Final Project COMP309

Jacob French - 300572009

Introduction

The primary objective of this project is to employ a Convolutional Neural Network (CNN) for a classification task. This task involves categorizing images as either strawberries, cherries, or tomatoes.

Initially, the project will establish a baseline neural network. It will then proceed to explore various modifications, including changes in optimizers, minibatch sizes, and other parameters, to transform it into a CNN. More advanced techniques such as data augmentation and transfer learning will also be investigated.

The dataset comprises 4,500 images, each with an expected resolution of 300x300 pixels. These images are evenly distributed among the three classes, with 1,500 images per class.

The final model's performance will be assessed on unseen data, exclusively available to the lecturer.

Problem Investigation

Exploratory Data Analysis (EDA)

For the EDA phase, the data is loaded using basic tensor transformations to facilitate a thorough examination of the images.

Initially, the data is divided into an 80/20 split for training and testing. During the exploration phase, only the training set will be examined. This approach is crucial because the ultimate goal is to perform well on unseen data, necessitating limited exploration of the training set. This precaution aims to prevent overfitting and enhance generalization. To ensure reproducibility, a random seed is manually set.

The primary issue under investigation involves identifying outliers in the form of images with incorrect dimensions. Upon inspection, it was discovered that 60 images had incorrect dimensions, with an even distribution among the classes (20 per class). This issue led to four key problems: images with no fruit present, incorrectly sized images with fruit present, images containing multiple classes, and images with incorrect labels. Some examples are provided below for reference:

Problem Examples:

1. Images with no fruit present.
2. Incorrectly sized images with fruit present.
3. Images containing multiple classes.
4. Incorrectly labeled images.

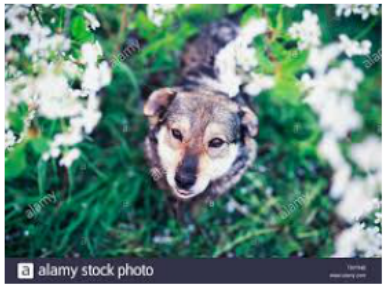
Images Without Fruit

A significant portion of these photographs lacks the specific fruit they are labeled with. Instead, they may consist of text spelling out the fruit's name or depict objects with the word "cherry" in their name, such as "cherry blossom".

Label: 0, Index: 144, Dimensions: (169, 299, 3)

Label: 2, Index: 3114, Dimensions: (197, 256, 3)

Label: 0, Index: 223, Dimensions: (193, 261, 3)

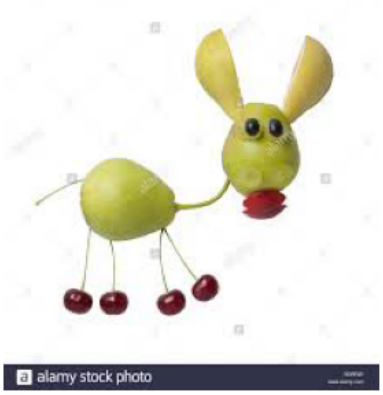


Images with Incorrect Dimensions

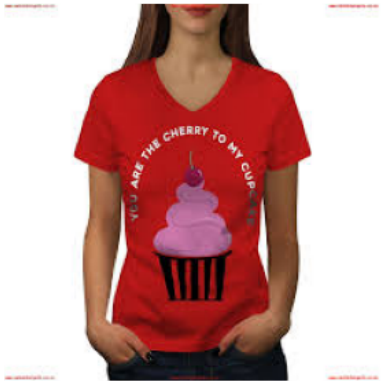
The majority of outliers exhibit this characteristic, with a total of 48 images falling into this category. These images do not conform to the 300x300 pixel dimensions, rendering them incompatible with a CNN designed for 300x300 images.

