A Controlled Experiment: Evolution for Learning Difficult Image Classification

Astro Teller and Manuela Veloso

Carnegie Mellon University, Pittsburgh PA 15213, USA

Abstract. The signal-to-symbol problem is the task of converting raw sensor data into a set of symbols that Artificial Intelligence systems can reason about. We have developed a method for directly learning and combining algorithms that map signals into symbols. This new method is based on evolutionary computation and imposes little burden on or bias from the humans involved. Previous papers of ours have focused on PADO, our learning architecture. We showed how it applies to the general signal-to-symbol task and in particular the impressive results it brings to natural image object recognition. The most exciting challenge this work has received is the idea that PADO's success in natural image object recognition may be due to the underlying simplicity of the problems we posed it. This challenge implicitly assumes that our approach suffers from many of the same afflictions that traditional computer vision approaches suffer in natural image object recognition. This paper responds to this challenge by designing and executing a controlled experiment specifically designed to solidify PADO's claim to success.

1 Introduction

The goal of the PADO (Parallel Algorithm Discovery and Orchestration) project is the supervised autonomous learning of signal understanding for arbitrary signal types. PADO accomplishes this goal through a new evolutionary computation architecture. In previous work, we have argued and demonstrated that PADO is already able to learn to accomplish difficult signal understanding tasks [10, 12]. In recent talks given on PADO, and through correspondence with other researchers, the idea has been expressed that because the end result of the PADO architecture is so opaque, it may be that PADO is accomplishing little and that the problems we have posed it may instead actually be quite simple [3].

This worry is analogous to the worry that, while the mechanism of Neural Networks (NNs) is easy to understand, typical NNs generated are very hard to understand, *and therefore* NNs may not really be doing anything of interest. 15 years ago this was a concern for the NN community, but it has been overcome. One of the main reasons why this worry about the opacity of learned NNs has been alleviated is that controlled experiments like *auto-encoders* effectively convinced the research community that the hidden units of the NN were in fact finding useful, compressed versions of their inputs [1, 2, 6].

While we are confident that the PADO learning architecture, through evolutionary computation, does learn non-trivial algorithms and can be applied to real

world signals, it seems that a tangent is in order. This paper is that tangent. We will show, through a controlled experiment, using manufactured images, that PADO can indeed learn abstract, "high-level" features of a signal in order to accomplish signal classification. These experiments, detailed in Sect. 4 of this paper, were specifically designed to dispel potential scepticism about PADO's power.

This paper will first give a brief description of PADO and the process involved in the creation of a PADO system: the discovery of algorithms that can be parallelly executed and the orchestration of these algorithms into a useful system. Then, two past experiments [10] will be summarized to situate this paper's controlled experiments. The heart of the paper follows, in which two controlled experiments remove doubt about "secretly simple features" available for signal classification.

The concept of this paper is simple and should be appealing. PADO has performed well in seemingly difficult object recognition tasks and this performance has excited many of our colleagues. A few of our colleagues, because the end result of the PADO architecture is so difficult to decipher, have wondered whether these tasks were in fact difficult, and therefore, whether PADO is capable of automatically learning algorithms that achieve reasonable recognition rates in truly difficult signal domains. Our response, this paper, is "Well, let's make a domain that we can all agree is not trivial to learn and see how PADO does."

2 The PADO Architecture

While we believe PADO does have merit as an original system, that is not the claim we are trying to substantiate in this paper. For the same reason that a paper on auto-encoding for NNs need explain the NN process, but need not justify the existence of NNs as a research tool, the purpose of this section is not to convince, but to explain. We will first sketch the PADO architecture and its extension of genetic programming (GP) [4]. Then, the section will detail exactly what kind of programs PADO is evolving and what access these programs have to the experimental signals in Sects. 3 and 4.

