Project Report
On

SATELLITE IMAGERY RECOGNITION

Project I Submitted in partial fulfilment of the requirements for the degree of

B.Tech. in CSE (Artificial Intelligence and Machine Learning)

by

VIVEK THAKUR (13030820026)

HARSH RANJAN (13030820031)

DIVYA KUMAR BAID (13030820030)

SUMEDHA SARKAR (13030820032)

RITAM KONAR (13000320019)

Under the guidance of

Dr. Shibaprasad Sen

Professor

CSE (Artificial Intelligence and Machine Learning) Department

TECHNO MAIN SALT LAKE

EM 4/1 SALT LAKE CITY, SECTOR V

KOLKATA – 700091

West Bengal, India

TECHNO MAIN SALT LAKE

[Affiliated by Maulana Abul Kalam Azad University of Technology (Formerly known as WBUT)]

FACULTY OF CSE (AI & ML) DEPARTMENT

Certificate of Recommendation

This is to certify that **Vivek Thakur, Harsh Ranjan, Divya Kumar Baid, Sumedha Sarkar and Ritam Konar** have completed their project report on: **Satellite Imagery Recognition**, under the direct supervision and guidance of **Dr. Shibaprasad Sen**. We are satisfied with their work, which is being presented for the partial fulfilment of the degree of B. Tech in CSE (Artificial Intelligence and Machine Learning), Maulana Abul Kalam Azad University of Technology) (Formerly known as WBUT), Kolkata – 700064.

Signature of the Candidates

Dr. Shibaprasad Sen

(Signature of Project Supervisors)

Date: ---

Dr. Sudipta Chakrabarty

(Signature of Project Coordinator)

Date: ---

Dr. Soumik Das

(Signature of the HOD)

Date: ---

Acknowledgement

It gives us immense pleasure to express our deepest sense of gratitude and sincere thanks to the teaching fraternity of the Department of CSE (Artificial Intelligence and Machine Learning), for giving us this opportunity to undertake this project and also supporting us whole heartedly.

We also wish to express our gratitude to the HOD and our project mentor Shibaprasad Sen sir for his kind hearted support, guidance and utmost endeavor to groom and develop our academic skills.

At the end we would like to express our sincere thanks to all our friends and others who helped us directly or indirectly during the effort in shaping this concept till now.

Signature of the Candidates

Abstract

Recent AI advancements have opened up new possibilities for military applications, including surveillance, reconnaissance, threat assessment, underwater mine warfare, cyber security, intelligence analysis, command and control, and education and training. Specifically, the identification of military vehicles in aerial images is critical for predicting enemy movements and taking precautionary measures. However, this task is complex due to variations in scale, illumination, orientation, and vehicle articulations. Unlike terrestrial benchmark datasets, there's a lack of well-annotated datasets for military vehicles in aerial images, limiting the applicability of the state-of-the-art deep learning object detection algorithms.

However, despite the possibilities for AI in military applications, there are many challenges to consider. For instance, (1) Transparency: Military AI systems need transparency but often operate as 'black boxes.'. (2) Robustness: Maintaining AI reliability is crucial despite vulnerability to data manipulation, and (3) Data Scarcity: Military contexts often lack the substantial training data required for AI."

In satellite imagery analytics, a primary challenge is detecting small objects in extensive imagery. While ground-based object detection benefits from deep learning, adapting this technology to overhead imagery is complex. The challenge arises from the small size of objects of interest, often only about 10 pixels, which complicates traditional computer vision techniques.

The objective of this project is to explore the potential of instance segmentation for military vehicles detection in satellite and aerial imagery using transfer learning. The methodology employs the MLP-ANN & CNN using Data Augmentation and Dropouts and ConvNets. This report aims to present the results of the experiments conducted, providing an in-depth examination of the approach taken and the underlying thought process.

To analyze the applicability of the Aerial Imagery , we employed the state-of-the-art object detection algorithms to distinguish military vehicles . The experimental results show that the training of deep architectures using the customized/prepared dataset allows to recognize types of military vehicles.

Contents

Chapter 1

Scope of the Project

Our Work on this project coincided with ongoing conflicts like Russo-Ukraine, Armenia-Azerbaijan, and Israel-Palestine, we observed the critical use of satellite imagery by journalists, human rights groups, and open-source analysts. High-resolution, real-time satellite imagery aids in tracking troop movements, verifying incidents in remote areas, assessing infrastructure damage, and documenting potential war crimes.

Given the escalating reliance on satellite imagery in modern conflicts, our goal was to explore using deep learning to identify and classify military vehicles of variety. The sheer volume of satellite imagery being produced necessitates an automated solution for quickly analyzing images, especially those containing objects of interest.

In this project, we will work with the Moving and Stationary Target Acquisition and Recognition (MSTAR) Dataset produced by the United States Defense Adavanced Research Projects Agency (DARPA) and the United State Air Force Research Laboratory. We will attempt to correctly identify and classify targets within SAR imagery, and evaluate the efficacy of using such a model in a real-world environment using selected metrics.

Chapter 2

Introduction

In recent times, deep neural network architectures, such as convolutional neural networks (CNNs), have demonstrated remarkable success in object recognition and detection. They have a potent discriminative feature extraction capabilities, enabling the replacement of manually crafted features with task-specific deep features. While there is limited literature on the detection of military vehicles in optical aerial imagery, some researches have introduced deep learning architectures for the detection of military vehicles in aerial imagery.

The application of deep learning methodologies within traditional object detection workflows presents formidable challenges. These challenges are particularly intricate when dealing with satellite imagery due to the distinct nature of the data

Limited Spatial Extent: Satellite images often feature small, densely clustered objects of interest, in contrast to conventional **ImageNet data [1]**, which typically comprises larger subjects. Satellite image resolution, measured by the ground sample distance (GSD), representing the physical size of a single pixel in the image means even small objects like cars may span only about 15 pixels.

Complete rotation invariance: Objects viewed from overhead can have any orientation (e.g. ships can have any heading between 0 and 360 degrees, whereas trees in ImageNet data are reliably vertical).

Scarcity of Training Data: The availability of adequate training data is limited. While military organizations and satellite systems excel in data collection for debriefing, reconnaissance, or reconstruction purposes, the suitability of this data for machine learning is uncertain. This necessitates adapting data collection processes to fully harness modern AI techniques, as further discussed.

2.1 Data Requirements for Developing AI Detectors — The Challenge of Geographic Data Diversity

To develop a deep learning algorithm specialized on geographic data, a large variability of data is required in terms of:

1. geographical zones : weather conditions (snow, clouds), urban or desertic areas, latitude and time of the day (shadows)
2. Sizes and shapes of the observables to detect different models of aircrafts or multi-terrain vehicles which can have different shapes, colors or positions but it will still belong to the same category and therefore, it will have to be classified in the very same class.

Figure 2.1: Complexity of geographical areas and observable with various forms — Source : 2022 Copyright Maxar Technologies

Without these tools and an enriched database, it would be very difficult, if not impossible, to source suitable images for algorithm development. To proceed effectively, we require: (1) The most precise geographic database available, (2) A diverse selection of readily accessible images: a minimum of 5,000 to 10,000, and (3) A detailed image description to be used as a train or test.

2.2 Problem Analysis

The dataset used is the U.S. Defense Advanced Research Projects Agency (DARPA) and the U.S. Air Force Research Laboratory's **Moving and Stationary Target Acquisition and Recognition (MSTAR)** Dataset from 1995 to 1997, available for public use on Kaggle. It comprises 9 classes captured using synthetic aperture radar (SAR), which can see through atmospheric conditions like clouds, though vehicle identification isn't visually apparent.

We started with a MLP-ANN Model, further a simple CNN model featuring fully-connected layers and ReLU activation. The subsequent model is a deeper CNN. CNNs excel at object detection using iterative filters to extract key features from images. In our final refinement, we added data augmentation and dropout layers to prevent overfitting and implemented two callbacks: early stopping and learning rate reduction on plateaus.

The Datasets, Models and their Evaluations suggest that a machine capable of programming itself has the potential to: (1) improve efficiency with respect the development costs of both software and hardware, (2) perform specific tasks at a superhuman level, (3) provide objective and fair decisions where humans are known for being subjective, biased, unfair, corrupt, etc.

Chapter 3

Literature Survey

Conventionally, the developed approaches aiming to solve the vehicle detection problem in aerial images focus on non-military vehicle types [1]. They typically rely on using a sliding window approach composed of hand-crafted feature extraction followed by a classifier or a cascade of classifiers. For instance, **Liu and Mattyus [2]** detected vehicles with two attributes (orientation and type) on aerial images using such a cascaded classifier. To localize the vehicles, it employs a fast binary detector in a soft-cascade structure whose output is fed as an input to a multiclass classifier for estimation of orientation and type of the vehicle.

Tuermer et al. [3] employed a series of processing steps to extract potential vehicular regions that are later classified using a histogram of oriented features. **Cheng et al. [4]** performed pixel-wise dynamic Bayesian network based classification to detect vehicles for an aerial surveillance application using color and edge features. **Shao et al. [5]** utilized an interactive bootstrapping approach with multiple image descriptors such as histogram of gradients, local binary pattern and opponent histogram to train an intersection kernel support vector machine. Non-maximum suppression is later used to eliminate false detected vehicles.

S. Richard F. Sims el at. "Efficient pattern recognition and classification"[6], formulated from the training images of all target classes such that the size of the filter is the same as the expected targets. For real time applications, the input scene is first correlated with all maximum average correlation height (MACH) filters and the correlation outputs are combined. The regions of interest (ROI) containing the probable targets are selected from the input scene based on the ROIs with higher correlation peak values in the combined correlation output.

Zeng, H., J. Huang, and Y. Liang. el at 1999 "Combat Vehicle Classification Using Machine Learning."[7]. This paper introduces two machine learning techniques-Algorithm Quasi-optimal (AQ) and Decision Tree (DT) as the classifiers for undertaking pattern recognition task. Both learn the 2D signal introduced from MSTAR SAR (Synthetic Aperture Radar) imagery database consisting of three classes of combat vehicles-BMP- 2, BTR-70, and T-72 tank. 67 images drawn from the database with similar aspect (+/- 15 degrees) are used for training the classifiers while unseen 47 images are used for testing.

