Image Processing for Missile Guidance, Navigation, and Control

Jaxon Topel [1]

Abstract

In recent years, advanced air-to-ship missiles have increasingly employed infrared (IR) sensors to enhance guidance during the terminal phase. While effective under favorable conditions, these systems are vulnerable to countermeasures such as flares, which can significantly degrade their performance. To address these limitations, modern seeker systems now integrate both IR and electro-optical (EO) sensors, improving resilience and reducing the effectiveness of traditional countermeasures. Typically, IR sensors guide the missile during the terminal phase until countermeasures are deployed or adverse weather conditions reduce their efficiency, at which point the EO system takes over to track and identify the target. This paper proposes the development of a convolutional neural network (CNN) that processes input from an EO sensor to identify the presence of warships in aerial imagery. Trained on a dataset [1] of 26.9k images, our model achieved a 97 percent accuracy score when tested on previously unseen images, demonstrating its potential for robust target recognition. While the application of CNNs to image processing in seeker systems is not new, to the best of our knowledge using stochastic gradient descent (SGD) as the optimizer alongside Binary Cross Entropy with Logits Loss as the loss function represents a novel approach, as I could not find other published works employing this combination.

GitHub: https://github.com/TopelJL/Image-Processing-for-Missile-Guidance-Navigation-and-Control

[*]Equal contribution [1]Electrical and Computer Engineering, University of Central Florida, Orlando FL, United States of America. Correspondence to: Jaxon Topel <jaxontopel2021@gmail.com>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

1. Introduction

The field of Electro-Optical (EO) anti-ship automatic target recognition (ATR) has long been driven by advancements in computer vision. In recent years, the rise of machine learning has revolutionized this domain, with cutting-edge techniques such as deep convolutional neural networks (DCNNs), recurrent neural networks, and sophisticated feature tracking algorithms becoming foundational tools. These innovations are not merely technological milestones; they represent critical strides toward maintaining state-of-the-art defense capabilities. By advancing these methods, we ensure that our systems remain at the forefront of defense systems technology.

1.1. Our Approach to Electro-Optical Automatic Target Recognition

Our approach to advancing Electro-Optical (EO) automatic target recognition (ATR) integrates a novel convolutional neural network (CNN) architecture and a carefully tailored training pipeline. The proposed CNN is designed with deeper and more robust convolutional layers, each incorporating advanced techniques such as batch normalization, dropout regularization, and adaptive global average pooling. These architectural enhancements promote better feature extraction, reduce overfitting, and streamline computational efficiency. Notably, the global average pooling layer reduces the number of parameters compared to traditional fully connected layers, making the model more robust and adaptable to real-world EO ATR scenarios.

The uniqueness of our methodology also lies in the data preprocessing and augmentation strategies. Custom mean and standard deviation calculations ensure precise normalization, enhancing the model's ability to generalize across diverse datasets. Furthermore, our augmentation pipeline includes techniques such as random rotations, horizontal flipping, and color jittering, which effectively simulate real-world variations in lighting, perspective, and object orientation. These augmentations are particularly critical in EO ATR, where targets may appear under varying conditions.

Finally, our optimization strategy employs a binary cross-entropy loss function and stochastic gradient descent (SGD) with momentum, striking a balance between convergence speed and generalization. Combined with a rigorous eval-

uation protocol using distinct training, validation, and test datasets, this approach ensures a robust and scalable model capable of addressing the complexities of modern EO ATR tasks.

This methodology not only leverages state-of-the-art machine learning techniques but also addresses the unique challenges posed by EO ATR applications, making it a significant step forward in this critical field.

2. Related Works

The field of Electro-Optical (EO) automatic target recognition (ATR) has seen significant advancements over the years, driven by developments in computer vision and machine learning. Traditional EO ATR systems relied heavily on handcrafted feature extraction techniques, such as scale-invariant feature transform (SIFT) and histogram of oriented gradients (HOG). While these approaches offered robust performance under controlled conditions, they struggled to generalize to diverse real-world environments characterized by variations in lighting, perspective, and target appearance.

