Detecting Prohibited Items in Airport X-rays

Alex Ferri, Ed Atkinson, Ed Loughrey, Ludo Oliver, Max Chesters, Arthur Yamaguchi

***Abstract*—This report developed two CNN classifiers, which evaluated the safety of bags that go through X-rays during airport security screening. The first provided a binary output of 'safe' or 'not safe', where the latter classified an X-ray image into 'gun', 'hammer', 'knife', 'scissors' or 'null'. Both models were trained and tested on a subset of the SIXRay10 dataset [1]. First, the data was transformed for uniformity and the class imbalance was dealt with using stratified sampling, and a custom weighted loss function. The architecture of the models was chosen to be made up of three convolutional layers. We considered three optimisers: SGD, Adam and RMSprop, and found SGD to be the most appropriate. The Binary classifier used a standard cross entropy (CE) loss function and the Multi-Class model explored a custom loss (CL) function as well. The Binary classifier achieved near perfect accuracy (99.7%). The Multi-Class classifier accurately identified non-threats, but struggled to categorise the prohibited items into what specific class they belonged to. The binary classifier was found to have a significantly higher F_2 score (0.9986) than either of our Multi-Class models (0.86 and 0.87). In any related future projects, access to improved computing resources would allow exploration into the impacts of further developments to the current model.**

I. Introduction

Baggage screening plays a critical role in protecting public spaces from the threats posed by violence, terrorism and smuggling operations. With international tourism having increased by over 100% in the last 30 years [2], growing importance is being placed on identifying prohibited items quickly and accurately. Manual screenings are dependent on security staff and their individual knowledge and experience. Fatigue, working conditions and staff competence all introduce unacceptable errors into this system, which can be both invasive for passengers and dangerous towards the public.

The deployment of machine learning systems offers a promising solution to these security concerns, with Convolutional Neural Networks (CNN) presenting an effective tool for image classification [3]. Implementing automatic classification of X-ray scans as either 'safe' or 'prohibited' would allow for rapid assessment and decision-making at security checkpoints. These automated security systems would then significantly streamline the inspection process, minimizing both passenger inconvenience and risks levels in such environments. Furthermore, classification of prohibited items into individual items such as 'scissors' or 'hammer' could allow for airport operators to respond appropriately based on the threat presented by such items.

In this paper we detail the development of two CNN models that classify prohibited items in airport X-rays: a binary and a multi-class classifier. We measure and evaluate the performance of several implementations of both models across a range of hyper-parameters.

II. Literature Review

There have been many attempts at using computer vision to solve the problem of Prohibited Item Detection (PID). Akçay et. al pioneered the integration of deep learning in PID in 2016, using a pre-trained GoogleNet model for object classification from X-ray baggage scans [4].

In 2018, they further conducted PID using various deep learning approaches as well as the bag-of-words model [5]. The study concluded that deep learning approaches such as CNNs were more effective in this problem than traditional machine learning methods. However, the model reported poor performance in instances where images contained several prohibited items - these were frequently misclassified. This may alert security staff to a much less significant threat than that of the other items mentioned. In this report, we combat this by only scanning for only high risk items, such as handheld weapons or explosives.

A. Augmentation Techniques

Many of the previous methods to this problem use data augmentation to improve generalisation and robustness of the model. Webb's 2021 paper [6] examines three sets of augmentations: geometric transformations, global image processing transformations, and methods involving combining images to generate new samples. They found that the effectiveness of any of these techniques differed for each dataset used. The only one that consistently improved the model's performance was the 'RandomFlip' transformation, which randomly applies vertical and horizontal flips during training. This may be because the other methods can overly distort the images. However, the performance of the model was only explored on two datasets, so it is not clear if 'RandomFlip' improves the model on every dataset it receives. Image processing based augmentation was decided against, as color data within X-rays encodes key information about material properties [7].

B. Addressing Class Imbalance

The datasets used in similar problems tend to have much fewer positive samples (images with at least one prohibited item labelled) than negative ones. This imbalance in class distribution can greatly impact the classifier's success, as the model will be biased towards the most prevalent class, reducing accuracy. Much like in baggage scans, the majority of medical images will belong to the safe ('null') class, and in both contexts biases in prediction can have severe consequences. To de-bias these predictions, in 2019, An et al. [8] implemented stratified sampling to create subsets of training data, with an equal class distribution when classifying medical images. Within the context of X-ray threat detection,

this will involve selecting images such that the training data is uniformly distributed across all of the chosen classes.

