Improve Decision Support System Operations of Real-Time Image Classification Utilizing Machine Learning and Knowledge Evolution

David Prairie
Electrical and Computer Engineering Dept.
University of Massachusetts Dartmouth
Dartmouth, Massachusetts United States
Email: dprairie@umassd.edu

Paul Fortier
Electrical and Computer Engineering Dept.
University of Massachusetts Dartmouth
Dartmouth, Massachusetts United States
Email: pfortier@umassd.edu

***Abstract*—This paper delves into the generation and use of image classification models in a real-time environment utilizing machine learning. The ImageAI framework was used to generate a list of models from a set of training images and also for classifying new images using the generated models. Through this paper, previous research projects and industry programs are analyzed for design and operation. The basic implementation results in models that classify new images correctly the majority of the time with a high level of confidence. However, almost a quarter of the time the models classify images incorrectly. This paper attempts to improve the classification accuracy and improve the operational efficiency of the overall system as well.**

Keywords-component; Machine Learning; Tensor Flow; Image Classification.

I. INTRODUCTION

Presented in this research paper is a new and novel approach to machine learning design that maximizes the balance between accuracy, efficiency, solution justification, and rule evolution. This paper is a more complete and informative description of the research presented at the IARIA Data Analytics conference [1]. This research improves four factors of machine learning within a single design. During this research, traditional open source machine learning methodologies were altered to improve different aspects of machine learning, tested against a single application. These traditional methodologies were integrated using ensemble methods for enhancing the performance along with providing justifications for the classification results. This paper discusses the results, data used, the analysis, and validation methodologies used.

This research attempted to find an optimal balance between maximizing a system's accuracy and minimizing a system's time and space requirements. As shown in Figure 1, these three attributes are directly correlated with each other and the system integrator needs to determine the trade-off required for the implemented system.

During this research the Identifiable Professionals (IdenProf) Dataset was used for training and validating models. The dataset contains ten image sets of distinguishable professions, each set containing 900 images for training and 200 images for validation per profession. The expected outcome of this research is a two part system capable of analyzing a real-time feed and perform profession classification based on a remotely generated model.

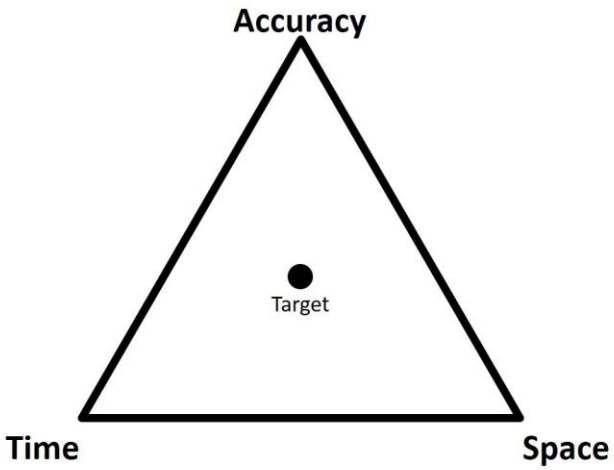

Figure 1: System Efficiency Tradeoffs

This research aimed at answering the following four questions based upon improved decision support system operations. Investigations and implementations conducted comprised of different components of the Decision Support System (DSS). This research reviewed prior research in the machine learning technical field and built upon previous research to answer the following hypothesis:

1. Utilizing a knowledge base, can a DSS be implemented to minimize the time and space requirements, while maximizing the accuracy of the suggested solutions?
2. How can a knowledge base be implemented to improve the overall accuracy of the system?
3. Is a DSS able to provide solution justification to the user, enabling users to have trust in the answers being provided?

4. Is it possible to improve rule evolution without needing to store locally all previous cases for doing rule re-validations?

This paper is broken up into a Background section, where high-level aspects of machine learning and knowledge-bases are discussed. The Methodology is discussed following the Background section. Following the Methodology section the preliminary results are discussed in the Data Analytics and Results sections. Finally, the conclusion discusses planned future work and recommended further work.

II. BACKGROUND

Knowledge base systems enable problems to be quickly and accurately solved based on previous cases. Knowledge base systems also allow for problem solving of very complex situations at speeds humans alone would not be able to achieve at such accuracies. Throughout the last 20 years, knowledge bases have grown in applications to assist in everyday tasks. More recently, IBM's Watson, an Artificial intelligence supercomputer, is in the limelight through its uses in the medical arena and in personal taxes with H&R Block. Other applications of machine learning have been completed or are being developed include detecting insider threats, big data analytics, market analysis for proposals, condition-based maintenance, and diagnostics in the medical field.

In the medical industry, Watson has proven capable of making the same recommended treatment plans as doctors 99% of the time. Unlike traditional human doctors, Watson can use all available medical resources when making a patient's diagnosis. By having such vast amounts of knowledge, Watson can provide treatment options doctors may miss. Watson utilizes powerful algorithms and immense computing resources to analyze all medical relevant data to find "…treatment options human doctors missed in 30 percent of the cases" [3]. Since Watson has so much computing power it is able to determine treatment plans for patients faster than human doctors could, allowing doctors to put patients on treatments faster with the intervention of Watson. Watson is one example of how knowledge base systems positively benefit society by efficient and accurate problem solving of complex problems. Additional research into knowledge bases will allow them to problem solve faster, with more accuracy, and with less compute power requirements.

Condition-Based Maintenance, shown in Figure 2, is the method of monitoring a system's components to determine what level of maintenance is required for the system to remain functioning. Using a Condition-Based Maintenance system for managing maintenance events allows professionals to be proactive with performing maintenance activities versus being reactive. Reactive maintenance involves replacing components after they already failed, causing system downtime. When a system goes down for unscheduled maintenance or repairs, many different repercussions can occur depending on the affected system's role. Using air traffic control centers as an example, any unplanned downtime has the potential to disrupt hundreds of flights and cost a significant amount of money due to flight delays [12]. Machine learning can be used to assist professionals to determine optimal maintenance schedules while minimizing system down time.