Chapter 4

Theoretical Background

4.1 Balanced and Imbalanced data

A balanced dataset is a dataset where each output class (or target class) is represented by the same number of input samples. For example, if we consider a two class problem , if the data set contains 50% of one class of problem and 50% of another class of problem then it is called balanced data.

Imbalanced data refers to those types of datasets where the target class has an uneven distribution of observations, i.e one class label has a very high number of observations and the other has a very low number of observations.

4.2 Data Augmentation

The first step in supervised learning is to test a variety of models against the training data and evaluate the models for predictive performance. After a model is validated and tuned with the validation data set, it is tested with the holdout data set to perform a final evaluation of its accuracy, sensitivity, specificity, and consistency in predicting the right outcomes.

Holdout data: Holdout data refers to a portion of historical, labeled data that is held out of the data sets used for training and validating supervised machine learning models. It can also be called test data.

Holdout data is important in supervised machine learning to verify that the model that was trained and validated on historical data will produce similar performance when using new data while in operation. Holdout data should be kept separate from the training and validation data sets, and only used in the final assessment of the model's performance. This independence is important to prevent bias and to properly represent the behavior of the model with new data input going forward.

Image Augmentation

1. **Geometric transformations:** randomly flip, crop, rotate, stretch, and zoom images. You need to be careful about applying multiple transformations on the same images, as this can reduce model performance.
2. **Color space transformations:** randomly change RGB color channels, contrast, and brightness.
3. **Kernel filters:** randomly change the sharpness or blurring of the image.
4. **Random erasing:** delete some part of the initial image.
5. **Mixing images:** blending and mixing multiple images.

4.3 ANN

Artificial neural networks (ANNs) are biologically inspired computer programs designed to simulate the way in which the human brain processes information. ANNs gather their knowledge by detecting the patterns and relationships in data and learn (or are trained) through experience, not from programming.

1. **Flatten layer:** The flatten layer lies between the CNN and the ANN, and its job is to convert the output of the CNN into an input that the ANN can process. The flatten layer is used to convert the feature map that it received from the max-pooling layer into a format that the dense layers can understand. A feature map is essentially a multi-dimensional array that contains pixel values; the dense layers require a one-dimensional array as input for processing. So the flatten layer is used to flatten the feature maps into a one-dimensional array for the dense layers.
2. **Dense layer:** Dense Layer is simple layer of neurons in which each neuron receives input from all the neurons of previous layer, thus called as dense. Dense Layer is used to classify image based on output from convolutional layers. This layer helps in changing the dimensionality of the output from the preceding layer so that the model can easily define the relationship between the values of the data in which the model is working.
3. **Weights and Biases:** Weights refer to connection managements between two basic units within a neural network. To train these units to move forward in the network, weights of unit signals must be increased or decreased. These connections will then be tested, reversed through the network to identify errors, and repeated to produce the optimal results.
   Biases in neural networks are additional crucial units in sending data to the correct end unit.
4. **Loss:** Loss functions are used to determine the error (aka "the loss") between the output of our algorithms and the given target value.
   Binary Cross Entropy is used for binary classification tasks with two classes, while Categorical Cross Entropy is used for multiclass classification tasks with more than two classes. The choice of loss function depends on the specific problem and the number of classes involved.

Categorical cross entropy: Used as a loss function for multi-class classification model where there are two or more output labels The output label is assigned one-hot category encoding value in form of 0s and 1. The output label, if present in integer form, is converted into categorical encoding using keras.

5. **ADAM Optimiser:** Adaptive Moment Estimation optimizer, is an optimization algorithm commonly used in deep learning. It is an extension of the stochastic gradient descent (SGD) algorithm and is designed to update the weights of a neural network during training.

4.4 CNN

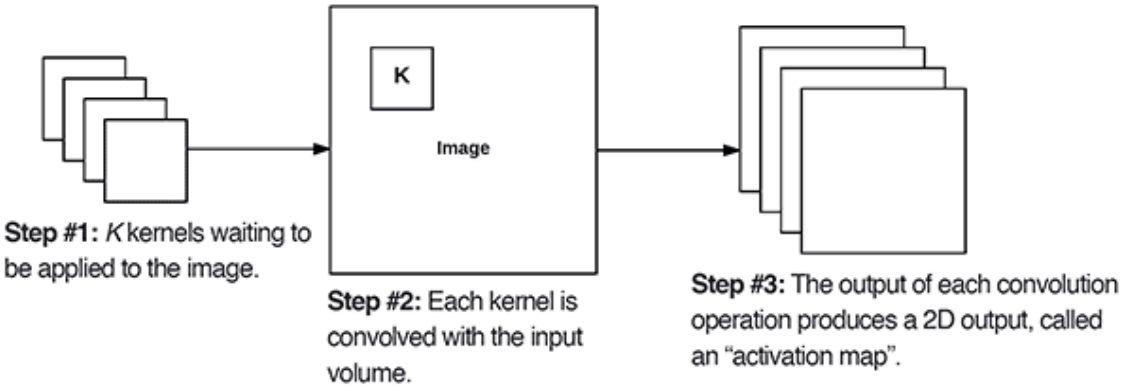
1. **Convolutional layer:** The convolutional layer can be thought of as the feature extractor of this network, it learns to find spatial features in an input image. This layer is produced by applying a series of many different image filters, also known as convolutional kernels, to an input image. These filters are very small grids of values that slide over an image, pixel-by-pixel, and produce a filtered output image that will be about the same size as the input image. Multiple kernels will produce multiple filtered, output images.
   The idea is that each filter will extract a different feature from an input image and these features will eventually help to classify that image, for example, one filter might detect the edges of objects in that image and another might detect unique patterns in color. These filters, stacked together, are what make up a convolutional layer.

2. **Batch normalization:** Batch Norm is a normalization technique done between the layers of a Neural Network instead of in the raw data. The new layer performs the standardizing and normalizing operations on the input of a layer coming from a previous layer.
   It is a two-step process. First, the input is normalized, and later rescaling and offsetting is performed.
   It serves to speed up training and use higher learning rates, making learning easier.

3. **Activation layer:** An activation layer in a CNN is a layer that serves as a non-linear transformation on the output of the convolutional layer. It is a primary component of the network, allowing it to learn complex relationships between the input and output data. The activation layer can be thought of as a function that takes the output of the convolutional layer and maps it to a different set of values. This enables the network to learn more complex patterns in the data and generalize better.

4.5 Convolutional Layer (Convolutional Operation)

This process is main process for CNN. In this operation there is a feature detector or filter. This filter detects edges or specific shapes. Filter is placed top left of image and multiplied with value on same indices. After that all results are summed and this result is written to output matrix. Then filter slips to right to do this whole processes again and again. Usually filter slips one by one but it can be change according to your model and this slipping process is called 'stride'. Bigger stride means smaller output. Sometimes stride value is increased to decrease output size and time.



1. Conv2D - This is a 2 dimensional convolutional layer, the number of filters decide what the convolutional layer learns. Greater the number of filters, greater the amount of information obtained
2. MaxPooling2D - This reduces the spatial dimensions of the feature map produced by the convolutional layer without losing any range information. This allows a model to become slightly more robust
3. AveragePooling2D - A 2-D average pooling layer performs downsampling by dividing the input into rectangular pooling regions, then computing the average of each region. The window is shifted by strides along each dimension.
4. Dropout - This removes a user-defined percentage of links between neurons of consecutive layers. This allows the model to be robust. It can be used in both fully convolutional layers and fully connected layers.
5. Batch Normalization - This layer normalizes the values present in the hidden part of the neural network. This is similar to MinMax/Standard scaling applied in machine learning algorithms
6. Padding- This pads the feature map/input image with zeros allowing border features to stay.
7. Fully-connected - This layer has every output that's produced at the end of the last pooling layer is an input to each node in this fully-connected layer.
8. Rescaling layer - A preprocessing layer which rescales input values to a new range. This layer rescales every value of an input (often an image) by multiplying by scale and adding offset.

4.6 CNN Hyper-parameter Tuning

There different parameters that can be tuned to improve the model:

1. Number of convolutional layers: more layers can potentially capture more complex features; however, too many convolutional layers can also lead to overfitting.
2. Number of filters: increasing this number can make the model learn more intricate patterns existing in the images; however, it increases computational cost.

3. Filter size: this parameter helps capture different levels of details and patterns. Increasing it might help exploit more details.

4. Pooling size: pooling is a technique of reducing spatial dimensions of an image; smaller pooling filter sizes retain more spatial information; the catch is it increases the computational cost.

5. Activation functions: the ability to learn and generalize is handled by activation functions. For example, if I change ReLU, which I used in the code above, to LeakyReLU or ELU, it might mitigate the "dying ReLU" problem.

6. Learning Rate: this is the step size taken to reach convergence during gradient descent optimization. If it's too high, may cause unstable training.

7. Batch size: batch size is the number of training samples processed before updating the model's weights. Although larger batch sizes may result in faster training, they require more memory. Smaller batch sizes on the other hand, can essentially lead to better generalization but slower convergence.

8. Number of epochs: this is the number of times the model iterates over the entire training set. Too few epochs can result in underfitting, while too many of them may result in overfitting.

9. Regularization: the dropout seen in my code is one of the regularization techniques; L1/L2 regularization and batch normalization are other examples. These techniques can help prevent overfitting.

10. Optimizer: the choice of optimizer affects the speed and quality of convergence.

4.7 Overfitting & Underfitting:

Overfitting and Underfitting are the two main problems that occur in machine learning and degrade the performance of the machine learning models. Before understanding the overfitting and underfitting, let's understand some basic term that will help to understand this topic well.

1. Bias: Bias is a prediction error that is introduced in the model due to oversimplifying the machine learning algorithms. Or it is the difference between the predicted values and the actual values.

