Threat Classification in X-Ray Security Scans

Exam Paper
Data Mining, Machine Learning, and Deep Learning (CDSCO1004E)

Cosmin-Mirel Budeanu (111677)
Torben Damm (130898)
Silvia Elisabeth Dietze (150417)
Emilia Konopko (149895)

Number of Characters: 33,909

Copenhagen Business School
M.Sc. Business Administration and Data Science
23^{rd} May 2022

Abstract

In a global and connected world, many precautions, such as X-ray scanners, are used to enable safe commuting and travel. This paper focuses on testing the robustness of machine learning and deep learning methods for image classification of prohibited items in X-ray security scans. One support vector machine and three different convolutional neural networks were trained on a merged data set of labelled security scan images with and without augmentation. The classification included binary (threat and no-threat) as well as multi-class classification. All models were found to be superior to the random classifier, with a CNN based on a simplified VGG-16 architecture achieving the highest performance. However, limited data accessibility, data set size, general generalisation issues, and class imbalance impose major limitations to the aforementioned findings.

Keywords: Image classification; threat object detection; X-ray; security scan; CNN.

1. Introduction

In the modern world, airplanes are the fastest and safest way to travel, even though being high-value targets for terrorists for several decades (Abadie and Gardeazabal, 2008).

With increasing flight demand, airports are becoming more and more crowded and the risk of attacks is growing. To minimise this danger, passengers' luggage is scanned with X-rays detecting a variety of threat objects. And although these scanners gained popularity since the terrorist attack in September 2001, they are only used as auxiliary means and the final decision is still made by a human being (Mery et al., 2013). During peak times, when airports are very crowded, the operators only have a few seconds to determine if there are prohibited items in the luggage. Their job is very demanding as detecting threat items can be further complicated due to the overlapping of many objects. "The object of interest may be mixed with arbitrary and meaningless items, and thus can be ignored even by human screeners" (Miao et al., 2019).

Since this work requires tremendous concentration and each worker has to scan multiple bags with similar images, there is a serious chance of human error and the current detection of prohibited objects is not at a satisfactory level yet (Wei et al., 2020). It is believed that machine learning algorithms with automated object detection can significantly improve performance and reduce the detection and response time of human scanners (Chavaillaz et al., 2018). But despite the increasing interest in X-ray screening, research on automated computer-aided screening methods is still underrepresented. Reasons for this lie largely in the lack of suitable data as well as the need for advanced learning algorithms (Hassan et al., 2021).

To address the research gap, this paper aims to test the ability of different machine learning algorithms in identify-

ing and classifying prohibited items in security imagery. Four models will be trained on three large-scale X-ray data sets and evaluated on their performance accordingly.

First, the conceptual framework and methodology are presented, followed by the data pre-processing and model design. Finally, the findings as well as challenges and limitations are discussed.

2. Related Work and Literature Review

Previous work has been mainly conducted on conventional image analysis and machine learning methods, including classification, detection, and segmentation tasks. In particular, the bag of visual words (BoVW) approach was dominating the field of research: Feature extraction was performed via detectors/descriptors, then clustering via k-means and lastly RF or SVM (Baştan et al., 2011; Breiman, 2001; Hartigan and Wong, 1979). Other machine learning techniques included structure estimation and segmentation together with a general tracking algorithm or focus on k-NN based sparse representation (Mery et al., 2013; Mery et al., 2016).

Object detection algorithms are relatively sparse in the literature due to their challenge of simultaneously predicting the bounding box coordinates and class labels. Yet, contrary to classification, the object detection approach also utilises multiple-view imagery which helps human operators and machines to improve their detection performance (Franzel et al., 2012).

In recent times, however, deep-learning-based algorithms have received wider attention in X-ray security imaging, particularly since convolutional neural networks (CNN) significantly outperform the traditional machine learning methods (Akcay and Breckon, 2022).

The research by Akçay et al. (2016) is one of the first studies that uses CNNs for X-ray security imagery. The authors applied transfer learning to evaluate to what extent it helps classifying X-ray objects in the case of limited data set availability. The results reveal that CNN significantly outperforms the BoVW approach, trained with SVM or RF.

What is more, Miao et al. (2019) explores a class-balanced hierarchical refinement (CHR) approach to deal with the difficulties of class imbalance and clutter in their presented *SixRay* data set. Their findings indicate that CHR is particularly advantageous in the scenario of fewer positive training samples and therefore has high potential to be applied in real-world security inspection.

Introducing their *OpixRay* data set, Wei et al. (2020) presents a similar approach based on a plug and play module that leverages edge and material information to position objects through the attention mechanism.

Further work by Wang et al. (2021) covers an additional large-scale data set (*PidRay*) to focus on prohibited item detection in real-world scenarios. They designed a selective dense attention network to learn the discriminative features and thus boost the model's performance. The results prove to be especially useful for detecting deliberately hidden items.

3. Conceptual Framework

This report evaluates machine and deep learning models and techniques regarding their performance to classify threat objects in X-ray security scans. At first, the problem is treated as a binary classification task, but later extended to a multi-class classification task.

The data pre-processing included combining labelled X-ray images of six classes from three data sets into one directory. Next, the images were cropped, filtered and cleaned as further described in section 6. The data augmentation was done through *keras* and integrated into the data loading process. The whole workflow is pictured in Figure 1.

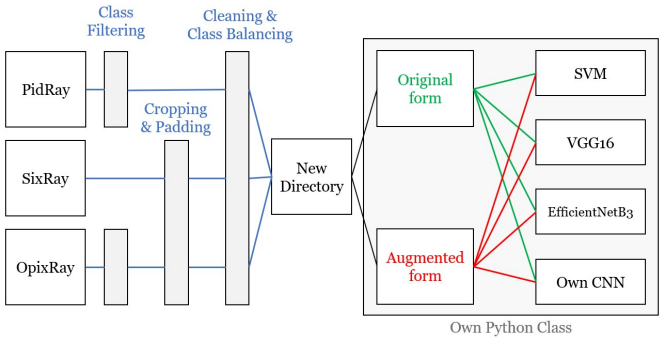


Figure 1: Process Flow

A total of four models (one SVM and three CNNs) were trained once on original and augmented data and then evaluated and compared accordingly. The four models are:

- Support Vector Machine (*SVM*)
- CNN based on simplified *VGG-16* architecture
- CNN transfer learning model based on *EfficientNetV2S*
- Self-constructed *own CNN*

4. Methodology

In the following, the chosen machine learning and deep learning models will be described and and justified in greater detail.

CNNs emerged from the examination of the brain's visual cortex and have been used in image recognition since the 1980s, making it the most common deep learning model to day. It analyzes and visualizes the pixel data within an image in an ideal manner with minimized processing and is equivalent to the structure of a multilayer perceptron with varying layers (Jena et al., 2022). Its benefit lies in the ability to develop an internal representation of a two-dimensional image, allowing the model to learn the position and scale in different data structures.

One of the reasons why computer vision has been revolutionised by CNNs is the discovery that feature representations learned in a pre-training task can carry useful information to target tasks (He et al., 2019). For this, transfer learning is used. The goal of transfer learning is to provide a framework for leveraging previously gained knowledge to cope with new but similar problems in a faster and more effective way (Lu et al., 2015). Thanks to this, even if a model contains a small training set, it can still achieve state-of-the-art results (Girshick et al., 2014). One example of pre-trained models is the *ImageNet*. It is a large-scale, accurate and diverse data set intended for visual recognition applications, among others image classification. It contains almost 15 million images belonging to more than 20,000 classes (Deng et al., 2009).

SVMs are powerful and versatile machine learning models capable of performing classification, regression as well as outlier detection (Géron, 2019). SVM follows the maximum margin concept and shows improved generalisation performance as it implements the structural risk minimization (SRM) principle (Archibald and Fann, 2007). It is particularly beneficial for the classification of complex small- or medium-sized data sets.

4.1. Perfomance Metrices

This report decided to display accuracy, precision, recall, and the f1-score to assess model performance in different ways. However, the recall metric is considered most valuable for the given scenario. Classifying false positives (predicting a threat even if there is none) imposes little cost of another baggage check or requiring a second opinion on the scan images. On the other hand, classifying false negatives (predicting no threat while there is one) is found to impose high costs: risking the use of threat objects like guns or knives on planes or trains and ultimately putting human lives in danger. The recall metric reflects on that issue in the best way being composed of the ratio of positives being identified by the models:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{1}$$

Hence, all models were optimised towards maximising recall during training and validation and were finally assessed in that way.

For the final assessment, a random classifier was used for benchmarking rendering an accuracy of 16.7% (50%) in the presence of six (two) classes. Notably, this scenario is not considered to be realistic since most baggage scans would fall into the "no threat"-category rendering the real life test data heavily imbalanced.

