Utilization of TensorFlow for Number Recognition

Michael Hayes, Amrutvyasa Ramasay, and Cody Bijeaux

Kennesaw State University

Department of Computer Science

CS 6045: Advance Algorithms

Dr. Michail Alexiou

May 4th, 2024

**Abstract**

*The paper below focuses on using a TensorFlow Model through a google Collaboration to train an object detection model to recognize numbers of theoretical RFID tags. The process is based off similar work using a similar system to track the water intake of cattle. Three Models were created from a dataset that included base images, rotated images, and tinted images. The Models differed either through how much of the dataset was used or the number of steps taken during the training process. The result was shown that a more substantial dataset being more impactful to average scores than more steps. The best of the three models was used to create a program to output the four-number tags on images fed through it. Once fed through, a separate merge and stitch together program was utilized to output the final result of four-digit numbers.*

**Introduction**

Through research and the recent high interest in Artificial Intelligence, increased research is desired under the subject with the idea of producing a tangible result that could be implement in a wide array of situations. Basing off the existing research, the goal of this project is duplicate and improve on an object detection program that is trained to recognize numerical id’s of a theoretical RFID tag through images.

**Related Work**

This project's work was based on similar methods. The original purpose of source paper was to track the water intake of livestock, more specifically tracking individual livestock intake of water. Of course, such a task would be very time consuming or coding intensive without Machine Learning. As such, the original work proposed using Artificial Intelligence object detection systems to track it instead (Biglari & Tang, 2022). It is important to note that the specifics of their process will be summarized and explained in greater detail when discussing the target project described in most of this paper.

Their methodology was to create a dataset based on the numbers represented on RFID tags, which would be placed on livestock later. They would train with the dataset using TensorFlow, convert it to TensorLite, and then utilize the trained model on a raspberry pi connected to a high-performance camera to investigate livestock water usage. They had an original data set of one-hundred and fifty images with a size of 640x480. This was duplicated and altered using Pytorch. This resulted in a total data set of four hundred and fourteen images. These images were trained using Tensorflow and guided using object detection collaboration. The final model was converted to a TensorLite model for application with the hardware (Biglari & Tang, 2022).

The result of this application was an accuracy rating of ninety to one hundred percent. However, there were some key issues with the current makeup of that product. The most substantial issue was the fact that the model was trained with images containing four numbers. These were the numbers to be directly used on the livestock. This means that the AI was only proficient at understanding those specific numbers. If new livestock were to be introduced, the process would need to be repeated (Biglari & Tang, 2022). The goal of this paper’s project is to create a version of this that can be used with any series of numbers.

**Tools Used**

There were two tools used for this program: google collaboration, and TensorFlow/TensorLite.

Google Collaboration

The source paper discussed above used a Github of an object detection system that was already pre-set for use. While the original example presented by source Github was creating program to recognize American coin-based currency when presented to the camera, the foundation could be used for any form of object detection. The process used by the Github repository was relatively simple and straightforward. The code was broken down into steps, which each step clearly explained. The process was essentially setting up the collab, taking the images in a specific format for training, training the model, converting the model to a tflite version, then showing how to create a python environment to utilize the tflite file. (Juras, n.d.)

TensorFlow and TensorLite

TensorFlow was the backbone of model used both this project and the one discussed in the source paper. Tensorflow is a google product that hopes to streamline machine learning and hopes its users utilize it for their own AI models. (“why Tensorflow”, n.d.) How it does perform this is centered around the key idea of machine learning. The model is trained with pre-determined images and given a test within a unique set of images. The model is highlighted with its correct choices, to which the model is trained again with the framework that made the correct choices before. This framework is redone multiple times until it reaches the amount of ‘steps’ desired by the user. This results in a graph inside the model used to determine how to detect desired objects from within the target image. (Brown,2021)

TensorLite, which is what the Tensorflow model is converted to for use on lighter platforms with lower latency, is mostly used for android, iOS, and raspberry pi. However, TensorLite is not itself a direct creation of machine learning, but instead a resulted conversion from an existing model created from TensorFlow. The TensorLite model, as stated by TensorFlow, “is built from the frozen graph using the TensorFlow optimizing converter tool or TOCO” (Moroney,2018,02:34). This means that it takes the current graph values of the TensorFlow model and converts it into constants to be used by the TensorLite model, this process is referred to as ‘freezing’. (Moroney,2018) Due to most application of object recognition is on smaller systems, often with just a mini-computer and a camera, it was decided to continue testing with the TensorLite model.

**Methodology**

To ensure that this is not a carbon copy of the project done previously, the methodology of this assignment was intensively examined. The result is split into three sections: The dataset, The Training, and The Stitching.

The Dataset

With any Machine Learning model, how the data set is formed is incredibly important. It was decided that the data set would be a base set of human-generated images, then that would be extended to a larger set of images copied and altered from the base original images. The base images were of numbers from zero to nine, represented in ten images. Three separate and independent image sets were created the same way. The size and location of the numbers varied, but the font and size of the images stayed consisted- Arial font and 320x320.

A grey background with black numbers

Description automatically generatedA number on a white background

Description automatically generated

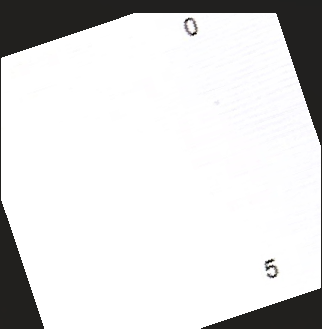
**examples of the original dataset**

The base dataset was then duplicated and altered using Pytorch, which was the same process used in the original paper. The following transformations were used:

Random Rotation: rotates the image a random amount.

