IEEE Computer Vision and Pattern Recognition Workshop on Advancing Computer Vision with Human in the Loop (ACVHL), June 2010

The HPU

James Davis[1], Joan Arderiu[1], Henry Lin[1], Zeb Nevins[1], Sebastian Schuon[2], Orazio Gallo[1], Ming-Hsuan Yang[3]
UC Santa Cruz[1], Stylight[2], UC Merced[3]

Abstract

Computer-mediated, human micro-labor markets have so far been treated as novelty services good for cheaply labeling training data and easy user studies.

This paper's primary contribution is conceptual, the claim that these markets can be characterized as Human co-Processing Units (HPU), and represent a first class computational platform. In the same way that Graphics Processing Units (GPU) represent a change in architecture from CPU based computation, HPU based computation is different, and deserves careful characterization and study.

We demonstrate the value of this claim by showing that simplistic HPU computation can be more accurate than complex CPU based algorithms on some important computer vision tasks. We also argue that HPU computation can be cheaper than state-of-the-art CPU based computation. Finally we give some examples of characterizing the HPU as an architectural platform.

1. Introduction

This paper explores the idea that humans can be used as a processor for certain tasks, in the same way that CPUs and GPUs are now used. Rather than thinking of humans as the primary director of computation, with computers as their subordinate tools, we explicitly advocate treating these computational platforms equally and characterizing the performance of Human Processing Units (HPUs). We expect to find that on some tasks HPUs outperform CPUs, and on some tasks the reverse is true, as cartooned in figure 1. The most efficient systems are likely to be joint systems that make use of each type of processor on the subroutines for which it is most appropriate. Importantly, we believe that HPUs are likely to have a long term impact on the way real-world applications that require "computer vision" are developed. This paper introduces the ideas, and provides some initial experiments.

The spread of networked computing has given rise to HPUs as a viable computational platform. Tools like Amazon Mechanical Turk allow small jobs to be micro-outsourced to humans for completion. The intermediation that the network provides allows the requester to abstract the complexities of who is accomplishing the task, and from where. Nearly all current use of micro-outsourcing is similar to the traditional way we might use an employed assistant in our office, to outsource from human to human. This proposal explicitly suggests we should quantify performance, treat this as a new computational platform, and build real systems which make use of HPU co-processors for certain tasks which are too computationally expensive, or insufficiently robust when computed on CPUs.

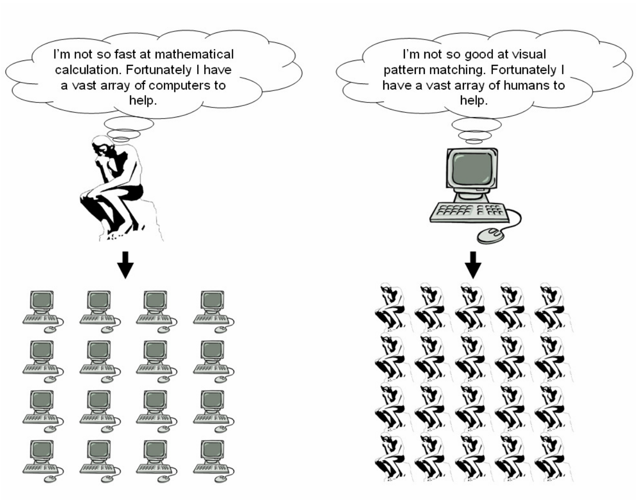

I'm not so fast at mathematical calculation. Fortunately I have a vast array of computers to help.
I'm not so good at visual pattern matching. Fortunately I have a vast array of humans to help.

Figure 1: We usually think of machines as computational tools to help humans perform better. This paper argues that humans are also computational tools to help machines perform better.

A survey of other papers using micro-labor for computer vision reveals two dominant frameworks in which the work is currently cast. The first is that *entertainment is a useful incentive structure* for getting work done [1,20,21,22,23,24,25]. The second is that micro labor is suitable for *research related* pre and post processing tasks such as labeled training data and user studies [2,13,14,15,16,17,19]. This paper argues explicitly that we should be thinking of HPU computation as a tool for getting real work done for real customers, not just as a research expediency. Very few other papers take this framing of treating HPUs as a first class computational device and asking how we can transform the tool into one that can take advantage of the full repertoire of computer science knowledge. The authors are aware of only one (not yet published) other article [7].

