DATE: February 25, 2020

TO: Robert Heath

FROM: Arkan Abuyazid, Shehryar Ahmed, Brandon Lee, Akash Shukla, Meenakshi Swaminathan, Kan Vanthanasuksan

SUBJECT: Comprehensive plan for implementing our project

1.0 INTRODUCTION

Overall, we aim to create a design that is basically a system that can automatically determine the optimal JPEG quantization parameter when compressing an image. We define this optimal quantization parameter as the point that maximizes compression while still maintaining human-perceptual quality. The main distinction between our design and other pre-existing compression-optimizer solutions is that our perceptual predictor will be based exclusively on real human perception datasets as opposed to quality references and often-used pseudo-perceptual metrics.

As our project requires a dataset of human decisions, our main method of data collection will be through Amazon's Mechanical Turk (MTurk), which is a crowdsourcing platform that involves the distribution of a virtual task to a large human workforce. Our task will allow workers to control the quality of the images through a real-time slider that compresses the image according to the slider value so that they can determine the perceived optimal point of compression. The collected dataset will be fed into a neural network that we will construct, customize, and train so that the model can learn and eventually automatically predict the optimal quantization parameter for any input image.

This System Design Report will go into detail about how we define and measure a successful image compressor. The motivations, methods, goals, target users, challenges surrounding the design will also be explained. Furthermore, the report will cover the necessary steps that we took to ensure a successful execution of the project such as any Risk Reduction research, experiments, and activities that were performed as well as a detailed plan of how our design will be tested during and after the implementation phase. A complete schedule, responsibility, and delivery plan will also be shown, which includes the Work Breakdown Structure (WBS), Linear Responsibility Chart (LRC), Gantt Chart, and Bill of Materials (BOM). The WBS gives a visual representation of our design broken down into multiple activities and major tasks. The LRC links these assignments and major tasks to each team member and describes the division of responsibilities. The Gantt chart provides a clear overall schedule of the implementation of all the design phases and specifies the milestones and deadlines of the assignments and major tasks.

2.0 DESIGN PROJECT BACKGROUND

Our end goal is to create a system that is automatically capable of determining the optimal JPEG quantization parameter for compressing an image without compromising human-perceptual quality. Many current compression technologies rely on various metrics that are pseudo-perceptual [1]; however, we will differentiate ourselves and propose a better alternative. We will do this by gathering data on the human-perceived balance between compression and quality and by constructing a perceptual predictor that is based solely on datasets of real human decisions. In order to accomplish this, the foundation of our solution will be built on image processing, accurate datasets, machine learning models, and automation. Ultimately, our end product will take the form of a trained system that imitates human perception, predicts the optimal level of compression and quality for any image, and produces the optimal compressed image. We define a successful project to be one that produces images that are, on average, 5% smaller than images compressed according to the Structural Similarity Index (SSIM) standard and have a quantization parameter that is within +/- 3 of the human-perceived optimal quantization parameter. Note that JPEG quantization parameters vary on a scale from 1 to 100, with 100 being no change in the original image and 1 being the lowest quality possible [2].

Additionally, we would like both individuals and large companies to be able to use our model. This poses two requirements for our solution. First, it should use small amounts of RAM and storage in order to run efficiently on personal computers. Second, it should be able to quickly process many images at once so that it can keep up with the data flow of a major company.

Existing models similar to the one that we propose use pseudo-perceptual metrics such as the SSIM to gauge the clarity of the compressed image [3]. SSIM assesses quality by measuring the similarity between the original image and the compressed version; if the original and the compressed images are highly similar, the compressed image is considered to be high quality, and vice versa [3]. Notice that using SSIM assumes that the original image was high quality before compression. If the original image was low quality initially, then SSIM may determine a compressed version to be high quality since it looks like the original, but the compressed image would subjectively be considered poor quality. Instead, our method will be unique in that it will use actual human perception to determine optimal compression levels. This should give our model an edge; it should be able to more accurately choose the compression level that a human would consider 'optimal.'

This end goal will require collecting large amounts of human-perceptual data, training a model with this dataset, and testing the accuracy of the predictor. In the data collection step, we will utilize Amazon's Mechanical Turk (MTurk), a crowdsourcing tool that allows us to distribute virtual tasks to a human workforce. The MTurk task interface will show a series of images with two sliders underneath each image to collect worker opinions. The MTurk workers need to visually see the effect of compression in real-time and determine the optimal point without knowing the change in file size. Through this process, we aim to gather an accurate dataset that is reliable, varied in users, and large enough in sample size. Based on Professor Bovik's past experience, humans tend to agree when evaluating image quality, so we expect the human-selected compression levels to be fairly consistent for a particular image. We do, however, expect some variance in mean compression level between different images, since some images are more compressible than others.

The second step involves constructing a neural network to process the collected data. There are a variety of methods we could employ in designing our network, which we discuss in Section 3.1.2. For whichever method we implement, the network must take an image of arbitrary size (within reason) and produce the optimal compression level. Once the network is trained, it can be saved as the architecture and the trained weights. We will need to store the architecture and the weights in the final application. Assuming equal performance, we will favor networks with a smaller architecture and fewer weights, since having fewer weights means the end product will be smaller in size.

The final deliverable product will take on two forms. The first is a command line interface (CLI), while the second will be a graphical user interface (GUI). Both will perform the same functionality, which is to take as input a set of images whose optimal compression level is desired and then output the optimal quantization parameters. Each form will also have the option to output the compressed images themselves. The CLI is designed to be run in a terminal, while the GUI will be more user-friendly. These options provide solutions geared toward companies that wish to embed our product into their own pipeline as well as everyday consumers who wish to compress their images and are more comfortable with a graphical application.

3.0 SYSTEM DESIGN

We will discuss our design concept and risk reduction activities we completed that gave us more insight regarding the problem and removed any early obstacles. We will explain the different submodules in our design concept and the decisions behind why we structured them this way. We will go in depth the intuition we gained through the risk reduction activities and what obstacles we did remove. The most notable risk reduction activities are 1, 6, and 7. Activity 1 is a proof-of-concept for our human study interface, demonstrating that we can use Amazon MTurk to collect our data. Activity 6 is an experimentation with a neural network, ensuring that we have a team member who is familiar with machine learning when we reach the neural network portion of our project. Activity 7 is the implementation of a slider that is capable of real-time compression, i.e. the image will compress while the slider moves, without requiring that the worker let go of the slider to see the results of the compression. Activity 7 is important because it demonstrates the viability of a core mechanic of our data collection. Furthermore, we will provide the details of the testing that we plan to accomplish with both submodules of our design concept and our entire solution as a whole.

3.1 Design Concept

As displayed in Figure 1, our design can be split into the Mechanical Turk (MT) submodule and the Neural Network (NN) submodule. The MT submodule involves three main tasks. Building the Amazon Mechanical Turk HIT page, verifying and clearing the crowdsource data, and then configuring the data to be inputted into the NN submodule for training. The third task involves ensuring that the HIT page properly outputs the results that we collect and making sure we know how to transfer the results to our machine learning environment. The NN submodule trains the neural network, produces the optimal compression parameters, and includes the command line and graphical user interfaces. This submodule involves training, tuning, and testing the processed crowdsource data. The final trained model in the NN submodule will be used to create the command line interface (CLI) and graphical user interface (GUI). This is shown in the project flow below in Figure 1.

