DS340 Project Paper

Authors: Gersian Collaku & Emilie Xiao

Introduction.

The goal of our project was to identify the type of contraband that is hidden within luggages. The dataset we worked with provided us with X-ray images of five contraband types including straight knife, multi-tool knife, folding knife, utility knife, and scissors. We noticed that the different types of knives eventually posed a challenge for our model to detect a specific type. However, this project was pursued to enhance both security, by detecting hard-to-see items, and efficiency, leading to shorter lines at security checkpoints. The main motivation was to improve the accuracy and speed of contraband detection processes. X-ray machines are commonly used at airports to inspect the contents of a luggage. It then creates images of the contents to identify potentially dangerous items that are not visible to the naked eye without physically opening the luggage. In general, accurate detection of contraband is important for maintaining safety and security. If you are interested in taking a look at the dataset, it can be found at this link: https://universe.roboflow.com/airport-security-scanning/airport-security-scans-dataset.

The project focuses on several types of knives, each with unique characteristics. Although all had some of their own features, it was still hard to detect the specific type of knife. In detail, we noticed that a multi-tool knife often has a spiral in its composition. For our diverse range of knives, we note the possibility of great variation in terms of their size, shape, and material. Moreover, objects throughout luggages can overlap or be positioned in ways that make it unclear to detect (different angles). The quality of our X-ray images seems to be relatively clear, despite the orientation of the items. With these challenges of detection, we decided a convolutional neural network (CNN) would work best due to its suited nature for image recognition tasks.

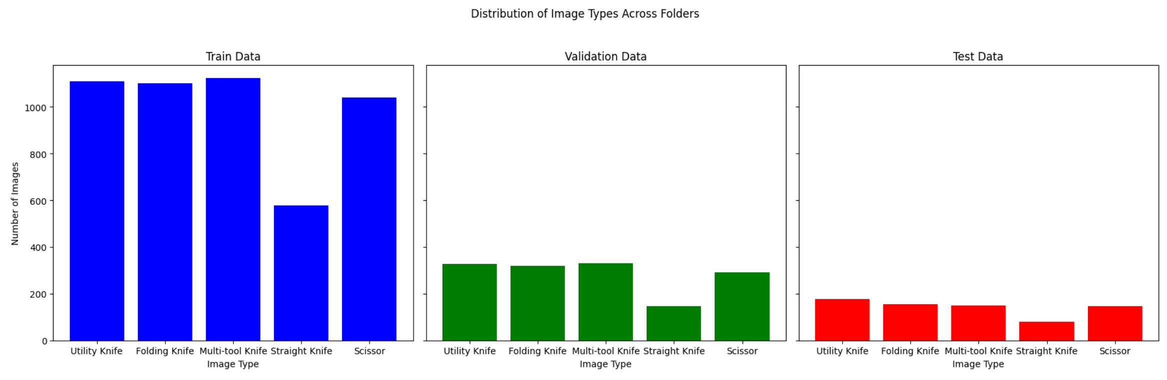


Figure 1. *Distribution of Image Types*

In *Figure 1*, it displays the distribution of our dataset: it has a total of 7,073 X-ray images of luggages which was already split into a training (4,951 images), validation (1,415 images), and test (707 images) set. There was a slight imbalance between these sets. Our dataset has 5 classes which were somewhat unbalanced throughout each set, with straight knives having low counts and utility knives having high counts.

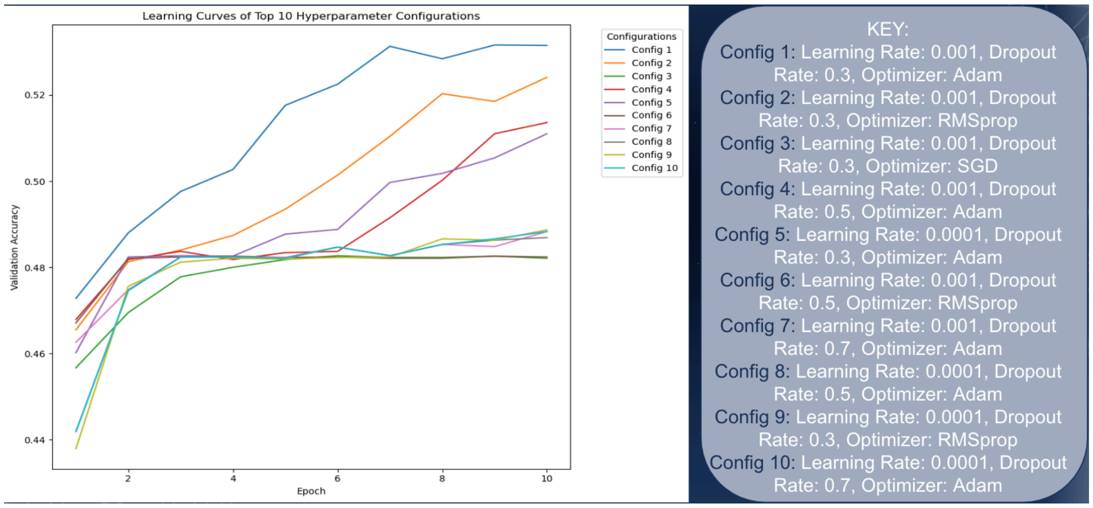
Methodology and Results.

