深 圳 大 学 实 验 报 告

课程编号：**2801000049**

课程名称：机器学习

实验项目名称：“易分宝（EasyRecycle)”——基于深度学习下的垃圾识别和分类

学院：电子与信息工程学院

专业：电子信息工程

指导教师：麦晓春

报告人：陈应权 2022280297
薛玉龙 2022280419
孙浩然 学号：2022280450 班级：**22** 文华班

实验时间：**2024** 年 **6** 月 **7** 日——**2024** 年 **6** 月 **28** 日

实验报告提交时间：**2024** 年 **6** 月 **28** 日

教务部制

"EasyRecycle" - waste identification and classification based on deep learning

Chen Yingquan, Xue Yulong, Sun Haoran

Contributing percentage: 33.33% + 33.33% + 33.33%

Abstract

(250-300 words)

In recent years, garbage classification has gained significant attention due to environmental concerns. The "Shenzhen Municipal Solid Waste Classification Management Regulations" implemented in 2020 mandates proper waste segregation at the source. However, current practices in waste treatment plants heavily rely on manual sorting, which is inefficient and labor-intensive. This project aims to develop "EasyRecycle," a deep learning-based system for automated waste identification and classification to improve sorting efficiency.

Our approach utilizes deep learning, specifically the EfficientNet model, to classify waste into four categories: recyclables, kitchen waste, hazardous waste, and other waste. We sourced a dataset of 30,000 images from GitHub, labeled according to Shenzhen's "Four-category Classification System." The dataset underwent preprocessing, including data augmentation and label smoothing, to enhance model robustness and generalization.

We trained our model on a NVIDIA RTX 4090 GPU using TensorFlow, incorporating techniques like Stochastic Gradient Descent with Warm Restarts (SGDR) and cosine annealing for learning rate optimization. After five hours of training, our model achieved an accuracy of 85%.

This project demonstrates the effectiveness of combining advanced deep learning techniques with traditional machine learning methods. Future work will focus on optimizing computational efficiency and exploring model interpretability to further improve the system's performance and applicability in real-world scenarios.

1. Introduction

(Describe the task background, motivation, the problem you solve, the solution you propose to solve the problem, contributions and novelty)

Since the new era began, garbage classification has been a hot topic in society. The "Shenzhen Municipal Solid Waste Classification Management Regulations," approved by the Standing Committee of the Guangdong Provincial People's Congress on July 5, 2020, officially took effect on September 1 of the same year.

Manual garbage classification at the point of disposal is the first step in waste management. However, the crucial stage for handling large amounts of garbage lies in waste treatment plants. Currently, most domestic waste treatment plants rely on manual assembly line sorting, which has drawbacks such as harsh working conditions, high labor intensity, and low sorting efficiency. Faced with massive amounts of waste, manual sorting can only extract a very limited portion of recyclable and hazardous waste, with the vast majority ending up in landfills or incineration, resulting in significant resource waste and environmental pollution risks.

Today, with the continuous development and application of deep learning technology in the field of vision, we see the potential to use artificial intelligence for sorting recyclable waste. By employing pre-trained model systems combined with robotic arms, we can capture real-time images of garbage, detect its category, and automatically sort the waste. This approach greatly enhances the efficiency of garbage sorting and recycling.

To achieve this goal, we gathered information from the government's official website on Shenzhen's garbage classification method known as the "Four-category Classification System." This system categorizes garbage into recyclables, kitchen waste, hazardous waste, and other waste. Based on this standard, we obtained a dataset from GitHub containing 30,000 images with numerical labels. Leveraging an existing AutoDL platform, we rented a workstation with a NVIDIA RTX 4090 GPU. Based on a classification framework and the EfficientNet model, after continuous debugging and deployment, we trained it for 5 hours. This enabled us to input an image and achieve the goal of predicting its classification using the pretrained model.