The goal of the PADO architecture is to learn to take signals as input and output correct class labels. *When there are $\mathcal{C}$ classes to choose from, PADO starts by learning $\mathcal{C}$ different "systems".* System_I is responsible for taking a signal as input and returning a confidence that class $\mathcal{I}$ is the correct label. $\text{System}_\mathcal{I}$ is built out of several programs learned by PADO. Each of these programs does exactly what the system as a whole does: it takes a signal as input and returns a confidence value that label $\mathcal{I}$ is the correct label. The reason for this seeming redundancy can be found in [10]. PADO performs object recognition by orchestrating the responses of the $\mathcal{C}$ systems. Here is a description of a simple orchestration scheme that was used in the experiments described in this paper. $\text{System}_\mathcal{I}$ is built from the $\mathcal{S}$ programs that best (based on the training results) learned to recognize the instances of class $\mathcal{I}$. The $\mathcal{S}$ responses that the $\mathcal{S}$ programs return on seeing a particular image are all weighted and their weighted average of responses

is interpreted as the confidence that System$_{\mathcal{I}}$ has that the signal in question contains an object from class $\mathcal{I}$. The responses of the C systems are weighted and combined in a similar way. Figure 1 summarizes the main functionality of PADO's evolutionary learning of signal understanding algorithms.

```
function PADO(population, signals, C, S) returns C groups of S algorithms
  inputs: population, a set of P randomly generated algorithms
          signals, a set of training signals
          C, the number of classes
          S, the number of algorithms from each class to return
  Repeat
    Loop over signals
          EvaluateFitness(population,signal_i)
    Split population into C distinct subpools of size P/C based on fitness
    Loop i from 1 to C
          MatingPool_i ← Reproduction(SubPool_i)
          NewSubPop_i ← Recombination(MatingPool_i)
    population ← ΣNewSubPop_i
  Until return requested
  return the most fit S algorithms in each of the C subpools
```

Fig. 1. A top level of view of PADO's algorithm evolution learning process.

PADO evolves programs in a PADO-specific graph structured language. At the beginning of a learning session, the main population is filled with $\mathcal{P}$ programs that have been randomly generated using a grammar for the legal syntax of the language. All programs in this language are constrained by the syntax to return a number that is interpreted as a confidence value between some minimum confidence and some maximum confidence. Crossover and mutation in PADO are more complicated than their standard forms in genetic algorithms or GP. Both the crossover and mutation operators are "SMART" operators that are co-evolved with the main population, as we describe in [9, 10].

2.1 PADO Program Representation

Figure 2 sketches the structure of a PADO program. Each program is constructed as an arbitrary directed graph of nodes. As an arbitrary directed graph of N nodes, each node can have as many as N *arcs* outgoing. These arcs indicate possible flows of control in the program. In a PADO program each node has two parts: an *action* and a *branch-decision*. Each program has a private stack and an indexed memory. All *actions* pop their inputs from this stack and push their result back onto the stack. These actions are the equivalent of GP's terminals and non-terminals. The indexed memory is effected in the usual way via READ and WRITE actions [8].

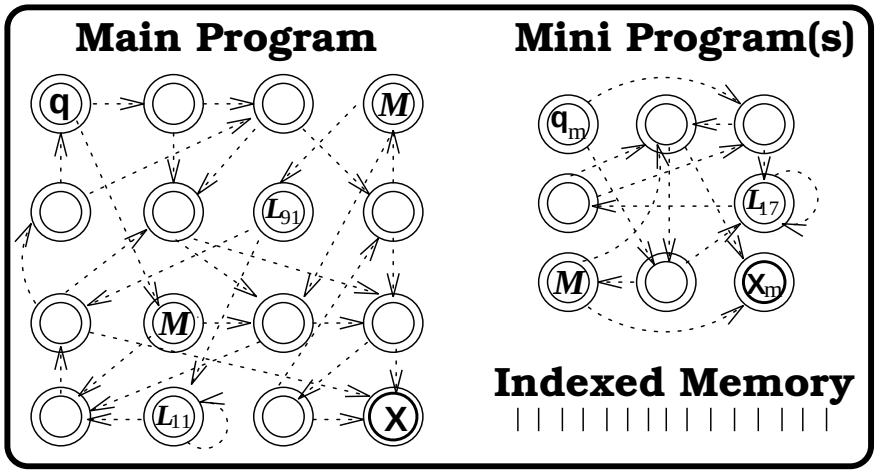

Fig. 2. This is the basic structure of a PADO program. There can be one or more Mini programs for each PADO program. Each Mini program may be referenced from the Main program, another local Mini program, or a *Library* program.