Our training process started with a baseline model in hopes of doing multi-class classification using CNN. We experimented utilizing libraries including TensorFlow and Keras. To begin, we altered the images using different parameters including rescale, zoom_range, width_shift_range, height_shift_range, and fill_mode. Along with that, we generated batches of the image data.

We applied different techniques to our model, including MaxPooling, Conv2D, Flatten, and Dense layers ('relu' and 'softmax' functions). However, although the accuracy was relatively high, we noticed the risk of overfitting on our test set as it produced an accuracy of 23.62%. For our next model, we looked into Dropout Layers, Early Stopping, and Regularization. Moreover, we applied changes to our compiling part of the model: 'adam' optimizer and 'categorical_crossentropy' loss. We ran 10 epochs and believed that it was sufficient enough to learn underlying patterns as we monitored the results. To do some extra experimentation, we added batch normalization in our model and compiled it with the same hyperparameters. We concluded that the base models were not quite good, despite the variations we added.

For a different approach, we implemented pre-built architecture using transfer learning, ResNet50 that had a base model along with GlobalAveragePooling2D, Dense, and Dropout layers. ResNet50, with its deep and complex architecture, uses residual connections and skips connections. This model performed better, providing us with a slightly higher accuracy of 30.55%. In terms of trying out another model, we applied VGG16 and noticed that there was no benefit compared to ResNet50, since it led to a lower accuracy overall. VGG16 has a simple architecture and is known for its uniform design, by using small filters and increasing depth by stacking layers. With various trials and low accuracy, we were interested in finding out if the dataset itself had challenges that led to these outcomes.

To overcome the problem of imbalanced data, we consolidated our images and redistributed them to their respective folders. Furthermore, we oversampled the minority class with data augmentation and undersampled the majority class to further balance our dataset. Keeping in mind that ResNet50 produced the best results thus far, we implemented it after the data cleaning. We ran a nested for loop with these following options: learning_rates = [0.001, 0.0001, 0.00001], dropout_rates = [0.3, 0.5, 0.7], and optimizers = [tf.keras.optimizers.Adam, tf.keras.optimizers.RMSprop, tf.keras.optimizers.SGD]. Overall, the best result out of this was the combination with optimizer as Adam, a dropout rate of 0.3, and a 0.001 learning rate. In terms of loss, we noticed it was decreasing which is a good indication that the model was improving its ability to predict the classes. This allowed our model to converge faster but still prevent overfitting in our neural network. In other words, this moderate learning rate provides a balance that allows adequate learning progress without skipping over important features in the data. Therefore, we noticed that these parameters were tuned for working on our specific task, the dataset, and the rest of the architecture of our model. In *Figure 2*, the configuration that led to the best results from the combination of Adam, a DR of 0.3, and LR of 0.001 are shown. The configurations that present a steadier increase in accuracy and a smoother decline in loss generally used a balanced learning rate which was not too high to cause instability, nor too low to stall the learning process.



KEY:
Config 1: Learning Rate: 0.001, Dropout Rate: 0.3, Optimizer: Adam
Config 2: Learning Rate: 0.001, Dropout Rate: 0.3, Optimizer: RMSprop
Config 3: Learning Rate: 0.001, Dropout Rate: 0.3, Optimizer: SGD
Config 4: Learning Rate: 0.001, Dropout Rate: 0.5, Optimizer: Adam
Config 5: Learning Rate: 0.0001, Dropout Rate: 0.3, Optimizer: Adam
Config 6: Learning Rate: 0.001, Dropout Rate: 0.5, Optimizer: RMSprop
Config 7: Learning Rate: 0.001, Dropout Rate: 0.7, Optimizer: Adam
Config 8: Learning Rate: 0.0001, Dropout Rate: 0.5, Optimizer: Adam
Config 9: Learning Rate: 0.0001, Dropout Rate: 0.3, Optimizer: RMSprop
Config 10: Learning Rate: 0.0001, Dropout Rate: 0.7, Optimizer: Adam

Figure 2. *Learning Curves of Top 10 HP Configurations*

Determining the Location:

