PC Parts classification from images with Customized ANN, CNN and Transfer learning Model

*

1st Jahed Hasan
Computer Science and Engineering
Dhaka University of Engineering & Technology,Gazipur
Dhaka, Bangladesh
jahed.duet@gmail.com

2nd Md.Sabbir Hossain Sojib
Computer Science and Engineering)
Dhaka University of Engineering &Technology
Dhaka, Bangladesh
204057@student.duet.ac.bd

***Abstract*—Classification of computer hardware components from images is a crucial task for automation in repair, e-commerce, and inventory management systems. Manual labeling is time-consuming and error-prone. In this paper, we apply three models—Artificial Neural Network (ANN), Convolutional Neural Network (CNN), and Transfer Learning using ResNet-18—to classify PC parts such as GPUs, CPUs, RAM, and motherboards from real-world images. Tailored augmentation and preprocessing techniques were implemented for each model type to improve generalization and training efficiency. Comparative performance analysis reveals that the Transfer Learning model outperforms the customized CNN and ANN in both accuracy and inference stability**

***Index Terms*—ANN, CNN, RNN, Augmentation, Accuracy**

I. Introduction

The growing diversity and scale of computer hardware have led to a demand for automated image classification systems. Identifying PC parts accurately from images has applications in e-commerce platforms, inventory systems, and technical support tools. Conventional approaches relying on human intervention are labor-intensive, error-prone, and unscalable.

With the evolution of deep learning, especially CNNs and Transfer Learning models, computer vision tasks have seen substantial improvements. This paper presents the use of deep architectures—ANN, CNN, and a pretrained ResNet-18—for automatic image-based classification of PC hardware components. Each model was trained on a curated dataset of PC part images, and performance was measured using standard classification metrics. Our objective is to evaluate and compare the accuracy, efficiency, and generalization ability of each model.

II. METHODOLOGY

A. Model Architecture and Flow

The proposed image classification model is structured as a multi-stage pipeline designed for efficient image dataset utilization, training, and evaluation using artificial neural networks (ANN), convolutional neural networks (CNN), and transfer learning with MobileNet. The complete workflow is illustrated in 'Fig. 1'.

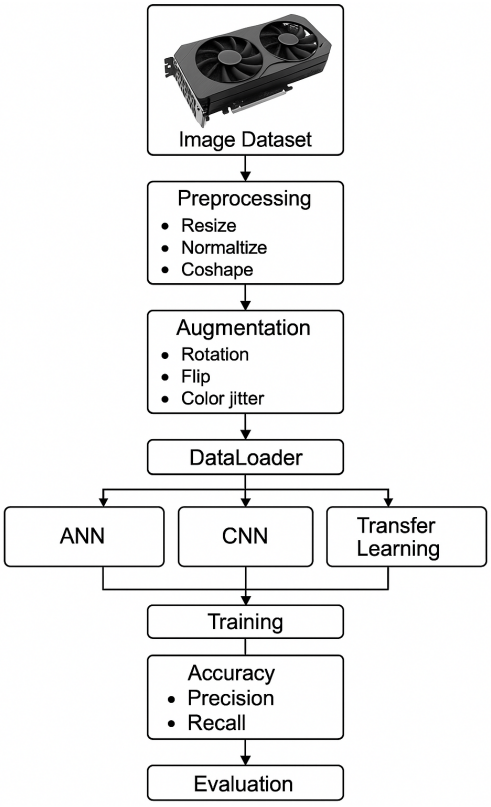


Fig. 1. Work Flow.

B. Datasets

The images are in the ImageNet structure, with each class having its own folder containing the respective images. The images have a resolution of 256x256 pixels.

Identify applicable funding agency here. If none, delete this.

Dataset Details:

- Total number of classes: 14
- Total number of images: 3279
- Resolution: 256x256 pixels
- Image format: JPG

C. Data Augmentation

To enhance the diversity of the training data and improve the robustness of the models, data augmentation techniques were applied to the original dataset obtained from Kaggle [**?**]. The dataset contains images of PC parts categorized into 14 classes.