After the action at node i is executed, an arc transfers control to a new node. The branch-decision function at the current node makes this decision. Each node has its own branch-decision function that may use the top of the stack, the previous state number, the memory, and constants to pick an arc. There is a separate (i.e., individually evolved) branch-decision function for each node.

There are several special nodes as illustrated in Fig. 2. Node q is the start node. It is special in no other way than that it is always the first node to be executed when a program begins. Node X is the stop node. When this node is reached, its action is executed and then the program halts. When a program halts, its response is considered to be the current value residing in some particular memory location (e.g., response = Memory[0]). If a program halts sooner than a pre-set time threshold, it is started again at its start node (without erasing its memory or stack) to give it a chance to revise its confidence value.

Node M executes the private *Mini* program as its action. The *Mini* program associated with each *Main* program bears similarity to the concept of ADF's [5]. It may be called at any point in the *Main* program and it evolves along with the *Main* program. The *Mini* programs may recursively call themselves or the globally available *Library* programs, just like a *Main* program may. The *Library* programs (e.g., L_{91} in Fig. 2) are globally available programs that can be executed at any time and from anywhere just like the Mini programs. But unlike the Mini programs, where each Mini may be run only during the execution of the MAIN program to which it belongs, the *Library* programs are available to the entire population. The method by which these *Library* programs change can be seen in [10]. For an example of a learned PADO program, see [10, 12].

We hope that these two subsections have been detailed enough to give a flavor, if not complete details, about the PADO learning architecture. In particular, it should be clear that PADO is *not* simply a genetic algorithm (GA).

2.2 PADO Program Primitives

Here is a brief summary of the language *actions* and their effects:

Algebraic Primitives: {ADD SUB MULT DIV NOT MAX MIN}

These functions allow basic manipulation of the integers. All values are constrained to be in the range 0 to 255. For example, DIV(X,0) results in 255 and NOT(X) maps the set {1..255} to 0 and {0} to 1.

Memory Primitives: {READ WRITE}

These two functions access the memory of the program. Each program has a memory which is organized as an array of 256 integers that can take on values between 0 and 255. READ(X) returns the integer stored in position X of the memory array. WRITE(X,Y) takes the value X and writes it into position Y of the indexed memory. WRITE returns the OLD value of position Y (i.e. a WRITE is a READ with a side-effect). The memory is cleared (all positions set to zero) at the beginning of a program execution.

Branching Primitives: {IF-THEN-ELSE EITHER}

Calling these "Branching" primitives may be misleading. In both cases the primitive pops X,Y, and Z off the stack and then replaces either Y or Z (not both) depending on the value of X. For IF-THEN-ELSE the test is (X less than 0). For EITHER the test is (X less than RandomNumber) where RandomNumber varies between 0 and 255. These primitives can be used as an action or a branch-decision functions. In the former case, they have no effect on the flow of control.

Signal Primitives: {PIXEL LEAST MOST AVERAGE VARIANCE DIFFERENCE}

These are the language functions that can access the signals. In order to demonstrate PADO's power and flexibility, these same primitives were used for both image and sound data [12]. PIXEL returns the intensity value at that point in the image (or sound). The other five "signal functions" each pop the top four values off the stack. These four numbers are interpreted as (X1,Y1) and (X2,Y2) specifying a rectangle in the image. If the points specify a negative area then the opposite interpretation was taken and the function was applied to the positive area rectangle. LEAST, MOST, AVERAGE, VARIANCE, and DIFFERENCE return the respective functions applied to that region in the image. DIFFERENCE is the difference between the average values along the first and second half of the line (X1,Y1,X2,Y2).