5. Data Set Description and Analysis

In their survey, Akcay and Breckon (2022) describe a number of data sets containing X-ray images from airport or subway security. However, they also highlight several limitations including that various data sets, especially those of high quality maintained by government or private security, are not publicly available. After looking into the availability and quality of the images of the remaining sets and further online search, this report found three data sets to be of potential use: *PidRay*, *SixRay* and *OpixRay*, which are further described in Table 1. All data sets contain images of real-life scans and were downloaded in colored form. Notably, the individual data sets hold a different number of classes with variation in the number of pictures for each class. Hence, the preprocessing of all data sets aimed at not only selecting and rendering images of high quality but also creating a balanced set of images for the six chosen classes[1], containing only labelled images with one object or no threat at all. This imposed some challenges that are further described in section 6.

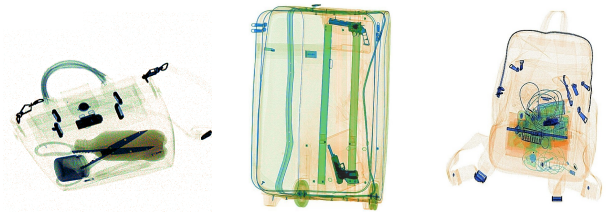
The data sets were downloaded from various GitHub repositories and first evaluated by taking a look at a number of pictures and the various forms of annotation[2] to check

[1] The classes are No Threat, Gun, Knife, Scissors, Pliers and Welds.
[2] The three data sets used annotation with .txt, .xml or .json-files.

	Total Images	Threat Images	Classes
PidRay	47,677	47,677	12
SixRay	1,059,231	8,828	6
OpixRay	8,885	8,885	5

Table 1: Data Set Description

for data quality and structure. Exemplary pictures from the three data sets can be seen in Figure 2. This exploratory data analysis also left the impression of some pictures being hard to classify, especially those containing threat objects pictured from the side or above.



(a) *PidRay* (b) *SixRay* (c) *OpixRay*

Figure 2: Example Images from Data Sets

6. Data Pre-Processing

The following section illustrates the process of selecting and pre-processing the images within each data set and accumulating them into one new directory of images.

6.1. PidRay

The *PidRay* data set[3] is the largest set of threat images used for this report, containing a total of 47,677 threat images. However, only 18,406 images were actually utilized since the original data set held a total of 12 different classes. Unfortunately, the annotation for the position of the threat objects was not intuitive and the documentation did not render sufficient information to crop the images adequately. This report consequently decided to use the pictures in their original form without cropping. Therefore, all pictures containing more than one object where omitted from the data set to allow for proper labelling and avoid mislabelling for multi-object images. This further reduced the size of the data set from 18,406 to 12,065 images.

[3] https://github.com/bywang2018/security-dataset

6.2. SixRay

The *SixRay* data set[4] includes a total of 1,059,231 images with as much as 8,828 containing threat objects. Since the documentation and annotation were well-structured, the images were cropped based on the information in the xml-annotation. By doing so, even multi-object images could be used to generate multiple single object images. This included calculating the location and boundaries of the threat object and then adding some padding pixels from the original image to fit them to the form of the *PidRay* images that were not cropped. Since only 4,152 of the images contained only one object, the padding increased the chances of another object being visible and some images were found to show overlapping objects.

To include a non-threat class accounting for the real-life usage of pure threat detection, a total of 5000 randomly selected non-threat images were added from the *SixRay* data set.

6.3. OpixRay

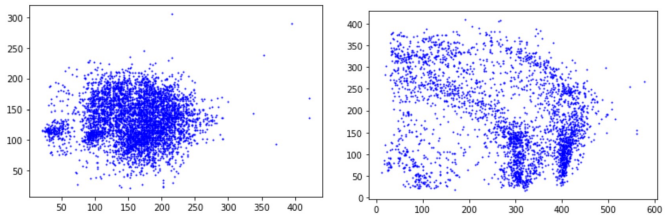
Since the class imbalance indicated a comparably small number of scissors and knives and many of the latter were found to be of low quality, the *OpixRay* data set[5] was added. Carrying images of various types of knives and scissors, this data set was used to overcome class imbalance. To do so the images showing *straight knives*, *utility knives* and *scissors* were cropped according to the annotation and padding pixels from the original image were added.

6.4. Data Filtering, Combination and Class Balancing

After creating a directory accumulating the pre-processed images from the three data sets, some manual and semi-manual cleaning was done to get rid of "bad" pictures, that is pictures with overlapping objects, patently misclassified objects or pictures of inadequate size or scale. Figure 3 shows the distribution of image size within the *SixRay* data set for guns and knives. It can be seen that the ratio for guns was more balanced than for knives. Consequently, a number of knives had to be excluded due to their low size and weak scaling.

Additionally, images mostly from the pliers and wrenches category were omitted to lower class imbalance on the training data to a maximum of 5% .

[4] https://github.com/MeioJane/SIXray
[5] https://github.com/OPIXray-author/OPIXray



(a) Guns (b) Knives

Figure 3: Size Distribution of *SixRay* images

The labelling was done by automatically sorting the pictures into different folders named after the respective classes which is later utilised by the *keras* functions when loading the pictures. The final number of images being split into training and validation is displayed in Table 2. The test set originated from a combination of the *SixRay* and *PidRay* data sets.

Initially, this report decided to process images in grayscale as indicated by Bui et al. (2016) and Sachin et al. (2017). However, the model utilising transfer learning required RGB colored three-channel images. Hence, all models were trained using colored images to ensure comparability. This comes at the cost of computational speed since three channel color images require more time for processing than one channel grey-scale images (Bui et al., 2016, p. 323).

6.5. Data Augmentation

The final models were each trained first on the original data and second on data including augmentation to assess in how far data augmentation enhances model performance. The augmentation enables the models to generalise in a better way and account for different positions of the threat objects inside of the luggage. Accordingly, augmentation with flipping and rotating was performed. Other forms of augmentation like changing the brightness and adding noise were left out since the scans are conducted in a closed environment of the X-ray machine.

7. Modeling, Methods and Tools

Describing the four models used for this report, this sections highlights key characteristics of each model and gives reasoning for the choice.

7.1. CNN Based on Simplified VGG-16 Architecture

For the purpose of this study, we decided to build one model based on the VGG-16 network which was presented for the first time by K. Simonyan and A. Zisserman and submitted to the ImageNet Challenge in 2014 (Theckedath and Sedamkar, 2020).

Compared to the competition's top-performing models, VGG-16 proposed the use of a very small 3x3 receptive field throughout the entire network with the stride of one pixel which enables faster network convergence (Simonyan and Zisserman, 2014). And despite not winning, the idea paved the way for subsequent innovations in the field of Computer Vision.

The VGG-16 structure was adopted with a smaller number of neurons. Further, three convolutional layers and one max pooling where omitted since the image size was too small to run another max pooling.

The first CNN is a sequential model that initially re-scales input data from the range [0, 255] to the range [-1, 1] to normalize the data. Moreover, after every two convolutional layers, a max pooling layer is added, and after the last one, a Global Average Pooling is implemented, which generates one feature map for each category of the classification task. Instead of adding fully connected layers on the top of the feature maps, it takes the average of each feature map. Global Average Pooling replaces the flattening layers.

What is more, a dropout method is used to prevent model over-fitting, and a Rectified Linear Unit (ReLU) is added to each layer so that negative values are not passed on to the next layer. In addition, Softmax activation is implemented so that the output values range from 0 to 1, indicating the class to which the images belong. In the last dense layer, six unit layers are used as six classes are predicted. Finally, an Adam optimizer with a learning rate of 0.0002 and a categorical cross-entropy loss function are utilised. The choice of the learning rate was made based on randomised optimisation.

7.2. Transfer Learning Model Based On EfficientNetV2S

The second model tested is a CNN utilising transfer learning based on EfficientNetV2S with *ImageNet* weights[6].

EfficientNet is considered as a family of CNNs with the advantage of a higher accuracy and improved efficiency by reducing the number of parameters (Tan and Le, 2019). According to Tan and Le (2021), EfficientNetV2 models train much faster and are around 6.8 times smaller than state-of-the-art models, determining the choice of this model.

[6]Keras, https://keras.io/api/applications/

	Sum	No Threat	Gun	Knife	Scissors	Pliers	Wrench
PidRay		0	1960	1803	2763	2622	2917
SixRay		5000	3037	1846	746	3641	2354
OpixRay		0	0	1996	1863	0	0
Deleted images		-105	0	-761	-607	-1183	-249
Train Set	**23,715**	3,916	3,998	3,907	3,812	4,064	4,018
Validation Set	**5,928**	979	999	977	953	1016	1004
Test Set	**5,043**	796	751	927	983	872	714

Table 2: Data Set Composition

As the model was only utilized for feature extraction, a fully-connected layer was not included. Instead, the shape of the input data was specified to (128, 128, 3). Moreover, a Global Average Pooling layer was used to reduce the total number of parameters and thus minimise over-fitting. Finally, a Softmax output layer was implemented, which outputs the probability distribution for each of the six classes and classifies the image according to the most likely class. The optimisation was performed using the Adam optimiser with a learning rate of 0.001, and categorical cross entropy as the loss function.

