# Falling Object Counting of Videos-Advanced Robotics Class Project Stage2 Report

Zihe Ye

***Abstract*— Video object counting problem usually is composed of two parts: Frame objects counting and objects tracking through frames. In this second stage, we build up a system based on pre-trained ResNet50 for predicting number of cubes in one frame using RGB image, Depth image, and RGB-D image. Image classification is a very common problem in Deep Learning and Computer Vision community, where CNN is a very commonly used method to extract and classify images based on image features. In this report, we talk about the problem we try to tackle and the methods we use, as well as the training and evaluation process. We achieve a 0.0335 MSE loss in the regression evaluation on test set for RGB modality, which is improved by recollecting more precise ground truth. In addition, as our problem is very complex, so there are many possible ways to get to a better performance, therefore the limitations and next stage works will be talked about in the conclusion part. The paper we are following and citing are in the related work section.**

## I. PROBLEM STATEMENT

In the first stage of the project, we try to tackle the cubes counting problem of a single frame based on rgb information. We formulate the problem as a regression problem as there is no certain number of classes and the distribution among possible numbers can be very unequal. Therefore, the supervised learning model is supposed to predict the number of cubes in one image, this original prediction can be fraction which will be round to an integer when applied to real-time prediction application.

## II. DATA PREPARATION

The data we use are generated from CoppeliaSim simulator with a given scene. Generated data will be divided into two parts, where the first part is the video itself of .h5 format with 60 Hz frequency, and the second part is the pouring information including seven elements for every frame: angular position, weight in receiver(how many have been poured out), total weight, target weight, H, D, pour speed, from which we create a new numpy file ground truth which is calculated from deducting weight in receiver by total weight. There are several different views and different types of cameras, we only use the rgb image which is rotating with and always facing the pouring cup to do the training. Inputs is the rgb image of a size 186*116 in height and width, outputs is a fraction number which predicts the number of cubes in the image (cubes have not been poured into the target container yet).

We collect in total of 50 videos which contain 33513 frames, which results in an average of 670 frames in every video which is about 11 seconds in real world. We further randomly re-arrange all frames and their corresponding labels, and divide the entire dataset as a 60 to 20 to 20 ratio to be training set, validation set, testing set.

### A. Simulation Setup

Figure 1 shows the setup in the simulation scene, where the cup rotates respect to the joint inserted inside the cup, and there is a RGB-D camera mounted on top of the cup to capture the video when the cup is pouring.The Depth and RGB information of the camera are recorded from the simulator with a 60Hz frequency and saved as a .h5 video, and we use h5py reader to convert the videos into a series of numpy array which contains all images. Similarly, we do the same thing to create the ground truth numpy file.