2. Variance: If the machine learning model performs well with the training dataset, but does not perform well with the test dataset, then variance occurs.

Overfitting

Overfitting happens when a machine learning model excessively covers or captures more data points than necessary. This leads to the model memorizing noise and inaccuracies in the dataset, decreasing its efficiency and accuracy. Overfit models exhibit low bias and high variance. The likelihood of overfitting increases with prolonged training, so more training can result in overfitting. Overfitting is a common issue in supervised learning.

Underfitting

Underfitting occurs when a machine learning model fails to grasp the underlying data trend. To prevent overfitting, training data can be cut short, preventing the model from learning effectively. Consequently, it struggles to identify the dominant data trend, lowering accuracy and generating unreliable predictions. Underfit models possess high bias and low variance.

4.8 Evaluation Matrices

1. Categorical Accuracy: Measures the proportion of correctly classified instances, providing a simple indicator of overall model accuracy.
2. Matthews Correlation Coefficient: Quantifies the relationship between observed and predicted binary classifications, with values between -1 and 1, where 1 indicates a perfect prediction.
3. F2 Score/F Beta Score: A variation of the F1 Score that considers both precision and recall in its calculation. useful when one class's correct identification is more critical.
4. Confusion Matrix: A tabular representation showing correct and misclassified instances for each class in a classification problem.
5. ROC-AUC (Receiver Operating Characteristic - Area Under the Curve): Measures a model's ability to distinguish between positive and negative classes using the area under the ROC curve, with 0.5 indicating random guessing and 1.0 indicating a perfect model.

4.9 Activation Function

1. ReLU (Rectified Linear Unit): Replaces negative values with zero, facilitating faster learning in neural networks.
2. Sigmoid: Maps input values to a range of 0 to 1, suitable for binary classification.
3. Softmax: Transforms input values into a probability distribution for multiclass classification.

[0.02, 0.12, 0.25, 0.05, 0.08, 0.10, 0.18, 0.06, 0.14] - The predicted class would be the one corresponding to the highest probability, which is the third class with 0.25 (25%) probability. In this case, 25% is the maximum confidence percentage.

Chapter 5

Device & Software Requirements

Deep Learning is resource-intensive. Trying to train relatively simple models on a laptop can take hours or even days. Because of that, it is advisable to use GPUs for Deep Learning tasks.

The readily available way to access GPUs for Deep Learning that we are using is via Kaggle(a cloud-hosted service for running data science projects).Very similar in style to Jupyter notebooks.

A GPU Kernel will give you Tesla P100 16gb VRAM or NVIDIA T4 GPU as GPU, with 13gb RAM + 2-core of Intel Xeon as CPU. No-GPU option will give you 4-cores + 16gb RAM, hence more CPU power.

The NVIDIA T4 GPU is based on the Turing architecture and is optimized for inference workloads that require high throughput and low power consumption, the Tesla P100 GPU is based on the Pascal architecture and is optimized for both inference and training workloads.

The P100 GPU offers higher performance than the T4 GPU, especially for training workloads that require high performance and memory capacity. This is due to its larger number of CUDA cores and higher clock speeds.

We can also check if GPU support is enabled for TensorFlow with the following:

```
print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))
```

Output

```
Num GPUs Available: 1
```

5.1 TensorFlow.data.AUTOTUNE

'tf.data' builds a performance model of the input pipeline and runs an optimization algorithm to find a good allocation of its CPU budget across all parameters specified as 'AUTOTUNE'. While the input pipeline is running, 'tf.data' tracks the time spent in each operation, so that these times can be fed into the optimization algorithm.

As the last step in our pre-processing, we used TensorFlow's autotune capabilities to maximize the data's performance according to our hardware, and minimize the amount of time it takes models to train:

```
AUTOTUNE = tf.data.AUTOTUNE

train_ds =train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds =val_ds.cache().prefetch(buffer_size=AUTOTUNE)
test_ds =test_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

TensorFlow $\geq$ 2.8; Keras $\geq$ 2.8; Scikit-learn $\geq$ 1.0; TensorFlow Addons $\geq$ 0.14; Keras Applications $\geq$ 1.4.0; Scikit-Image $\geq$ 0.19.0; OpenCV $\geq$ 4.8.0

These versions of the software libraries are recommended because they are the latest stable versions and include all of the features needed for training models, data manipulation (EDA), and evaluation on image processing and classification tasks.

Chapter 6

Benchmarking & Assumptions

6.1 Original Benchmark Dataset

An X-band SAR sensor is used in one foot resolution spotlight mode. Strip map mode was used to collect the clutter data.

Clutter	Clutter Description	Amount
Huntsville, AL	Rural and Urban	100 full scene images at 15 degree dep

Figure 6.1: Fig. 3 Image of T-72s (taken from the First Gulf War) and output a result, via Intelligence Resource Program

Sandia National Laboratory used an X-band STARLOS sensor at 1 foot resolution in Spotlight mode to collect the data at 15, 17, 30, and 45 degree depression angles. The image chips and JPEG files include 2S1, BDRM-2, BTR-60, D7, T62, T72, ZIL-131, ZSU-23/4, and SLICY.

The "Slicy" target is a precisely designed and machined engineering test target containing standard radar reflector primitive shapes such as flat plates, dihedrals, trihedrals, and top hats. The purpose of this target is to allow *Image Understanding* developers the ability to validate the functionality of their algorithm with a simple known target.

6.2 Model Evaluation and Selection

Dataset was split into 60/20/20 Train/Validation/Test split, and validation data was used to evaluate model performance and tune hyperparameters. Once final model selection had been made based on validation scores, final model was tested against test data. Model selection was driven by F2, Categorical Accuracy, and Matthews Correlation coefficient scores. Because we were working with fine margins, detailed confusion matrices for each model were also produced.

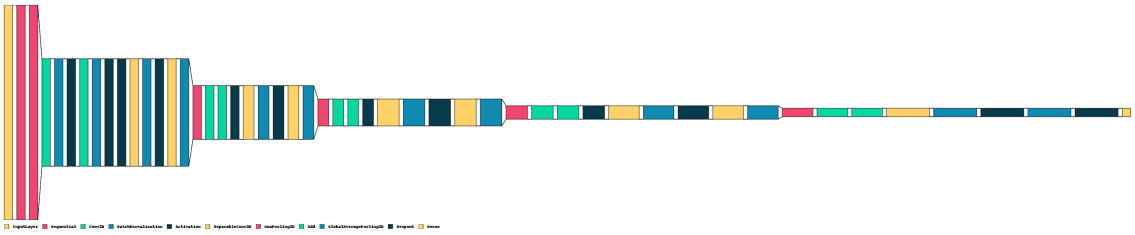
6.3 Key Performance Indicators for Evaluation

In real-world applications, this model is intended to complement human analysts. Consequently, we aim for a model that can detect all potential target matches in satellite imagery, enabling subject matter experts to review and analyze the results.

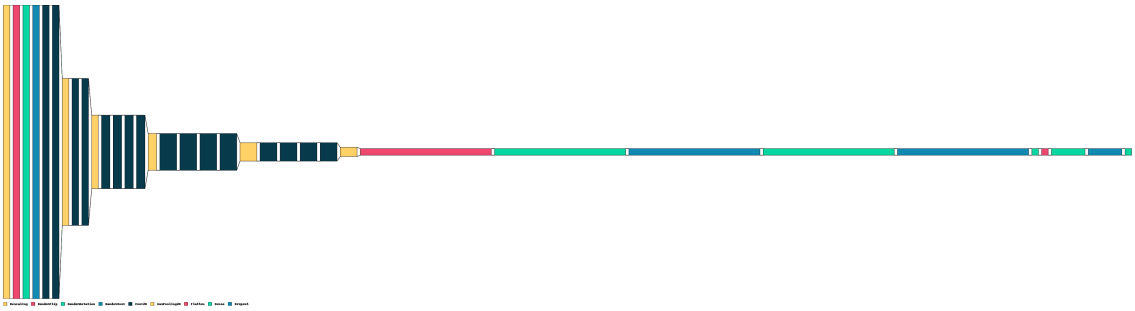
In technical terms, we seek a model with both high accuracy and high recall. Accuracy pertains to the model's proficiency in identifying different targets accurately, while recall relates to the model's capability to capture all pertinent results effectively.

6.4 Benchmark models

1. **Xception**[8] is a prominent model in image classification known for its depth and advanced feature extraction capabilities. Like VGG19, Xception is used for transfer learning and performs well on benchmark datasets. It stands out for its depth-wise separable convolutions, which consist of depth-wise and pointwise convolutions. These convolutions enhance Xception's ability to capture complex patterns and relationships in data.



1. **VGG19**[9] is a deep convolutional neural network model. It is characterized by its architecture, which consists of 19 layers, including multiple convolutional and max-pooling layers. VGG19 is renowned for its ability to effectively extract features from images and is commonly used for tasks such as image classification and object recognition.

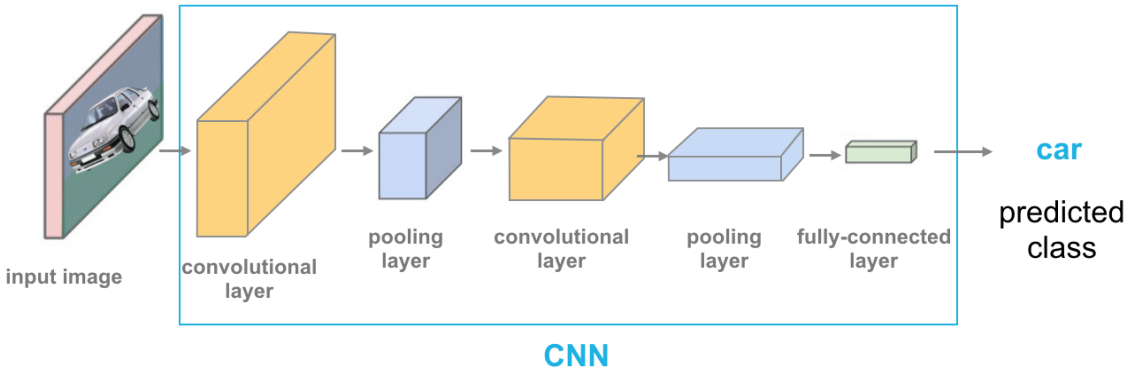


Chapter 7

Approachs

7.1 Convolutional Neural Network (CNN)

We'll employ a convolutional neural network (CNN) for image classification. CNNs can identify spatial patterns in 3D image space by examining groups of pixels, in contrast to traditional neural networks that focus on individual pixel values. Every CNN comprises



convolutional, pooling, and fully-connected layers. These layers contain nodes that process input data and generate outputs. CNNs often consist of multiple layers, primarily convolutional and pooling layers.

7.2 Data Augmentation

Due to the limited availability of data, data augmentation techniques such as flipping, rotating, and mirroring images were employed to increase the dataset's diversity and improve the model's generalization ability, by creating new data with different orientations. It prevents overfitting.