Alternatively, Miao's 2019 paper [9] navigates this issue by building their own loss function. In the case of this heavy class imbalance, regular loss functions such as Euclidean loss, $\mathcal{L}\{\boldsymbol{y_n^*}, \boldsymbol{y_n}\} = |\boldsymbol{y_n^*} - \boldsymbol{y_n}|^2$ and the Binary Cross-Entropy (BCE) loss $\mathcal{L}\{\boldsymbol{y_n^*}, \boldsymbol{y_n}\} = -[\boldsymbol{y_n^*}^T \log \boldsymbol{y_n} + (1 - \boldsymbol{y_n^*})^T \log (1 - \boldsymbol{y_n})]$ are less effective, biasing the network towards the negative examples. By modifying the loss function to heavily penalise misclassifications of the minority classes, the impact of the class imbalance is significantly reduced. After conducting research into the published PID deep learning models, no examples were found that compared the effect of implementing a custom loss function to stratified sampling methods. The comparison of these techniques will therefore be an area of focus of our initial investigation.

III. Methodology

A. Data Pre-processing

For the purposes of this report, a subset of the chosen dataset was selected, containing only JPEG images belonging to the following classes: gun, knife, hammer, scissors, and null (where the scan contained no prohibited items). All images were resized to 224×224 pixels to ensure uniform input data. Furthermore, all images were normalised to have a mean of 0 and standard deviation of 1 across each of the RGB colour channels. This makes the training of the model more stable and efficient by ensuring all data is consistent [10]. The result of this can be seen in Figure 1.

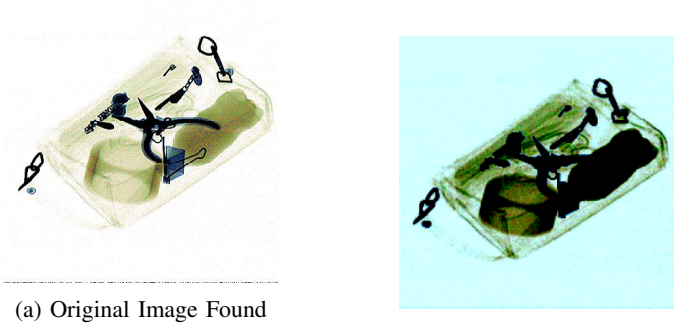
A problem we came across was that the chosen dataset had a considerable class imbalance - Figure 2 shows that the null class of the training set is significantly larger than all the other classes. Training on an imbalanced set could decrease the accuracy of the model, as a naive implementation may train to predict 'null' for every image in the test set. In the real world, this could translate to a prohibited item such as a gun making it through security, which presents a major problem. To mitigate this, two previously mentioned strategies are deployed: stratified sampling and a custom weighted loss function. Stratified sampling trains the model on a subset of the original data, with the aim of producing batches of data with a uniform class distribution. This works by taking weighted random samples based on the original distribution.

For each class i, the selection weights w_i were calculated to be

$$w_i = \frac{1}{N_i}, w_j = w_{y_j}, \tag{1}$$

where N_i is the number of samples, w_j is the weight assigned to an individual sample j, and y_j is the class label of sample j.

It should be noted that augmentation was not chosen as a technique to address class imbalance, as it expands the size of our original dataset ($\approx$ 9800 images) beyond the capacity of our equipment.



(a) Original Image Found in the Dataset

(b) Transformed Image

Fig. 1: Original Image Taken from the Dataset and the Same Image Transformed when Loaded into the Models

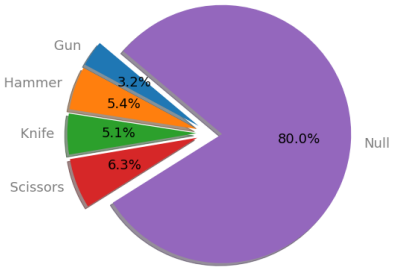


Fig. 2: Class Distribution of the Training Set Prior to Stratification

B. Custom Loss Function

In classification, there are two possible types of errors: false positives and false negatives (type-I and II respectively). Type-I errors occur when the model incorrectly predicts the positive class for an input that is actually negative; Type-II errors occur when the model incorrectly predicts the negative class for an input that is positive [11]. In our context, the negative class is where there are no prohibited items, while the positive class is any class with a prohibited item in it.