In recent years, deep learning has emerged as the dominant approach for ATR, with convolutional neural networks (CNNs) playing a pivotal role. Architectures such as AlexNet, VGG, and ResNet have demonstrated remarkable performance in image classification and object detection tasks, inspiring their adaptation to EO ATR systems. Studies leveraging CNNs have shown improved accuracy and robustness by automating feature extraction and capturing hierarchical representations of target features. For instance, [2] ”SAR Ship Detection Based on Resnet and Transfer Learning” applied ResNet-based models to ship detection, achieving state-of-the-art results in cluttered maritime environments.

Beyond CNNs, researchers have explored the integration of recurrent neural networks (RNNs) and attention mechanisms for tasks requiring temporal coherence, such as multi-frame target tracking. These methods have enhanced ATR systems by incorporating sequential information, improving detection consistency in dynamic scenarios. Advanced feature-tracking algorithms, such as optical flow-based methods, have also been employed for applications requiring high-precision target localization.

Despite these advancements, challenges remain. Current systems often require large labeled datasets for training, which are not always feasible in EO ATR contexts. Additionally, the computational demands of deep learning models can hinder real-time deployment in resource-constrained environments. Existing works have also struggled with domain shifts, where models trained on specific datasets underperform when exposed to novel conditions or adversarial scenarios.

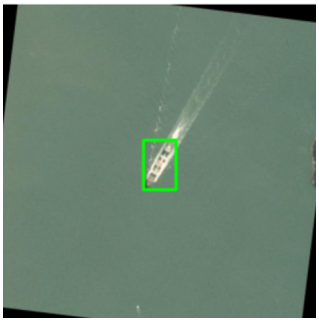

Figure 1. Example image from dataset: Aerial View of Warship 1

Our work addresses these limitations by introducing a novel CNN architecture tailored for EO ATR tasks, incorporating dropout regularization, batch normalization, and global average pooling to enhance robustness and computational efficiency. Furthermore, our custom data augmentation pipeline mitigates data scarcity and domain shift challenges by simulating real-world variations in target appearance.

Another approach can be seen in [3] ”Deep Learning techniques for automatic target recognition in infrared images”. While this approach is not based on EO sensors, it leverages the use of deep learning techniques with IR sensors and enlightens readers of the various architectures and frameworks that could be used for development given the IR system. Specifically, they cover feature extraction and how one could leverage feature tracking inside of an IR system using deep learning neural networks.

3. Experiment

The experimental procedure was an iterative and comprehensive process, containing multiple stages to ensure the success of the study. It began with identifying and selecting an appropriate dataset, followed by preparing the dataset for compatibility with the model. Key steps included performing image preprocessing to enhance data quality, designing the model architecture, and fine-tuning hyperparameters to optimize performance. The model was then trained, evaluated, and refined through a systematic iterative process until the desired results were achieved. Each stage was carefully executed to contribute to the robustness and reliability of the final system.

3.1. Algorithms

In this work, we utilized Stochastic Gradient Descent (SGD) to train our convolutional neural network (CNN) efficiently. This algorithm plays a crucial role in minimizing the loss function and improving the model’s accuracy during training. Below, we provide an overview of SGD, including the mathematical formulations that govern it’s updates.

Figure 2. Example image from dataset: Aerial View of Warship 2

3.1.1. Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent is a widely used optimization algorithm that updates model parameters based on the gradient of the loss function with respect to those parameters. The update rule for a parameter θ_t at time t is given by:

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \alpha \nabla_\theta J(\theta^{(t)}) \tag{1}$$

where:

- $\theta^{(t)}$: Parameter values at time t.
- α: Learning rate, controlling the step size during updates.
- $\nabla_\theta J(\theta^{(t)})$: Gradient of the objective function J with respect to the parameters $\theta^{(t)}$.