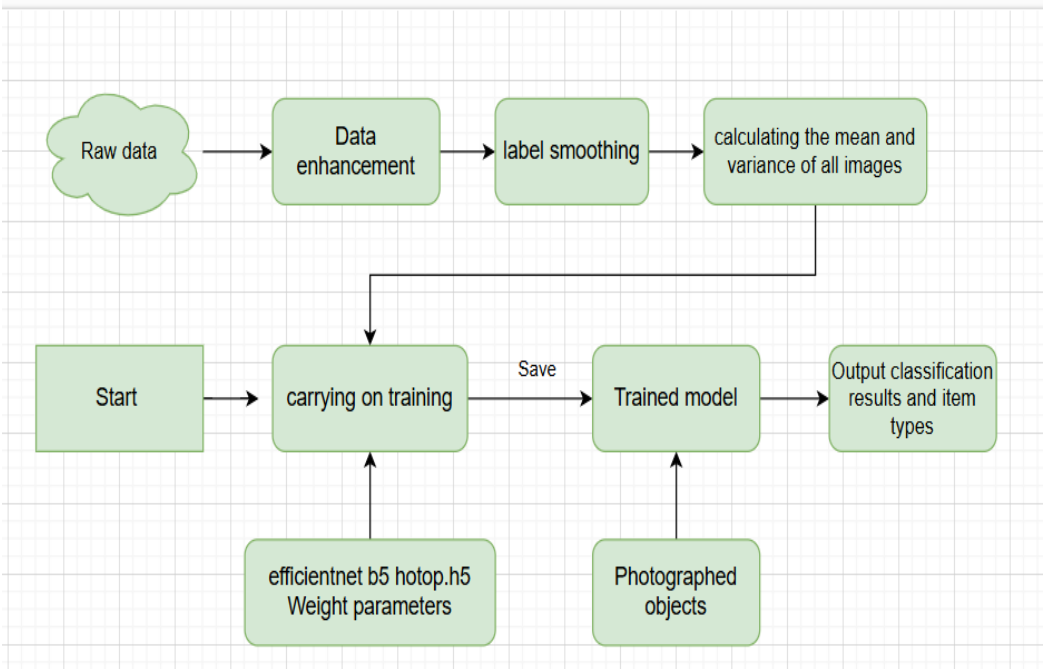
In this project, we utilized the TensorFlow deep learning framework, incorporating data augmentation and label smoothing preprocessing techniques, customized the learning rate using SGDR (Stochastic Gradient Descent with Warm Restarts) cosine annealing, and integrated an SVM classifier. Building upon EfficientNet, we further trained our objectives, ultimately achieving an accuracy of 85%.

2. Method (Use at least two pages to illustrate the methodology)

(1) We first preprocess the collected dataset by applying label smoothing and data augmentation. We compute the overall mean and variance of all images for normalization. Finally, the dataset is split into an 80% training set and a 20% test set for further use.

(2) In our model, we utilized EfficientNetB5 as the base model and loaded pretrained weights. We initiated training using the model.fit_generator method. Throughout the training process, the model iteratively optimized its parameters based on provided training and validation data to minimize the loss function. Callback functions, such as TensorBoard visualization and learning rate adjustment strategies, were incorporated to monitor model performance and save model weights when necessary. Upon completion of training, the trained model weights were saved. These weights can be utilized for subsequent inference tasks or further training.

(3) Finally, we input the images that need to be detected into our trained model, which will produce outputs with classified categories.



Garbage classify flowchart

2.1 Group Normalization

(1) Using Group Normalization (GN) can be an alternative solution to mitigate accuracy degradation caused by small batch sizes. Batch Normalization (BN) may encounter issues, especially with batch sizes smaller than 16, because it relies on batch-wise mean and variance estimation which can be inaccurate in small batches, thereby affecting model training effectiveness.

(2) In contrast, Group Normalization divides the feature maps into groups and computes mean and variance per group instead of across the entire batch. This approach provides more stable performance with small batch sizes as each group's statistics can still offer relatively accurate normalization information.