In PID, type-II errors present a much greater threat towards public safety, especially in comparison to the minor inconvenience experienced by a passenger flagged by a type-I error. For this reason, we developed a custom loss function that penalises false-negative predictions.

The Custom Loss Function operates with the same normalised weights as the stratified sampling inverse law method shown in Equation 1. The model becomes more sensitive to classes which are underrepresented in the dataset, overcoming bias without the need for modified sampling methods.

In this report we will compare the effects of using our Custom Loss (CL) function, with the standard Cross Entropy (CE) Loss function used in conjunction with stratified sampling.

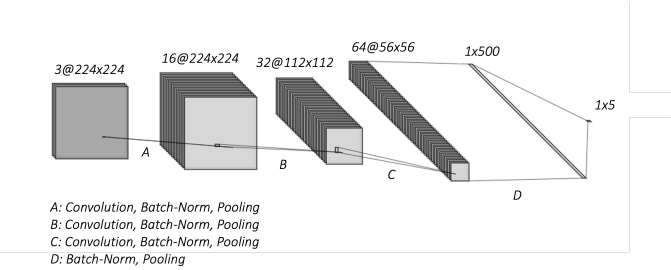


Fig. 3: Architecture of the Multi-Class Classifier

C. Architecture

Both models in this report use nearly identical architecture. Within both models there are three convolutional layers, each followed by a batch normalisation layer and a max pooling layer. Three convolutional layers were chosen to strike a balance between feature selection and computational complexity. The convolutional layers use kernels of size 3×3 as well as padding and stride values of 1 to maintain the size of the feature maps (inputs and outputs of the CNN). The previously mentioned RGB normalisation techniques are applied in each batch normalisation layer to standardise the outputs of the previous layer, allowing for faster and more stable training [10]. The max pooling layers reduce the size of the feature maps by half, decreasing the computational load and helping to abstract the features of the images [12]. Following the convolutional layers, two fully-connected layers are implemented to reduce the dimensionality in preparation for the model to make a prediction of the image class. A dropout layer with a rate of 0.25 is implemented before each fully-connected layer. This dropout layer was chosen to reduce overfitting to the training data, improving the models' ability to generalise to unseen data[reference]. A ReLU activation function is then applied after each of the convolutional layers, as well as the first fully-connected layer, to provide non-linearity to the model.

The key difference in architecture between each model is within the second fully connected layer, in which the Multi-Class model aggregates the inputs into a length 5 vector instead of the single value outputted for the Binary classifier. Figure 3 shows a visual representation of the architecture of the Multi-Class model. The original dataset [1] was modified for use within the Binary classifier by aggregating all unsafe classes into a single 'prohibited' class. A training-test-validation split of 80%-10%-10% was used for both models. Training is performed over 3 epochs, each consisting of minibatches of 50 images.

D. Optimisers

In order to maximise the performance of our models, we experimented with a number of optimisers and their hyperparameters. Three commonly-used optimisers were chosen: Stochastic Gradient Descent (SGD), Adam, and RMSprop. Across each optimiser, learning rates were varied, with investigations also being conducted into the weight decay parameter for RMSProp and Adam, and the momentum value for SGD. The validation loss was calculated for each iteration of optimiser and hyperparameter as a means of measuring the efficacy.

