A new paradigm for computer vision based on compositional representation

Vinjai Vale

MIT PRIMES
May 14, 2018

Abstract.

Deep convolutional neural networks - the state-of-the-art technique in artificial intelligence for computer vision - achieve notable success rates at simple classification tasks, but are fundamentally lacking when it comes to representation. These neural networks encode fuzzy textural patterns into vast matrices of numbers which lack the semantically structured nature of human representations (e.g. "a table is a flat horizontal surface supported by an arrangement of identical legs"). This paper takes multiple important steps towards filling in these gaps. I first propose a series of tractable milestone problems set in the abstract two-dimensional ShapeWorld, thus isolating the challenge of object compositionality. Then I demonstrate the effectiveness of a new compositional representation approach based on identifying structure among the primitive elements comprising an image and representing this structure through an augmented primitive element tree and coincidence list. My approach outperforms Google's state-of-the-art Inception-v3 Convolutional Neural Network in accuracy, speed, and structural representation in my object representation milestone tasks. Finally, I present a mathematical framework for a probabilistic programming approach that can learn highly structured generative stochastic representations of compositional objects from just a handful of examples. This work is foundational for the future of general computer vision, and its applications are wide-reaching, ranging from autonomous vehicles to intelligent robotics to augmented and virtual reality.

1 Introduction

In the past half-decade, deep neural networks have claimed their spot at the forefront of artificial intelligence algorithms for image processing [1]. These neural networks and related state-of-the-art techniques are designed for tasks of *classification* – problems that require the computer to sort images into categories. Such problems include "is this a picture of a cat," "does this mammogram show signs of early-stage breast cancer," etc. Deep learning has demonstrated promising results on a wide variety of classification tasks, even outperforming human experts in some examples [2]. On the other hand, modern computer vision techniques are far behind humans when it comes to tasks testing *representation* – understanding of what an object *is*, rather than being able to assign a label to it [3].

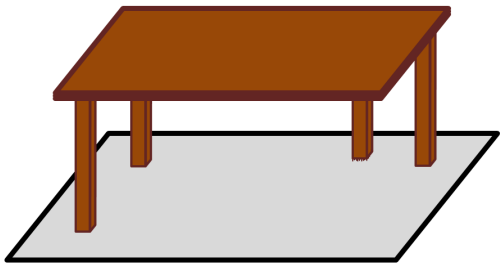

Figure 1: What is depicted in this image – and what is wrong with it?

Consider the image in Figure 1. A human observer quickly and intuitively perceives its contents: it is a table. A modern image recognition deep neural network can make the same classification. However, this is where the similarities end. A human observer demonstrates a deep level of *scene understanding* by instantly noticing something awry – the bottom half of the front right leg has been broken off. The human can intuitively diagnose that the table must be fairly unstable – it would be especially unsafe to sit on the front-right corner.

How do humans perform such a complex series of observations from just the given image? The answer is that we have an intuitive grasp of *object compositionality* – the idea that an object may be recursively represented by the sum of its components and how they relate to one another. When presented with an image of a table, we can apply an abstract definition of the table concept – "a flat surface with an arrangement of identical legs for support underneath" – to discern the different parts of the table, and construct an internal *compositional representation* in our minds of how these parts relate. State-of-the-art computer vision algorithms are incapable of this feat.

Additionally, algorithms that classify and cannot represent are clueless upon encountering new objects composed of old components. However, a human who sees a new object – even without knowing its name – may visualize it, brainstorm uses for it, and articulate a description of it. Again, this is due to our understanding of object compositionality.

Another key difference is that humans need only a few (and often just one) examples of a new object in order to learn how to tell it apart from other objects. On the other hand, computers need thousands upon thousands of examples in order to reliably identify objects as simple as handwritten digits (for instance, the classic MNIST dataset contains 60,000 images of digits written with different styles, bold or italic, clearly or sketchily) [4]. Because we perceive structures such as loops and strokes, we are able to very quickly learn an abstract conception of the digit 3, while a computer that works purely off of pixels needs a massive data set [5]. The idea of compositional abstract conceptions applies to more complex everyday objects; for instance, our perception of a table as a flat object supported by an array of identical legs is easily learnable and articulable.

The need for a fundamentally different approach can be motivated by more closely examining contemporary models. The best-performing algorithms on object recognition tasks are deep convolutional neural networks (CNNs), which use small filters as feature detectors by convolving them across an entire image [1]. The use of local textural features allows CNNs to be invariant to translations and rotations of the image of an object. This has allowed CNNs to reach unprecedented levels of accuracy in many classification tasks. However, it also means that CNNs have no perception of spatial structure – a significant flaw when it comes to scene understanding. While CNN-based models can classify some objects well, they fail in illustrative ways on more sophisticated compositional objects, such as the leopard print sofa and horse in a zebra costume in Figure 2.

Figure 2: A leopard print sofa, which virtually all state-of-the-art convolutional neural network object recognizers classify as a jaguar, panther, or leopard, and a horse in a zebra costume, which is classified as a zebra. (Image sources: [6], [7])

The lack of spatial and compositional understanding exhibited by CNNs becomes apparent when a trained CNN is tasked to generate new examples of the objects it has learned to classify. Figure 3 depicts the results of a deep CNN that draws dogs: note that while the generated textures are correct, the CNN does not know how many heads, noses, eyes, ears, legs, or tails a dog should have, or where they should go. This produces anatomically inaccurate results [8].

The deep-CNN "Show and Tell" system open-sourced by Google in 2016 takes object recognition a

Figure 3: "Dogs" generated by a deep CNN. The textures are correct, but the spatial and compositional arrangement is far from realistic. (Image source: [8])

step farther by tackling the problem of image captioning [9]. These captions, such as "a herd of elephants walking across a dry field" and "a person riding a motorcycle on a dirt road," account for very high-level interobject relations. Show and Tell achieved a 93.9% accuracy rate on the ImageNet classification task. This is an impressive accomplishment but it still suffers from the same drawbacks as the object-recognizer CNNs. Indeed, Show and Tell was trained on over 1 million images. Additionally, many of the failure cases – such as "a refrigerator filled with lots of food and drinks" for a road sign – illustrate the shallowness of the understanding present (Figure 4). We do not want black boxes that have no explanations for their sporadic failures to be driving our cars – human-understandable compositional representations come with an important safety element as well.

