UDC: 004.627 https://doi.org/10.15407/jai2025.02.041

M. Mihai[1], T. Filimonova[2]
[1,2]State University of Trade and Economics, Ukraine
19, Kyoto str., Kyiv, 02156
[1]m.mihay_fit_7_21_b_d@knute.edu.ua
[2]t.filimonova@knute.edu.ua
[1]https://orcid.org/0009-0001-4710-0709
[2]https://orcid.org/0000-0001-9467-0141

BINARY IMAGE CLASSIFICATION WITH RESNET

Abstract. The paper discusses the approach to binary image classification with ResNet based deep learning. Its actuality is the automatic image classification process and subject-the use of the methods of data augmentation and in the improvement of the model accuracy-deep neural networks. The goal of this research was to develop and assess the efficiency of using the pre-trained ResNet network for solving the binary classification problem. Such image preprocessing and implementation for the augmentation method and mixed learning were conducted to optimize the classification process. The experimental results discussed point towards integration in preprocessing methods, dynamic image loading, and computational process optimization that drive substantial growth in the quality of classification, which carries great practical value in further explorations in this area.

Keywords: image classification, ResNet, deep learning, binary classification, data augmentation.

Introduction

The relevance of automatic image classification is driven by its widespread use in areas such as medical diagnostics, security systems, and video surveillance. Traditional approaches based on manual feature selection and classical machine learning algorithms often fail to meet accuracy requirements due to high data variability. Modern deep learning technologies, in particular the use of convolutional neural networks, can achieve significant improvements in the ability of models to generalise and effectively analyse complex visual patterns.

The object of research is the process of automatic image classification. Modern approaches to classification include the use of pre-trained architectures, such as ResNet, which allow achieving high accuracy even on complex datasets. However, there are problems related to the limited amount of data and the risk of overtraining, which makes researchers look for new methods to improve the robustness of models.

The subject of this study is the application of data augmentation and deep neural networks to improve model accuracy. The use of augmentation allows to increase the amount of training data and improve the network's ability to generalise, which is important for improving model efficiency. The aim of this paper is to develop and evaluate the effectiveness of using a pre-trained ResNet network to solve a binary classification task.

This article presents an approach to binary image classification using the ResNet model in the PyTorch environment, which takes into account both the historical development of methods and modern technologies. The study focuses on developing an integrated approach that combines the advantages of pre-trained models and methods of expanding the training space, which reduces the risk of overtraining and increases the system's resilience to external influences.

Literature Review

Binary image classification using ResNet architectures has been explored in various domains, demonstrating its effectiveness in handling complex image datasets. ResNet models, known for their deep architecture and skip connections, are particularly suited for binary classification tasks due to their ability to mitigate vanishing gradient problems and enhance feature extraction. The following sections delve into specific applications and methodologies involving ResNet for binary image classification.

ResNet50, among other CNN architectures, has been evaluated for classifying skin cancer from binary images. The use of transfer learning significantly

enhances the model's performance, making it a strong candidate for medical image analysis tasks [1].

The Binary-SE-ResNet model, which incorporates SE-ResNet34, has been used for classifying pneumonia in chest X-rays. By combining original and LBP image features and employing ensemble learning with XGBoost, the model achieved a high accuracy of 97.7% [2].

ResNet18, enhanced with transfer learning, has been applied to classify coronary artery stenosis from CT angiography images. This approach achieved an accuracy of 96.97%, outperforming other models by a significant margin [3].

ResNet50, pre-trained on ImageNet, has been used in conjunction with the Binary Marine Predators Algorithm for feature selection. This method optimizes feature extraction, leading to improved classification accuracy by filtering out irrelevant features [4].

Although not directly using ResNet, the concept of binary neural networks, which utilize binary values for weights and inputs, presents an alternative approach to reduce computational requirements while maintaining reasonable accuracy, as demonstrated with the MNIST dataset [5].

While ResNet models have shown remarkable success in binary image classification, particularly in medical applications, the integration of feature selection algorithms and ensemble learning further enhances their performance. However, exploring binary neural networks could offer a computationally efficient alternative, albeit with potential trade-offs in accuracy.