During training it was observed that the model got stuck in a local minimum and the loss did not further decrease. Hence, the learning rate was increased which solving the issue of convergence to a local minimum.

7.3. Support Vector Machine

The third model is the Support Vector Machine, whose main idea is to find the hyperplane with maximum margins that best divides the data set into classes and has the greatest separation to the nearest sample (Baly and Hajj, 2012). Like the first model, it includes convolutional layers with ReLu activation and max pooling layers with a pool size of (2.2), a stride equal to 2 and a dropout parameter set to 0.2. The main difference from the other models is that since we are dealing with a multi-class classification problem, the squared hinge loss is used and the last layer is regularized with l2.

7.4. Self-Constructed CNN

The final model is a self-constructed CNN network which was created by trying different hyperparameters and observing their effect on the outcome. Throughout the process, it was checked whether the model improves or deteriorates after each layer addition or change in the number of neurons, image size, batch size or learning rate. Another step was to decide between Flatten or Global Average Pooling for transforming from 2D to 1D. In practice, the model showed a slower convergence with Global Average Pooling. Flattening rendered over-fitting on the training set indicating that the model did not have enough time to learn complex patterns.

Another aspect of interest was the influence of dropout. With dropout, the model did not over-fit as fast as without it.

The final version settled on a sequential model with five Conv2D and five max pooling layers, followed by a Global Average Pooling and a dropout layer. In each layer the activation is set to ReLu, except for the Output layer where the activation is set to Softmax.

The biggest difference between this model and the first one is that there is one max pooling layer after each convolution layer rather than one max pooling layer after every two convolution layers significantly reducing model complexity.

7.5. General Explanation for Choice of Hyperparameters

The image shape was set to (128,128,3) since this shape turned out be optimal for training the data set. It was further decided on including all three channels (color images) since using training layers from *EfficientNetV2S* requires having three channels. For the loss function, either of the best performing ones could have been applied: *Adam* or *RMSprop*. But based on the recommendation given by Ruder (2016), *Adam* was selected since it is superior to the *RMSprop* towards the end of the optimization. The batch-size was chosen to be 64 as it offers a sufficient overview over the different classes. The learning rate was set to be 0.0002 for most of the models since by trying different values it was observed that values of 0.0003 and over would converge too fast and values of 0.0001 or smaller would get stuck in a local minimum.

7.6. Model Complexity

Assessing model complexity, the model architectures and summaries including model structure, configuration, and parameterisation were evaluated (see Table 3).

	Number of Parameters	Layers
SVM	1,059,662	8
VGG-16	23,996,582	22
EfficientNet2VS	22,339,046	4
Own CNN	476,550	16

Table 3: Model Complexity (see Figures 10 to 17 in the Appendix)

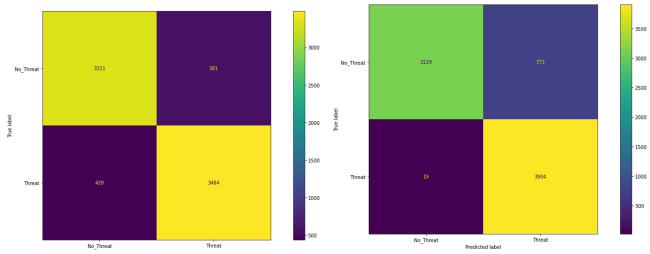
Both *VGG-16* and *EfficientNetV2S* are considered highly complex models especially since multiple convolutional layers are put in a row. The input layer for *EfficientNetV2S* applies transfer learning based on a pre-trained model with 20.3 million parameters. Further, the structure of *VGG-16* had to be simplified by omitting one set of layers and reducing the number of neurons due to the small shape of the images. Still, the model is the most complex model with as many as 24 million parameters.

This rendered the idea to create one less complex model to see in how far complexity adds to model performance. The personally created *own CNN* utilises a five-fold combination of a single convolutional layer and a max pooling layer, thus reducing model complexity and the number of parameters to less than 0.5 million.

8. Results

The performance metrices of all models are displayed for binary classification in Table 4 and for multi-class classification in Table 5, respectively.

For binary classification, both *VGG-16* and the *own CNN* performed well with recall up to 0.90 and accuracy up to 0.89 (see Figure 4). Notably, the *VGG-16* showed a more balanced classification while the *own CNN* over-classified non-threat as threat images.



(a) VGG-16 (b) Own CNN

Figure 4: Confusion Matrix for Binary Classification

The multi-class classification did not perform accordingly, with *VGG-16* being the best performing model but only reaching recall and accuracy of 0.47. The confusion matrix for multi-class classification of the *VGG-16* model is displayed in Figure 5 showing significant miss-classification especially for scissors.

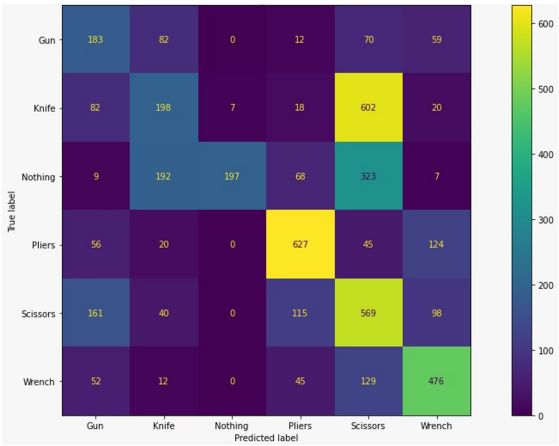


Figure 5: Confusion Matrix of *VGG-16* for Multi-Class Classification

9. Discussion

9.1. Performance Assessment

All models significantly outperformed the random classifier both in the binary as well as in the multi-class classification task (see Tables 4 and 5).

Besides *SVM* converging to a single class classification, all models performed well on the binary classification task. CNN significantly outperformed SVM, underlining its use for image classification tasks (Hussain et al., 2018; Wang et al., 2016). Nonetheless, X-ray scans are crucial for security thus increasing the need for even higher performance considering the high cost of false negatives (see Section 4.1).

Unfortunately, the multi-class classification did not render satisfying results. The *VGG-16* confusion matrix (see Figure 5) indicates that the classes for hard to detect objects (knives and scissors) seem to have a significantly lower performance and drive miss-classification. Furthermore, non-threat objects were mainly classified as various threat objects, while little to no threat objects were classified as non-threat.

Surprisingly, data augmentation did not increase performance of the models and even lowered certain performance metrices (see Table 5). This contradicts the assumption of data augmentation enhancing the performance regarding size of the data set and the variation in the orientation of

	Accuracy	Precision	Recall	F1-Score
Random Classifier	0.50			
SVM	0.50	0.25	0.50	0.33
VGG-16	**0.89**	0.87	0.86	0.85
EfficientNetV2S	0.69	0.77	0.69	0.66
Own CNN	0.88	**0.91**	**0.90**	**0.90**

Table 4: Binary Classification: Performance Metrices (see Figures 6 and 7 in the Appendix)

	Original Data				Augmented Data			
	Accuarcy	Precision	Recall	F1-Score	Accuarcy	Precision	Recall	F1-Score
SVM	0.30	0.42	0.30	0.28	0.31	0.44	0.30	0.29
VGG-16	**0.47**	**0.60**	**0.47**	**0.48**	0.46	0.57	0.44	0.45
EfficientNetB3	0.42	0.44	0.42	0.41	0.42	0.41	0.43	0.42
Own CNN	0.43	0.51	0.43	0.41	0.40	0.49	0.42	0.39

Table 5: Multi-Class Classification: Performance Metrices (see Figures 8 and 9 in the Appendix)

the object inside of the luggage. After assessing these unexpected results, further research indicated that the *keras* function does not truly augment the data set by adding variations of the original images but simply replaces them instead. Therefore, the initial augmentation performed can be understood as using a transformed training set rather than an enlarged one. Accordingly, it seems reasonable that the models performed similarly.

9.2. Generalisation and Real-Life Applicability

The binary classification rendered promising results regarding threat object detection thanks to comparably high recall metrices (see Table 4). Further testing with single images revealed that the models mostly classified images with large and dense objects as threat objects. Unfortunately, smaller objects especially knives and scissors where not classified as threats. This report finds that more data of smaller threat objects is needed to enhance the generalisability of the model. Especially, objects with common, harder to distinguish outlines like knives tend to be not detected and might be mistaken as non-threat objects like pens or toothbrushes.