Figure 4: Three illustrative failure cases of "Show and Tell," Google's highly optimized state-of-the-art image captioning system [9]. The unrelated captions highlight the shallow level of understanding offered by the deep CNN approach. (Image source: Vinyals et al. 2016)

A fundamentally different approach is needed to solve the conceptual understanding problem, which would thereby lead to more powerful machine intelligence systems. To create such an approach, we must isolate a problem that encompasses the highly abstract notions of object compositionality and scene understanding. This work approaches the problem from the perspective of the *vision as inverse graphics* paradigm. I introduce a new dataset called ShapeWorld and concrete and tractable tasks that test for compositional object

representation. Then I present my novel inverse graphics approach combined with human-vision-inspired representational features, and demonstrate their effectiveness in addressing the ShapeWorld tasks through a controlled experiment against a state-of-the-art baseline. Finally, I discuss a mathematical framework for a probabilistic programming approach that can learn representations of new objects from just a handful of examples.

This paper builds on related work that argues the importance of the compositional approach to scene understanding. The key differentiator of my work is that I lay out a practical and generalizable framework for compositional object representation, and then go on to implement a proof of concept and prove its effectiveness in a problem space that I designed in order to practically isolate the challenge of compositionality. A more thorough discussion of related work can be found in Section 9.

2 The ShapeWorld dataset and representation tasks

The ShapeWorld dataset was designed to make tractable the abstract challenges discussed in Section 1. All images in ShapeWorld are composed of combinations of grayscale rectangles and circles. Each image contains a single *object* which is an instance of an abstract *concept*. In real-world images, a specific cat would be an object and the idea of a cat would be the corresponding concept.

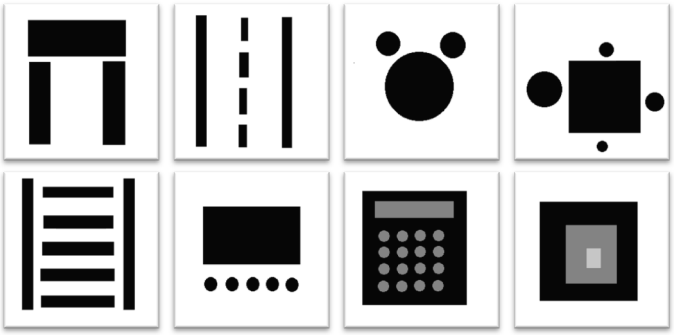

Figure 5: Example images from ShapeWorld

The key justification for this reduction of the problem is that objects in ShapeWorld exhibit the same compositionality as real-world objects. I claim that all real-world objects are made up of a collection of 2-D and 3-D *primitive elements* – the "atoms" of vision, also known as *geons* in the field of psychology [10]. From this perspective, ShapeWorld is just like the real world except with fewer primitives and fewer ways in which they can be combined. Removing all the noise of real-world images through this reduction allows for the specific targeting of the problem of compositional representation.

Note that some concepts always manifest themselves with the exact same collection of primitive elements (e.g. all Rubik's cubes have the same structure) – I call these concepts and objects *monoconfigu-*

rational. On the other hand, some concepts can manifest themselves via a wide variety of primitive element collections, each of which is represented under the same single concept (e.g. tables can have different shapes and numbers of legs). For this I use the term *multiconfigurational* (Figure 6). ShapeWorld includes the additional axis of difficulty posed by multiconfigurational objects.

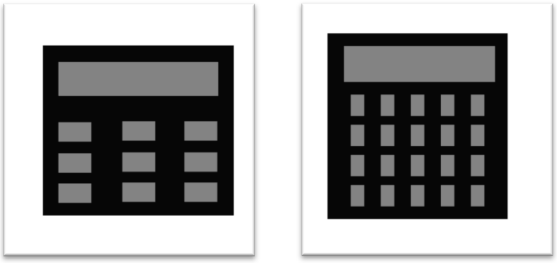

Figure 6: The multiconfigurational calculator concept, which can be explained to a human as "a rectangle containing another rectangle above a grid of small rectangles."

I now define five concrete tasks of increasing complexity to test for compositional representation ability on ShapeWorld objects.

Task 1: Decomposition into primitive elements. This task is the most fundamental and is a prerequisite to the other four. It consists of creating a parse function P that takes an image of an object, $\mathcal{I}$, to a collection of its primitive elements (rectangles and circles), $\mathcal{E}$.

Task 2: Compositional representation based on primitive elements. This task entails generating a compositional representation based on the output of Task 1. This representation would encode information about the spatial relationships between the primitive elements in the image.

Task 3: Classification via prewritten definitions of compositional concepts. In this task an algorithm should be able to apply prewritten compositional definitions of several concepts to an image in order to find a match. These definitions can be fairly abstract: for example, "a traffic light consists of a tall, narrow rectangle with three identical vertically stacked circles centered on the inside." However, the definitions are still spelled out by a user and are not learned.

Task 4: Classification via one-shot learning on features based on compositional representations. Learning is introduced in this task, but instead of learning from raw pixels, the algorithm learns from a set of features derived from the compositional representations produced by Task 2. One-shot learning – that is, learning from only one or a few examples – is specifically chosen because it is an important aspect of human learning that can potentially be achieved via compositional representation. Models that complete this task can be compared against state-of-the-art neural network baselines.

Task 5: Learning a generative model. The final task is to infer probability distributions over the space of compositional representations – I call these distributions *stochastic representations*. Note that stochastic representations act as generative models for concepts. A stochastic representation can be used for

one-shot learning and classification by computing the probability that a new image belongs to a concept. It can also be used for generating new examples of a learned concept – the computational equivalent of imagination – by sampling from the probability distribution.

The remainder of the paper is devoted to discussing my approaches to these tasks.

3 Task 1: Decomposition into primitive elements

The first task is to create a function P to identify each of the n primitive elements in an input image. My approach is based on the paradigm of *vision as inverse graphics* [11]. Graphics programming languages are designed to take the code for primitive elements and render an image consisting of them. I seek to accomplish the opposite, hence the name "inverse graphics." My approach employs the structure of a graphics program – a sequence of drawing instructions – to represent primitive elements.

Incorporating color images into ShapeWorld is an important and immediate venue of future work, so the current implementation of primitive element decomposition is designed to work for color images as well as grayscale. The implementation first employs Canny edge detection [12] on each of the red, green, and blue channels of a color image $\mathcal{I}$ to construct three bitmaps marking the edges. Their union is taken as the edge map of the original image and is inputted into the contour detection algorithm by Suzuki et al. [13]. The next step is polygonal approximation via the Ramer-Douglas-Peucker algorithm [14]; from here I may conclude whether each element is a rectangle or a circle. (These image processing algorithms are implemented in the OpenCV Python library [15].)

Note that this approach can be extended to include other primitive elements, ranging from simple triangles to much more complex and specific shapes. These elements can be identified by comparison of the Hu moments – seven values for any two-dimensional shape that are invariant to scale, rotation, and reflection (with the exception of the seventh, which undergoes a sign change upon reflection) [16].