Materials and methods

The dataset used for the study is the one published on Kaggle: AI vs Human Generated Dataset [6]. This dataset contains examples of both human and artificially generated images. Information about the images is organised in the form of a CSV file containing image paths and corresponding class labels, which allows for binary classification tasks, shown in Table 1.

Table 1. Fragment of the CSV file

file_name	label
train_data/a6dcb93f596a43249135678dfcfc17ea.jpg	1
train_data/041be3153810433ab146bc97d5af505c.jpg	0
train_data/615df26ce9494e5db2f70e57ce7a3a4f.jpg	1
train_data/8542fe161d9147be8e835e50c0de39cd.jpg	0
train_data/5d81fa12bc3b4cea8c94a6700a477cf2.jpg	1
train_data/25ea852f30594bc5915eb929682af429.jpg	0

The study was implemented using the PyTorch platform. Image preprocessing is performed using the torchvision.transforms module, which includes random cropping and resizing to 224 × 224 pixels [7], horizontal mirroring [8], rotation within ±30 degrees [9], and changing colour parameters such as brightness, contrast, and saturation [10]. The images are converted to PyTorch tensors [11] and normalised according to the ImageNet standard [12].

Validation transformations include resizing up to 256 pixels, cropping to 224 pixels, transforming to a tensor, and normalising. The dataset is organised through the developed CustomDataset class, which inherits torch.utils.data.Dataset [13]. It allows you to read a CSV file with information about images and their labels, load images from the disc according to the specified paths, process them through a specific set of transformations, and return image-label pairs in the format of tensors.

For efficient batch processing, we use DataLoader [14], which optimises the data representation in the model. The neural network is trained using the torch.nn module [15], which contains the necessary tools for building a deep neural network. Optimisation is performed using torch.optim [16], which uses SGD [17] and Adam [18] methods to effectively adjust model weights.

To improve performance, torch.amp [19] is used, which provides automatic accuracy

mixing, allowing the use of FP16 instead of FP32 without loss of accuracy, which reduces the load on computing resources. The methods used contribute to the efficient training of the neural network and improve its generalisation ability.

Experiments

As part of the experiments, the ResNet model was trained on a pre-processed dataset in the PyTorch environment using GPU computing power to ensure high speed and efficiency of the optimisation process. A stratified dataset split of 80:20 was implemented to form training and validation samples, which ensured the balance of classes in each subset. Data preprocessing included normalisation, image augmentation to increase the model's robustness to input variability, and balancing methods to avoid possible bias during training. As part of the experiments, the ResNet model was trained on a pre-processed dataset in the PyTorch environment using GPU computing power to ensure high speed and efficiency of the optimisation process. The utilisation of GPU acceleration significantly enhanced computational efficiency, allowing for faster iterations and improved convergence. A stratified dataset split of 80:20 was implemented to form training and validation samples, which ensured the balance of classes in each subset. This method prevented the model from developing biases due to an imbalanced dataset and improved its generalisation capability.

Data preprocessing was a critical step that included normalisation, image augmentation, and various balancing techniques to improve the model's robustness to input variability. Normalisation ensured that the pixel intensity values were standardised, reducing the influence of varying lighting conditions. Image augmentation introduced transformations such as rotation, flipping, and contrast adjustments to create a more diverse dataset, thereby minimising the risk of overfitting. Balancing methods were applied to avoid bias in training, ensuring that the model did not over-prioritise a dominant class at the expense of a less-represented category.

The configuration of the experimental environment is shown in Table 2.

Table 2. Configuration of the experimental environment

Component	Specification
Optimiser	Adam
Initial learning curve	0,001
Loss function	Crossentropy
Number of epochs	30
Batch Size	64
Threshold value of tolerance (Patience)	5

The network parameters were optimised over 30 epochs using the Adam algorithm, known for its ability to quickly adapt the learning coefficients to the peculiarities of the loss function landscape. The Adam optimiser effectively adjusted the step size for each parameter, resulting in faster convergence while avoiding sharp oscillations. The initial training coefficient was set at 0.001 to ensure stable and controlled convergence dynamics. A cross-entropy function was chosen as the loss function, as it is standard for classification tasks and provides a probabilistic interpretation of the model's confidence in its predictions.

