AxGames: Towards Crowdsourcing Quality Target Determination in Approximate Computing

Jongse Park Emmanuel Amaro Divya Mahajan Bradley Thwaites Hadi Esmaeilzadeh

Alternative Computing Technologies (ACT) Lab
Georgia Institute of Technology
{jspark, amaro, divya_mahajan, bthwaites}@gatech.edu hadi@cc.gatech.edu

Abstract

Approximate computing trades quality of application output for higher efficiency and performance. Approximation is useful only if its impact on application output quality is acceptable to the users. However, there is a lack of systematic solutions and studies that explore users' perspective on the effects of approximation. In this paper, we seek to provide one such solution for the developers to probe and discover the boundary of quality loss that most users will deem acceptable. We propose AxGames, a crowdsourced solution that enables developers to readily infer a statistical common ground from the general public through three entertaining games. The users engage in these games by betting on their opinion about the quality loss of the final output while the AxGames framework collects statistics about their perceptions. The framework then statistically analyzes the results to determine the acceptable levels of quality for a pair of (application, approximation technique). The three games are designed such that they effectively capture quality requirements with various tradeoffs and contexts.

To evaluate AxGames, we examine seven diverse applications that produce user perceptible outputs and cover a wide range of domains, including image processing, optical character recognition, speech to text conversion, and audio processing. We recruit 700 participants/users through Amazon's Mechanical Turk to play the games that collect statistics about their perception on different levels of quality. Subsequently, the AxGames framework uses the Clopper-Pearson exact method, which computes a binomial proportion confidence interval, to analyze the collected statistics for each level of quality. Using this analysis, AxGames can statistically project the quality level that satisfies a given percentage of users. The developers can use these statistical projections to tune the level of approximation based on the user experience. We find that the level of acceptable quality loss significantly varies across applications. For instance, to satisfy 90% of users, the level of acceptable quality loss is 2% for one application (image processing) and 26% for another (audio processing). Moreover, the pattern with which the crowd responds to approximation takes significantly different shape and form depending on the class of applications. These results confirm the necessity of solutions that systematically explore the effect of approximation on the end user experience.

ASPLOS '16, April 02 - 06, 2016, Atlanta, GA, USA
Copyright© 2016 ACM 978-1-4503-4091-5/16/04…$15.00
DOI: http://dx.doi.org/10.1145/http://dx.doi.org/10.1145/2872362.2872376

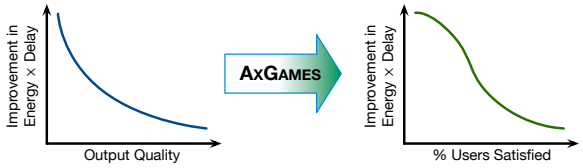


Figure 1: AxGames is a crowdsourcing solution that transforms the tradeoff between quality and energy-performance gains from approximation to the tradeoff between the gains and user satisfaction.

1. Introduction

Power efficiency is a primary concern in modern systems. Battery capacity often limits mobile devices, while power consumption and cooling imposes budgetary constraints on data centers. Moreover, traditional CMOS scaling has slowed to a point that threatens the longstanding cadence of continuously improving performance [1–3]. Meanwhile, emerging workloads must manage ever-growing datasets with high responsiveness and availability to end users. Expert analyses show that in 2011, 1.8 zettabytes (1.8 trillion gigabytes) of information was created and replicated by all sources, with individual consumers responsible for 75% [4]. By 2020, the world's data centers will be responsible for managing 50× this staggering figure [4]. This level of demand for computing raises serious concerns about the capabilities of current computing systems to match emerging trends. Apropos these confluent challenges, a growing body of recent work seeks to exploit a common property of many emerging applications: tolerance to approximate computation. These techniques relax the traditional abstraction of full accuracy in data processing, storage, and retrieval, thus trading losses in output quality for improved performance and efficiency [5–16].

While these techniques provide promising gains, the effects of quality loss on the users are not well understood, leaving approximation techniques in a position of questionable utility. The challenge is determining the level of quality loss that the large majority of users deem acceptable. Discovering this level requires end users, who are not readily available during the development phase. Even after the application is deployed, frameworks that enable users to provide feedback on the quality loss are currently unavailable. To this end, we aim to develop a framework that methodically utilizes crowdsourcing to identify the desirable application output quality without exposing the details of approximation to the users. The objective is to aid the developers in identifying the acceptable level of quality loss and enable the crowd of users to directly help in determining this level. The crowdsourcing process needs to also be engaging and enjoyable enough to retain users. To address these challenges, this paper describes AxGames, a game-based crowdsourcing framework that statistically projects *user-driven* quality targets for approx-

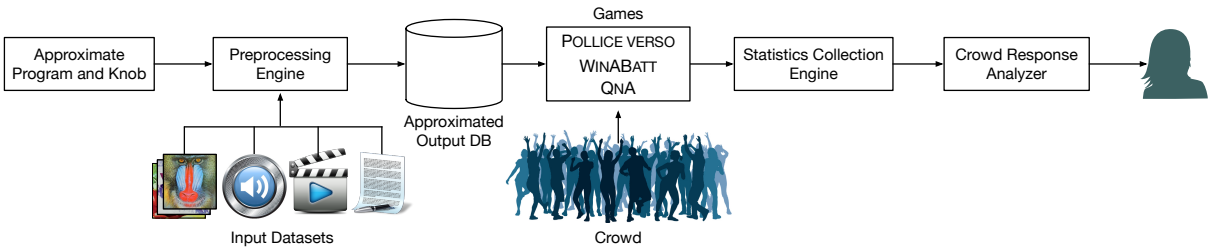


Figure 2: An overview of the AXGAMES crowdsourcing solution which determines the user-driven quality target for a given approximated application.

imate computing. As Figure 1 illustrates, AXGAMES changes the tradeoff between output quality and energy-performance gains to a tradeoff between the gains and the percentage of the users who are satisfied with the output.

AXGAMES comprises three web-based games that enable players to collectively identify the acceptable level of quality for the application in question. The games are designed to find a statistical consensus among the players on which level of quality loss is acceptable. Finding the statistical consensus is imperative in ensuring that the majority of the application users will accept the quality loss caused by the approximation technique. The first game allows the users to express their perception about the quality of the approximated output without regard to the quality-cost tradeoff. The second game enables users to choose a level of quality while considering an abstract cost tradeoff. The third game adds an element of context by asking the players to answer a multiple-choice question about the approximated output. The users are given an incentive to select the lowest output quality that allows them to answer the question. All the three games involve betting, spending, losing, and winning virtual money. The virtual money is an abstract metaphor for compute resources (time, energy, storage) that need to be spent to achieve a higher quality output. The rewarding procedure in the games is designed to place players in competition with previous players. This strategy uses the overall group to act as a check mechanism for the feedback that is provided by the players. While the participants/users play, the games collect statistics about their choices.

We use the Clopper-Pearson exact method [17] to statistically project the acceptable level of accuracy based on the statistics collected by the games. These projections provide a statistical basis for the developers to decide which degree of approximation will provide a satisfactory experience for the users. Our analysis is impartial to the benefits of approximation and independent of the approximation technique that is utilized.

To evaluate our solution, we study seven applications that produce user perceptible outputs and cover a wide range of domains including image processing, optical character recognition, speech recognition, and audio processing. Humans are naturally tolerant to approximation; hence, many approximation techniques target these domains of applications. We recruit 700 participants/users through Amazon's Mechanical Turk to play the games. The study shows that level of acceptable quality changes significantly across applications. For instance, to satisfy 90% of users, the level of acceptable quality loss is 2% for one application and 26% for another. Moreover, the study shows that generally users have higher tolerance to approximation when exposed to the tradeoff between cost and quality. The users' tolerance is even higher when they consider a context. Moreover, the pattern with which the crowd responds to approximation takes significantly different shape and form depending on the class of applications. These results suggest the necessity of solutions that systematically explore the effect of approximation on the end user experience.

By introducing the AXGAMES framework, this paper makes the following contributions:

1. **Crowdsourcing for approximate computing:** We develop a game-based crowdsourcing solution as an effective step towards enabling developers to systematically assess the effect of approximation from the user's perspective.
2. **Statistical inference:** We couple the crowdsourcing with statistical analysis to quantitatively translate raw data from the games to actionable results.
3. **Deployment:** Through deployment on Amazon's Mechanical Turk, we investigate the effectiveness of the proposed solution and show the necessity of the end user feedback by examining a diverse of real applications from different domains.

This study open a new axis, that of user experience, for the growing research in approximate computing. This work also sheds light on previously unexplored effects of approximation on the users. Moreover, it provides a development tool–rather unconventional–for the research community to better assess their innovative approximation techniques. Our tool is open source and is publicly available at http://act-lab.org/artifacts/axgames.

2. Overview

Figure 2 provides an illustration of AXGAMES's overall structure. AXGAMES is comprised of four major components: (1) approximated output database, (2) the three games namely POLLICE VERSO[1], WINABATT, and QNA; (3) the data collection engine; and (4) the crowd response analyzer. This section provides an overview of these components.

Approximated output database. The first step in using the AXGAMES solution is generating outputs of a given approximated application with varying degrees of quality loss. Note that AXGAMES is independent of the approximation technique and does not depend on how approximation is applied in the program. The developer provides a program which: (1) has an approximation knob to vary the degree of quality loss, and (2) can measure the quality loss for an approximated output[2]. For each input in the

[1] Wikipedia: "Pollice verso refers to the hand gesture or thumbs signal used by Ancient Roman crowds to pass judgment on a defeated gladiator."