Additionally, the models tend to overestimate the number of threat objects both in the binary and the multi-class classification. This also limits real-life usage since the models indicated far too many luggage pieces to be further checked. This seems especially problematic in the light of class imbalance: threat objects only represent a small share of all real-life X-ray security scans. While this report decided to stick with the recall metric as the most important performance indicator (see section 4.1), this issue should be addressed when further optimising the models towards generalisation.

All models are trained on known forms of threat objects present in the data sets. Assessing real-life applicability, this imposes the challenge of a concept drift. The structure, outline, and form of threat objects could evolve over time or be masked and the models have to be adjusted accordingly.

Various forms of theses changes were already observed in some images: A number of knives where wrapped with cables to deter the outline of the scan in the X-ray image and make the whole scan more cluttered. Notably, the data sets included a number of classes of threat objects that were not included in the training data thus limiting threat object detection to the classes trained on.

9.3. Model Complexity

Besides performance, model complexity plays an important role when assessing machine and deep learning model architectures. Not only does the complexity influence the computational time and the number of parameters but allows to assess the capabilities of the model (Hu et al., 2020; Raghu et al., 2017). Herein, one might look especially at "expressive capacity" and "effective model complexity" (Hu et al., 2021, p. 2588). The first describes the capability of the model and to what extent it is able to solve complex problems. The latter deals with the complexity of the models functions and its configuration. This report focused on effective model complexity (see Section 7.6).

It was found that model complexity does not necessarily indicate better performance. The individually created model being the least complex one performed almost as well as the

compound models at the same time requiring only a fraction of computational power. Moreover, for the binary classification it was also the overall best performing model of all four choices. Notably, this is also influenced by hyperparameter optimisation that was extensively performed on the *own CNN* thanks to reduced complexity and faster convergence.

9.4. Challenges and Limitations

The performance of the models was limited since a lot of large and high quality data sets are kept private and were not found online. This is also indicated by the survey of Akcay and Breckon (2022, p. 6) showing worse performance of models trained on public data sets. Therefore, this report finds that object classification or detection could be conducted in a better way collaborating for example with a business partner like an airport or security firm which would be able to provide further data.

Additionally, the number of images seems too small for some of the complex model structures since the independently created model with a simpler structure performed comparably well. Data augmentation could play an important role to increase the number of images and address some of the generalisation issues discussed in section 9.2. Unfortunately, the data augmentation could not be performed for all models on a reasonable number of epochs with a truly augmented data set in a timely manner due to computational limitations.

10. Conclusion and Future Work

In this paper, we investigate prohibited item classification in complex X-ray scanned images with a focus on comparing state-of-the-art machine learning and deep learning techniques. The current findings indicate potential for binary classification but limited applicability in real-world environments. Multi-class classification, however, cannot yet show satisfactory results for the use of security purposes. In conclusion, this report already tackled a number of challenges dealing with complex X-ray images but aims at continuing the work for this projects in two ways.

Firstly, our team decided to create the images in an truly augmented form (see section 6.5), thus quadrupling the size of the data set. Running all models on a reasonable number of epochs on this enlarged data set will take further time but is expected to perform better and enhance generalisation. Assessing the benefits of true augmentation is therefore an interesting topic for future work and research.

Secondly, the annotation of the *SixRay* and *OpixRay* data sets can be utilised to perform object detection rather than image classification. The idea is to treat the object boundaries (four numbers indicating the edges of the object box) as the labels for the specific objects. By doing so, trained models try to predict these boundaries for the boxes of different threat objects. This task is not only considered to better reflect on a real-life use case and thus of high interest to security companies. Object detection might also overcome the problem of smaller objects to be identified but faces challenges with overlapping objects. Object detection is also a topic of interest for following research trying to detect non-threat objects like laptops and aiming at analysing all objects inside of a single piece of luggage (Mery et al., 2013; Wang et al., 2021).

References

Abadie, A. and Gardeazabal, J. (2008). Terrorism and the world economy. *European economic review*, 52(1):1–27.

Akcay, S. and Breckon, T. (2022). Towards automatic threat detection: A survey of advances of deep learning within x-ray security imaging. *Pattern Recognition*, 122:108245.

Akçay, S., Kundegorski, M. E., Devereux, M. G., and Breckon, T. (2016). Transfer learning using convolutional neural networks for object classification within x-ray baggage security imagery. *2016 IEEE International Conference on Image Processing (ICIP)*, pages 1057–1061.

Archibald, R. and Fann, G. (2007). Feature selection and classification of hyperspectral images with support vector machines. *IEEE Geoscience and remote sensing letters*, 4(4):674–677.

Baly, R. and Hajj, H. (2012). Wafer classification using support vector machines. *IEEE Transactions on semiconductor manufacturing*, 25(3):373–383.

Baştan, M., Yousefi, M. R., and Breuel, T. M. (2011). Visual words on baggage x-ray images. In *International conference on computer analysis of images and patterns*, pages 360–368. Springer.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Bui, H. M., Lech, M., Cheng, E., Neville, K., and Burnett, I. S. (2016). Using grayscale images for object recognition with convolutional-recursive neural network. In *2016 IEEE Sixth International Conference on Communications and Electronics (ICCE)*, pages 321–325. IEEE.

Chavaillaz, A., Schwaninger, A., Michel, S., and Sauer, J. (2018). Automation in visual inspection tasks: X-ray luggage screening supported by a system of direct, indirect or adaptable cueing with low and high system reliability. *Ergonomics*, 61(10):1395–1408.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Franzel, T., Schmidt, U., and Roth, S. (2012). Object detection in multi-view x-ray images. In *Joint DAGM (German Association for Pattern Recognition) and OAGM Symposium*, pages 144–154. Springer.

Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. " O'Reilly Media, Inc.".

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

Hartigan, J. A. and Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108.

Hassan, T., Akçay, S., Bennamoun, M., Khan, S., and Werghi, N. (2021). Unsupervised anomaly instance segmentation for baggage threat recognition. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–12.

He, K., Girshick, R., and Dollár, P. (2019). Rethinking imagenet pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4918–4927.

Hu, X., Chu, L., Pei, J., Liu, W., and Bian, J. (2021). Model complexity of deep learning: A survey. *Knowledge and Information Systems*, 63(10):2585–2619.

Hu, X., Liu, W., Bian, J., and Pei, J. (2020). Measuring model complexity of neural networks with curve activation functions. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1521–1531.

Hussain, M., Bird, J. J., and Faria, D. R. (2018). A study on cnn transfer learning for image classification. In *UK Workshop on computational Intelligence*, pages 191–202. Springer.

Jena, B., Nayak, G. K., and Saxena, S. (2022). Convolutional neural network and its pretrained models for image classification and object detection: A survey. *Concurrency and Computation: Practice and Experience*, 34(6):e6767.

Lu, J., Behbood, V., Hao, P., Zuo, H., Xue, S., and Zhang, G. (2015). Transfer learning using computational intelligence: A survey. *Knowledge-Based Systems*, 80:14–23.

Mery, D., Mondragon, G., Riffo, V., and Zuccar, I. (2013). Detection of regular objects in baggage using multiple x-ray views. *Insight-Non-Destructive Testing and Condition Monitoring*, 55(1):16–20.

Mery, D., Svec, E., Arias, M., Riffo, V., Saavedra, J. M., and Banerjee, S. (2016). Modern computer vision techniques for x-ray testing in baggage inspection. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(4):682–692.

Miao, C., Xie, L., Wan, F., Su, C., Liu, H., Jiao, J., and Ye, Q. (2019). Sixray: A large-scale security inspection x-ray benchmark for prohibited item discovery in overlapping images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2119–2128.

Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., and Sohl-Dickstein, J. (2017). On the expressive power of deep neural networks. In *international conference on machine learning*, pages 2847–2854. PMLR.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Sachin, R., Sowmya, V., Govind, D., and Soman, K. (2017). Dependency of various color and intensity planes on cnn based image classification. In *International symposium on signal processing and intelligent recognition systems*, pages 167–177. Springer.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Tan, M. and Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR.

Tan, M. and Le, Q. (2021). Efficientnetv2: Smaller models and faster training. In *International Conference on Machine Learning*, pages 10096–10106. PMLR.

Theckedath, D. and Sedamkar, R. (2020). Detecting affect states using vgg16, resnet50 and se-resnet50 networks. *SN Computer Science*, 1(2):1–7.

Wang, B., Zhang, L., Wen, L., Liu, X., and Wu, Y. (2021). Towards real-world prohibited item detection: A large-scale x-ray benchmark. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5412–5421.

Wang, J., Yang, Y., Mao, J., Huang, Z., Huang, C., and Xu, W. (2016). Cnn-rnn: A unified framework for multi-label image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2285–2294.

Wei, Y., Tao, R., Wu, Z., Ma, Y., Zhang, L., and Liu, X. (2020). Occluded prohibited items detection: An x-ray security inspection benchmark and de-occlusion attention module. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 138–146.