Broad computer science goals: Transforming HPUs into first class compute devices will require characterizing their performance. What are their strengths and weaknesses? How fast can they compute? How accurate are the results? What is the cost for a given throughput and accuracy? Is there an equivalent benchmarking so that we can give SPEC ratings to HPUs? Are their variations in performance based on the socio-economic background of the person doing the computing? How might we aggregate multiple HPUs to increase performance or accuracy?

Joint HPU/CPU computational systems are very likely to be the most efficient way to engineer many desirable tools. Computer science has focused nearly exclusively on engineering CPU based solutions. This proposal advocates that it is time to promote HPUs to the status of first class computational devices.

The research direction outlined above will ultimately have an impact on computer science generally. However, focusing attention on a single sub-area of computer science will likely do more in the immediate future to showcase the potential of HPUs.

Machine vision is a research domain in which there are a number of “unsolved” problems requiring visual pattern matching. Humans are extremely good at these tasks, and CPU based solutions tend to lack robustness and be computationally heavy. Thus this is a promising area in which to start.

In this paper: This paper is focused around three topics. First we argue philosophically that we *should* use HPUs. This framework of computation leads both to better efficiency and to new capabilities. Second we present experiments that focus on showing that HPUs outperform CPUs on a variety of machine vision tasks, using datasets and metrics already used to evaluate existing machine vision algorithms. This fact is known to researchers investigating human in the loop computer vision, but is not widely understood in the wider research community so is included for completeness. Lastly, we perform some simple performance characterization on the HPU platform to give a flavor of what it might mean to evaluate it carefully.

We wish to emphasize that our experiments are neither exhaustive nor perfect. They are meant to convince the reader that HPU/CPU comparison is a meaningful and valuable undertaking.

The primary contribution of this paper is advancing the concept of “HPU” as a first class computational architecture and as a useful framework in which we should be casting our work in this field.

2. Related Work

Historical relation between people and machines: In the early days of computation it was assumed that people did the real work. Indeed in the terminology of the period “computers” were employees. Figure 2 shows “computers” at NASA in 1949. These computers performed repetitive, well defined tasks, just as computers do today. It turned out that CPU based computation was better/faster/cheaper for some tasks, notably mathematical calculation, and so it replaced the HPU based computation of the day.

This paper argues that modern computation is *limited* by assuming all functions should be done on a CPU, and that some functions are better computed on HPUs. We seek to systematically explore the relationship between the two platforms from a modern computer science perspective.

HCI: The whole field of HCI is concerned with how humans and machines work together in joint systems. A single example related to computer vision is human assisted image segmentation [5, 6, 26]. Building better computer control interfaces enables humans to perform better. This paper is in some sense concerned with the reverse, the fact that building better human control interfaces will allow computers to perform better.

Outsourcing: The business phenomenon of outsourcing transfers tasks performed by one set of people to another set of people. There are whole fields of literature dedicated to understanding and quantifying human organizational behavior and performance. As normally conceptualized this doesn’t involve computers at all. However, computer mediated outsourcing, such as that from elance.com, odesk.com, and rentacoder.com, has brought computers into the equation. Extremely important to this paper is the rise of *micro-outsourcing* marketplaces, such as Amazon Mechanical Turk, CrowdFlower, and txteagle.com, that frequently contain very small, well defined tasks. These

Figure 2: HPU based NASA Computers at work in 1949 Photo-E49-54 [11].

tasks approach the level of simplicity that they can usefully be thought of as computational sub-routines. The online (computer accessible) nature of these sites, as well as change in scale and specificity opens the door for outsourcing, not from person to person, but from *computer to person*. Exploring this new form of outsourcing is the focus of this paper.

Human computation via games: It is possible to crowdsource labor using the human desire for entertainment, by hiding tasks inside games that people play for fun [25, 23]. Examples include providing labels and descriptions for images [24], locating objects within images [22], providing commonsense facts [21], and predicting protein folding [1]. All of these are fascinating examples of HPU computation. This paper seeks to extend the above work by hypothesizing that many sub-routines can not be easily encoded in a game, are nevertheless suitable for HPU computation, and that computer science generally will be transformed if HPUs are brought on par with CPUs as a well defined target platform.

Labeling training data: The use of humans to hand label "ground truth" training data sets is well established. This is boring work and traditionally limited by the researchers patience. With the rise of micro-outsourcing, this sort of work can simply be put online, allowing better and larger training datasets to be acquired [17, 2, 16, 14]. Indeed some researchers have started to characterize the properties of this computational platform [15, 8] as well as build better labeling algorithms on top of it [13, 19].