Routine Primitives: {MINI LIBRARY[i]}

These are programs that can be called from the Main program. In addition, they may be called from each other. Because they are programs, and not simple functions, the effect they will have on the stack before completion is unknown.

3 Previous Experiments

Several publications already exist about the application of the PADO learning architecture to image understanding [9, 11, 10]. These papers address signal understanding issues such as object recognition in real video images of everyday objects and classification of high-quality sound samples. To illustrate and summarize two of those experiments, Fig. 3 shows some examples from the training and testing sets of two separate image databases. Each of these two databases contain 7 classes of images (only 2 classes from each database are shown in Fig. 3). Each 256x256 image is in 256 shades of grey.

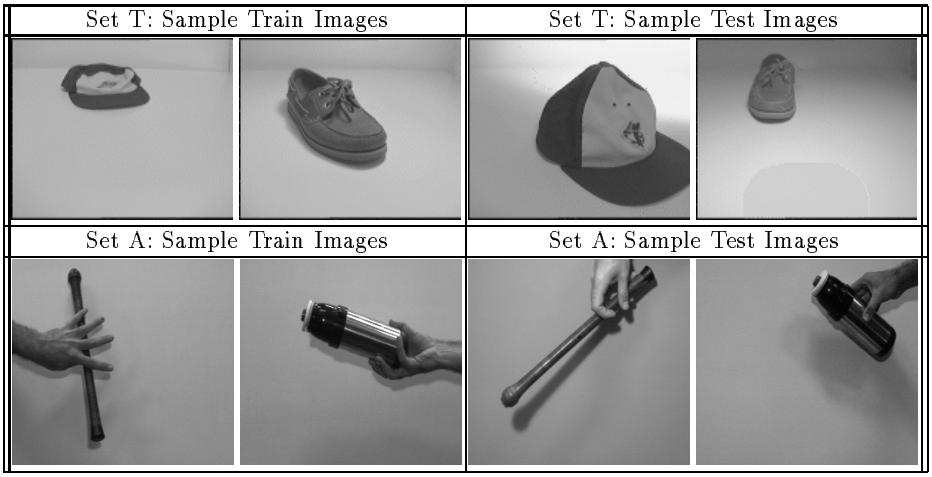

Fig. 3. 8 randomly chosen images from the experimental image database.

Figure 4 shows the ability of PADO to perform object recognition on these two sets of image data. The crux of previous papers has been "Does PADO succeed at the task of object recognition?" It is important to point out that if we constructed a system that simply guessed at the class of the image by choosing one of the seven classes at random, it would be right about 14% of the time (shown as a dotted line in Fig. 4). At generation 80, the percent of the time that PADO correctly identifies the image class is about 4 to 5 times random performance. On images as unconstrained as these images are, of objects as unfriendly as these objects are, this is a real difference.

There is little to which these results can appropriately be compared. The only other learning architecture for which there are vision results with such an unbiased, "low-level" starting point are NNs (e.g., [7]). NN cannot currently be trained on images as rich as these (65536 eight bit inputs). Even if NNs could be trained on inputs of this size, they are notoriously bad at finding "features" that are local, non-stationary, and can not be directly "observed" through the inputs.