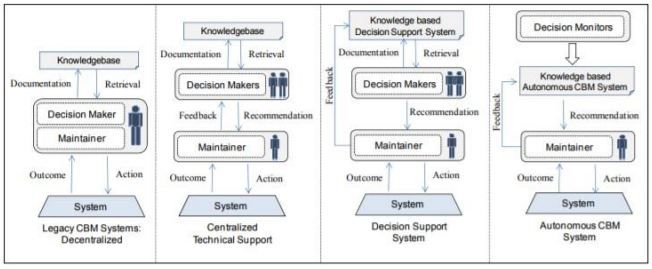

Figure 2: Condition Based Maintenance [2]

Although some of the described applications may not be as drastic as life or death medical decisions, all still can greatly affect society. Utilizing machine learning allows organizations to detect threats, conduct predictive maintenance, and perform many repeatable decision-making tasks consistently and efficiently. By allowing a machine to learn over time through historical cases and building a knowledge base, the machine allows operators to make informed decisions by providing every available piece of information. Systems are able to make decisions in a fraction of the time compared to a human expert attempting to come to the same decision, however additional advancement is needed to make machine learning more accurate and efficient. Areas needing additional inquiries include indexing algorithms, storage solutions, and finally the decision-making algorithms themselves. Machine learning is important because of the wide range of applications and benefits provided through the decision making and predictions capable. As the field advances, machines will create predictions and perform decision making faster and more completely.

A. *Watson*

Watson originally became well-known for competing in Jeopardy. Watson is a knowledge base altered for various applications including Jeopardy, medical field, and taxes. Breaking down questions from a complex human language was required for Watson to compete at the Jeopardy game [3]. The analysis of the Jeopardy questions and identifying the correct answer needed to happen almost instantaneously to compete on a high caliber level.

With the Jeopardy Challenge, Watson needed to break down questions out of human language to a format Watson could understand. The questions needed to break down into the main statement and then separate supporting statements out. "…decompose the question into these two parts and ask for answers to each one, we may find that the answer common to both questions is the answer to the original clue" [3].

In recent years, IBM altered Watson to handle taxes by collaborating with H&R Block. Although there is not much technical information available discussing the design of

Watson's work with taxes, there are a few assumptions that can be made. We would assume Watson uses a rule and case-based design. The rules would take in data on a new client, which determine what tax actions could take place. Watson would compare the new client to all previous clients allowing for more accurate and consistent tax evaluation.

B. *Deep Learning Frameworks – Tensor Flow*

Tensor Flow is a deep learning framework built on the first generation framework called DistBelief. Both frameworks were developed by Google to advance technology for the public and for use in Google's wide range of data products [4]. One of TensorFlow's major improvements over DistBelief is its ability to scale up onto large distributed hardware platforms utilizing multiple CPUs and GPUs. Tensor Flow utilizes a master orchestrator to distribute work across the number of hardware platforms available, each individual platform then breaks the work down to be solved across each system's available CPUs and GPUs.

Benchmarks conducted by Google researchers showed the Tensor Flow framework performs, as well as other popular training libraries. However, Tensor Flow did not have the best performance statistics as other libraries in the study when tested on a single machine platform [9]. Researchers at Google are continuing development in the Tensor Flow framework to incorporate additional optimization and automation to improve the performance of the framework.

C. *Rule Evolution IB1 & IB2 Algorithms*

The IB1 and IB2 algorithms are used to evolve a system's rules used for classification by incorporating new cases. The addition of more instances over time causes the machine to alter its rules to improve the probability of giving a correct prediction on future instances. Instances can either enforce existing rules or go against existing rules. Over the course of a training period, the IB1 algorithm will converge to the actual results based on altering its rules. IB1 requires data to have specific attributes, making cases distinct enough for the algorithm to learn over time. If the data does not have distinct attributes then the machine will not learn, since no strong points of comparison are available between cases [5].

A downside of the IB1 algorithm is the need to store all correct and incorrect classifications over the lifetime of the machine. The IB2 algorithm is a branch of the IB1 algorithm that does not require the storage of all classifications, only the incorrect classifications. The tradeoff of saving storage space is the increase in time required for the IB2 algorithm to learn to predict with strong accuracy [5].

During the evaluation of both the IB1 and IB2 algorithms, researchers determined both algorithms are able to achieve acceptable prediction accuracies in some situations. However, IB1 attains greater accuracies on each scenario when compared to the IB2 algorithm. The increase in accuracy for IB1 could be attributed to the storing of all classification events versus only the incorrect classifications.

D. *Ensemble Method*

Ensemble methods is the practice of implementing multiple machine learning algorithms and incorporating an additional algorithm to vote the responses to a single response. "Ensemble methods are learning algorithms that construct a set of classifiers and then classify new data points by taking a (weighted) vote of their predictions" [6]. Ensemble methods help to remove model bias and overconfidence for models against specific applications. "Ensemble methods are meta-algorithms that combine several machine learning techniques into one predictive model in order to decrease variance (bagging), bias (boosting), or improve predictions (stacking)" [7].

Figure 3: Weather Model Example [8]

Ensemble methods are used in other industries; for example meteorologist compare numerous models to generate a cone of certainty for hurricane predictions. Figure 3 shows an example of a cone of certainty track for hurricane Dorian in 2019. This cone was generated by averaging multiple hurricane track models to a single cone. The cone shows decreasing assurance in the accuracy the further in time of the prediction.

One ensemble method, the Bayesian method, is a common practice of integrating multiple classifiers into a single classifier. The Bayesian method utilizes a weighted average method for determining the proper response. However, researchers from the University of Washington found the Bayesian method has a higher rate of error than other methods of ensemble [9].

