Principles of Machine Learning

MSML603

Final Project

Title: Smart Waste Management System Using Object Detection and Classification

Abstract:

This project aims to develop a smart waste management system leveraging advanced object detection and classification techniques to automate and optimize waste segregation. The implemented system achieved an accuracy of 82.14% using CNN with learning rate=0.0001 and 86.97% using SVM in classifying waste into the defined categories, demonstrating its potential for real-world deployment in smart bins or industrial waste management settings. The lower accuracy for CNN is observed as it was not trained to its full capability due to trouble finding optimal hyperparameters.

1. **Research question:** To automate the sorting of recyclable and non-recyclable waste using computer vision to improve the efficiency of waste management systems.

    1.1. What were you trying to find in the dataset: By accurately categorizing waste into biodegradable, recyclable, and non-recyclable categories, the system contributes to reducing environmental pollution and promoting efficient recycling practices.

    1.2. Describe why is this problem important: Consumers also can be confused about how to determine the correct way to dispose of a large variety of materials used in packaging. Efficient waste management is critical for environmental sustainability. Automating waste sorting reduces the need for manual labour, increases recycling rates, and minimizes the volume of waste that goes to landfills. An automated system can help in waste sorting facilities, reducing errors and improving throughput.

    1.3. How did you formulate the problem: Improper waste management remains a significant environmental challenge, contributing to resource wastage, increased landfill overflow, and environmental degradation. Current systems lack the automation and accuracy required for effective waste segregation, often leading to inefficiencies in recycling and disposal processes.

    1.4. Which ML task you used: I used **Object Detection and Classification**, the task is to detect and classify waste items (e.g., plastic, paper, metal, organic waste) from images captured at recycling centres or bins. The system utilizes convolutional neural networks (CNNs) and SVM for object detection and classification, trained on a curated dataset of waste images.

2. **Dataset:**
   Describe the dataset that was used.

    2.1 What is the modality: Image data of size: **656 MB** of data and the sample size: **4752 rows** of images. All images are resized to **224x224 pixels** which ensures that all input images have uniform dimensions.

    2.2 Features: Image features such as pixel values (RGB channels) and classified based on visual characteristics (shape, texture, colour)

    2.3 Labels: Each image (or detected object in an image) is labelled with a category of waste (plastic, paper, glass, metal, organic, cardboard, vegetation, textile etc.)

2.4 How the data is collected: Data is collected from public dataset available online like **TrashNet** or **RealWaste**.

2.5 Describe the importance of this dataset: This dataset is chosen as it is publicly available and I wanted to work on the application of object detection and classification in real-world scenarios.

3 ML Methodology:

3.1 Describe the methods you ended up using: I used SVM and Neural Networks for object detection and classifying the images into their respective categories.

3.2 How do they work:

3.2.1 **Convolutional Neural Networks (CNN)**:

- CNNs are used for image feature extraction and classification in this project.
- They learn spatial hierarchies of features (e.g., edges, shapes, patterns) through convolutional layers, pooling layers, and activation functions.
- CNNs process raw image data directly by resizing, normalizing, and passing the image through convolutional filters to capture spatial information.
- The final layers typically map these extracted features to class predictions i.e. distinguishing between different waste classes.

3.2.2 **Support Vector Machines (SVM)**:

- SVM is used for classification after feature extraction by CNN.
- These feature vectors are then fed into the SVM, which learns a hyperplane to separate different waste categories.
- The SVM leverages these high-dimensional features to classify images into categories like Cardboard, Plastic, Metal, etc.

3.3 How did you use the dataset: The data consists of images of different types of waste items (e.g., Cardboard, Plastic, Metal).
The dataset is organized in a folder structure under /data/realwaste-main/RealWaste, with subfolders representing different categories of waste.
Loading: Dataset is loaded from /data/realwaste-main/RealWaste/.