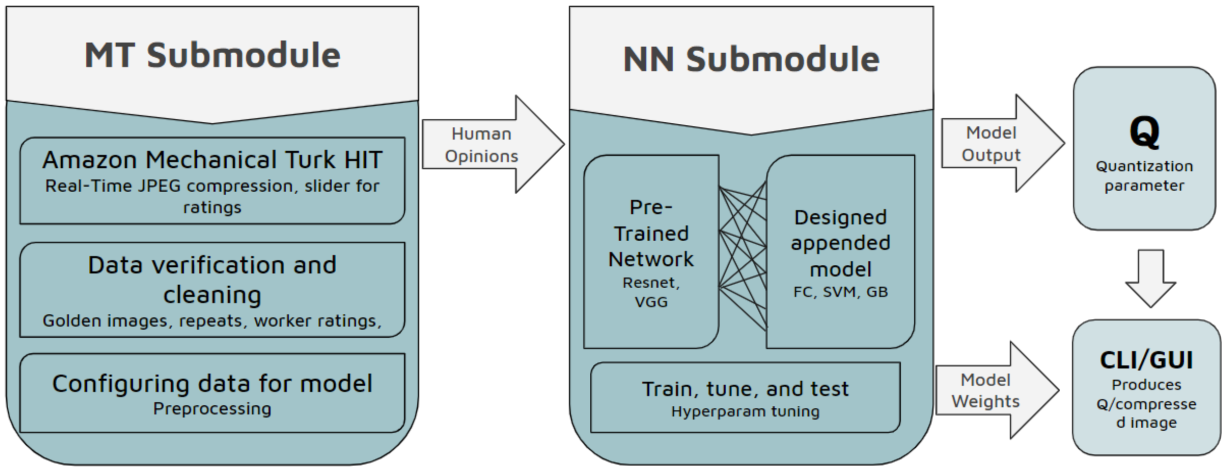


Figure 1. System Block Diagram

3.1.1 Amazon Mechanical Turk Submodule

The MT submodule's inputs are the approximately 8000 images, which we refer to as "the 8k images," that will be shown to the MTurk workers, and the outputs will be the optimal compression levels selected by the MTurk workers. In each human intelligence task (HIT), workers will be presented with a sequence of pages that each show one of the input images, and below the image will be two slider bars. In each HIT, we expect to show the worker around 55 images that are randomly selected from the 8k images. We will also include integrity checks in the study in the form of repeated images and so-called "golden images." These integrity checks are explained in detail in Section 3.1.1.1. With the integrity checks, each HIT will have 5 golden images and 5 repeated images, so we will gather 50 scores from each HIT (since 5 of the images are repeated). In total, we will run the study until we have at least 30 scores per image, at which point we will conclude the data collection portion of the project.

Each slider bar controls the compression level of the input image, but the worker only controls one slider at a time, as seen in Figure 2. When a worker first sees an image, the second slider will be locked and the worker will be asked to move the first slider to the point just before they notice any change between the original image and the displayed compressed image. Then, they will click "Next," which locks the first slider and unlocks the second slider. They will then be asked to move the second slider to the point right before where they think the compression has compromised the quality of the image. Once the worker picks the second slider value and clicks "Next," the worker will move on to the next image.

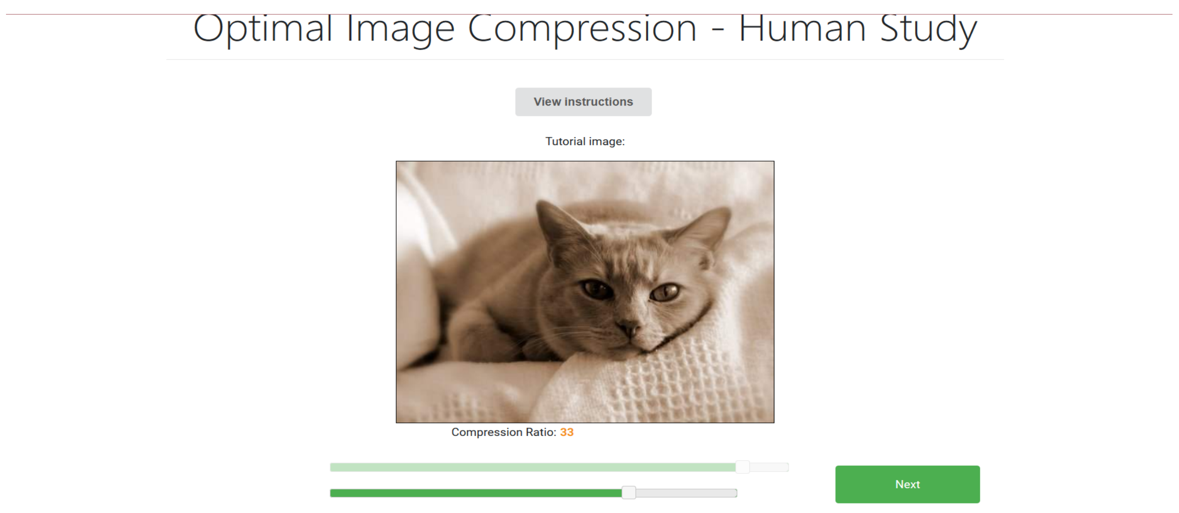


Figure 2. Two Slider HIT Page

We chose to implement two sliders because early tests of using just one slider (the second slider from above) showed that workers are often confused about where we want them to move the slider, and having the extra data from the second slider can also help in our model building. In addition to heavily clarifying the instructions, we added another slider that forces workers to make the distinction between the point where the compressed image looks *different* from the original image and the point where the compressed image looks so different that the compression has now caused the quality to drop. We hope to clarify the notion that an image can look different due to compression but still have the same quality as the original. Furthermore, we may try training on the values gathered from the additional slider, since some companies may wish to find the point where a compressed image looks different from the original.

The MT submodule's input images are a collection of images that are statistically similar to Facebook images. In other words, the distributions of the 8k images' resolution, luminance, contrast, and other image characteristics are similar to those of Facebook images. The average image on Facebook has decent resolution, while some have high resolution and others have low resolution. The resolutions of the 8k images follow a similar distribution as well. Similarity to Facebook images is desirable since one of the major applications of our project is to optimally compress hundreds of millions of social media images. Training our network on images that look similar to social media pictures will help make our model more accurate. Also, these images were provided to us by our faculty mentor's lab, the Laboratory for Image and Video Engineering (LIVE), and we have permission to use these images in our MTurk study.

3.1.1.1 Integrity Checks

In order to ensure that workers are diligently completing our task, we have implemented integrity checks within our HIT page. Namely, 5 of the images will be repeated and 5 of the images will be golden images. A golden image is an image for which we have already determined the optimal compression level, and our golden image data was collected from trusted friends in laboratory-like settings. In total, we have 20 responses for each of our 65 golden images, but we only have responses for the second slider. The golden image study was conducted before adding what is now the first slider, so the golden image integrity checks will only be used on answers for the second slider. Both the repeated images and the golden images serve to ensure that the worker is providing genuine responses and is taking the task seriously. The worker is expected to provide similar scores for each repeated image; workers will be rejected if repeated scores differ too much. Similarly, the worker is expected to provide scores for the golden images where the ranking, or order, or scores correlate well with the ranking of the true golden image scores.

In the repeated images quality check, 5 of the displayed images will appear twice. If the worker is answering the questions diligently, we expect the worker's answered compression levels for each slider to be very similar for each appearance of a repeated image. We will determine if the scores for repeated images are close enough if the difference between the scores is less than a threshold value. We will calculate the threshold value by calculating the standard deviation of all scores for a given golden image. Then, we will average the 65 standard deviations (one for each golden image) to calculate an average standard deviation. This average standard deviation, rounded to the nearest integer value, will be our threshold.

The workers' second slider scores for the golden images are expected to rank similarly with the rankings of the actual golden image scores. In other words, the worker's lowest-scored golden image should be the same image as the actual lowest-scored golden image, and the worker's second-lowest-scored golden image should be the same image as the actual second-lowest-scored golden image, and so on and so forth. The alignment of rank orders can be measured with the Spearman's Rank Correlation Coefficient (SRCC) between the worker's scores for the golden images and the actual golden image scores. The SRCC is a constant between -1 and 1, and if the worker's SRCC score for golden images is below a certain threshold, we will reject the worker. The threshold was determined by taking 5 golden images whose scores are low in variance and contrasted in mean and then calculating the average SRCC value over all participants in the golden image study.

The repeated images will be used for in-session rejection, where we notify the worker that they failed our integrity checks halfway through the task (around image 30). The golden images will be used for post-session rejection, which will occur after workers have submitted the task. In-session rejection will have little negative consequences for both the team and the worker, while the opposite is true for post-session rejection. The worker's HIT approval rate will not drop and the team's rejection percentage will not rise if a worker is rejected in-session. On the other hand, the worker's HIT approval rate and the team's rejection percentage will both increase if they are officially rejected after the HIT is submitted.

We are using repeated images for in-session rejection and golden images for post-session rejection because, according to Professor Bovik, repeated images are a stronger metric and will be able to reject more people. The golden images are a weaker integrity check because the scores were collected in lab-like conditions, which will differ from a typical MTurk worker's environment. Furthermore, the golden image checks will only be valid for the second slider scores. This makes repeated images a more attractive method for in-session rejection since we want to reject the most people in-session, where doing so poses the least amount of negative consequences for both our team and the workers.