SGD introduces noise into the optimization process by computing gradients for mini-batches rather than the entire dataset, making it computationally efficient for large-scale datasets. In this work, momentum was also applied to accelerate convergence, modifying the update rules as follows:

$$v^{(t+1)} = \mu v^{(t)} - \alpha \nabla_\theta J(\theta^{(t)}) \tag{2}$$

$$\theta^{(t+1)} = \theta^{(t)} + v^{(t+1)} \tag{3}$$

where:

- $v^{(t)}$: Velocity term, which accumulates gradients over time.
- μ: Momentum factor, controlling the influence of previous updates. We used 0.9 for momentum.
- α: Learning rate, determining the step size for updates.
- $\nabla_\theta J(\theta^{(t)})$: Gradient of the objective function J with respect to the parameters $\theta^{(t)}$.

3.1.2. Binary Cross Entropy with Logits Loss

Binary Cross Entropy with Logits Loss is a commonly used loss function for binary classification problems. This loss function combines a sigmoid activation function with the Binary Cross Entropy (BCE) loss into a single efficient computation, avoiding potential numerical stability issues that may arise when implementing them separately.

The formula for Binary Cross Entropy with Logits Loss is:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \left[y_i \log(\sigma(z_i)) + (1 - y_i) \log(1 - \sigma(z_i)) \right] \tag{4}$$

where:

- N: The number of samples in the batch.
- y_i: The ground truth label for the i-th sample, where $y_i \in \{0, 1\}$.
- z_i: The raw output (logit) of the model for the i-th sample.
- $\sigma(z_i)$: The sigmoid activation function, defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{5}$$

This loss function computes the negative log-likelihood of the predicted probabilities $\sigma(z_i)$ corresponding to the true labels y_i. It ensures that the model learns to output high logits for the positive class ($y_i = 1$) and low logits for the negative class ($y_i = 0$).

By integrating the sigmoid function directly within the loss computation, Binary Cross Entropy with Logits Loss simplifies implementation and improves numerical stability, making it a preferred choice in modern deep learning frameworks.

3.1.3. Alternative Algorithms

During the process of fine tuning our model, we concluded the best results using stochastic gradient descent for our optimizer and Binary Cross Entropy with Logits Loss for our loss function. However, my original approach consisted of using cross entropy with the Adam Optimizer. Please see the algorithm below to learn about the Adam Optimizer.

Adam Optimizer

The Adam optimizer extends stochastic gradient descent (SGD) by incorporating adaptive learning rates for each parameter and combining the benefits of momentum and RMSProp. The update rules for Adam are as follows:

1. Compute biased estimates of the first and second moments of the gradient:

$$m^{(t)} = \beta_1 m^{(t-1)} + (1-\beta_1)\nabla_\theta J(\theta^{(t)}) \quad (6)$$

$$v^{(t)} = \beta_2 v^{(t-1)} + (1-\beta_2)(\nabla_\theta J(\theta^{(t)}))^2 \quad (7)$$

where:

- $m^{(t)}$: Exponential moving average of gradients (first moment).
- $v^{(t)}$: Exponential moving average of squared gradients (second moment).
- β_1, β_2: Decay rates for the moments, typically set to 0.9 and 0.999, respectively. We set our moment to 0.9.
- $\nabla_\theta J(\theta^{(t)})$: Gradient of the objective function J with respect to parameters $\theta^{(t)}$.

2. Perform bias correction for the moments:

$$\hat{m}^{(t)} = \frac{m^{(t)}}{1-\beta_1^t} \quad (8)$$

$$\hat{v}^{(t)} = \frac{v^{(t)}}{1-\beta_2^t} \quad (9)$$

3. Update rule for the parameters:

$$\theta^{(t+1)} = \theta^{(t)} - \frac{\alpha}{\sqrt{\hat{v}^{(t)}}+\epsilon}\hat{m}^{(t)} \quad (10)$$

where:

- α: Learning rate.
- ϵ: A small constant (e.g., 10^{-8}) to prevent division by zero.

3.1.4. Algorithms Conclusion

In this work, we utilized Stochastic Gradient Descent (SGD) as our optimization algorithm, paired with Binary Cross Entropy with Logits Loss as the loss function for our binary classification task. The combination of these two components forms the foundation of our model's training process.