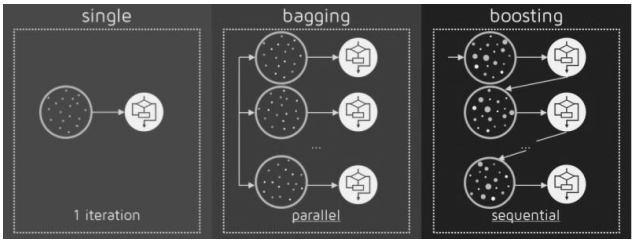

Figure 4: Ensemble Voting Methods [10]

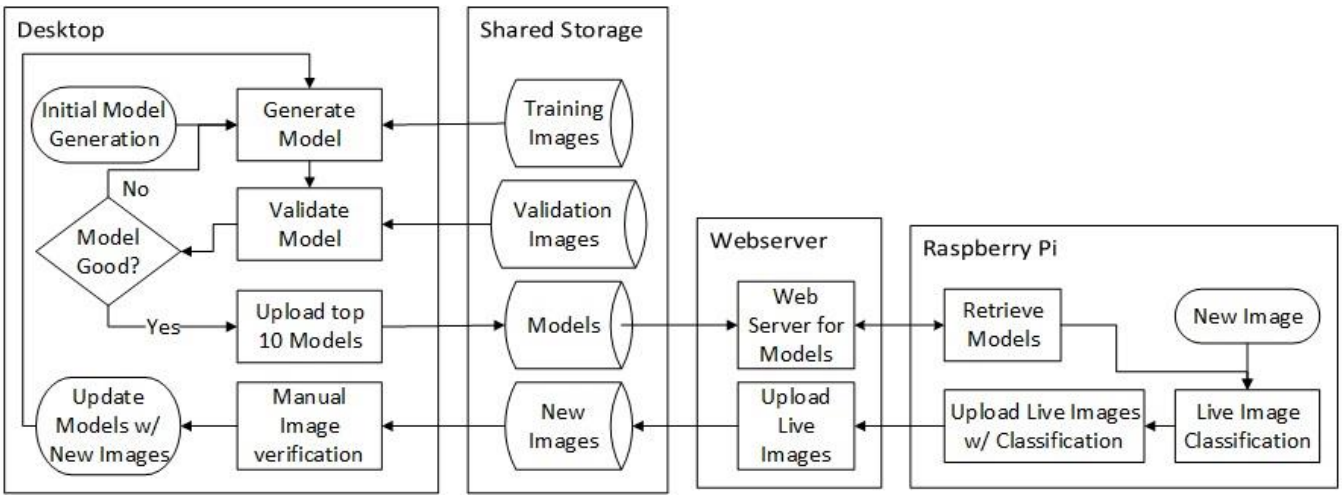

Figure 5: System Flow Diagram

Another method used when incorporating ensemble with machine learning is the bagging method. Bagging involves multiple training models on different subsets of data and integrating the outputs from each model to produce a single result [7]. Boosting is another method for using numerous predictions to create a single result. Both the Bayesian and bagging methods are depicted in Figure 4.

III. METHODOLOGY/THEORY

This section discusses the methodology used in the completion of this research. The research process followed the system engineering v-diagram during development, as shown in Figure 6.

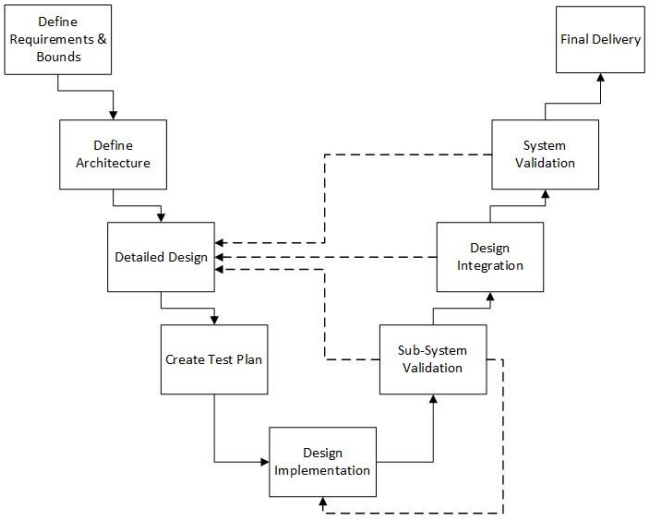

Figure 6: Design Methodology

This section is further broken into a system architecture and software architecture sections.

A. *System Architecture*

The implementation, which is shown in Figure 5, is broken into four sections; a workstation computer, web server, raspberry pi, and a shared storage box. The workstation computer contains an NVidia GTX 980ti and is used for generating models based on the training images. Once the models are generated they are stored on a centralized shared storage array.

The model generator portion of the system handles ingesting a multitude of images and generating 200 models per generation algorithm. The model generation is a compute-intensive operation, taking about 90 seconds per model at 200 models per algorithm to run, and requires a high level of resources to provide accurate results. Additional hardware options were investigated, including Amazon's Web Service and Digital Ocean's droplets. Both of these alternatives allow users to utilize a pay by use virtual Linux environment having a wide range of hardware scaling options. These options were not chosen to eliminate variables introduced by relying on another company's infrastructure.

A Raspberry Pi 3 B+ [11] was used for real-time image classification utilizing an onboard camera. During the initial testing and validation, the model generator was used. The model generator was capable of classifying a large number of images in rapid succession to validate the improvements implemented. The Raspberry Pi was best suited for completing single image classifications.

During initial testing and validation, the Model Generator was also used in place of the Raspberry Pi. Connected to the workstation is a networked HDD used for storing the generated models.

The web server is the middle point between the workstation and the raspberry pi, by serving the models generated for the Pi to download. To enable future learning from real imaging, the Pi will upload classified images to the web server for the workstation to use in future model generation.