To further mitigate overfitting risks, the learning rate was adaptively reduced based on observed fluctuations in the loss function. This approach ensured that training did not continue at an unnecessarily high rate, which could lead to suboptimal generalisation. In addition, early stopping was applied to prevent overfitting to the training set. This mechanism monitored validation performance and halted training if no significant improvement was detected over several epochs, thereby preserving model efficiency while avoiding unnecessary complexity.

During each epoch, the values of the loss function and accuracy were closely monitored on both the training and validation samples. This monitoring enabled real-time assessment of training progress, the detection of potential generalisation issues, and the identification of

phases where the model showed signs of overfitting. The relationship between the rate of decrease of the loss function on the training and validation samples played a key role in evaluating model performance at different stages of learning.

Results

The experimental results demonstrate a gradual decrease in the loss function values on the training set, indicating the effectiveness of the optimisation process. The steady improvement in classification accuracy on the training and validation samples confirms the model's capacity to learn meaningful representations from the dataset. However, a detailed analysis of the validation loss revealed periodic fluctuations, suggesting the possibility of overtraining at certain stages. These fluctuations underline the necessity of fine-tuning hyperparameters and employing more robust regularisation techniques.

The loss function and accuracy trends, as illustrated in Figure 1, provide valuable insights into the learning dynamics. A consistent reduction in training loss reflects the model's increasing ability to distinguish between classes. At the same time, periodic spikes in validation loss highlight the need for additional strategies to stabilise generalisation performance.