This paper is directly inspired by this work, but argues that micro-outsourcing has far greater potential than gathering training data for a few thousand academic researchers. It represents a new computational architecture which could conceivably run live code in production systems, directly influencing millions of peoples lives.

3. Why use an HPU?

HPUs can be cheaper than CPUs in theory: In a production environment the appropriate implementation architecture for a given task is the one which provides the highest usefulness at the lowest cost.

For the purpose of illustration, let us consider the task of tracking the location of a vehicle in a video sequence in a traffic monitoring application. The computer vision literature is full of CPU based algorithms one might use to perform this tracking. We could specify the accuracy and robustness required, and then simply choose the algorithm with lowest computational cost.

CPU based computation typically requires anywhere from 5 milliseconds to 1 hour to compute tracking for a single frame, depending on algorithm complexity. Let us suppose we need a relatively robust midrange algorithm that requires 1 minute per frame to compute. Amazon EC2 Elastic Compute Cloud sells CPU cycles on a mid-range machine for $0.40 per hour. This is roughly equivalent to the fully loaded cost of purchasing and maintaining private compute equipment, and equates to 150 labeled frames per dollar of computational cost.

This tracking subroutine could alternatively be computed by specifying it in Amazon's equivalent elastic cloud of HPUs, Mechanical Turk. Behind the scenes, a human somewhere would click on the desired object's location in the video frame. Human labeling has a computational cost related to the payments made to humans for their labor. This cost varies with task, but numbers in the range of 333-3500 labels per dollar were measured in one study [15].

Comparing performance, we find that for this particular example, HPU computation has a *lower cost* than does traditional CPU computation for the same task.

Of course we may invent new CPU algorithms in the future that change the tradeoff, or CPU time may get cheaper, or we may also invent new HPU algorithms. The primary point is that HPUs *can* be cheaper than CPUs in some settings.

HPUs can be cheaper than CPUs in practice: The color labeling task described later in this paper comes from a real company, and serves as an example. They originally implemented this module using 4 months of programmer time and ran it using 20% of a CPU that costs $180/month to maintain. Thus the total cost of the CPU algorithm was $20,000+$36/month. While working with us they implemented an HPU version of the algorithm, taking 1 month and requiring $675/month in HPU costs (3000 images per day * 0.25 cents per image * 3 trials per image). Thus the HPU total cost was $5,000+$675/month.

The crossover point in terms of real world cost is about 2 years out, and the HPU implementation gets a functioning system 3 months sooner. It also happens to be more accurate than the CPU implementation. For a startup company faced with testing a new market opportunity in which time to market and quality of results impact the bottom line, the HPU implementation is clearly the right choice.

After the success of the above, the company has used another 1 month of programmer time to implement an HPU algorithm for classifying clothing style. They have never attempted implementing this as a CPU algorithm but estimate it would take 6 months of programmer time for roughly half the accuracy of the HPU method. The accuracy requirements were such that only the HPU algorithm was worth implementing.

Global impact on poverty: A large fraction of the planet earns $1/day. Thanks to the network revolution, a substantial fraction of these people live in places where they can in fact access the internet. The question is now becoming one of “What can people *do* with the internet?” not one of “How do we get them connected?” Poverty is a quite simple problem – people don’t have enough money. If you can figure out how to employ them reliably, at wages above their current wage, they will, by definition*, be less in poverty*.

Characterizing what functions can be effectively processed on an HPU is directly related to understanding what jobs people (HPUs) *can* do effectively, and for what salary. Is there anything useful a $1/day worker is capable of accomplishing? Or are we really targeting the $10/day global middle class? Characterizing the HPU is addressing exactly this question.

Unthinkable new opportunities: There are many products and services which could exist now, but don’t exist, because some critical computational component is not yet robustly and efficiently computable. If HPUs are shown to provide a solution for some of these components, new companies will arise, offering products we would *currently believe to be impossible* (or at least too costly to implement robustly).

Completely new applications are possible which would be impossible to consider using current CPU based algorithms, because they simply could not possibly be made sufficiently robust. Consider a diet aid application running on a smartphone. Every time you eat something, you take a picture of it, and the application computes the calories and other nutritional information, keeping statistics for the user. Would current CPU based object recognition applications be able to tell which food I’m eating? Doubtful. Would HPU based algorithms do better? Probably not perfect, but perhaps well enough we can at least imagine the service.