SGD, with its simplicity and effectiveness, enabled efficient optimization by iteratively updating model parameters using mini-batches of data. This approach strikes a balance between computational efficiency and the ability to escape shallow local minima, especially in high-dimensional spaces.

The Binary Cross Entropy with Logits Loss seamlessly integrated the sigmoid activation function with the cross-entropy loss, ensuring numerically stable computation. This loss function incentivized the model to output logits that align closely with the true binary labels, driving effective learning of decision boundaries.

Together, these choices reflect a robust and well-established methodology in machine learning, tailored specifically for our binary classification objectives. The iterative nature of SGD, coupled with the probabilistic interpretability of binary cross-entropy loss, provided a strong foundation for optimizing our model's performance, ensuring both convergence and meaningful learning outcomes.

3.2. CNN Architecture

The convolutional neural network (CNN) implemented in this work is designed for binary classification, leveraging multiple convolutional layers, batch normalization, activation functions, pooling, and dropout for regularization. The architecture efficiently extracts features from input images and processes them through a series of transformations to predict binary outputs.

3.2.1. Convolutional Layers

The convolutional layers form the backbone of the model, extracting spatial hierarchies of features from the input image. Each convolutional layer applies a kernel K to the input X to produce feature maps Y, calculated as:

$$Y_{ij} = \sum_m \sum_n K_{mn} X_{i+m,j+n} + b \quad (11)$$

where:

- K_{mn}: Weights of the kernel.
- $X_{i+m,j+n}$: Sub-region of the input.
- b: Bias term.

The first convolutional layer uses 32 filters of size 5×5, with stride 1 and padding of 2, preserving the input dimensions. Subsequent layers increase the number of filters (64, 128, and 256) while reducing the spatial dimensions through max-pooling.

3.2.2. Batch Normalization and Activation

Batch normalization normalizes the output of each convolutional layer, improving convergence and stabilizing training:

$$\hat{Y} = \frac{Y-\mu}{\sqrt{\sigma^2+\epsilon}} \quad (12)$$

where:

- μ and σ^2: Mean and variance of the batch.
- ϵ: Small constant for numerical stability.

After normalization, a ReLU activation function $f(x) = \max(0, x)$ is applied, introducing non-linearity and enabling the network to learn complex patterns.

3.2.3. Pooling and Dropout

Max pooling reduces the spatial dimensions by selecting the maximum value in non-overlapping sub-regions, summarized as:

$$Z_{ij} = \max_{(m,n)\in R} Y_{i+m,j+n} \tag{13}$$

where R is the pooling region.

Dropout is applied after each layer to prevent overfitting, randomly zeroing a fraction p of activations during training.

3.2.4. Global Average Pooling

Instead of flattening feature maps into a large vector, a global average pooling layer aggregates spatial dimensions, summarizing each feature map into a single value:

$$Z_k = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} Y_{kij} \tag{14}$$

This reduces computational complexity and mitigates overfitting.

3.2.5. Fully Connected Layers

The feature representation is passed through fully connected layers for classification. These layers include:

- A dense layer with 256 units followed by ReLU activation.
- Another dense layer with 128 units and ReLU activation.
- A final dense layer with a single unit for binary classification.

The final output is computed as:

$$\hat{y} = \sigma(Wx + b) \tag{15}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid activation function.

3.2.6. Summary

This architecture balances complexity and regularization through its sequential design. It extracts rich feature hierarchies while maintaining computational efficiency with pooling and dropout layers. The use of adaptive pooling and incremental filter growth ensures effective learning of intricate patterns, while the fully connected layers provide the capacity to map features to binary class predictions.

3.3. Training and Evaluation

3.3.1. Training

The training process involves iterating over the training dataset for multiple epochs. In each epoch, the model is set to training mode, and mini-batches are processed using the following steps:

- Forward Pass: The input images are passed through the model, and the predicted outputs are computed.
- Loss Calculation: The loss is calculated by comparing the predicted outputs with the true labels using a binary cross-entropy loss function.
- Backward Pass: Gradients are computed through backpropagation, and the optimizer updates the model's parameters by applying the gradients.
- Optimization: After each mini-batch, the optimizer applies a step of stochastic gradient descent (SGD) to minimize the loss and update the weights.