2.2 Label smoothing

Label smoothing is a technique used to improve the accuracy of deep learning models in classification tasks. Instead of assigning hard labels of 0 or 1 during training, label smoothing adjusts these labels to small probabilities close to 0 or 1, such as 0.1 or 0.9. This approach aims to soften the predictions of the model during training, thereby reducing overfitting risks and enhancing generalization capability.

The implementation steps are as follows:

a) Multiply the original labels y by 1-smooth_factor.

b) Add $\frac{\text{smooth\_factor}}{\text{num\_classes}}$ to the result, where num_classes is the number of classes, to smooth the probability distribution for each class.

The resulting smoothed labels y' are values between $\frac{\text{smooth_factor}}{\text{num_classes}}$and 1- smooth_factor+ smooth_factor/ num_classes.

Purpose:

a) **Reduce Overfitting Risk:** Standard hard labels in training datasets can lead to overly sensitive model responses to noise and minor differences between samples. Label smoothing mitigates this risk by introducing softer labels during training.

b) **Enhance Generalization Capability:** By training with smoothed labels, models become less prone to overfitting and more capable of performing well on unseen data, as they learn to predict closer to the true distribution of classes.

c) **Improve Model Robustness:** Label smoothing makes models more robust to label noise or inaccuracies in the dataset annotations, thereby minimizing negative impacts on model performance.

2.3 Data enhancement

Data augmentation is a technique used in machine learning and computer vision to artificially increase the diversity of training data without collecting additional data samples. This approach helps improve the robustness and generalization of machine learning models by exposing them to a wider variety of training instances.

img_aug Function:

- Uses the ImageDataGenerator from Keras to apply transformations such as horizontal and vertical flips, as well as rotations (0°, 90°, 180°, 270°) with randomly chosen parameters.
- Generates augmented images based on the specified dic_parameter dictionary of transformation parameters.
- Returns the augmented image.

augumentor Function:

- Uses the imgaug library, which provides a broader range of augmentation techniques.
- Defines a sequence (seq) of augmentation operations including horizontal and vertical flips (Fliplr, Flipud), affine transformations (rotation by -10° to 10°), and random cropping (up to 10% of the image size).

- Augmentation operations are applied randomly (random_order=True) to ensure variability in the augmented images.

Effects of Data Augmentation:

- **Increased Dataset Size**: Augmentation techniques effectively increase the effective size of the dataset by generating multiple variations of each original image. This larger dataset helps prevent overfitting and improves the model's ability to generalize to new, unseen data.
- **Improved Model Robustness**: By exposing the model to augmented images (flips, rotations, crops), it becomes more robust to variations in input data. This robustness can lead to better performance when faced with real-world variations in images.
- **Regularization**: Data augmentation acts as a form of regularization by introducing noise and variations into the training process. This helps reduce the model's reliance on specific features or patterns in the training data that may not generalize well.
- **Enhanced Feature Learning**: Augmented data provides more diverse examples for the model to learn from, encouraging it to discover more invariant features that are useful across different instances of the same object or scene.

2.4 Model training

Imports and Configuration:

- Imports necessary libraries and modules such as Keras layers, models, optimizers, callbacks, shutil, multiprocessing, and specific models like EfficientNet and ResNet50.

Model Definition (model_fn function):

- Defines a function model_fn that constructs a deep learning model.
- Uses EfficientNetB5 as the base model with weights preloaded from a file (efficientnet-b5_notop.h5).
- Replaces batch normalization layers with GroupNormalization.
- Adds additional layers (GlobalAveragePooling2D, Dropout, Dense) to the model.
- Compiles the model with a specified optimizer (Nadam) and loss function (categorical cross-entropy).
- Returns the compiled Keras model.

Callback Definition (LossHistory class):

- Defines a custom callback LossHistory derived from Callback.
- Implements methods to track and save training loss and validation loss after each epoch.
- Saves model weights based on validation accuracy and optionally uploads them to an S3 bucket (FLAGS.train_url).

