Augmenting Crowdsourcing techniques with Artificial Intelligence

Travis Archer
Morningside College
Sioux City, Iowa 51106
tra001@morningside.edu

Abstract

In this paper I show that, based on my research with Project SCENIC, artificial intelligence (AI) can be used to augment and enhance crowdsourcing techniques. In Project SCENIC (SCENIC is Crowdsourcing Enhanced by Neural-nets Interested in Cell-noise) I conducted an experiment between two programs. The control group was a genetic algorithm that created random images using Cell noise that users could rate. Based on these ratings, the genetic algorithm threw out poorly scored images and combined or altered highly rated images into new images to be rated. This evolution of images eventually lead to a higher average score for the images. Meanwhile the other group seemed identical to the outside, but a neural-net was taking user-scores from this group to alter its own perception of beauty. When the neural-net was confident an image would be scored below average, the image was removed from the study. By removing unpleasant images, the neural-net enabled users to stay focused on more interesting images, instead of spending part of their time immediately dismissing boring or unappealing images. The neural-net used many different criteria to judge an image, including edge-detection, color relationships, and color diversity, amongst others. Ideally, the neural-net would have enough grasp of what makes an image appeasing to save humans the trouble of rating truly ugly or boring images. In reality, the augmented group started out with roughly similar scores to the control group while the neural-net became accustomed to what the users saw as beautiful. After enough time had passed, the control group plateaued at a relatively high score, while the augmented group achieved even higher average scores. Crowdsourcing is becoming increasingly used in research applications. By augmenting that with artificial intelligence, I argue that results could be found faster with less strain put on the users. While this technique may not be applicable to all projects, many could save time and even save money.

Travis Archer
Morningside College
Sioux City, Iowa 51106
tra001@morningside.edu

1 Crowdsourcing and Artificial Intelligence Currently

Over the last decade crowdsourcing (also refered to as 'human-based computation') has emerged as a new tool to process large amounts of data that cannot be accurately processed by computers. Crowdsourcing is the act of outsourcing tasks from an individual or small group to a large crowd of users, giving each user a small chunk of the task to complete. A prime example of crowdsourcing is the Amazon.com owned site MechanicalTurk, where a large collection of users perform short tasks for businesses or groups. Each user does very little work, but collectively it amounts to large results. More recently, DARPA launched the Shredder challenge, awarding fifty thousand dollars to the first team to successfully reconstruct 5 documents put through various shredding machines. Teams could use any techniques they desired, and artificial intelligence and crowdsourcing were the primary tactics used by teams. In the end, team "All Your Shreds Are Belong To U.S." won with an artificial intelligence that tried to pair chunks of paper together, then sent the result to human testers to verify [1]. This is perhaps the most successful example of crowdsourcing used to augment artificial intelligence. It was this that made me wonder if crowdsourcing could in turn be augmented with artificial intelligence. I believe that in a Venn diagram of crowdsourcing and artificial intelligence the efficiency peaks at the intersection. Project SCENIC was a proof of concept to show that crowdsourcing combined with artificial intelligence creates synergy, boosting efficiency beyond the sum of both parts.

2 Project SCENIC

SCENIC stands for *SCENIC is Crowdsourcing Enhanced by Neural-nets Interested in Cell-noise*. SCENIC combines noise algorithms, crowdsourcing, genetic algorithms, neural-nets, SQL, HTML, CSS, JavaScript, and C++. It was created to assess the synergy between crowdsourcing and artificial intelligence. This would require a problem that needed to be solved, something the average layman could easily provide input on, yet something too difficult for an artificial intelligence to grasp on its own. While the field of computer vision is quickly evolving, using computers to judge beauty is still in its infancy. There are various projects interested in many types of beauty, such as the work done by D. Gray et al. with facial beauty [3]. However, the area of abstract art is still relatively untouched, largely because it is so subjective. Most authors agree that art, and especially abstract art, are judged partially on the emotions elicited in the viewer. Seemingly an emotionless machine would be incapable of properly judging abstract art. Yet nearly every human with computer access has had some exposure to art, and should be able to judge an abstract image based on its artistic value. This forms the problem that can be solved easily by humans but so far has not been addressed with artificial intelligence.