B. Software Architecture

The following software packages and frameworks are used to generate the models for real-time image classifications and for classifying new images:

- Python 3.6
- Tensorflow-GPU
- Numpy
- SciPy
- OpenCV
- Pillow
- Matplotlib
- H5py
- Keras
- ImageAI [12]

ImageAI is an API that can generate models based on an image set and perform image classifications based on the generated models. The API can generate models using the Desenet, Inveption v3, Resnet, and Squeezenet algorithms.

The workstation utilizes the above listed software when generating the initial models used for classification. ImageAI is a wrapper framework for the rest of the libraries, simplifying the development process. The same ImageAI framework is used on the raspberry pi for real-time image classification utilizing an add-on camera board. The raspberry pi is capable of handling the classification algorithms because the model generation and model evolution is offloaded to the workstation [13]. This heavily reduces the compute requirements, enabling the mobile real-time classification.

The web server is a repository for the raspberry pi to retrieve the latest generated models and to upload classified images for further analysis. Real-time images are uploaded to the webserver for manual verification of the image's classification and are then loaded into the workstation as additional training images to evolve the models. The combination of the workstation and Raspberry Pi enables an overall system supporting model evolution while increasing the efficiency of the real-time classifier [13].

A future implementation adds a system capability of justifying the classifications provided. The current design behind the justification uses the built in confidence levels provided when classifying images. An alternative approach includes providing sample images that were classified using the same models and produced the same results.

C. Model Training

The generation of the classification models requires one data input and five configuration settings to generate the models. The script takes a folder path to a subset of the image dataset used for training as a script input. The remainder of the images is used for validating the generated models. The folder structure, as seen in Figure 7, consists of one folder per profession with each folder containing at least 200 images.

The next section of the script generates four different types of models based on four different generation algorithms. Each generation takes five configuration settings. The first one defines the number of image types, in this scenario being the ten different professions. The next input is the number of experiments, which determines the number of models to generate. The enhance_data input is an optional input, “This is used to state if we want the network to produce modified copies of the training images for better performance" [14]. The batch size input is dependent on the computing hardware available; the number is set to the maximum amount allowed by the equipment available. Finally, the show_network_summary input is used for providing detailed information to the console during model generation. The script cycles through the algorithms for generation and then generates 200 models and one JSON file per algorithm. The script serially generates the models for all four algorithms: DenseNet, Inceptionv3, ResNet, and SqueezeNet.

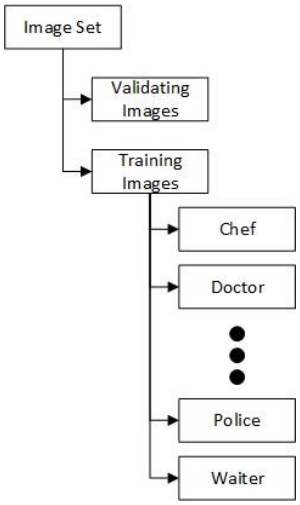

Figure 7: Image Folder Structure

Models are generated by breaking apart an image into simpler parts. These parts determine a rule set that goes into different layers. A culmination of each of these layers allows the model to make predictions based on an inputted image. The algorithm structures the layers in an optimal fashion to maximize the efficiency of image predictions.

During model generation the number of models to generate determines the number of variations the underlying algorithms with attempt. For instance, this research used 200 variations for the parameter settings producing 200 different models. The algorithms then determines an accuracy rate for each model generated, allowing the user to select the model with the highest evaluated accuracy. The generated models are given a specific naming structure for easy identification. The first parameter is an identification number ranging from one to the number of models generated. The second parameter gives the evaluated accuracy of the model, given in decimal format.

Once all the models are generated, a PowerShell script identifies the top three accurate models per algorithm. The script copies the chosen models to a separate folder for use in image classifications. When the models are generated, the calculated accuracy is added to the filename, which is how

the PowerShell script selects the top three. At this point, the models are either used for individual image classification or bulk image classification.

D. Image Classification

Classifying a single image is done through the individual image classification script. This script takes in two directory paths and a probability threshold as inputs. The directory paths contain the location of the image to classify and the models to use during classification. The probability threshold determines what predictions from the models are used in the voting. When the ImageAI algorithm attempts to classify an image with a model, the algorithm returns a single prediction with the probability of correctness. The probability threshold variable only allows predictions with greater than 80% confidence to be included in voting for the final predicted profession.

After all twelve models preform their prediction, the classifications that do not need the threshold requirements are thrown out. The remaining predictions are used in a simple majority voting scheme. The prediction with the majority of the votes from the various models is presented to the user with the average prediction confidence level and the number of models agreeing with the prediction. The script is capable of providing multiple predictions per image; however, for this application, only single predictions are needed.

E. Learning Algorithms

The ImageAI algorithm is used for the generation of the models and acts as a wrapper library to various machine learning algorithms, including Tensorflow. “ImageAI is an easy to use Computer Vision Python library that empowers developers to easily integrate state-of-the-art Artificial Intelligence features into their new and existing applications and systems" [15]. By implementing the system utilizing ImageAI, the overall development cycle was simplified by masking the low-level coding. For this system, custom recognition is utilized. However, the algorithm is capable of also performing objection recognition and live video detection [15].

F. Ensemble Methods

Two ensemble methods were used for combining the results of the individual models. The initial method used a simple majority voting scheme were each model has a single vote to the prediction. During analysis the simple majority method was altered to take into account each vote's confidence level. This weighted voting method calculated, which prediction has the highest confidence with a high number of votes. For example, if four models prediction waiter at a 95% confidence but five models predicted chef at 80% then the simple majority would produce chef as the answer, while the weighted voting would produce waiter.