[2] When developing an approximated program the developer needs to provide both the approximation knob and the quality measurement procedure [8, 10, 12, 13]. Therefore, in this regard, using AXGAMES does not require extra effort.

input dataset, the application is executed with different degrees of quality losses. A database records the approximated outputs, their degree of quality loss, and the setting of the approximation knob. We refer to this database as the approximated output database.

AxGames is designed to study a wide range of applications that generate outputs perceivable by humans through output devices such as a monitor or a speaker. Therefore, AxGames currently provides a large collection of images, audio, and text. This collection can be used by a wide variety of applications that span different domains to generate their own specific approximated output database. By playing the games, players collectively build a judgment regarding the acceptable quality for the collection of outputs. Additional information may be stored in the approximated output database. For instance, in the QnA game, players will need to answer one simple question for each approximated output. These questions are stored in the database as well. Section 3 discusses these questions and describes the three games in detail. Populating the approximated output database is performed offline to avoid unnecessary involvement of developers with the internals of gaming and crowdsourcing.

The three games. AxGames used the approximated output database as an input to its three different games. In all the games, a player is given an initial allowance of virtual money. The player's objective is to earn more money by guessing the statistical common ground among the previous players. In a sense, each player is playing with all of the past players and her guess affects the majority vote for the future players. As the crowd of gamers play the games, the players are *iteratively* converging to a statistical common ground. AxGames can then statistically infer the acceptable level of quality from the gamers' choices. Section 3 presents the details of the three games.

Statistics collection engine. As the users play, the games record the player choices and the game state in a database along with the player user IDs. AxGames uses this data to perform statistical projections about the percentage of users that deem a certain level of quality acceptable.

User response analyzer. After collecting the statistics from all the players, AxGames uses the Clopper-Pearson exact method [17] to calculate the binomial proportion confidence interval [18] for each level of quality loss. These intervals represent the percentage of users that deem a certain level of quality acceptable. Section 4 elaborates on the calculation and use of these intervals to recommend quality targets for the approximated applications.

3. The Three Games

AxGames includes three web-based games which aim to enable the crowd to iteratively converge to a statistical common ground. The players register with a unique user ID on the website to play the games without revealing personal information. Each user plays all three games independently and each game is played for 10 rounds. From the player's perspective, all three games revolve around betting, earning, spending, and losing virtual money. The score is the player's balance at the end of the game. We intentionally avoided exposing the direct relationship between virtual money and the computation cost to avoid biasing the gamers in choosing any level of quality. This relationship is a parameter in the games and can be exposed if desired. We also intended to make the games entertaining and enjoyable by using virtual money as the score and as a proxy for compute resources in two of the games. Our surveys show that 84% of the users were entertained when playing the games.

We devised the three games with different intuitions about inferring user-driven acceptable level of quality through crowdsourcing. The first game, Pollice verso, is a betting game. In each round, the player is presented with an approximated output and its corresponding precise version. The player is asked to guess whether or not the majority of other players thought that the approximated output is *good enough*. The player bets money on her guess and wins money back if the guess is correct[3]. This game aims to find a statistical common ground about the acceptability of an approximated output. However, Pollice verso does not have any notion of tradeoff between quality and cost. To include the notion of cost, we designed WinABatt which presents the player with a very low quality output and asks the player "How much would your spend to receive a better output?". The player can spend money to improve the output quality with a slider. The player wins money back depending on how close her choice of quality is to the previous players' selection. The objective of these two games is to find the statistical common ground while players judge the quality of the output in an abstract and context-insensitive manner. To provide some general context to the players, the third game QnA, presents the player with a very low quality output and asks the player a multiple choice question about that specific output. To answer the multiple-choice question, the player can improve the quality of the output by spending money with a slider. The player wins money back based on both the correctness of her answer and the closeness of her choice of quality to the previous players' selection. QnA gives incentive to the players to strike a balance between quality and cost while considering some context. The rest of this section describes these three games in further detail.

3.1 Pollice verso

As Figure 3 illustrates, Pollice verso is a betting game that gives each player an initial allowance of $500 virtual money. By keeping track of players' bets, the game aims to infer the statistical common ground for the acceptable level of quality for a given application. The game randomly selects an approximated output o, and displays o along with its precise counterpart o^*. Internally, we represent each approximated output o with the following tuple:

$$(o,q,s,n,n_{GoodEnough}) \quad (1)$$

In Equation (1), q is the output quality; s is the setting of the approximation knob that led to this quality; n is the total number of past players that have played this particular output; and $n_{GoodEnough}$ is the number of players who thought the output is good enough. The last two parameters capture the history of the previous players' choices.

After displaying the outputs, a player is asked "How much do you want to *bet* that this approximation is **Good**

[3] Pollice verso shares similarities with A/B testing [19]. However, A/B testing does not incorporate (1) games and betting (provided by all three games) (1) a sense of tradeoff (provided by WinABatt) and (2) a sense of context (provided by QnA) to the users. As the statistical results show, userss tolerance to approximation increases when the tradeoff and/or context is added to the games.

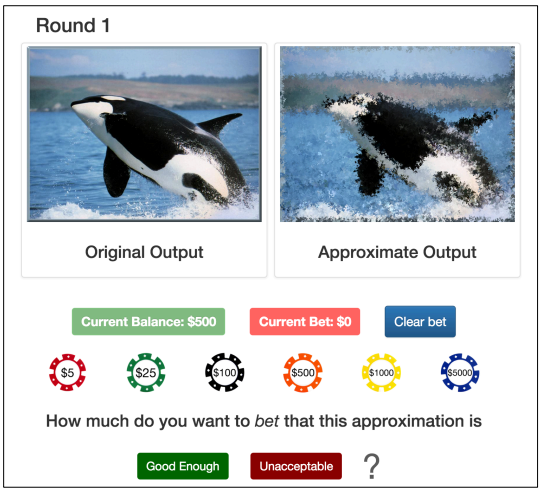


Figure 3: A round of the POLLICE VERSO game when deployed for an approximated implementation of the jpeg application.

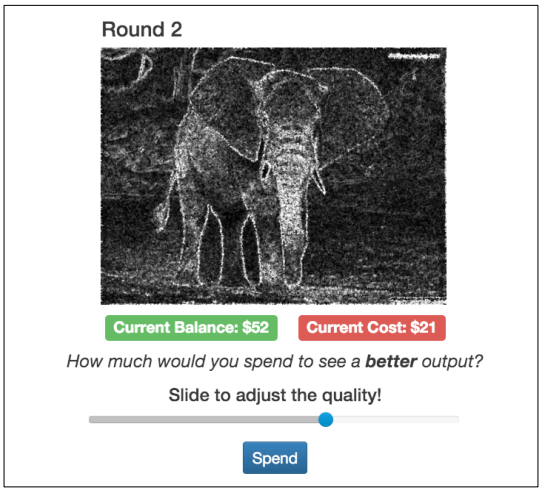


Figure 4: A round of the WINABATT game when deployed for an approximated implementation of the sobel application.

Enough/Unacceptable?" Using the gaming chips shown in Figure 3, the player chooses to bet b amount of money on her answer. The player's choice c, is a binary decision.

Rewarding procedure. The player may win money or lose the bet depending on whether the past players agree with her choice. This rewarding strategy incentivizes the players to gradually come to a statistical common ground without directly interacting with each other. As Equation (2) shows, the winnings w, is a function of the player's choice c, the amount of bet b, and the past player's choices, captured by n and $n_{GoodEnough}$. Note that the values of n and $n_{GoodEnough}$ are updated after the player receives her reward. Therefore, the player's choice affects the winnings of future players.

$$w(c,b,n,n_{GoodEnough}) = b \cdot (reward(c,n,n_{GoodEnough}) - 1) \quad (2)$$

where

$$reward(c,n,n_{GoodEnough}) = \begin{cases} 2 \cdot f(c,n,n_{GoodEnough}) & \text{if } 0 \le f \le 0.5 \\ -7.8 \cdot f(c,n,n_{GoodEnough}) + 8.9 & \text{if } 0.5 < f \le 1 \end{cases} \quad (3)$$

As Equation (2) shows, the player wins money proportional to the amount of bet b. The $reward$ function (Equation (3) and Equation (4)) defines this proportion based on whether or not the majority of previous players agree with the player's choice c. In Equation (3), the constants (2, -7.8, 8.9) are picked such that the player loses all her bet in the worst case or quadruples her bet in the best case. Moreover, if the output is controversial, the loss is low and the gain is high. An output is controversial if $n_{GoodEnough}/n \approx 0.5$. That is, almost half of the past players think the output is good enough and the other half thinks otherwise. In Equation (3), f captures the level of agreement between the player's choice with the majority vote of the previous players as presented in Equation (4).

$$f(c,n,n_{GoodEnough}) = agreement(c,n,n_{GoodEnough}) = \begin{cases} n_{GoodEnough}/n & \text{if } c = Good\ Enough \\ 1 - n_{GoodEnough}/n & \text{if } c = Unacceptable \end{cases} \quad (4)$$

To enable players to make choices primarily based on their own perception, the reward function is hidden. Additionally, they play the game with no knowledge of the majority vote.

3.2 WINABATT

POLLICE VERSO enables users to perceptively judge the quality of an approximated output without regard to the tradeoff between quality and cost. To add this notion of tradeoff, we designed WINABATT as shown in Figure 4. The player starts with \$100 of initial allowance and in each round, the game displays an approximated output at its lowest quality and asks the player: "How much would you spend to receive a **better** output?" The player is also given a slider with which she can adjust the output quality. The slider controls the quality and the cost associated with each quality level. Selecting a higher quality translates to spending more virtual money. Unlike POLLICE VERSO, the player's choices in WINABATT are no longer *yay/nay* binary decisions, and the player uses a continuous slider to choose a *level* of quality while considering its cost. If the game was naively designed, the player would always choose the lowest level of quality since it costs the least. However, in WINABATT, the player will be rewarded or penalized depending on how her choice of quality is close to the previous players. Hence, the player is also trying to guess the statistical common ground among the past players in a cost-conscious manner.