2.1 General Structure

Project SCENIC is relatively simple in theory. Two sets of abstract images are randomly generated, one the control group, one the experimental group. A user opens the webpage and is presented with one of the images from an unknown set and asked to rate it on a scale

of one to five. After all the images have been rated, a genetic algorithm is used to 'breed' the pictures. These children are the original image with slight mutations, or the result of breeding with another image. Once the breeding process is done, the new images are rated by users and the process repeats. Eventually, the average rating of the pictures should rise in both groups. However, the experimental group is different in one critical way. In the experimental group, after an image is generated it is rated by a neural net. If the rating is below a given threshold then the image is dropped and is not rated by users. Before each breeding session, the neural net trains itself based on the difference between its own rating of an image and the user rating. In theory the experimental group will progress faster and with less user interaction than the control group.

This structure is the perfect environment for crowdsourcing. Each user is presented with a task that requires no special knowledge, only the *ability* to judge the aesthetic quality of an image, which is a skill very few people lack. That means there is no learning curve or barriers to entry. Each image is small and self-contained, with no real relation to any other image, and each image is rated independently. That means there is no time commitment required; a user can rate one image or a hundred.

2.2 Making abstract art

Unfortunately, a large data set is required to train an artificial intelligence, as well as obtain a good sample for study. While there are easily thousands if not millions of pieces of abstract art in the world of varying quality, collecting and prepping enough of them would be extremely time consuming. Instead I turned to Cell noise, described by Steven Whorley in 1996 [4]. Cell noise is extremely flexible, capable of creating many different shapes. Cell noise works by scattering points across a surface, then assigning a value to a pixel by its distance to the n^{th} closest point. Altering n and combining different values of n creates vastly different images. Using an n of 1 is refered to as $F1$, while an n of 2 is $F2$, and so on. Combinations like $F1 - F2$ or $F3^3$ $-F2 * F1$ are not uncommon. Altering the distance function also creates remarkably different images, even with the same set of points. By choosing a random distance function and creating a random combination of Fn distances, a myriad of images can be quickly created. Once the value-map of the image is created by the Cell Noise algorithm, a randomized gradient can be applied to create a finished image.

For this application four distance functions were used: Euclidean-Squared, Manhattan, Quadratic, and Minkowski. The Euclidean distance function is the most obvious: $distance = \sqrt{(\Delta x)^2 + (\Delta y)^2}$, used almost universally in physics and mathematics. Euclidean-Squared removes the costly square-root function. As long as the image is normalized, this considerably boosts speed while producing roughly the same image. The Euclidean function generally creates images full of circles at $F1$. The Manhattan distance function is defined as $distance = |\Delta x| + |\Delta y|$. This creates diamond shapes in the image at $F1$. The Quadratic distance function is as it sounds, $distance = (\Delta x)^2 + (\Delta x)(\Delta y) + (\Delta y)^2$. This produces elongated circular shapes. The result of these three functions can be seen in Figure 1.