4. Experiments – HPU vs CPU

The experiments in this section are meant to establish that it is meaningful to directly compare CPU and HPU performance on standard computer vision tasks and that in some instances HPU algorithms will outperform CPU.

Comparisons of HPU/CPU accuracy include bar code reading, color labeling, text summarization, and gender classification. In all cases we find that simple HPU algorithms are competitive with published CPU based algorithms. All HPU experiments in this paper were performed using Amazon’s Mechanical Turk.

4.1. Accuracy HPU vs CPU: Barcodes

Since their first commercial use in 1966 barcodes have become the de facto standard for processing and handling goods. Their design was optimized to maximize the accuracy of reading with laser scanners. The introduction and widespread adoption of cell phones with digital cameras, however, has recently motivated researchers to develop algorithms to recognize barcodes from blurry and low-grade cell phone images.

Current state-of-the-art algorithms are capable of dealing with some degree of blur and with a number of artifacts such as image compression [4, 18].

We use datasets provided by prior authors and compare HPU accuracy to CPU methods for both Easy and Hard datasets, with examples shown in figure 3.

Our simple HPU algorithm runs the following “code”:

```
return_value = HPU(image, “For the image
below, please type in the numbers you see
below the barcode”);
```

HPU computation is not deterministic. Thus, in contrast to CPU computation, it is frequently easy to improve performance simply by making multiple calls to the same function and aggregating the results. In this example, we iterate 6 times over the call to the HPU. This aggregation can be done on the CPU, leading to HPU/CPU hybrid algorithms. We implement a simple aggregator that rejects answers with the wrong number of digits or which fail barcode checksum, and then uses voting to determine which of several answers is correct.

Figure 4 gives a comparison of the percentage of barcodes detected accurately by each method. Note that the HPU method is comparable to the best CPU method we tested, and that the HPU/CPU joint method

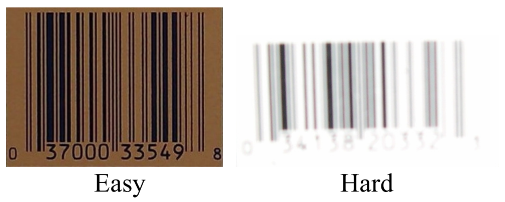


Figure 3: Examples of barcode images found in our Easy and Hard datasets. Note that the Hard image has significant blurring effects.

Barcode Recognition Accuracy: HPU and CPU methods

Method	Easy (%)	Hard (%)
HPU/CPU	100%	83%
HPU	92%	60%
CPU [Gallo09]	98%	54%
CPU [Tekin09]	95%	6%
CPU DataSymbol	0%	0%
CPU DTK	98%	3%
CPU OCR	59%	0%

Figure 4: A comparison of a variety of CPU and HPU based methods for determining barcode values on both Easy and Hard datasets. Note that the joint HPU/CPU method outperforms either HPU or CPU based computation alone. (Many comparison numbers from [Gallo09])

outperforms all others. Also note that the HPU method is really an OCR method rather than a barcode method. Thus we included a commercial CPU based OCR reader in our tests, and unsurprisingly found it to be less accurate than the CPU based barcode readers.

4.2. Accuracy HPU vs. CPU: Text summarization

Pattern recognition on textual data frequently uses the same underlying machine learning algorithms as pattern recognition in images. We compared the accuracy of a simple HPU algorithm against a baseline text classifying algorithm that might appear in an introductory class [12].

The task is to guess how many stars a product review should be assigned, based on the text of the review. We drew our sample of 60 textual paragraphs from yelp.com, and since users of that site provide star ratings, we had ground truth in the test set. Our HPU algorithm consists of uniform averaging of the results from multiple query iterations.

To build a joint HPU/CPU algorithm, we treat each human agent (not each iteration, but each actual human) as a weak classifier and train an expert system update scheme to weight the classifiers according to deviation from mean prediction of all weak classifiers in a batch setting [3].

Figure 5 compares algorithms using percentage of exactly correct star ratings. We see that the naive HPU algorithm does not perform as well as the CPU algorithm regardless of the number of iterations, but the joint algorithm performs better than either alone after only a few iterations of HPU data.