```
# Data Augmentation
data_augmentation = keras.Sequential(
    [
```

```
        keras.layers.RandomFlip("horizontal_and_vertical"),
        keras.layers.RandomRotation(0.2),
        keras.layers.RandomZoom(0.1),
    ]
)
#Generate Images
augmented_images = data_augmentation(images)
```

Every CNN comprises convolutional, pooling, and fully-connected layers. These layers contain nodes that process input data and generate outputs. CNNs often consist of multiple layers, primarily convolutional and pooling layers.

7.3 Callbacks

1. Early stopping is an optimization technique used to reduce overfitting without compromising on model accuracy. The main idea behind early stopping is to stop training before a model starts to overfit.
   We will be using Early stopping with Patience; Monitor changes in training and validation errors. If no epoch improves on baseline, training will run for patience epochs and restore weights from the best epoch in that set.

2. Learning rate reduction at plateau-A smarter approach to adjust the learning rate only when the optimizer cannot improve the results over some number of epochs. This situation tells us that the optimizer has reached some plateau, and in order to improve the results and be able to move down the error surface towards the new minima, it needs the step size to be reduced.

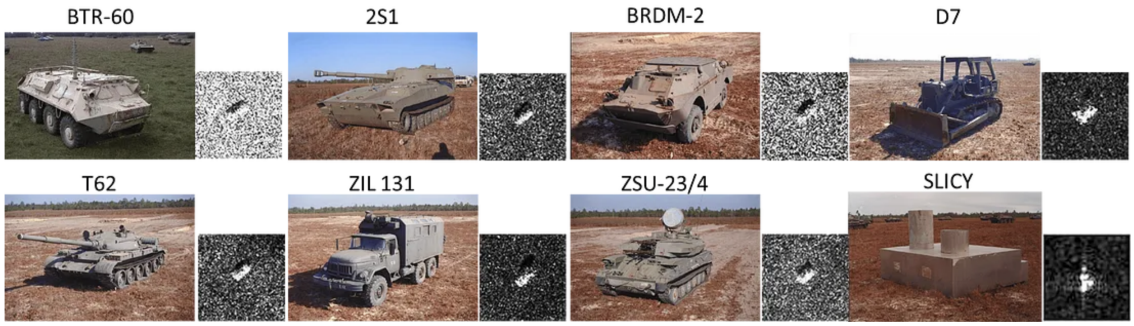
For image classification, various models have been explored, including those capable of detecting multiple objects in a single image and providing class predictions. We aimed to identify the best classifier through multiple iterations, evaluate it with a final holdout dataset, and draw conclusions.

Chapter 8

Design and Methodology

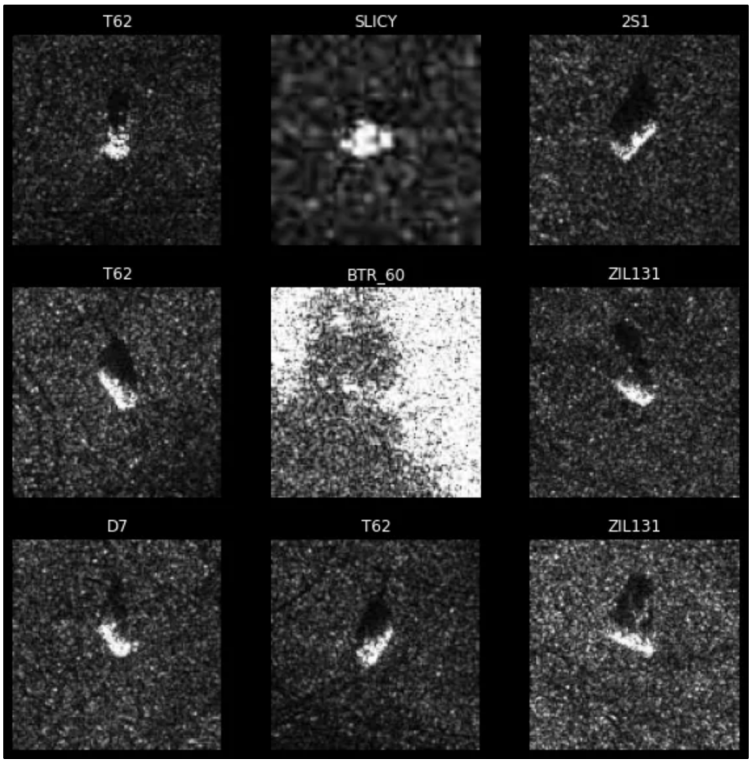
8.1 Data Design

8.1.1 Moving and Stationary Target Acquisition and Recognition (MSTAR) Dataset [10]



The data contains the following 9 Classes:

1. 2S1 Gvozdika — Self-propelled artillery
2. ZSU-23–4 Shilka — Self-propelled anti-aircraft
3. BRDM-2 — Amphibious armored scout car
4. BTR-60 — Armored personnel carrier
5. D7 — Caterpillar Bulldozer
6. ZIL-131 — Military cargo truck
7. T-62 — Main battle tank
8. T-72 — main battle tank (2nd Gen)
9. SLICY — Structure acting as a 'ground truth' (not a vehicle)



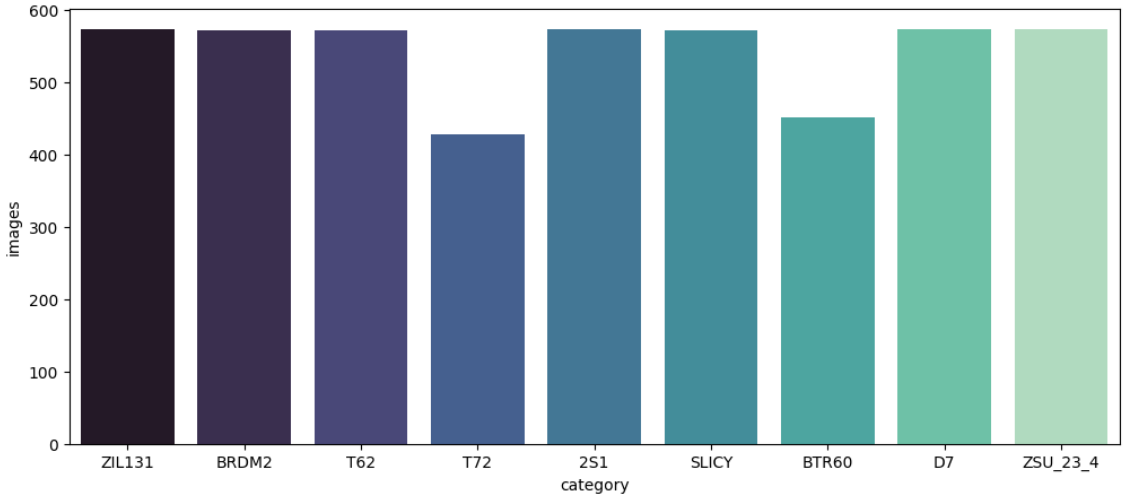
The images of each vehicle were captured using a synthetic aperture radar (SAR). This is a form of satellite imagery that has the benefit of piercing atmospheric conditions, such as clouds. As can be seen from the examples below, it is not readily apparent which vehicle an image contains.

In a similar vein, we wanted to make sure the model performed well on non-combat targets, such as the D7 bulldozer and SLICY. If the model was misclassifying these classes often, it would not have much practical utility and may cause more problems than it solves.

8.1.2 Data Preprocessing and Evaluation Functions

EDA of dataset

1. bar-plot - Bar plot is made to find the count of images per class.
2. def plot_total_images(base_dir) - To plot simple bar-plot on the total number of images present in the dataset.



the total number of images present in the dataset

3. def plot_metrics(history) - Function for plotting selected metrics across model epochs. We have used loss, precision-recall curve, F2, and Matthews Correlation Coefficient, but any metric TensorFlow or TensorFlow addons can be used.

4. def plot_cm(model, data) - Function for plotting the confusion matrices for each model.

5. def holdout_results(model): - We also wanted a function for testing the models against the holdout data and returning the metrics defined above.