Label: 1, Index: 1548, Dimensions: (177, 284, 3)

Label: 0, Index: 72, Dimensions: (230, 219, 3)



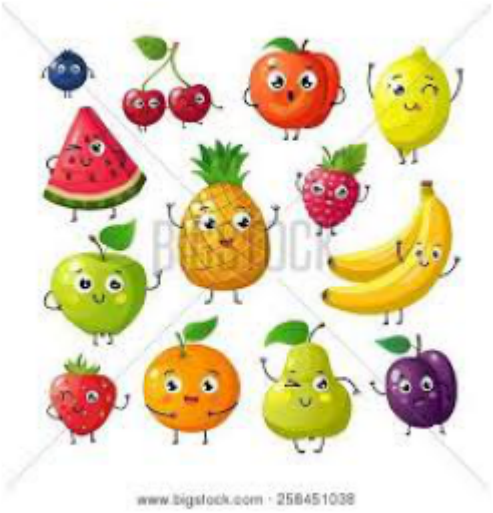
Label: 0, Index: 462, Dimensions: (225, 225, 3)



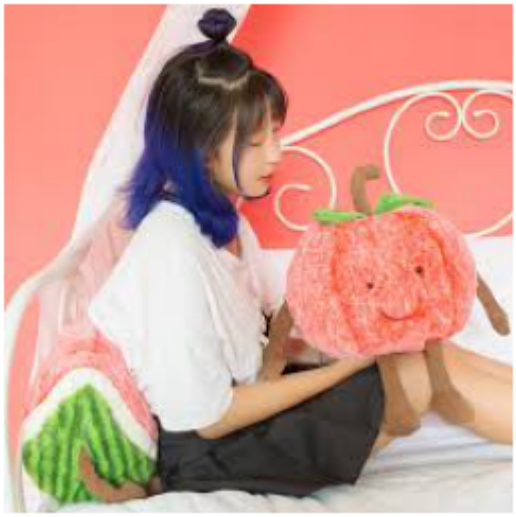
Images with Multiple Classes & Incorrectly Labeled Images

It's worth noting that there is only one image in the training set for each of these issues, namely images with multiple classes and incorrectly labeled images. This observation is significant, as it suggests that there might be more instances of these issues in the unseen data.

Label: 0, Index: 157, Dimensions: (229, 220, 3)



Label: 0, Index: 314, Dimensions: (225, 225, 3)



Preprocessing

In an effot to enhance the dataset's quality, several preprocessing steps were implemented. All images with issues were removed, with the exception of those with incorrect dimensions. These problematic images were deliberately removed from the dataset. Their exclusion was considered necessary because they would not provide meaningful information for the model's learning process. Moreover, there was a considerable risk that they might be misclassified due to the absence of any inherent fruit presence, and they would not contribute to improving the model's generalization.

For images with incorrect dimensions, a resizing operation was performed to standardize them to a 300x300 resolution using the PyTorch resize transformation. This step was crucial to ensure compatibility with the CNN architecture.

The overall impact of this preprocessing was relatively modest, resulting in the removal of 10 items from the dataset. However, this had a minimal effect on the model's performance, as these problematic images constituted only a small fraction of the dataset.

Methodology

Baseline Model

To establish a benchmark model for further improvement, a neural network was trained without convolutional layers. This specific network architecture incorporates two dense layers, which are fundamental components of artificial neural networks.

1. **Dense Layers**: Dense layers, also known as fully connected layers, are essential neural network components where each neuron or unit is connected to every neuron in the preceding and succeeding layers. In our model, we used two dense layers, with the first layer having 128 units and the second layer having 3 units. These layers serve as powerful feature extractors, capturing complex patterns in the data.

The activation function used in these layers was Rectified Linear Unit (RELU), abbreviated as RELU:

2. **RELU Activation**: RELU is a widely employed activation function in neural networks. It introduces non-linearity into the model by outputting the input value if it's positive or zero, and outputting zero if the input is negative. This non-linearity is essential for enabling the network to learn complex relationships within the data.

For optimization, Stochastic Gradient Descent (SGD) was employed:

3. **Stochastic Gradient Descent (SGD)**: SGD is an optimization algorithm used to update the network's parameters during training. It operates by adjusting the weights and biases to minimize the loss function and improve model performance. In our model, we set the learning rate to 0.001 and the momentum to 0.9 to control the size of weight updates and accelerate convergence.

The loss function used for training and evaluation was Cross-Entropy Loss:

4. **Cross-Entropy Loss**: Cross-Entropy Loss, often referred to as log loss, is a loss function commonly used for classification tasks. It quantifies the dissimilarity between the predicted probabilities and the actual class labels. Minimizing the cross-entropy loss encourages the model to make accurate predictions.

The basic model, incorporating these elements, achieved an accuracy score of 51%. The experiment was conducted with a batch size of 64 over 5 epochs.