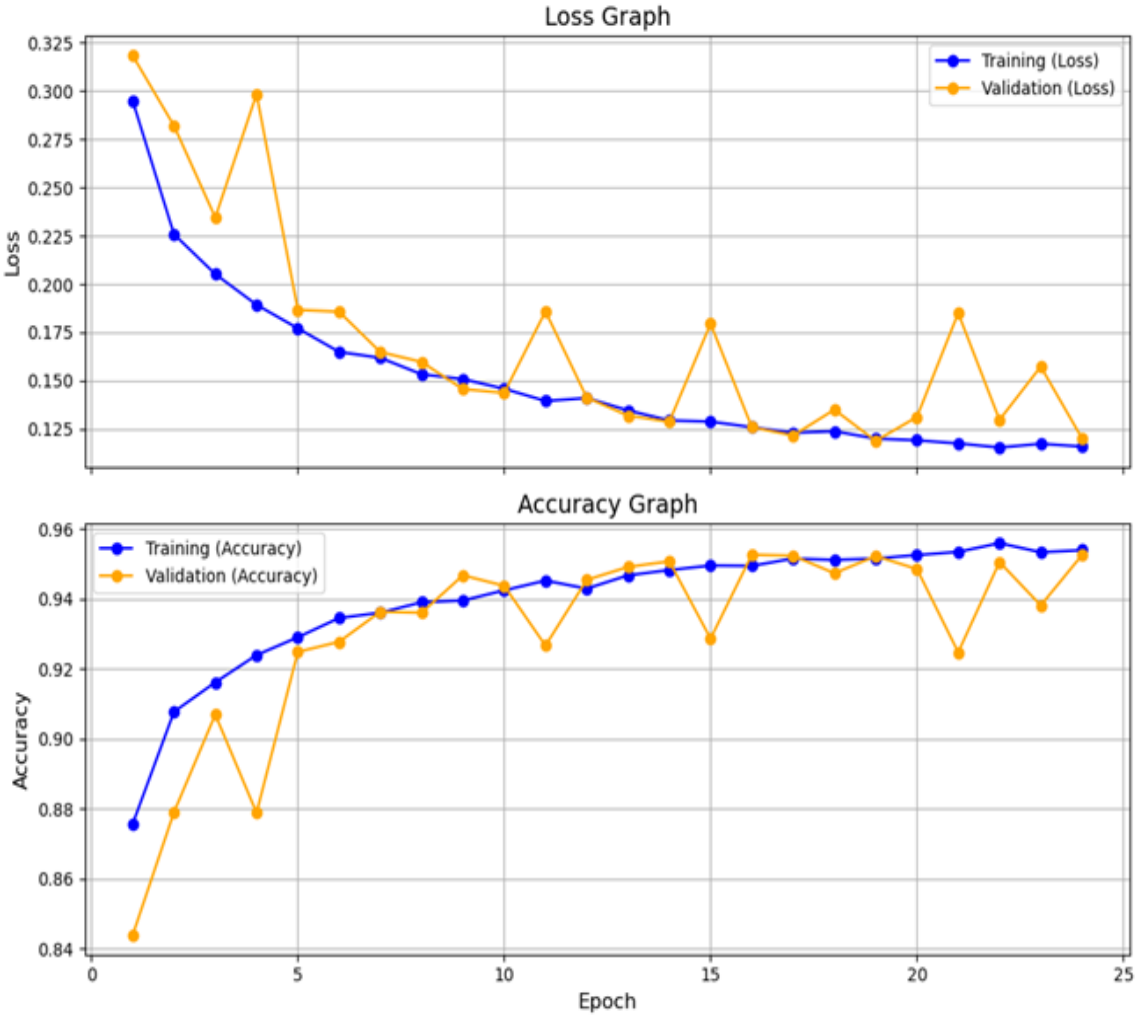

Figure 1. Graphs of changes in the loss function and model accuracy during training and validation

The training process was terminated at epoch 24 due to early stopping, which prevented unnecessary overfitting and ensured optimal model generalisation. The final accuracy of the model reached 95.39% on the training set and 95.25% on the validation set, with loss values of 0.1159 and 0.1201, respectively. These results demonstrate the effectiveness of the training strategy while also

highlighting potential areas for improvement in future implementations.

Discussion

The experimental results highlight the robustness of ResNet18 for binary image classification, especially when combined with dynamic preprocessing and regularization techniques. The validation accuracy of 95.25% is in line with the trend in modern deep learning, where pretrained models achieve high performance even on specialized datasets. This success is consistent with research using deeper architectures such as ResNet50 [1] or hybrid frameworks such as Binary-SE-ResNet34 [2], where differences in task complexity may result in slight changes in accuracy. For instance, distinguishing AI-generated images from human-created ones, as in this work, may inherently involve subtler features compared to medical diagnostics [3], where anatomical patterns are more defined. The systematic use of augmentation (random cropping, rotation, color adjustments) echoes methodologies in [4], reinforcing its universal value in reducing overfitting and enhancing model adaptability.

A critical limitation lies in the dependency on ImageNet normalization parameters, which, while standard, may not optimally align with domain-specific data distributions. This constraint, noted in prior work [5], underscores the need for adaptive normalization strategies in specialized applications. Furthermore, fluctuations in validation metrics, though mitigated by early stopping, suggest unresolved instability, likely tied to static learning rates or insufficient regularization. Comparative studies [6] emphasize that adaptive learning rate schedulers (e.g., cyclical learning rates) could stabilize training, a direction meriting further exploration.

The framework’s practicality extends to domains requiring efficient binary classification, such as content authenticity verification or automated screening systems. However, its deployment in high-stakes environments (e.g., medical diagnostics) demands rigorous external validation to address potential biases from dataset homogeneity. Future research should prioritize architectural comparisons (e.g., ResNet vs. Vision Transformers) to evaluate scalability and generalization. Additionally, integrating explainability tools (e.g., Grad-CAM) could bridge the gap between empirical performance and interpretability, fostering trust in critical applications.

In summary, while the ResNet18-based approach demonstrates state-of-the-art performance, its broader significance lies in balancing computational efficiency with accuracy—a trade-off central to real-world AI adoption. By addressing its sensitivity to hyperparameters and expanding validation across diverse domains, this framework could serve as a versatile foundation for binary classification challenges.