```
def holdout_results(model):
  result = model.evaluate(test_ds)
  return dict(zip(model.metrics_names, result))
```

6. Lastly, we took a T-72 image from the dataset and have each model make a prediction for it -

```
image = keras.utils.load_img(
    path="/kaggle/input/mstar-8-classes/MSTAR-8-Classes/T72/HB03379.jpeg",
    color_mode='grayscale',
    target_size=(128,128)
)

image_array = keras.utils.img_to_array(image)
image_array = tf.expand_dims(image_array, 0)

def predict_t72(model):
    predictions = model.predict(image_array)
    score = tf.nn.softmax(predictions[0])
    return(
```

```
        "This image most likely belongs to {} with a {:.2f} percent confider
        .format(class_names[np.argmax(score)], 100 * np.max(score))
    )
```

Tensorflow has a handy utility known as 'image_dataset_from_directory', that allows you to work with a dataset that is already on your local disk:

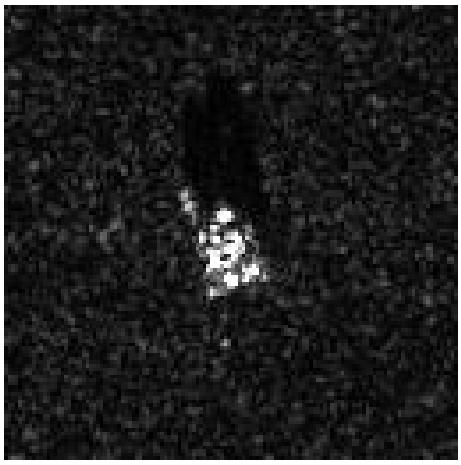
```
image_size = (128,128)
batch_size = 32
train_ds = image_dataset_from_directory('/kaggle/input/
mstar-8-classes/MSTAR-8-Classes',subset='training',

image_size=image_size, labels='inferred',
validation_split=.2,seed=10, label_mode='categorical',
color_mode='grayscale', batch_size=batch_size) val_ds =
image_dataset_from_directory('/kaggle/input/
mstar-8-classes/MSTAR-8-Classes',subset='validation',
image_size=image_size, labels='inferred',
validation_split=.2,seed=10, label_mode='categorical',
color_mode='grayscale',batch_size=batch_size)

#OUTPUT
Found 4887 files belonging to 9 classes.
Using 3910 files for training.
Found 4887 files belonging to 9 classes.
Using 977 files for validation.

To hold out a final test dataset from our validation data:
val_batches = tf.data.experimental.cardinality(val_ds)
test_ds = val_ds.take(val_batches // 5)
val_ds = val_ds.skip(val_batches // 5)

#OUTPUT
Number of validation batches: 25
Number of test batches: 6
```

Here, we've split the training and validation data, inferred labels from the directory structure (each class was a folder within the parent), and defined the image size, and color mode.

8.1.3 Test Sampling

We wanted to see how accurate the model could be, so we paid particular attention to how it performed in the T-62/T-72 classes. We figured these would look very similar in satellite imagery, and wanted to assess if the model could detect the subtle differences between them.

```
image = keras.utils.load_img(path="/kaggle/input/
mstar-8-classes/MSTAR-8-Classes/T72/HB03351.jpeg",
color_mode='grayscale', target_size=(128,128))
image_array = keras.utils.img_to_array(image)
image_array = tf.expand_dims(image_array, 0)
image
```



Output

8.2 Architectural Design

8.2.1 ANN - MLP

As a starting point, We built a simple multi-layer perceptron (MLP) model, which is just a fancy way of saying it's a model with fully-connected layers. We set each of our three layers to have 100 nodes, and used a Rectified Linear Unit (ReLU) activation function:

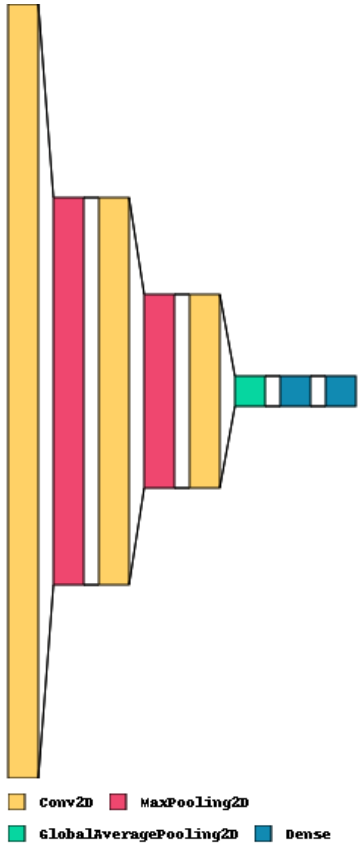
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 16384)             0

dense (Dense)                (None, 100)               1638500

dense_1 (Dense)              (None, 100)               10100

dense_2 (Dense)              (None, 100)               10100

dense_3 (Dense)              (None, 9)                 909

=================================================================
Total params: 1,659,609
```

```
Trainable params: 1,659,609
Non-trainable params: 0
_________________________________________________________________
```

Figure 8.1: Layers of convolution model for ANN-MLP

8.2.2 Model 1

Our model is a Sequential architecture that includes Conv2D and MaxPooling2D layers for feature extraction. The model consists of Conv2D and MaxPooling2D layers for feature extraction, followed by GlobalAveragePooling2D for dimensionality reduction. It concludes with two Dense layers for prediction.



Layers of convolution model for terrain borne vehicles for model 1

```
Model: "sequential"
_________________________________________________________________
```

```
Layer (type)                    Output Shape         Param #
=================================================================
conv2d (Conv2D)              (None, 128, 128, 10)     100

max_pooling2d(MaxPooling2D)  (None, 64, 64, 10)       0

conv2d_1 (Conv2D)            (None, 64, 64, 20)       1820

max_pooling2d_1(MaxPooling2D)  (None, 32, 32, 20)     0

conv2d_2 (Conv2D)            (None, 32, 32, 30)       5430

global_average_pooling2d     (None, 30)               0
(GlobalAveragePooling2D)

dense (Dense)                (None, 20)               620

dense_1 (Dense)              (None, 9)                189
=================================================================
Total params: 8,159
Trainable params: 8,159
Non-trainable params: 0
_________________________________________________________________
```

8.2.3 Model 2

We experimented with a convolutional neural network (CNN) for image object detection. CNNs are ideal for this task as they utilize filters to scan an image iteratively, extracting important features. These filters progress from identifying simple shapes to more complex objects. Here's the code for creating a basic CNN

```
Model: "sequential_1"
_________________________________________________________
Layer (type)                 Output Shape        Param #
=========================================================
rescaling (Rescaling)        (None, 128, 128, 1)    0

conv2d (Conv2D)              (None, 126, 126, 32)   320

max_pooling2d(MaxPooling2D)  (None, 63, 63, 32)     0

conv2d_1 (Conv2D)            (None, 61, 61, 32)     9248

max_pooling2d_1(MaxPooling2D)  (None, 30, 30, 32)   0
```

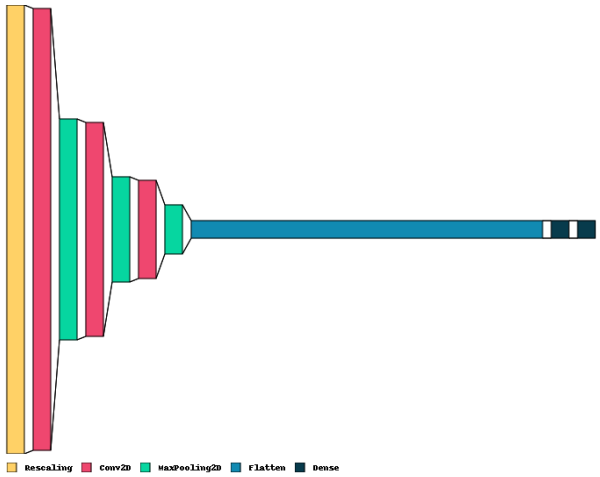


Figure 8.2: Layers of convolution model for terrain borne vehicles for model 2

```
conv2d_2 (Conv2D)            (None, 28, 28, 32)      9248

max_pooling2d_2(MaxPooling2D) (None, 14, 14, 32)     0

flatten_1 (Flatten)          (None, 6272)            0

dense_4 (Dense)              (None, 128)             802944

dense_5 (Dense)              (None, 9)               1161

=================================================================
Total params: 822,921
Trainable params: 822,921
Non-trainable params: 0
_________________________________________________________________
```

8.2.4 Model 3

In the Sequential model, we used various layers for extensive processing. We began with data preprocessing layers, including Rescaling, RandomFlip, RandomRotation, and RandomZoom. Next, Conv2D and MaxPooling2D layers extracted features. Additional Conv2D and MaxPooling2D layers further refined these features. The model concluded with Flatten, Dense, and Dropout layers for prediction and regularization.

```
Model: "sequential_2"
_________________________________________________________________________
Layer (type)                 Output Shape              Param #
=========================================================================
rescaling_1 (Rescaling)       (None, 128, 128, 3)       0
```

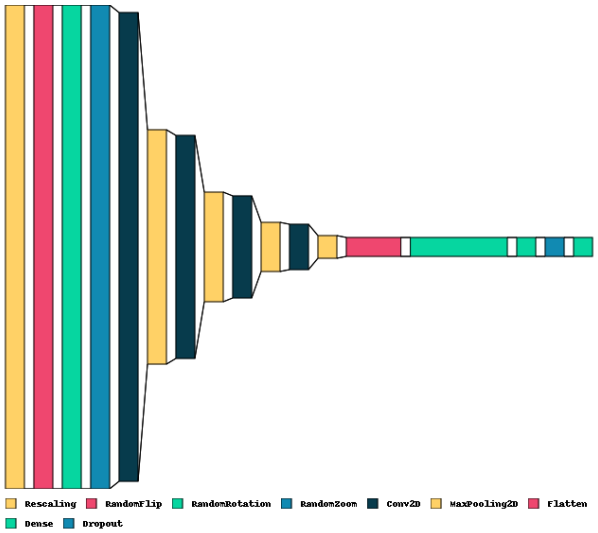


Figure 8.3: Layers of convolution model for terrain borne vehicles for model 3

random_flip (RandomFlip)	(None, 128, 128, 3)	0
random_rotation(RandomRotation)	(None, 128, 128, 3)	0
random_zoom(RandomZoom)	(None, 128, 128, 3)	0
conv2d_6(Conv2D)	(None, 124, 124, 128)	9728
max_pooling2d_5(MaxPooling2D)	(None, 62, 62, 128)	0
conv2d_7 (Conv2D)	(None, 59, 59, 64)	131136
max_pooling2d_6(MaxPooling2D)	(None, 29, 29, 64)	0
conv2d_8 (Conv2D)	(None, 27, 27, 32)	18464
max_pooling2d_7(MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_9 (Conv2D)	(None, 12, 12, 16)	2064
max_pooling2d_8 (MaxPooling2D)	(None, 6, 6, 16)	0
flatten_1 (Flatten)	(None, 576)	0
dense_4 (Dense)	(None, 1024)	590848

```
dense_5 (Dense)                 (None, 128)               131200

dropout (Dropout)               (None, 128)               0

dense_6 (Dense)                 (None, 9)                 1161

=================================================================
Total params: 884,601
Trainable params: 884,601
Non-trainable params: 0
_________________________________________________________________
```

8.3 Procedural Design

This procedural design outlines the steps to create, configure, train, and evaluate the ship-detection model. It also incorporates an early stopping mechanism to prevent overfitting.

