A NOVEL CNN APPROACH FOR AUTOMATED KNIFE DETECTION IN SURVEILLANCE IMAGES

SVECHA JINDAL | PRACHI TEWARI

This work was supported by DRDO, under the guidance of Ashish Jindal

Abstract

In this study, we present an innovative approach to enhancing surveillance systems through automated weapon detection, with a particular emphasis on identifying knives in images. As the need for reliable and effective security measures intensifies in response to rising public safety concerns, our work seeks to advance the development of sophisticated surveillance technologies capable of real-time threat identification. We assembled a comprehensive dataset of approximately 16,000 images from various sources, meticulously categorized into knife and non-knife classes. Diverging from models like YOLO that demand extensive annotated datasets, we implemented a custom Convolutional Neural Network (CNN) tailored for binary classification. This model was specifically selected to efficiently distinguish between the presence and absence of knives, utilizing the precision of our binary-labeled dataset. Our findings highlight the effectiveness of the custom CNN, underscoring its potential for seamless integration into existing surveillance systems, thereby contributing to improved public safety.

Introduction

Modern times have made public safety even more important, hence it is imperative that monitoring technology evolve. Even though they are effective in many aspects, traditional surveillance systems frequently cannot identify and address dangers in real time. This paper presents a new method for improving surveillance systems using automated weapon identification, specifically concentrating on detecting knives in pictures.

Our research addresses this need by developing a custom Convolutional Neural Network (CNN) designed specifically for the binary classification of images to detect the presence of knives. Unlike existing models such as YOLO, which require extensive annotated datasets, our approach utilizes a tailored CNN that efficiently distinguishes between knife and non-knife images using a precisely curated dataset of approximately 16,000 images. This binary classification model not only offers a significant advancement in real-time threat identification but also holds promise for seamless integration into current surveillance frameworks.

This introduction covers the basic concepts of CNNs and the digital photo data structure, delving into the ways that pooling layers, convolution operations, padding, stride, activation functions, and other features enhance the performance of our unique model. We hope to provide a thorough understanding of how our technique improves surveillance systems' capacity to detect potential threats and promote public safety by outlining the architecture of our CNN, the training configuration, and the assessment criteria employed.

Convolutional Neural Network

Convolutional neural networks, or CNNs, have transformed computer vision in recent years by offering cutting-edge solutions for a range of image processing applications. Convolutional operations enable CNNs to understand intricate patterns, which has made them a vital tool in digital image processing. This study provides a thorough examination of a customized CNN model, describing the convolution processes, the digital photo data format, and the functions and layers that make up the network.

Digital Photo Data Structure

Digital images are represented as large matrices of numbers, where each number corresponds to the brightness of a pixel. In color images, the RGB model is used, consisting of three matrices representing the red, green, and blue channels. In contrast, black-and-white images require only one matrix. The values in these matrices range from 0 to 255, providing a balance between efficient information storage and the sensitivity of the human eye .

Convolution in CNNs

In computer vision techniques and convolutional neural networks (CNNs), convolution is a fundamental function. In order to produce a feature map that emphasizes particular aspects of an image, a tiny matrix known as a kernel or filter is applied to the image. This process can be mathematically expressed as:

$$G[m,n] = (f * h)[m,n] = \sum_j \sum_k h[j,k] f[m-j, n-k]$$

where ‘f’ represents the input image, ‘h’ is the kernel, and (m,n) are the coordinates of the resulting feature map.

The operation entails positioning the kernel over a pixel, multiplying the kernel values with corresponding image values, and summing them. The result is then placed in the feature map. This process is repeated across the image, utilizing different filters to extract various features like edges, textures, and patterns.

For instance, a 5x5x1 image convolved with a 3x3x1 kernel and a stride of \(s = 1 \) produces a feature map, accounting for boundaries via zero-padding. Multiple filters allow the detection of diverse features, essential for tasks like image recognition and object detection. The stride defines the movement step of the filter across the image, affecting the operation's resolution and efficiency.

Padding and Stride

One drawback of convolution operations is their tendency to focus more on the center of the image than its corners. This issue can be mitigated by padding, where rows and columns of zeros are added around the input tensor. The stride, which defines the step size of the filter over the image, also plays a crucial role. Larger strides result in smaller output feature maps and can help reduce the computational load while capturing essential features .

