Database and Benchmark for Early-stage Malicious Activity Detection in 3D Printing

Xiaolong Ma[1], Zhe Li[2], Hongjia Li[1], Qiyuan An[2], Qinru Qiu[2], Wenyao Xu[3], Yanzhi Wang[1]
[1]Northeastern University, [2]Syracuse University, [3]The State University of New York at Buffalo
E-mail: [1]{ma.xiaol, li.hongjia,}@husky.neu.edu, [1]yanz.wang@northeastern.edu,
[2]{zli89, qan100, qiqiu}@syr.edu, [3]wenyaoxu@buffalo.edu

Abstract— Increasing malicious users have sought practices to leverage 3D printing technology to produce unlawful tools in criminal activities. It is of vital importance to enable 3D printers to identify the objects to be printed and terminate at early stage if illegal objects are identified. Deep learning yields significant rises in performance in the object recognition tasks. However, the lack of large-scale databases in 3D printing domain stalls the advancement of automatic illegal weapon recognition. This paper presents a new 3D printing image database, namely C3PO, which compromises two subsets for the different system working scenarios. We extract images from the numerical control programming code files of 22 3D models, and then categorize the images into 10 distinct labels. These two sets are designed for identifying: (i). printing knowledge source (G-code) at beginning of manufacturing, (ii). printing procedure during manufacturing. Importantly, we demonstrate that the weapons can be recognized in either scenario using deep learning based approaches using our proposed database. The quantitative results are promising, and the future exploration of the database and the crime prevention in 3D printing are demanding tasks.

1 Introduction

3D printing, also known as additive manufacturing, has been widely observed that it is superior on the traditional manufacturing techniques on customization, limited material requirement, low equipment costs and wide accessibility [1], [2]. However, these characteristics are easily leveraged by malicious users to manufacture unethical and even illegal products (especially weapons as shown in Fig. 1).

Due to the wide accessibility, **illegal weapon production** is among one of the most immediate concerns to be addressed in 3D printing domain. A 3D printed gun could be produced with neither a serial number nor background check [3,4] which makes it hard to trace. The development of 3D printed bullet has also raised the concern for security implications of 3D printing [5].

It is an urgent challenge to prevent the illegal weapon production in real time manner. The intuitive idea is to embed a recognition system to the 3D printer and make

Figure 1: Examples of illegal products (weapons) manufactured by 3D printing.

it aware of the identity of a printed object so that production can be administrated and the illegal weapon will not be manufactured. Fig. 2 is an overview of our solution to remedy the illegal weapon production in 3D printing. Deep learning techniques such as Convolutional Neural Networks (CNN) have been widely used for computer vision tasks. Nevertheless, few researches have focused on combining deep learning techniques with 3D printing to address aforementioned concern - illegal weapon production. The lack of large-scale databases stalls the advancement of automatic illegal weapon recognition in 3D printing.

To tackle the illegal weapon production issue in 3D printing, we propose a new image database for 3D printing objects, namely "C3PO" (representing cognitive 3D printing objects). The first dataset in C3PO comprises $62,200$ RGB images for 10 classes collected from 22 well-defined 3D models. The 3D models includes real guns, gun-like objects, common objects. The images are constructed from projections of randomly posed 3D objects on the $xy-$, $xz-$, and $yz-$planes. In particular, we demonstrate that these commonly printed objects can be recognized via a supervised image classification formulation. The above scenario is based on the full prior knowledge of the printed objects, i.e., the recognition system can pre-compute the projections of an object from the control file for printing and

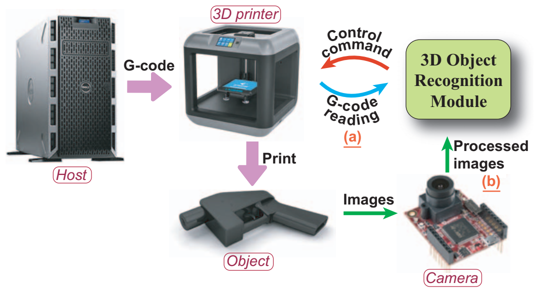


Figure 2: Overview of our proposed early-stage malicious activity detection system for 3D printing. (a) module reads G-code; (b) visually monitors the printing process.