Rewarding procedure. To calculate the player's winnings and the statistical common ground, the game internally represents each output with the following tuple:

$$(\mathbb{O}, \mathbb{Q}_c, q_{MA}, n) \quad (5)$$

In Equation (5), $\mathbb{O}$ is the set of different approximated versions of an output; $\mathbb{Q}_c$ is the set of previous players' choice of quality, q_{MA} is the cumulative moving average of the past players' choice of quality; and n is the number of previous players that played the $\mathbb{O}$ set. The q_{MA} captures the statistical common ground among the past n players and is updated based on Equation (6) after the current player is rewarded.

$$q_{MA}^{(n+1)} = \frac{q_{MA}^{(n)} \cdot n + q_c}{n+1} \quad (6)$$

As shown in Equation (7), the player's winnings w, is a function of her choice of quality q_c and q_{MA}. As shown, the player's reward is deducted by her bet money b, which is the cost associated with her choice of quality q_c. This cost function

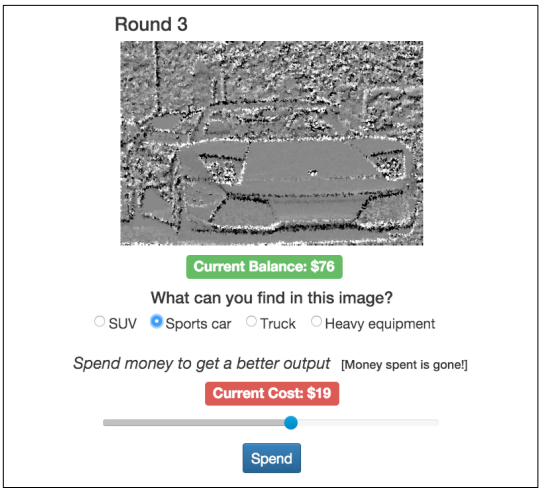


Figure 5: A round of the QNA game when deployed for an approximated implementation of the emboss application.

is linear to avoid bias towards any specific quality with \$5 for the lowest quality version (q_{min}) and \$30 for the highest quality version (q_{max}). The conditional part of Equation (7) is the reward that is determined by $f(q_c,q_{MA})$, which is presented in Equation (8) and captures how the player's choice of quality, q_c, is close to the choice of previous players' moving average, q_{MA}. The constants in Equation (7) is chosen such that the player's winnings, w, is between $-\$35$ and $+\$35$.

$$w(q_c,q_{MA}) = -b + \begin{cases} 5 \cdot f(q_c,q_{MA}) - 10 & \text{if } f \leq 10 \\ 40 = 5 \cdot 10 - 10 & \text{if } f > 10 \end{cases} \quad (7)$$

$$f(q_c,q_{MA}) = agreement(q_c,q_{MA}) = \left(\frac{|q_c - q_{MA}|}{q_{max} - q_{min}} \right)^{-1} \quad (8)$$

This rewarding procedure incentivizes the players to balance the cost and quality *while* guessing the past player's consensus.

3.3 QNA

While WINABATT provides an opportunity to the players to explore the tradeoff between quality and cost, they do so in an abstract and context-insensitive manner. To provide some context to the players, we design the QNA game. As Figure 5 illustrates, in each round, QNA displays an approximated output initially set to its lowest quality level, along with a slider, and a multiple-choice question about the output. The questions are in the form of "What can you find in this image? Sports car / SUV / Truck / Heavy equipment." The player needs to answer the question and can spend money to improve the quality using the slider. Similar to WINABATT, the initial allowance provided is \$100. In contrast to WINABATT, the slider cannot move backwards. This feature is to prevent the players from cheating, increasing the quality to answer the question, and then decrease the quality to minimize the cost. In other words, once the player improves the quality, she cannot recover the cost of seeing the higher quality output.

Rewarding procedure. The winnings are calculated based on the rewarding procedure explained for WINABATT with the exception that the player also pays a \$20 penalty for answering the question incorrectly. There is no extra reward for correct answers. The player wins money back depending on the correctness of her answer and the closeness of her choice of quality to the moving average of the previous players. QNA incentives the players to find a statistical common ground while balancing quality and cost with respect to some context about the output.

4. Statistical Analysis

As mentioned before, the games internally collect the player's choices and decisions for a series of outputs at different levels of quality. To enable the application developer to draw meaningful conclusions from this raw data, we devise a statistical framework that projects the user-driven quality target. Due to the large space of possible inputs and the diversity of users, it is practically infeasible to find a quality target that satisfies the entire population of the users for any arbitrary input. However, coupling the games with statistical analysis provides a pragmatic approach to determine the quality target that, with high confidence, satisfies the large majority of users.

4.1 Binomial Proportion Confidence Interval

We calculate the binomial proportion confidence interval [18] for each level of quality loss. Given the decisions of a *sample population* (players of the games), the binomial proportion confidence interval projects what percentage of the *statistical population* (all the users) are likely to deem a certain level of quality good enough (acceptable). After the players play the games, AXGAMES calculate this confidence interval for a range of quality losses which resulted from approximating the application-under-study. Based on the confidence interval, we can determine the level of quality loss that is highly likely to satisfy, for example, 90% of the statistical population[4] of the users.

AXGAMES leverages a commonly used method, the Clopper-Pearson exact method [17] to compute the binomial proportion confidence interval. We chose the Clopper-Pearson method as it has certain advantages over the other available options [20, 21] such as, (1) higher accuracy as the number of samples becomes relatively large, and (2) it can calculate the confidence interval even when the opinion of the sample population is very skewed towards a decision. These features are important in our setup since the games provide a relatively large number of statistical samples, and the large majority of the players are highly likely to think that 1% quality loss is almost always acceptable and similarly a 50% quality loss is almost never good enough. Moreover, the Clopper-Pearson exact method calculates a conservative confidence interval that reduces the risk of being too aggressive when it comes to approximation. The binomial confidence interval is calculated based on a set of binary decisions from the sampled population. For example, in the case of the POLLICE VERSO game, a binary decision comes directly from the player's choice on whether or not an approximated output with the quality of q is good enough. Later in this section we describe how the WINABATT's and QNA's sliders are translated to binary decisions.

[4] Statistical population is the entire pool from which a sample population is drawn. Here, the sample population are the gamers who are drawn from the entire pool of the application users. Thus, statistical population is the entire users.

To calculate the confidence interval, we first need to calculate the sampled binomial proportion for each level of quality. The sampled binomial proportion is calculated for each approximated output by computing the fraction of votes that deem a level of quality good enough to the total number of votes. This sampled binomial proportion is calculated based on the $(n_{Votes},n_{GoodEnough})^{(q)}$ pair, where n_{Votes} is the total number of decisions on outputs with the quality of q, and $n_{GoodEnough}$ is the number of decisions that deem these outputs good enough. This pair is calculated for each level of quality.

As Equation (9) shows, the Clopper-Pearson exact method computes the one-sided confidence interval of success rate, $\theta^{(q)}$, when the number of sample trials, n_{Votes}, and the number of successes among the trials, $n_{GoodEnough}$, are measured for a sample of the population.

$$\frac{1}{1+\frac{n_{Votes}-n_{GoodEnough}+1}{n_{GoodEnough}\cdot\mathbf{F}[1-\alpha;2\cdot n_{GoodEnough},2\cdot(n_{Votes}-n_{GoodEnough}+1)]}} < \theta^{(q)} \tag{9}$$

In Equation (9), $\mathbf{F}$ is the F-critical value that is calculated based on the F-distribution [22]. The *discontinuous* nature of the binomial distribution precludes any interval with exact coverage for all values of n_{Votes} and $n_{GoodEnough}$ (all possible values of the binomial proportions). However, because of the relationship between the cumulative binomial distribution and the *continuous* F-distribution, we use the common alternative form of the binomial confidence interval that provides exact coverage for all population proportions. $\mathbf{F}[1-\alpha;d_1,d_2]$ is the $(1-\alpha)$ quintile of the F-distribution with d_1 and d_2 degrees of freedom. The $(1-\alpha)\cdot 100\%$ is degree of confidence on the interval. For instance, for 95% confidence interval, α is 0.05 and for 90% confidence interval, α is 0.10. The two degrees of freedom, d_1 and d_2, decide the shape of the F-distribution based on the collected statistics, n_{Votes} and $n_{GoodEnough}$ in our case.

To understand the meaning of $\theta^{(q)}$, we discuss an example deployment of the game that resulted in n_{Votes}=60 and $n_{GoodEnough}$=56 for quality level q=97%. From Equation (9), the lower limit of the 95% confidence interval, $\theta^{(97\%)}$, is 85.4%. That is, with 95% confidence, we can project that at least 85.4% of the users will deem 97% quality level acceptable. This projection is conservative because the Clopper-Pearson exact method calculates a conservative lower bound for the confidence interval. The degree of confidence is the probability of the projection being true. The projection based on 95% confidence interval is true with probability of 0.95.

For each level of quality, AxGames projects the fraction of user population (statistical population) that deems that level of quality acceptable. Using this information, the developer can choose the level of quality that satisfies a target majority of users.