IV. Data Analytics

During this research, the Identifiable Professionals (IdenProf) dataset [16] is used for evaluating the proposed changes to the ImageAI algorithm. IdenProf contains 10 distinguishable professions, listed in Table I. A sample image for each profession is shown in Figure 8. The dataset consists of over 900 images per profession used for training the system's models and an additional 200 images per profession for validating the models. All images are sized to a common pixel dimensions of 224 by 224 for uniformity. The image set has a makeup of mostly white males from the top 15 most populated countries [16], compared to other genders or nationalities. During the duration of this research project additional images can be gathered by pulling images from Google's search engine.

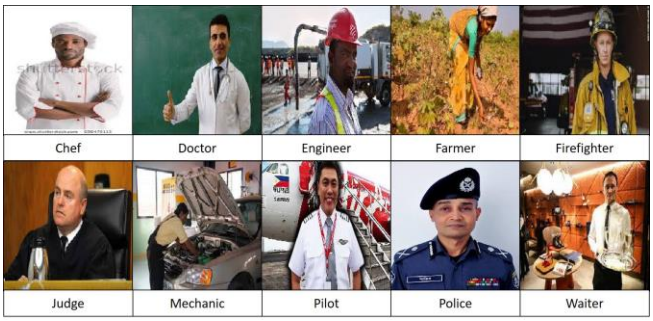

Figure 8: Sample Profession Images [16]

The experiments included testing the base algorithms against the training and validation images. These 200 images allowed analysis and validation of the models generated at all three stages of development. Additional experiments utilized the raspberry pi to simulate processing images on a low-powered machine. The models used in classifications are selected based on the assigned accuracy defined during model generation. For these experiments the models selected have over eighty percent accuracy.

Table I: Training Images Classification

Training Images Classifications	
Professions	Accuracy
Chef	74.5%
Doctor	76.5%
Engineer	86.0%
Farmer	89.5%
Firefighter	90.5%
Judge	92.0%
Mechanic	84.5%
Pilot	87.5%
Police	87.5%
Waiter	72.0%

Figure 10 depicts a collection of the test images for a pilot, one of the professions used in this research project. When running a classification against a pilot image, the system provides three results. Each result comes with a probability that the answer is correct. Typically the models generate one answer with a probability of over 95% and then the remaining two answers will make up the remaining percentage. During single image predictions, the system

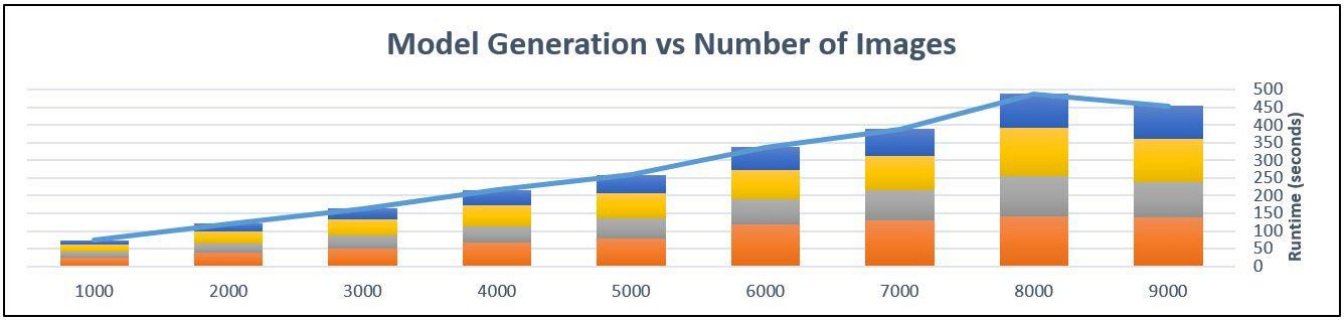

Figure 9: Model Generation Runtime

provided a profession prediction, with a confidence level, and the number of models agreeing with the answer. The confidence level is the average of the models in agreeance on the vote disclosing the models that do not meet the threshold. During this sample run three of the twelve models did not meet the required threshold and were dropped from the calculations. The remaining nine models resulted in pilot as the answer with a combined confidence of 99.99%. Similar results were found on additional tests with different images. Single image predictions was tested on both the workstation and the Raspberry Pi B+ to act as a low powered system.

Figure 10: Sample Profession Images - Pilot [16]

While testing the performance of the model generation algorithm, the runtimes for each model algorithm were compared at different image dataset sizes. Figure 9 shows the runtime of the four model algorithms and the total runtime at nine image set sizes. With the exception of the final image set of 9000, each runtime gradually increased compared to the next smallest image set.

V. RESULTS

The results were gathered through two different methods, the first conducting single image predictions and the other doing an automated bulk analysis.

A. Single Image Classification

During testing using the Raspberry Pi 3 B+ for image prediction, the Raspberry Pi required slightly different software. The Raspberry Pi required older versions of some libraries because the newer versions were not yet compatible. Figure 11 shows the results when testing the Raspberry Pi against a pilot image. The Raspberry Pi was able to correctly classify the test image with a 99.99% confidence level; based on four separate models, with three of the four models agreeing on the prediction. The same image was used for tests on the desktop and both the high performance desktop and the Raspberry Pi 3 B+ were capable of providing the correct prediction and same confidence level. The only difference was the use of four models instead of twelve, to minimize the run time required and counter issues of not enough memory for twelve models.

```
1 of 4 completed
densenet_model_ex-170_acc-0.871000.h5
2 of 4 completed
squeezenet_model_ex-001_acc-0.100000.h5
3 of 4 completed
resnet_model_ex-097_acc-0.852000.h5
4 of 4 completed


Filename:       pilotImage.jpg
Prediction:     pilot
Probability:    99.99999602635701
Voted by 3 of 4 models
(tensorflow35) root@raspberrypi:/home/pi/Downloads/imageai#
```

Figure 11: Single Image Prediction on a Raspberry Pi

During execution the Raspberry Pi was consistently over 90% memory utilization after running through five of the twelve models. Shortly into the sixth model assessment the python script crashed due to not enough memory available. Based on this limitation, the Raspberry Pi was configured to only use the top model from each algorithm instead of top three models per algorithm. All twelve models were ran individually and manually combined to verify only four models could perform as accurately as twelve. The manual combination resulted in nine of the twelve models agreeing with pilot for the prediction with a confidence level of 98.1%. The four model implementation produced the same correct profession prediction, but with an increased confidence level at 99.99%. Since the accuracy and performance increased the Raspberry Pi implementation was altered to the four model design.