The following augmentation methods were employed:

- Random rotations within a range of $\pm 15^{\circ}$
- Horizontal and vertical flips
- Zoom-in and zoom-out transformations with scale variations between 0.9 and 1.1
- Random brightness and contrast adjustments
- Small random translations in both x and y directions

These augmentations help the model generalize better by simulating variations commonly encountered in real-world images, thereby reducing overfitting and improving classification accuracy.

D. Dataset Preprocessing

Each model type required specific preprocessing:
Resize:

- ANN & CNN: 128×128
- ResNet-18: 224×224

Normalization:

- CNN: mean=[0.5]*3, std=[0.5]*3
- ResNet-18: ImageNet normalization
- Reshape: All images reshaped to (C, H, W) for PyTorch
- Label Encoding: Applied via ImageFolder class
- Splitting: 80% for training, 20% for validation

E. Data Loader

PyTorch's DataLoader was used for efficient data batching, shuffling, and multithreaded loading.

F. Model Development

In this study, we employed three widely used models over time—Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN), and the ResNet-18 architecture based on transfer learning—to compare and analyze their efficiency and performance.

a)Artificial Neural Network (ANN): Artificial Neural Networks (ANNs) are the fundamental deep learning architectures inspired by the structure and functioning of the human brain. Here, ANN is used as a base classifier with the following architecture Shown in Table I.

TABLE I
ANN MODEL ARCHITECTURE SUMMARY

Layer (Type)	Output Shape	Parameters
Linear-1	[-1, 512]	25,166,336
Linear-2	[-1, 256]	131,328
Linear-3	[-1, 128]	32,896
Linear-4	[-1, 14]	1,806

b)Convolutional Neural Network (CNN): Convolutional Neural Networks (CNNs) are specialized for image data, offering advantages over traditional ANNs through translation invariance, parameter sharing, and effective local feature extraction— making them highly suitable for image classification tasks.It's architecture Shown in Table II.

TABLE II
CNN MODEL ARCHITECTURE SUMMARY

Layer (Type)	Output Shape	Parameters
Conv2d-1	[-1, 32, 128, 128]	896
MaxPool2d-2	[-1, 32, 64, 64]	0
Conv2d-3	[-1, 64, 64, 64]	18,496
MaxPool2d-4	[-1, 64, 32, 32]	0
Conv2d-5	[-1, 128, 32, 32]	73,856
MaxPool2d-6	[-1, 128, 16, 16]	0
Linear-7	[-1, 512]	16,777,728
Linear-8	[-1, 14]	7,182

c)Transfer Learning — ResNet-18

TABLE III
RESNET-18 LAYER CONFIGURATION

Layer (Type)	Output Shape	Parameters
Conv2d	$[-1, 64, 112, 112]$	9,408
BatchNorm2d	$[-1, 64, 112, 112]$	128
ReLU	$[-1, 64, 112, 112]$	0
MaxPool2d	$[-1, 64, 56, 56]$	0
Residual Block 1 (2 x BasicBlock)		
Conv2d × 2 + BatchNorm + ReLU	$[-1, 64, 56, 56]$	110,208
Residual Block 2 (2 x BasicBlock)		
Conv2d × 2 + BatchNorm + ReLU	$[-1, 128, 28, 28]$	382,464
Residual Block 3 (2 x BasicBlock)		
Conv2d × 2 + BatchNorm + ReLU	$[-1, 256, 14, 14]$	1,506,048
Residual Block 4 (2 x BasicBlock)		
Conv2d × 2 + BatchNorm + ReLU	$[-1, 512, 7, 7]$	4,720,320
AdaptiveAvgPool2d	$[-1, 512, 1, 1]$	0
Linear (FC)	$[-1, 14]$	7,182
Total Parameters		**6,735,758**

G. Train Model

The selected architecture is trained using the training dataset, while performance on the validation dataset is continuously monitored to fine-tune hyperparameters and improve learning efficiency.