SGD, the simplest of the three optimisers, updates model parameters θ by moving in the opposite direction of the gradient ∇_θ of the loss function $J(\theta)$ with respect to the parameters, scaled by a learning rate η: $\theta = \theta - \eta \nabla_\theta J(\theta)$. Adam adjusts the learning rate for each parameter based on the first (m_t) and second (v_t) moment estimates of the gradients: $m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_\theta J(\theta)$, $v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla_\theta J(\theta))^2$, and the parameters are updated with $\theta = \theta - \eta \cdot m_t / (\sqrt{v_t} + \epsilon)$, where β_1 and β_2 are the decay rates for these moment estimates, and ϵ is a small constant to prevent division by zero. RMSProp, on the other hand, divides the learning rate for a parameter by a running average of the magnitudes of recent gradients for that parameter: $v_t = \gamma v_{t-1} + (1-\gamma)(\nabla_\theta J(\theta))^2$, $\theta = \theta - \eta / (\sqrt{v_t} + \epsilon) \nabla_\theta J(\theta)$, where γ is the decay rate.

Whilst RMSProp and Adam both provide opportunity for faster convergence due to their adaptive learning rates, both are computationally intensive, especially in comparison to SGD [13]. Adam can also be prone to overfitting due to poor feature selection, decreasing its potential for reusability [14]. SGD provides a solution to this, as past research has shown that models optimised with SGD often generalize better to unseen data, than those with adaptive learning rates [15]. This could be beneficial for future research that looks to expand on our model and findings. Together, these three optimisers cover a number of problems that our models may encounter, such as slow convergence, overfitting, and the need for model robustness and reusability. Experimentation across the three will aid in identifying the best training strategy for our CNNs.

E. Performance Measures

To quantify the performance of the models, we evaluate the loss and accuracy for both the training and validation sets. As mentioned in subsection III-B, we also want to minimise type-II errors as these are a greater threat to public safety compared to type-I errors. Recall is a measure of a classification model's ability to identify all relevant instances, and therefore, gives us an indication of the model's ability to avoid type-II errors [16]. Precision is a measure of a classification model's ability to identify only the relevant instances, and therefore, gives us an indication of the model's ability to avoid type-I errors [16]. To assess the performance of the models using these errors, we will use the F_β-score as this allows us to place more emphasis on recall rather than precision [17]. The F_β-score is calculated by,

$$F_\beta = \frac{(1+\beta^2) \cdot precision \cdot recall}{(\beta^2 \cdot precision) + recall}, \tag{2}$$

$$\text{where } precision = \frac{TP}{TP+FP}, \; recall = \frac{TP}{TP+FN},$$

and β is a weight controlling the emphasis on recall and precision. As we care more about recall, we can set $\beta = 2$ which means recall is twice as important as precision [17].

The F_β score will be computed for every class in both the Binary and Multi-Class classifiers when the test set is applied. Utilising these scores, an overall performance measure can be derived using two approaches: Macro Average (MA) and Weighted Average (WA). MA computes an unweighted mean of all the metrics calculated for each class [18]. On the other hand, WA computes a weighted mean based on the class sizes, providing a balanced evaluation that considers the contribution of each class [18]. Given that the test set contains the same class imbalances present in the training set, our focus is primarily on WA as it considers the class sizes, providing a more comprehensive evaluation. The WA can be calculated using

$$W = \frac{\sum_{i=1}^{n} w_i X_i}{\sum_{i=1}^{n} w_i}, \tag{3}$$

where W represents WA, w_i represents the weights for class i, X_i represents the metrics for class i, and n represents the total number of classes.

IV. Experimentation

A. Optimiser Selection

Sixteen combinations of hyperparameters were tested for every optimiser; Figure 4 and Figure 5 shows the combinations that produced the lowest loss for each. Figure 4 indicates that SGD with a momentum of 0.99 and learning rate of 0.001 is the best hyperparameter for the Binary classifier. Furthermore in Figure 5, SGD with a momentum of 0.5 and a learning rate of 0.01 is the best optimiser-hyperparameter combination for the Multi-Class classifier, although it is only slightly better than Adam after the third epoch. Hence, SGD with the specified parameters were used for both models.

For both hyperparameter and optimiser combinations the loss function reaches a minima at the third epoch. Due to computational constraints, both models are only trained for 3 epochs.