Convolutional Neural Network

To improve the baseline model, convolutional layers were introduced, playing a critical role in the model's ability to recognize essential spatial patterns within the image data.

1. **Convolution Layers**: Two convolutional layers were integrated into the architecture, representing a pivotal step in its development. The initial convolutional layer is tasked with transforming the Red, Green, and Blue (RGB) image features into a set of six distinct features. These features serve as fundamental building blocks for the recognition of patterns and characteristics within the images. Subsequently, the second convolutional layer refines these initial six features, mapping them into a more intricate set of 16 features. This sequential transformation equips the network to identify and learn intricate visual patterns and nuances within the images, a crucial aspect for accurate classification.
2. **2D Max Pooling**: Placed between the convolutional layers, 2D max pooling serves a pivotal role in the data processing pipeline. This technique is employed to downsample the data and capture the most critical features. By reducing the spatial dimensions and emphasizing essential aspects within each localized region, 2D max pooling ensures that the network focuses on the most informative and distinctive attributes in the images. This step is fundamental for reducing complexity and preparing the data for classification.

The processed features from the convolutional layers and max pooling are then forwarded into dense layers, mirroring the architecture of the baseline model, with 128 units and 3 units respectively. The training configuration remains consistent with the same batch size and number of epochs. This refined model achieved a notable accuracy score of 55%.

Optimizers

An optimizer in the context of deep learning is a crucial component of the training process for neural networks. Its primary role is to adjust the model's parameters, such as weights and biases, in order to minimize a defined loss function. Optimization is vital in machine learning because it helps the model learn and adapt, ultimately improving its performance during training. By finding the right set of parameters, the optimizer guides the neural network to make accurate predictions.

Up to this point, the same optimizer, Stochastic Gradient Descent (SGD), has been employed for training the models. However, there are alternative optimizers available, such as Root Mean Squared Propagation (RMSprop) and Adam. The reason for exploring different optimizers is to determine which one yields the best results for the given task. To assess this, a 3-fold Cross-Validation method has been implemented. Cross-validation is particularly useful for assessing the model's generalization performance and preventing overfitting. In this case, 3-fold cross-validation was chosen to balance training times and efficiency.

In the cross-validation process, a separate test set is created for evaluation, while the training set is solely used for cross-validation.

Results

	Fold 1	Fold 2	Fold 3	Average	Time
SGD	52.00%	48.50%	43.33%	47.94%	158s
ADAM	54.58%	52.42%	50.00%	52.33%	133s
RMSProp	41.17%	48.75%	46.83%	45.583%	135s

As observed in the results, the Adam optimizer performs the best on the validation set, achieving the highest average accuracy (52.33%). Consequently, this optimizer will be used for further model trials.

When training the model with the Adam optimizer on the entire training set, it attains an accuracy of 62%. Additionally, it's important to note that there might be significant differences between the k-fold score and the total score, mainly because less data is utilized during cross-validation.

MiniBatch Sizes

A minibatch, in the context of deep learning, is a subset of the training dataset that is used to update the model's parameters during each iteration of the training process. Instead of presenting the entire dataset at once, the training data is divided into smaller batches. Minibatches allow for more efficient and faster training as the model's parameters are updated more frequently. This process is crucial in achieving a balance between computational efficiency and model performance.

In the investigation of minibatch sizes using cross-validation, the initial test employed a minibatch size of 64. Subsequently, minibatch sizes of 32 and 128 were tested to explore whether increasing or decreasing the batch size would lead to improved model performance.

Results

Batch Size	Fold 1	Fold 2	Fold 3	Average	Time
32	51.58%	53.75%	46.83%	50.72%	156.54s
64	53.25%	49.75%	55.75%	52.92%	149.65s
128	52.42%	41.33%	49.58%	47.78%	149.38s

As observed in the results, the batch size 64 performs the best on the validation set, achieving the highest average accuracy (52.92%). Consequently, this minibatch size will be employed in the subsequent experiments. It's important to note that this model should perform equally to the previous one, as they share the same parameters.

Tuning Model/Hyper-Parameters

In the process of fine-tuning the model and its hyper-parameters, several critical parameters were examined, including learning rate, the number of convolutional layers, and the number of epochs. Let's delve into the significance of these parameters and their impact:

Learning Rate:

The learning rate is a critical hyper-parameter that controls the step size during the optimization process. It determines how quickly or slowly a neural network learns. If the learning rate is too high, the model might converge too quickly and overshoot the optimal solution. Conversely, if it's too low, the training process might be too slow or get stuck in local minima. Thus, choosing the appropriate learning rate is essential for successful training.