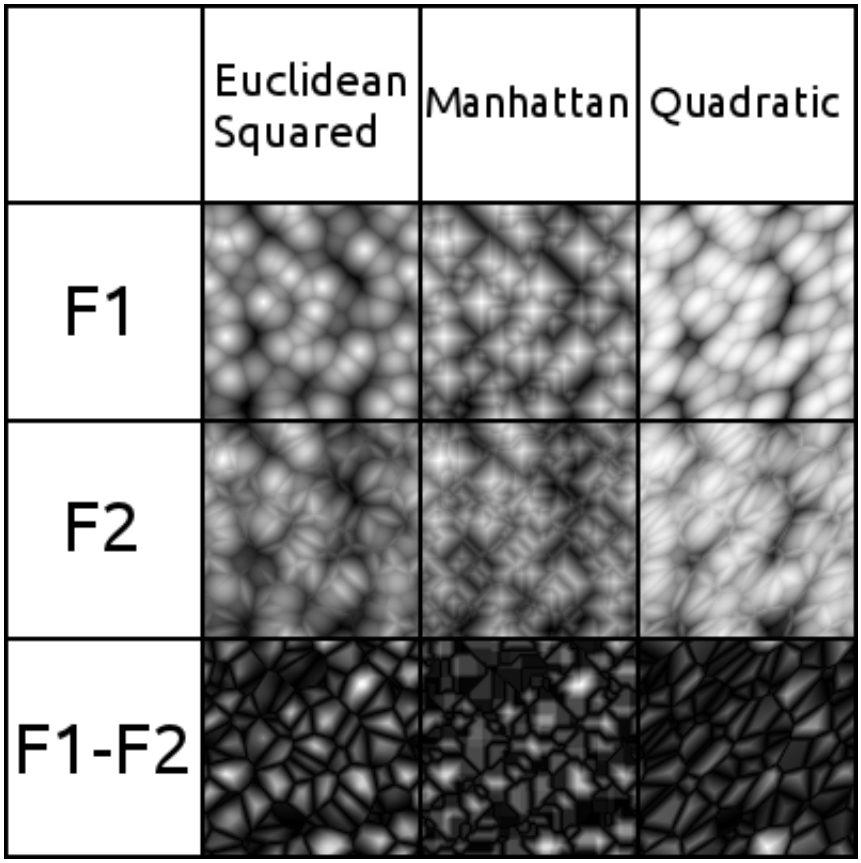
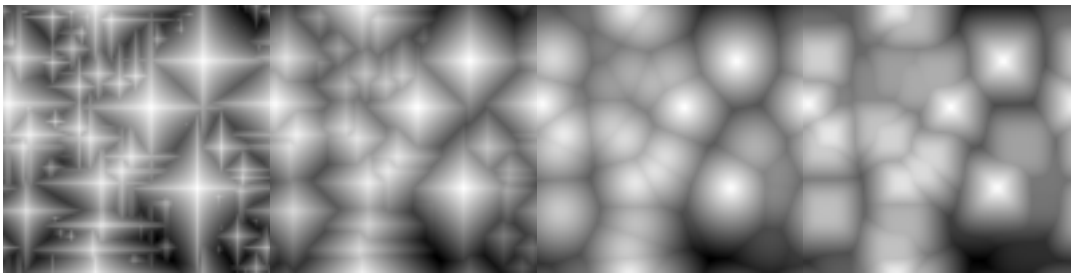


Figure 1: Comparison of three distance functions and 3 different Fn equations. Each image has the same number of points in the same places, but each is rendered differently by the different settings.

The last distance function, the Minkowski distance function, is a generalization of all other distance functions. It is defined as $distance = \sqrt[m]{|\Delta x|^m + |\Delta y|^m}$, with m being the Minkowski coefficient. Depending on the number used for m, different images can be made. At $m = 2$, the Minkowski function is equivalent to the Euclidean function. At $m = 1$, it is equivalent to the Manhattan function. Higher values produce increasingly abnormal shapes. While the Minkowski coefficient could have been varied with great effect, the Minkowski function is the most computationaly expensive of the four, and so only $m = 4$ was used. A comparison of some Minkowski functions can be seen in Figure 2.
Finally, a gradient must be generated to convert the image from grayscale to color. Here a gradient is defined as a set of colors with a position on a scale. The color of a point is found by normalizing the image and determining which two colors the point would be between. After that, linear interpolation is used to blend the two colors. The gradient provides more than just colors to look at, it can in many cases provide extra structures to the image beyond what the Cell noise produces. If two points are very close to each other in the gradient, a hard edge is formed in the image, creating new objects. An example of this behavior can be seen in Figure 3.