3.4 Describe the data splits: The dataset was split into training, validation, and testing subsets The entire dataset is split into three subsets:

- **Training Set (75%)**: Used to train the CNN and SVM models.
- **Validation Set (15%)**: Used to validate the model's performance during training and tune hyperparameters.
- **Test Set (10%)**: Used to evaluate the final model's performance on unseen data

3.5 Any preprocessing steps did you use: Converted the images into a PyTorch tensor of shape [C,H,W]. The images are normalized using the mean and standard deviation values as [0.485, 0.456, 0.406] and [0.229, 0.224, 0.225] respectively.

3.5.1 Using **Training Loader**: shuffle=True: Ensures that the training data is shuffled at every epoch to avoid learning spurious patterns.

3.5.2 **Validation/Test Loaders**: shuffle=False: Ensures consistent order during evaluation for reproducibility

3.6 Did the data have any missing points or needed to be cleaned: No, the data was already clean as it was publicly available as an established dataset.

3.7 What preprocessing methods did you use: The Preprocessing methods used were: Resize, normalize, and transform images into tensors.

3.8 Which ML framework or libraries did you use:

3.8.1 **PyTorch**:
Purpose: Used for building and training the CNN (Convolutional Neural Network).
Key Features:
torchvision: For image transformations and dataset handling.
torch.nn: For defining and implementing the neural network architecture.
torch.optim: Used for optimizing the model's weights during training.

3.8.2 **Scikit-learn (sklearn)**:
Purpose: Used for implementing Support Vector Machine (SVM) and other classical machine learning utilities.
Key Features:
sklearn.svm.SVC: For creating and training the SVM classifier.
sklearn.metrics: For evaluating model accuracy and performance.
Dataset splitting and preprocessing tools.

3.8.3 **TorchVision**:
Purpose: Provides datasets, models, and image transformations specific to PyTorch.
Key Features:
torchvision.transforms: Used for resizing, normalization, and other preprocessing steps on image data.
torchvision.datasets.ImageFolder: Used to handle datasets stored in folder structures.

3.8.4 **NumPy**:
Purpose: Essential for numerical computations, preprocessing image data, and manipulating arrays.
Key Features:
Efficient numerical array processing for data handling and matrix computations.

3.8.5 **Matplotlib**:
Purpose: Used for visualization of training loss, accuracy, and other metrics during training.

3.8.6 **PIL** is the **Python Imaging Library** which provides the python interpreter with image editing capabilities

4 Results:

4.1 **Accuracy for CNN with learning rate=0.001**:

```
[12] test_model(model, test_loader, device)
     torch.save(model.state_dict(), "trashnet_cnn.pth")
```

```
Test Accuracy: 79.20%
```

4.2 **Accuracy for CNN with learning rate=0.0001**:

```
[15] test_model(model, test_loader, device)
```

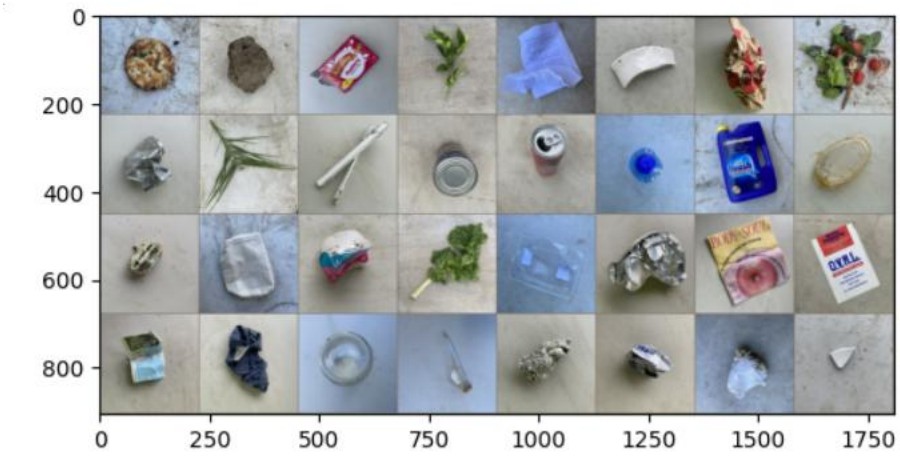
```
Test Accuracy: 82.14%
```

4.3 **Accuracy for SVM**:

```
# Calculate accuracy
svm_accuracy = accuracy_score(test_labels_encoded, test_predictions)*100
print("Test Accuracy:", svm_accuracy)
```

```
Test Accuracy: 86.97478991596638
```