Once the contour corresponding to each primitive element is identified as either a rectangle or a circle, the shape's specific coordinates are extracted (center, height, and width for a rectangle; center and radius for a circle). Finally, the areas between contours are used as masks to determine the color of each shape.

This procedure is a function P such that $P(\mathcal{I})$ returns the list $\mathcal{E} = \{\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_n\}$ of primitive elements, where each $\mathcal{E}_i$ includes the element's shape, its specific coordinates, and its color.

4 Task 2: Compositional representation based on primitive elements

My compositional representation of the primitive elements of an image comes in two parts: an augmented primitive element tree $\mathcal{T}$ and a coincidence set Λ.

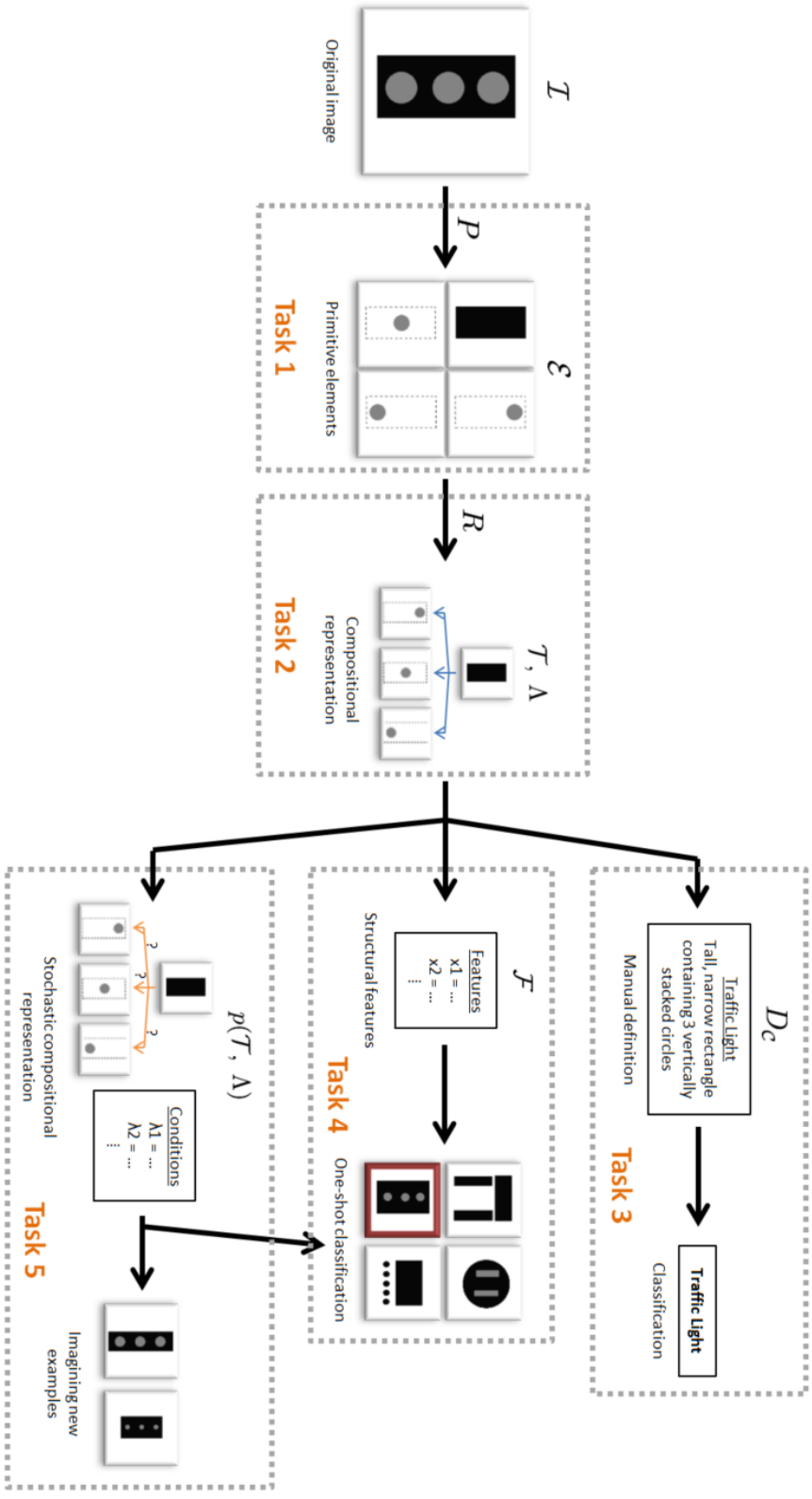

Figure 7: The five milestone compositional representation tasks proposed in Section 2.

4.1 Augmented primitive element tree

The augmented primitive element tree (APET) $\mathcal{T}$ is a set of *augmented primitive elements* $\{S_1, S_2, \ldots, S_n\}$, where each S_i includes the following information:

- the primitive element $\mathcal{E}_i$
- the next largest primitive element containing $\mathcal{E}_i$
- a list of the primitive elements contained within $\mathcal{E}_i$
- the contour of $\mathcal{E}_i$
- the depth of $\mathcal{E}_i$ in the tree

The APET of a set of elements is computed by iterating through the contour lists generated in Section 3 and identifying which contours contain which others. In constructing $\mathcal{T}$ I make the clarifying assumption that if two primitive elements share some area, one is fully contained in the other; i.e. no two primitive elements partially overlap. Object occlusion is a separate problem which is important to address in future work. Now, however, it adds an unnecessary layer of complexity and is orthogonal to the problem of compositionality, so it may safely be omitted from this discussion.

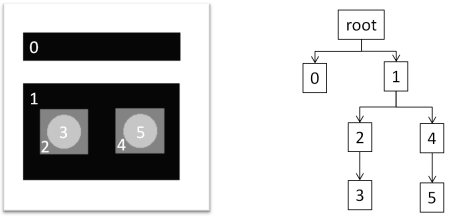

Figure 8: An example augmented primitive element tree for the ShapeWorld image on the left.

4.2 Coincidences

The second component of my compositional representation is the coincidence list Λ. A *coincidence* is any special configuration among the primitive elements in an image that is unlikely to occur in a random arrangement of elements. I include coincidences in my compositional representations because such a configuration, if present, is likely an important aspect of the concept itself. My implementation of the compositional representation focuses on row and cluster coincidences, and can be extended to include other coincidences like duplicate elements, edge alignment, horizontal/vertical alignment, radial arrangements, grids, etc.