978-1-7281-4123-7/20/$31.00 ©2020 IEEE

then classify the object by its projections.

On the other hand, what if the full prior knowledge is not accessible by recognition system? In this case, the system needs to observe what is printing by using cameras. To address the recognition problem in this scenario, we propose another dataset included in "C3PO" to simulate a random object's projection on the $xy-$, $xz-$, and $yz-$planes with respect to steps (a step is a printing completeness of a layer). The dataset includes sequences of a total of $671,677$ images. Each sequence is extracted for a randomly posed object defined in one of 22 3D models. The objects are categorized to 10 labels. We demonstrate that the image sequence recognition can be formulated as a unified model combining convolutional neural network with recurrent neural network (RNN).

This paper is the first to present a large-scale image database in 3D printing domain. The proposed approaches shows promising quantitative results using the proposed database. We share our datasets in an anonymous link `http://bit.ly/2YOEa5Z`.

2 Related work

With the wide utilization and spread of 3D recognition, several important datasets have been established to help improve the recognition. Geiger *et al.* released the KITTI dataset for autonomous driving, which comprises 389 stereo and optical flow image pairs, stereo visual odometry sequences of 39.2 km length, and more than 200k 3D object annotations captured in cluttered scenarios [6]. PASCAL3D+ dataset is proposed by Xiang *et al.* in 2014, providing 2D-3D alignments to 12 rigid categories containing $30,899$ images [7]. Based on the 3D information provided, supervised learning techniques are introduced and developed to recognize the 3D objects. In 2015, Maturana *et al.* proposed VoxNet, an architecture which integrates a volumetric Occupancy Grid representation with a supervised 3D CNN [8]. Johns *et al.* applied CNN to generic multi-view recognition, by decomposing an image sequence into a set of image pairs, classifying each pair independently, and then learning an object classifier by weighting the contribution of each pair [9]. However, the related researches on 3D printing object identification are almost blank due to the lack of large-scale database in the 3D printing domain.

3 Construction of malicious activity recognition database

In this section, we describe the approach for building a large-scale malicious activity recognition database for 3D printing, namely "C3PO", extracted from our pre-defined 3D models. First, we introduce the basics about 3D printing details. Then, we define two working scenarios in the 3D printing malicious activity recognition system. We consider the 3D object recognition module in Fig. 2 in the system being able to either access the numerical control programming code from 3D printer or visually monitor the printed objects by cameras. For two scenarios, we collect two sets of projections of the printed objects on a/some certain plane(s). The projections are converted to images to form the database "C3PO".

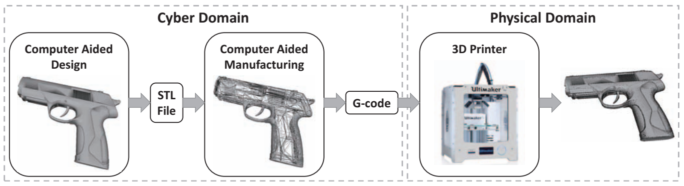


Figure 3: The overview of the 3D printing procedure (a pistol as example). Computer aided design (CAD) software creates the standard object file (STL) which describes the surface geometry of the 3D object. The computer aided manufacturing software slices the model into uniform layers and generates G-code. The 3D printer interprets to follow the instructions in the G-code to make spatial movement and extrude the melt material.