4.4 **Showing the image data and labels corresponding to it**: the classification results for a set of waste items are shown in a matrix format. Each image represents a single sample, and the tensor labels (in the figure) correspond to their predicted classes.




```
Labels: tensor([1, 4, 0, 8, 5, 4, 5, 1, 3, 8, 4, 3, 3, 6, 6, 6, 4, 0, 4, 1, 6, 3, 5, 5,
        5, 7, 2, 2, 3, 5, 4, 4])
```

4.5 We evaluated how hyperparameter tuning (e.g., learning rate, number of layers in CNN, batch size) and different machine learning algorithms (SVMs vs CNNs) affected the model's performance.

1. **CNN Performance**:
    - The CNN model was trained with epochs=40. With optimal configurations, CNNs achieved **accuracy** of **82.14%**.
2. **SVM Performance**:
    - The SVM model was tested with features extracted using CNNs. SVMs performed well for low-dimensional feature sets and had higher accuracy compared to CNNs.

4.6 Comment on which algorithm or set of parameters performed the best and why?

CNN and SVM both performed well on but **SVM Outperforms CNN**, the CNN achieved lower **accuracy** and lesser generalization on unseen data compared to SVM.


```
Predicted class: 5
Class Probabilities:
1: 0.05%
4: 0.02%
0: 0.01%
8: 0.42%
5: 0.03%
4: 0.05%
5: 99.42%
1: 0.00%
3: 0.00%
```

4.7 Comment on the computational complexity and performance of the algorithms.

4.7.1 Computational Complexity for CNN:

- CNNs involve a series of convolutional, pooling, and activation layers that process input images spatially.
- CNNs are computationally intensive but scale well with GPUs.

4.7.2 SVM Complexity:

- Training SVM has higher computational costs with large datasets, especially because it solves quadratic optimization problems.

4.8 Include figures and/or tables and show the effect of the hyperparameters or different algorithms.:

4.8.1 **Performance of CNN with learning rate=0.001**:

```
Final Evaluation on Validation Set:
Validation Loss: 0.7379, Validation Accuracy: 77.11%
Final Loss: 0.7379, Final Accuracy: 77.11%
Final Precision: 0.7665
Final Recall: 0.7784
Final Confusion Matrix:
[[ 45   1   4   5   6   3   3   2   2]
 [  0  60   0   1   2   0   2   1   4]
 [  2   0  54   5   0   0   3   1   1]
 [  0   1   1 103   3   3   7   1   0]
 [  0   7   0   8  42   2   5   6   3]
 [  1   1   0   3   1  59   3   4   0]
 [  2   4   1  15   4   3  89   3   0]
 [  0   1   0   3   4   6   1  37   1]
 [  0   4   0   0   2   0   0   1  60]]
```

4.8.2 **Performance of CNN with learning rate=0.0001**

```
Final Evaluation on Validation Set:
Validation Loss: 0.7621, Validation Accuracy: 83.57%
Final Loss: 0.7621, Final Accuracy: 83.57%
Final Precision: 0.8291
Final Recall: 0.8431
Final Confusion Matrix:
[[ 55   2   3   4   0   2   4   1   0]
 [  0  63   0   0   1   1   0   0   5]
 [  0   0  60   4   0   0   1   0   1]
 [  2   1   4 100   2   1   9   0   0]
 [  2   3   0   5  51   2   7   1   2]
 [  3   0   1   2   3  59   3   1   0]
 [  1   2   2   4   1   2 108   1   0]
 [  0   1   0   4   4   4   1  38   1]
 [  0   4   0   0   2   0   0   0  61]]
```