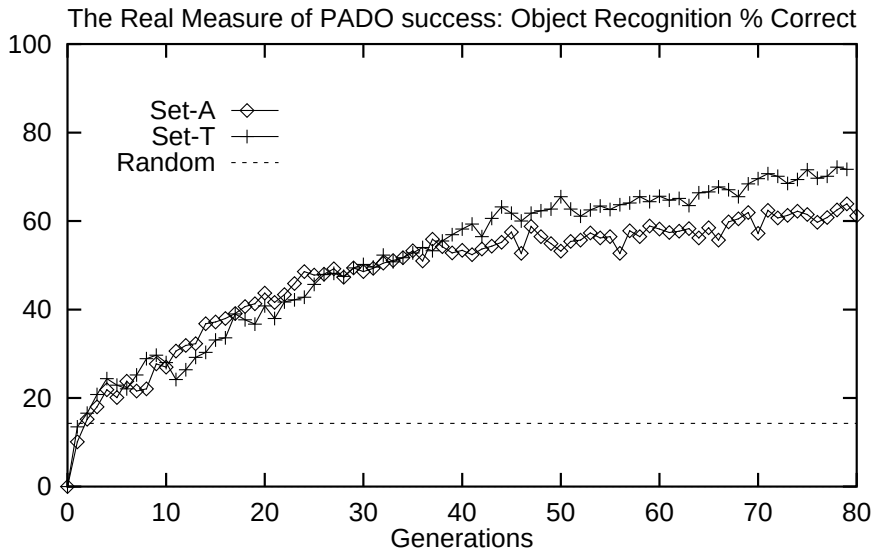

Fig. 4. PADO object recognition percentage correct on Sets A and T **test** images.

There are, of course, a large number of computer vision techniques that don't involve machine learning. Comparison was not made against any of these for two reasons. The first is that the type of images shown in Fig. 3 are less constrained (translation, rotation, foreshortening, lighting, etc.) than most of the computer vision object recognition domains of today. The second reason is that when learning is not involved, the choice of any particular set of computer vision techniques to solve a particular object recognition problem (like the discrimination of the images in Fig. 3) is effective only for that specific domain. In other words, while traditional computer vision techniques can outperform PADO in almost any specific domain, they do so at the repeated cost of a researcher's time and energy. As computer cycles become increasingly cheap relative to human effort, the attractiveness of machine learning in computer vision rises.

Because there is substantial difference between "guessing" and PADO's actual performance, a few researchers have suggested that these classification problems may be secretly "easy." In other words, how can PADO, a learning architecture with so little initial world knowledge, create a system that so reliably predicts the contents of novel images from the same classes? PADO's performance on images like those shown in Fig. 3 is very good. The question is, "Where does credit belong? With PADO (a successful learning algorithm), or with the images that have subtle but simple distinguishing characteristics?" The following section will *not* explain *how* PADO performs so well. What the following section will provide is conclusive evidence *that* PADO does succeed in difficult (from a learning perspective) image understanding domains.

4 Doing the Controlled Experiment

In order to create a convincing experiment to satisfy the doubting Thomases in the audience, we should first know what criteria we (and they) are looking for in such an experiment. It must be a "good" control to the experiments summarized in the previous section, and it must be a "hard" experiment in order to qualify as a convincing one. The criteria for such an experiment could be summarized as follows:

1. A controlled experiment should be similar to the original one in most details.
2. The classes that we require PADO to "understand" the difference between should be completely separable. That is, the task should be possible.
3. The distinction between the classes should be readily apparent to human observers to make the issues clear.
4. The distinction between the image classes should only exist in abstract (high-level) local features of the imaged objects. By abstract or high-level, we mean a feature that can not be directly measured using a signal primitive.
5. Most importantly, we should be able to *prove* the criteria above true.

4.1 Constructing a Simple Controlled Experiment

In order to satisfy criterion 1, our new experiment will be an object recognition/signal classification problem with a few classes. Let's pick 5 as the number of classes. In order to satisfy criterion 5, we will manufacture the images in the different classes. By manufacturing the images we can *guarantee* certain properties of the images. This image fabrication is *not* an attempt to create a "toy problem", but an attempt to eliminate ambiguity in the image feature space. The classification of these constructed images *is* a simple problem for hand-coded vision systems; it is *not* simple for a learning system (like PADO) that is given no information specific to the task at hand. However, the real need and goal of PADO is to attack and conquer problems that frustrate hand-coded solutions (See Sect. 3).