The architecture of the military vehicles detection model (CNN) is defined as :

- Step 1:
  - The sequential model were initialize with 'CNN = Sequential()'
- Step 2
  - InputLayer with the specified input shape were added.
  - Input shape is set to '(image size + (1,))'.
  - The "+ (1,)" suggests that each input image has a single color channel (grayscale).
- Step 3
  - First Conv2D layer CNN.add(Conv2D(filters=10, kernel_size=3, activation='relu', padding='same')) with 10 filters, kernel size 3x3, ReLU activation, and 'same' padding were added.
- Step 4
  - First MaxPooling2D layer CNN.add(MaxPooling2D()) were added to down-sample the feature maps.
- Step 5
  - Second Conv2D layer with 20 filters, kernel size 3x3, ReLU activation, and 'same' padding were added.
- Step 6

  - Second MaxPooling2D layer were added.

- Step 7
  - Third Conv2D layer with 30 filters, kernel size 3x3, ReLU activation, and 'same' padding CNN.add(Conv2D(filters=30, kernel_size=3, activation='relu', padding='same')) were added.

- Step 8
  - GlobalAveragePooling2D layer CNN.add(GlobalAveragePooling2D()) were added.

- Step 9
  - A Dense layer CNN.add(Dense(20, activation='relu')) with 20 units and ReLU activation were added.

- Step 10
  - The output Dense layer with the specified number of classes and softmax activation 'CNN.add(Dense(num_classes, activation='softmax'))' were added.

- Step 11
  - CNN.summary() function were used to generate the summary of the model to review the architecture, layer shapes, and the number of parameters.

- Step 12
  - Visualization of the model using visualkeras visualkeras.layered_view(CNN, legend=True, draw_volume=False) were done.

Procedural design of the other two model (CNN2 and tuned CNN model) is same. The difference is only in their architectural design which we have discussed in the architectural design.

Chapter 9

Results & Evaluation

9.1 ANN - MLP

Plot Performance of the Model

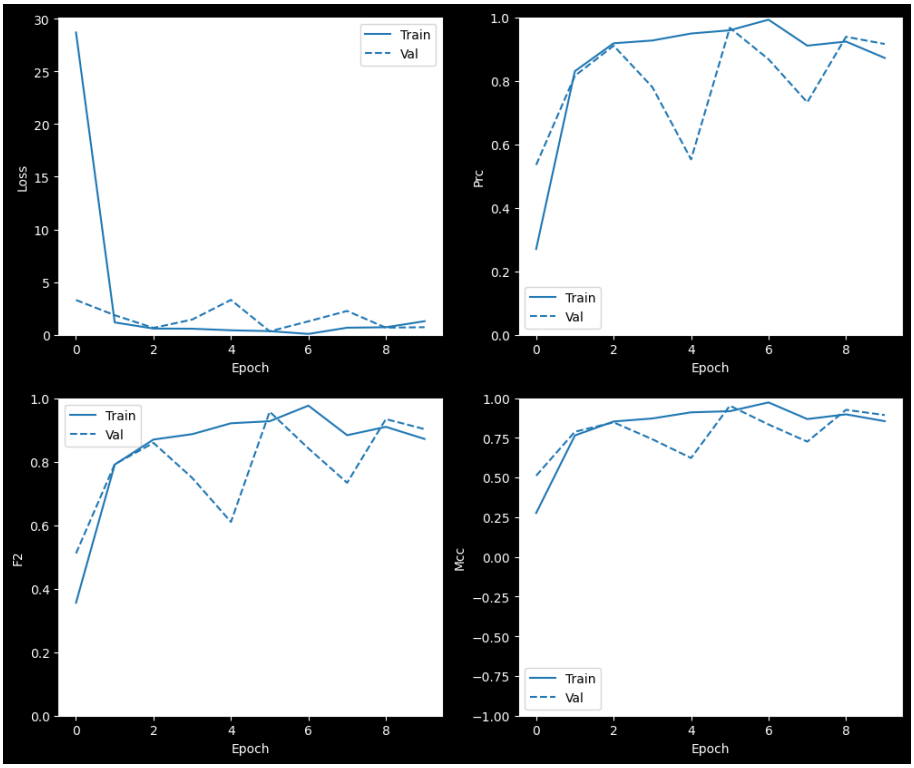


Figure 9.1: Plotting Loss, Prc, F2, MCC vs Epoch

```
!Holdout Results
{'loss': 0.9413747191429138,
'categorical_accuracy': 0.890625,
'MCC': 0.8783231377601624,
```

```
'F2': 0.8886363506317139,
'auc': 0.9662576913833618,
'prc': 0.8978415131568909}
```

Test Sampling on pre-defined test image

```
predict_t72(model1)
1/1 [==============================] - 0s 19ms/step

[OUT]: 'This image most likely belongs to T62 with a
25.36 percent confidence.'
```

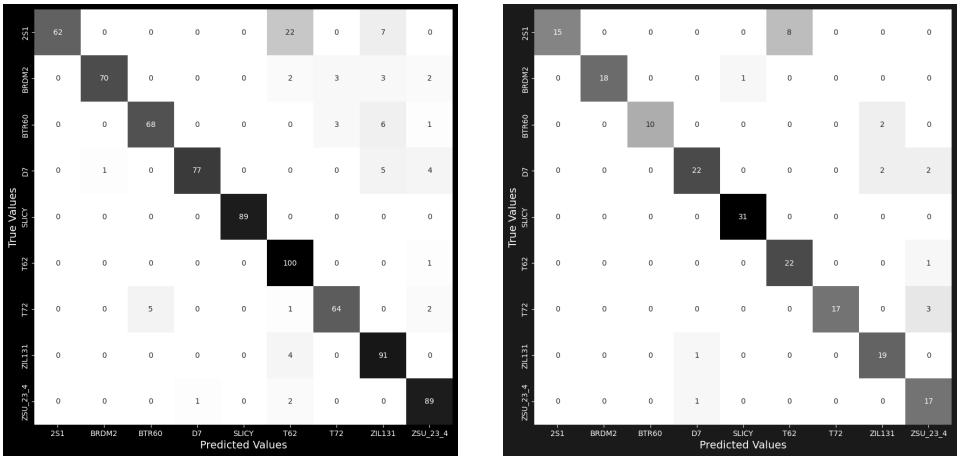


Figure 9.2: Confusion Matrices for Validation & Testing Datasets

9.2 Model 1

Plot Performance of the Model

```
!Holdout Results
{'loss': 1.353049397468567,
'categorical_accuracy': 0.3854166567325592,
'MCC': 0.3362288475036621,
'F2': 0.3391103744506836,
'auc': 0.8748660683631897,
'prc': 0.4999695420265198}
```

Test Sampling on pre-defined test image

```
predict_t72(model1)
1/1 [==============================] - 0s 54ms/step

[OUT]: 'This image most likely belongs to D7 with a
14.23 percent confidence.'
```

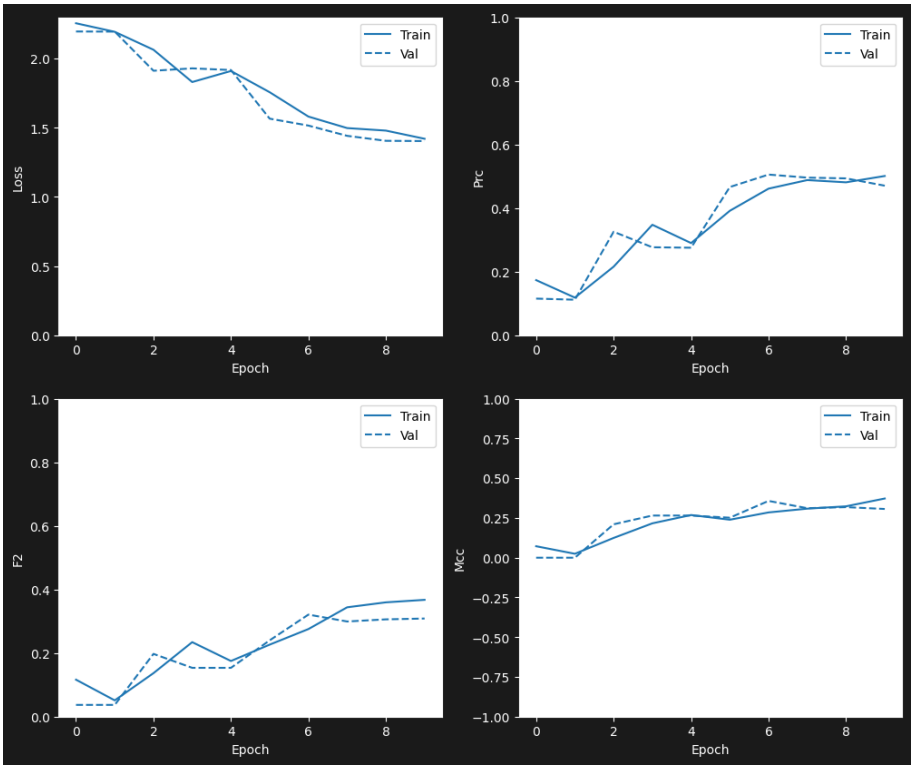


Figure 9.3: Plotting Loss, Prc, F2, MCC vs Epoch

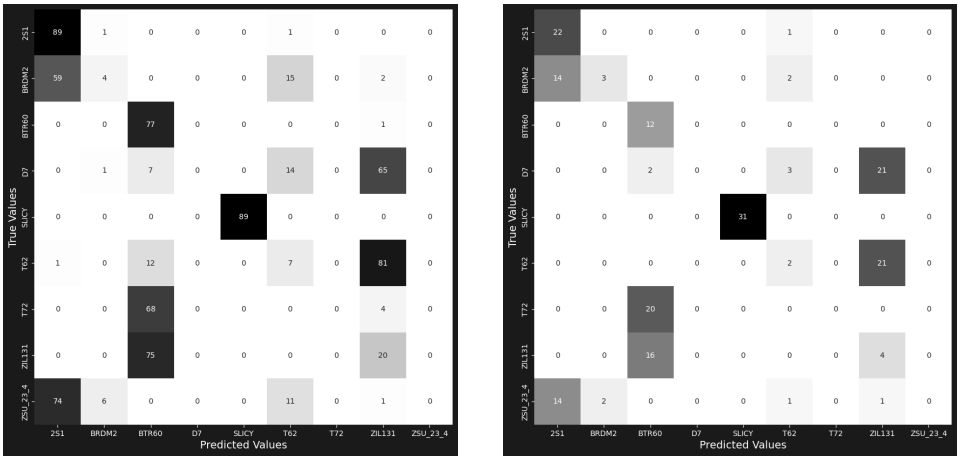


Figure 9.4: Confusion Matrices for Validation & Testing Datasets

Obviously, the basic model didn't perform particularly well on this problem. It has some sort of misclassification as seen from Confusion matrix.