- **Learning Rates Tested:** 0.001, 0.0015, 0.003, 0.005, 0.01

Results:

Learning Rate	Fold 1	Fold 2	Fold 3	Average	Time
0.0010	55.50%	50.08%	54.75%	53.44%	152.38s
0.0015	52.00%	49.17%	52.42%	51.19%	149.70s
0.0030	46.08%	42.25%	33.92%	40.75%	149.67s
0.0050	33.92%	31.92%	33.33%	33.06%	152.40s
0.0100	33.92%	31.83%	32.58%	32.78%	150.87s

A learning rate of 0.0010 yielded the best results in terms of cross-validation accuracy, and this learning rate will be adopted for subsequent experiments.

Convolutional Layers:

Convolutional layers play a crucial role in feature extraction from images. The number of convolutional layers can significantly impact the model's ability to recognize complex patterns in the data.

Models Tested:

- **Conv1:** This model consists of one convolutional layer with the first layer mapping 3 features to 6.
- **Conv2:** The Conv2 model incorporates two convolutional layers, with the second layer mapping 6 features to 16.
- **Conv3:** In the Conv3 model, there are three convolutional layers, with the final layer mapping 16 features to 32.
- **Conv4:** The Conv4 model includes four convolutional layers, and the final layer maps 32 features to 64.
- **Conv5:** The Conv5 model has five convolutional layers, with the last layer mapping 64 features to 128.

Results:

Model	Fold 1	Fold 2	Fold 3	Average	Time
Conv1	46.58%	53.58%	43.17%	47.78%	153.88s
Conv2	54.00%	59.42%	53.75%	55.72%	151.71s
Conv3	56.75%	58.00%	57.25%	57.33%	153.10s
Conv4	59.00%	59.33%	63.08%	60.47%	152.45s
Conv5	54.00%	48.58%	54.17%	52.25%	154.64s

The experiment demonstrates that the model featuring four convolutional layers (Conv4) achieved the highest cross-validation score 60.47%. Subsequently, this configuration was selected for further training.

Upon applying these optimized parameters to train the model on the entire test dataset, the results exhibited an accuracy of 60%. Notably, the model displayed class-specific accuracies of 52.3% for cherry, 63.8% for strawberry, and 61.7% for tomato. Despite a marginally lower overall accuracy when compared to the previous model trained on the entire training dataset, the model with four convolutional layers should excels in terms of generalization due to a greater cross-validation score and is well-suited for performance on unseen data.

The decision to proceed with the model featuring four convolutional layers was made due to its capacity to provide a more balanced set of predictions across the various classes.

Number of Epochs:

Epochs are an essential component in training a neural network. An epoch refers to one complete cycle through the entire training dataset. During each epoch, the model learns from the data, updates its weights, and refines its performance. The number of epochs is a critical hyper-parameter to be tuned during model development. It influences how many times the model will iteratively adjust its parameters to minimize the loss function and improve its predictive capabilities.

- **Epochs Tested:** 5, 10, 15

Results:

Model	Epochs	Fold 1	Fold 2	Fold 3	Average	Time
Conv4	5	63.67%	58.50%	61.08%	61.08%	164.11s
Conv4	10	71.00%	63.75%	65.17%	66.64%	299.55s
Conv4	15	56.50%	67.33%	62.58%	62.14%	445.00s

The investigation demonstrates that 10 epochs yield the highest average cross-validation score. Subsequently, these parameters were adopted to train a model on the entire test dataset, resulting in an accuracy of 58%. The model displayed class-specific accuracies 40.0% for cherry, 86.5% for strawberry, and 49.7% for tomato.

Although the overall accuracy on the test set has decreased, this model displayed the highest performance in cross-validation 66.64%, indicating its potential for superior generalization to unseen data.

Augmenting the Dataset

Data augmentation is a technique employed to enhance the variety of images within the training dataset, effectively expanding its size. In image classification tasks, having a more extensive dataset is invaluable for training a robust model. Data augmentation introduces variations in the existing images without the need for external data sources, making it an effective strategy for improving model performance.