(a) $m = 0.5$ (b) $m = 1.0$ (c) $m = 2.0$ (d) $m = 4.0$

Figure 2: Comparison of four different Minkowski coefficients. Notice that $m = 1$ is identical to the Manhattan function, while $m = 2$ reproduces the Euclidean function.

To create a random piece of abstract art, all of these must be combined. First the number of points for the Cell noise function is randomly generated. Next, one of the four distance function is randomly chosen, with Euclidean-Squared given a slightly higher weight and Minkowski given a lower weight to improve speed. Then an Fn equation is generated by choosing a random n between 1 and 7, then a random operator (sum, difference, quotient, or product) and $F(n + 1)$ is added to about half of the images. Finally a gradient is randomly generated by defining colors for the 0.0 and 1.0 end-points, then creating a random amount of points between the two with random colors. In addition a random seed is generated so that the image can be exactly reproduced from the above specifications at a later date if necessary.

3 Breeding Pictures

Once all the images have been created, they are rated by the users on a scale of one to five. After all the images have been rated, they are bred to create new images. To do this, each image is given a set amount of offspring based on the image's score. An image with a score of five will get more children than an image with a score of one. The children are either the result of combining the arguments with those of another image (known as breeding), or by altering one of the arguments from the parent image (known as mutating). In most genetic algorithms breeding is used in as much as ninety-nine percent of all cases while mutating is used only one percent of the time. This is perfect for most problems because it assumes there is a correct answer and slowly ascends towards that answer. However for this problem there is no single definitive right answer.

Because of the intricacy of the images, combination would most often lead to dizzyingly complex images that are too overwhelming or disjointed to be considered aesthetically pleasing. On the other hand, by giving a highly rated image more children to mutate it will branch out and experiment with different styles, creating an image that is similar to the parent image but still distinctly different. For this reason, mutation occurs much more frequently than breeding in this application. The probability of breeding is given by the following equation: $\frac{49}{80} - (score * \frac{9}{80})$. For an image with a score of five (already beautiful)

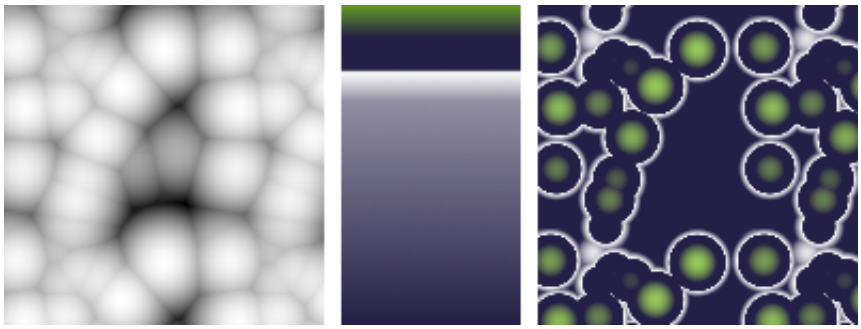


Figure 3: The image on the left becomes the image on the right after applying the gradient seen in the center. Notice the added edges and structures created by the gradient.

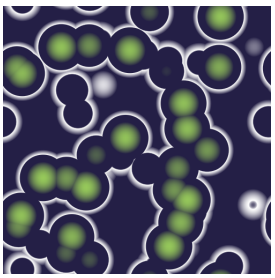
this equation gives a probability of breeding of only five percent. Meanwhile an image with a score of one, which has nothing to lose, has a probability of breeding of fifty percent. Because each generation of images is the same size (500 images per set) and images with higher scores have more children, not all children will fit in the new generation. For this reason, the images are ordered according to the user-assigned score in decending order, and the process is finished when 500 children have been created. This means that the process generally completes before lower-scored images have a chance to breed, meaning their lineage is lost forever. This process of natural selection means the new generation has a high probability of having a higher average rating.