4.2.1 Rows

A *row* is a coincidence of three or more elements arranged in a linear fashion. I implemented a straightforward $O(n^3)$ algorithm to detect these rows based on a threshold value.

4.2.2 Clusters

I define the distance between two primitive elements to be the smallest Euclidean distance between a point in one and a point in the other. I then generate a graph with n vertices, each corresponding to a primitive element. An edge is drawn between two vertices if the distance between the corresponding primitive elements is less than some threshold value. I can then group the primitive elements into sets called *clusters* if their corresponding vertices in the graph form a connected component. These connected components are identified via a depth-first search, yielding the sets of clusters.

5 Task 3: Classification via prewritten definitions of compositional objects

The goal of the third task is to take advantage of my compositional representation method to write down a dictionary of concepts with human-readable, compositional definitions.

Each definition is a function that attempts to reconstruct an appropriate augmented primitive element tree and coincidence list ($\mathcal{T}'$, Λ') from the compositional representation of a candidate image ($\mathcal{T}$, Λ). The function first latches onto an *anchor element* in the topmost layer of $\mathcal{T}$, and attempts to build up $\mathcal{T}'$ and Λ' by searching for other elements relative to the anchor element.

My implemented definitions also assign contextual descriptions to elements. This is important because it accounts for an important aspect of compositionality, namely that different collections of elements can play different roles in scene understanding in different contexts. For example, a circle could be the eye in a face concept, a light in a traffic light concept, or a wheel in a car concept – and it is impossible to determine what the circle represents without knowing the broader concept that it is a part of.

At any point if a `check()` operation returns false, the function determines that the image is not an instance of the concept being tested with the chosen anchor element, and the algorithm either reattempts the function with a different anchor element or tests a different concept. The definition function of the `TrafficLight` concept is below. Thanks to the compositional representation of the image, I am able to write a definition function that parallels the abstract English definition: "A traffic light has a skinny rectangular back panel containing three circular lights that are vertically stacked, centered, and identical."

```
def buildTrafficLight(anchor):
    check(anchor is a rectangle)
    check(anchor has aspect ratio between 1:4 and 1:1.4)
    check(anchor contains three elements)
    check(all three elements are circles)
    check(centers of anchor and circles form a row)
    check(row is vertical)
```

```
check(all three circles are same size)
add_context(anchor, "back panel")
add_context(first circle, "top light")
add_context(second circle, "middle light")
add_context(third circle, "bottom light")
```

Because the definition is framed as program, I can use `if` and `else` conditionals to include multiconfigurational objects.

6 Task 4: Classification via one-shot learning on features based on compositional representations

In this section I show how my compositional representation is used to generate a list of features to assist in one-shot learning of new concepts. The feature vectors from my representations are compared to the feature vectors generated by a state-of-the-art convolutional neural network, Google's Inception-V3 [17]. This comparison is done by training a linear support vector machine (SVM) on the two groups of features and comparing their accuracy in classifying ShapeWorld images (Figure 9).

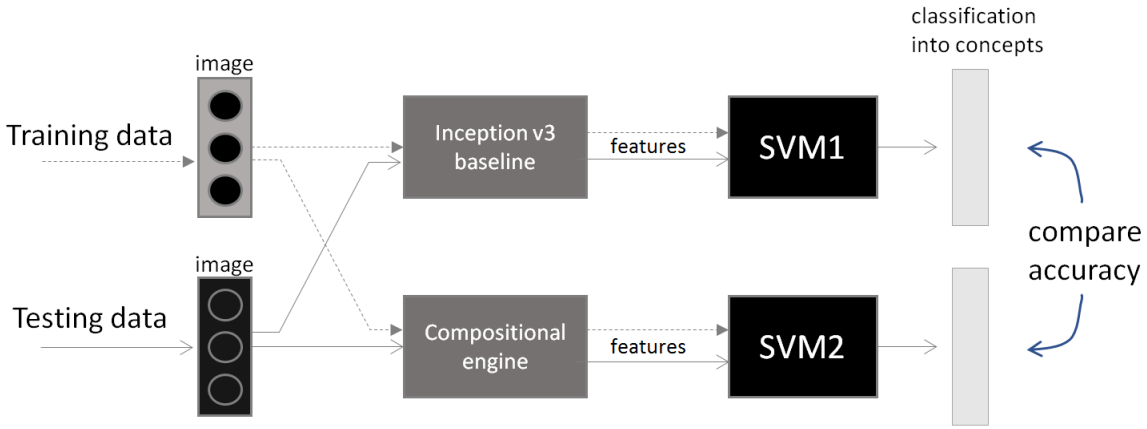

Figure 9: Schematic of the comparison to be conducted in this section. The central question: how do the features computed using my compositional representation compare in performance to the features computed by a state-of-the-art convolutional neural network?

6.1 Inception-v3 baseline features

Inception-v3 is the deep convolutional neural network by Google that won the ImageNet classification challenge in 2015 [17]. Inception-v3 was trained on 1.2 million images covering 1000 classes of everyday objects, and achieved accuracy ratings over 90%.

Since 2014, massive CNNs like AlexNet [1] – the first successful neural network trained on the ImageNet dataset – have been incorporated into baselines for feature sets following the method explained by Razavian et al. [18]. Like any other neural network, Inception-v3 consists of several interconnected layers of neurons; between every two consecutive layers is a series of mathematical operations that combines the outputs of one layer to yield the inputs of the next. Through the training of the neural network, the parameters of these mathematical operations are optimized to yield the highest possible classification accuracy.

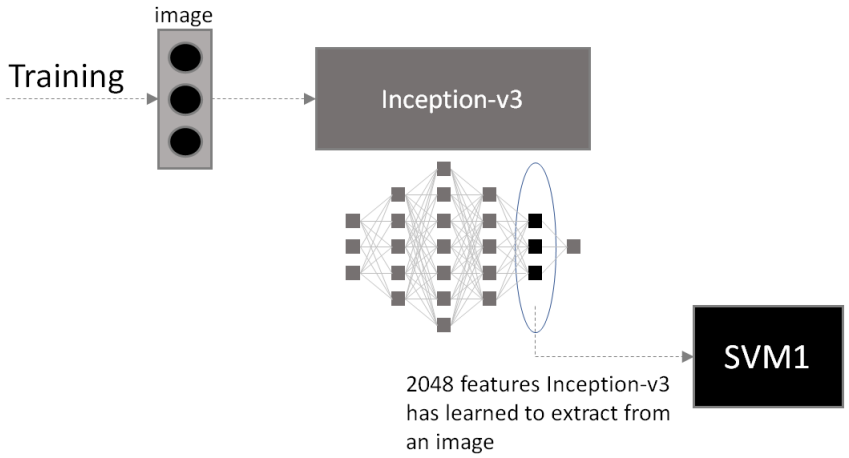

Figure 10: A schematic of feature extraction from the Inception-v3 CNN, which has been trained on the 1.2 million images in the ImageNet database.