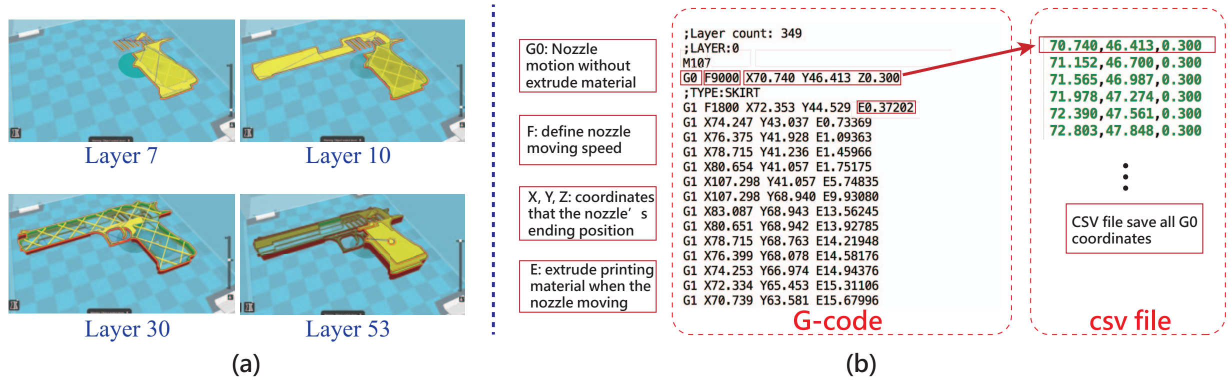
3.1 3D printing basics

Fig. 3 illustrates the general procedure of 3D printing. The complete printing process happens in two domains, cyber domain and physical domain. The cyber domain always resides in a host machine, such as a computer. In order to instruct a 3D printer, a 3D object model is created through computer aided design (CAD) software and converted to the standard object file (STL). The computer aided manufacturing happens in a controlling software. The controlling software of a 3D printer (Ultimaker Cura [10] in our case) decodes the STL file and the user is able to customize the printing setting through its GUI. Once the user initiates the printing, the controlling software slices the decoded 3D model into uniform layers and generates a data stream called G-code, which is a numerical programming language that humans use to instruct a machine to operate. A G-code file contains all the information that the 3D printer needs to print an object. It is the most widely used file format containing all the control commands including shape, dimensions, and volume. In the physical domain, the data stream feeds the 3D printer to direct the printer's movements and actions. And the 3D printer's nozzle extrudes the melt material to form the lines while the positioner follows the order encoded in the G-code.

3.2 Two scenarios in 3D printing object recognition

We define two working scenarios for an intelligent 3D printer to detect the printed objects.

G-code interpretation. In this scenario, we consider that the 3D object recognition module has full access to the G-code. The host computer sends the G-code to the 3D printer and a copy to the 3D object recognition module as the blue arrow in Fig. 2. The module transforms the G-code internally to images through which the module recognizes the object before it is printed. If the object is a weapon, the module will send a control signal to the 3D printer to terminate the printing process. In this method,


Layer 7
Layer 10
Layer 30
Layer 53
(a)
G0: Nozzle motion without extrude material
F: define nozzle moving speed
X, Y, Z: coordinates that the nozzle' s ending position
E: extrude printing material when the nozzle moving
G-code
csv file
CSV file save all G0 coordinates
(b)

```
;Layer count: 349
;LAYER:0
M107
G0 F9000 X70.740 Y46.413 Z0.300
;TYPE:SKIRT
G1 F1800 X72.353 Y44.529 E0.37202
G1 X74.247 Y43.037 E0.73369
G1 X76.375 Y41.928 E1.09363
G1 X78.715 Y41.236 E1.45966
G1 X80.654 Y41.057 E1.75175
G1 X107.298 Y41.057 E5.74835
G1 X107.298 Y68.940 E9.93080
G1 X83.087 Y68.943 E13.56245
G1 X80.651 Y68.942 E13.92785
G1 X78.715 Y68.763 E14.21948
G1 X76.399 Y68.078 E14.58176
G1 X74.253 Y66.974 E14.94376
G1 X72.334 Y65.453 E15.31106
G1 X70.739 Y63.581 E15.67996
```

```
70.740,46.413,0.300
71.152,46.700,0.300
71.565,46.987,0.300
71.978,47.274,0.300
72.390,47.561,0.300
72.803,47.848,0.300
```

Figure 4: (a) Layer view from Ultimaker Cura. The STL file is decoded and the G-code is generated. The layer view of the software demonstrates the expected printing results (b) G-code and extracted csv files. G0 indicates the starting coordinates of each printing movement and we save all G0 coordinates in the csv files. Other commands are explained in the figure but not considered in this work as they do not depict the contour of the object.

the malicious activity in 3D printing is prevented at the beginning stage.

Cumulative visual recognition. In the above scenario, the 3D object recognition module stores the copy of the complete G-code file which is usually huge and the recognition module has to transform the G-code to images. To maintain a lightweight 3D object recognition module, we consider an alternative scenario that the module will not handle the G-code directly, but monitor the printing process to recognize the object. In this scenario, three cameras are mounted to capture three perpendicular view of "screenshot" for the printed object in $xy-$, $xz-$ and $yz-$surfaces. As the object is being printed layer by layer, the recognition module recognizes the object with higher and higher confidence. When the module is confident on that the object is a weapon by cumulatively observing any view of printed object, the module will send a control signal to the 3D printer to terminate the printing process.