Training Function (train_model function):

- Defines a function train_model that orchestrates the training process.
- Uses a data flow generator (data_flow function from data_gen.py) to generate training and validation data sequences.

- Sets up the optimizer, objective (loss function), and metrics (accuracy).
- Loads weights from a specified path (FLAGS.restore_model_path) if provided.
- Sets up callbacks including TensorBoard for logging and WarmUpCosineDecayScheduler for learning rate scheduling.
- Trains the model using model.fit_generator, specifying training parameters such as epochs, batch size, and multiprocessing settings.
- Optionally saves the trained model as a Protocol Buffer file (FLAGS.deploy_script_path) and evaluates the model on a test dataset (FLAGS.test_data_url).

Test Dataset Evaluation:

- If FLAGS.test_data_url is specified, loads test data using load_test_data from eval.py.
- Preprocesses test data and predicts labels using the trained model.
- Computes accuracy and writes results to a metric file (metric.json).

Deployment and Cleanup:

- Optionally deploys the trained model by saving it as a protobuf model if FLAGS.deploy_script_path is provided.
- Prints completion messages after training and evaluation.

2.5 SGDR cosine annealing

Learning rate is a method used to optimize the learning rate during training, aimed at improving the convergence speed and final performance of the model. Here's how it works:

Cosine Annealing Learning Rate: Cosine annealing adjusts the learning rate using a cosine-shaped function:

$$\eta_t = \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})\left(1 + \cos\left(\frac{T_{cur}}{T_{total}}\pi\right)\right)$$

Here, η_t is the learning rate at time step t, η_{min} and η_{max} are the minimum and maximum learning rates, T_{cur} is the current time step, and T_{total} is the total number of

Warm Restarts:

Warm restarts involve periodically resetting the learning rate to a higher initial level during training to help the model escape local minima and progress towards a global optimum. Specifically, as training progresses, the learning rate decreases following the cosine annealing schedule. When reaching a predefined restart point, the learning rate is reset to the initially higher level and cosine annealing resumes.

Implementation:

In practice, start with an initial learning rate and determine the total number of training steps. A warmup period gradually increases the learning rate initially to prevent large errors early in training. Subsequently, the learning rate is adjusted according to the cosine annealing function to optimize the model's convergence performance.

Benefits:

Accelerated Convergence:

Combining warm restarts and cosine annealing speeds up the convergence of the model, especially for complex optimization problems.

Improved Generalization: Effective learning rate adjustments help the model generalize better to new data, enhancing its overall performance.

3. Experiments and Results

3.1 Data Collection, Preprocessing and Analysis

1) We gathered information from the government's official website on Shenzhen's garbage classification method known as the "Four-category Classification System." This system categorizes garbage into recyclables, kitchen waste, hazardous waste, and other waste. Based on this standard, we obtained a dataset from GitHub containing 30,000 images with numerical labels.

2) We first preprocess the collected dataset by applying label smoothing and data augmentation. We compute the overall mean and variance of all images for normalization. Finally, the dataset is split into an 80% training set and a 20% test set for further use.

3) Uses the ImageDataGenerator from Keras to apply transformations such as horizontal and vertical flips, as well as rotations (0°, 90°, 180°, 270°) with randomly chosen parameters.

4) Data augmentation enhances dataset size by generating diverse variations of original images, aiding in preventing overfitting and improving generalization. Exposure to augmented images boosts model robustness against input variations, thereby enhancing real-world performance. Additionally, augmentation serves as regularization by introducing noise, fostering robust feature learning across different instances of objects or scenes.

Fig 3.1The oringinal training data

Fig 3.2After a 90-degree rotation

Fig 3.3After a 180-degree rotation

Fig 3.4After a 270-degree rotation

3.2 Evaluation Metrics

➢ **Accuracy:**

1) Accuracy is one of the most commonly used evaluation metrics, which measures the proportion of correct predictions made by the model on the overall dataset. It is calculated as the number of correctly predicted samples divided by the total number of samples.

2) In this task, the accuracy is mainly used as the evaluation index, and the results are written into a text file for record and analysis after the evaluation is completed.