Random Horizontal Flip:

After fine-tuning the model using cross-validation, a new approach involving data augmentation is implemented to further enhance the model's performance. Specifically, a random horizontal flip with a 50% probability is incorporated into the data loader. This means that for each batch during training, there is a chance that images will be horizontally flipped. Consequently, this augmentation technique effectively doubles the number of image possibilities available for training. These horizontally flipped images introduce additional diversity and noise into the training dataset. Importantly, this augmentation is applied exclusively to the training dataset.

Results:

Including the random horizontal flip transformation results in a significant improvement in accuracy, reaching 70% using 10 epochs. This training process took 148 seconds to complete.

Recognizing the potential for further gains with increased image variation, an experiment with 20 epochs was conducted. The model's accuracy improved to 76%, justifying the use of an extended training duration.

- Accuracy for class: cherry is 72.0%
- Accuracy for class: strawberry is 89.3%
- Accuracy for class: tomato is 66.7%

The model's performance continued to improve with 25 epochs, achieving a 77% accuracy rate.

- Accuracy for class: cherry is 80.0%
- Accuracy for class: strawberry is 79.7%
- Accuracy for class: tomato is 72.7%

This progress encouraged the investigation of 30 epochs, this exploration highlighted the potential for overfitting, as it resulted in a final accuracy of 64%. This observation underscores a crucial consideration in model training: the need to strike a balance between achieving higher accuracy and preventing overfitting.

The trade-off observed between 20 and 25 epochs underscores another critical aspect of model training. In this scenario, extending the training duration by an additional 5 epochs yielded only a marginal 1% increase in accuracy. This observation suggests that the extra time invested in training may not be warranted by the limited gain in accuracy. It emphasizes the significance of determining the optimal training duration to strike a balance between enhancing model performance and preserving robust generalization capabilities.

Other Augmentations

Additional augmentation techniques include random rotations, which apply image rotations within a defined degree range, such as -30 to 30 degrees. Another method is color jitter, which introduces minor random adjustments to image attributes, including brightness, contrast, saturation, and hue. Random crop is a further technique that involves the random extraction of a portion of an image, aiding the model's ability to adapt to variations in object placement within images.

It's important to note that the use of these transformations may be constrained due to limitations in available computing resources. For this experiment, these augmentation techniques were considered unviable due to a lack of computing power.

Transfer Learning ImageNet

Transfer learning is a technique in which a model initially developed for one task is repurposed as a starting point for another task. The underlying idea is that a pretrained model, trained on an extensive dataset such as ImageNet with 1.2 million images, can offer a strong foundation for various other tasks.

In this particular experiment, ResNet-18, an 18-layer neural network pretrained on a large image dataset, is leveraged for transfer learning on the specific task at hand. ResNet-18 is characterized by its architecture, which includes:

- Initial Convolution: The network begins with an initial convolution layer that processes the input image.
- Convolutional Blocks: The network is divided into several convolutional blocks, each typically containing multiple convolutional layers, batch normalization, and ReLU activations.
- Residual Blocks: ResNet introduces residual blocks that incorporate skip connections. These connections add the input of the block to its output, allowing for the direct flow of gradients during backpropagation. This facilitates the learning of identity mapping, preventing vanishing gradients. Various types of residual blocks, including basic and bottleneck blocks, are employed.
- Pooling and Downsampling: After some convolutional blocks, max-pooling or average-pooling layers are used to reduce the spatial dimensions of the feature maps.
- Fully Connected Layer: The network concludes with a fully connected layer, often followed by a softmax activation function for classification tasks.

To adapt ResNet-18 for the current task, the only necessary modification is setting the output of the final dense layer to match the number of classes, which is three in this case.

Summary & Discussion

This report outlines the development and performance of two distinct models: a baseline Multilayer Perceptron (MLP) and a Convolutional Neural Network (CNN) for image classification, specifically distinguishing between cherry, strawberry, and tomato images.

Baseline MLP:

- **Structure:** The baseline MLP consisted of two dense layers with 128 and 3 units, employing Rectified Linear Unit (ReLU) activation functions. Stochastic Gradient Descent (SGD) served as the optimizer with a learning rate of 0.001 and a momentum of 0.9. Cross-Entropy Loss was the chosen loss function.
- **Training time:** Training the baseline MLP with SGD for 5 epochs took approximately 417.21 seconds.