Translating a choice of quality to a set of binary decisions. Players in Pollice verso make binary decision on the quality of an approximated output. These yay/nay decisions can be directly used in the Clopper-Pearson statistical analysis. In contrast, the players in WinABatt and QnA, choose a level of quality using a slider. To be able to use the Clopper-Pearson statistical analysis, each chosen level of quality needs to be translated to a series of binary decisions. Intuitively, when a player chooses the quality level of q_c to be good enough, she implies that any level of quality higher than q_c is also good enough. Because of the rewarding procedure, the player has the incentive to choose the lowest acceptable quality. Choosing higher quality translates to a higher cost. The choice of player also implies that lower quality outputs are not good enough, otherwise, she would have chosen a lower quality to pay a lower cost. Based on this intuition, we convert one chosen level of quality, q_c, which is in the range of (q_{min},q_{max}) to $q_{max}-q_{low}+1$ binary decisions (Equation (10)).

$$decision(q)=\begin{cases}\text{“}GoodEnough\text{”} & \text{if } q\geq q_c\\ \text{“}Unacceptable\text{”} & \text{if } q<q_c\end{cases} \quad \forall q\in(q_{min},q_{max}) \tag{10}$$

By using this conversion, the same method of statistical projection can be applied to all three games. Moreover, WinABatt and QnA provide larger number of decisions per round. We did not use this conversion in Pollice verso to enlarge the number of decisions since the players do not have an opportunity to see a range of quality losses for each output. Whereas, WinABatt and QnA allows the players to see the same output at different quality levels using the slider.

5. Evaluation

To evaluate the effectiveness of the AxGames crowdsourcing framework, we deploy the three games on the web for seven different applications. We use Amazon's Mechanical Turk to recruit a large number of users to play the games. Using the collected data through the games and our statistical analysis, we measure what level of quality is acceptable for the majority of users. We also study how the acceptable level of quality varies across different applications and how the statistical analysis effectively captures these trends.

5.1 Methodology

Applications. As Table 1 shows, we examine AxGames using a wide range of applications from diverse domains that include image processing, audio processing, optical character recognition, and speech to text conversion. AxGames is not limited to these applications and can be used with other applications that produce outputs perceptible by humans. As shown in Table 1, our set of programs includes four image processing applications. The emboss application is an image filter that replaces each pixel either by a highlight or a shadow. Applying this filter to an image usually results in an image resembling a paper or metal embossing of the original image. The jpeg application implements the JPEG image compression algorithm. The sobel application is an edge detection algorithm which employs the Sobel operator. The mean is a sliding-window spatial filter that replaces the center pixel with the average (mean) of the pixel values in the window and blurs the image. Additionally, we also evaluate audio-enc, an audio compression engine that compresses WAV files and transforms them into MP3 files [23] . The quality metric for the image processing and audio compression applications is the normalized root mean square error (NRMSE) which is calculated by comparing an approximated output with its precise counter part.

We also use two applications that recognize text and speech. The ocr application is an optical character recognition program that converts raster images of written text to characters. The speech2txt application is a speech recognition engine that con-

Table 1: Applications and their quality metric.

	Description	Domain	Quality Metric
emboss	Embossing filter	Image Processing	Normalized Root Mean Square Error (NRMSE)
jpeg	Lossy compression		
mean	Blurring filter		
sobel	Edge detection		
audio-enc	Audiio encoder	Audio Processing	
ocr	Optial character recognition	Pattern Recognition	Text Similarity Ratio
speech2txt	Speech to text		

verts speech audio files to text [24]. The quality metric for these two applications that produce text output is the text similarity ratio that is computed by comparing the original application output with the approximated application output. We use an open-source implementation of the Gestalt Pattern Matching algorithm [25] to measure the text similarity ratio. This algorithm assigns higher scores to the outputs that *look right* to a human reader. The set of applications have often been used to evaluate the benefits of approximation techniques in the approximate computing literature [5, 6, 8, 13, 26, 27].

Approximation techniques. For the image processing applications, we use a variation of loop perforation [12] as the approximation technique. This technique skips computing some of the pixels and instead, copies the values from neighboring pixels. The rate at which the computation is skipped is the knob for controlling the quality. We refer to this technique as tiling. We chose tiling for image processing applications since it is a simple yet effective coarse-grained approximation technique. For the audio compression and pattern recognition applications, we use the same model used in previous fine-grained approximation works [7, 10, 11, 13, 28] that adds stochastic noise to the computation by leveraging voltage overscaling, bit-width reduction, and reducing the DRAM refresh rate. The rate and the magnitude of noise are the knobs for controlling approximation. We refer to this technique as fine-grained approximation. We use tiling and fine-grained approximation to examine AxGames, since they represent the two main categories of approximation techniques, coarse-grained and fine-grained, respectively. AxGames is general and is not limited to these techniques. This flexibility is inherent in the framework since it only needs a set of approximated outputs with varying degrees of quality loss. Naturally, if the approximation technique changes, the games need to be redeployed in order to understand the user response to the new technique.

Datasets. As part of the AxGames framework, we include the input dataset that contains 200 images, 200 audio files, 200 speech files, and 200 printed text files. We collect this dataset from open-source databases or public data archives such as ImageNet [29], Freesound [30], VoxForge [31], and New York Times TimeMachine [32]. The applications can use this collection to generate the approximated output database which is used in the three games. AxGames also assigns one multiple-choice question to each data element. These questions are used during the QnA game to provide some context to the gamers when they trade quality for cost. The questions are part of the framework and do not need to be regenerated for other applications that produce similar outputs. Additionally, the approximated output database is generated offline by running the application-under-study with different settings of the approximation knob. The AxGames's requirement is to generate each output with varying degrees of quality loss. For our experiments, we generate 51 versions of each output, with quality loss ranging from 0% to 50%, and a step size of 1%.

Game deployment. We separately deploy the three games on the web for each of the seven applications. In each round, each game randomly selects an output from the approximated output database. In the Pollice verso game, the players bet and vote on whether the quality of the approximated output is good enough. Thus, the game randomly picks a quality loss for the displayed output. The following ten quality losses are used for the selected approximated output: 1%, 3%, 5%, 7%, 10%, 15%, 20%, 30%, 40%, and 50%. The other two games use all 51 quality levels which are generated for each approximated output and allow the player to pick any of them. We have made AxGames open source in our artifact portal (http://act-lab.org/artifacts/axgames). The deployed games, which are used for evaluations, are also available through the same portal.

Crowdsourcing through Amazon's Mechanical Turk. To collect a reasonable amount of statistics from a diverse set of users, we leverage a crowdsourcing Internet marketplace, Amazon's Mechanical Turk [33]. The games for each application is played by 100 individual Mechanical Turk workers (turkers)[5]. For the seven applications, 700 turkers contributed to our study. Each turker plays the games once. We do not allow a worker to play the games more than once to avoid bias in statistics from a few heavy gamers. Each game has 10 rounds, our 100 turkers make a total of 1,000 binary decisions in Pollice verso for each individual application. Therefore, each of the 10 discrete levels of quality in Pollice verso receives 100 votes for each application. As discussed in Section 4, in each round of WinABatt and QnA, a player's choice of quality on the slider is translated to 51 binary decisions, which correspond to one of the 51 quality levels of an approximated output. Hence, for each of these two games the turkers make a total of 51,000 (= 100 players $\times$ 10 rounds $\times$ 51 binary decisions per round) binary decisions for each individual application . Thus, each of the 51 levels of quality receive 1000 votes in WinABatt and QnA for each application. This amount of information provides the grounds for making high confidence statistical projections on the acceptable level of quality for each application. We also conduct an optional survey at the end of the games to understand whether or not the games were entertaining. The survey results show that 84% of players were entertained.

Game initialization. The results are reported when the games are deployed with random initial votes for 10 imaginary gamers on each output. These random initial votes are not used in the statistical analysis. To understand the effect of initialization, in a separate experiment, we also asked 10 graduate students to play the games without regard to the rewards. We then extrapolated their choices across all the outputs with the same quality. We observed that both initialization strategies yield similar trends and therefore, we report the results with random initial values. This similarity is the result of recruiting large number of turkers

[5]We received Institutional Review Board (IRB) approval before deploying AxGames. The request was approved in three weeks.

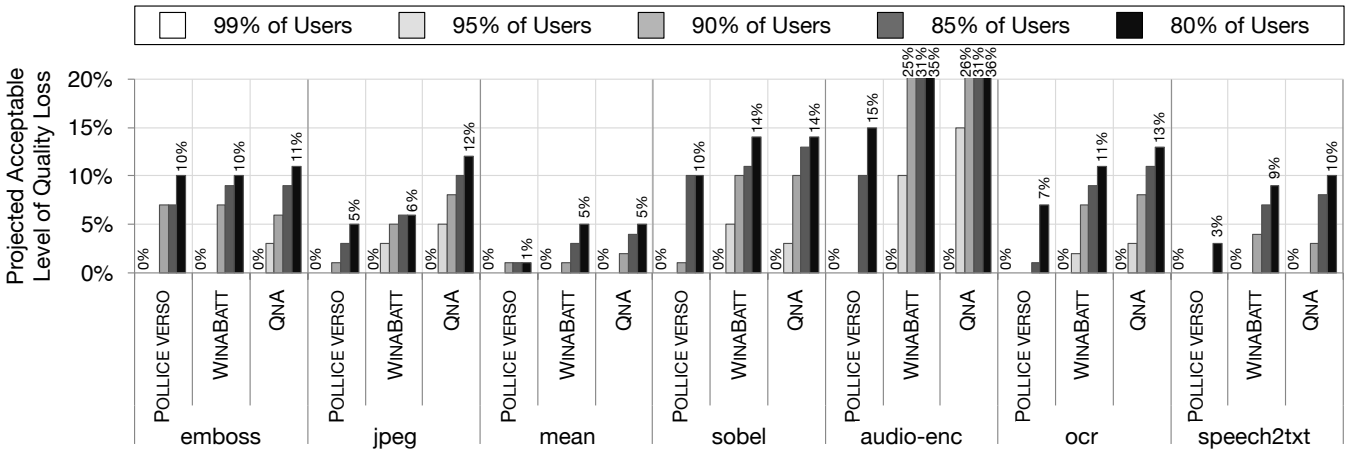


Figure 6: Projected acceptable level of quality loss with 95% confidence. These levels of quality are projected by our statistical analysis based on the game plays of 100 Mechanical Turk workers. Starting from left, each bar corresponds to the level that is projected to satisfy 99%, 95%, 90%, 80%, and 80% of the applications' users. These projections vary significantly across the applications.

and the fact that the human turkers ultimately make decisions based on their own perception of the approximated outputs.