3.1.2 Neural Network Submodule

The NN submodule will have different types of inputs and outputs depending on the stage of the project. During training, the NN submodule's input will be the 8k training images and the associated human-selected compression levels, and the submodule's output will be the predicted optimal compression level. The goal of training the network is to iteratively adjust the network's weights so that an image's predicted compression level is close to its optimal compression level. Once the network is trained, the network will be saved and then used to predict compression levels for new candidate images. During this phase, the inputs will be a directory that contains the images that the user wishes to compress optimally and a directory location to store the results. The images cannot be too large (50MP) or too small (256 pixels by 256 pixels), but can otherwise be any size. The outputs will be a file that contains the image names and the associated predicted compression levels and, if specified, the optimally compressed images that have been compressed with the predicted compression parameter.

As mentioned previously, there are multiple approaches to designing our neural network. We are primarily considering two methods: constructing a custom model and transfer learning. The former involves building a fully customized neural network and training all weights in the network ourselves. We would have full control over the architecture and we would be able to extensively finetune the network. However, the disadvantage is that we would need more training images to train an accurate model, which is more expensive to acquire. Transfer learning, on the other hand, involves using a pre-trained network, appending a few layers after it, and training just the layers that we added to tailor the network to our specific task. The advantage of transfer learning is that we would need to learn fewer model parameters compared to training a fully custom network. Therefore, we would need less time, less computational resources, and fewer training images and labels to train the entire network. The disadvantage, however, is that we would lose some control over the architecture of the model.

Transfer learning is based on the assumption that the early layers of a network capture general features of images, such as structures, that are useful in a wide range of learning tasks. In the case of visual data, this relies on the assumption that the images of interest can be reduced to a simpler, lower-dimensional representation without destroying the features of interest.

There are two common methods for transfer learning: finetuning and feature extraction. In feature extraction, one typically removes the last fully connected layer of a pretrained network and replaces it with a new randomly initialized fully connected layer with dimensions corresponding to the new task. During training, the weights of all layers preceding the last are held fixed; only weights of the new final layer are updated. In finetuning, one allows all of the weights to be updated during the training stage. Some weights in the pretrained model can be fixed, the number depending on the size of the new dataset to be trained on, the similarity of the new and old datasets, and the similarity of the new and old tasks. An alternative to attaching a fully connected layer at the end is attaching a Support Vector Machine (SVM). We plan to try this as well.

One idea we have for building a custom neural network is to build a General Adversarial Network (GAN) which differentiates between good and bad quality images (i.e. over-compressed and well compressed images). A GAN has two parts, a generator and a discriminator. For our purposes, the generator would learn to generate both good and bad quality images by compressing images that we feed to it from our dataset to different levels. The discriminator would learn to label these images as good or bad (optimally or non-optimally compressed) based on human opinions of the optimal compression level that we feed it from our dataset. This GAN could be used as a standalone network, but layers from the discriminator could also be used as part of a transfer learning model.

3.2 Risk Reduction

We have conducted research and experiments and successfully determined the usability of TACC for training our model, the inner workings of MTurk for data collection, and which deep learning libraries and methodologies will be used for our final project. The following is what we have done to accomplish this task:

1. As a team, we created an MTurk HIT page. We demonstrated that we could create a page, display images, and implement a slider bar that controls the compression level of the displayed image. The slider bar is a major feature of our HIT page, and demonstrating its feasibility was very important to proving that we could create a viable MTurk study.
2. As a team, we trained small neural networks (both from pretrained models and fully custom networks) on TACC using dummy data and LIVE Subjective Image Database. In doing so, we are now able to train multiple neural networks in parallel through TACC. With more trained neural networks readily available, we can see which of those networks provide the best results - the optimal compression levels closest to those chosen by the humans participating in our MTurk study.

3. As a team, we found sustainable storage and stored the 8k images we got from Professor Bovik there. The images are stored in an Amazon S3 bucket and can be accessed and displayed by the MTurk HIT page. Because of this, we can store a large number of data points from the human study. There are storage and access costs associated with the bucket, but current usage is low enough to qualify for free storage and cheap access costs. The access costs are expected to increase during the duration of the MTurk study. Images stored and accessed through S3 ensures that our HIT page can get the necessary data efficiently for use when our human study begins.
4. Arkan gained access to TACC machines, installed deep learning packages there, and set up his personal environment. All team members made TACC accounts and have been able to log in to Maverick2, TACC's dedicated deep learning machine. Also, the members of the neural network subteam downloaded packages necessary for machine learning. This allows us to build and train models seamlessly as we would have personal hotkeys and shortcuts available in those machines. This ensures that the member's TACC accounts and machines have all the necessary resources and are ready to begin training as soon as we have the human study data.
5. Brandon determined how to transfer data collected from MTurk to the TACC machines and convert it to a form readable by the neural networks. The first step is to make sure the necessary information is written to the output file. Then, after the study concludes, the output file can be downloaded and transferred to TACC via the terminal for Linux or MacOS or an ssh client for Windows. Because of this, we have a robust interface between our MT module and NN module, which ensures the transfer of data between MTurk and NN is smooth and efficient. For more details, see Appendix B.
6. Meenakshi experimented with a toy network with the LIVE Subjective Image Quality Database. Using fast.ai, she appended a pre-trained network with some custom layers and was able to get the network to converge. As a result, we determined the feasibility of our project and know for certain that it can be done.
7. Kan determined how to compress an image on-the-fly to the specified compression level while the slider bar is being moved. Kan's risk reduction activity allows us to save money

on storage costs, since we will not have to precompress each image and store each copy. Instead, we can store only the original and compress the image every time the slider is moved. Furthermore, the HIT page will operate more smoothly, as the compressed image that is displayed will update without requiring that the slider be released. This ensures that we do not need a large amount of space in the S3 bucket for precompressed images, allowing us to allocate more money and resources to the human study.

8. Akash experimented with transfer learning in Python. He used GluonCV to retrain an object detection network so that it can predict an additional class. In doing so, we familiarize ourselves with the implementation of transfer learning such that when we do have to build the final model, we would encounter fewer obstacles along the way.

In doing activities 2 and 4, we have understood how to use TACC to train our models. This is necessary as training a neural network with potentially millions of parameters requires a large amount of computing power and time -- a task unfit for our laptops and home desktops. By doing activities 1 and 3, we familiarized ourselves with MTurk and S3, both of which are a part of Amazon Web Services. We also know how to bridge the data collection side and machine learning side of our project through activity 5, which will prove to be important as any neural network is as good as the data it is provided with. Completing activities 6 and 8 gave us insight on how to create a model that will satisfy the project requirements as defined by Professor Bovik. By experimenting with PyTorch, fast.ai, and GluonCV, we have gained more intuition on transfer learning and how we can reap its benefits for our project. After comparing the libraries, we have decided to use PyTorch as our library of choice due to its extensive documentation and ease of use. GluonCV has limited functionality and fast.ai is less widely used and hence has less useful documentation.

3.3 Test Plan

We will break down our test plan into two components for the two submodules. For the MT submodule, our criteria for success would be to obtain data from those who took our HIT page seriously instead of obtaining data from those who put minimal effort just to get easy money.

Before we launch the true human study on MTurk, we would first run a smaller human study with golden images and repeated images incorporated into the HIT page to see whether our measures to counter falsifying data works and if there may be concerns regarding our study. After we determined the effectiveness of those countermeasures and addressed any issues, we would then launch the real human study.

As for the NN submodule, our criteria for success would be whether our model can accurately predict the correct JPEG compression level based on human perception. To test this, we will partition our collected MTurk data into two types: training data and test data. We will use the former to train our model and then confirm its accuracy on data the model has not seen before using the latter. 80% of the collected MTurk data will be used as the training data, and the remainder will be used as the test data. In our testing phase, we will input the test data into the neural network, carefully keeping track of the human scores of each image and the output scores from our model. If our model can predict the correct compression level of all the test data images with an accuracy above 90%, then our model is successful in its task. Furthermore, we consider a predicted compression level to be correct if it is within +/- 3 quantization levels from the true value, and we define accuracy to be the number of correctly predicted compression levels divided by the total number of test images. If the trained network failed to achieve this level of accuracy, then we would have to modify our model and train our model again until it does satisfy that criteria.