Once the breeding process is complete, the old images are deleted to save space, however the arguments to recreate it are saved in the database. Then the new images are presented to the crowd to rate, and the cycle continues. Eventually this process of breeding and natural selection should result in higher average ratings for each generation by encouraging beautiful images to experiment and pruning ugly or boring images.

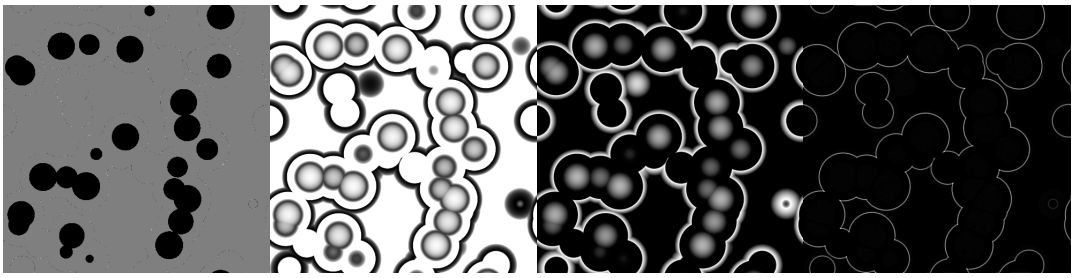
4 The Neural Net

The control group continues in the above fashion indefinitely, however the experimental group uses a neural net to weed out images that are definitely undesirable. One danger of crowdsourcing is user burn-out, where a user is 'burnt out' on doing a menial task over and over again. Many of the random images created are boring, and getting many such images in succession can be tedious. By dropping these undesirable images, the neural net leaves the user with more interesting images, reducing the tedium of the task at hand. This keeps users interested longer, meaning each user in the crowd will rate more images individually, reducing the time necessary to complete the set.

This all rests on the assumption that the neural net can do its job correctly. To do this, the neural net rates an image based on eight numbers created from four functions of the image. First and foremost the neural net creates an edge-map of the image by using the Difference of Two Gaussians technique. This is performed by creating two copies of the image, then using Gaussian blurring on each with a different kernel size. For this applica-



(a) Before



(b) Hue (c) Saturation (d) Value (e) Edges

Figure 4: The maps of each image created for the neural net.

tion I used kernel sizes of three and five. Then a new image is created by subtracting the value of each pixel in one image from the cooresponding pixel in the other. This creates a map of the edges and their clarity in a new image. An image with many shapes and objects will have a very active edge-map, while a very dull image will have a very inactive edge-map. This is believed by many to be very similar to how the human eye preprocesses images coming to the brain [5]. Once this map is created the average value is found as well as the standard deviation.

Next, the image is converted from the Red-Green-Blue (RGB) color space to the Hue-Saturation-Value (HSV) color space, also known as Hue-Saturation-Lightness (HSL). The average and standard deviation of each of the three is found, each performing a different function. The value (or lightness) map contains the basic structures of the image, so the standard deviation coorelates to the number and size of structures in the image. The overall saturation of an image can separate a very gray boring picture from a bright exciting image Meanwhile an image with many distinct colors with have a high standard deviation of the hue, while an image with only one or two colors will score low in this area. These six values plus the two from the edge detection make up the eight inputs for the neural net. An example of each of the four maps created can be seen in Figure 4.