Appendix

SVM

	precision	recall	f1-score
No_Threat	0.50	1.00	0.67
Threat	0.00	0.00	0.00
accuracy			0.50
macro avg	0.25	0.50	0.33
weighted avg	0.25	0.50	0.33

VGG16

	precision	recall	f1-score
No_Threat	0.88	0.85	0.87
Threat	0.86	0.89	0.87
accuracy			0.87
macro avg	0.87	0.87	0.87
weighted avg	0.87	0.87	0.87

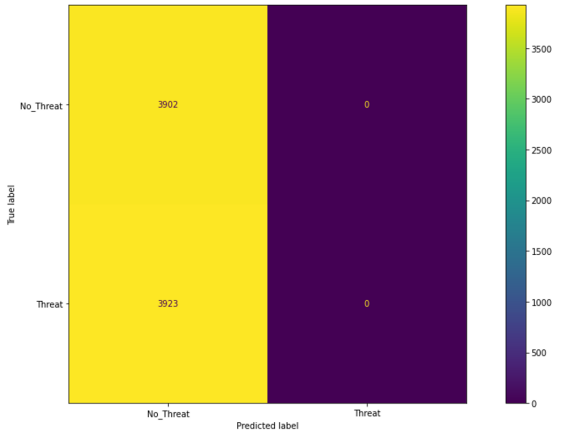
Own_Model

	precision	recall	f1-score
No_Threat	0.99	0.80	0.89
Threat	0.83	1.00	0.91
accuracy			0.90
macro avg	0.91	0.90	0.90
weighted avg	0.91	0.90	0.90

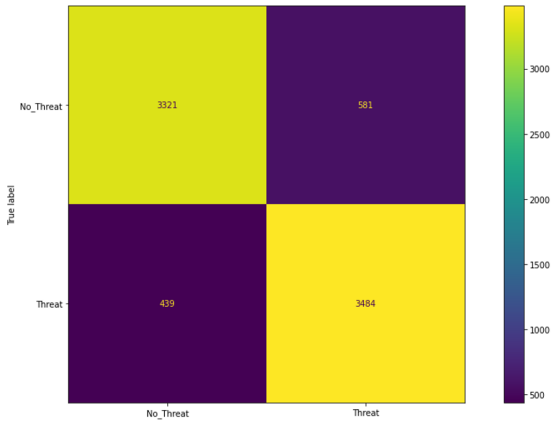
CNN-TL

	precision	recall	f1-score
No_Threat	0.62	0.96	0.76
Threat	0.92	0.41	0.57
accuracy			0.69
macro avg	0.77	0.69	0.66
weighted avg	0.77	0.69	0.66

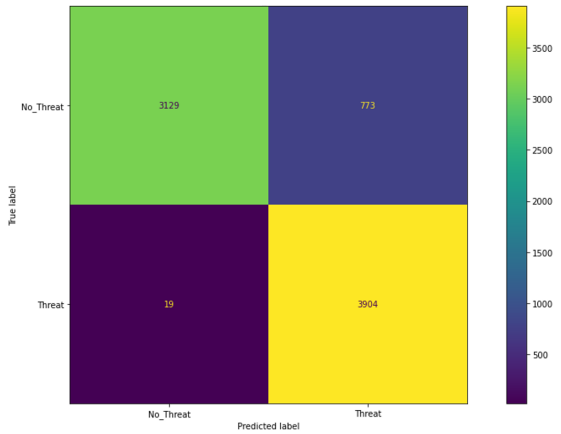
Figure 6: Classification Reports: Binary Classification



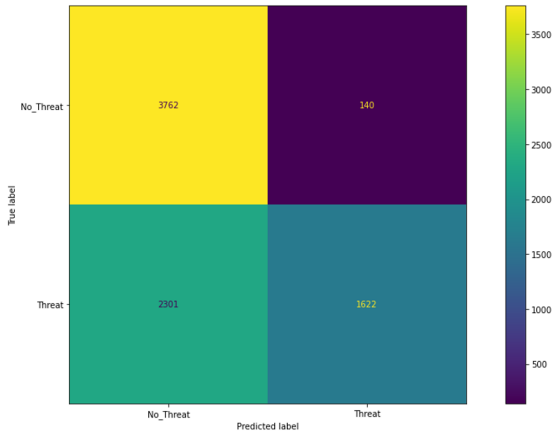
(a) SVM



(b) VGG-16



(c) Own CNN



(d) EfficientNetV2S

Figure 7: Confusion Matrices: Binary Classification

SVM

	precision	recall	f1-score
Gun	0.12	0.08	0.10
Knife	0.46	0.17	0.25
Nothing	0.97	0.25	0.40
Pliers	0.26	0.29	0.27
Scissors	0.32	0.17	0.22
Wrench	0.25	0.83	0.38
accuracy			0.30
macro avg	0.40	0.30	0.27
weighted avg	0.42	0.30	0.28

Vgg16arh

	precision	recall	f1-score
Gun	0.23	0.43	0.30
Knife	0.42	0.42	0.42
Nothing	0.96	0.20	0.33
Pliers	0.80	0.64	0.71
Scissors	0.37	0.65	0.47
Wrench	0.68	0.43	0.52
accuracy			0.47
macro avg	0.58	0.46	0.46
weighted avg	0.60	0.47	0.48

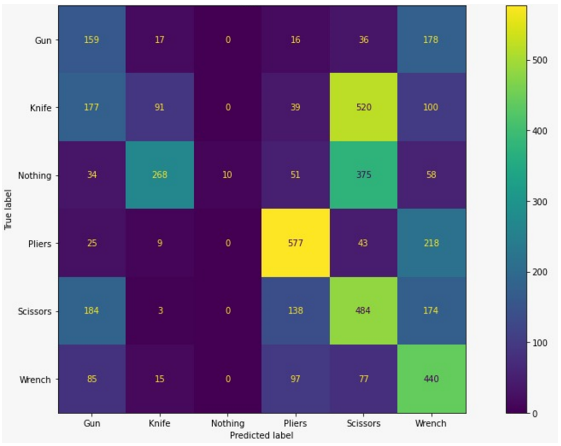
own_model

	precision	recall	f1-score
Gun	0.23	0.54	0.32
Knife	0.45	0.19	0.27
Nothing	0.88	0.16	0.27
Pliers	0.65	0.74	0.69
Scissors	0.30	0.31	0.30
Wrench	0.47	0.76	0.58
accuracy			0.43
macro avg	0.50	0.45	0.41
weighted avg	0.51	0.43	0.41

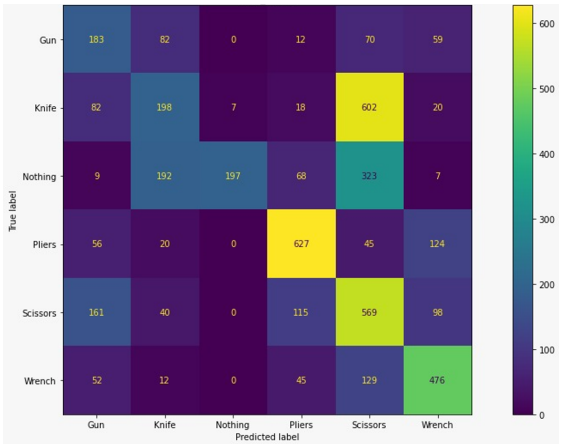
CNN-TL

	precision	recall	f1-score
Gun	0.37	0.38	0.37
Knife	0.28	0.15	0.20
Nothing	0.81	0.47	0.59
Pliers	0.44	0.52	0.48
Scissors	0.34	0.40	0.37
Wrench	0.41	0.65	0.50
accuracy			0.42
macro avg	0.44	0.43	0.42
weighted avg	0.44	0.42	0.41

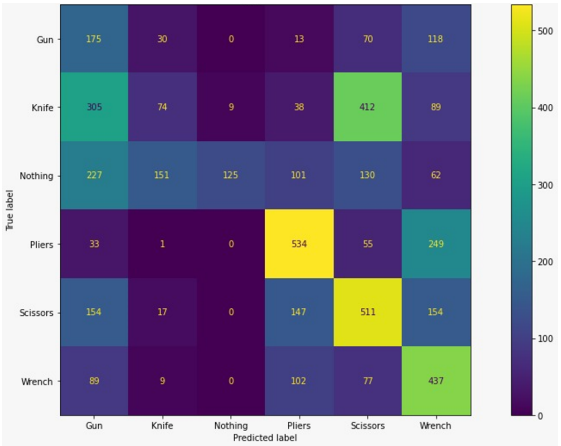
Figure 8: Classification Reports: Multi-Class Classification