Once we have thoroughly tested the MT and NN submodules, we will test the entire system as a whole. Since our solution is designed to compete with other forms of image compressors, we will also compare the performance of our final predictor with pseudo-perceptual metrics such as SSIM. Our comparison will involve feeding the same images into the two predictors and comparing the results of the images. If the compressed images that are produced by our design solution are on average 5% smaller than images compressed using the SSIM standard, then we will define the test as being successful. The images that we will use for this test will be high quality images since SSIM is a reference metric that compares the luminance, contrast, and structure of small windows across the two images [3] and assumes that the input images are high quality images. Our predictor will also be trained on high quality images, which is ideal for the comparison test.

4.0 PROJECT MANAGEMENT

In order to begin training and testing our model, we need to start collecting data. However, before we open up our HIT on MTurk, we need to collect scores for the golden images by doing a human study. This is the first thing we will do next semester. Additionally, we will need to finalize the HIT instructions based on feedback from participants from the golden image study. Initially after posting our HIT, we will have to monitor the responses to make sure everything is going smoothly. We need to make sure that the golden images are doing their job, preventing people from just selecting random levels, and that the instructions are being interpreted correctly.

Once we have confirmed that we are collecting good quality data that conforms to our metrics, we will set up a pipeline that feeds it to the TACC computers where the neural network is being trained and tested. During training and testing, if the neural network group finds any issues with the data or identifies any changes to the HIT that could potentially improve performance, they will need to relay this information to the MTurk team.

We will continue to refine our neural network until it meets our standard of outperforming similar models trained on scores generated using SSIM. We will need to experiment with different base models and different transfer learning methods to achieve good performance.

4.1 Project Activities

Since our project has two submodules, we will split our work resources evenly between the two submodules MT submodule and NN submodule as described in our work breakdown structure (WBS) in appendix C. What should be noted is that only the MT submodule will use all of the budget as we must employ MTurk workers for the study and allocate space within Amazon S3. Under the MT submodule, there is the MTurk Human Study and the S3 Storage. Under the

MTurk Human Study, we must plan the golden image study, conduct that study, design the HIT page for the MTurk study, build the algorithm that selects which images are presented to the ones undertaking the study, and conducting the study. Designing the HIT page involves designing the mechanism for randomly and uniformly choosing images from the S3 bucket, the specific text instructions that will be displayed to the user, the mechanism of clicking “submit” and showing another image, and the mechanism of stopping a task early if the user fails too many integrity checks. Under the NN Submodule, we must create the model, train it, test it, and build the GUI and CLI. Lastly, under the Model Creation, we must utilize pre-trained models for transfer learning and develop the custom model that will be appended to those models.

In regards to the MT submodule, S3 only takes up 5% of work because setting it up is almost trivial in comparison to getting MTurk set up and working. If we go down another level from MTurk Human Study and compare Conduct Human Study and Golden Image Study, one can see that Golden Image Study takes up 15% of work while Conduct Human Study takes up 20%. This is the case because we do not need to intensive integrity checks in the Golden Image Study whereas in the Conduct Human Study, it is vital to have them. Also, due to the difference in magnitudes, the Conduct Human Study must be treated with absolute care. The other two parts, Designing HIT Page and Image Selector, pertain to the structure of the study itself and how it will determine which images the participants will need to score.

As for the NN submodule, Training and Testing take up an equal amount of work since they draw their data from the same pool (i.e. data from MTurk). Model Creation will take up the most work since we would first lay out the initial architecture and then optimize our model depending on our results from Training and Testing. Optimizations would come from Transfer Learning and Custom Model as we try to find the best configurations between the existing transfer learning models and our custom-built model. Custom Model will take up twice the amount of work as Transfer Learning as creating a custom model is more hands-on than picking the best pre-existing model as our base.

4.2 Task Assignments

The tasks listed in the Linear Responsibility Chart can be divided amongst the MTurk and Neural Network subteams (see Appendix D). Each team member is responsible for their own task. For the most part, the other members of the relevant subteam will be consulted, while the other team members will either be accountable or informed of the task.

We conducted human studies to collect compression scores for the golden images. All team members recruited friends and family members to participate in the studies. Initially, we conducted a study from Monday, February 10th to Tuesday, February 18th with 19 participants. However, we made a mistake when constructing the study and only showed subjects the lower half of the available compression levels. The levels range from 0-100, but we only showed them levels 0-50. We fixed the error and conducted a second golden study from Monday, February 17th until Friday, February 21st with 20 participants. We are using the results from the second golden study in our final HIT page that we will deploy on MTurk on Wednesday, February 26th.

Most of the HIT page has already been built, so we only need to finalize the instructions and test the page before we deploy the study. This should be completed soon, with a target deadline of February 28th. Brandon and Kan have worked most on the HIT page, so they will be responsible for its construction and for selection of images. Akash has also worked on the page, so he can be consulted. Kan will be held responsible for both of these tasks because he is in charge of the funds for the crowdsource study.

Akash, Brandon, Arkan, and Meenakshi all have experience with or interest in machine learning, so they will all work on the pretrained model for which Brandon will have final responsibility. Arkan and Meenakshi will work on building the custom neural network as they have both expressed interest in this task. Meenakshi will be held accountable because she has experience building neural networks from scratch. One of the models will likely be abandoned if it proves to be weaker than the other. The remaining one needs to be finished by the beginning of May before the end of the course.

4.3 Project Schedule

Our schedule for next semester is detailed in our Gantt chart in Appendix E. In the first 3 weeks of the spring semester, from Monday, January 20 to Monday, February 10, we will plan the golden image study. At the same time, we will prepare the HIT page and everything that is needed to run the MTurk study, like storing the 8k images in an Amazon S3 bucket and writing the code that chooses which images will be shown in each HIT. We originally planned to start the golden image study on Monday, February 10, but after completing the study we realized that we made a mistake, and needed to conduct the study again from the beginning. As a result, we updated the Gantt chart to include a Golden Image Study II task, which ran from Monday, February 17 to Friday, February 21. After that, we planned to deploy the HIT page and conduct the human study on Monday, February 24. At this point, the NN subteam will also begin working on designing models. The human study will run for 3 weeks. On March 1, members of the MTurk subteam will start coding the GUI/CLI, since at this point we expect the study to be running fairly smoothly and will require less oversight. At the same time, the NN subteam can begin training, since we will have collected some data at that point. Model creation and training will continue until April 1, at which point most of the NN subteam will begin to focus more on testing the models and evaluating performance. We plan to begin wrapping up the project by the end of April.

4.4 Project Budget

Professor Bovik has agreed to provide $5000 in funding for our project. The only major expense for our project is paying the MTurk workers for completing our HIT (see Appendix F). A second, negligible expense is the cost of sustainable storage for the 8k images that will be used in the MTurk study. We aim to collect 30 responses for each of the 8460 images in the 8k dataset, and we will collect 50 responses total per HIT, so we will need to fund at least 5076 HITS. We plan to pay $0.75 per HIT, so we have budgeted $3807 for paying workers. The other expense for our project is the cost of storing the 8k images in a location that is stable and accessible from the HIT. Using Amazon's own pricing calculator, the cost for S3 storage is expected to be no higher than $0.53 (see Figure 3). Note that the number of PUT and GET requests are estimates of expected usage plus a guardband, so the actual storage cost will likely be lower. We chose Amazon S3 for our sustainable image storage since both S3 and Mechanical Turk are part of the Amazon ecosystem. S3 is reliable, and we have verified that images stored in an S3 bucket can be displayed in an HIT.

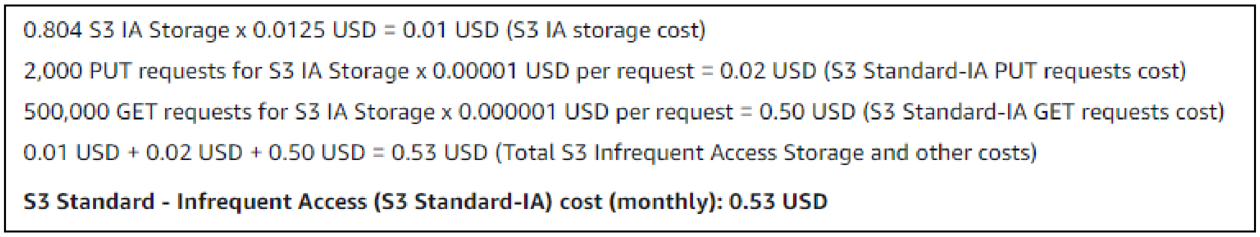

0.804 S3 IA Storage x 0.0125 USD = 0.01 USD (S3 IA storage cost)
2,000 PUT requests for S3 IA Storage x 0.00001 USD per request = 0.02 USD (S3 Standard-IA PUT requests cost)
500,000 GET requests for S3 IA Storage x 0.000001 USD per request = 0.50 USD (S3 Standard-IA GET requests cost)
0.01 USD + 0.02 USD + 0.50 USD = 0.53 USD (Total S3 Infrequent Access Storage and other costs)
S3 Standard - Infrequent Access (S3 Standard-IA) cost (monthly): 0.53 USD

Figure 3. Pricing Calculator for S3 Storage

5.0 CONCLUSION

We have gone into detail the criteria of a successful image compressor and have explained the motivations, methods, goals, target users, and challenges regarding the design. We have shown the feasibility of our design through our risk reduction research, experiments, and activities, which further ensure the success of our project. Two of those risk reduction activities were designing the HIT page and training multiple neural networks in parallel. Designing the HIT page gave us the necessary intuition on how the data can be collected using MTurk as well as verify that an image compression slider can be implemented in real-time on the page. Similarly, training multiple neural networks in parallel proved to us that we can test and consider many neural networks with the given time remaining to complete our project. We have a comprehensive plan, schedule, budget, and list of materials for our project that will guide and constantly push us forward.