The final layer of Inception-v3 outputs a number corresponding to the predicted classification of the input image. Hence the penultimate layer can be interpreted as a set of features that are computed from the original image, which Inception-v3 has identified as generally useful for classification. Indeed, Razavian et al. found that the generic descriptors extracted from a pre-trained image classification CNN formed a powerful and effective feature set for SVM classification [18]. I use a similar approach, yielding 2048 baseline features corresponding to the 2048 neurons in the penultimate layer of the Inception-v3 network (Figure 10).

6.2 Custom features

My custom features to be compared with the Inception-v3 CNN must be derived from $\mathcal{T}$ and Λ, the compositional representations found in Task 2. In total, 34 features are computed, including functions of the numbers of primitive elements, the coordinates of primitive elements, the sizes and arrangements of rows and clusters, and the distribution of elements at various depths of the APET.

num_circles	avg_circle_area_per_largest_shape_area
num_low_thresh_rows	num_circles_in_rects
num_high_thresh_clusters	depth_of_tree
average_rect_width	average_circle_radius

Table 1: A handful of representative features used by my model.

6.3 Comparison

A dataset was created of 90 ShapeWorld images, with 5 instances of each of 18 concepts. A subset of this dataset is depicted in Figure 11.

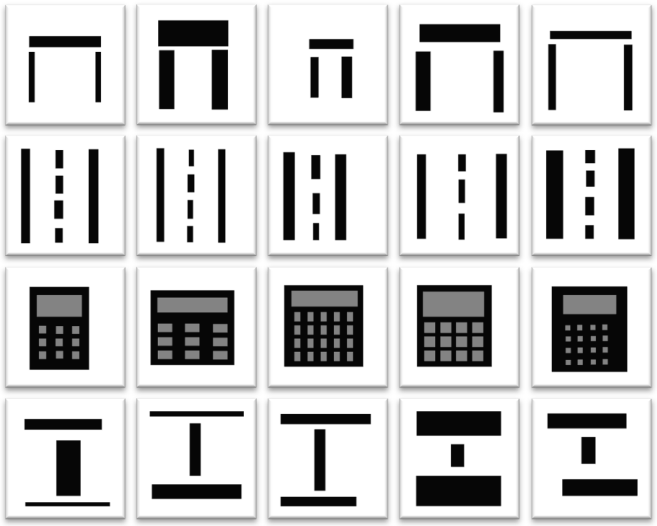

Figure 11: Four example concepts in the ShapeWorld dataset with five instances each.

To conduct a comparison, several test runs were conducted. Each run consisted of training a linear support vector machine (SVM) on i examples of each concept. These i training examples were randomly chosen from the 5 available instances of each concept, and the remaining $5-i$ were used for test data. Classification accuracy on the $18(5-i)$ test data images was determined for both the compositional representation features and the Inception-v3 features. Finally, 150 test runs for each of $i = 1, 2, 3, 4$ were run with both sets of feature algorithms. The results are summarized in Figure 12.

My compositional representation features achieve results that are consistently more accurate than Inception-v3 for training from one, two, three, and four examples respectively. My approach uses only 34 features as opposed to the 2048 of Inception-v3, a sixty-fold reduction in the number of parameters. Additionally, my features are several times quicker to compute than those of Inception-v3, which is widely regarded as the state-of-the-art. Finally, my features carry clear meaning as to the compositional structure of an image, while the 2048 arbitrary mathematical functions generated by Inception-v3 are uninterpretable.

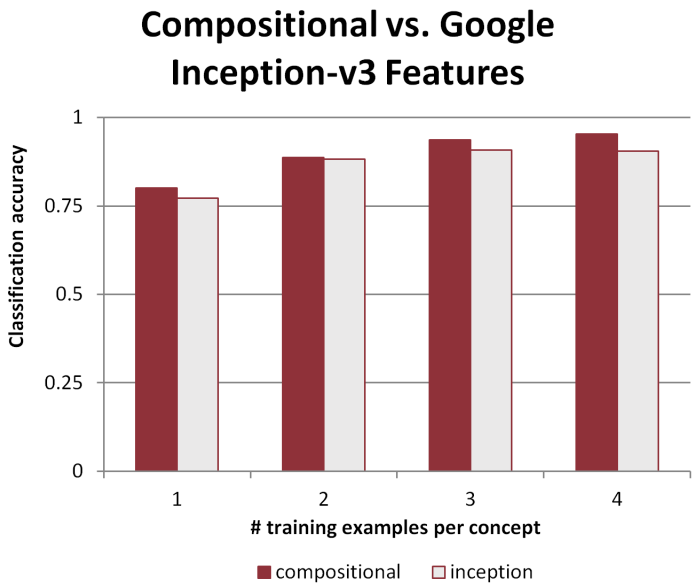

Figure 12: Results of the comparison between the features derived from my compositional representation and the features derived from Inception-v3. Accuracy rates are higher, while my approach also has 60x less features and is based on a meaningful representation.

One may conclude that my model achieves greater accuracy than Inception-v3 on my tasks with a significantly more meaningful compositional representation and with much less compute.

A closer examination of the specific failure cases even more clearly illuminates the promise of the compositional representation approach.

Throughout the experiment, the failure cases of Inception-v3 generally involved images that had similar textures or patches of pixels, but which were structurally completely different concepts. The example in Figure 13 illustrates this idea. The specific failure depicted can be explained by the deep CNN seeking out patches of the image that involve horizontal/vertical lines or resemble the pixels around the areas between adjacent rectangles. This reflects the fundamental flaw with CNNs explained in the Introduction: they may derive surprisingly high performance out of analyzing textures and local features, but they have little to no spatial understanding.

On the other hand, the failure cases of my feature set generally involved concepts that were structurally similar but had their primitive elements positioned slightly differently. The example in Figure 13 is representative of this failure mode. However, this failure is not inherent to the compositional representation approach in the way that Inception-v3's failure was inherent to the deep CNN approach. It is instead the result of a suboptimal coincidence list and feature set. The addition of more sophisticated coincidence types and features that account for relative positioning could significantly reduce or completely rule out this failure mode, enabling my compositional representation approach to further surpass the state-of-the-art Google Inception-v3 in accuracy.