The Raspberry Pi storage requirements grow at a rate of 12 KBs per image classified with a constant model storage rate of 205 MBs for four models. These requirements are portable to any edge node device used. The Raspberry Pi takes an average of five minutes to classify a new image and about 600 MB of memory for each image classification. The time to classify is dependent on the hardware used, were the Raspberry Pi takes five minutes per image the desktop takes only three minutes per image.

B. Bulk Image Classification

During the bulk image processing, the scripts ingested two thousand images, evenly distributed between ten different professions. All the images received a profession prediction from all twelve models. The Squeezenet models

consistently produced the wrong predictions, with only 10% of the predictions being correct. After further review, the model Squeezenet was only able to correctly predict a single profession. The Densenet and Inceptionv3 models all performed with an 87% accuracy and the Resnet algorithm performed slightly worse at 85%. Individually the models produced accurate results, but when implementing the voting schemes the results improved.

Table II depicts the results of two different automated ensemble methods and a third with manual intervention. Implementing simple majority voting, also known as bagging, with a confidence level threshold resulted in an 88% accuracy for predictions. The minimum required threshold eliminated the Squeezenet predictions from the voting. Where some models performed poorly for some professions other models performed strongly, resulting in increased correct predictions.

The second ensemble method involved expanding on the bagging algorithm and incorporating the confidence levels into the vote determination. Implementing this design improvement resulted in a one percent accuracy increase over the simple majority voting scheme, at 89% accuracy when averaging the predictions from all 2000 image predictions.

Table II: Training Images Classification

Ensemble Results			
Ensemble Method	Positively Predicted	Total Images	Percent Accuracy
Majority Rule	1764	2000	88%
Weighted Vote	1777	2000	89%
Weighted with Object Recognition	1807	2000	90%

Part of the future recommended work is incorporating object recognition to the predictions. The final row of Table II shows the improvement of doing manual object recognition for the waiter and chef professions. For manual recognition a tray or check book was searched for in the waiter images and a chef hat or side pocket thermometer for the chef images. The manual searching resulted in an improvement of one percent in accuracy over the weighted voting. Table III extracts the results for just the waiter and chef images. Adding object recognition gave an 8% accuracy improvement when looking solely at the chef and waiter images.

Table III: Training Images Classification

Object Recognition Result				
Prof.	Weighted Vote		Object Recognition	
	Positively Predicted	Percent Accuracy	Positively Predicted	Percent Accuracy
Chef	165 / 200	83%	174 / 200	87%
Waiter	148 / 200	74%	169 / 200	85%
Chef & Waiter	313 / 400	78%	343 / 400	86%

C. Additional Image Test

To verify the implemented algorithms additional police and firefighter images were chosen from Google searches and evaluated through the prediction script. These images were all classified correctly with over 98% certainty with at least six models agreeing on the vote. This test was conducted using the simple majority voting method. Each image was run multiple times against the same models to ensure the results were consistent each time.

An additional test of five new chef and five new waiter images were chosen from Google searches and evaluated through the prediction script. Eight of the ten new images classified correctly, at a rate of 80%. The majority of the correctly identified images had a certainty of over 90%. These images were also classified using the simple majority voting method.

VI. Hypothesis Results

This section discusses how the implemented design addresses the four hypothesizes discussed in the introductory section. The hypotheses are also listed below:

1. Utilizing a knowledge base, can a DSS be implemented to minimize the time and space requirements, while maximizing the accuracy of the suggested solutions?
2. How can a knowledge base be implemented to improve the overall accuracy of the system?
3. Is a DSS able to provide solution justification to the user, enabling users to have trust in the answers being provided?
4. Is it possible to improve rule evolution without needing to store locally all previous cases for doing rule re-validations?

A. Hypothesis 1

The implemented design does not decrease the time complexity compared to using a traditional single model process. However, by implementing a real-time image classifier on an endpoint node and the model generator on a remote server the space complexity does improve. The thought behind this is to remove heavy storage requirements from endpoints that typically have minimal resources, while the remote server typically has excesses resources.

B. Hypothesis 2

The implementation of a voting scheme with multiple algorithms used for model generation has allowed an improvement in prediction accuracy on average. Individually the models perform worse than when all models are used in a voting scheme.

C. Hypothesis 3

The voting scheme provides additional confidence in the provided result while increasing accuracy. The system provides the number of models agreeing with a prediction to assist in providing the user with greater confidence in the

result. The system also provides the probability the system believes in the response provided.

D. Hypothesis 4

The feedback loop introduces manually verified predicted images and adds those images to the pool for future model generation to enhance the model evolution. By improving the model evolution, predictions improve accuracy and allow for the system to handle input changes over time. For example, as a profession's physical characteristics evolve the system will evolve as well.

To validate the feedback loop for incremental evolution nine separate simulations was ran. Nine sets of models were generated using different training set sizes ranging from one thousand to nine thousand in one thousand increments. After the models were generated the same five hundred images were classified against the top twelve models from each model set. Table IV shows the simulations with the results. The results did not show a perfect upwards curve for accuracy, but did show that as more images were added to the training set the accuracy did improve. From this test the accuracy went from 78% at one thousand training images to 84% at nine thousand training images. The model set with one thousand images for training was an anomaly at a higher accuracy rate than the later model generations. This anomaly could be explained by the subset of images used in training just being an ideal set of images compared to the rest of image subsets.