Through these topics we have covered, it is clear we prepared a strong foundation for the semester to come. As previously mentioned, our plan is to spend a month each to conduct the MTurk study starting on February 1st and begin training our models starting March 1st. We will also evaluate our models on April 1st, and finally wrap up the project by the end of the month. However, these dates are not set in stone and can be shifted to accommodate certain changes. For example, if the MTurk data collection process is taking longer than we anticipated and given that we have constructed a working model, we can accommodate by training our models in conjunction with the data collection since the two processes do not necessarily have to be executed linearly. Inversely, if we are able to collect MTurk data very quickly, all of the timeline can be shifted forward, which may create more time for testing and improving the GUI or project presentation. If all activities are executed on schedule and we complete each milestone according to our plan, then we are sure to create a strong image compressor.

REFERENCES

[1] Lin, Weisi, and C-C. J. Kuo. "Perceptual visual quality metrics: A survey." *Journal of visual communication and image representation* 22.4, 2011, pp. 297-312.

[2] Mazen Abuzaher, Jamil Al-Azzeh (2017). JPEG Based Compression Algorithm. *International Journal of Engineering and Applied Sciences*, 4(4), pp.95-97.

[3] Wang, Z., Bovik, A., Sheikh, H. and Simoncelli, E. (2004). Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4), pp.600-612.

[4] CCITT Study Group VIII and the Joint Photographic Experts Group. T.81 – DIGITAL COMPRESSION AND CODING OF CONTINUOUS-TONE STILL IMAGES – REQUIREMENTS AND GUIDELINES" (PDF). CCITT. September 1992. Retrieved 25 February 2020.

APPENDIX A: RELEVANT STANDARDS

In 1992, the Joint Photographic Experts Group published the JPEG standard, which sets out guidelines and requirements for encoding and decoding of continuous-tone still images [4]. It specifies the processes for compressing a source image, storing compressed image data, and reconstructing an image from compressed image data.

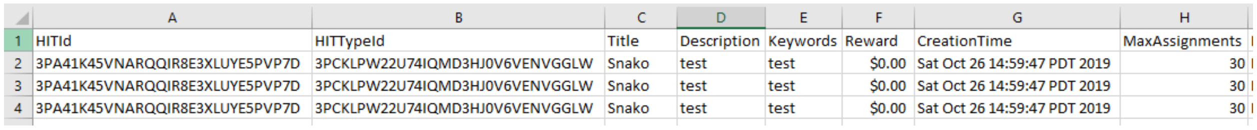
APPENDIX B: RISK REDUCTION ACTIVITY DOCUMENTATION

Activity 5 Documentation: MTurk to TACC

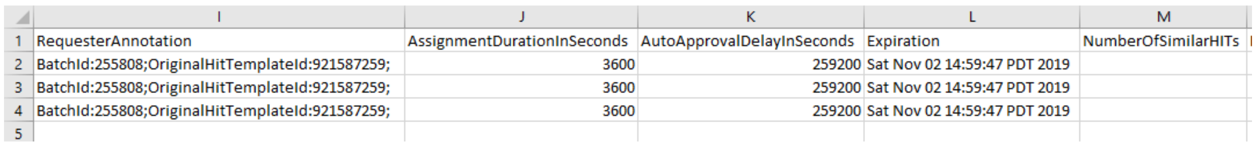
1. When creating the MTurk page, make sure that the image name is written to the output .csv file by inserting a hidden crowd-input element with the name of the displayed image. This can be accomplished in Javascript by placing the following line in a <script></script> block:

   ```
           document.write("<crowd-input name='img_name' value=" + name_of_image + "
   type='hidden'></crowd-input>");
   ```

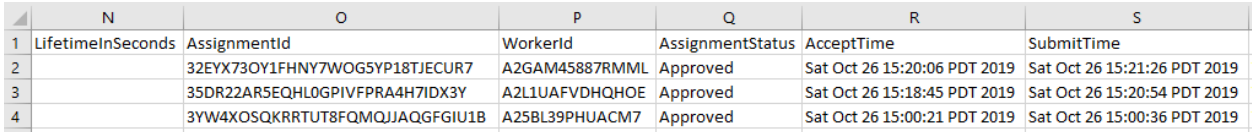
   where "name_of_image" should be replaced with the actual name of the image. This line will result in an "Answer.img_name" column in the output .csv file.



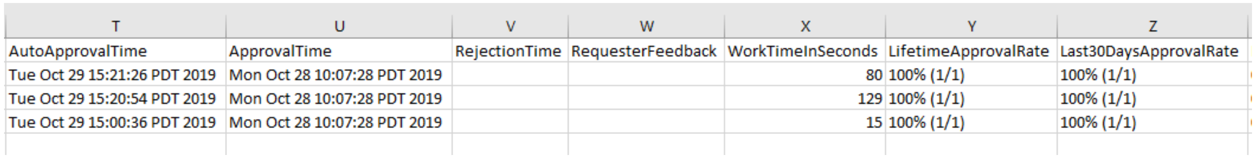
	A	B	C	D	E	F	G	H
1	HITId	HITTypeId	Title	Description	Keywords	Reward	CreationTime	MaxAssignments
2	3PA41K45VNARQQIR8E3XLUYE5PVP7D	3PCKLPW22U74IQMD3HJ0V6VENVGGLW	Snako	test	test	$0.00	Sat Oct 26 14:59:47 PDT 2019	30
3	3PA41K45VNARQQIR8E3XLUYE5PVP7D	3PCKLPW22U74IQMD3HJ0V6VENVGGLW	Snako	test	test	$0.00	Sat Oct 26 14:59:47 PDT 2019	30
4	3PA41K45VNARQQIR8E3XLUYE5PVP7D	3PCKLPW22U74IQMD3HJ0V6VENVGGLW	Snako	test	test	$0.00	Sat Oct 26 14:59:47 PDT 2019	30



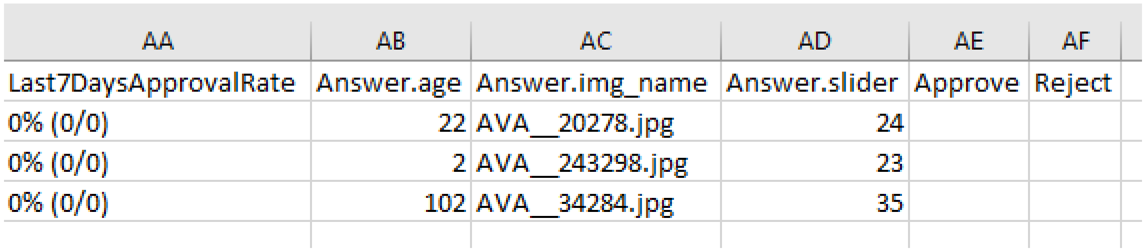
	I	J	K	L	M
1	RequesterAnnotation	AssignmentDurationInSeconds	AutoApprovalDelayInSeconds	Expiration	NumberOfSimilarHITs
2	BatchId:255808;OriginalHitTemplateId:921587259;	3600	259200	Sat Nov 02 14:59:47 PDT 2019	
3	BatchId:255808;OriginalHitTemplateId:921587259;	3600	259200	Sat Nov 02 14:59:47 PDT 2019	
4	BatchId:255808;OriginalHitTemplateId:921587259;	3600	259200	Sat Nov 02 14:59:47 PDT 2019	
5					