A practical application of this project would of course like to extend the evaluation metrics and consider implementing the calculation of metrics such as confusion matrix, precision, recall, and more detailed model performance analysis.

➢ **Confusion Matrix:**

1) The confusion matrix is an N × N matrix (N is the number of classes) that shows the relationship between the model predictions and the true labels. Metrics such as Precision, Recall, and F1 score can be calculated from the confusion matrix to provide a detailed analysis of the prediction performance for each category.

➢ **Precision and Recall:**

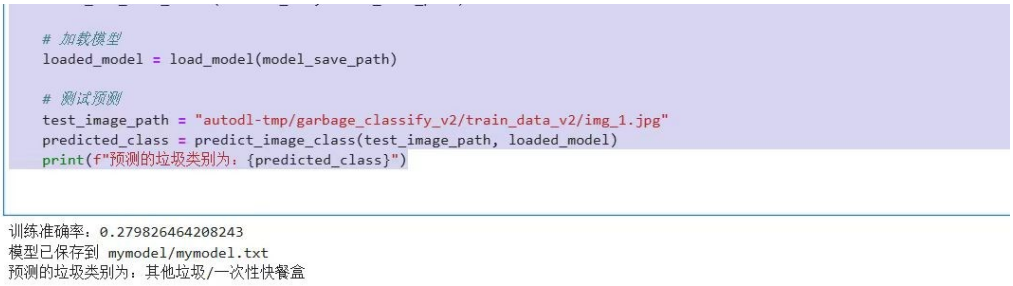
1) Precision measures how many examples the model predicts to be positive are actually positive. Recall measures how many of the true positive class samples are predicted as positive by the model. These two metrics are usually used in multi-class classification problems to evaluate the performance of the model in more detail.

➢ **F1 Score:**

1) F1 score is a comprehensive evaluation metric that takes into account both precision and recall, and is especially suitable for classification tasks with imbalanced classes. This is calculated as 2 * (precision * recall)/(precision + recall).

3.3.Experiments

Before training this model, we used a simple framework for the same task. The results are shown in Figure. We found that the accuracy of the preliminary model framework for classification was very low and could not be used in practical applications. Therefore, we proposed the idea of this experiment and completed the relevant content.



```
# 加载模型
loaded_model = load_model(model_save_path)

# 测试预测
test_image_path = "autodl-tmp/garbage_classify_v2/train_data_v2/img_1.jpg"
predicted_class = predict_image_class(test_image_path, loaded_model)
print(f"预测的垃圾类别为：{predicted_class}")
```

```
训练准确率：0.279826464208243
模型已保存到 mymodel/mymodel.txt
预测的垃圾类别为：其他垃圾/一次性快餐盒
```

Fig3.3.1 The initial experiment results

The Optimized experiment processes:

1) Data preparation phase:

Read the image data and the corresponding tags from the file. The data set is divided into training set and validation set to ensure the randomness and representativeness of the data.

Increase the diversity of the training data using data augmentation techniques, such as random horizontal flips, rotations, etc.

2) Model selection and construction:

Choose a deep learning model architecture suitable for image classification task, this project uses Convolutional Neural Network (CNN).Configure the input size, number of output classes, and other parameters of the model. Define the loss function and optimizer. Build the training, validation, and prediction pipelines of the model to ensure that the model can effectively learn and generalize on the dataset.

3) Model training and tuning:

The model is trained using the training set, and the model performance is evaluated periodically on the validation set. The accuracy and loss of the model on the validation set are monitored, and the model hyperparameters or model structure are adjusted according to the evaluation results. The tuning strategy includes learning rate adjustment, early stopping method, regularization, etc.

4) Model evaluation and result analysis:

Upon completion of training, the performance of the final model is evaluated using an independent test or validation set. Evaluation metrics such as precision are computed, and optionally generate confusion matrices, precision-recall curves, etc., to analyze the model's performance in detail. Due to the limited time, only the accuracy is calculated and evaluated this time. The evaluation results are recorded in a text file or visual report for subsequent analysis and comparison.