5.2 Statistical Projections

Figure 6 shows the projected acceptable level of quality for each approximated application. The confidence level of these projections is 95%. As shown, each pair of (application, game) yields a set of projections. Starting from left, each bar corresponds to the level that satisfies 99%, 95%, 90%, 85%, and 80% of statistical population of the application users. The details of these projections are provided in Figure 7 and are discussed later in the section. As shown in Figure 6, for all the pairs of (application, game), the projected level of quality loss that satisfies more than 99% of the users is 0%. That is, for developers who aim to satisfy 99% of their users, the specific approximation techniques that are used in our evaluations (tiling and fine-grained approximation) are not a viable option. However, when the target is to satisfy a large majority of the users, starting from 90% of the users, there are opportunities to utilize these approximation techniques across all the image processing applications. Based on the statistics from POLLICE VERSO, if a developer chooses to satisfy 90% of the users, emboss can utilize tiling with 7% quality loss while jpeg, mean, and sobel can utilize tiling with 1% quality loss. Based on POLLICE VERSO, audio-enc and ocr can be only approximated if target is to satisfy 85% or less percentage of the users. For speech2txt, only if the target is to satisfy 80% or less percentage of the users, approximation can be enabled given the statistics from POLLICE VERSO.

As Figure 6 depicts, based on the statistics from QNA, all the applications can be approximated if the target is to satisfy 90% of the users. For this target, the acceptable level of quality loss is 6% for emboss, 8% for jpeg, 2% for mean, 10% for sobel, 26% for audio-enc, 8% for ocr, and 3% for speech2txt. As these results show, there is a clear difference between the three games when they assess the user satisfaction even for one application. This difference emanates from the fact that WINABATT and QNA provide an opportunity for the users to consider tradeoff and context, while POLLICE VERSO does not. We will discuss these differences in more detail later in this section.

User response varies significantly across applications. Another observation from these results is that the user-driven level of acceptable quality varies significantly across applications. Consider the four image processing applications that use the same approximation technique, tiling. Users show relatively higher tolerance to the tiling approximation for two of the applications, emboss and sobel. However, for other applications, especially for mean, the users show significantly lower tolerance. This significant variation in user response to the same approximation technique across different applications shows the necessity of solutions that statistically evaluate the acceptable level of quality. AXGAMES is one such solution, that effectively enables the users to provide statistical feedback to the developers who intend to leverage approximation techniques. With QNA, 90% of the users only accept 2% quality loss for mean while they tolerate 26% for audio-enc. These results shed light on the significant variation of users response to quality loss from approximation for different applications. To this end, AXGAMES provides the grounds for researchers to statistically evaluate their innovations from the users' perspective.

Users show higher tolerance to approximation when they consider cost or context. From the results in Figure 6, it is evident that users show higher level of tolerance to the approximation when playing WINABATT and QNA. Whereas the level of tolerance to approximation is lower in POLLICE VERSO. Intuitively, in these two games the players choose a level of acceptable quality in a cost-conscious manner while also considering the context in the case of QNA. In the case of POLLICE VERSO, the players vote only based on their personal impression of the approximated output without a chance to explore the cost-quality tradeoff. On the other hand, the other two games, WINABATT and QNA, introduce a notion of tradeoff between cost and benefits of approximation. Moreover, the players' choice of quality in QNA is driven by their ability to answer the associated multiple choice question with the output. This analysis shows that these three games collectively provide a deeper understanding of the user' reaction to the approximation technique. The developer may choose to use the results from any of these games on her

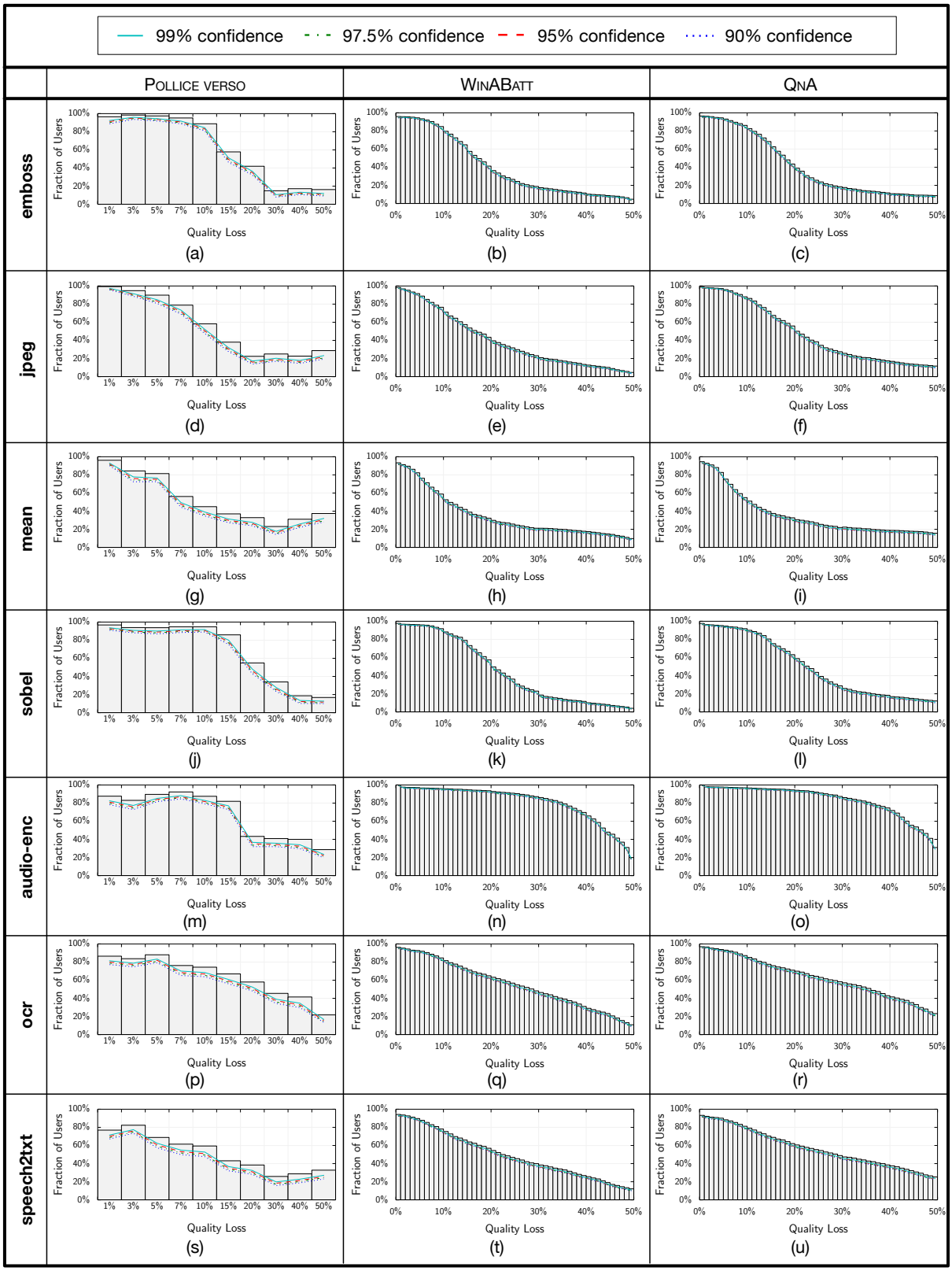
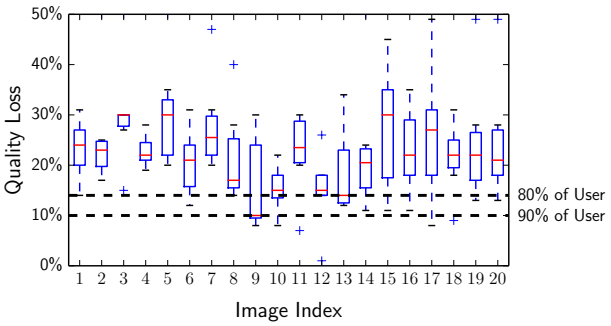
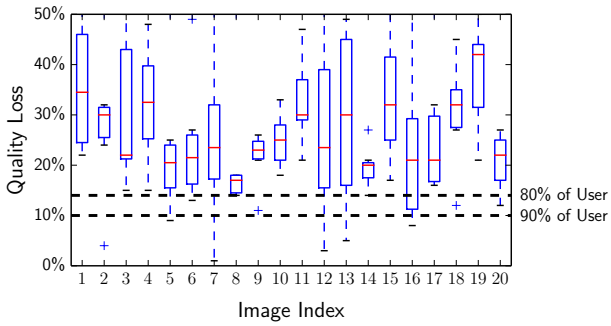


Figure 7: The collected statistics and the statistical projections. The bars show the fraction of players that selected a certain level of quality loss as "Good Enough." The lines represent the lower bound of the binomial confidence interval with different degrees of confidence, including 99%, 97.5%, 95%, and 90%.