We chose as the 5 signal classes to manufacture: SQUARE, CIRCLE, GRID, CROSS, and TRIANGLE. To create an image instance for one of these classes:

- The object occupies between 6% and 24% (chosen randomly) of the image.
- The location of the object center in the image is chosen randomly.
- The foreground color is chosen randomly from grey levels {191...255}
- The background color is chosen randomly from grey levels {0...63}

If we take care (which we did) to adjust the relative thicknesses of these object patterns, then no "global" application of the signal primitives (LEAST, MOST, AVERAGE, VARIANCE, DIFFERENCE) to an image can help in distinguishing between classes. This is so because the images were manufactured with that criteria in mind. These images, by the construction just described, satisfy criterion 4. Figure 5 shows two examples from each class. As we can see, these objects are trivial for people to discriminate, satisfying criteria 2 and 3.

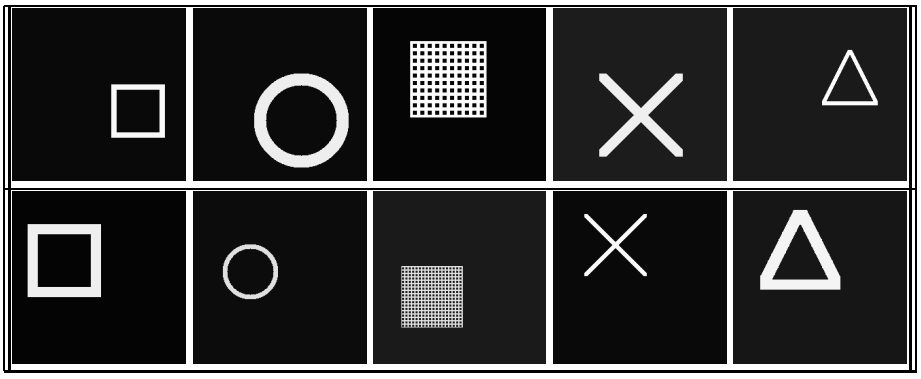

Fig. 5. 2 randomly chosen images from each of the 5 manufactured classes.

To see how difficult these images are for a *learning system* to discriminate among, let's take a look at a few pairs of classes. The only reliable distinction between the SQUARE and CIRCLE images is the *shape* of the brighter area. Both are continuous borders with background in their centers; their curvature sets them apart. Exactly the opposite is true of the SQUARE and GRID. An examination of their external borders shows them to be identical. Internal to their borders is where their distinction lies. For other examples of the difficulty of the space, notice that the bottom leg of the triangle is identical to the bottom of the square, and the top two legs of the triangle are similar to the bottom half of the Cross. Mid-level features like "dark-in-the-middle" or "straight-lines-at-right-angles" can be of some use, but still do not fully disambiguate the image classes (e.g., SQUARE vs. CIRCLE and SQUARE vs. CROSS respectively). In short, the distinctions between these classes are not only abstract, they are conjunctions of abstract features.

The curve labeled "Without Noise" in Fig. 6 shows how PADO learns to classify images its has never seen into their correct classes at a much higher rate (about 60% by generation 100) than random guessing would yield (shown as the dotted line in Fig. 6). This experiment and the experiment in the following section were performed using the same methodology as the experiments whose results on natural images are summarized in Sect. 3. 70 images were used for training during each generation, 100 images were used for orchestration, and each testing phase used 100 images. All images were chosen randomly from the distribution of images described above. Since there are over 2 billion different pictures in each class, the test images, with very high probability, had never been seen during the training phase. The lower curve ("With Noise") will be discussed in the next section.