Conclusions

The research thus confirms the applicability of the pre-trained ResNet architecture, combined with appropriate data augmentation approaches for tasks of binary image classification. Such transformations, like random cropping, color and rotation adjustments could enhance the model's robustness about variations in its input, and reduce the overfitting thereof. Despite achieving high accuracy, the fluctuations in validation losses highlighted the need for deeper tuning of hyperparameters, including adaptive learning rate adjustment and additional regularization.

The implementation in PyTorch, using GPU acceleration and mixed precision (FP16), optimized the efficiency of using computing resources, making the proposed approach suitable for practical applications in such fields as medical diagnostics or security systems. The most promising areas for further research include comparing the performance of ResNet with modern architectures (e.g., Vision Transformers) and integrating artificial intelligence methods to analyze model solutions. The results highlight the importance of a holistic approach that combines pre-trained models, data augmentation, and careful optimization of the learning process to achieve high quality classification.

References

1. Korkut, Ş. G., Kocabaş, H., & Kurban, R. (2024). A Comparative Analysis of Convolutional Neural Network Architectures for Binary Image Classification: A Case Study in Skin Cancer Detection. *Karadeniz Fen Bilimleri Dergisi*, *14*(4), 2008–2022. https://doi.org/10.31466/kfbd.1515451
2. Qiu, Y. (2023). *Binary-SE-ResNet Model Research for Binary Classification of Chest X-Ray*. 462–466. https://doi.org/10.1109/ichci58871.2023.10278004
3. Ye, X., Jin, Y., Wang, Z., Feng, N., Mu, L., & Xie, G. (2023). *Image Binary Classification of Coronary Artery Stenosis Based on Resnet18 and Transfer Learning*. https://doi.org/10.1109/ccdc58219.2023.10326590
4. Noori, N. M., & Qasim, O. S. (2023). *ImageNet Classification using ResNet50 and Binary Marine Predators Algorithm*. 340–344. https://doi.org/10.1109/icitams57610.2023.10525313
5. Romeu, E., & Shashev, D. (2024). *Binary Convolution Model for Image Classification*. https://doi.org/10.1109/itnt60778.2024.10582370
6. Detect AI vs. Human-Generated Images: Dataset. URL:https://www.kaggle.com/competitions/detect-ai-vs-human-generated-images/data
7. PyTorch documentation: RandomResizedCrop. URL:https://pytorch.org/vision/stable/transforms.html#torchvision.transforms.RandomResizedCrop.
8. PyTorch documentation:RandomHorizontalFlip. URL:https://pytorch.org/vision/stable/transforms.html#torchvision.transforms.RandomHorizontalFlip.
9. PyTorch documentation: RandomRotation. URL:https://pytorch.org/vision/stable/transforms.html#torchvision.transforms.RandomRotation.
10. PyTorch documentation: ColourJitter. URL:https://pytorch.org/vision/stable/transforms.html#torchvision.transforms.ColorJitter.
11. PyTorch documentation: ToTensor. URL:https://pytorch.org/vision/stable/transforms.html#torchvision.transforms.ToTensor.
12. PyTorch documentation: Normalise. URL:https://pytorch.org/vision/stable/transforms.html#torchvision.transforms.Normalize.
13. PyTorch documentation: Dataset. URL:https://pytorch.org/docs/stable/data.html#torch.utils.data.Dataset.
14. PyTorch documentation: DataLoader. URL:https://pytorch.org/docs/stable/data.html#torch.utils.data.DataLoader.
15. PyTorch documentation: nn module. URL: https://pytorch.org/docs/stable/nn.html.
16. PyTorch documentation: optim. URL: https://pytorch.org/docs/stable/optim.html.
17. PyTorch documentation: SGD. URL:https://pytorch.org/docs/stable/generated/torch.optim.SGD.html.
18. PyTorch documentation: Adam. URL:https://pytorch.org/docs/stable/generated/torch.optim.Adam.html.
19. PyTorch documentation: amp. URL:https://pytorch.org/docs/stable/amp.html.
20. NN for automatic image classification repository URL:https://github.com/Mihai-MP/NN-for-automatic-image-classification

The article has been sent to the editors 26.03.25.
After processing 15.04.25.
Submitted for printing 30.06.25.