3.3 Transforming prior knowledge to images

3D printing is a procedure of successively adding each layer's material from beginning to the ending position as Fig. 4(a) showed. The controlling software (Ultimaker Cura [10] in our case) decodes the STL file and generate layer-wise commands in the G-code format for the 3D printer. The G-code files is composed of lines which define how the printer nozzle's behaviors for the corresponding layer, and is fed into a 3d printer sequentially. We extract useful coordinates from G-code and save into a csv file. Fig. 4(b) shows example G-code that Ultimaker Cura generates and the csv file extracted from it.

As we design two working scenarios for a intelligent 3D printer, we collect two sets of images to fit corresponding scenarios. For G-code interpretation, we construct a dataset in which the objects' projections are obtained at completion stage (a completion stage means recognition module has full access to the object's G-code which can be interpreted as a completely printed object), while for cumulative visual recognition, we prepare a dataset that

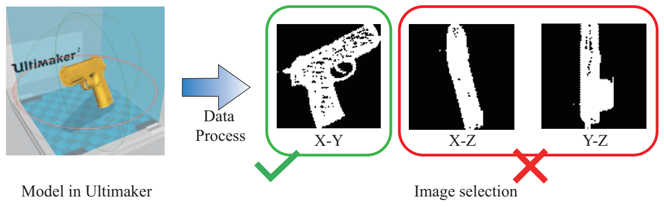


Figure 5: A G-code is processed to projections on three planes, $xy-$, $xz-$ and $yz-$.

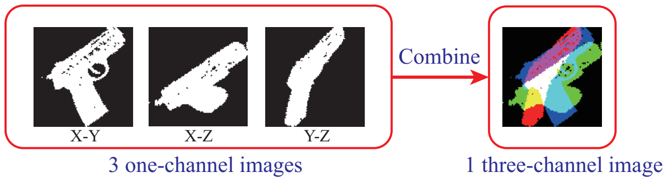


Figure 6: The process of generating a 3-channel image.

contains the projection images describing the printing procedure. The two construction approaches to fulfill above two objectives are shown as follows:

Objects' projections at completion stage. For the first scenario, given that the entire G-code is obtained, we can process it into 3 images from its $xy-$, $xz-$ and $yz-$projections for its completion stage. From Fig. 5, the object is a illegal gun and we get three projections images from it. At first, only the projection that is recognizable to human can be saved ($xy-$plane image in Fig.5), we keep this projection and abandon the others.

However, the other two projections that we discard catch our attention. As most deep learning approaches reveal that some implicit information is not recognizable by human, we keep the other non-recognizable two projection images to reserve the integrity of feature space. Fig. 6 illustrates that we combine three single-channel grayscale projections into one three-channel RGB image. We can observe that the three-channel image is hard to comprehended by humans. For computers, however, three-channel images enlarge the feature space of the object, and are supposed to give better performance for the recognition system.

Table 1: Dataset specifications for the cumulative visual recognition task. The number of layers information is based on the different postures of the 3D object in controlling software.

Class	Minimum Number of Layers	Maximum Number of Layers	Average Number of Layers	Number of Images
Pistol	217	1,020	678	85,362
Revolver	165	998	599	75,028
Special Revolver	64	1,021	592	74,092
Toy Gun	173	953	526	65,941
Gun Part	155	746	452	55,952
Gun-like Object	217	1,050	678	85,568
Cup	439	683	538	73,290
Father's Day Trophy	177	673	373	46,098
Horse	166	454	343	37,710
Portal Gun	294	859	578	72,636

Sequencing the objects' projections. For the second scenario, the 3D printer is supposed to observe the printing process from beginning to the end. The image sub-dataset in the first scenario will not fit for this job, since the projection images are collected from the completion stage of the object and cannot reflect the actual procedure of the printing. Learning from the images in the first scenario, without the G-code, a 3D object recognition module can only recognize the object when the printing procedure is finished through the projections captures by cameras. To mimicking printing procedure, we separate csv files by extracting the layer-wise information from the G-code, and accumulating csv files layer by layers which just like the printing procedure. The extracted image sequence that depicts the printing procedure is shown in Fig. 7. We can see that this sequence-based dataset properly describes the printing procedure. The pixels in the image are accumulating alongside as the printing procedure goes on. With a deep learning model using the image sequences, the 3D object recognition module is able to recognize the object through mounted cameras without G-code information. Table 1 shows the technical specifications of our sub-datasets which fit the cumulative visual recognition task.