Best CNN (Transfer Learning):

- **Structure:** The best CNN model harnessed the power of transfer learning through the ResNet-18 architecture, pretrained on ImageNet. The final dense layer was adapted to accommodate the three target classes. The optimizer of choice was SGD, and the model was trained for 10 epochs.

- **Training time:** The best CNN model, utilizing SGD with 10 epochs, was trained in approximately 168.9 seconds.

Comparison of Results:

- The baseline MLP achieved an accuracy score of 51% on the test set, while the best CNN, incorporating transfer learning with ResNet-18, delivered a significantly higher accuracy of 92% on the same test set. This is due to ResNet-18 being trained on a significantly larger dataset allowing it to capture complex features.
- Training time for the baseline MLP was extended due to the larger input size (3*300*300), whereas the best CNN, employing pooling and being pretrained produced significanlty shorter training duration (10 epochs), achieved significantly better performance.

Discussion: The marked variations in performance and training time between the baseline MLP and the best CNN can be attributed to the CNN's architectural advantages. The CNN, especially when enhanced with transfer learning from a model pretrained on ImageNet, excels at capturing complex image features and generalizing effectively to new tasks. In contrast, the MLP struggled to capture intricate spatial relationships within the images, leading to lower accuracy. The CNN's shorter training duration & substantial improvement in accuracy, made it the preferred choice for this task.

In summary, the best CNN model, employing transfer learning and the SGD optimizer, significantly outperformed the baseline MLP in terms of classification accuracy. This showcases the potential of CNNs for image classification tasks.

Conclusions and Future Work

In conclusion, this study has provided valuable insights into the performance and potential of two distinct models for image classification, namely, a Multilayer Perceptron (MLP) and a Convolutional Neural Network (CNN).

Conclusions:

1. **CNN Outperforms MLP:** The primary conclusion is that the CNN, particularly when enhanced with transfer learning, significantly outperforms the MLP in the task of image classification. The CNN achieved a remarkable accuracy of 92%, while the MLP lagged with an accuracy of 51%. This substantial performance difference underscores the power of CNNs in capturing complex image features.
2. **Training Time:** Despite the significant accuracy boost, the CNN required less training time compared to the MLP. This trade-off between training time and performance is an essential consideration. It's crucial to select the model that balances accuracy and practicality.
3. **Data Augmentation:** The study highlights the importance of data augmentation techniques, such as random horizontal flipping, which significantly improved the CNN's accuracy from 70% to 77%. This suggests that exploring other data augmentation methods could further enhance model performance.

Pros

- **Utilizing Transfer Learning:** One of the key advantages of transfer learning, as demonstrated in this study, is the substantial reduction in training time. Transfer learning leverages pre-trained models on large datasets (in this case, ImageNet), saving considerable time and computational resources. It allows the model to inherit knowledge about a wide range of features and patterns from the source domain, which can significantly boost performance on the target task.

Cons

- **Black-Box Nature:** A notable drawback of using transfer learning is the black-box nature of the approach. While it delivers remarkable results, it comes at the cost of limited interpretability. Stakeholders, especially

in critical domains like healthcare or finance, may be concerned about deploying models when they lack insight into what goes on beneath the hood. Understanding the features that contribute to predictions and potential biases in the model can be challenging with transfer learning. Therefore, there's a trade-off between accuracy and interpretability.

Future Work:

1. **Optimizers:** While this study compared SGD, RMSprop, and Adam, further exploration of optimizers and hyperparameter tuning could lead to even better performance. Other optimizers like AdaGrad and learning rate schedules should be considered.
2. **Data Augmentation:** Beyond horizontal flipping, exploring additional data augmentation techniques, such as random rotations or color jitter, could contribute to better generalization and improved model performance.
3. **Model Architectures:** This study utilized ResNet-18 for transfer learning. Future work could involve experimenting with different architectures and comparing their performance. The addition of deeper architectures or more specialized pre-trained models might yield even better results.
4. **Ensemble Models:** Combining multiple models into an ensemble could lead to improved accuracy. Combining the strengths of different architectures or training models with various optimizers might enhance the robustness of the classification.

In summary, this study provides a strong foundation for image classification using CNNs, and it highlights several avenues for future research and improvement. It demonstrates the exciting potential of deep learning in solving real-world problems and encourages further exploration of advanced techniques and approaches to enhance classification performance.