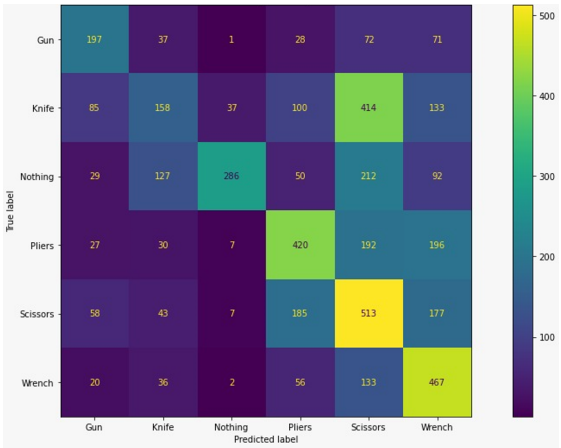
(a) SVM



(b) VGG-16



(c) Own CNN



(d) EfficientNetV2S

Figure 9: Confusion Matrices: Multi-Class Classification

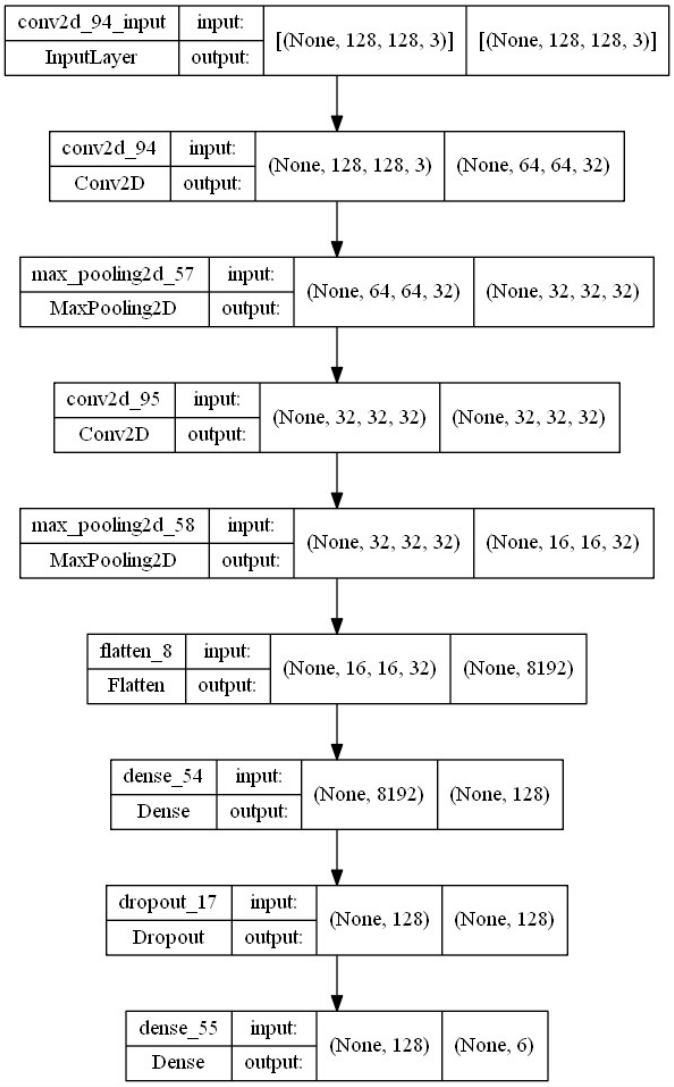


Figure 10: Model Architecture: SVM

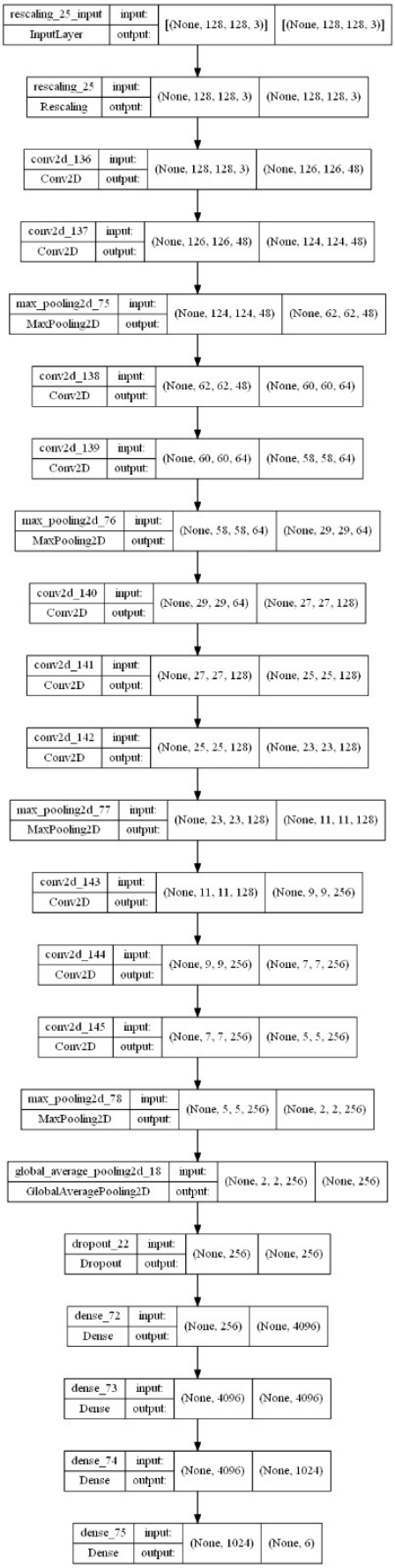


Figure 11: Model Architecture: Simplified VGG-16

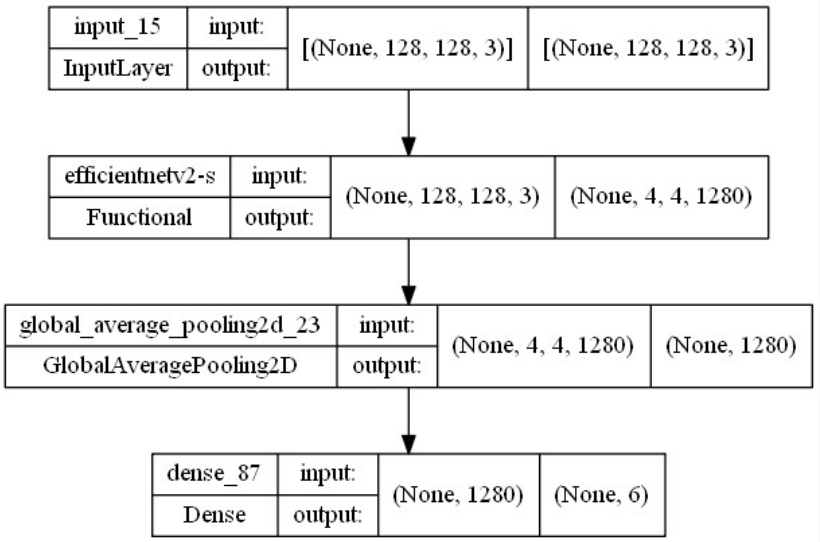


Figure 12: Model Architecture: Transfer Learning Based on EfficientNetV2S

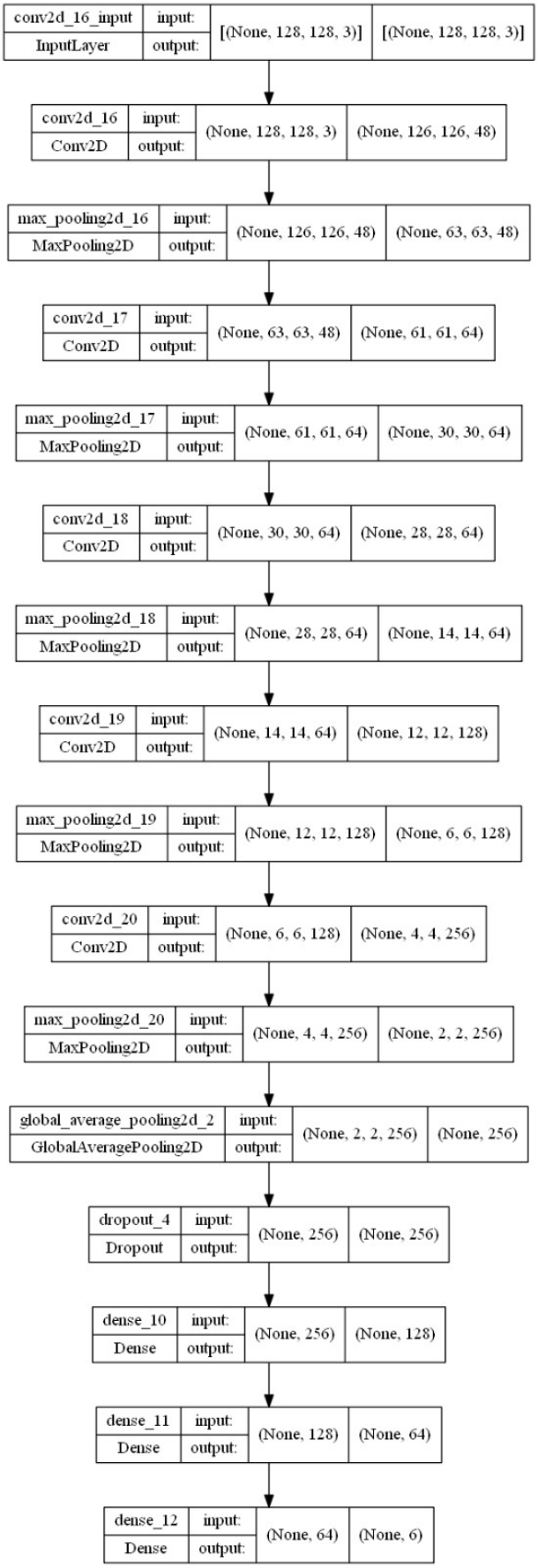