4.8.3 **Performance of SVM**:

```
Precision: 87.14060575191391
Recall: 86.97478991596638
Confusion Matrix:
 [[36  0  0  3  1  2  3  0  0]
 [ 0 41  0  0  1  0  0  0  1]
 [ 0  0 42  1  0  0  1  0  0]
 [ 4  0  0 68  1  0  3  0  0]
 [ 0  2  0  2 45  0  2  4  1]
 [ 3  0  0  1  4 50  2  0  0]
 [ 0  0  7  6  4  0 68  0  0]
 [ 0  0  0  0  1  0  0 30  0]
 [ 0  2  0  0  0  0  0  0 34]]
```

4.9 Are there any tradeoffs?

4.9.1 **CNNs**:

Pros:

- Higher accuracy, as they capture spatial hierarchies effectively.
- Automatic feature extraction from raw image data.

Cons:

- Computationally expensive (GPU required).
- Require more data to generalize well.

4.9.2 **SVMs**:

Pros:

- Simpler models and faster for smaller datasets.
- Can perform well with hand-engineered features.

Cons:

- Struggles with high-dimensional image data unless pre-processed effectively.
- Lower accuracy compared to CNNs in most image classification tasks.

5 Lessons learned:

5.1 What did you learn:

- **CNNs are Superior for Image Classification**: Convolutional Neural Networks (CNNs) excel at learning spatial hierarchies and patterns in images directly from raw pixel data. CNNs outperform SVMs due to their ability to perform automatic feature extraction.

- **SVMs Have Limitations in Raw Image Data**: SVMs require feature extraction before classification, making them less suitable for high-dimensional image data compared to CNNs, which can learn features automatically.

- **Data Preprocessing & Augmentation Matter**: Preprocessing techniques like resizing, normalization, and augmentation (rotation, flipping, scaling, and cropping) improved model generalization and robustness, highlighting their importance in practical applications.

- **Hyperparameter Tuning Improves Performance**: Properly setting hyperparameters like the learning rate, batch size, number of layers, and optimizer choice directly impacted the CNN model's performance.

5.2 What was important in this problem- The most critical aspects of this problem were:

5.2.1 **Hierarchical Feature Learning**:

CNN's ability to capture spatial relationships and extract features automatically was pivotal to the success of the model in distinguishing between various waste categories.

5.2.2 **Choice of Algorithm**:

Comparing CNNs and SVMs revealed that CNNs were much more effective for image-based classification tasks, especially with raw data.

5.2.3 **Computational Resources**:

CNNs require powerful hardware, particularly GPUs, due to their computationally expensive forward and backward passes. Understanding this resource dependency was crucial.

5.3 What were the challenges:

Several challenges encountered during this project were:

5.3.1 **Computational Costs with CNNs**:

Training CNNs is computationally expensive, particularly with larger datasets. Efficient use of GPU resources was necessary to handle training efficiently.

5.3.2 **Feature Overfitting**:

The CNNs showed overfitting during initial experiments. Hyperparameter tuning and data augmentation were essential to counteract this.

5.3.3 **Dataset Imbalance**:

Uneven class distributions in real-world waste classification datasets posed challenges. Balancing these classes required careful splitting and preprocessing strategies.

5.3.4 **Learning from Limited Data with SVMs**:

The SVM struggled to generalize well unless manually engineered features were extracted, unlike CNNs, which could generalize by automatically learning from raw data.

5.4 What should the reader get out of reading your project:

The reader should understand the following points.

1. Understand how **CNNs** and **SVMs** operate in image classification tasks.
2. Learn the importance of feature extraction, data preprocessing, and augmentation for achieving better classification performance.
3. Recognize the trade-offs between computational costs, performance, and accuracy when selecting machine learning techniques (CNNs vs. SVM).
4. Appreciate that CNNs are much better suited for raw image classification compared to SVMs, especially with complex, spatially structured data.
5. Gain insights into practical challenges, such as computational complexity and hyperparameter tuning, that arise during real-world machine learning projects.