Table IV: Model Evolution Results

Model Evolution Results				
Training Images	Positively Predicted	Incorrectly Predicted	Total Processed	Percent Accuracy
1000	391	109	500	78%
2000	316	184	500	63%
3000	349	151	500	70%
4000	360	140	500	72%
5000	381	119	500	76%
6000	399	101	500	80%
7000	400	100	500	80%
8000	424	76	500	85%
9000	421	79	500	84%

VII. CONCLUSION

Throughout this project, there were limitations to development based on the hardware available. For future development in this area, additional work should include different types of hardware platforms. The algorithms were implemented to handle both low and high-performance hardware. Implementing the system on hardware like the Nvidia 2080 Ti should improve model generation because of the additional memory and CUDA cores, allowing for improved accuracy for profession predictions. Beyond improving the specific hardware available, work with a distributed computing system should be researched.

A distributed system, shown in Figure 12, would grant a significant amount of computing power beyond what a single system would provide. A considerable improvement a distributed system would bring is a level of fault tolerance. A single system, similar to the one used in this project, has mostly all single points of failure. A distributed system would relieve the issue with single points of failure. During this project, using the Raspberry Pi attempted to simulate half the system in a distributed environment by using the Raspberry Pi as a low-powered endpoint node. The Raspberry Pi can be combined with an onboard camera to provide live image recognition as well.

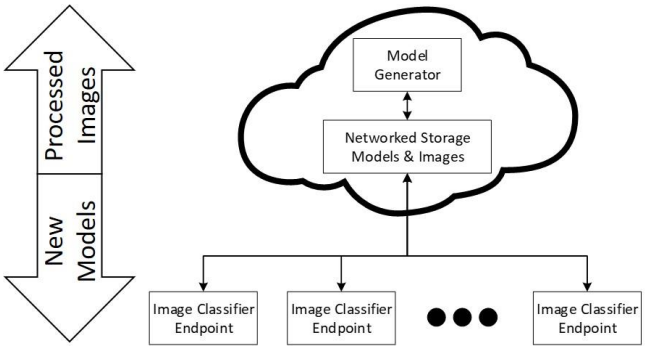

Figure 12: Distributed Network

This system's edge nodes are scalable based on the quality of the back haul bandwidth from the edge node to the centralized repository. Each individual edge node acts autonomously, with no knowledge of the other edge nodes. As long as the centralized repository has enough resources to handle obtaining all the images from the edge nodes, the generation and distribution of models then the overall system can easily scale.

Incorporating object recognition, visual shown in Figure 13, into this design would also improve the overall performance. As discussed in the results section, object recognition improved the predictions of the chef and waiter professions by 8%. This was done with only four objects manually identified, using additional objects would improve the accuracy even more.

Figure 13: Object Recognition Example [16]

Figure 14 depicts the image classification flow when incorporating object recognition. Each image is classified by twelve models using a whole image classification method. The twelve predictions are then combined through the weighted majority voting algorithm. Separately, the image is analyzed for object recognition, with the results compared against a repository of known objects linked to professions. The result from the object recognition and whole image classification is compared, if the results are the same then the system will export the result. However, if the comparison shows a disagreement then both results are evaluated for their confidence to determine what prediction to make.

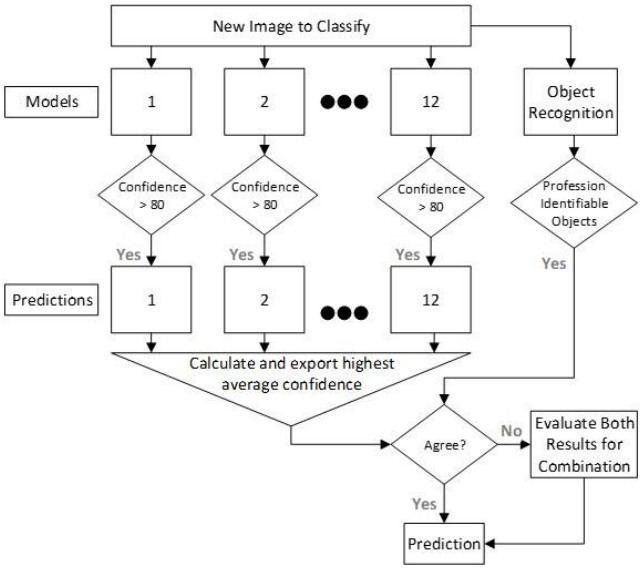

Figure 14: Object Recognition with Voting

Throughout this project machine learning algorithms and applications were reviewed to determine what improvements could be made to enhance the field. One of the greatest areas needing improvement was providing users with justification and confidence in the predictions a system is providing. This project attempted to use multiple machine learning algorithms in a voting scheme to increase the confidence level in the predictions, while also improving the accuracy of the predictions.

By using predictions as a feedback loop into the model generator, the system attempted to improve predictions over time. Improvement of the knowledge evolution is crucial for a system operating for any length of time. For instance, with the profession application, police officers over the last hundred years have evolved through their uniforms. If a system were generated using photos of police from the 1920's, then the system would have a difficult time providing correct predictions of present-day police.

With improved knowledge evolution, accuracy improvements become possible. The accuracy of the system was improved by implementing a voting scheme and a confidence threshold to drop low confident predictions. Specific algorithms showed higher performance against certain professions and were able to counteract lower-performing algorithms.

To improve the efficiency of the system, the space complexity was improved with a slightly worse time complexity. By implementing a server and client system, the size complexity was greatly reduced at the endpoint node while maintaining typical size complexity at the server-side. The client-side system only needed to download models from the server when available. Otherwise, the client can perform predictions uninterrupted. However, since the overall system utilizes the integration of open source elements there is minimal control over the inner workings of the libraries affecting size and complexity.