Activation functions

By adding non-linearity to the network, activation functions help the model recognize intricate patterns. The generated feature map is subjected to an activation function after convolution and optional bias addition. ReLU, Sigmoid, and Tanh are popular activation functions that, depending on the task at hand, each offer different advantages.

$$\begin{aligned} c &= F + b \\ c &= I * K + b \\ \mathrm{Conv}(I, K) &= \phi_a(c) \\ &= \phi_a(I * K + b) \\ \text{where } \phi_a & \text{ is an activation function.} \end{aligned}$$

Pooling layer

By reducing the spatial size of the feature maps, the pooling layer helps to extract prominent characteristics and strengthens the model's resistance to input fluctuations. Different pooling operations exist, including average and maximum pooling. While average pooling determines the average value, max pooling chooses the maximum value from a patch of the feature map. By lowering the dimensionality of the feature maps, pooling contributes to a reduction in the number of parameters and computations.

$$\mathrm{Conv}(I, K) = C$$

$$P = \phi_p(C)$$

where ϕ_p is a pooling function.

$$\text{dim of } P = \left(\frac{m_1 + 2p - n_1}{s}\right) \times \left(\frac{m_2 + 2p - n_2}{s}\right) \times m_c$$

where,

$m_1 \times m_2 =$ the dimensions of input image

$n_1 \times n_2 =$ the dimensions of padding kernel

here, s stands for stride and p stands for padding.

Custom CNN Model

This study's own CNN model is made up of several convolutional layers, each of which is followed by a pooling layer and an activation function. The architecture of the model is built to gradually capture more complex information, and the last layer does classification by figuring out the likelihood of each class.

Model Training Configuration: Optimizer, Loss Function, and Metrics

Convolutional neural networks (CNNs) perform best when its architecture is combined with the metrics, loss functions, and optimization strategies applied during training. In order to compile our project, we utilized the following key components to optimize the training process and ensure accurate predictions:

1. ***Optimizer='adam'***:
   Overview: Because Adam (Adaptive Moment Estimation) may adjust learning rates for any parameter, it is a popular optimization approach in deep learning. It combines the benefits of two other widely used techniques: RMSProp, which performs well in non-stationary environments, and AdaGrad, which works well with sparse gradients.Adam adjusts the model's parameters by estimating the first and second moments of the gradients, enabling it to converge faster and more efficiently.
   Significance: The use of Adam optimizes the model by dynamically adjusting the learning rate during training, helping to avoid issues such as vanishing or exploding gradients. This is particularly important when training deep networks with complex architectures, where consistent and stable convergence is critical for achieving high performance.

2. ***Loss Function: Binary Cross-Entropy:***
   Overview: A loss function utilized for binary classification problems is binary cross-entropy, sometimes referred to as log loss. It calculates the difference between the actual class label (ground truth) and the expected probability (model output).
   Significance: The model learns to distinguish between the two classes with accuracy by reducing binary cross-entropy. This makes it an essential part of tasks like weapon detection, where the output is binary (e.g., weapon present or not). With the help of this loss function, the model is penalized for making inaccurate predictions, which helps it perform better.

3. ***Metrics='accuracy'***:
   Overview: One common metric used to assess how well categorization models function is accuracy. The ratio of accurate forecasts to total predictions is used to compute it.
   Significance: When the dataset is balanced, accuracy offers a clear-cut and logical way to assess the model's performance. Accuracy tracking during training facilitates early identification of problems like overfitting or underfitting and helps track the model's development. To provide a more thorough assessment of the model's performance, however, further metrics like precision, recall, and the F1-score could be needed for unbalanced datasets.

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Number of Predictions}}$$

Evaluation Metrics:

1. ***Confusion Matrix***

   A confusion matrix is a 2x2 table used to evaluate the performance of a binary classification model. It consists of:

   True Positives (TP): Correctly predicted positive cases.

   True Negatives (TN): Correctly predicted negative cases.

   False Positives (FP): Incorrectly predicted positive cases (Type I error).

   False Negatives (FN): Incorrectly predicted negative cases (Type II error).

   From these values, several metrics can be derived:

Formulas for Key Metrics

- **Accuracy**:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precision**:

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall (Sensitivity)**:

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F1 Score**:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Gain insight into the classifier's advantages and disadvantages with the use of these measures.
Where there is a high cost of false negatives, recall is more significant, although precision is critical in the former case. Between recall and precision, the F1 score offers a balance.

2. ***Receiver Operating Characteristic (ROC) Curve***

   When working on binary classification tasks, like weapon detection, the ROC curve and *AUC*(area under curve) are really important for understanding how well your model is performing. These metrics go beyond just showing accuracy and are especially useful when your dataset isn't balanced.

   The *ROC curve* lets you see how well your model identifies true positives compared to how often it mistakenly labels negatives as positives at various thresholds. It's essentially a way to evaluate how well your model is balancing the act of catching the right positives while minimizing false alarms. If your ROC curve heads toward the upper-left corner, it means your model is doing a great job at distinguishing between the classes.

3. ***Precision-Recall Curve***

   Precision measures the relevance of positive predictions made by a model. Specifically, it calculates the proportion of true positive predictions relative to the total predicted positive instances. A high precision indicates that when the model predicts a positive class, it is usually correct, demonstrating a low false positive rate.