The neural net itself is composed of an input layer, two hidden layers, and an output layer. Most neural nets use a single hidden layer to increase efficiency, however the first hidden layer in this case provides a special function. The first hidden layer contains four neurons, each of which only connect to the two inputs in a given area. For instance, one neuron is connected to the average and standard deviation of the hue, while another is connected to the average and standard deviation of the edge-detection. In this way, each area is judged

independently first, then a value is sent up to the next layer. The theory here is that one number from each area is more important than the other, yet it could be different for each area. By using the first hidden layer to weigh one side of an area over the other, the neural-net will eventually learn which side is more important and deliver more relevant results. The second hidden layer has two neurons, while the output layer has one, which returns a result between zero and one inclusively. This value is multiplied by five, rounded down, and incremented by one. This produces an output between one and five inclusively. This is the score given to the image by the artificial intelligence, and if it is below a threshold of acceptability then the image is dropped and never seen by users. While these eight numbers certainly can't encompass all that makes an image beautiful in the eyes of a human, they are enough to let a neural net judge when an image is too boring to be graded by humans.

5 Optimizations and Pitfalls

This process of creating four different maps of the image can be time-consuming, especially when the image is written to a PNG file, then must be read back into memory later to be graded by the artificial intelligence. To optimize this process the neural net grades the image just before it is written to file, while it is still in memory. Then the information from the neural net is uploaded to the database along with the new image information.

Other optimizations were planned to save valuable server resources but were never enacted. One plan was to store the image arguments in the database and cache a set amount of images, then have a supervisor program replenish that cache as each image is rated. However, the server architecture does not allow CGI scripts to be called by a client when a program is being run by the owner of the account, even if it is a daemon. Therefore, the cache is as large as the quota on the server allows, and when the cache runs out the entire set is bred to create the next generation. This shuts down the rating functionality during that time, and so the rating page automatically reroutes to a waiting page after 10 consecutive failed attempts to contact the server. This is important, as the breeding process generally takes about six hours to complete.

This restriction required that the GetImage CGI script do much more than just return an available image. Instead the script has to update the database of the last score input by the user, pick a random image from one of the two sets, unstick images that have been 'checked out' by a user for longer than ten minutes, and start the breeding process if no images are available. By combining all this functionality into one program the efficiency is increased dramatically, but the added complexity makes maintenance difficult. It also leaves the system with a single vulnerability that is insurmountable on the server-side with the given architecture: if a user is given the last image of the thousand produced for the two sets, the breeding will not start until that user finishes grading it, even if the user forgets about it and leaves the page open for hours. This is because the page will call the server again after 10 minutes of activity to release that image and request a new one. However, if that image is the only one left then the server will keep sending back to that user, even if a hundred other users are ready and waiting. This race condition is unlikely, but still a

definite threat. To address this, the webpage itself will reroute to the waiting page if it is idle for too long and recieves the same image three times in a row. This locks the process for a half-hour at most, but is still better than allowing a rogue user to intentionally disrupt the system.

One final issue faced was that in the unlikely event that the neural net rated all the images in the experimental group below the threshold then the entire experimental group would stop functioning altogether. This 'genocide' condition did occur in the second generation, before the neural net had a chance to fully develop. The solution was to manually reset the generation, and change the way the images are dropped. Instead of having the neural net drop images itself, it simply returns its score to the database and exits. After all the images in the next generation have been graded, a separate program determines which images to drop. If a given threshold would result in extinction for the set, then none are dropped. Conversely, if the given threshold would result in no images being dropped, then the threshold is raised to account for this.

6 Results

In the end, both groups plateaued at a certain point. This is likely due to the inherent randomness of the creation process, meaning there will always be some images that are ugly or boring. However, the experimental group plateaued at a higher average score than the control group. After seven generations the control group plateaued at an average score of 3.075 after 3500 user interactions. Meanwhile the experimental group had an average score of 4.092 after 3192 user interactions. This score is based on the average score placed by a human user, excluding those dropped by the neural net. These statistics can be found in Figure 5.