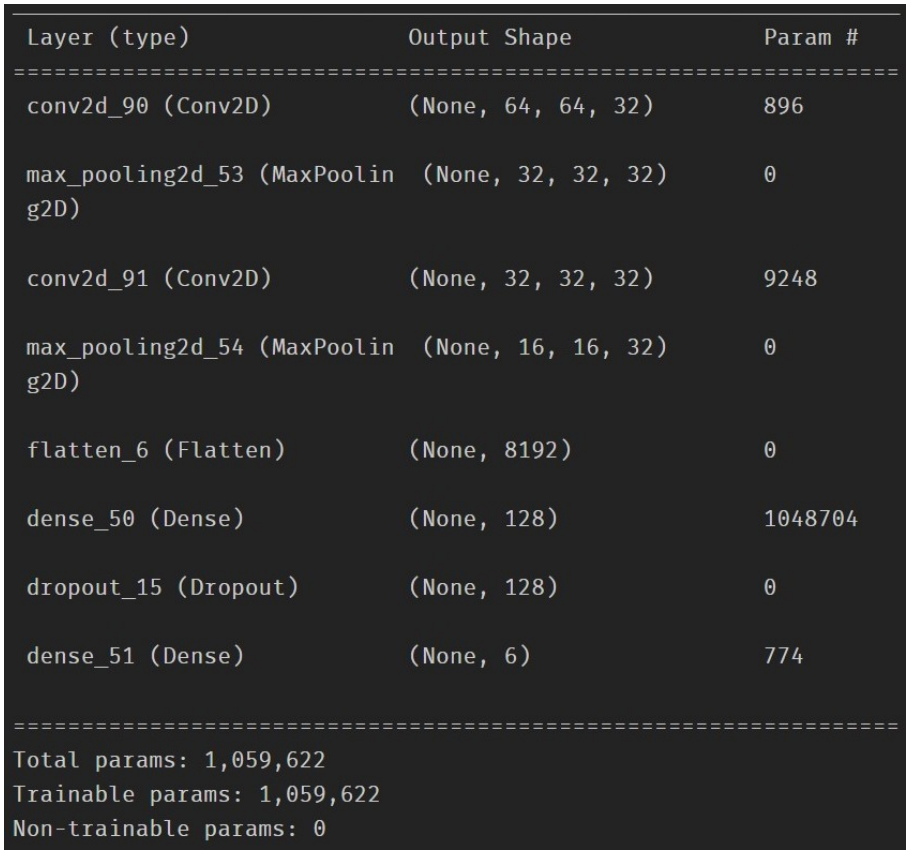


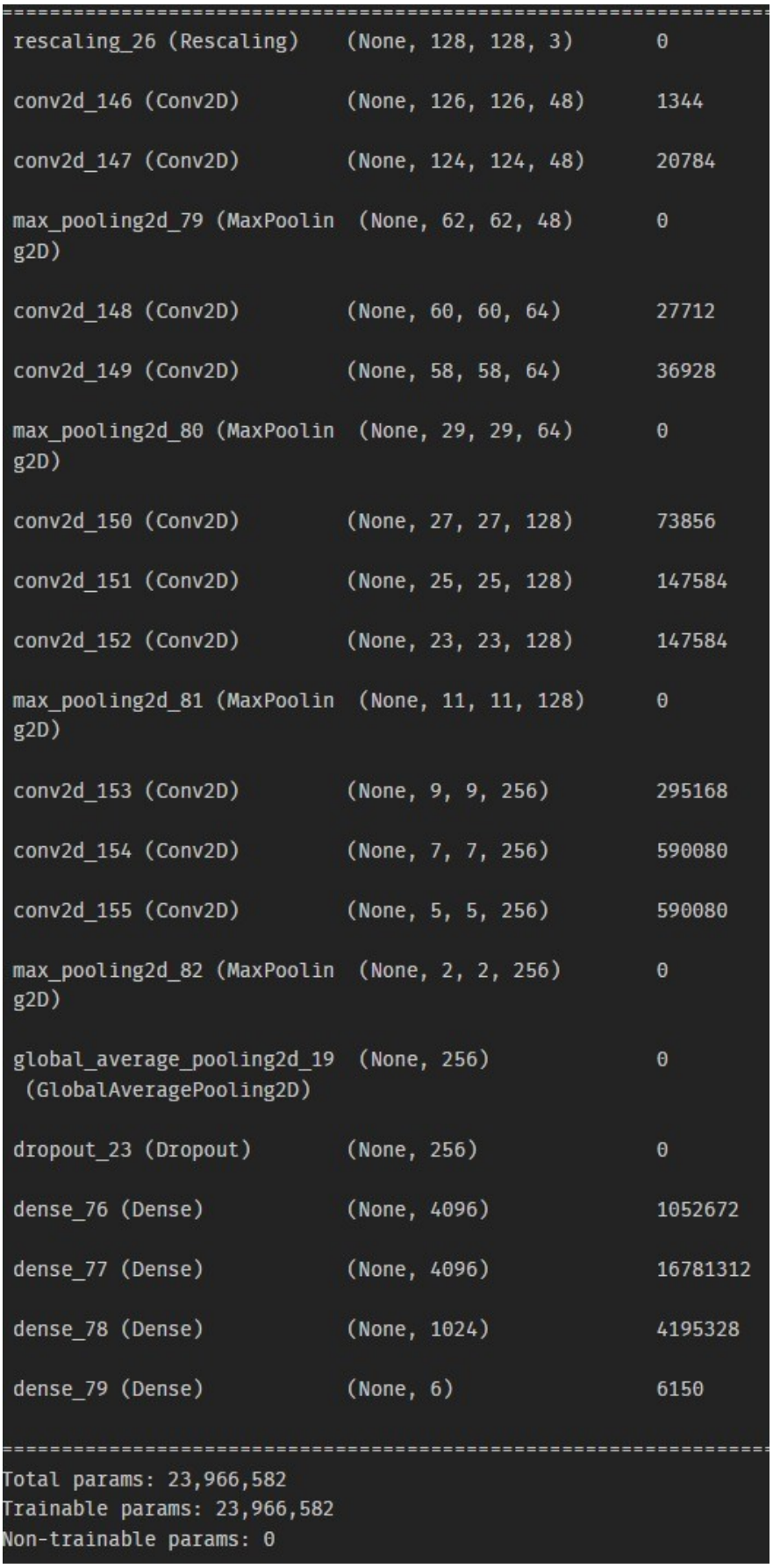
Figure 13: Model Architecture: Own Model



Layer (type)	Output Shape	Param #
conv2d_90 (Conv2D)	(None, 64, 64, 32)	896
max_pooling2d_53 (MaxPooling2D)	(None, 32, 32, 32)	0
conv2d_91 (Conv2D)	(None, 32, 32, 32)	9248
max_pooling2d_54 (MaxPooling2D)	(None, 16, 16, 32)	0
flatten_6 (Flatten)	(None, 8192)	0
dense_50 (Dense)	(None, 128)	1048704
dropout_15 (Dropout)	(None, 128)	0
dense_51 (Dense)	(None, 6)	774