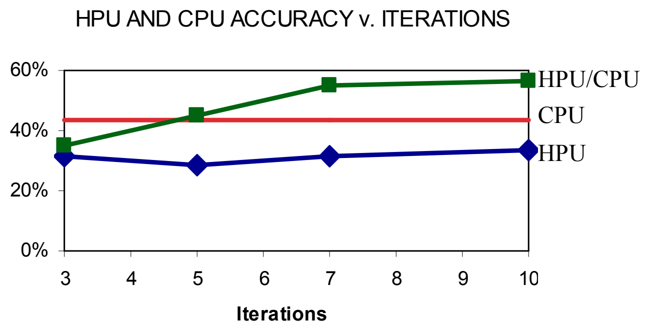


Figure 5: A comparison of accuracy on a text understanding task. The plot compares a CPU, HPU and joint HPU/CPU methods. Note that methods making use of HPUs can sometimes benefit from multiple iterations. In this case the HPU only method does not benefit, and the joint HPU/CPU method does.

We also compared algorithms using all available iterations and an L2 norm based on distance to the correct star rating. This resulted in CPU 1.6, HPU 1.1, and HPU/CPU 0.5, where lower numbers imply less distance and thus more accurate results. Again the HPU/CPU algorithm outperforms either alone.

4.3. Accuracy HPU vs. CPU: Color labeling

Color labeling is the kind of task which should be easy for machine vision algorithms. We suspect that the state of the art in academic publication would score near 100%. Thus in this case we looked at a real world implementation and dataset drawn from live code and optimized by experienced engineers (but not PhD level researchers) who care about getting the functionality right.

The task is to identify which color category an article of clothing belongs in. The CPU algorithm uses color histograms, with various outlier rejection heuristics to determine shirt color. We chose this task since we knew pictures, stripes and other irregularities to occasionally foil the algorithm.

Our initial dataset was composed of 5235 t-shirts. We split the dataset into Easy and Hard datasets based on polling of 3 people. If they agreed on the color, we called this Easy (3309 shirts), if they all 3 disagreed, we called this Hard (182 shirts). We labeled the Hard data sets with “ground-truth” based on our own “expert” assessment (an error prone task since they were usually multi-colored). Shirts with a 2-1 split were discarded from this study.

Our first question was simply whether our CPU based algorithm performed well on the Easy set. We were surprised to find only a 53% agreement between the CPU method and the color perceived by our human judges. The Hard dataset was of course worse with only 23% of the CPU algorithm’s results matching “ground truth”.

We employed an HPU/CPU algorithm for comparison. The HPU portion of the code is shown in figure 6. The CPU was used to aggregate results from multiple iterations using simple voting.

Figure 7 shows the accuracy of our HPU/CPU method on the Hard dataset as a function of HPU iterations and compared against “ground truth”. Although we understand the difficulties of comparing HPU results to human generated “ground truth”, note that the results are significantly better than the comparison CPU algorithm. Said another way, the inter-human HPU results are better because they are more inter-correlated than are the CPU results.

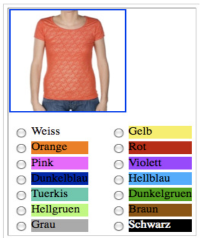


Figure 6: Sample color labeling task as specified for computation on our HPU processor. This shirt is from our Easy dataset.

4.4. Accuracy HPU vs CPU: Gender classification

We explored gender identification using a dataset from the FERET database [10]. The dataset was formed by 280 faces (males and females) in a 12 x 21 pixels format. All of those faces came with ground truth labeled by experts with access to both images and name.

Figure 8 shows our HPU implementation. We made 20 parallel requests on different faces in each HPU iteration (paying $0.01/iteration).

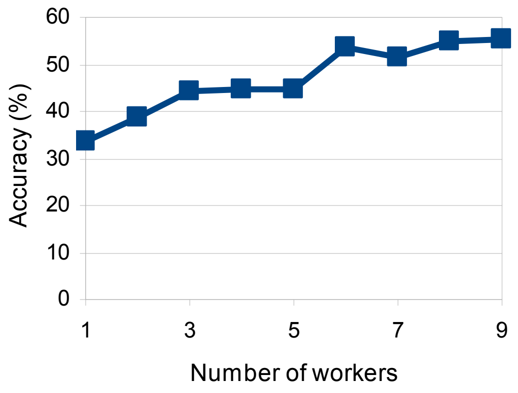
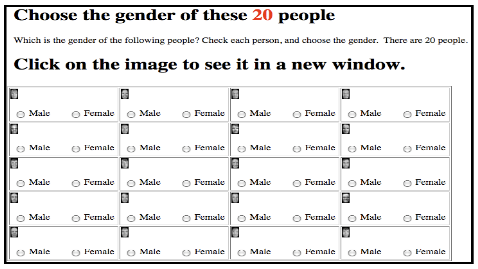


Figure 7: The accuracy of our HPU/CPU method on the Hard color labeling dataset as a function of HPU iterations and compared against "ground truth". Note that the comparison CPU algorithm had accuracy of 23% on this dataset**.**



Choose the gender of these 20 people

Which is the gender of the following people? Check each person, and choose the gender. There are 20 people.