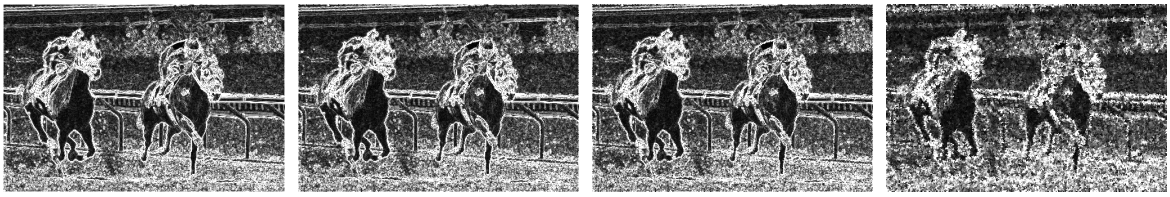


(a) Distribution of the quality choices in WINABATT



(b) Distribution of the quality choices in QNA

Figure 8: The box plot distribution of the players' choices of quality for sobel. The dashed horizontal lines show the projected acceptable level of quality loss that satisfies 80% and 90% of the users with 95% confidence (a) based on WINABATT and (b) QNA.



(a) Precise output (b) Output with 10% quality loss. (c) Output with 14% quality loss. (d) Output with 30% quality loss.

Question: *What can you see in the image?* **Correct answer**: *Horse racing* **Wrong answers**: *Bowling, Bridge, Buffalo wings*

Figure 9: Outputs from the edge detection filter, sobel, for image 13 in Figure 8(b). The leftmost output is the precise version and the rest are the approximated outputs. The approximated outputs (b) and (c) have 10% and 14% of quality loss that satisfy 90% and 80% of the users, respectively. The output (d) has 30% of quality loss, which is the median of the votes for the image 13 from the QNA plays.

own discretion depending on her constraints on user experience and her target deployment environment.

5.3 Collected Statistics from the Games

The collected statistics from the games are presented in Figure 7. The bars represent the fraction of players who chose "Good Enough" for each level of quality loss. Our analysis uses the Clopper-Pearson method to computes the one-sided confidence interval for each level of quality loss. This one-sided lower-bound conservatively projects what fraction of the users will deem a particular level of quality loss as "Good Enough." The lines in Figure 7 represent the statistical projections with different levels of confidence, including 99%, 97.5%, 95%, and 90%. As shown, the projection lines fall below the collected statistics from the sampled population (the turkers who played the games). That is because the Clopper-Pearson confidence interval, by definition, covers the sampled statistics in a conservative manner. A point on a projection line with the (x,y) coordinates predicts that *at least* $y\%$ of the users will deem $x\%$ quality loss as "Good Enough." For instance, suppose that we want to use the statistical results from WINABATT to find an acceptable level of quality for emboss. Let's target to find the quality loss level that provides satisfactory experience at least for 90% of the users with 95% confidence level. The quality loss is 7%, which is the x coordinate of the intersection point between the y=90% line and the red dashed projection line in Figure 7(b). The results in Figure 6 are obtained in this manner and also summarizes the statistical data given in Figure 7. As depicted, the results for POLLICE VERSO do not decrease monotonically whereas the results for WINABATT and QNA do. That is because only for these two game, the statistical analysis converts the players' selected level to binary decisions for all the levels of quality loss (see Section 4).

From the results in Figure 7, we observe different patterns for the different classes of applications. For the image processing applications, the graphs have three regions, the top region where the majority of users agree on the low quality loss; the middle region where there is no clear consensus among the users; and the tail where the majority of users reject low quality outputs The quality level from which the middle region starts captures the point that the majority's opinion is shifting. This point of shift in opinion can be found by calculating changes in the derivative of the projection lines. This point of shift may be used by the developers to choose the acceptable level of quality. AXGAMES provides the opportunity to find this point of shift for the developers. Instead of choosing the level of quality loss that certain percentage of users prefer, a developer can optimistically choose this point of shift in opinion.

The audio-enc application, which generates auditory output, shows a different pattern. The gamers show a significantly higher tolerance to the quality loss in the audio outputs. The majority of users tolerate the quality loss up to a significantly high level, after which the user satisfaction drastically declines. For the pattern recognition applications, ocr and speech2txt, the graphs show that the fraction of satisfied users almost linearly decreases as the quality loss increases. These two applications generate textual output. Unlike the other applications, there is not a clear point of shift in the crowd's opinion about the quality loss. These difference are significant and depend on the inherent characteristics of application, its output format, and the applied approximation techniques. Our experimental results show that the AXGAMES

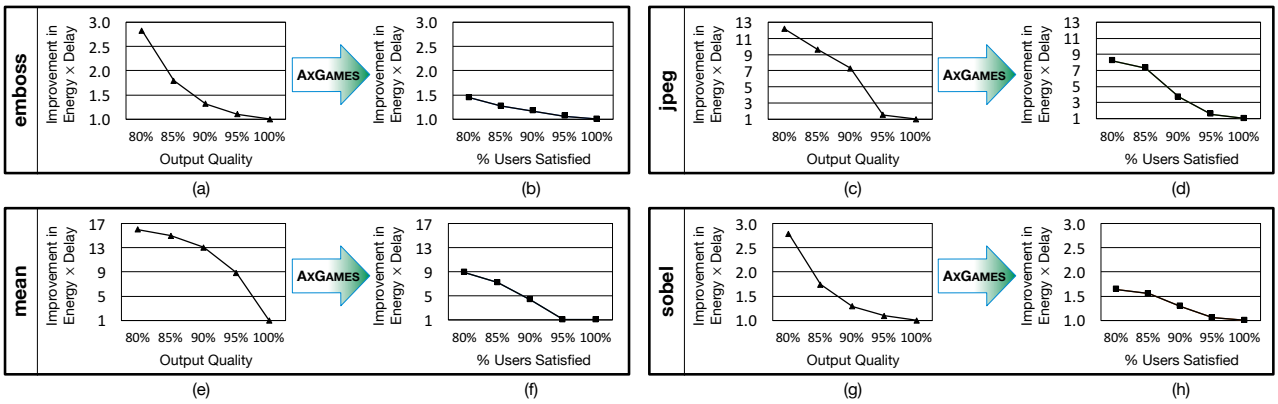


Figure 10: Improvement in energy-delay products vs. output quality (a, c, e, and g), and vs. % users satisfied. (b, d, f, and h). The results in (b), (d), (f), and (h) are based on the statistics collected from the QNA plays.

framework can effectively capture such patterns that need to be taken into account when approximation is employed.

5.4 User Response Variations

We investigate the user response variation across the same approximated images in Figure 8. Figure 8 shows the distribution of the responses from the players for a randomly selected subset of the output images from sobel. The trends are similar for the other applications. This distribution is shown as a box plot. The bottom and top of each box represent the lower and upper quartiles, respectively, and the band near the middle of the box is the 50th percentile (the median). The bottom whisker represents the lowest datum that is still within 1.5 inter quartile range (IQR) of the lower quartile. The top whisker denotes the highest datum that is still within 1.5 IQR of the upper quartile. The dashed horizontal lines show the projected acceptable levels of quality loss that satisfy 80% and 90% of the users with 95% confidence. It is even visually evident that these levels of quality cover their corresponding majority of users.

As expected, there is a large variation in the players choices. That is each player is providing her own personal judgment on the quality of the output, which is one of the main objectives of the games. Interestingly, the variation is higher in the QNA game. We investigate this higher variation in Figure 9 by showing the output image with the highest variation in Figure 8(b), image 13. The associated question with this image is "What can you see in the image: Horse racing/Bowling/Bridge/Buffalo wings?". This question can be answered even with high quality loss (30%) as shown Figure 9. However, it is understandable if a player chooses a lower quality loss to answer the question. Qualitatively, the context provided by the question allows a fraction of the users to choose levels of quality loss that are relatively higher. We speculate that approximation can be applied more aggressively if the output of this image processing is fed to a machine learning algorithm that performs scene detection or object recognition.

5.5 Changing the Tradeoff for Approximate Computing

The AXGAMES framework enables programmers to *change* the tradeoff between quality and performance-energy gains into the tradeoff between the users' satisfaction and the gains. We demonstrate this ability by measuring the speedup and energy savings for the image processing applications. A full investigating of the benefits of approximation is out of the scope of this work. This study is not to advocate approximation or show how much gains are possible; rather, it is to investigate the users' perspective on the output quality loss. For the performance and energy modeling, we use the same setup as the one used in [9]. We use the MARSSx86 x86-64 cycle-accurate simulator [34] and McPAT [35] for timing and energy modeling, respectively. The processor is modeled after a single-core Intel Nehalem (3.4 GHz with 0.9 V at 45 nm). The use statistics are based on QNA.

Figure 10 shows the improvement in energy-delay product when the output quality and the fraction of satisfied users change from 80% to 100%. The baseline is the application running on the processor without any approximation. Figure 10(a, c, e, and g) represent the tradeoff between quality and the gains from approximation while Figure 10(b, d, f, and h) represent the tradeoff between the user satisfaction and the gains. All applications see a disparity in the energy-delay product improvement for the same level of quality loss and fraction of satisfied users. For instance, in Figure 10(e), mean sees 8.9× energy-delay product improvement with 95% quality. Even though this level of quality seems high, it only satisfies about 80% of users. For mean, as shown in Figure 10(f), to satisfy 95% of users, only 1.3× energy-delay product improvement can be achieved, which is significantly lower than the gains that can be achieved with the 95% quality. Although the results in Figure 10 are specific to the pair of (application, approximation technique), they show a clear change in the tradeoffs when user response is considered.