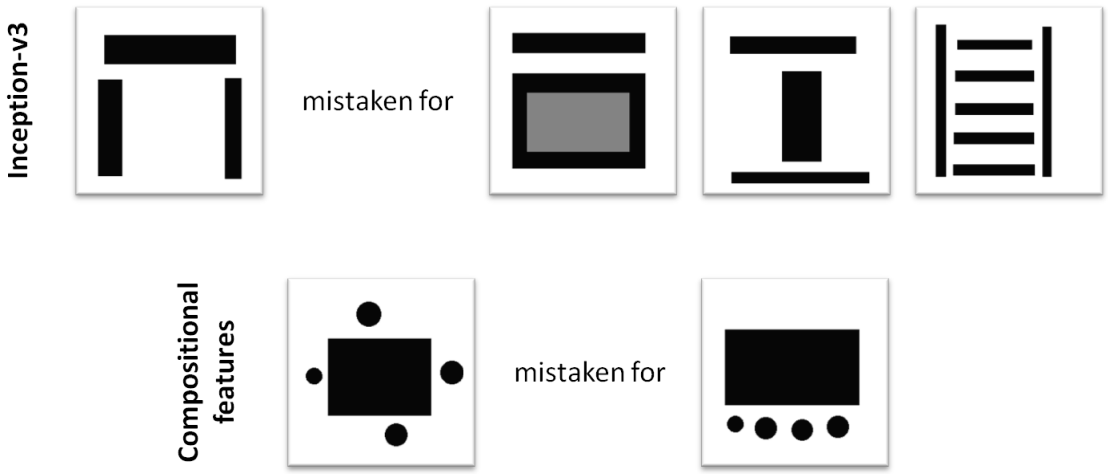

Figure 13: Examples of consistent misclassifications by the Inception-v3 features and my features. The misclassifications made by my feature set can likely be avoided with the inclusion of more coincidence types into Λ and more features into the feature set.

7 Task 5: Learning a generative model

This section describes how a probabilistic approach can be used to learn a generative model that infers concept definitions in the form of a *stochastic representation* – that is, a probability distribution over the space of compositional representations.

Let $\mathcal{S}$ be the set of all possible images, and define $\mathcal{S}_C$ as the set of all images that are instances of a particular concept C. I will now state the problem formally: given $X = \{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_N\} \subset \mathcal{S}_C$, a set of N images that are all instances of C, can one infer $p(\mathcal{I} \in \mathcal{S}_C \mid X)$, a probability distribution that tells how likely it is that any image is an instance of C conditioned on the training data?

Two important goals can be accomplished once this probability distribution is known. First, I can use it for classification, by computing the probability that a new collection of elements is an instance of the concept. Second, I can sample randomly from this distribution to generate new examples of the concept.

The desired conditional probability can be written in shorthand as $p(\mathcal{I} \mid X)$. Using the function P from Section 3, any image $\mathcal{I}$ can be represented as a collection of elements $\mathcal{E} = P(\mathcal{I})$. Then from Section 4, the corresponding APET and coincidence list can be constructed as $(\mathcal{T}, \Lambda) = R(\mathcal{E})$. Thus,

$$p(\mathcal{I} \mid X) = p\left(P^{-1}(R^{-1}(\mathcal{T}, \Lambda)) \,\middle|\, X\right).$$

It therefore suffices to find a way to compute $p(\mathcal{T}, \Lambda \mid X)$.

7.1 Bayesian inference

An expression for $p(\mathcal{T},\ \Lambda \mid X)$ can be derived using a Bayesian approach. By Bayes' Theorem,

$$p(\mathcal{T},\ \Lambda \mid X) = \frac{p(X \mid \mathcal{T},\ \Lambda)\ p(\mathcal{T},\ \Lambda)}{p(X)}.$$

The marginal probability $p(X)$ is constant as $\mathcal{T}$ and Λ vary, because X is a fixed input to this entire procedure. Hence the two important terms are the likelihood, $p(X \mid \mathcal{T},\ \Lambda)$, and the prior, $p(\mathcal{T},\ \Lambda)$.

7.1.1 Prior $p(\mathcal{T},\ \Lambda)$

Note that the space of possible coincidences is dependent on the structure of the tree. This motivates splitting the prior as $p(\mathcal{T},\ \Lambda) = p(\mathcal{T})\ p(\Lambda \mid \mathcal{T})$. I can now discuss $p(\mathcal{T})$ and $p(\Lambda \mid \mathcal{T})$ separately.

Because $\mathcal{T}$ can consist of any number of elements of multiple types, $\mathcal{T}$ can be split into two terms: $\mathcal{T}^*$, the structure of the tree, and $\Theta_{\mathcal{T}}$, the numerical parameters of the tree. This yields

$$p(\mathcal{T}) = p(\mathcal{T}^*,\ \Theta_{\mathcal{T}}) = p(\mathcal{T}^*)\ p(\Theta_{\mathcal{T}} \mid \mathcal{T}^*).$$

Similarly, Λ can consist of any number of coincidences of several types, so Λ is split into two terms: Λ^*, the set of the types of coincidences, and Θ_Λ, the numerical parameters of the coincidences. This gives the result

$$p(\Lambda \mid \mathcal{T}) = p(\Lambda^*,\ \Theta_\Lambda \mid \mathcal{T}) = p(\Lambda^* \mid \mathcal{T})\ p(\Theta_\Lambda \mid \Lambda^*,\ \mathcal{T}).$$

Putting the two together yields the expression

$$p(\mathcal{T},\ \Lambda) = p(\mathcal{T}^*)\ p(\Theta_{\mathcal{T}} \mid \mathcal{T}^*)\ p(\Lambda^* \mid \mathcal{T}^*,\ \Theta_{\mathcal{T}})\ p(\Theta_\Lambda \mid \mathcal{T}^*,\ \Theta_{\mathcal{T}},\ \Lambda^*).$$

In practice, Λ^* is independent of $\Theta_{\mathcal{T}}$, and Θ_Λ is only directly conditional on Λ^*, so the expression can be simplified to

$$p(\mathcal{T},\ \Lambda) = p(\mathcal{T}^*)\ p(\Theta_{\mathcal{T}} \mid \mathcal{T}^*)\ p(\Lambda^* \mid \mathcal{T}^*)\ p(\Theta_\Lambda \mid \Lambda^*).$$

I elaborate on a practical probabilistic programming method of implementing this breakdown of the prior in Section 7.2.

7.1.2 Likelihood $p(X \mid \mathcal{T},\ \Lambda)$

Define a function K such that $K(X,\ \lambda)$ is a real number between 0 and 1 that quantifies the extent to which the images in X satisfy a single particular coincidence λ.

Then $K(X,\ \lambda)$ can be treated as a multiplier to the probability distribution $p(X)$, to obtain $p(X \mid \lambda) \propto p(X)\ K(X,\ \lambda)$.