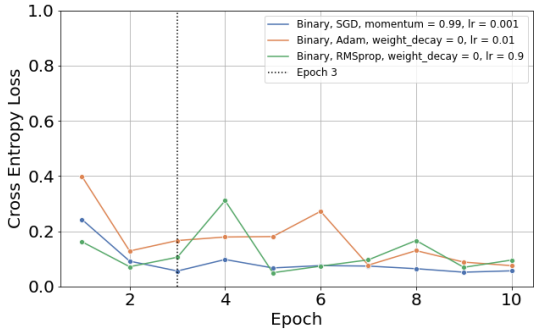


Fig. 4: Lowest Validation Loss for Each Optimiser for Binary Classifier.

B. Cross-validation

Figure 6 shows the training and validation accuracies for the Multi-Class classifier with both the CE Loss function and the CL function. The validation curve of the CE and CL

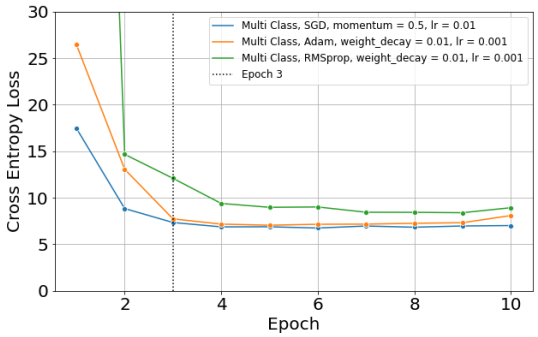


Fig. 5: Lowest Validation Loss for Each Optimiser for Multi-Class Classifier.

function swiftly reaches a plateau, achieving an accuracy of $\approx 85\%$ within the first epoch. Figure 7 shows the training and validation losses for the CE function, with the validation curve potentially showing a slight upwards trend and increase in volatility after the first epoch.

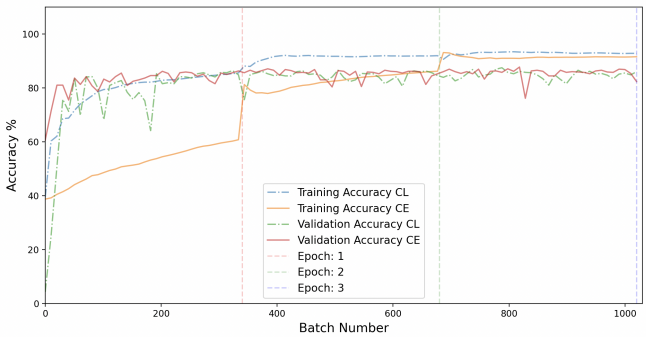


Fig. 6: Training and Validation Accuracies using the Cross Entropy Loss Function (CE) and the Custom Loss Function (CL) for the Multi-Class Classifier

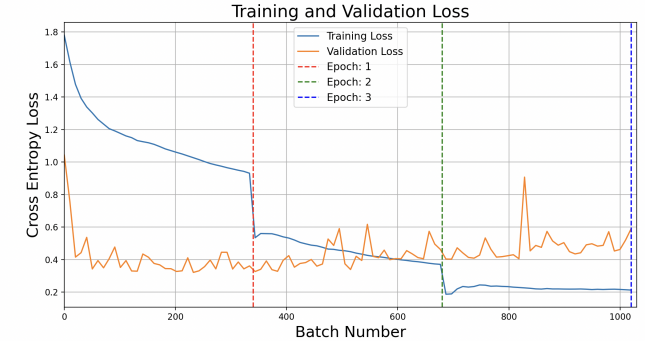


Fig. 7: Training and Validation Loss using the Cross Entropy (CE) Loss Function for the Multi-Class Classifier

Figure 8 shows the training and validation accuracies for the Binary classifier in conjunction with the CE Loss Function. It shows the validation accuracy converges to $\approx 100\%$ within

one epoch. Unfortunately, we were unable to implement the CL function with the Binary model.