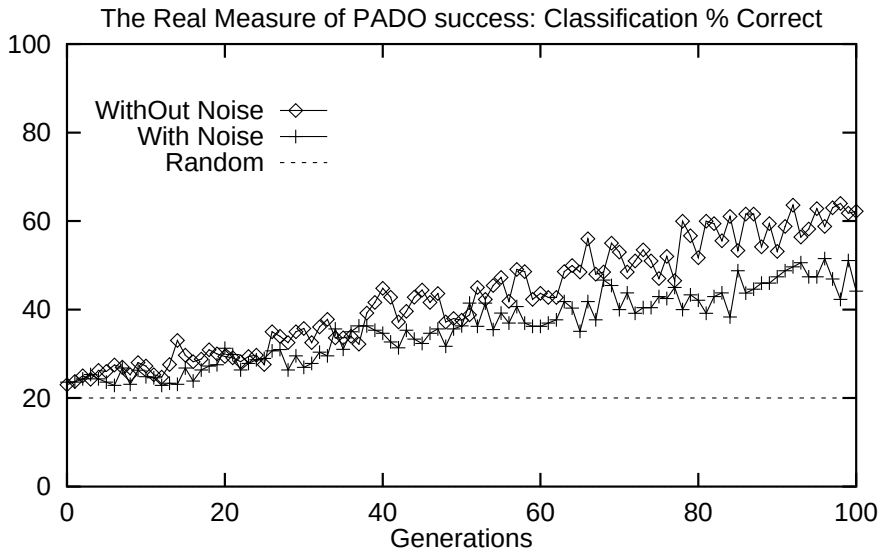

Fig. 6. PADO classification percentage correct on manufactured **test** images.

4.2 Increased Difficulty in the Controlled Experiment

The results in the previous subsection convincingly portray PADO as being able to construct a system that learns non-trivial, abstract features and uses them to successfully perform its signal classification task.

There were many possible ways to increase the difficulty of these images. Among the most traditional (from a computer vision "noise" standpoint) are: Gaussian noise, rotation, and obstruction. Gaussian noise and obstruction were chosen as our images complicators simply because they are both trivial to add to the image construction process. The noise added varies each pixel with a standard deviation of 16 shades of grey. The obstruction was added in the form of two grey bars of constant thickness in random orientations. The color of these bars varies around the central 64 shades of grey. Example images using this new image manufacturing process are shown in Fig. 7.

The curve labeled "With Noise" in Fig. 6 shows that PADO still manages to perform significantly better than random even on this harder set of 5 image classes. Given more time PADO's ability continues to increase. In order to complete a sufficient number of experiments to provide statistically significant data, generation 100 was chosen as a cut-off for the experiments. Each run took about 2.5 days of CPU time on a DECStation5000/20. Both curves in Fig. 6 are averaged over 5 runs. Each program, is given about 30 milliseconds to examine the picture and return a confidence. Because an entire PADO system orchestrates many programs in order utilize a wide variety of learned information, each classification answer was produced in about 1 second.

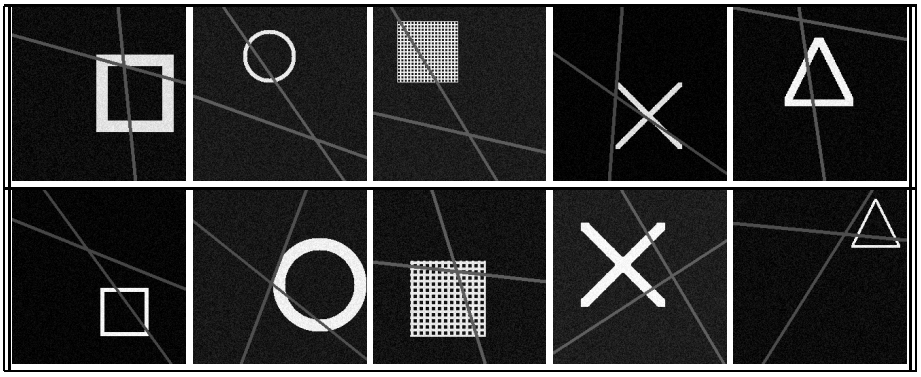

Fig. 7. 2 randomly chosen images from each of the 5 manufactured classes.