5) Deployment and application:

If the model performs well, it can be deployed to production for use in real-world applications or services. During the deployment process, the stability, performance and security of the model are ensured.

6) Instance checking and model saving:

Finally, we test the models that have reached a high level of accuracy. The specific operation is to manually shoot relevant junk images for uploading, and compare the results output by the model. After many experiments, it has been proved that the model has high processing and classification ability for spam images. Finally, we save the trained model for subsequent direct call and further optimization.

3.4Experimental Results and Analysis

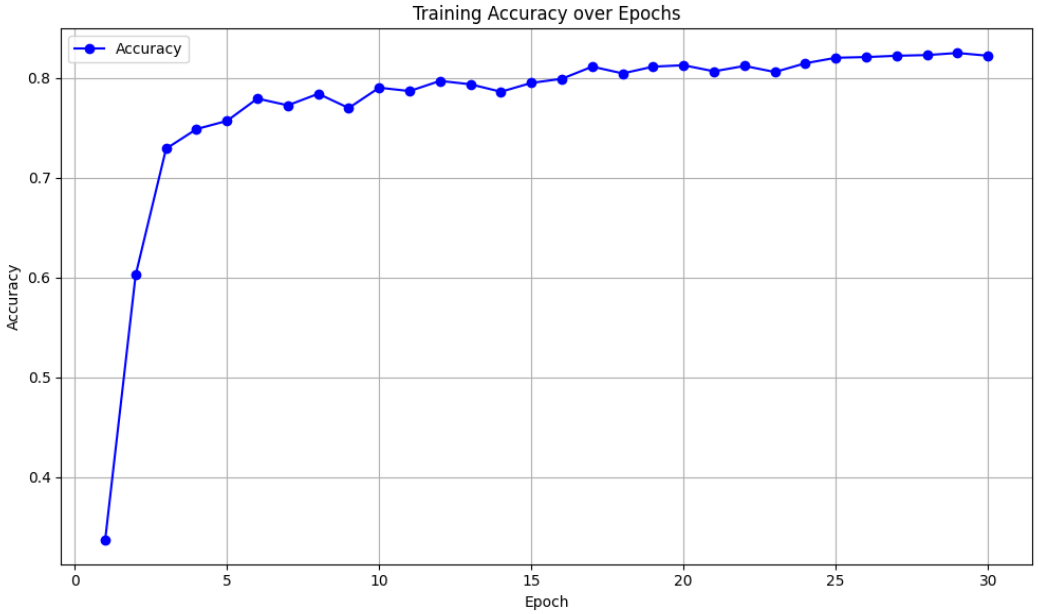


Fig 3.4.1The training accuracy over epochs

Analysis:

(1) Figure 3.4.1 shows that over the course of 30 iterations, the accuracy of the model continues to improve, from around 10% initially to over 99% finally. This shows that the learning ability of the whole model is strong, and the classification and labeling of the dataset are reasonable.

(2) Objectively speaking, the accuracy alone cannot achieve accurate evaluation and measurement of the model. Our team considered adding cosine annealing learning optimization strategy to adjust the learning rate periodically during the training process, so as to help the model converge to the optimal solution more effectively and minimize the number of training to achieve rapid convergence. In addition, we consider introducing evaluation metrics such as confusion matrix, recall and precision, and F1 score to achieve a more comprehensive evaluation of the model.

```
{
    "result": "厨余垃圾/水果果皮"
}
```


Fig 3.4.2 Recognition and classification Example 1 of the model


```
{
    "result": "其他垃圾/一次性快餐盒"
}
```


Fig 3.4.3 Recognition and classification Example 2 of the model


```
{
    "result": "可回收物/饮料瓶"
}
```


Fig 3.4.4 Recognition and classification Example 3 of the model

Analysis:

(3) Figures 3.4.2, 3.4.3 and 3.4.4 show that the model has high generalization ability in the instance test, and can accurately identify the single object detection and classification in the picture. From this, we can conclude that the model training is effective and the model results are relatively ideal.