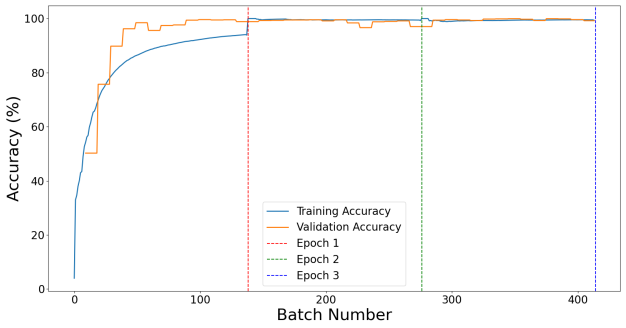


Fig. 8: Training and Validation Accuracies using the Cross Entropy Loss Function for the Binary Classifier

C. Confusion Matrices

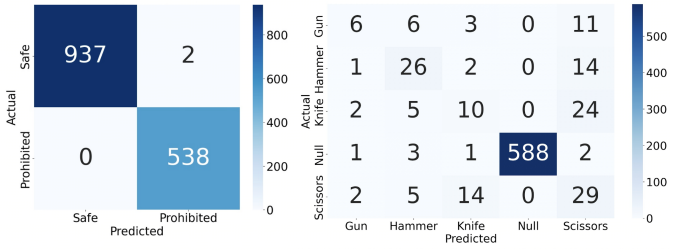


Fig. 9: Confusion Matrices for Binary and Multi-class Classifiers

Figure 9 shows the confusion matrices for the Binary and multi-class models. The Binary model correctly classifies 99.86% of its inputs. We observe there are just 2 cases of type-I error, but more notably, there are zero cases of type-II error.

The Multi-Class model has 100% precision in the null class, although its accuracy across the prohibited classes is much poorer with only $\approx 35\%$ of images being correctly labelled. Figure 9 also shows the absence of any type-II errors for this model as well.

D. Classification Reports

Table I shows the precision, recall and F_2 scores for the Multi-Class classifier, for both the CL and CE Loss Functions, along with the weighted and macro averages across each class for each score. It also shows the testing accuracies for the two models. Table II shows the same scores for the Binary classifier in use with the CE Loss Function.

It is notable that the testing accuracy of the model on the Binary classifier is significantly higher than that seen in both CL and CE models in the Multi-Class classifier.

Class	Precision		Recall		F_2-score		Support
	CL	CE	CL	CE	CL	CE	-
Gun	0.28	0.50	0.46	0.23	0.41	0.26	26
Hammer	0.46	0.58	0.51	0.60	0.50	0.60	43
Knife	0.23	0.33	0.20	0.24	0.21	0.25	41
Null	1.00	1.00	0.99	0.99	0.99	0.99	595
Scissors	0.50	0.36	0.38	0.58	0.40	0.52	50
MA	0.49	0.55	0.51	0.53	0.51	0.53	755
WA	0.87	0.88	0.86	0.87	0.86	0.87	755
Acc.	Custom 86%				CE 87%		

TABLE I: Classification Report for the Custom Loss Function (CL) and Standard Cross Entropy (CE) Function. **Acc**. = Testing Accuracy, **MA** = Macro Average, and **WA** = Weighted Average.

Class	Precision	Recall	F_2-score	Support
Safe	1.0000	0.9979	0.9983	939
Prohibited	0.9963	1.0000	0.9993	538
MA	0.9981	0.9989	0.9987	1477
WA	0.9987	0.9986	0.9986	1477
Acc.	99.7%			

TABLE II: Classification Report for the Standard Cross Entropy Function. **Acc**. = Testing Accuracy, **MA** = Macro Average, and **WA** = Weighted Average.

V. Discussion

Figure 4 and Figure 5 suggest that SGD produces the least loss for the CE loss function, whilst also converging the quickest, reaching the lowest loss after the third epoch. This is different to what we expected, with previous findings suggesting that RMSProp and Adam converge quicker than SGD [13]. This could be due to the sparse nature of the features defining prohibited items within our dataset, which could make it difficult for Adam and RMSProp to extract features. SGD has also been shown as a more generaliseable optimiser [14], allowing us to potentially extend our model into other related PID problems. The discrepancy between the Multi-Class validation values in Figure 5 compared to the final model, in Figure 7, is a result of training on a subset of the data for hyperparameter tuning, due to computational restraints.