While many approximate techniques provide significant benefits, their utility cannot be established without understanding the users' perspective. Our framework and our analysis are impartial to approximation techniques and are intended to shed light on how application users react to approximation. The games and the statistical analysis provide an effective mean for developers to understand the user experience before employing approximation. In fact, the developers can explore the tradeoffs and the benefits in a new light that considers users' perspective.

5.6 Discussion

Currently, AXGAMES is implemented for applications that directly produce human-perceivable outputs, such as image processing, audio processing, and pattern recognition applications. These programs represent a large body of applications that are designed for humans as the end users. These applications can ben-

efit from approximation due to the inherent tolerance of humans to inexact results. In fact, many of the approximate computation research includes such applications [5, 6, 8–10, 12, 13, 27, 28]. Furthermore, these classes of applications, such as Instagram, Microsoft HoloLens, Qualcomm Vuforia, are gaining prominence as the computing services, more and more, aim to provide natural and targeted experiences for the end users. This trend has been amplified by prevalence of mobile devices and will grow in importance as wearable electronics is gaining traction and smart and interactive home/office environments are emerging. Moreover, even complex machine learning algorithms are being deployed in interactive data visualization [36] and analytics tools [37] that allow humans to interact with complex and large amounts of data. Besides the direct use of AxGames in these domains, the results of AxGames can be used as an upper bound for inaccuracy in cases where the approximate output is fed to a machine learning engine. Humans are the ultimate recognition engines.

As an end-to-end attempt toward crowdsourcing the target quality determination, our approach takes an initial and effective step toward enabling the end users to become a helping force in the development and deployment of approximation techniques.

6. Related Work

There has been a substantial amount of effort to leverage crowdsourcing tools and games to solve computationally difficult problems. However, there is a lack of solutions that leverage crowdsourcing or game design for approximate computing which statistically determine the level of quality considered acceptable by the majority of application users. This paper provides one such solution and lies at the intersection of (a) approximate computing, (b) crowdsourcing, and (c) game with a purpose.

Approximate computing. A growing body of recent work explores a variety of approximation techniques. These techniques include (a) approximate storage designs [38, 39] which trade data quality for reduced energy [38] and larger capacity [39], (b) voltage over-scaling [7, 13, 28, 40–43], (c) computation and data sampling [12, 14, 44, 45], (d) loop early termination [46], (e) computation substitution [5, 46–48], (f) memoization [6, 49, 50], (g) limited fault recovery [44, 51–56], (h) precision scaling [10, 57], (i) approximate circuit synthesis [27, 58–63], and (j) neural acceleration [8, 9, 26]. Many of these solutions include applications which produce perceptible outputs for users. Although these techniques report promising benefits from approximate computing, they do not explore the acceptability of the quality loss from the users' perspective.

Crowdsourcing. The reCAPTCHA [64] project is a successful crowdsourcing application in production. This system uses the CAPTCHA, human Turing test, to classify text images from scanned books when other techniques fail. Automan [65] is an automatic crowd programming system which enables the integration of human computation in conventional programming languages and rigorously studies scheduling, budgeting, and statistical quality control in programming with human computational resources. TurKit [66] provides a programming model to integrate human computation in JavaScript using templates which provide close integration with Mechanical Turk. TurKit also supports checkpointing and recovery. Russell et al. [67] developed a web-based tool that leverages crowdsourcing to identify objects and locations in images and annotate them. These research efforts are not concerned with quality loss in approximate computing.

Games with a purpose. Games with a purpose provide an opportunity to engage the crowd in entertaining applications while providing insight to researchers. Several prior works have successfully utilized game-based crowdsourcing to label random images and locate objects [68–70]. Furthermore, Dietl et al. [71] proposed to transform verification tasks into a puzzle games which can be solved by humans. The solution of the puzzle is then translated back and used for proving correctness and verification. Foldit [72] is an online multiplayer game whose players interact with protein structures while they compete and collaborate to optimize the computed energy. They discovered that players of their game are able to search the state space of proteins configurations faster than computational algorithms. We are inspired by these efforts and exclusively developed a solution that uses games with a purpose and crowdsourcing to statistically determine the users' acceptable level of quality for approximate computing.

7. Conclusion

Approximate computing is an emerging area that breaks the long-held fundamental abstraction of near-perfect accuracy in PL [10, 11, 13, 73–75], OS [76], and Architecture [5–9, 26, 38]. While these techniques provide promising gains, they cause quality loss whose effects on the users are not well understood; leaving approximation in a position of questionable utility. Many of these inspiring studies argue that a certain quality loss may be acceptable without systematically considering the users' perspective. It is timely to systematically explore and study users' perspective on the effects of approximation. This paper takes an effective initial step in this direction. This work provided an automated programming tool–rather unconventional–to methodically utilize crowdsourcing in identifying the desirable application output quality from the final users. This readily available tool provides a path for the research community to better assess their innovative approximation techniques. The framework enables developers to conveniently study user responses at scale and gain statistical confidence when deploying approximated applications. Our results from examining a variety of applications show the necessity of solutions such as AxGames since the crowd's response to approximation varies drastically across different applications. Moreover, when the users consider tradeoff and context, they tend of to be more tolerant to approximation. These results suggests that AxGames can add an unexplored, yet important, dimension to the research and development in approximate computing.

8. Acknowledgments

We thank the anonymous reviewers for their insightful comments. We thank Richard Lipton and Faramarz Fekri for discussions that helped refine this work. We also thank Zhijian Li, Nick Liu, Elizabeth Perakis, Akshay Sawant, and Wen Xin for assistance in developing the games. We thank Amir Yazdanbakhsh for his contributions to energy-performance measurements. This work was in part supported by Qualcomm Innovation Fellowship, NSF award CCF#1553192, Semiconductor Research Corporation contract #2014-EP-2577, and a gift from Google.

References

[1] Hadi Esmaeilzadeh, Emily Blem, Renee St. Amant, Karthikeyan Sankaralingam, and Doug Burger. Dark silicon and the end of multicore scaling. In *ISCA*, 2011.

[2] Nikos Hardavellas, Michael Ferdman, Babak Falsafi, and Anastasia Ailamaki. Toward dark silicon in servers. *IEEE Micro*, 31(4):6–15, July–Aug. 2011.

[3] Ganesh Venkatesh, Jack Sampson, Nathan Goulding, Saturnino Garcia, Vladyslav Bryksin, Jose Lugo-Martinez, Steven Swanson, and Michael Bedford Taylor. Conservation cores: Reducing the energy of mature computations. In *ASPLOS*, 2010.

[4] John Gantz and David Reinsel. Extracting value from chaos. http://www.emc.com/collateral/analyst-reports/idc-extracting-value-from-chaos-ar.pdf.

[5] Mehrzad Samadi, Janghaeng Lee, D. Anoushe Jamshidi, Amir Hormati, and Scott Mahlke. Sage: Self-tuning approximation for graphics engines. In *MICRO*, 2013.

[6] Mehrzad Samadi, Davoud Jamshidi, Janghaeng Lee, and Scott Mahlke. Paraprox: Pattern-based approximation for data parallel applications. In *ASPLOS*, 2014.

[7] Hadi Esmaeilzadeh, Adrian Sampson, Luis Ceze, and Doug Burger. Architecture support for disciplined approximate programming. In *ASPLOS*, 2012.

[8] Hadi Esmaeilzadeh, Adrian Sampson, Luis Ceze, and Doug Burger. Neural acceleration for general-purpose approximate programs. In *MICRO*, 2012.

[9] Renée St. Amant, Amir Yazdanbakhsh, Jongse Park, Bradley Thwaites, Hadi Esmaeilzadeh, Arjang Hassibi, Luis Ceze, and Doug Burger. General-purpose code acceleration with limited-precision analog computation. In *ISCA*, 2014.

[10] Adrian Sampson, Werner Dietl, Emily Fortuna, Danushen Gnanapragasam, Luis Ceze, and Dan Grossman. EnerJ: Approximate data types for safe and general low-power computation. In *PLDI*, 2011.

[11] Jongse Park, Hadi Esmaeilzadeh, Xin Zhang, Mayur Naik, and William Harris. Flexjava: Language support for safe and modular approximate programming. In *FSE*, 2015.

[12] Stelios Sidiroglou, Sasa Misailovic, Henry Hoffmann, and Martin Rinard. Managing performance vs. accuracy trade-offs with loop perforation. In *FSE*, 2011.

[13] Michael Carbin, Sasa Misailovic, and Martin Rinard. Verifying quantitative reliability for programs that execute on unreliable hardware. In *OOPSLA*, 2013.

[14] Inigo Goiri, Ricardo Bianchini, Santosh Nagarakatte, and Thu D. Nguyen. Approxhadoop: Bringing approximations to mapreduce frameworks. In *ASPLOS*, 2015.

[15] Joshua San Miguel, Mario Badr, and Natalie Enright Jerger. Load value approximation. In *MICRO*, 2014.

[16] Adrian Sampson, Jacob Nelson, Karin Strauss, and Luis Ceze. Approximate storage in solid-state memories. In *MICRO*, 2013.

[17] C. J. Clopper and E. S. Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4):404–413, Dec. 1934.

[18] Sean Wallis. Binomial confidence intervals and contingency tests: mathematical fundamentals and the evaluation of alternative methods. *Journal of Quantitative Linguistics*, 20(3):178–208, July. 2013.

[19] A/b testing. https://en.wikipedia.org/wiki/A/B_testing.

[20] Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158): 209–212, Jun. 1927.

[21] Jeff Sauro and James R. Lewis. Estimating completion rates from small samples using binomial confidence intervals: Comparisons and recommendations. In *HFES*, 2005.