	N	O	P	Q	R	S
1	LifetimeInSeconds	AssignmentId	WorkerId	AssignmentStatus	AcceptTime	SubmitTime
2		32EYX73OY1FHNY7WOG5YP18TJECUR7	A2GAM45887RMML	Approved	Sat Oct 26 15:20:06 PDT 2019	Sat Oct 26 15:21:26 PDT 2019
3		35DR22AR5EQHL0GPIVFPRA4H7IDX3Y	A2L1UAFVDHQHOE	Approved	Sat Oct 26 15:18:45 PDT 2019	Sat Oct 26 15:20:54 PDT 2019
4		3YW4XOSQKRRTUT8FQMQJJAQGFGIU1B	A25BL39PHUACM7	Approved	Sat Oct 26 15:00:21 PDT 2019	Sat Oct 26 15:00:36 PDT 2019



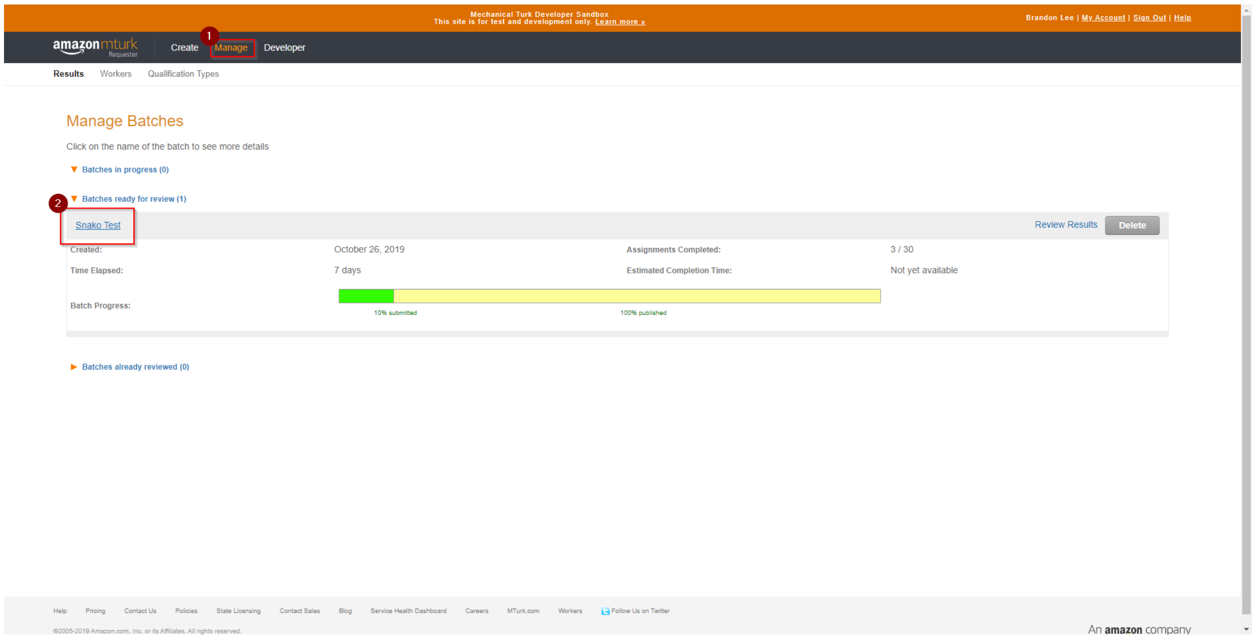
T	U	V	W	X	Y	Z
AutoApprovalTime	ApprovalTime	RejectionTime	RequesterFeedback	WorkTimeInSeconds	LifetimeApprovalRate	Last30DaysApprovalRate
Tue Oct 29 15:21:26 PDT 2019	Mon Oct 28 10:07:28 PDT 2019			80	100% (1/1)	100% (1/1)
Tue Oct 29 15:20:54 PDT 2019	Mon Oct 28 10:07:28 PDT 2019			129	100% (1/1)	100% (1/1)
Tue Oct 29 15:00:36 PDT 2019	Mon Oct 28 10:07:28 PDT 2019			15	100% (1/1)	100% (1/1)



AA	AB	AC	AD	AE	AF
Last7DaysApprovalRate	Answer.age	Answer.img_name	Answer.slider	Approve	Reject
0% (0/0)	22	AVA__20278.jpg	24		
0% (0/0)	2	AVA__243298.jpg	23		
0% (0/0)	102	AVA__34284.jpg	35		

Note that the columns with the "Answer." prefix are values provided by the MTurk workers.

2. Go to the MTurk Requester portal, go to the “Manage” tab, and select the MTurk HIT.


Mechanical Turk Developer Sandbox
This site is for test and development only. Learn more »
Brandon Lee | My Account | Sign Out | Help
amazon mturk Requester
Create
1
Manage
Developer
Results
Workers
Qualification Types

Manage Batches

Click on the name of the batch to see more details

▼ Batches in progress (0)

▼ Batches ready for review (1)

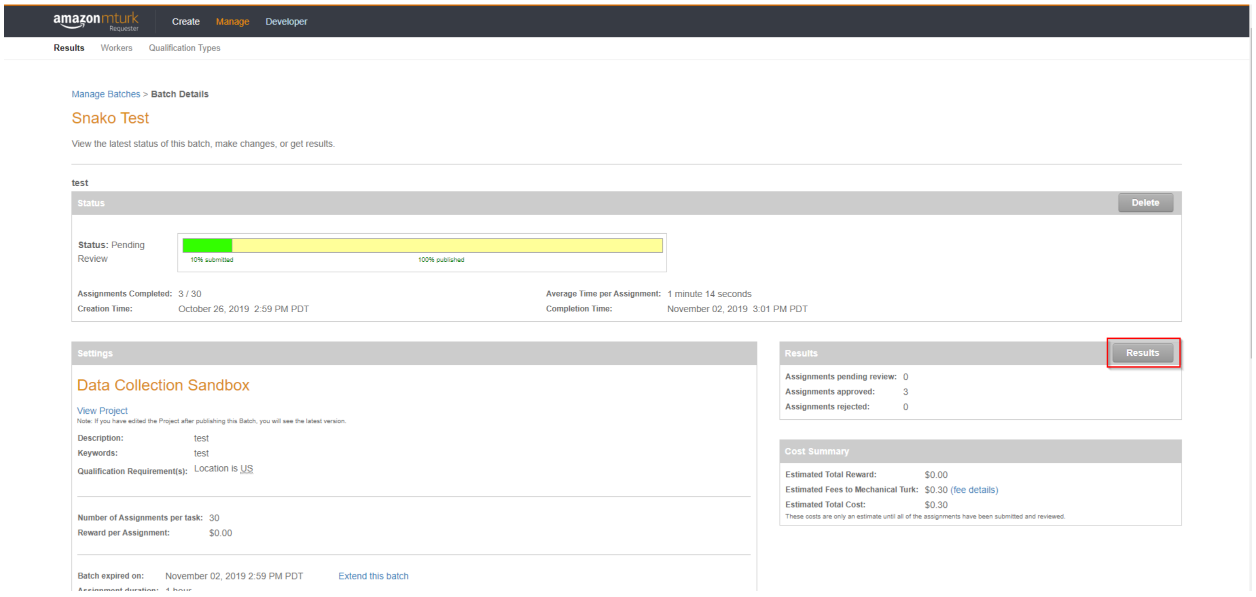
Snako Test — Review Results — Delete

Created:	October 26, 2019	Assignments Completed:	3 / 30
Time Elapsed:	7 days	Estimated Completion Time:	Not yet available
Batch Progress:	10% submitted, 100% published		

▶ Batches already reviewed (0)

©2005-2019 Amazon.com, Inc. or its Affiliates. All rights reserved.

3. Click on “Results”, click on “Download CSV”, and then download the .csv file. Save the file to a location where you can find the file again.



Manage Batches > Batch Details

Snako Test

View the latest status of this batch, make changes, or get results.

test

Status — Delete

Status: Pending Review — 10% submitted, 100% published

Assignments Completed:	3 / 30	Average Time per Assignment:	1 minute 14 seconds
Creation Time:	October 26, 2019 2:59 PM PDT	Completion Time:	November 02, 2019 3:01 PM PDT

Settings

Data Collection Sandbox

View Project

Note: If you have edited the Project after publishing this Batch, you will see the latest version.