Training Configuration:

- **Framework:** Implemented using the PyTorch deep learning library.
- **Optimizer:** Adam optimizer with:
  - Learning rate: 0.001
  - Weight decay: 0.001

- **Loss Function:** Cross-entropy loss for multi-class classification.
- **Batch Size:** 64 samples per mini-batch.
- **Data Shuffling:** Enabled during training to improve model generalization.
- **Epochs:** Model trained for 10 epochs.
- **Tracking Metrics:** Accuracy, Precision, Recall, and F1-Score computed using `torchmetrics`.
- **Validation Strategy:** Metrics computed on both training and validation sets at each epoch to monitor model performance.
- **Model Checkpointing:** Best-performing model state saved based on:
  - Minimum validation loss
  - Maximum validation F1-score

H. Evaluation Metrics

The following metrics were used to assess model performance:

- **Confusion Matrix:** Each metric captures a different tradeoff between false positives, false negatives, and overall classification correctness.
  Let the following notations be used:
  - TP (True Positive): Defective product correctly identified as defective
  - TN (True Negative): Non-defective product correctly identified as non-defective
  - FP (False Positive): Non-defective product incorrectly identified as defective
  - FN (False Negative): Defective product incorrectly identified as non-defective
- **Accuracy:** Proportion of correctly classified samples over the total number of samples.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

- **Precision (Positive Predictive Value):** Proportion of predicted defective products that are actually defective.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

- **Recall (Sensitivity or True Positive Rate):** Proportion of actual defective products that were correctly identified.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

- **F1-Score:** Harmonic mean of precision and recall, balancing both metrics.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

III. RESULTS AND DISCUSSION

a)The train and validation metrics curve visualization of Artificial Neural Network Model are given in the below "Fig. 2".

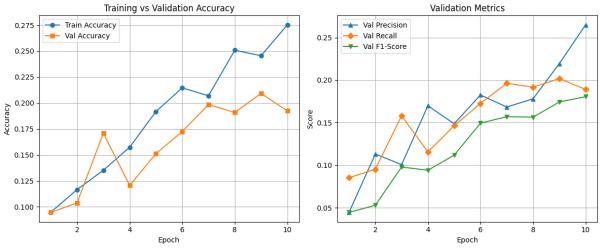


Fig. 2. ANN Training vs Validation Metrics.

Confusion Matix of test dataset using bast train and validation model is shown in "Fig. 3" for better understanding of the performance

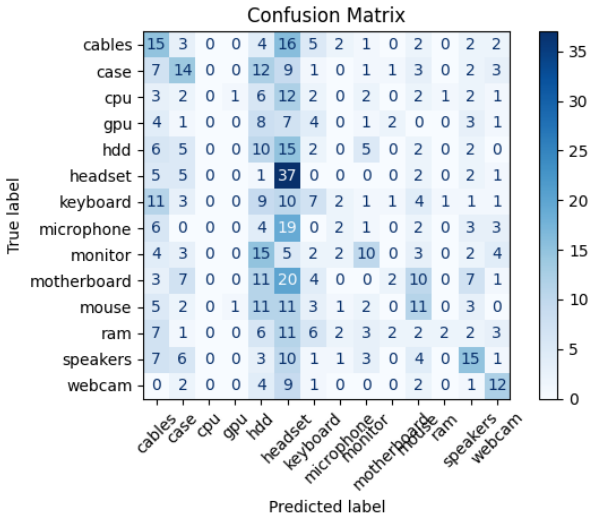


Fig. 3. Confusion Matrix of test data Evaluation.

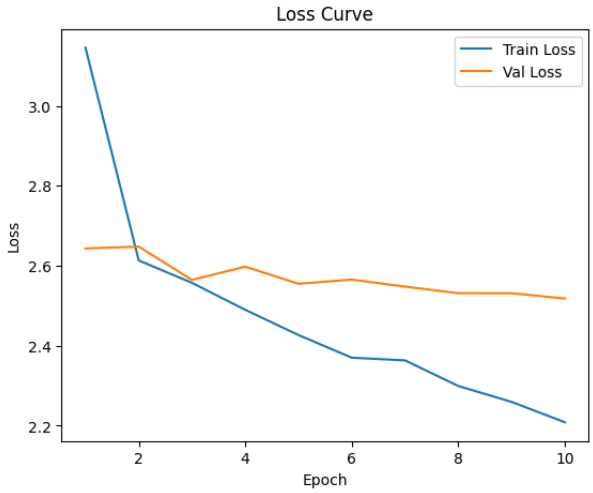


Fig. 4. ANN Lose Curve.

b) CNN: The train and validation metrics curve visualization of Convulational Neural Network Model are given in the below "Fig. 5".