The main pitfall of the CE Loss model is potential over-fitting, suggested by the slight upward trend in the validation loss after the first epoch in Figure 7. This indicates the model is learning specific patterns in the training set at the cost of performance on the validation set. However, a decrease in validation accuracy would be expected for this to be true; Figure 6 shows that this is not the case. Therefore, further exploration would need to be conducted to confirm overfitting, as it is difficult to determine this solely from our results. Using a smaller train and larger validation and testing split would mean there is a broader and more diverse set of data to test over, which would improve the reliability of the performance metrics shown in subsection III-E. This would enhance our ability to understand if the model is generalising well on the validation set and to identify potential overfitting. Furthermore, our use of regularisation techniques such as weight decay

and dropout in each model's architecture contributes to the reduction of each models propensity to overfit.

Figure 6 and Figure 8 show that the validation accuracy is higher for the Binary model than the Multi-Class model when used with the CE loss function. The near-perfect validation accuracy in the Binary classifier's CE loss function suggests a strong ability to distinguish classes. However, this may also indicate a lack of complexity in the dataset, warranting further investigation.

Whilst Figure 9 shows that the Multi-Class classifier is able to detect the null class with 100% accuracy, the classification accuracy of prohibited items to their correct class is only 35%. This means this model's ability to detect the level of threat is poor, and may translate to airport security being over-prepared for the detected item. For example, there were two cases of scissors being classified as guns, and in a country such as the UK with tight gun laws, this could lead to an overreaction and a potentially dangerous situation for someone who innocently brought over a kitchen utensil. Figure 9 also shows that only 24% of knives were correctly predicted, whilst 58% were misidentified as scissors, showing that the model has the potential to underestimate the threat as well. This could lead to airport security being underprepared for possible threats. Due to this lack of ability to classify prohibited items correctly, the Multi-Class classifier has little benefits over the Binary. It should be noted, however, that both models produced no type-II errors.

The classification reports shown in subsection IV-D, derived from the confusion matrices, provide results which allow us to determine that the CE variation of the Multi-Class model is better than the CL variation. This is determined by comparing the F_2 scores for each of the variations. We see that the F_2 score for the MA and WA for the CE variation is 0.02 and 0.01 points higher than the CL variation respectively. The increase in WA suggests an improvement in the CE models ability to reduce the number of type-II errors. Although this improvement in accuracy is minimal, the reduced computational complexity of the CE model makes this model preferable for our application.

Comparing the WA F_2-score of the CE Multi-Class model to that of the Binary model, we conclude that the Binary Model is superior to the Multi-Class model, due to the notable difference of 0.129, from 0.87 to 0.9986. Finally, drawing from the overall testing accuracies for both models, the Binary Model being at 99.7% is significantly higher than the Multi-Class accuracy of 87%.

Given the higher accuracy of our current binary classifier, we conclude that it is the preferable model. The significantly higher accuracy score outweighs the utility of any extra information the current Multi-Class model provides.

VI. Future Work

Initial efforts in improving the model should begin with an expanded hyperparameter search. This would require access to significant computing power, but would allow for a more nuanced exploration of these values, and introduces potential for an improvement in performance. Access to more computing resources would also allow us to explore the impacts of data augmentation techniques upon our models. This could be combined with the inclusion of a more diverse training data set to allow for a more robust feature selection. This may produce a model with an improved ability to generalise to other PID problems.

For similar reasons, this report did not consider the computational cost of optimiser-hyperparameter combinations, focusing only on model loss. Further research could involve measuring running times for each configuration, varying batch sizes, and increasing epochs, with the aim of reducing running times.

When considering the commercial applications of each model, significant developments must be made. Any viable product would need to maintain an exceptionally low error rate, whilst running locally and in real time. Real world results from this model would also be useful to quantify the system's behaviour for low quality inputs, such a noisy scans or misaligned scanners. The ability to recognise and deal with these errors is one of the main advantages of human operators and so a robust system would have to handle such errors appropriately.

Another key consideration would be that of user privacy. Whilst continuing to train the model in situ may offer an increase in performance, privacy concerns of users who do not want the contents of their personal possessions stored within our system must be addressed. In addition to this, the implementation of a segmentation algorithm within this system could be immensely valuable. Accurate image segmentation would allow for workers to safely investigate potential prohibited items, whilst reducing the invasive nature of any searches conducted. This would be a significant step in broadening the model's commercial appeal.