[22] Morris H. DeGroot. *Probability and Statistics*. Chapman & Hall, 1974.

[23] Lame mp3 encoder. http://lame.sourceforge.net.

[24] Speech recognition toolkit. http://cmusphinx.sourceforge.net.

[25] John W Ratcliff and David E Metzener. Pattern matching: The gestalt approach. *Dr. Dobb's Journal*, 13(7):46, 1988.

[26] Amir Yazdanbakhsh, Jongse Park, Hardik Sharma, Pejman Lofti-Kamran, and Hadi Esmaeilzadeh. Neural acceleration for gpu throughput processors. In *MICRO*, 2015.

[27] Amir Yazdanbakhsh, Divya Mahajan, Bradley Thwaites, Jongse Park, Anandhavel Nagendrakumar, Sindhuja Sethuraman, Kartik Ramkrishnan, Nishanthi Ravindran, Rudra Jariwala, Abbas Rahimi, Hadi Esmaeilzadeh, and Kia Bazargan. Axilog: Language support for approximate hardware design. In *DATE*, 2015.

[28] Sasa Misailovic, Michael Carbin, Sara Achour, Zichao Qi, and Martin Rinard. Chisel: Reliability- and accuracy-aware optimization of approximate computational kernels. In *OOPSLA*, 2014.

[29] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. URL http://image-net.org.

[30] Freesound.org. https://freesound.org.

[31] Voxforge. https://developer.nvidia.com/cuda-llvm-compiler.

[32] The new york times timesmachine. http://timesmachine.nytimes.com.

[33] Amazon's mechanical turk. https://www.mturk.com.

[34] Avadh Patel, Furat Afram, Shunfei Chen, and Kanad Ghose. MARSSx86: A full system simulator for x86 CPUs. In *DAC*, 2011.

[35] Sheng Li, Jung Ho Ahn, Richard D. Strong, Jay B. Brockman, Dean M. Tullsen, and Norman P. Jouppi. McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *MICRO*, 2009.

[36] Jaegul Choo, Changhyun Lee, Hannah Kim, Hanseung Lee, Barry L. Drake, and Haesun Park. Apolo: Making sense of large network data by combining rich user interaction and machine learning. In *VAST*, 2014.

[37] Duen Horng (Polo) Chau, Aniket Kittur, Jason I. Hong, and Christos Faloutsos. Apolo: Making sense of large network data by combining rich user interaction and machine learning. In *CHI*, 2011.

[38] Song Liu, Karthik Pattabiraman, Thomas Moscibroda, and Benjamin Zorn. Flikker: Saving refresh-power in mobile devices through critical data partitioning. In *ASPLOS*, 2011.

[39] Adrian Sampson, Jacob Nelson, Karin Strauss, and Luis Ceze. Approximate storage in solid-state memories. In *MICRO*, 2013.

[40] Lakshmi N. Chakrapani, Bilge E. S. Akgul, Suresh Cheemalavagu, Pinar Korkmaz, Krishna V. Palem, and Balasubramanian Seshasayee. Ultra-efficient (embedded) SOC architectures based on probabilistic CMOS (PCMOS) technology. In *DATE*, 2006.

[41] Sriram Narayanan, John Sartori, Rakesh Kumar, and Douglas Jones. Scalable stochastic processors. In *DATE*, 2010.

[42] Rajamohana Hegde and Naresh Shanbhag. Energy-efficient signal processing via algorithmic noise-tolerance. In *ISLPED*, 1999.

[43] Larkhoon Leem, Hyungmin Cho, Jason Bau, Quinn Jacobson, and Subhasish Mitra. ERSA: error resilient system architecture for probabilistic applications. In *DATE*, 2010.

[44] Sasa Misailovic, Stelios Sidiroglou, Hank Hoffman, and Martin Rinard. Quality of service profiling. In *ICSE*, 2010.

[45] Martin Rinard, Henry Hoffmann, Sasa Misailovic, and Stelios Sidiroglou. Patterns and statistical analysis for understanding reduced resource computing. In *Onward!*, 2010.

[46] Woongki Baek and Trishul Chilimbi. Green: a framework for supporting energy-conscious programming using controlled approximation. In *PLDI*, 2010.

[47] Jason Ansel, Cy Chan, Yee Lok Wong, Marek Olszewski, Qin Zhao, Alan Edelman, and Saman Amarasinghe. Petabricks: a language and compiler for algorithmic choice. In *PLDI*, 2009.

[48] John Sartori and Rakesh Kumar. Branch and data herding: Reducing control and memory divergence for error-tolerant gpu applications. *IEEE Transactions on Multimedia*, 15(2), 2013.

[49] Carlos Alvarez, Jesus Corbal, and Mateo Valero. Fuzzy memoization for floating-point multimedia applications. *IEEE Trans. on Computers*, 54 (7), 2005.

[50] Jose-Maria Arnau, Joan-Manuel Parcerisa, and Polychronis Xekalakis. Eliminating redundant fragment shader executions on a mobile gpu via hardware memoization. In *ISCA*, 2014.

[51] Marc de Kruijf, Shuou Nomura, and Karthikeyan Sankaralingam. Relax: an architectural framework for software recovery of hardware faults. In *ISCA*, 2010.

[52] Xuanhua Li and Donald Yeung. Application-level correctness and its impact on fault tolerance. In *HPCA*, 2007.

[53] Xuanhua Li and Donald Yeung. Exploiting application-level correctness for low-cost fault tolerance. *Journal of Instruction-Level Parallelism*, 2008.

[54] Marc de Kruijf and Karthikeyan Sankaralingam. Exploring the synergy of emerging workloads and silicon reliability trends. In *SELSE*, 2009.

[55] Yuntan Fang, Huawei Li, and Xiaowei Li. A fault criticality evaluation framework of digital systems for error tolerant video applications. In *ATS*, 2011.

[56] Vicky Wong and Mark Horowitz. Soft error resilience of probabilistic inference applications. In *SELSE*, 2006.

[57] Swagath Venkataramani, Vinay K. Chippa, Srimat T. Chakradhar, Kaushik Roy, and Anand Raghunathan. Quality programmable vector processors for approximate computing. In *MICRO*, 2013.

[58] Ashish Ranjan, Arnab Raha, Swagath Venkataramani, Kaushik Roy, and Anand Raghunathan. Aslan: Synthesis of approximate sequential circuits. In *DATE*, 2014.

[59] Swagath Venkataramani, Amit Sabne, Vivek Kozhikkottu, Kaushik Roy, and Anand Raghunathan. Salsa: Systematic logic synthesis of approximate circuits. In *DAC*, 2012.

[60] Jin Miao, A. Gerstlauer, and M. Orshansky. Approximate logic synthesis under general error magnitude and frequency constraints. In *ICCAD*, 2013.

[61] Kumud Nepal, Yueting Li, R. Iris Bahar, and Sherief Reda. ABACUS: A technique for automated behavioral synthesis of approximate computing circuits. In *DATE*, 2014.

[62] Avinash Lingamneni, Christian Enz, Krishna Palem, and Christian Piguet. Synthesizing parsimonious inexact circuits through probabilistic design techniques. *ACM Transactions on Embedded Computing Systems*, 12(2s): 93:1–93:26, May 2013.

[63] Avinash Lingamneni, Kirthi Krishna Muntimadugu, Christian Enz, Richard M. Karp, Krishna V. Palem, and Christian Piguet. Algorithmic methodologies for ultra-efficient inexact architectures for sustaining technology scaling. In *CF*, 2012.

[64] Luis Von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. recaptcha: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008.

[65] Daniel W Barowy, Charlie Curtsinger, Emery D Berger, and Andrew McGregor. Automan: A platform for integrating human-based and digital computation. In *OOPSLA*, 2012.

[66] Greg Little, Lydia B Chilton, Max Goldman, and Robert C Miller. Turkit: human computation algorithms on mechanical turk. In *UIST*, 2010.

[67] Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman. Labelme: a database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1-3):157–173, 2008.

[68] Luis Von Ahn. Games with a purpose. *Computer*, 39(6):92–94, 2006.

[69] Luis Von Ahn, Ruoran Liu, and Manuel Blum. Peekaboom: a game for locating objects in images. In *CHI*, 2006.

[70] Luis Von Ahn and Laura Dabbish. Labeling images with a computer game. In *CHI*, 2004.

[71] Werner Dietl, Stephanie Dietzel, Michael D Ernst, Nathaniel Mote, Brian Walker, Seth Cooper, Timothy Pavlik, and Zoran Popović. Verification games: Making verification fun. In *FTfJP*, 2012.

[72] Seth Cooper, Firas Khatib, Adrien Treuille, Janos Barbero, Jeehyung Lee, Michael Beenen, Andrew Leaver-Fay, David Baker, Zoran Popovi, and Foldit players. Predicting protein structures with a multiplayer online game. *Nature*, 466(7307):756–760, 2010.

[73] Henry Hoffmann, Stelios Sidiroglou, Michael Carbin, Sasa Misailovic, Anant Agarwal, and Martin Rinard. Dynamic knobs for responsive power-aware computing. In *ASPLOS*, 2011.

[74] J. Bornholt, T. Mytkowicz, and K. McKinley. Uncertain<T>: A first-order type for uncertain data. In *ASPLOS*, 2014.

[75] Michael Ringenburg, Adrian Sampson, Isaac Ackerman, Luis Ceze, and Dan Grossman. Monitoring and debugging the quality of results in approximate programs. In *ASPLOS*, 2015.

[76] Phillip Stanley-Marbell and Martin Rinard. Lax: Driver interfaces for approximate sensor device access. In *HotOS*, 2015.