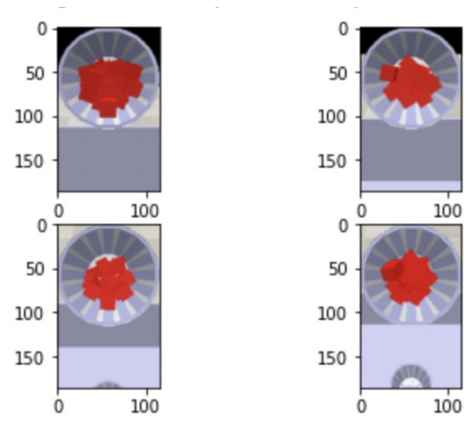


Fig. 1. Four frames in one video

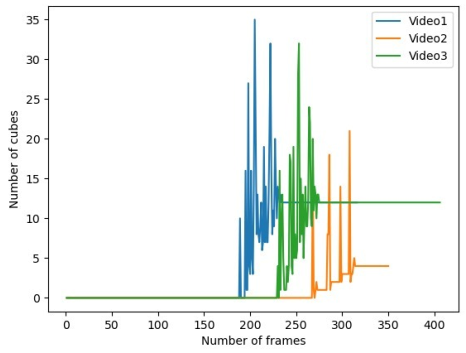


Fig. 2. Three videos ground Truth sequence series

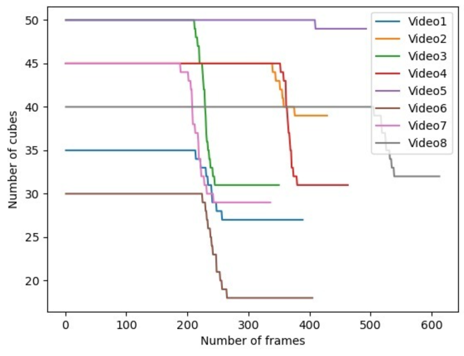


Fig. 3. New data sequence

As for data preprocessing, we get images as inputs similar to images in Figure 1, and ground truths are points from series like the sequence in Figure 2 and Figure 3. However, it is obvious there are some spikes in each sequence which can be treated as noises which is discussed further in conclusion part. Besides getting images from the h5 video file, we also try do normalization on images as a preprocessing methods. Other methods such as cropping, resizing are not included in preprocessing stage.

## III. Related Works and Paper Implementation

CNN based methods object detection have been a hot focus by computer vision for a long time, [4] works on leaf counting using bounding box by YOLO and achieves a good result in counting leafs for each individual plant. While authors of [1] use similar strategy to do fruit counting. A common feature of these tasks is that the surrounding noises are very big and may affect the performance of the counting system, which in turn in our task is the occlusion of objects. Our work has the same goal as these past works, which is to count how many cubes have been poured out of the cup in real-time. However, while we also use pre-trained models such as VGG16 [3] and ResNet50 [2] for the regression problem, we currently do not apply bounding box and tracking system into our entire pipeline which is because we want to see how our system performs while there is no YOLO-like bounding box condition and no tracking system included in the entire pipeline. Current result shows very promising result on cubes number prediction for single frame and decent result on entire video cubes number prediction. In the future stage, we may try adding bounding box if the current strategy cannot achieve satisfying result.

As we do not have bounding boxes and tracking system for the video counting, we use an alternative which is to count how many cubes are in the current frame, just so we can perform a deduction operation to get the difference from later from to previous frame as the number of poured out cubes. This approach is inspired by the approaches in [6] and [5]. In [6], the authors use multi-column CNN network on a single image to predict number of people in a crowd image. In [5], the authors use multi-views via ground plane density maps. The idea of both papers match our idea and method which is to predict a single image objects number, and we can extend our approach to video objects counting as a difference.

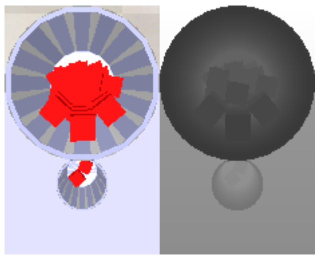


Fig. 4. Concat RGB and Depth images

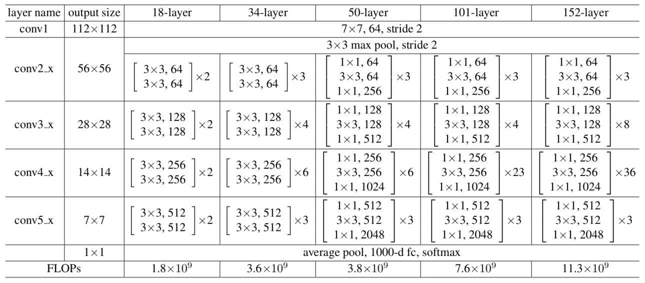
## IV. Methodology

A first stage fine-tuning and using noisy ground truth training is done in stage 2-1. After stage 2-1, we decide to work on dataset and models separately to get better result. In this stage, we redo the data collection part, mainly focus on getting a more precise ground truth for every frame. As shown in Figure3, there is no spikes in every sequence we collect. The way we do this is to get ground truth z-value for every object and compare to a certain fixed horizontal line. There are 80 videos in total of 57449 frames. After data-recollection, we redo the same training procedure as last stage and get a decent result with only RGB images. Then we also try using only depth images, and concatenation of rgb images and depth images. It in turn shows that the concatenation shows a promising potential improvement which verifies our assumption, while depth itself performs decently. For image classification and regression problem, CNN is the most commonly used approach. In this stage, we only use ResNet50 [2] architecture which is because of the experience from last stage. The pretrained ResNet model is originally for 1000 classes ImageNet classification, so we remove the final layer and add a new layer so the output will be one regressed value. The optimizer used for all training combinations is Adam, and all code is done and ran in TensorFlow 2.7.0 on a NVIDIA 3090 GPU. The loss function used is MSE loss.