Description:	test
Keywords:	test
Qualification Requirement(s):	Location is US
Number of Assignments per task:	30
Reward per Assignment:	$0.00
Batch expired on:	November 02, 2019 2:59 PM PDT — Extend this batch
Assignment duration:	1 hour

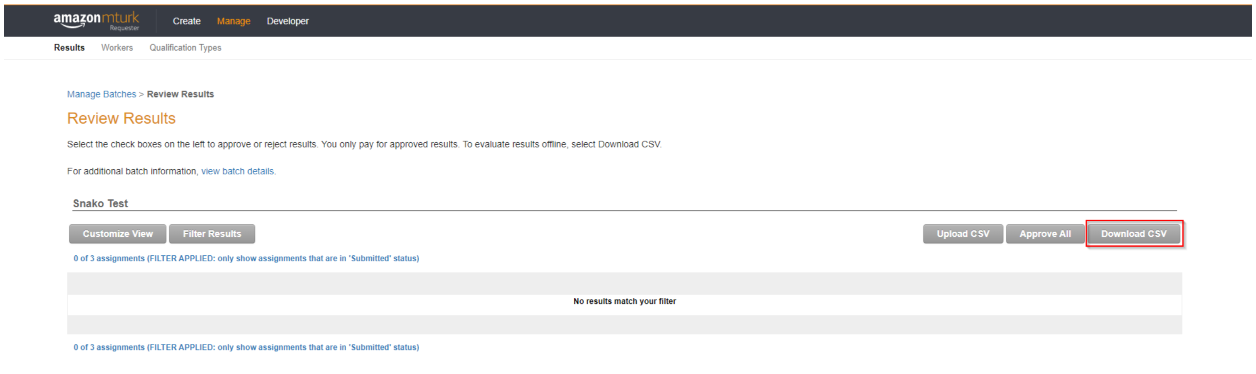
Results — Results

Assignments pending review:	0
Assignments approved:	3
Assignments rejected:	0

Cost Summary

Estimated Total Reward:	$0.00
Estimated Fees to Mechanical Turk:	$0.30 (fee details)
Estimated Total Cost:	$0.30

These costs are only an estimate until all of the assignments have been submitted and reviewed.



Manage Batches > Review Results

Review Results

Select the check boxes on the left to approve or reject results. You only pay for approved results. To evaluate results offline, select Download CSV.

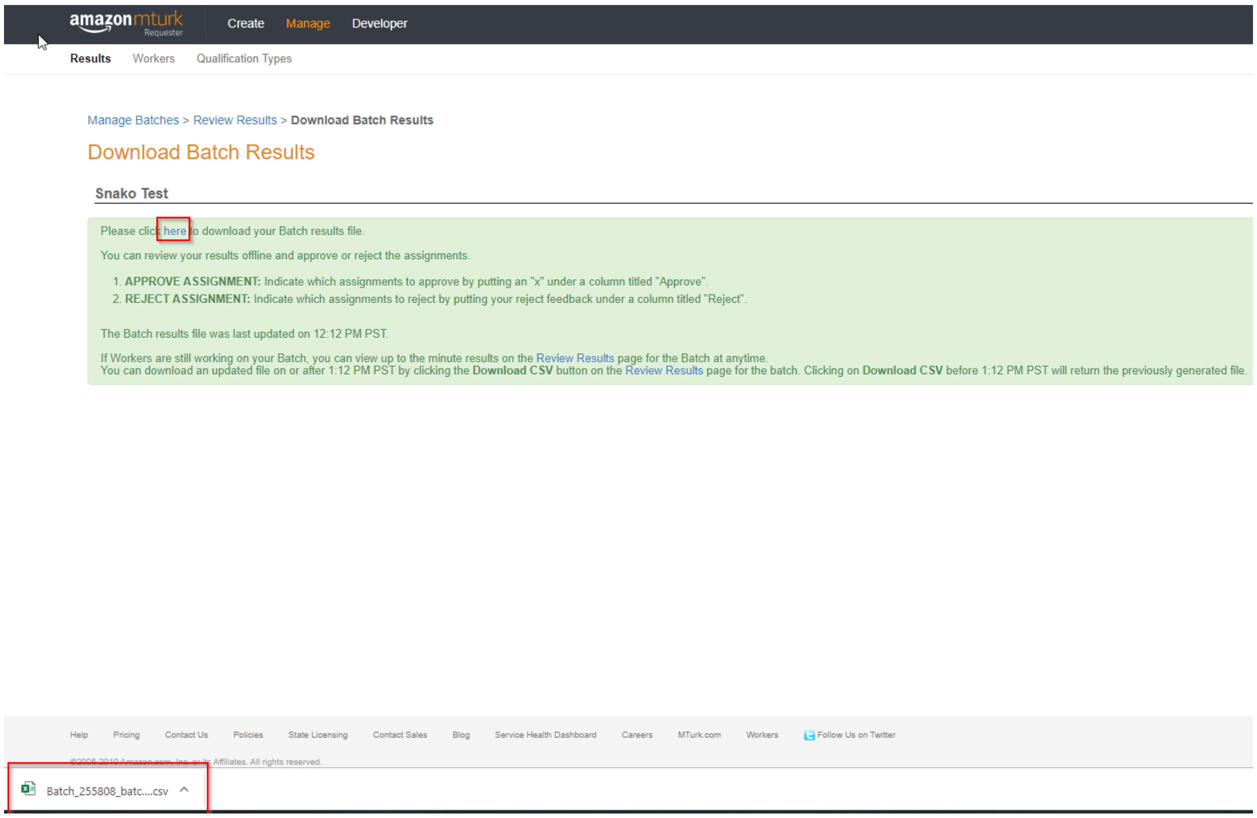
For additional batch information, view batch details.

Snako Test

0 of 3 assignments (FILTER APPLIED: only show assignments that are in 'Submitted' status)

No results match your filter

0 of 3 assignments (FILTER APPLIED: only show assignments that are in 'Submitted' status)



Manage Batches > Review Results > Download Batch Results

Download Batch Results

Snako Test

Please click here to download your Batch results file.

You can review your results offline and approve or reject the assignments.

1. **APPROVE ASSIGNMENT:** Indicate which assignments to approve by putting an "x" under a column titled "Approve".
2. **REJECT ASSIGNMENT:** Indicate which assignments to reject by putting your reject feedback under a column titled "Reject".

The Batch results file was last updated on 12:12 PM PST.

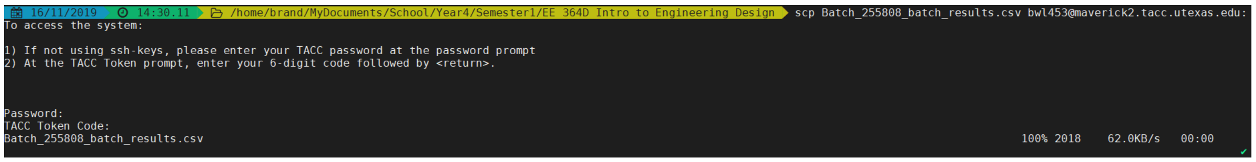
If Workers are still working on your Batch, you can view up to the minute results on the Review Results page for the Batch at anytime.
You can download an updated file on or after 1:12 PM PST by clicking the **Download CSV** button on the Review Results page for the batch. Clicking on **Download CSV** before 1:12 PM PST will return the previously generated file.

©2005-2019 Amazon.com, Inc. or its Affiliates. All rights reserved.

4. Transfer the file from your local system to Maverick2 on TACC. On Windows, this can be accomplished using an ssh client. On MacOS and Linux, this can be accomplished in the terminal. Run the command:

   scp file_path [tacc_username]@maverick2.tacc.utexas.edu:dest_file

   Enter in your TACC credentials when prompted.