Overall, the implemented system addresses all hypotheses originally made by implementing common, open-source software in an untraditional manner. The delivered system from this project is capable of predicting a person's profession solely based on a single image of them, in a manner and speed humans would not be capable of achieving.

REFERENCES

[1] D. Prairie and P. Fortier, "Improve Operations of Real-Time Image Classification Utilizing Machine Learning and Knowledge Evolution", IARIA Data Analytics, 2019.

[2] A. Saxena, “Knowledge-Based Architecture for Integrated Condition Based Maintenance of Engineering Sysems," Georgia Institute of Technology, Tech. Rep., 2007, Accessed: Aug. 2019. [Online]. Available: https://smartech.gatech.edu/handle/1853/16125

[3] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, W. Murdock, E. Nyberg, J. Prager, N. Schlaefer, and C. Welty, "Building Watson: An Overview of the DeepQA Project," AI Magazine, 2010, Accessed: Aug. 2019. [Online]. Available: https://www.researchgate.net/publication/220605292 Building Watson An Overview of the DeepQA Project

[4] Google, "Tensor Flow" Google, [Online]. Available: https://www.tensorflow.org/. [Accessed Apr. 20, 2019]

[5] D. Aha, D. Kibler and M. Albert, "Instance-Based Learning Algorithms," Machine Learning, vol. 6, no. 1, pp. 37-66, 1991.

[6] F. Kittler and Josef; Roli, “Multiple Classifer Systems," in First International Workshop, MCS 2000.

[7] V. Smolyakov, “Ensemble Learning to Improve Machine Learning Results,"Statsbot, 2017, Accessed: Sep. 2019. [Online]. Available: https://blog.statsbot.co/ensemble-learning-d1dcd548e936

[8] F. P. R. E. Network, “Chances Increasing That Tropical Storm Dorian Will A ect Parts of Florida This Weekend," WUSF News, 2019, Accessed: Oct. 2019. [Online]. Available: https://wusfnews.wusf.usf.edu/post/chances-increasing-tropical-storm-dorian-will-a ect-parts-florida-weekend

[9] P. Domingos, “Bayesian Averaging of Classifiers and the Overfitting Problem," University of Washington, Seattle, Tech. Rep., 2002.

[10] J. D'Souza, “A Quick Guide to Boosting in ML," GreyAtom, 2018, Accessed: Sep. 2019. [Online]. Available: https://medium.com/greyatom/a-quick-guide-to-boosting-in-ml-acf7c1585cb5

[11] “Raspberry Pi 3 Model B+," Raspberry Pi Foundation, Tech. Rep., 2019, Accessed: Aug. 2019. [Online]. Available: https://www.raspberrypi.org/

[12] DeepQuest AI, "Official English Documentation for ImageAI!" DeepQuest AI. [Online]. Available: https://imageai.readthedocs.io/en/latest/. [Accessed: Feb. 11, 2019].

[13] S. Jain, "How to easily Detect Objects with Deep Learning on Raspberry Pi", medium.com, Mar. 20, 2018. [Online]. Available: https://medium.com/nanonets/how-to-easily-detect-objects-with-deep-learning-on-raspberrypi-225f29635c74. [Accessed May. 11, 2019].

[14] M. Olafenwa, "Custom Training," ImageAI, Tech. Rep., 2019, Accessed: Sep. 2019. [Online]. Available: https://github.com/OlafenwaMoses/ImageAI/blob/master/imageai

[15] M. Olafenwa and J. Olafenwa, "ImageAI," ImageAI, Tech. Rep., 2019, Accessed: Sep. 2019. [Online]. Available: http://imageai.org/

[16] M. Olafenwa, "IdenProf Datasheet" Olafenwa, [Online]. Available: https://github.com/OlafenwaMoses. [Accessed Mar. 16, 2019]

[17] D. Galeon, "Paging Dr. Watson," 28 October 2016. [Online]. Available: https://futurism.com/ibms-watson-ai-recommends-same-treatment-as-doctors-in-99-of-cancer-cases/. [Accessed 22 February 2019].

[18] Dtex Systems, "The Hidden Security Threat," Dtex Systems, 2016. [Online]. Available: https://dtexsystems.com/portfolio-items/infographic-findings-from-the-2016-costs-of-insider-threats-report/. [Accessed 21 March 2019].

[19] Q. Althebyan and B. Panda, "A Knwoledge-Base Model for Insider Threat Prediction," Proceedings of the 2007 IEEE Workshop on Information Assurance, vol. June, pp. 20-22, 2007.

[20] P. Bakkum and K. Skadron, "Accelerating SQL Database Operations on a CPU with CUDA," University of Virginia, Charlottesville, 2010.

[21] J. Jean, G. Dong, H. Zhang, X. Guo, and B. Zhang, "Query Processing with An FPGA Coprocessor Board," in Proceedings of the International Conference on Engineering and Reconfigurable Systems and Algorithms, 2001.

[22] Martín Abadi et al, "TensorFlow: A System for Large-Scale Machine Learning," in 12th USENIX Symposium on Operating Systems Design, Savannah, 2016.

[23] D. Patil and P. Jayantrao. "Malicious URLs Detection Using Decision Tree Classifiers and Majority Voting Technique." Cybernetics and Information Technologies. vol. 18. no. 1, pp. 11-29. 10.2478/cait-2018-0002.

[24] C. Nichols, "How many flights come in and out of LAX every day?," Los Angeles Magazine, 1 May 2011. [Online]. Available: http://www.lamag.com/askchris/how-many-flights-come-in-and-out-of-lax-every1/. [Accessed Mar. 20, 2018].

[25] Homeland Security, "Combating the Insider Threat," Homeland Security, 2014.