Click on the image to see it in a new window.

○ Male ○ Female	○ Male ○ Female	○ Male ○ Female	○ Male ○ Female
○ Male ○ Female	○ Male ○ Female	○ Male ○ Female	○ Male ○ Female
○ Male ○ Female	○ Male ○ Female	○ Male ○ Female	○ Male ○ Female
○ Male ○ Female	○ Male ○ Female	○ Male ○ Female	○ Male ○ Female
○ Male ○ Female	○ Male ○ Female	○ Male ○ Female	○ Male ○ Female

Figure 8: Sample gender identification task as specified for computation on our HPU processor.

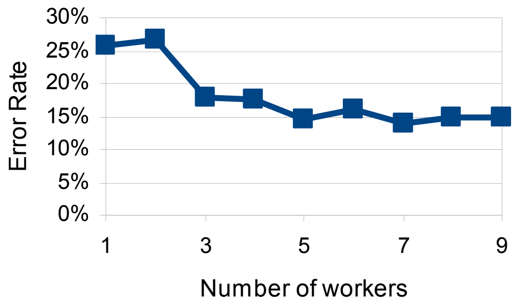


Figure 9: Error rate as a function of HPU iterations (number of workers asked), on the gender identification task.

As usual we created an HPU/CPU algorithm using aggregation on the CPU with simple voting. As shown in figure 9, additional iterations on the HPU lower the error rate.

Figure 10 compares our HPU/CPU method to a variety of CPU algorithms on the same dataset. We find that our overall accuracy is not especially good, presumably due to the tiny image size, making it difficult for humans to make judgments. Although we hypothesize that humans will perform much better on large images, the fact that modern CPU algorithms perform better in this case is an interesting finding. It both suggests that future work is needed on new HPU based gender identification algorithms, and that it is important to study which computer vision tasks are best done on HPU and which CPU.

5. Experiments – HPU Characterization

The experiments in this section are meant to show that it is meaningful to characterize the work of HPUs (humans) as if they are computational elements. We believe that this characterization will lead to the same sort of abstraction that allows software programmers to forget about the underlying hardware platform they are developing for.

To characterize the HPU architecture, we analyze the color labeling task, measuring statistics such as: computational cost vs accuracy, computational cost vs computation time, and behavior under conditions analogous to parallel processing. We find that this platform sometimes behaves similarly to CPUs, and sometimes differently.

5.1. HPU Characterization: Cost vs. Accuracy

CPU based computation is deterministic, and thus does not have a dependency between price and accuracy. If I buy a $50 CPU, it's going to provide the same answer as if I buy a $500 CPU. Does HPU based computation behave the same way? Or do you get what you pay for, and $0.01

Gender Identification Error Rate

Method	% Error
HPU/CPU	15.1%
CPU Moghaddam02	3.3%
CPU Classical RBF	7.8%
CPU Bayesian	10.6%
CPU Fisher linear discriminant	13.0%
CPU Nearest neighbor	27.1%
CPU Linear classifier	59.0%

Figure 10: A comparison of a variety of CPU and HPU based methods for gender identification. Note that our HPU/CPU algorithm performs worse than modern CPU based algorithms. (Comparison numbers from [Moghaddam02])

buys you a sloppy answer? Naturally we assume that there is a minimum payment under which answers revert to random guessing.

In order to investigate the correlation between price and accuracy we used the *shirt color* task described above. We specified to judge the color of 20 different shirts, and iterated 3 times. We presume that shirts labeled the same color in all three iterations are correct, and that other outcomes are suspect.

We repeatedly made calls to our HPU algorithm, offering different payments. Figure 11 shows the results.

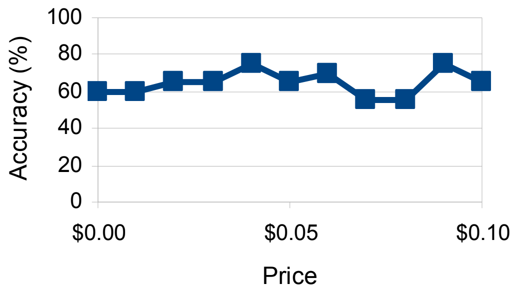


Figure 11: Comparison of HPU cost vs. accuracy on a color labeling task. Note that even at $0.00 cost (no payment made) accuracy did not suffer.