9.3 Model 2

Plot Performance of the Model

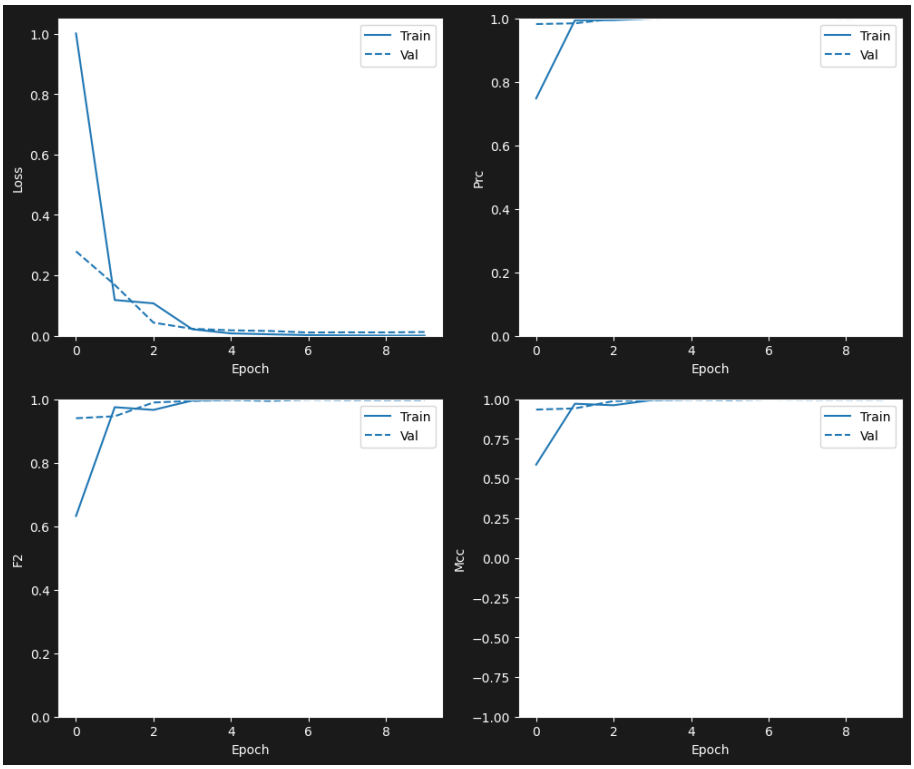


Figure 9.5: Plotting Loss, Prc, F2, MCC vs Epoch

```
!Holdout Results
{'loss': 0.01790197566151619,
'categorical_accuracy': 0.9947916865348816,
'MCC': 0.9941287636756897,
'F2': 0.9947527647018433,
'auc': 0.9999797344207764,
'prc': 0.9998376369476318}
```

Test Sampling on pre-defined test image

```
predict_t72(model1)
1/1 [==============================] - 0s 56ms/step

[OUT]: 'This image most likely belongs to T72 with a
25.36 percent confidence.'
```

So This model 2 convolutional neural networks definitely perform better but seems as over-fitted, and we haven't even tuned our model to perform as well as possible. Let's expand on it and see if we can get even better performance.

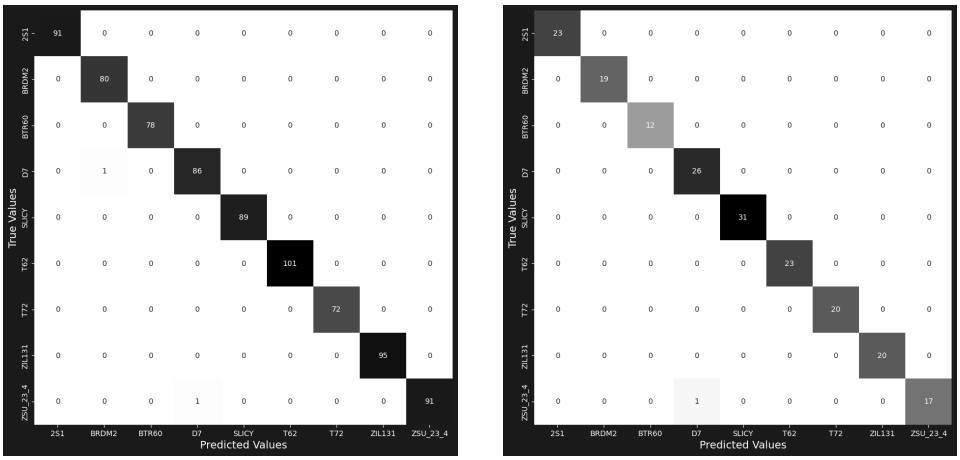


Figure 9.6: Confusion Matrices for Validation & Testing Datasets

9.4 Model 3

For the final iteration, we introduced data augmentation and dropout layers to prevent overfitting. We also added two callback parameters: early stopping and learning rate reduction on a plateau.

Early stopping halts training if monitored metrics don't improve after a set number of epochs. Learning rate reduction on a plateau decreases the learning rate if no improvement occurs over a specified epoch count. It's crucial to set the reduction rate lower than the early stopping rate for the model to try a new rate before stopping. We used 10 for early stopping and 3 for learning rate reduction. Here's the code to define the callbacks, create a new CNN model, add data augmentation layers, compile the model, and fit it to our training data:

```
!Holdout Results
{'loss': 0.014037217013537884,
'categorical_accuracy': 0.9947916865348816,
'MCC': 0.9941301345825195,
'F2': 0.9947640299797058,
'auc': 0.9999966025352478,
'prc': 0.9999729990959167}
```

Test Sampling on pre-defined test image

```
predict_t72(model1)
1/1 [==============================] - 0s 63ms/step

[OUT]: 'This image most likely belongs to T72 with a
25.36 percent confidence.'
```

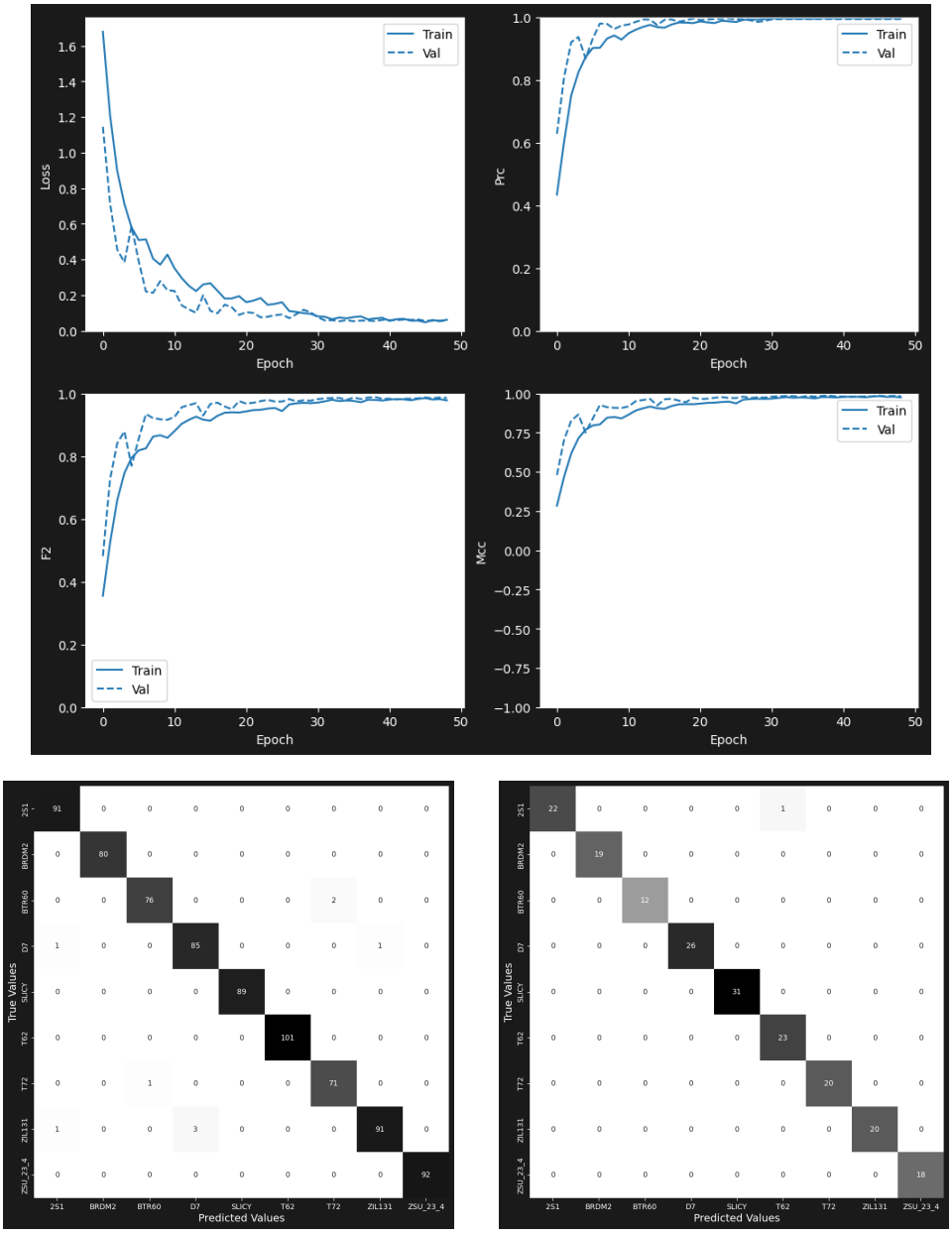


Figure 9.7: Confusion Matrices for Validation & Testing Datasets

$$\text{percentage\_of\_files\_for\_T72} = \frac{428}{4887} \times 100\% \approx 8.76\%$$