Define $K(X, \Lambda)$ for a set of coincidences $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_m\}$ as $\Pi_{i=1}^{m} K(X,\ \lambda_i)$. Applying this result to the likelihood function yields

$$p(X \mid \mathcal{T},\ \Lambda) \propto p(X \mid \mathcal{T}) \prod_{i=1}^{m} K(X,\ \lambda_i).$$

Now to express the exact value, renormalize the probability distribution:

$$p(X \mid \mathcal{T},\ \Lambda) = \frac{p(X \mid \mathcal{T}) \prod_{i=1}^{m} K(X,\ \lambda_i)}{\int p(X \mid \mathcal{T}) \prod_{i=1}^{m} K(X,\ \lambda_i)\, dX}.$$

Computing $p(X \mid \mathcal{T})$ is its own subtask, which can be solved directly using probabilistic programming methods.

7.2 Bayesian network and probabilistic programming

The method of *probabilistic programming* enables one to write down distributions in the form of a stochastic computer program that samples from that distribution. This program is then combined with general-purpose Bayesian inference machinery (usually Markov Chain Monte Carlo (MCMC) algorithms like Metropolis-Hastings) to infer unknown random variables from known ones. Church is a probabilistic programming language based on the Lisp model of lambda calculus [19, 20].

One can implement a concrete program in a language like WebPPL that can compute the various probabilities in the mathematical machinery of Section 7.1. Figure 14 diagrams the probabilistic program's architecture as a Bayesian network. The entire network acts as a function that samples images belonging to a particular concept. In other words, it models the stochastic compositional representation that is being sought.

Each of the five circles in the Bayesian network represents a function, and the red square represents a condition to be enforced when sampling. The first step is a function that samples from the distribution of APET structures. This can be accomplished by recursively generating APETs in a stochastic manner. At each step, a choice is made with specific probabilities to either add one of the primitive elements or create a new branch in the tree. This function corresponds to the $p(\mathcal{T}^*)$ term in the expression for the prior probability.

After generating a specific APET structure $\mathcal{T}^*$, a second function can generate a corresponding set of APET parameters $\Theta_{\mathcal{T}}$. The conditioning on $\mathcal{T}^*$ is represented by an arrow from one function to the other in the Bayesian network. This function corresponds to the $p(\Theta_{\mathcal{T}} \mid \mathcal{T}^*)$ term in the prior.

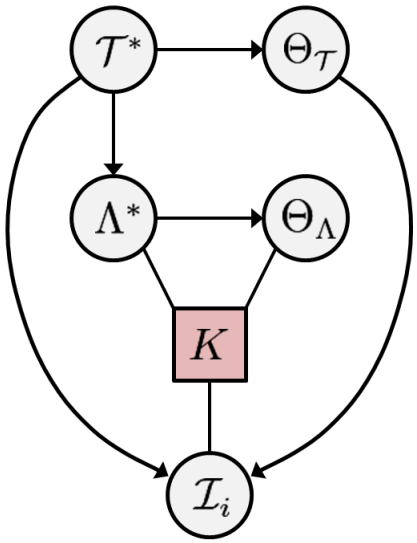

Figure 14: A Bayesian network summarizing the proposed probabilistic approach to learning stochastic compositional representations.

Next, a third function can be written to sample coincidence list structures Λ^* from a given APET structure $\mathcal{T}^*$. This function would employ a stochastic recursion approach similar to the function that samples APET structures. At each step, either a new coincidence type is added to the list, or the list is capped and the sampling process concluded. This function corresponds to the $p(\Lambda^* \mid \mathcal{T}^*)$ term in the prior.

The fourth function samples Θ_Λ from Λ^*, paralleling the second function which samples $\Theta_{\mathcal{T}}$. This function corresponds to the final term in the prior, $p(\Theta_\Lambda \mid \Lambda^*)$.

An actual image can now be generated from $\mathcal{T}^*$ and $\Theta_{\mathcal{T}}$; this is the fifth function.

The final aspect of the Bayesian net, conditioning on the coincidences, is vital. By imposing the condition of K (a *factor* à la factor graphs) on the three random variables Λ^*, Θ_Λ, and $\mathcal{I}_i$, I can ensure that the images outputted by this probabilistic program satisfy the coincidences that come along with it.

Now that I have a stochastic function that samples from the space of images, I can employ probabilistic programming's Bayesian machinery to condition this function on the training data and learn the probability distribution of images that satisfy a given concept, $p(\mathcal{I} \mid X)$. Finally, the distributions of $p(\mathcal{T})$ and $p(\Lambda)$ can be directly extracted, yielding the stochastic representation (the probability distribution of a certain concept over compositional representations).

8 Beyond

Through my implemented solutions to Tasks 1 - 4, I have demonstrated the practical effectiveness of the compositional approach to vision. In Section 7, I proposed a probabilistic programming approach

that can infer knowledge about compositional structure. So far, these methods focus primarily on object representation – a challenging problem on its own.

The broader problem is understanding scenes that consist of multiple and more complex objects. A full future solution to scene understanding may follow the schematic in Figure 15. In particular, the generalization to more primitive elements will require more sophisticated decomposition processes, potentially assisted by neural networks (following a method similar to [21], but for primitive elements as opposed to the objects themselves).

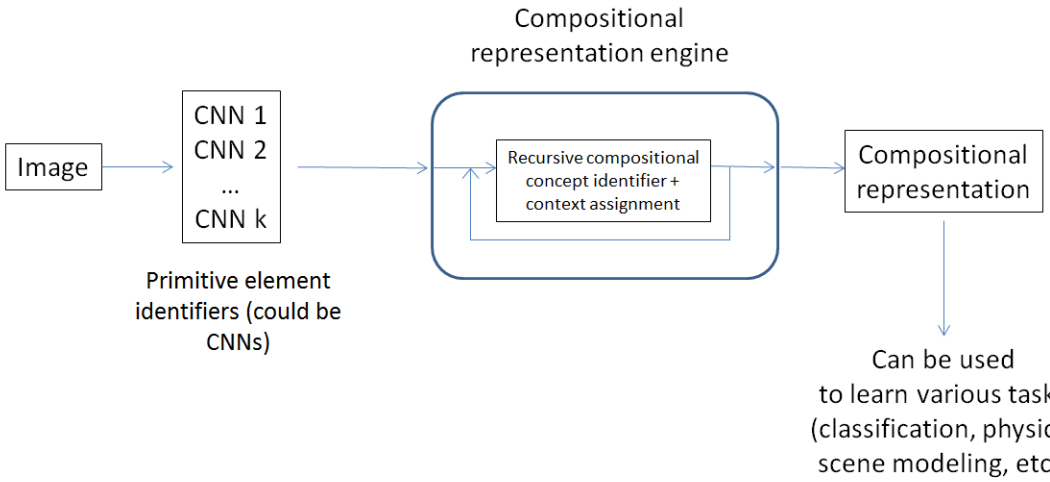

Figure 15: The future of scene understanding algorithms based on compositional representation.