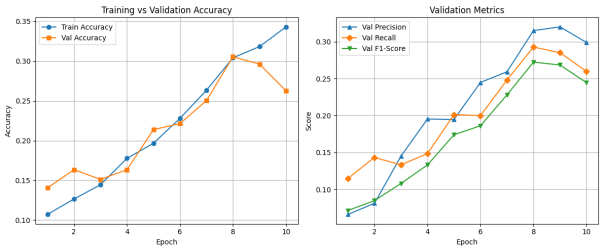


Fig. 5. CNN Training Vs Validation metrics

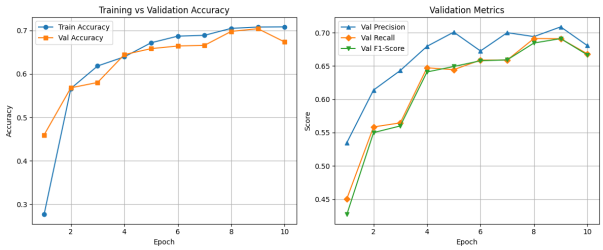


Fig. 8. ResNet-18 Training Vs Validation Metrics

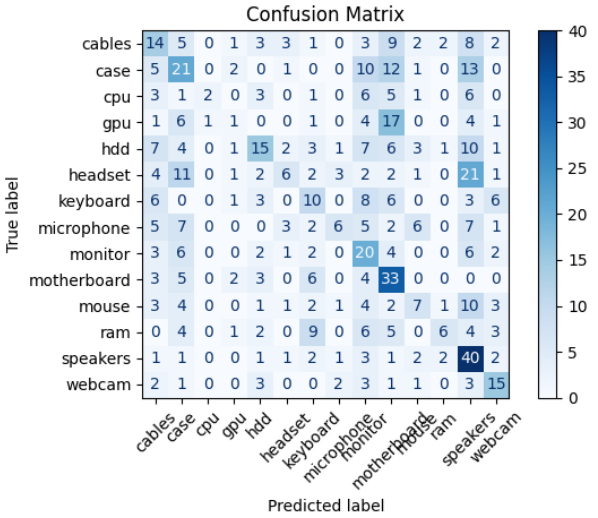
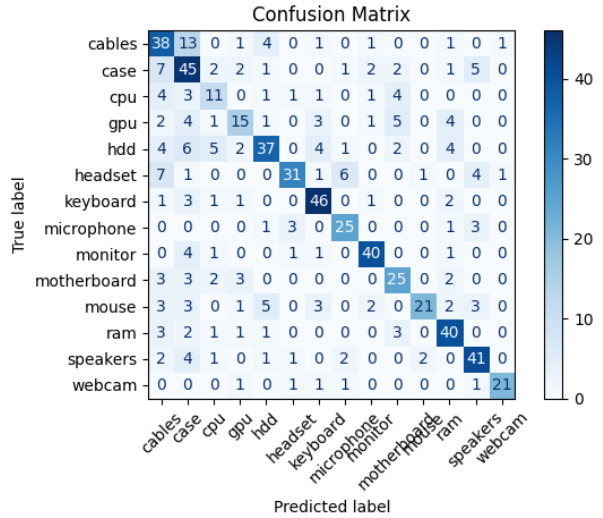