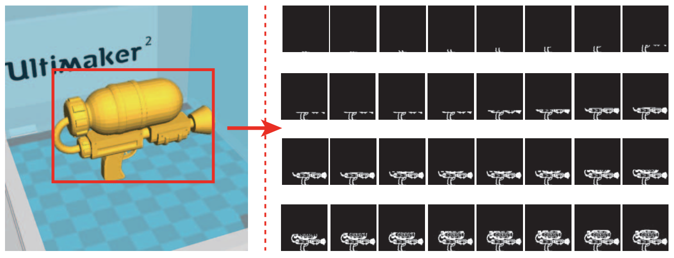


Figure 7: Selected images from one sequence for example model (splatter).

4 Case study: early-stage 3D object recognition with G-code

In the first scenario, the recognition module has full access to the G-code, we consider that the module is capable of recognizing the object before printing starts. We apply convolutional neural network on first scenario dataset.

4.1 Model

We build a convolutional networks [11] to conduct the classification on our three-channel dataset. The proposed network consists of three convolution layers followed by global pooling layer, and then the pooled feature maps are convolved by ten 1×1 filters to obtain the classification logits. The detail about the network architecture is as below. The three convolutional layers consists of 128 5×5, 256 3×3, and 512 1×1 filters. A Log-Sum-Exp (LSE) pooling layer proposed in [12] is utilized after the first two convolution layers. The LSE pooled value x_p is defined as,

$$x_p = \frac{1}{r} \cdot \log[\frac{1}{S} \cdot \sum_{m,n \in \mathbb{S}} exp(r \cdot x_{mn})] \tag{1}$$

where x_{mn} is the activation value at (m, n), (m, n) is one location in the pooling region $\mathbb{S}$, and $S = s \times s$, $s = 3$ is the pooling filter size in $\mathbb{S}$. By adjusting the hyper-parameter r, the pooled value ranges from the maximum in $\mathbb{S}$ (when $r \to \infty$) to average (when $r \to 0$). We derived the probabilities of predictions by softmax activation function as follows:

$$y_i = \text{softmax}(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j)} \tag{2}$$

where x_i is the logit for a particular class i. We train the network using cross-entropy loss , which can be represented as,

$$\mathcal{L}_{y'}(y) = -\sum_i [y'_i \log(y_i) + (1 - y'_i) \log(1 - y_i)] \tag{3}$$

where y'_i represents the actual label for class i and y_i is the i-th element in the prediction output.

4.2 Experimental results

Dateset and training. The dataset in this scenario consists of $62,200$ 128×128 RGB (3-channel) images divided into $51,840$ training images and $10,360$ testing images. We add L2 regularization [13] to the loss function to prevent overfitting. We optimize the model by Adam [14] method on an Nvidia GTX1080 GPU. The model is implemented in TensorFlow [15].

Table 2: Classification accuracies using 3-channel projection images. Bold text denotes the best results.

Dataset	Channel	r	Accuracy (%)
first scenario	3	0 (AVG)	91.7
		0.1	92.9
		5	90.9
		10	**93.6**
		∞ (MAX)	91.9

Classification results. We apply the proposed network on the 3-channel dataset. We vary the r value to adjust pooling value during training, r is assigned with value of 0.1, 5, and 10. We also compare the applied LSE pooling with average pooling and max pooling. Table 2 shows the evaluation results of our experiment. When r is very small as 0.1, the pooling is close to the method of average pooling. The accuracy is 92.9%, which is better than the average/max pooling results. When we increase r to 5, the performances reach the bottom as 90.9% Then it achieve the best performance at $r = 10$, i.e. 93.6%. Overall, the use of LSE pooling improve the performance compared with using simple average/max pooling with a good choice of r.