In stage 2-3, we try different things such as fine-tuning on hyper-parameters, splitting the dataset into three sets, predicting on the difference between two frames. The current model is the best model we get after different trials on three different modalities respectively. The purpose of splitting the dataset into three sets is to reduce the memory load for each time and improve the running time performance, however as we need to load train and valid data in the training stage anyways and that occupies around 80%, therefore we still cannot fully get rid of memory issue to load all 80 videos subset for training stage. The predictions of difference between frames is shown in the training process and result sub-section.

## A. *Training Setup*

Although the entire dataset of 80 videos contains more than 50000 frames, we cannot use all of them for training and validation because of memory problem. Therefore, the result shown in I for depth and rgb-d are only trained on 25000*0.6=15000 frames. In Figure 5, the ResNet50 architecture is shown. As pre-trained model architecture ResNet can accept any size of three channels images, we do not apply any kind of resizing to fit the original 224 by 224 input for these pre-trained models. Therefore, for RGB and Depth only training and testing our model input is 186*116*3 image, and for the concatenation we have 186*232*3 image.



| layer name | output size | 18-layer | 34-layer | 50-layer | 101-layer | 152-layer |
|---|---|---|---|---|---|---|
| conv1 | $112\times112$ | $7\times7$, 64, stride 2 | | | | |
| conv2_x | $56\times56$ | $3\times3$ max pool, stride 2 | | | | |
| | | $\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$ | $\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$ | $\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$ | $\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$ | $\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$ |
| conv3_x | $28\times28$ | $\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$ | $\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$ | $\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$ | $\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$ | $\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$ |
| conv4_x | $14\times14$ | $\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$ | $\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$ | $\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$ | $\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$ | $\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$ |
| conv5_x | $7\times7$ | $\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$ | $\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$ | $\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$ | $\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$ | $\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$ |
| | $1\times1$ | average pool, 1000-d fc, softmax | | | | |
| FLOPs | | $1.8\times10^9$ | $3.6\times10^9$ | $3.8\times10^9$ | $7.6\times10^9$ | $11.3\times10^9$ |

Fig. 5. ResNet50 architecture

## B. *Training process and result*

In this stage, we only apply one training setting: with a fixed learning rate 0.0001, adam optimizer with MSE loss. Fine tuning will be performed and shown in future stage. We try the same architecture on three different modalities image and see a difference in the training and testing result, possible explanation is shown in the evaluation section.

TABLE I
COMPARISON OF DIFFERENT ARCHITECTURES

| **Modality** | **Train loss** | **Valid loss** | **Test loss** | **Frames difference** |
|---|---|---|---|---|
| Depth | 0.0404 | 0.0456 | 0.0457 | 0.0481 |
| RGB | 0.0327 | 0.0421 | 0.0335 | 0.0362 |
| RGB-D | 0.0419 | 0.0651 | 0.0568 | 0.0547 |

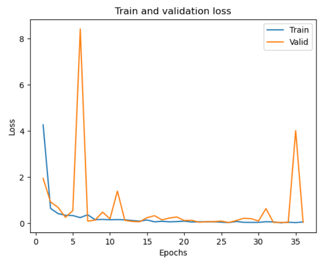


Fig. 6. RGB training curve

Figure6 shows the training and validation curve of RGB only input training procedure, this procedure is done using the entire dataset. Figure7 shows the training and validation curve of Depth only input training procedure, this procedure is done using 25000 frames for train, valid and test set. From TableI it shows that all three modalities perform decently. RGB only performs the best among all three, however this is under the condition that RGB only ablation is done under the bigger dataset, therefore we will do the study under same dataset in next stage to keep fairness among three methods. Moreover, depth solely seems to perform better than RGB-D, this is not as ideal as the assumption, one possible explanation can be that two independent ResNet for extracting features and then concatenate features channel-wise can help the result improve in next stage. In next stage, we plan to do the following things to get improvement and further information about our study: 1. We plan to do fine tuning on architectures and hyper parameters for all three modalities. 2. We plan to try channel wise feature concatenation using two ResNet50 models for RGB and Depth. 3. We plan to try other colors or shapes and see if our model can generalize to other color or shape. 4. We plan to take in a sequence and predict number of cubes poured out in that sequence.