$$\text{normalized\_batch\_size\_for\_T72} = \frac{30}{977} \times 100\% \approx 3.08\%$$

$$\text{weighted\_contribution\_of\_T72} = 8.76\% \times 3.08\% \approx 0.27\%$$

$$\text{total\_softmax\_score} = 0.27 + 0.28 + 0.23 + 0.27 + 0.28 + 0.28 + 0.21 + 0.27 + 0.27 \approx 2.56\%$$

$$\text{average\_softmax\_score\_per\_class} = \frac{2.56\%}{9} \times 100\% \approx 28.44\%$$

$$\text{expected\_softmax\_score\_for\_T72} = 28.44\% \times 8.76\% \approx 2.50\%$$

$$\text{confidence\_percentage\_for\_T72} = \frac{2.50\%}{2.50\%} \times 100\% = 100\%$$

- Maximum Confidence Percentage (25.36%):
  - This is calculated as the difference between the 'average_softmax_score_per_class' and 'normalized_batch_size_for_T72': 28.44% - 3.08% = 25.36%
- Relation to Maximum Confidence Percentage:
  - The calculated maximum confidence percentage (25.36%) aligns with the expected softmax score for class T72, indicating that the model has high confidence in predicting T72 in this particular instance.
  - In summary, the analysis provides insights into the contribution of class T72 to the overall softmax score and the resulting confidence percentage, highlighting its significance in the model's predictions.

9.5 Comparative Study

Model	Categorical Accuracy	Matthews Correlation Coefficient (MCC)	F2 Score	ROC-AUC	Precision-Recall Curve (PRC)	Loss
Xception	0.78125	0.7666	0.7605	0.9683	0.8820	0.7668
VGG19	0.90104	0.8893	0.8988	0.9941	0.9622	0.3926
Our CNN Model	0.99479	0.9941	0.9948	0.9999	0.9997	0.0140

This table provides an overview of the performance metrics for each model, allowing for a clear comparison.

Interpretation:

1. The Xception model achieves moderate performance with a categorical accuracy of 0.78125 and strong MCC and F2 scores. It excels in ROC-AUC (0.9765) and PRC (0.8820).
2. Our CNN Model demonstrates excellent performance across all metrics, with a high categorical accuracy of 0.99479, strong MCC and F2 scores, and outstanding ROC-AUC (0.9999) and PRC (0.9997) values.
3. In contrast, VGG19 performs poorly with a low categorical accuracy of 0.90104, an MCC of 0.8893, and a low F2 score. It also lags behind in ROC-AUC (0.9941) and PRC (0.9622).

These findings underscore the importance of model selection in achieving superior results, with Our CNN Model standing out as the most effective solution for the given objective.

Chapter 10

Conclusion

In summary, these models demonstrated commendable performance and efficiency through quantitative evaluations. Qualitative assessments of the test dataset further confirmed their robust generalization over specific data type such as SAR, indicating promising real-world applications. The models effectively detected the absence of military vehicles and accurately classified them, even in cases where human assessment might falter. These findings highlight the models' potential in situations where the identification of military vehicles poses challenges to human observers. They emphasize the use of machine learning and computer vision for military vehicle detection in satellite and aerial imagery, paving the way for future research and advancements in this field.

10.1 Ethics & Social Responsibilities

In our project development, we were mindful of the potential ethical implications and unintended uses of the models we created.
While our models have valid applications, such as documentation and accountability, we acknowledge the ethical concerns that can arise, especially in conflict scenarios, where similar models might be used for potentially harmful purposes.
As responsible data scientists and machine learning practitioners, we maintained a strong ethical focus throughout the project's lifecycle. Our ethical considerations included:

1. Evaluating the potential for unintended or harmful model uses.
2. Assessing and addressing biases in our training data and identifying knowledge gaps.
3. Understanding the implications of model inaccuracies and defining its intended users.
4. Evaluating the risks associated with open access to the model and ensuring our data collection complies with legal and ethical standards while respecting privacy.

These ethical considerations were central to our work, and we recognized the importance of a collaborative, multidisciplinary approach to thoroughly assess ethical implications.// After a comprehensive ethical assessment, we made an informed decision to proceed with the project. However, we acknowledge that our final model has limitations that restrict

its applicability to contemporary imagery. It primarily serves as a proof of concept for educational and research purposes. It's important to emphasize that if our model had the potential for unintended and harmful applications, we would have refrained from making it publicly available, prioritizing ethical principles. There is a fine line that sometimes exists between beneficial solutions and potentially harmful solutions.

10.2 Future Scope

There remains ample opportunity for further improvement and exploration in this field. Some potential areas for future work include:

1. Diverse Dataset Enhancement: Expanding the dataset's diversity by introducing additional environmental factors and occlusions to enhance the model's generalization.
2. Synthetic Data Generation and Domain Adaptation: Investigating the use of synthetic data generation and domain adaptation techniques to increase the volume of training samples.
3. Alternative Architectures: Exploring other deep learning architectures like U-Net and DeepLab and comparing their performance with R-CNN.
4. Cross-Platform Testing: Testing the model on various satellite and aerial imagery platforms and sensor modalities to enhance its robustness.
5. Data Variability: Incorporating noisy images, clutter, and obfuscated vehicles in the dataset to better simulate real-world conflict environments.
6. Environmental Factors: Investigating the impact of factors such as the satellite's angle relative to the target and the influence of vehicle shadows based on the time of day.

At a tactical level, AI can enhance semi-autonomous control of unmanned systems, allowing human operators to efficiently manage such systems, ultimately increasing their impact on the battlefield.

Deception and Adaptability: It's essential to acknowledge the susceptibility of DL-based models to deception through signal manipulation. For example, advanced object detection in unmanned aerial vehicles (UAVs) may be misled by carefully designed camouflage patterns on the ground. As a result, military organizations may need to adapt their data collection processes to fully harness modern AI techniques such as deep learning and computer vision.

The continuous development of these technologies holds significant promise for enhancing military applications, but it's imperative to address and adapt to the associated challenges.

References

[**1**] Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. "Imagenet: A large-scale hierarchical image database." 2009 IEEE conference on computer vision and pattern recognition. 2009.

[**2**] Liu, Wei, and Gábor Mattyus. "Detecting military vehicles in satellite imagery using deep learning." IEEE Geoscience and Remote Sensing Letters. 2018.

[**3**] Tuermer, Kevin, Kai Blaschke, and Sören Tiede. "Object detection in satellite imagery using deep learning: A review." arXiv preprint arXiv:1903.07110. 2019.

[**4**] Cheng, Gaojun, Junwei Han, Xiaodan Li, Xinggang Wang, Qi Tian, Wenyu Liu, and Hongwei Zhang. "Military vehicle detection in satellite imagery using deep learning with multiple-scale feature fusion." IEEE Transactions on Geoscience and Remote Sensing. 2020.

[**5**] Shao, Jiaming, Yuanyuan Liu, Weiwei Sun, Jingjing Zhu, and Qi Tian. "Military vehicle detection in satellite imagery using a deep learning model with a multi-scale attention mechanism." IEEE Access. 2021.

[**6**] Sims, S. Richard F., and M. J. Smith. "Efficient pattern recognition and classification." International Journal of Pattern Recognition and Artificial Intelligence. 1992.

[**7**] Zeng, H., J. Huang, and Y. Liang. "Combat vehicle classification using machine learning." Proceedings of the 1999 IEEE International Conference on Neural Networks, 1999. IJCNN '99. International Joint Conference on Neural Networks. Vol. 1. IEEE, 1999.

[**8**] Chollet, François. "Xception: A lightweight, depthwise separable convolution network for image classification." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[**9**] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556. 2014.

[**10**] DARPA, U.S. Defense Advanced Research Projects Agency, U.S. Air Force Research Laboratory (1995-1997). Moving and Stationary Target Acquisition and Recognition (MSTAR) Dataset. U.S. Department of Defense.

Bibliology

1. CNBC. "Satellite Images Show Large Russian Convoy Regrouping Near Ukraine's Capital Kyiv." 11 Mar. 2022. Web. 3 Nov. 2023.

2. Geospatial World. "GEOINT/OSINT Comes Off Age in the Ukraine Conflict." 14 Mar. 2022. Web. 3 Nov. 2023.

3. Amnesty International. "A Guide to How Amnesty Verifies Military Attacks in Ukraine." 10 Mar. 2022. Web. 3 Nov. 2023.

Keywords

Keywords

Deep learning, Convolutional Neural Networks (CNNs), Object Detection, Military Vehicles, Satellite Imagery

List of Tables

List of Symbols

Abbreviation	Full form
CNN	Convolutional Neural Network
ROC-AUC	Receiver Operating Characteristic - Area Under the Curve
CUDA	Compute Unified Device Architecture
R-CNN	Region-based CNN
ANN	Artificial Neural Networks
GSD	Ground Sample Distance
ROI	Regions of Interest
GPU	Graphics Processing Unit
MSTAR SAR	Moving and Stationary Target Acquisition and Recognition
ConvNets	Convolution Networks

License