Grant, E., & Lieder, F. (2019). Doing more with less: Metareasoning and meta-learning in humans and machines. *Current Opinion in Behavioral Sciences*, *29*, 24–30.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770–778). Las Vegas, NV: IEEE Computer Society.

Horrocks, I. (2005). OWL: A description logic based ontology language. In P. van Beek (Ed.), *Principles and practice of constraint programming - CP 2005*, 5–8. Springer Berlin Heidelberg.

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). SqueezeNet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size. From `http://arxiv.org/abs/1602.07360`.

Kaggle, Inc. (2019). Competitions | Kaggle. Retrieved May 21, 2019, from `https://www.kaggle.com/competitions`.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Proceedings of the Twenty-Fifth International Conference on Neural Information Processing Systems* (pp. 1097–1105). Lake Tahoe, Nevada: Curran Associates.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing Atari with deep reinforcement learning. From `http://arxiv.org/abs/1312.5602`.

Morbini, F., & Schubert, L. (2011). Metareasoning as an integral part of commonsense and autocognitive reasoning. In M. T. Cox & A. Raja (Eds.), *Metareasoning: Thinking about thinking*, 267–282. Cambridge, MA: MIT Press.

Murdock, J. W., & Goel, A. K. (2008). Meta-case-based reasoning: Self-improvement through self-understanding. *Journal of Experimental & Theoretical Artificial Intelligence*, *20*, 1–36.

Musen, M. A. (2015). The Protégé project: A look back and a look forward. *AI Matters*, *1*, 4–12. From `https://doi.org/10.1145/2757001.2757003`.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 779–788). Las Vegas, NV: IEEE Computer Society.

Robertson, P., & Laddaga, R. (2011). Metareasoning for multispectral satellite image interpretation. In M. T. Cox & A. Raja (Eds.), *Metareasoning: Thinking about thinking*, 101–117. Cambridge, MA: MIT Press.

Rugaber, S., Goel, A. K., & Martie, L. (2013). GAIA: A CAD environment for model-based adaptation of game-playing software agents. *Procedia Computer Science*, *16*, 29–38.

Russell, S., & Wefald, E. (1991). Principles of metareasoning. *Artificial Intelligence*, *49*, 361–395.

Schmill, M. D., Anderson, M. L., Fults, S., Josyula, D., et al. (2011). The metacognitive loop and reasoning about anomalies. In M. T. Cox & A. Raja (Eds.), *Metareasoning: Thinking about thinking*, 183–198. Cambridge, MA: MIT Press.

Shivhare, R., & Kumar, C. A. (2016). On the cognitive process of abstraction. *Procedia Computer Science*, *89*, 243–252.

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., & Katz, Y. (2007). Pellet: A practical OWL-DL reasoner. *Web Semantics*, *5*, 51–53.

SPARQL Working Group (2013). SPARQL 1.1 Query Language. Retrieved June 28, 2019, from `https://www.w3.org/TR/sparql11-query/`.

Stanford Vision Lab (2012). ImageNet large scale visual recognition competition 2012 (ilsvrc2012). Retrieved May 21, 2019, from `http://image-net.org/challenges/LSVRC/2012/index`.

Stanford Vision Lab, Stanford University, & Princeton University (2019). ImageNet. Retrieved July 6, 2019, from `http://www.image-net.org/download-images`.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). Cambridge, MA: MIT Press.