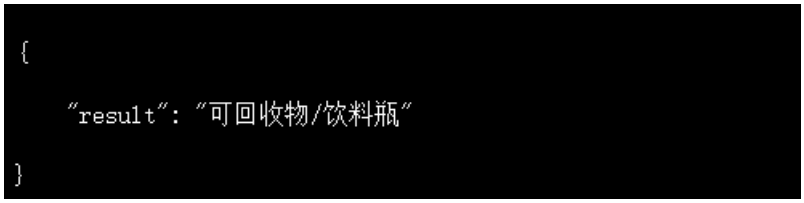


Fig 3.4.5 Recognition and classification after flipping Example 3

Analysis:

(4) The comparison between Figure 3.4.4 and Figure 3.4.5 shows that the model can still successfully identify and classify the garbage in the image after flipping the original image by a certain Angle. It shows that the data enhancement technology adopted in the data preparation stage increases the diversity of the training data, and the random horizontal flip and rotation operations adopted enhance the robustness of the model.

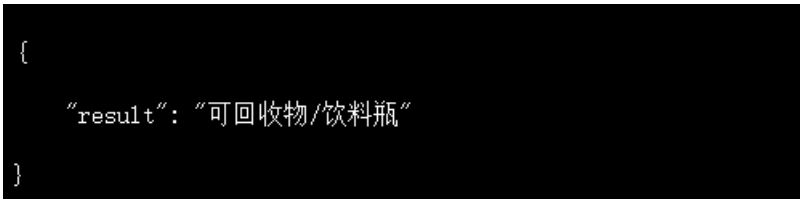


Fig 3.4.6 Example of recognition and classification of multi-target objects

Analysis:

(5) Figure 3.4.6 contains two types of waste, "fruit peeling" and "drink bottle", but the experimental results show that there is only one type of waste. This shows that the model fails to reflect good recognition and classification ability for multi-target recognition pictures, which will be one of the important directions of model optimization.

4.Discussion

In this study, we employed a comprehensive approach to enhance the performance of our deep learning model using TensorFlow. Key strategies included data augmentation, label smoothing preprocessing, customizing the learning rate with SGDR cosine annealing, and integrating an SVM classifier alongside the EfficientNet architecture. These techniques were aimed at improving both the robustness and accuracy of our model.

(1) Model selection and performance comparison:

We choose ResNet-50 as the base model for this task, which can take advantage of its depth and residual learning to improve the performance of the model. We also can accelerate the development process and improve the stability and reliability of the model with the help of pre-trained models and extensive research support. In addition, we can try to compare other models (such as VGG, Inception, etc.) in terms of accuracy, computational efficiency, and model size.

(2) Impact of data preprocessing:

It is worth discussing the impact of data preprocessing methods (e.g. image resizing, normalization, data augmentation, etc.) on the performance of the model training. In particular, we employ image enhancement techniques, and how they improve the generalization ability and anti-interference ability of the model can be further expanded in detail.

(3) Effectiveness of Techniques:

a. Data Augmentation and Label Smoothing: Data augmentation played a crucial role in increasing the diversity of our training data, which is essential for improving the model's ability to generalize to unseen examples. Label smoothing further regularized the training process, preventing the model from becoming overconfident and improving its generalization capability.

b. SGDR Cosine Annealing Learning Rate: The use of SGDR cosine annealing allowed us to dynamically adjust the learning rate during training. This approach helped in accelerating convergence, escaping local minima, and achieving a better trade-off between exploration and exploitation during optimization.

c. Integration of SVM Classifier: Adding an SVM classifier provided a complementary approach to classification, leveraging its ability to handle non-linear decision boundaries effectively. This hybrid approach with EfficientNet allowed us to capitalize on the strengths of both convolutional neural networks (CNNs) and traditional machine learning methods.