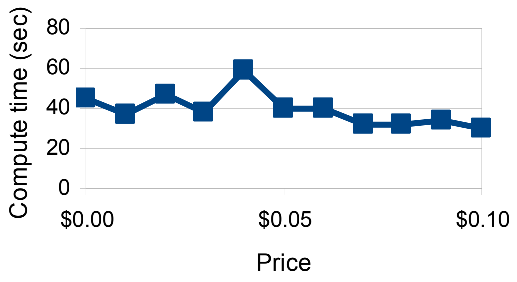


Figure 12: Comparison of cost vs. compute time. Unlike hardware CPUs in which paying more gets you a faster processor, cost does not seem to be strongly correlated with compute time.

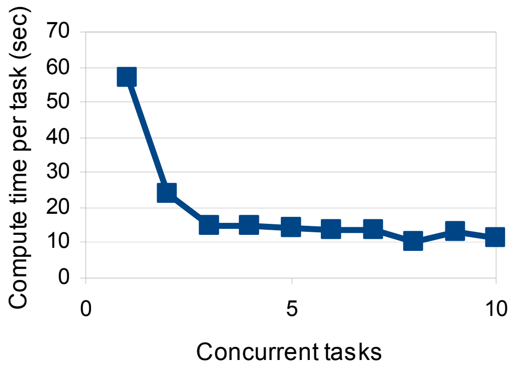


Figure 13: Plot of compute time per task versus concurrent task requests. For this particular task our HPU implementation (Amazon Mechanical Turk) behaves similarly to a multi-core platform.

Surprisingly, for this particular task, HPU computation appears to behave similarly to CPU based computation. Accuracy does *not* depend on price. Even when we offered $0.00 (no payment), the accuracy of computation remained stable.

We think it would be very dangerous to generalize from this one example, and would not want to be cited as claiming that HPU computation is price invariant in general. However we think the outcome illustrates our high level point. HPU based computation can and should be quantitatively characterized, because this characterization allows surprising counter-intuitive trends to be discovered. This in turn will allow better algorithm design.

5.2. HPU Characterization: cost vs. compute time

CPU based computation does have a correlation between price and compute time. Buying a $500 CPU will result in dramatically shorter compute time than will be achieved on a $50 CPU.

Do HPUs have the same characteristics as CPUs in which poorly paid people work slower? Or is work speed independent of pay rate? In order to find out, we plotted the compute time from the *color shirt* experiment above. Figues 12 shows the results. It appears that there is a slight correlation between price and compute time, but that it is not nearly as strong as with CPU computation. Note also, that this is compute time, the time from a task being started to task completion, not wall clock time, the time from our initial request (including waiting around for a worker to accept) until completion. We suspect that there may be a correlation between price and wall clock time, but do not have sufficient data to demonstrate this.

5.3. HPU Characterization: Parallelization

Multi-core architectures are amenable to parallel code optimization in which the programmer or compiler inserts tags which indicate “If you run these tasks at the same time, rather than sequentially, the observed computation time per task will decrease”. Is there an analogy for HPU code? If we ask the human worker to perform a bundle of tasks “in parallel”, rather than sequentially requesting one after the other, do we lower the compute time per task, or does it just linearly expand the time taken to complete?

We investigated this question using the *shirt color* task, and varying the number of shirts included in each call to the HPU. Figure 13 shows the results. To our surprise, bundling several tasks into each call dramatically improved performance. On this particular task it is as if the human is a 3-way multi-core processor, with no further performance gain obtained by bundling more tasks. This seems unlikely to be physically true, since there is good evidence that humans do process tasks sequentially. We speculate that there are substantial one time costs, possibly related to web page load time, or time to read the instructions that account for the observed behavior. Regardless of the reason, if this behavior is consistent, there are algorithm design implications.

6. Discussion

This paper argues that HPUs should be researched as a first class computational architecture. We performed simple experiments to show that HPUs can outperform CPUs on some computer vision tasks and that it is meaningful to talk about characterizing HPUs as a processing platform.

There are limitations of course, the latency involved in current micro-work marketplaces results in HPU function calls taking minutes to hours to complete. Thus HPUs are not (yet) a suitable platform for real time applications. It is possible to imagine solutions down to 1 sec of latency or so, but it is unlikely that HPUs will ever provide answers with very short times like 1ms latency. The need for understanding and carefully characterizing these limitations and tradeoffs is precisely the point of this paper.