5 Discussion and Conclusions

The purpose of this paper has been to convince the reader that PADO can find and utilize local, abstract features of a signal in order to perform its signal classification task. PADO does this by evolving a set of algorithms that can be orchestrated and run in parallel to "analyze" signals. This discovery and utilization of "high-level" features of a signal was accomplished using no domain knowledge.

The goal of the PADO project is the supervised autonomous learning of signal understanding for arbitrary signal types. In previous work, we have argued and demonstrated that PADO is already able to learn to accomplish difficult signal understanding tasks. Because the PADO programs and orchestration do not evolve to be understandable by people, a few researchers have wondered whether PADO is or would be able to achieve the kind of complex understanding necessary to distinguish between everyday objects in natural images. The experiments detailed in Sect. 5 of this paper were designed to dispel this scepticism.

This paper gave a brief description of PADO and the process involved in the creation of a PADO system: the discovery of algorithms that can be parallelly executed and the orchestration of these algorithms into a useful system. Past experiments were then summarized to situate the following controlled experiments. Because the presentation of PADO as a learning model was not the focus of the paper, little direct evidence was given for evaluating of the originality of the PADO architecture.

The first controlled experiment detailed in this paper asked PADO to distinguish between 5 classes of manufactured images. The features of these images that set them apart were purposefully constrained to be high-level features such as "square" or "circle." Mid-level features like "dark-in-the-middle" or "straight-lines-at-right-angles" can be of some use, but still do not fully disambiguate the classes. On this task, where guessing a class would yield a 20% result, PADO manages a 60% classification rate by Generation 100.

The second controlled experiment increased the difficult of the task by adding confusing elements to the images. This confusion was added both through Gaussian noise at every point and through the inclusion of two grey "occluding" bars that obscure the background and foreground they cross. Even under these circumstances, PADO's ability to correctly identify images it has never seen reaches almost 2.5 times the rate that random guessing would provide.

The number of applications in need of increased autonomous signal understanding is innumerable. Most of these applications don't know in advance what form all the signals will take or what aspects of the signals will be of greatest interest. The creation of a system that can meet the needs of these applications is one of the most exciting current demands of the information age.

References

1. H. Bourlard and Y. Kamp. Autoassociation by multilayer perceptrons and singular value decomposistion. In *Biological Cybernetics.*, 1988.
2. G. Cottrell. Extracting features from faces using compression networks. In *Proceedings of the 1990 Connectionist Models Summer School.*, 1990.
3. The Symbolic Visual Learning Group. In *personal correspondence*, 1995.
4. John Koza. *Genetic Programming.* MIT Press, 1992.
5. John Koza. *Genetic Programming II.* MIT Press, 1994.
6. Mark Kramer. Nonlinear principal components analysis using autoassociative neural networks. In *AIChe Journal.*, volume 37:2, 1991.
7. Dean Pomerleau. *Neural Network Perception for Mobile Robot Guidance.* PhD thesis, Carnegie Mellon University School of Computer Science, 1992.
8. Astro Teller. The evolution of mental models. In Jr. Kenneth E. Kinnear, editor, *Advances In Genetic Programming*, pages 199–220. MIT Press, 1994.
9. Astro Teller. Evolving programmers: The co-evolution of intelligent recombination operators. In K. Kinnear and P. Angeline, editors, *Advances in Genetic Programming II*, 1995. Submitted for review.
10. Astro Teller and Manuela Veloso. PADO: A new learning architecture for object recognition. In Katsushi Ikeuchi and Manuela Veloso, editors, *Symbolic Visual Learning.* Oxford University Press, 1995.
11. Astro Teller and Manuela Veloso. PADO: Learning tree structured algorithms for orchestration into an object recognition system. Technical Report CMU-CS-95-101, Department of Computer Science, Carnegie Mellon Unversity, 1995.
12. Astro Teller and Manuela Veloso. Program evolution for data mining. In Sushil Louis, editor, *The International Journal of Expert Systems. Third Quarter. Special Issue on Genetic Algorithms and Knowledge Bases.* JAI Press, 1995.

This article was processed using the LaTeX macro package with LLNCS style