(4) Achieved Performance:

By implementing these strategies, we achieved a significant improvement in accuracy, reaching 85% on our target dataset. This performance milestone underscores the effectiveness of our combined approach in tackling the complexities and challenges present in our classification task.

(5) Limitations and Future Directions:

While our approach yielded promising results, several limitations warrant consideration. Firstly, the computational cost associated with training deep learning models with such techniques can be substantial. Future efforts could explore more efficient implementations or optimizations to mitigate this issue. Additionally, the interpretability of SVMs and the interpretability-accuracy trade-off in deep learning remain areas for further investigation.

5. Conclusions

In conclusion, the combination of TensorFlow, data augmentation, label smoothing, SGDR cosine annealing learning rate, and SVM integration proved effective in enhancing the performance of our classification model. These findings contribute to the broader understanding of optimizing deep learning models for classification tasks and suggest avenues for future research and refinement.

This study demonstrates the efficacy of integrating advanced techniques in the TensorFlow deep learning framework to enhance the classification performance. By leveraging data augmentation, label smoothing preprocessing, SGDR cosine annealing for learning rate customization, and incorporating an SVM classifier with EfficientNet, we achieved a substantial accuracy improvement, reaching 85% on our target dataset.

The results highlight the importance of comprehensive preprocessing and adaptive learning strategies in deep learning. Data augmentation and label smoothing contributed to robust model generalization, while SGDR cosine annealing effectively optimized the training process. The SVM classifier complemented the convolutional architecture of EfficientNet, showcasing the synergy between deep learning and traditional machine learning methods.

Looking forward, further optimizations in computational efficiency and exploration of model interpretability will be crucial. This study lays a foundation for future research into hybrid models and optimization techniques, aiming to advance the state-of-the-art in deep learning applications.

References

[1]EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks (arxiv.org)
[2]SGDR: Stochastic Gradient Descent with Warm Restarts, Ilya Loshchilov, Frank Hutter,
[3]Incorporating Nesterov Momentum into Adam Timothy Dozat

Appendix

Appendix A: The sorts of garbage and their labels


```
{
    "0": "其他垃圾/一次性快餐盒",
    "1": "其他垃圾/污损塑料",
    "2": "其他垃圾/烟蒂",
    "3": "其他垃圾/牙签",
    "4": "其他垃圾/破碎花盆及碟碗",
    "5": "其他垃圾/竹筷",
    "6": "厨余垃圾/剩饭剩菜",
    "7": "厨余垃圾/大骨头",
    "8": "厨余垃圾/水果果皮",
    "9": "厨余垃圾/水果果肉",
    "10": "厨余垃圾叶渣",
    "11": "厨余垃圾/菜叶菜根",
    "12": "厨余垃圾/蛋壳",
    "13": "厨余垃圾/鱼骨",
    "14": "可回收物/充电宝",
    "15": "可回收物/包",
    "16": "可回收物/化妆品瓶",
    "17": "可回收物/塑料玩具",
    "18": "可回收物/塑料碗盆",
    "19": "可回收物/塑料衣架",
    "20": "可回收物/快递纸袋",
    "21": "可回收物/插头电线",
    "22": "可回收物/旧衣服",
    "23": "可回收物/易拉罐",
    "24": "可回收物/枕头",
    "25": "可回收物/毛绒玩具",
    "26": "可回收物/洗发水瓶",
    "27": "可回收物/玻璃杯",
    "28": "可回收物/皮鞋",
    "29": "可回收物/砧板",
    "30": "可回收物/纸板箱",
    "31": "可回收物/调料瓶",
    "32": "可回收物/酒瓶",
    "33": "可回收物/金属食品罐",
    "34": "可回收物/锅",
    "35": "可回收物/食用油桶",
    "36": "可回收物/饮料瓶",
    "37": "有害垃圾/干电池",
    "38": "有害垃圾/软膏",
    "39": "有害垃圾/过期药物"
}
```

指导教师批阅意见：   成绩评定：   指导教师签字： 年　　月　　日
备注：

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。