9 Related Work

The most closely related work to this is Lake et al.'s work on one-shot learning for handwritten character recognition [5, 22]. The approach taken by Lake is to break down characters into primitive elements that are handwritten strokes. However, his approach is specifically tailored to the problem of character recognition and does not easily generalize to more complex objects and scenes. Additionally, my work addresses object compositionality much more directly, building my entire framework for vision upon this idea.

The high-level approach of this paper – identifying specific gaps in the state-of-the-art techniques and creating datasets and new approaches to tackle them – is a well-established model used by several influential papers in the artificial intelligence field [23, 3, 24].

The idea of the world being constructed of primitive spatial elements has been around for decades. Learning compositional representations was an early goal of artificial intelligence research, dating as far back as Patrick Winston's 1970 doctoral thesis, which learned a primitive method of distinguishing arches

among images composed of children's blocks [25]. Compositional representations were also identified as a fundamental open problem in David Marr's 1982 book *Vision* [26]. In 1987, Biederman proposed the psychological theory of "Recognition-by-Components" (RBC) – that humans perceive the world by decomposing it into primitive spatial elements – and supported it with an argument and experimental data [10]. Biederman identified 36 of these elements, which he called *geons*, and claimed that they comprise the objects in the world in an analogous manner to the widely-accepted linguistic theory of phonemes comprising human speech. Biederman also proposed that the perception of coincidences, which he called *nonaccidental properties,* plays an important role in human image recognition. Finding a practical approach to these abstract problems first posed in the 1970s and 1980s remains an open problem that has largely been left behind by the field in pursuit of deep neural networks and other unstructured statistical approaches. But as their cognitive limits become more and more apparent, so does the need for a solution to object compositionality: this is where my work comes in.

10 Contributions

The work presented in this paper can be broken down into five main contributions.

1. The ShapeWorld dataset, the reduction of the problem (learning compositional representations) from three dimensions and 36 geons to two dimensions and two primitive elements, and the identification of five milestone tasks.
2. Implementation of a system to parse images in ShapeWorld and generate compositional representations in the form of an augmented parse element tree (APET) and coincidence list (solutions to Tasks 1 and 2).
3. Implementation of a software system to write down compositional definitions of concepts that parallel the abstract nature and semantic structure of English definitions (a solution to Task 3).
4. Construction of a feature set based on compositional representations and implementation of a support vector machine learning algorithm to classify concepts based on one or a few training examples (a solution to Task 4). Also, conduction of a controlled experiment to compare the effectiveness of learning from compositional representations to learning from the state-of-the-art image recognition deep convolutional neural networks, yielding favorable results.
5. Proposal of a theoretical generalized stochastic approach to computer vision based on inferring knowledge about compositional structure (a solution to Task 5).

It is my hope that these contributions may pave the way for a new compositional approach to computer vision that is closer to human vision, faster, and safer compared to other state-of-the-art approaches.

11 Acknowledgements

I would first like to thank my mentor Kevin Ellis, who is currently a fourth-year PhD candidate in the Department of Brain and Cognitive Sciences at MIT, for sharing his technical expertise and providing high-level guidance throughout my research process. I am also grateful to my teachers Sean Campbell and Szczesny Kaminski for facilitating my independent studies on computer vision and statistical machine learning at school. I would like to thank the MIT PRIMES program for connecting me with Kevin and giving me the opportunity to conduct this research. Finally, I would like to thank my family and my school for their support and encouragement.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[2] H. Wang, B. Gao, J. Bian, F. Tian, and T. Liu. Solving verbal comprehension questions in IQ test by knowledge-powered word embedding. *CoRR*, abs/1505.07909, 2015.

[3] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. B. Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *CoRR*, abs/1612.06890, 2016.

[4] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 1998.

[5] B. Lake, R. Salakhutdinov, J. Gross, and J. Tenenbaum. One shot learning of simple visual concepts. *Proceedings of the Annual Meeting of the Cognitive Science Society*, 33, 2011.

[6] Image of a leopard-print sofa. https://rocknrollnerd.github.io/ml/2015/05/27/leopard-sofa.html.

[7] Image of a horse in a zebra costume. https://sleezybarbhorsewear.com/customers-page/.

[8] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016.

[9] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: Lessons learned from the 2015 MSCOCO image captioning challenge. *CoRR*, abs/1609.06647, 2016.

[10] I. Biederman. Recognition-by-components: A theory of human understanding. *Psychological Review*, 94(2):115–147, 1987.

[11] T. D. Kulkarni, P. Kohli, J. B. Tenenbaum, and V. Mansinghka. Picture: A probabilistic programming language for scene perception. *Computer Vision and Pattern Recognition*, 2015.

[12] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):679–698, 1986.

[13] S. Suzuki and K. Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.

[14] D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

[15] OpenCV. Open source computer vision library. `https://github.com/opencv/opencv`, 2017.

[16] M.-K. Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, 8(2):179–187, 1962.

[17] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[18] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. *CoRR*, abs/1403.6382, 2014.

[19] N. D. Goodman, V. K. Mansinghka, D. M. Roy, K. Bonawitz, and J. B. Tenenbaum. Church: a language for generative models. *CoRR*, abs/1206.3255, 2012.

[20] V. K. Mansinghka. *Natively Probabilistic Computation*. PhD thesis, Massachusetts Institute of Technology, 2009.

[21] J. Wu, J. B. Tenenbaum, and P. Kohli. Neural scene de-rendering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[22] B. Lake, R. Salakhutdinov, and J. Tenenbaum. Concept learning as motor program induction: A large-scale empirical study. *Proceedings of the Annual Meeting of the Cognitive Science Society*, 34, 2012.

[23] F. Fleuret, T. Li, C. Dubout, E. K. Wampler, S. Yantis, and D. Gemanc. Comparing machines and humans on a visual categorization test. *Proceedings of the National Academy of Sciences of the United States of America*, 108(43):17621–17625, 2011.

[24] J. Weston, A. Bordes, S. Chopra, and T. Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *CoRR*, abs/1502.05698, 2015.

[25] P. H. Winston. *Learning structural descriptions from examples*. PhD thesis, Massachusetts Institute of Technology, 1970.

[26] D. Marr. *Vision*. MIT Press, Cambridge, Massachusetts, 1982.