5 Case study: early-stage 3D object recognition with cumulative visual monitoring

In the second scenario that the 3D object recognition module has no access to the G-code, we consider that the module receives the data stream of images from the embedded cameras perpendicular to $xy-$, $xz-$, and $yz-$planes facing the printing workspace. The module executes three recognition threads which keep reading the object's projection images correspondingly, and terminates the printing process when any of aforementioned threads recognizes the object is a weapon.

5.1 Model

Our goal is to enable the system to read a stream of images and gives the prediction every time it receives an image. We keep track of moving averages for the predictions of all classes and take it as the fused prediction at step t. When the prediction of a certain class C_i have been made above the confidence threshold th, the model makes a decision to recognize the object is within class C_i. The images are converted to vector representations, and the recurrent neural network (RNN) model is utilized to handle the sequence of image representations. The proposed architecture is summarized in Fig. 8.

Image representation. In this work, we use a simple 3-layer CNN similar to that in the last section to extract the image features. At each step t of the sequence, the grayscale image I_t with shape $h \times w \times 1$ is computed for its representation as follows,

$$v = W_T[\text{CNN}_\theta(I_t)] + b_T, \tag{4}$$

where $\text{CNN}_\theta(I_t)$ first transforms the image I_t into $h' \times w' \times c'$ feature tensor with respect to CNN parameters θ,

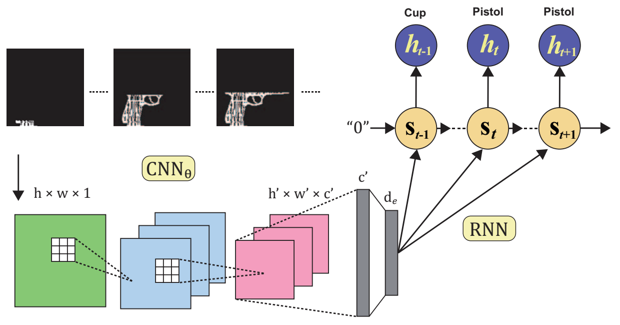


Figure 8: RNN Model overview.

then the tensor is globally pooled on spatial dimensions (width and height) to obtain a c'-dimensional feature. W_T is the transformation matrix which has dimensions of $d_e \times c'$, where d_e is the size of the embedding space. b_T is the bias. Thus each image I_t at step t is represented as a d_e-dimensional vector v.

Sequence representation. We propose to use an RNN to model the sequence of projection images in the order of observation during printing. Please note that as our aim is not to generate predictions for the complete sentence of images and our goal is to make a confident recognition for the object, we use the forward RNN instead of a Bidirectional RNN (BRNN) [16] to follow the nature of our objective. We use two different RNN cells in this work, Long short-term memory (LSTM) [17] and Gated recurrent unit (GRU) [18]. The hidden units in the RNN cells are set to be 512. The output o_t of the RNN at step t is connected with the final classification layer, a fully-connected layer with dimension C, where C is the number of classes. We use the truncated backpropagation through time (BPTT) learning algorithm [19] to compute parameter gradients on short subsequences of the training image embeddings. The average number of layers is $\sim$ 500 and the corresponding average length of the image sequence is $\sim$ 100 as we collect the images every 5 layers. During training, a fixed step T_{bptt} (20, typically in this work) is adopted to handle the relatively long length of the sequence. We use average softmax cross-entroy in the truncated BPTT window. And we use moving average value of the predictions during testing. When the average softmax output value for any class C_i exceeds th, we stop the inference for the sequence and take the C_i as the label prediction of the sequence. If none of the moving average exceeds th until the last image of the sequence, we take the class with the maximum softmax value at the last step as the label prediction.

5.2 Experimental results

Dateset and training. The dataset in this scenario consists of 4,752 [1] 1-channel 128×128 grayscale image sequences categorized to 10 classes. Each sequence contains as many as 200 images. The training and test images are randomly selected from each class with the partitioning

[1]we only use the sequence variants in the first quadrant of the 3D Euclidean space.

Table 3: Benchmark results using 1-channel projection image sequence. Bold text denotes the best results.