   Recall, conversely, measures a model's ability to identify all relevant instances. It calculates the proportion of actual positive instances that were correctly predicted positive. A high recall signifies that the model is capturing most true positive cases, demonstrating a low false negative rate.

   Precision-Recall Curve:

   The precision-recall curve visually represents the recall vs. precision trade-off at different classification levels. A model's threshold, which determines the point at which an event is labeled as positive, affects precision and recall. You can change the threshold to increase recall at the cost of precision, or vice versa. Studying the precision-recall curve helps in examining these trade-offs.

   Precision-Recall Trade-offs:

   High Recall, Low Precision: Retrieves many results, but many are incorrect. For instance, in a medical context, it might identify most cases of a disease but also mislabel healthy individuals, leading to unnecessary treatments.

   High Precision, Low Recall: Retrieves fewer results, but most are correct. This model minimizes false positives but may miss many true positives. For example, it might detect only a few disease cases but with high accuracy, leaving many cases undetected.

Ideal Scenario Achieves both high precision and high recall, accurately identifying most positives while minimizing false positives. This is crucial for applications where both false positives and negatives are costly.

Building the model:

The process began with a convolutional layer using 32 filters, each sized 3x3 pixels. The *ReLU activation function* was applied to introduce non-linearity by converting all negative values to zero. The input shape was set to 150x150 pixels with 3 color channels (RGB).

Next, *max pooling* with a 2x2 window was used to reduce the image size by half. This step simplifies the data, emphasizes important features, and reduces computational effort.

After pooling, the data, initially in a 2D format, was converted into a 1D vector using a *Flatten layer*, preparing it for the subsequent stages.

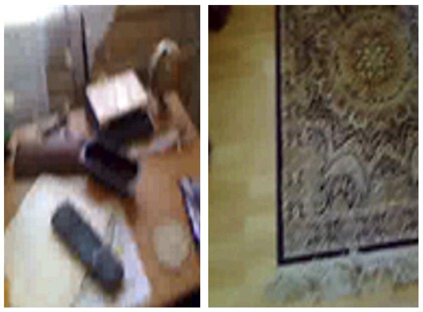
A dense layer with 512 neurons was then added. This fully connected layer combines features from earlier layers and uses the *ReLU activation function* to manage non-linearity.

To prevent overfitting, a dropout layer was included, which randomly drops 50% of the neurons during training. This helps the model generalize better rather than memorizing the training data.

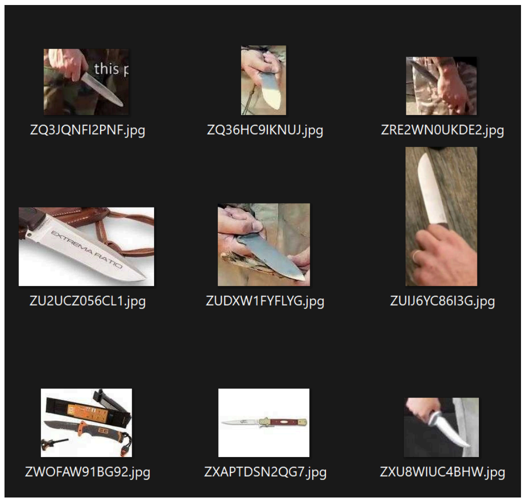
Finally, the output layer, with a single neuron and a *sigmoid activation function*, produces a probability between 0 and 1, used to make the *final classification decision.*

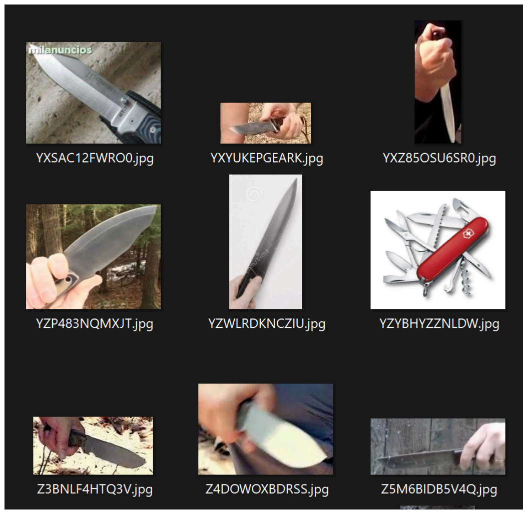
Overview of the Dataset

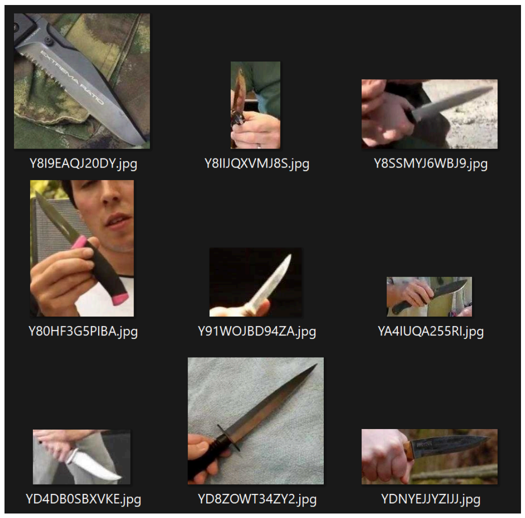
Non-Knife Images (Sample)