As an extension to the previous component of this project, we decided to try and find out where contraband was often detected. To do this, we used a systematic sliding window technique in addition to a heatmap. Employing such techniques, the model was able to examine each section of the image in detail, predicting the likelihood of contraband presence. The outcomes of these predictions are represented in a heatmap, which varies in intensity from yellow to red, indicating regions of increasing suspicion. Upon closer inspection of the heat maps generated, we observe distinct areas highlighted that correspond to potential contraband within the luggage. These predictions are then superimposed onto the original X-ray images, producing an intuitive visual representation of the model's findings. This overlay combines the detail of the original X-rays with the vivid, color-coded heatmap, enabling immediate identification of areas warranting further inspection. The final images present compelling evidence of the model's ability to not only detect the presence of contraband items, but also to pinpoint their precise location. The results were not perfectly accurate as it detects "warm" areas in the image that do not contain contraband. We do believe that with better computing power and imaging capabilities, this model could work with high accuracy and ability to pinpoint the contraband efficiently. The results of this assessment can be found in the notebook and an example is captured in *Figure 3*. Among the different colors, purple represents lower activity areas and yellow represents higher activity areas. We notice that purple appears frequently, suggesting the areas where contraband was not found. Different modifications could be made to this, including but not limited to, using a different colormap, adjusting the step size or window dimensions, improving the model's accuracy, or employing post-processing techniques to reduce noise in the heatmap.

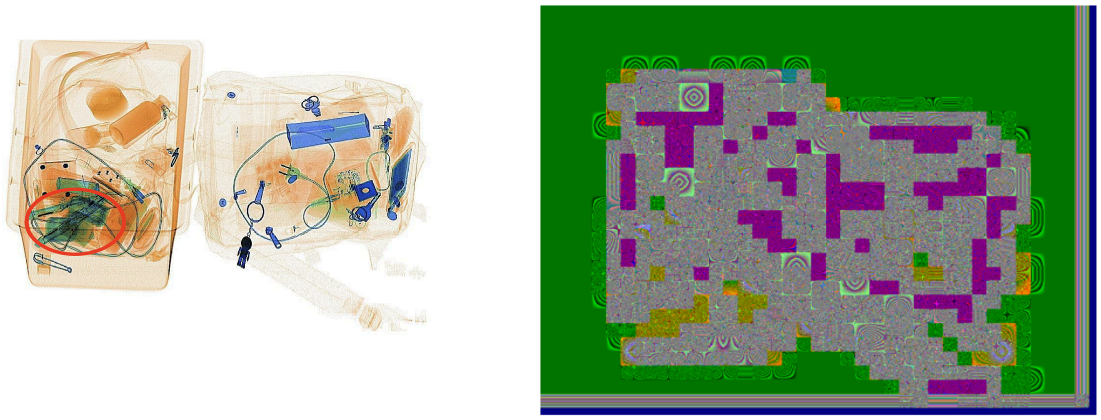


Figure 3. *Original Image vs. Heatmap*

Conclusions.

In conclusion, this project was both challenging and rewarding. We learned how to work with a complex dataset, given the fact that there are many options and approaches to take. We believe we covered a diverse array of methods efficiently. For us, the results were quite interesting. It was surprising to note the challenges that we came across as they were not expected. In general, this experience was a fun challenge to take on and overcome as it gave us a lot of hands-on experience with many different components and aspects of neural networks. The failure in the beginning was what really motivated us to experiment more, as it makes any slight progress feel extremely rewarding. Taking the long time period to run our models into consideration, it came with a lot of constraints alongside using Google CoLab to share models with each other. The main obstacle was placing all our results into a polished notebook since we had to rerun everything and troubleshoot certain places if something went wrong. Along the way, many models ranged from ten minutes to a few hours depending on various factors. In the future, we hope that one day there could be technology developed for this as it could be an extremely effective tool. With that being said, we hope the results of this project were interesting to hear about.

Works Cited:

Airport Security Data, https://universe.roboflow.com/airport-security-scanning/airport-security-scans-dataset. Accessed 29 April 2024.

"[CV] 9. Object detection with Sliding Window and Feature Extraction(HoG)." *Medium*, 2 December 2020, https://medium.com/jun94-devpblog/cv-9-object-detection-with-sliding-window-and-feature-extraction-hog-cf1820c86b46. Accessed 29 April 2024.

Krispective. "Multi-Class Image Classification using transfer learning with deep convolutional neural networks." *Medium*, 26 January 2021, https://medium.com/analytics-vidhya/multi-class-image-classification-using-transfer-learning-with-deep-convolutional-neural-networks-eab051cde3fb. Accessed 29 April 2024.

Mishra, Prateek. "MultiClass Image Classification using keras." *Kaggle*, 2019, https://www.kaggle.com/code/prateek0x/multiclass-image-classification-using-keras. Accessed 29 April 2024.

Mostafid, Tannaz. "Overview of VGG16, ResNet50, Xception and MobileNet Neural Networks." *Medium*, 12 Dec. 2023, medium.com/@t.mostafid/overview-of-vgg16-xception-mobilenet-and-resnet50-neural-networks-c678e0c0ee85. Accessed 29 Apr. 2024.

Rosebrock, Adrian. "Sliding Windows for Object Detection with Python and OpenCV." *PyImageSearch*, 23 March 2015, https://pyimagesearch.com/2015/03/23/sliding-windows-for-object-detection-with-python-and-opencv/. Accessed 29 April 2024.