RNN cell	th	Accuracy (%)	avg. stop step
LSTM [17]	0.1	12.5	17.5
	0.3	46.8	39.6
	0.5	69.1	58.3
	0.7	**77.4**	73.4
GRU [18]	0.1	14.6	19.4
	0.3	44.7	41.2
	0.5	68.5	50.1
	0.7	**78.4**	70.8

ratio of 5. We use the same optimization method and platform as the first scenario.

Recognition Results. We apply the proposed approach on the 1-channel image sequences. Table 3 shows the evaluation results of our experiment. We report the the average step indices when the inference for one sequence stops. GRU and LSTM achieve approximately equivalent performance in this task. We can observe from Table 3 that with a small th, it is too quick for the system to make a decision. The prediction results are quite low. As the th gets bigger, the accuracy performance increases as the average stop step index gets larger as well. Because the moving average values of softmax outputs indicate the prediction confidence for each class. With more steps, the system gets more information about the object and is able to make more confident predictions. A large th ensures that the inference process accepts enough images to make confident predictions before stop. Since the unconfident predictions give low values for a specific class, the inference stop does not happen easily with a large th. Please note that we are able to achieve a better accuracy than 78.4% with higher confidence than 0.7, but more steps are needed. The setting of the confidence threshold th depends on practical requirement of accuracy or the stop step index. The 3D printer can save printing material by stop the printing process in early steps at the risk of mis-recognition of the object.

6 Conclusion

The abuse of 3D printing technology to produce illegal weapons requires an intelligent 3D printer with early stage malicious activity detection. The 3D printer should identify the objects to be printed, so that the manufacturing procedure of an illegal weapon can be terminated at early stage. The lack of large-scale dataset obstructs the development of the intelligent 3D printer equipped with deep learning techniques. The construction of 3D printing image database in such scale with the recognition benchmarks has not been addressed until this work. We attempt to design two working scenarios for an intelligent 3D printer and provides corresponding image datasets (tens of hundreds and tens of thousands images). We also conduct quantitative performance benchmarking on ten 3D object recognition given single images and image sequences using C3PO database. This work brings the new thought of designing an object-aware 3D printing system. As the 3D models are highly customized and diverse, building a robust recognition system remains a tough task. For the furture work, C3PO will include more common 3D models especially for firearms.

References

[1] C. Bayens, G. L. Le T, R. Beyah, M. Javanmard, and S. Zonouz, "See no evil, hear no evil, feel no evil, print no evil? malicious fill patterns detection in additive manufacturing," in *26th USENIX Security Symposium)*, 2017.

[2] C. Song, F. Lin, Z. Ba, K. Ren, C. Zhou, and W. Xu, "My smartphone knows what you print: Exploring smartphone-based side-channel attacks against 3d printers," in *Proceedings CCCS*, 2016.

[3] C. R. McCutcheon, "Deeper than a paper cut: Is it possible to regulate three-dimensionally printed weapons or will federal gun laws be obsolete before the ink has dried," *U. Ill. JL Tech. & Pol'y*, p. 219, 2014.

[4] R. K. Little, "Guns don't kill people, 3d printing does: Why the technology is a distraction from effective gun controls," *Hastings LJ*, vol. 65, p. 1505, 2013.

[5] "3d printed bullets developed and tested by russian researchers," https://3dprint.com/156003/russia-3d-prints-bullets/.

[6] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *CVPR*, 2012.

[7] Y. Xiang, R. Mottaghi, and S. Savarese, "Beyond pascal: A benchmark for 3d object detection in the wild," in *WACV*, 2014.

[8] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on.* IEEE, 2015, pp. 922–928.

[9] E. Johns, S. Leutenegger, and A. J. Davison, "Pairwise decomposition of image sequences for active multi-view recognition," in *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on.* IEEE, 2016, pp. 3813–3822.

[10] "Ultimaker cura software," https://ultimaker.com/en/products/ultimaker-cura-software.

[11] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[12] P. O. Pinheiro and R. Collobert, "From image-level to pixel-level labeling with convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1713–1721.

[13] P. Bühlmann and S. Van De Geer, *Statistics for high-dimensional data: methods, theory and applications.* Springer Science & Business Media, 2011.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[15] M. Abadi, A. Agarwal, P. Barham, and el.at., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[16] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[19] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le *et al.*, "Large scale distributed deep networks," in *NeurIPS*, 2012, pp. 1223–1231.