Fig. 6. Confusion Matrix of test data Evaluation



Matrix of test Data Evaluation

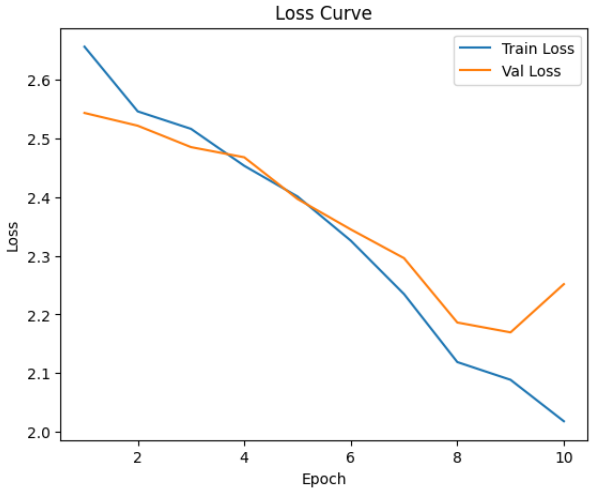


Fig. 7. CNN Loss Curve.

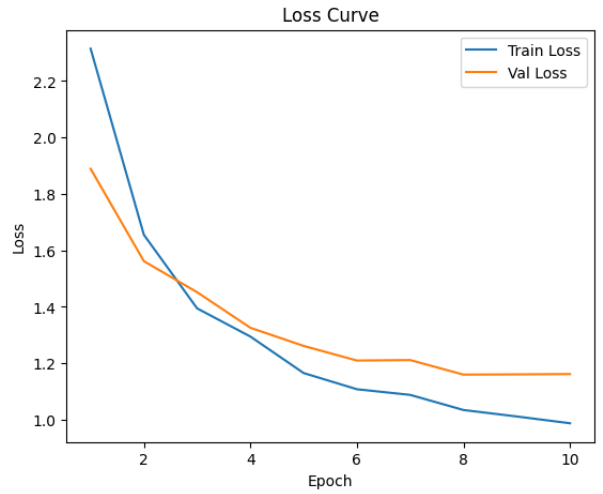


Fig. 9. ResNeT-18 LOSS FUNCTION

c) Transfer Learning: ResNet-18: The train and validation metrics curve visualization of ResNet-18 Transfer Learning Pretrained model are given in the below "Fig. 8".

TABLE IV
BEST VALIDATION METRICS FOR EACH MODEL

Model	Accuracy	Precision	Recall	F1-Score
ANNModel	0.209160	0.219381	0.201806	0.174316
SimpleCNN	0.305344	0.314868	0.292670	0.272264
TL (ResNet-18)	0.703817	0.708709	0.690938	0.691103

1) Performance Comparison: ANNModel shows the weakest performance among the three, with an accuracy of 20.92%, precision of 21.94%, recall of 20.18%, and an F1-score of 17.43%. These low values suggest that the model struggles to generalize and classify effectively.

SimpleCNN performs better than ANNModel, achieving an accuracy of 30.53%, precision of 31.49%, recall of 29.27%, and F1-score of 27.23%. This indicates that using a simple convolutional neural network improves performance, but it still lacks sufficient capability for high accuracy.

Transfer Learning with ResNet-18 (TL) outperforms the other models significantly, with an accuracy of 70.38%, precision of 70.87%, recall of 69.09%, and F1-score of 69.11%. These results demonstrate the effectiveness of leveraging a pre-trained deep learning model, particularly ResNet-18, in achieving high classification performance.

IV. Acknowledgment

This project was completed as part of the Neural Networks and Pattern Recognition Sessional course. The author(s) would like to thank Professor Dr. Amran Hossain sir And Assistant Professor Dr. Umme Fawzia Rahim mam the course instructor, for their insightful guidance and learning throughout lab. The experiments and model implementations were conducted using the PyTorch deep learning framework and on GPU in COLAB environment.

V. CONCLUTION

We presented a comparative study of three deep learning models-ANN, CNN, and Transfer Learning via ResNet-18 for PC part classification. Results suggest that CNN and Transfer Learning significantly outperform ANN. ResNet-18, being pretrained and efficient, emerges as the most effective solution. Future work can explore deployment on edge devices and real-time integration in assembly or repair stations.