Knife Images (Sample)

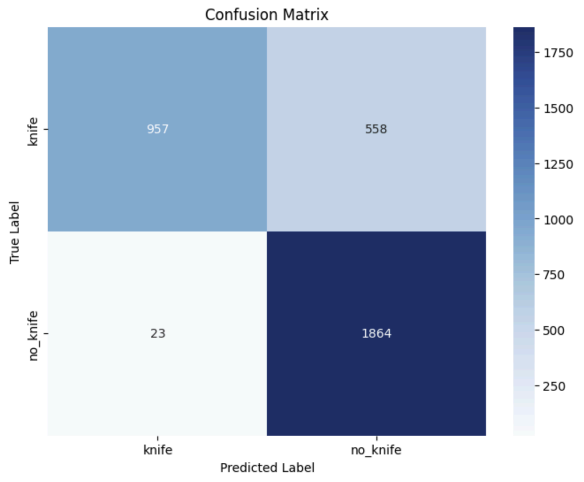




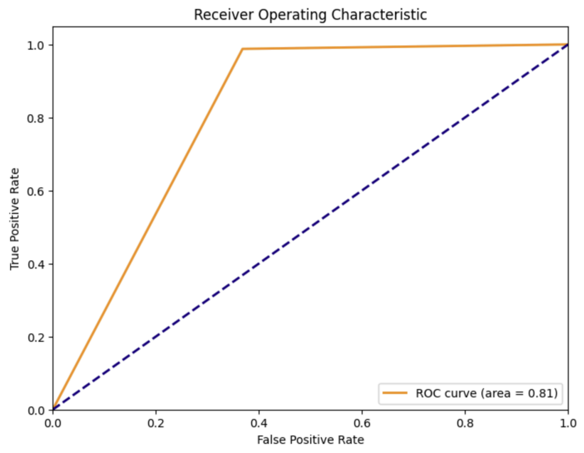


Results:

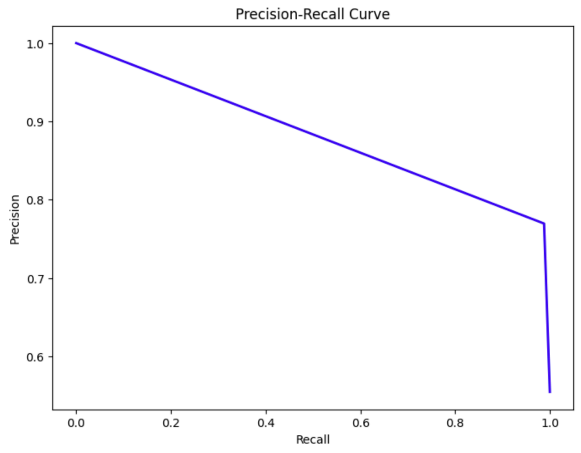
Confusion Matrix Chart



ROC Curve



PR Curve



\

Confusion Matrix:

```
[[ 957  558]
 [  23 1864]]
```

Classification Report:

```
              precision    recall  f1-score   support

       knife       0.98      0.63      0.77      1515
    no_knife       0.77      0.99      0.87      1887

    accuracy                           0.83      3402
   macro avg       0.87      0.81      0.82      3402
weighted avg       0.86      0.83      0.82      3402
```

Conclusion:

In this paper, we presented a custom Convolutional Neural Network (CNN) approach for automated knife detection in surveillance images. By leveraging a curated dataset of approximately 16,000 images, our model demonstrated high effectiveness in binary classification tasks, distinguishing between knife and non-knife images with significant accuracy.

Our methodology diverges from traditional models like YOLO that require extensive annotated datasets, focusing instead on a tailored CNN that efficiently processes images in real-time, making it suitable for integration into existing surveillance systems. The model's performance was validated using metrics such as confusion matrices, ROC curves, and precision-recall curves, which confirmed its robustness and reliability.

The implementation of this CNN model represents a significant advancement in automated threat detection, particularly in enhancing public safety measures. Our findings suggest that this approach can be seamlessly integrated into current surveillance frameworks, providing a valuable tool for real-time threat identification and contributing to the ongoing efforts to improve security systems.

References:

https://www.mdpi.com/2076-3417/11/16/7535
https://www.ijarst.in/public/uploads/paper/672821668418043.pdf
https://ieeexplore.ieee.org/iel7/6287639/6514899/09353483.pdf