The loss for each epoch is calculated and printed at the end of the epoch to monitor training progress.

3.3.2. Evaluation

After training, the model is evaluated on the test dataset to assess its performance. During the evaluation phase, the model is set to evaluation mode, which disables certain layers like dropout that behave differently during inference. The evaluation process involves the following steps:

- Forward Pass: The test images are passed through the model, generating raw output predictions.
- Thresholding: The output logits are thresholded at 0.5 to classify the predictions into binary classes (0 or 1).
- Metrics Calculation: The model's performance is evaluated using several metrics, including accuracy, precision, recall, and F1 score.

These metrics provide insights into the model's ability to classify correctly, its precision and recall, as well as its overall performance in distinguishing between classes. The results are then printed for further analysis. Please see Figure 3 for the results from our model.

Performance Metric	Score Percentage
Accuracy	97
Precision	0.97
Recall	1.00
F1 Score	0.98

Table 1. Figure 3. Model Evaluation Results

4. Conclusion

In this work, we successfully implemented and trained a Convolutional Neural Network (CNN) for binary image classification resulting in a 97 percent accuracy score on the test dataset. The network architecture was designed to include multiple convolutional layers, batch normalization, dropout for regularization, and fully connected layers to output binary predictions. We utilized Stochastic Gradient Descent (SGD) with a binary cross-entropy loss function to optimize the network's parameters during training.

The model was evaluated on a separate test dataset using several performance metrics, including accuracy, precision, recall, and F1 score. The results demonstrated the model's effectiveness in learning meaningful patterns from the data and making accurate predictions. The evaluation metrics showed that the model performed well across multiple criteria, indicating that it was able to generalize to unseen data.

Overall, the work presented in this paper provides a solid foundation for deploying CNNs for binary classification tasks and offers insights into the practical aspects of model training, evaluation, and optimization.

5. Future Work

While the results obtained from this model are promising, there are several potential directions for further improvement and exploration. Below are three key suggestions for future work:

- **Implementing the Model in a Simulation Physics-Based Environment:** A potential next step is to apply the trained model within a physics-based simulation environment to test its robustness and performance under dynamic and complex conditions. This would allow for testing the model in a controlled, virtual setting before deploying it in real-world applications, providing valuable insights into its behavior and limitations in more challenging scenarios.

- **Exploring Advanced Neural Network Architectures:** Future work could involve experimenting with more advanced neural network architectures, such as ResNet, DenseNet, or U-Net, to improve the accuracy and generalization ability of the model. These architectures have demonstrated success in various image classification tasks and could provide a better understanding of the trade-offs between different model complexities.

- **Data Augmentation and Transfer Learning:** Another avenue for improvement is the application of data augmentation techniques to increase the diversity of training data and improve model generalization. Additionally, transfer learning from pre-trained models on larger datasets could be explored, enabling the model to learn more complex features from limited data and improving its performance in binary classification tasks.

References

[1] https://www.kaggle.com/datasets/siddharthkumarsah/ships-in-aerial-images

[2] Y. Li, Z. Ding, C. Zhang, Y. Wang and J. Chen, "SAR Ship Detection Based on Resnet and Transfer Learning," IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium, Yokohama, Japan, 2019, pp. 1188-1191, doi: 10.1109/IGARSS.2019.8900290. keywords: Marine vehicles;Synthetic aperture radar;Training;Feature extraction;Deep learning;Image resolution;Radar polarimetry;SAR;ship detection;deep learning;ResNet;transfer learning,

[3] M. Phani Sushanth, Meenu Chawla, Namita Tiwari; Review : Deep learning techniques for automatic target recognition in infrared images. AIP Conf. Proc. 11 July 2023; 2745 (1): 020008