Random Perspective: changes the perspective of the image

ColorJitter: changes the color of the images, usually in the same manner as adjusting the brightness on an image.

**A white card with black numbers

Description automatically generated**A close-up of a number

Description automatically generated

**ColorJitter Change** **Random Perspective** **Random Rotation**

The total aggregate dataset resulted to be around three hundred and ten images.

The Training

The training step of the process is another deviation from the source paper. It was decided to not train just one model, but three. These three would differ in one of two aspects: the size of the dataset or the number of steps used in the training. The idea was to test, if there was some kind of restraint that forced a choice, would be the best to focus on for training a model. The first model would be training with twenty thousand steps with the full dataset. The second one would be using twenty thousand steps, but only with half of the dataset available. The final one would utilize the full dataset but would only use ten thousand steps.

Model One: 20k steps with full dataset

Model Two: 10k steps with full dataset

Model three: 20k steps with half dataset

It is important to the that the base paper decided to use forty thousand steps for their model. However, it was decided to decrease the steps to save on processing power on google collaboration. In addition, the study charts showed very little improvement between the twenty-thousandth and forty-thousandth step (Biglari & Tang, 2022).

Each of these models would be trained independently and the results compared. After which, the best model, which was assumed to be Model One, would be the final model used for the third and final step of this methodology.

The Stitching

Since it was decided to train with individual numbers, that means some work needed to be done to convert the results into four-digit numbers for analysis. After some examination, it was determined that the process would include three steps: pulling the data, converting to list, and sorting the list, then finally outputting the results.

The data would first be pulled from the model after it determined the numbers on the image. The detection results would be converted to a list of detections of the format “[Class,score,xmin,ymin,xmax,ymax]”. After which the list would be sorted using a merge sort algorithm. Which the sorted list, a group detection program would then be used to combine the lists into four-digit numbers. This was done by comparing the minimum x and y values of the numbers to determine where they were clustered. In instances where the group of numbers were above each other, the y values would also be considered to ensure that each number was placed in their correct grouping. After the grouping was successful, the program would output the results on the image for viewing.

With the parameters of the methodology set, the training was conducted, and results examined.

**Evaluation and Results**

Once again, what could determine as evaluation and results can be split into two sections: Comparing Models and Final Model Output.

Comparing Model**s**

This was seen as a crucial part of this project, as it was meant to determine what to prioritize when training a model. To do this, the mAP (Mean Average Precision) scores, which shows the general accuracy or the program, were used and compared:

A screenshot of a graph

Description automatically generated

As can be seen above, Model One was the best model for use based on average mAP score. Of course, this was not really a point of contention. The real results worth investigating were comparing Model Two and Model Three. Even Though Model Two only had only gone through half the steps, it had a much higher accuracy score than Model Three. This emphasized the importance of having a robust data set over the number of steps used. The model with half the dataset had some particularly interesting detection issues.

A grey and black background with green squares and numbers

Description automatically generatedA screenshot of a computer

Description automatically generated

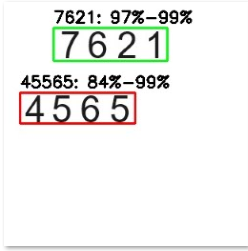
**Third Model’s individual results**

In addition, numbers that were ambiguous when rotated made it difficult for the Models. This is why average recognition for nine was low across the board. It would confuse the number for a six if rotated a certain way. Regardless, it was ultimately decided to use Model One for the Final Model Output.

Final Model Output

With the stitching program created and the model trained, the images below were created with the program.

A close-up of numbers

Description automatically generated

**Output of the Final Model with the added Stitching program**

As can be seen above, the program was successful in reading and grouping the numbers to some extent. A common issue was the program would duplicate an extra number. Since the program was trained with individual numbers, it never knew what length of numbers to check for. As a result, it would sometimes come up with an additional number. It would very rarely mistake another for different number. Each of the numbers on the image would be present in the output, and often it would be correct completely. However, the additional number added does show a weakness in training the numbers individually.

**Conclusion**

This paper's purpose was to find an efficient way to use object detection to recognize four-digit number sequences. An additional purpose was added to examine how important steps and a robust dataset are to the results. Once the results were found, it was determined that a more robust dataset is more impactful to the final average accuracy score. This could push future implementation of similar architecture to dedicate more time on building a more comprehensive dataset and have a deeper discussion on whether individual numbers or grouped numbers should be tested.

**References**

Biglari, A., & Tang, W. (2022). A Vision-based cattle recognition system using tensorflow for livestock water intake monitoring. *IEEE Sensors Letters*, *6*(11), 1–4. https://doi.org/10.1109/lsens.2022.3215699

Brown, S. (2021, April 21). *Machine Learning, explained*. MIT Sloan. https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained

Juras, E. (n.d.). *TensorFlow Lite Object Detection API in Colab*. Google colab. https://colab.research.google.com/github/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi/blob/master/Train\_TFLite2\_Object\_Detction\_Model.ipynb

Moroney, L. (2018, April 5). *Introducing Tensorflow Lite (coding tensorflow)*. YouTube. https://www.youtube.com/watch?v=5J72iMSQmy8

*Why tensorflow*. TensorFlow. (n.d.). https://www.tensorflow.org/about