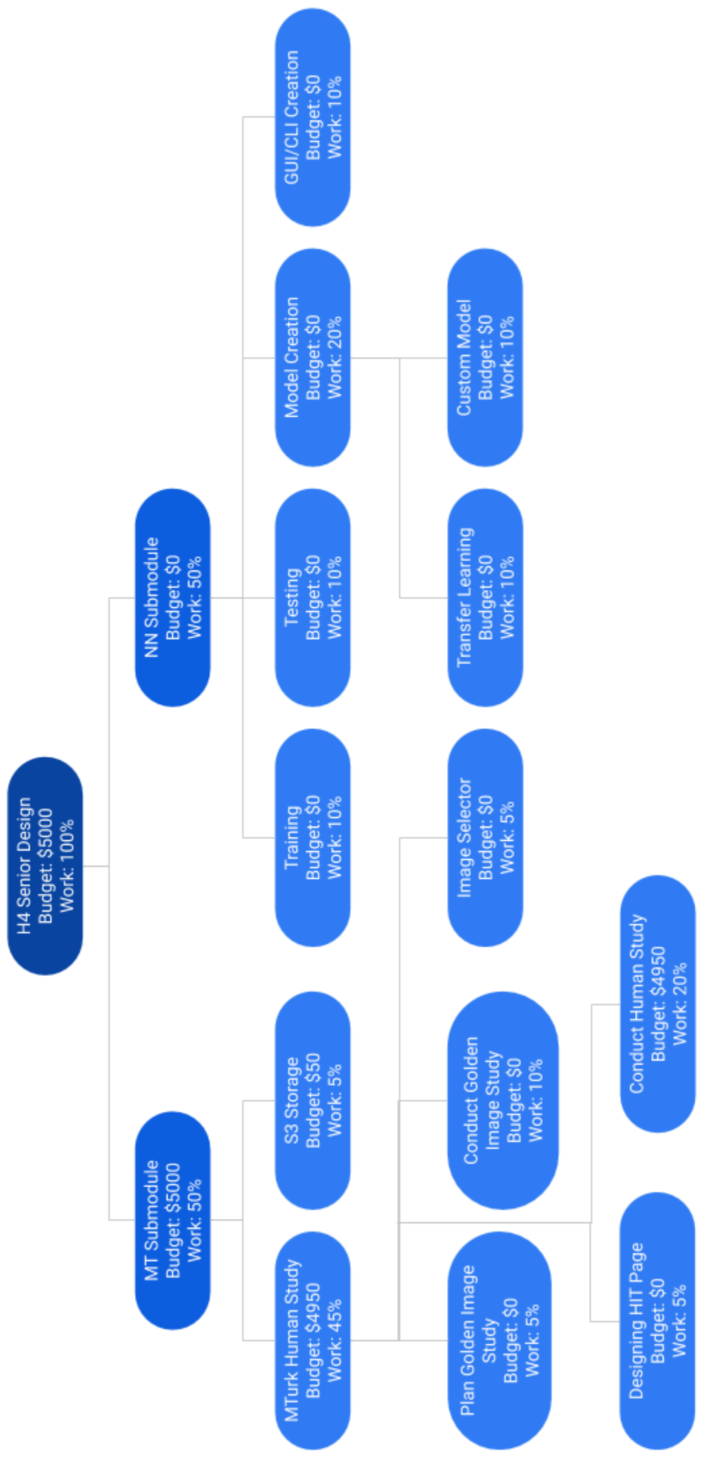
```
 16/11/2019  14:30.11  /home/brand/MyDocuments/School/Year4/Semester1/EE 364D Intro to Engineering Design  scp Batch_255808_batch_results.csv bwl453@maverick2.tacc.utexas.edu:
To access the system:

1) If not using ssh-keys, please enter your TACC password at the password prompt
2) At the TACC Token prompt, enter your 6-digit code followed by <return>.


Password:
TACC Token Code:
Batch_255808_batch_results.csv                                                                            100% 2018    62.0KB/s   00:00
```

5. Move the file to a location in TACC where the team can access the file.

APPENDIX C: WORK BREAKDOWN STRUCTURE

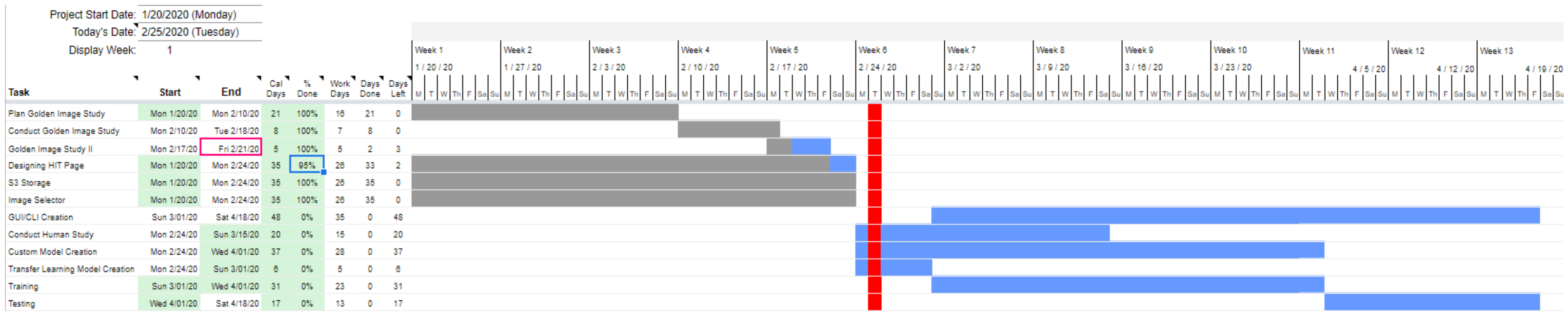

H4 Senior Design Budget: $5000 Work: 100%
MT Submodule Budget: $5000 Work: 50%
NN Submodule Budget: $0 Work: 50%
MTurk Human Study Budget: $4950 Work: 45%
S3 Storage Budget: $50 Work: 5%
Training Budget: $0 Work: 10%
Testing Budget: $0 Work: 10%
Model Creation Budget: $0 Work: 20%
GUI/CLI Creation Budget: $0 Work: 10%
Plan Golden Image Study Budget: $0 Work: 5%
Conduct Golden Image Study Budget: $0 Work: 10%
Image Selector Budget: $0 Work: 5%
Transfer Learning Budget: $0 Work: 10%
Custom Model Budget: $0 Work: 10%
Designing HIT Page Budget: $0 Work: 5%
Conduct Human Study Budget: $4950 Work: 20%

APPENDIX D: LINEAR RESPONSIBILITY CHART

Task	Akash Shukla	Arkan Abuyazid	Brandon Lee	Kan Vanthanasuksan	Meenakshi Swaminathan
Conduct Golden Image Study	R	R	A	R	R
Construct HIT Page	C	I	R	A	I
Select Images for HIT	C	I	R	A	I
Build Custom Network	C	R	C	I	A
Transfer Learning	R	R	A	I	R

R=Responsible, A=Accountable, C=Consulted, I = Informed

APPENDIX E: GANTT CHART



Project Start Date: 1/20/2020 (Monday)
Today's Date: 2/25/2020 (Tuesday)
Display Week: 1

Task	Start	End	Cal Days	% Done	Work Days	Days Done	Days Left
Plan Golden Image Study	Mon 1/20/20	Mon 2/10/20	21	100%	16	21	0
Conduct Golden Image Study	Mon 2/10/20	Tue 2/18/20	8	100%	7	8	0
Golden Image Study II	Mon 2/17/20	Fri 2/21/20	5	100%	5	2	3
Designing HIT Page	Mon 1/20/20	Mon 2/24/20	35	95%	26	33	2
S3 Storage	Mon 1/20/20	Mon 2/24/20	35	100%	26	35	0
Image Selector	Mon 1/20/20	Mon 2/24/20	35	100%	26	35	0
GUI/CLI Creation	Sun 3/01/20	Sat 4/18/20	48	0%	35	0	48
Conduct Human Study	Mon 2/24/20	Sun 3/15/20	20	0%	15	0	20
Custom Model Creation	Mon 2/24/20	Wed 4/01/20	37	0%	28	0	37
Transfer Learning Model Creation	Mon 2/24/20	Sun 3/01/20	6	0%	5	0	6
Training	Sun 3/01/20	Wed 4/01/20	31	0%	23	0	31
Testing	Wed 4/01/20	Sat 4/18/20	17	0%	13	0	17

APPENDIX F: BILL OF MATERIALS

Item to Purchase	**Description**	**Amount Needed**	**Estimated Cost**	**Performance Criteria**	**Total Cost**
MTurk HITs	Pay Turk workers to complete our HIT	5076 HITs	$0.75/HIT	HITs need to pass integrity checks. Will not pay workers that fail checks.	$3807
S3 storage	Pay for S3 bucket to store the images that will be displayed in our HITs	Storage, database operations for 8529 images and miscellaneous files.	$0.53	Need to be able to efficiently and reliably access images in the bucket from the HIT page.	$0.53

Total cost: $3807.53