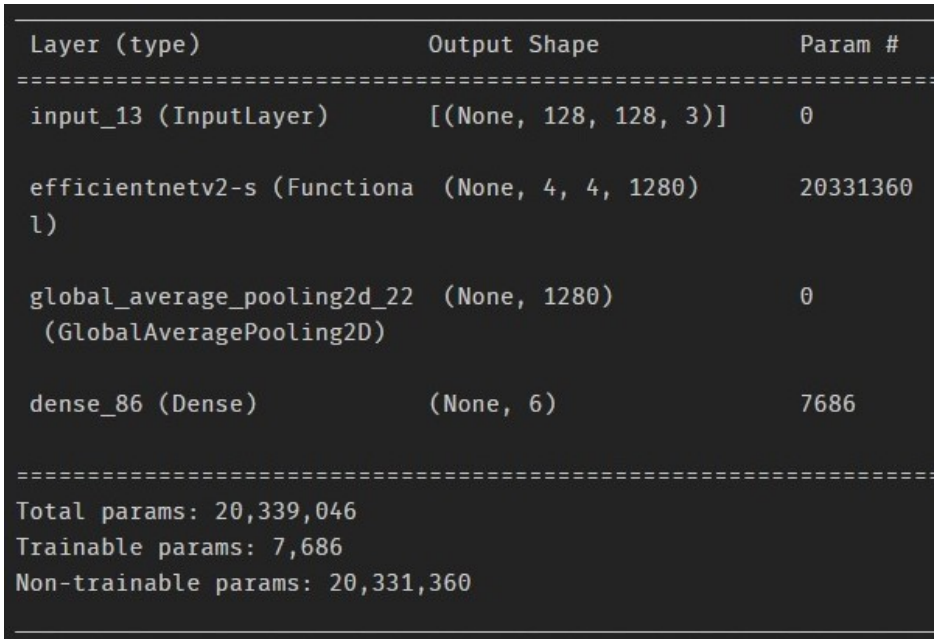
Total params: 1,059,622
Trainable params: 1,059,622
Non-trainable params: 0

Figure 14: Model Summary: SVM



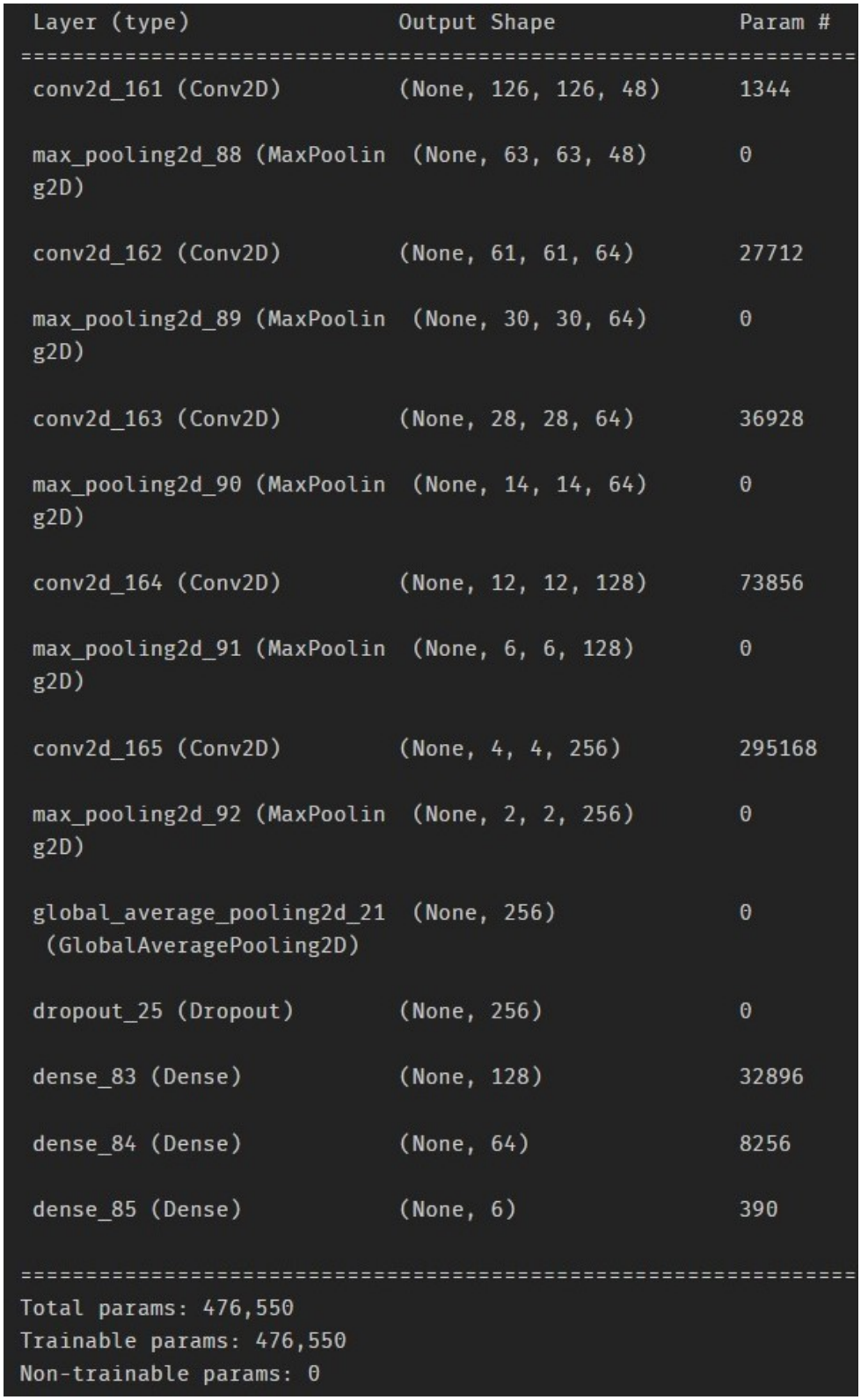
```
=================================================================
rescaling_26 (Rescaling)    (None, 128, 128, 3)       0

conv2d_146 (Conv2D)         (None, 126, 126, 48)      1344

conv2d_147 (Conv2D)         (None, 124, 124, 48)      20784

max_pooling2d_79 (MaxPoolin  (None, 62, 62, 48)       0
g2D)

conv2d_148 (Conv2D)         (None, 60, 60, 64)        27712

conv2d_149 (Conv2D)         (None, 58, 58, 64)        36928

max_pooling2d_80 (MaxPoolin  (None, 29, 29, 64)       0
g2D)

conv2d_150 (Conv2D)         (None, 27, 27, 128)       73856

conv2d_151 (Conv2D)         (None, 25, 25, 128)       147584

conv2d_152 (Conv2D)         (None, 23, 23, 128)       147584

max_pooling2d_81 (MaxPoolin  (None, 11, 11, 128)      0
g2D)

conv2d_153 (Conv2D)         (None, 9, 9, 256)         295168

conv2d_154 (Conv2D)         (None, 7, 7, 256)         590080

conv2d_155 (Conv2D)         (None, 5, 5, 256)         590080

max_pooling2d_82 (MaxPoolin  (None, 2, 2, 256)        0
g2D)

global_average_pooling2d_19  (None, 256)              0
 (GlobalAveragePooling2D)

dropout_23 (Dropout)        (None, 256)               0

dense_76 (Dense)            (None, 4096)              1052672

dense_77 (Dense)            (None, 4096)              16781312

dense_78 (Dense)            (None, 1024)              4195328

dense_79 (Dense)            (None, 6)                 6150

=================================================================
Total params: 23,966,582
Trainable params: 23,966,582
Non-trainable params: 0
```

Figure 15: Model Summary: Simplified VGG-16



```
 Layer (type)                Output Shape              Param #
=================================================================
 input_13 (InputLayer)       [(None, 128, 128, 3)]     0

 efficientnetv2-s (Functiona  (None, 4, 4, 1280)       20331360
 l)

 global_average_pooling2d_22  (None, 1280)             0
  (GlobalAveragePooling2D)

 dense_86 (Dense)            (None, 6)                 7686

=================================================================
Total params: 20,339,046
Trainable params: 7,686
Non-trainable params: 20,331,360
_________________________________________________________________
```

Figure 16: Model Summary: Transfer Learning Based on EfficientNetV2S



Layer (type)	Output Shape	Param #
conv2d_161 (Conv2D)	(None, 126, 126, 48)	1344
max_pooling2d_88 (MaxPooling2D)	(None, 63, 63, 48)	0
conv2d_162 (Conv2D)	(None, 61, 61, 64)	27712
max_pooling2d_89 (MaxPooling2D)	(None, 30, 30, 64)	0
conv2d_163 (Conv2D)	(None, 28, 28, 64)	36928
max_pooling2d_90 (MaxPooling2D)	(None, 14, 14, 64)	0
conv2d_164 (Conv2D)	(None, 12, 12, 128)	73856
max_pooling2d_91 (MaxPooling2D)	(None, 6, 6, 128)	0
conv2d_165 (Conv2D)	(None, 4, 4, 256)	295168
max_pooling2d_92 (MaxPooling2D)	(None, 2, 2, 256)	0
global_average_pooling2d_21 (GlobalAveragePooling2D)	(None, 256)	0
dropout_25 (Dropout)	(None, 256)	0
dense_83 (Dense)	(None, 128)	32896
dense_84 (Dense)	(None, 64)	8256
dense_85 (Dense)	(None, 6)	390

Total params: 476,550
Trainable params: 476,550
Non-trainable params: 0

Figure 17: Model Summary: Own Model