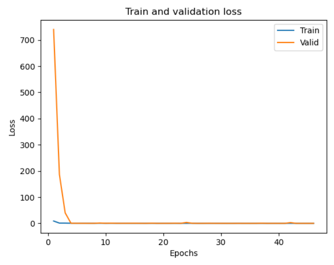


Fig. 7. Depth training curve

## V. EVALUATION

As shown in TableI, our model achieve a much lower MSE loss for train, valid and test set with all three different modalities combination compared to stage one, this is mainly because our modified ground truth retrieving procedure. With Depth only input, we achieve a train loss of 0.0404, a valid loss of 0.0456, a test loss of 0.0457. With RGB only input, we achieve a train loss of 0.0327, a valid loss of 0.0421, a test loss of 0.0355. With RGB-D input, we achieve a train loss of 0.0419, a valid loss of 0.0651, and a test loss of 0.0568. In next stage, we may perform the training on a larget dataset if the memory issue is resolved. More linear layers or global pooling may also be involved to try for the best combination. And we will evaluate our model performance on both frames number prediction or a sequence of frames to calculate the video poured out cubes number.

In the TableI result it shows that the difference between frames prediction is very close to the prediction of two frames ground truth, and single frame prediction is also very accurate. These points are the advantages and good performance by our methods. However, there is one problem that we have not resolved yet and need to work more on in the future stages that the model doesn't generalize very well to unseen videos with different number of initial cubes and targets. It is reasonable and intuitive as this problem with much occlusion is very hard to get a very precise model that can also generalize to other scenarios very easy with good performance, more details are discussed in conclusion and Discussion section.

## VI. Conclusion and Discussion

To conclude, we re-design the procedure to get more precise ground truth and re-train the model based on RGB, Depth, and RGB-D modality, which in turn gives us a decent result on cubes number prediction. In next stage, we plan to collect more data and evaluate our model performance on both images and videos, and we also plan to do more studies on different colors or different shapes to see if our model has the ability to generalize to other scenarios.

As mentioned in the related works section, very well model can generalize to other scenes can be trained on crowd with density map and multi-views. We have collected multi-view images too but we did not use them since the cup is not transparent at the moment and having a transparent cup in real world setting is not very realistic. However, we do believe that making our architecture more complex to be precise and robust is necessary and promising in future works. Besides, since our frames are not independent but extracted from each video, and there are two stopping periods at the beginning and the end of one video which might cause data set distribution issue, which can be better tuned by more experiment. Overall, we train deep learning models with different modalities using rgb and depth information to do cube number prediction using regression and achieves a good performance among seen videos images, and we propose a methods to avoid potential tracking to predict poured out cubes number. We can have improvements in terms of generalization and data distribution in the future works.

## References

[1] Enrico Bellocchio et al. "Weakly Supervised Fruit Counting for Yield Estimation Using Spatial Consistency". In: *IEEE Robotics and Automation Letters* 4.3 (2019), pp. 2348–2355. DOI: 10.1109/LRA.2019.2903260.

[2] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[3] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. DOI: 10.48550/ARXIV.1409.1556.

[4] Jan Weyler et al. "Joint Plant Instance Detection and Leaf Count Estimation for In-Field Plant Phenotyping". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3599–3606. DOI: 10.1109/LRA.2021.3060712.

[5] Qi Zhang and Antoni B. Chan. "Wide-Area Crowd Counting via Ground-Plane Density Maps and Multi-View Fusion CNNs". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 8289–8298. DOI: 10.1109/CVPR.2019.00849.

[6] Yingying Zhang et al. "Single-Image Crowd Counting via Multi-Column Convolutional Neural Network". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 589–597. DOI: 10.1109/CVPR.2016.70.