VII. Conclusion

The Binary classifier analysed in this report, using a CE loss function and optimized with SGD, achieved near-perfect accuracy, effectively distinguishing between safe and prohibited items. The Multi-Class classifier accurately identified non-threats but struggled with categorisation into specific classes, leading to potential security risks.

This report has helped establish CNN models as tools for improving airport security, with current findings favoring our Binary classifier for its reliability. Further advancements could improve our Multi-Class classifier, helping improve global security infrastructure.

References

[1] DU. lll dataset. https://universe.roboflow.com/du-im6u9/lll-x8jih, sep 2023. visited on 2024-03-18.

[2] UNWTO. Number of international tourist arrivals worldwide from 1950 to 2023. https://www.statista.com/statistics/209334/total-number-of-international-tourist-arrivals/, January 2024. visited on 2024-03-19.

[3] Farhana Sultana, Abu Sufian, and Paramartha Dutta. Advancements in image classification using convolutional neural network. In *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, pages 122–129, 2018.

[4] Samet Akçay, Mikolaj E Kundegorski, Michael Devereux, and Toby P Breckon. Transfer learning using convolutional neural networks for object classification within x-ray baggage security imagery. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 1057–1061. IEEE, 2016.

[5] Samet Akcay, Mikolaj E Kundegorski, Chris G Willcocks, and Toby P Breckon. Using deep convolutional neural network architectures for object classification and detection within x-ray baggage security imagery. *IEEE transactions on information forensics and security*, 13(9):2203–2215, 2018.

[6] Thomas W Webb, Neelanjan Bhowmik, Yona Falinie A Gaus, and Toby P Breckon. Operationalizing convolutional neural network architectures for prohibited object detection in x-ray imagery. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 610–615. IEEE, 2021.

[7] Besma R Abidi, Yue Zheng, Andrei V Gribok, and Mongi A Abidi. Improving weapon detection in single energy x-ray images through pseudocoloring. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 36(6):784–796, 2006.

[8] Guangzhou An, Masahiro Akiba, Kazuko Omodaka, Toru Nakazawa, and Hideo Yokota. Hierarchical deep learning models using transfer learning for disease detection and classification based on small number of medical images. *Scientific reports*, 11(1):4250, 2021.

[9] Caijing Miao, Lingxi Xie, Fang Wan, Chi Su, Hongye Liu, Jianbin Jiao, and Qixiang Ye. Sixray : A large-scale security inspection x-ray benchmark for prohibited item discovery in overlapping images, 2019.

[10] Neha Bhati. Normalizing an image dataset for cnn. https://www.researchgate.net/post/Normalizing_a_image_dataset_for_CNN#:~:text=Normalizing%20an%20image%20dataset%20for,learn%20faster%20and%20perform%20better., September 2023. visited on 2024-03-19.

[11] Google. Classification: True vs. false and positive vs. negative. https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative. visited on 2024-03-19.

[12] DeepAI. Max pooling. https://deepai.org/machine-learning-glossary-and-terms/max-pooling#:~:text=What%20is%20Max%20Pooling%3F,dimensions%20of%20an%20input%20volume. visited on 2024-03-19.

[13] Harpreet Sahota. Intuitive explanation of sgd, adam, and rmsprop. https://www.kaggle.com/code/harpdeci/intuitive-explanation-of-sgd-adam-and-rmsprop#Training-with-Adam, 2023. visited on 2024-03-19.

[14] Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. *Advances in neural information processing systems*, 30, 2017.

[15] Chao Ma Pan Zhou, Jiashi Feng. Towards theoretically understanding why sgdgeneralizes better than adam in deep learning. *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, 2020.

[16] Will Koehrsen. Precision and recall: How to evaluate your classification model, mar 2023. visited on 2024-03-19.

[17] Jason Brownlee. A gentle introduction to the fbeta-measure for machine learning, Jan 2020. visited on 2024-03-19.

[18] Jino Mathew, Rohit Kshirsagar, Dzariff Z Abidin, et al. A comparison of machine learning methods to classify radioactive elements using prompt-gamma-ray neutron activation data. *PREPRINT*, February 2023. Version 1 available at Research Square.