Our future research agenda includes building toolboxes and programming interfaces so that computer vision researchers can easily compare CPU and HPU based algorithms, and industrial computer vision practitioners can easily use HPU algorithms to enable new commercial ventures that are currently impossible.

Acknowledgements

This work was funded in part by LANL ISSDM and NSF CCF-0746690.

References

[1] J. Bohannon, Gamers Unravel the Secret Life of Protein. Wired, 2009.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. IEEE Computer Vision and Pattern Recognition (CVPR) 2009

[3] Y. Freund and R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 1997.

[4] O. Gallo and R. Manduchi, Reading Challenging Barcodes with Cameras, IEEE Workshop on Applications of Computer Vision (WACV), 2009.

[5] Michael Gleicher. Image Snapping. Proceedings of SIGGRAPH Computer Graphics Annual Conference, 1995.

[6] Hyung Woo Kang , Sung Yong Shin, Enhanced lane: interactive image segmentation by incremental path map construction, Graphical Models, 64(5), September 2002

[7] Greg Little, Lydia B. Chilton, Robert C. Miller, and Max Goldman. "TurKit: Tools for iterative tasks on mechanical turk," Not Yet Published. http://glittle.org

[8] Mason, W. and Watts, D. J. 2009. “Financial incentives and the "performance of crowds"”. ACM SIGKDD Workshop on Human Computation (Paris, France, June 28 - 28, 2009).

[9] B. Moghaddam, MH Yang, Learning Gender with Support Faces, IEEE Trans. on Pattern Analysis and Machine Intelligence, 24(5), May 2002.

[10] P. J. Phillips, H. Moon, S. A. Rizvi, and P. J. Rauss, "The FERET evaluation methodology for face-recognition algorithms," IEEE Trans. on Pattern Analysis and Machine Intelligence, 22(10), pp. 1090-1104, 2000

[11] "NACA High Speed Flight Station" Computer Room, NASA Photo E49-54, http://www.dfrc.nasa.gov/gallery/photo/index.html

[12] K.M. Schneider, Techniques for Improving the Performance of Naive Bayes for Text Classification. Computational Linguistics and Intelligent Text Processing, 2005.

[13] V. Sheng, F. Provost, and P. Ipeirotis, “Get Another Label? Improving Data Quality and Data Mining Using Multiple, Noisy Labelers”, ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008.

[14] A. Sorokin and D. Forsyth, Utility Data Annotation with Amazon Mechanical Turk. First Workshop on Internet Vision at IEEE CVPR, 2008.

[15] R. Snow, B. O'Connor, D. Jurafsky, and A.Y. Ng, Cheap and Fast---but is it Good?: Evaluating Non-expert Annotations for Natural Language Tasks. EMNLP, 2008.

[16] Steinbach S., Rabaud V., Belongie S., "Soylent Grid: it's Made of People!", Workshop on Interactive Computer Vision (ICV), Rio de Janeiro, 2007.

[17] David G. Stork, The Open Mind Initiative, IEEE Expert Systems and Their Applications pp. 16-20, May/June 1999.

[18] E. Tekin and J. Coughlan, A Bayesian Algorithm for Reading 1D Barcodes. CRV, 2009.

[19] S. Vijayanarasimhan and K. Grauman, “What’s It Going to Cost You? : Predicting Effort vs. Informativeness for Multi-Label Image Annotations”, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009.

[20] L. Von Ahn and L. Dabbish, Labeling Images with a Computer Game. SIGCHI, 2004.

[21] L. Von Ahn, M. Kedia, and M. Blum, Verbosity: A Game for Collecting Common-sense Facts. SIGCHI, 2006.

[22] L. Von Ahn, R. Liu, and M. Blum, Peekaboom: A Game for Locating Objects in Images. SIGCHI, 2006.

[23] L. Von Ahn, Games with a Purpose. IEEE Computer, 2006.

[24] L. Von Ahn, S. Ginosar, M. Kedia, and M. Blum, Improving Image Search with Phetch. IEEE ICASSP, 2007.

[25] L. Von Ahn and L. Dabbish, Designing Games with Purpose. Communications of the ACM, 2008.

[26] Jue Wang , Pravin Bhat , R. Alex Colburn , Maneesh Agrawala , Michael F. Cohen, Interactive video cutout, ACM Transactions on Graphics (TOG), 24(3), July 2005