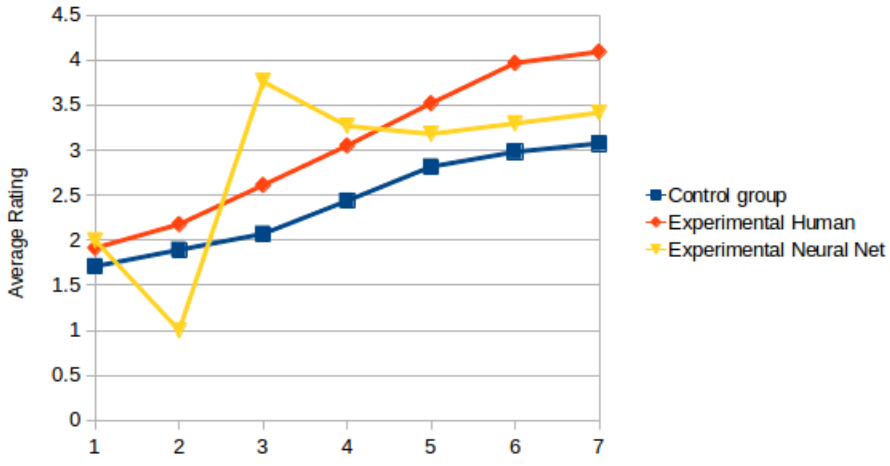


Figure 5: The collected scores from each generation.

In the first generation the neural net had no prior training and gave every image a score of

exactly 2. In the second generation it had minimal training and resulted in the extinction event described before. After a reset of the group the neural net continued to train, and by the third generation was giving images unique scores. These scores were highly optimistic, but the neural net eventually became better aclimated to the task and ended up between the two groups. This isn't surprising, as the experimental group averaged better partly because the scores that were dropped weren't included in the human average. By the seventh generation the experimental group greatly surpassed the control group, and the neural net scores closely matched those of the human participants.

7 Potential Applications

The applications for this technique go far beyond the trivial example SCENIC provides. Both SETI and NASA are releasing massive amounts of data to the public to categorize. Undoubtedly they already use some hard-coded discriminating factors to weed out seriously useless information. However, by utilizing the intersection between artificial intelligence and crowdsourcing, they could possibly speed up the process considerably.

One area where this could be particularly useful is in a business setting. Data mining is becoming increasingly popular in large corporations, and artificial intelligences are used most often for the job. These same corporations are constantly fighting to keep employees engaged on their work by blocking Facebook and removing Solitaire. Yet research like that done by Henning et al. is showing that taking short breaks can improve overall efficiency [2]. I believe these two issues need not be treated separately. Many crowdsourcing applications present the problem in the format of a game. If the data-mining were presented as a game to employees, they could take breaks by playing the game and classify the data. In doing so, the corporation would be getting useful output from employees even as they are winding down and relaxing. Then the employee returns to his or her duties with improved efficiency. The artificial intelligence will get large amounts of data to train with, and can filter out tedious data that could be counter-productive to relaxing employees.

8 Conclusion

In this way, the neural net enhances the crowdsourcing by dropping undesirable images and speeding up the rating process, while the crowdsourcing enhances the neural-net by providing training data. This synergy improves the efficiency of the entire system more than the sum of the two separately. The doors to such technology are only now opening, and with the ever growing deluge of data piling up, I believe this technology will be seen increasingly often.

References

[1] DARPA, (2011). DARPA Shredder Challenge. http://archive.darpa.mil/shredderchallenge/

[2] Henning, R., JACQUES, P., KISSEL, G., SULLIVAN, A., AND ALTERAS-WEBB, S. (1997). Frequent short rest breaks from computer work: effects on productivity and well-being at two field sites. *Ergonomics 40*, 78-91.

[3] GRAY, D., YU, K., XU, W., AND GONG, Y. (2010). Predicting Facial Beauty without Landmarks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, Crete, Greece, September 2010.

[4] WORLEY, S. (1996). A cellular texture basis function. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (SIGGRAPH '96)*. ACM, New York, NY, USA, 291-294.

[5] YOUNG, R. (1987). *The Gaussian derivative model for spatial vision: I. Retinal mechanisms. Spatial Vision 2*. 273293(21).