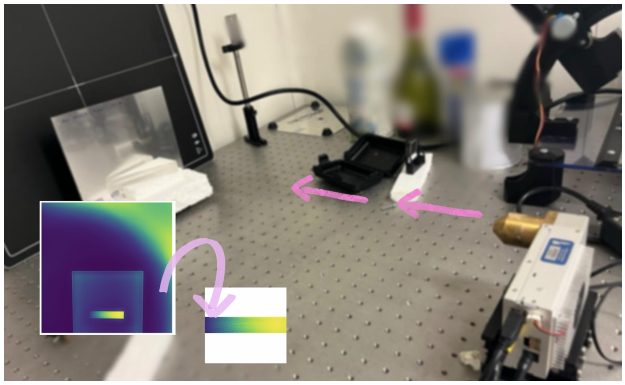
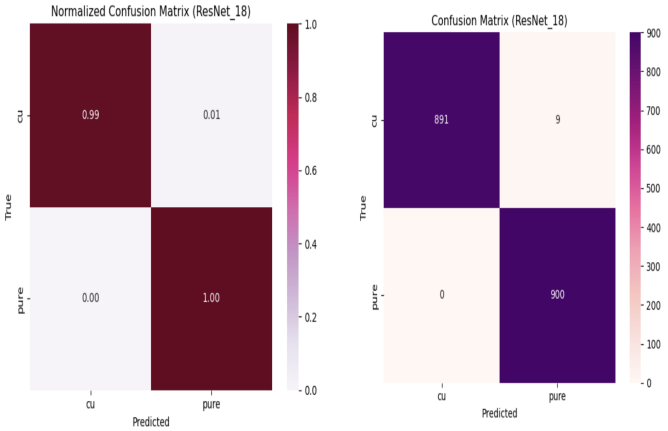
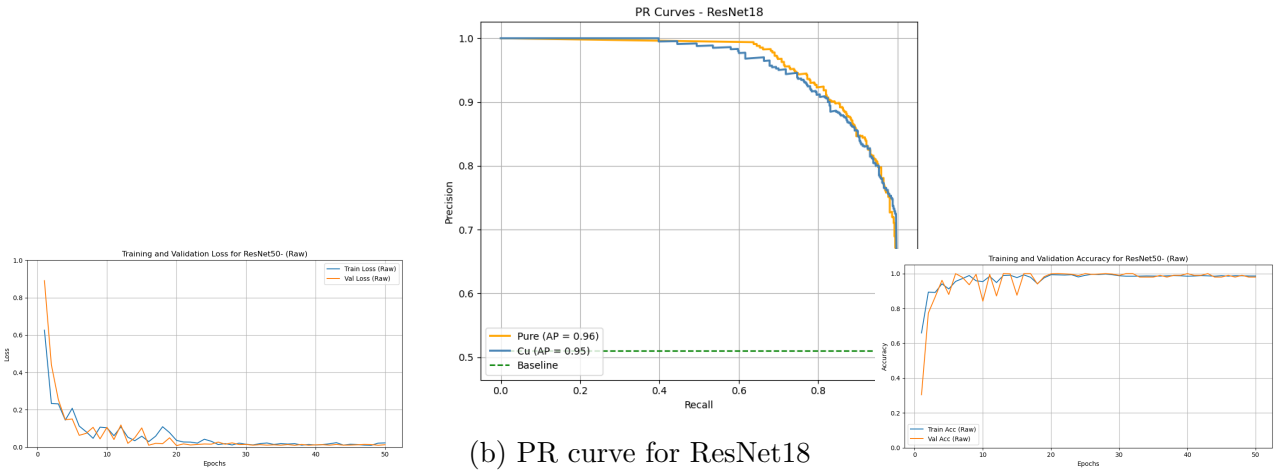
Last Update : August 4, 2025

Scientific Project II - EL061A-"Advanced Material Classification Using X-ray Imaging and Deep Learning Models: A Comparative Study for Scientific Exploration"



(a) The diagram of "The experimental setup in the laboratory with x-ray tube and flat panel detector.".





(b) PR curve for ResNet18

Figure 1: Performance of CNN models

Chethana Johans
(noch2300@student.miun.se)
Mid Sweden University, Sundsvall

Contents

List of Figures

List of Tables

1 Abstract

X-ray-based materials characterization continues to evolve as a powerful approach for understanding electronic and atomic structures, supporting both scientific research and industrial applications. Although Project 01 focused on classifying X-ray images using YOLO[1], Project 02 advances the methodology by shifting to absorption-driven spectral analysis with deep learning-based classification.

In this study, we examine two aluminum-based materials, pure aluminum (Al 1050) and aluminum-copper alloy (Al 2017) using laboratory-based X-ray imaging at 50 keV and 300 μA. Samples were prepared in three thicknesses, 1 mm, 2 mm, and 5 mm. The overall wedge size was increased from 5 cm to 7 cm to enhance spatial resolution and absorption contrast.

To classify and differentiate these materials based on their X-ray absorption behavior, we implemented and compared multiple CNNs[2] architectures. Each model was trained using carefully tuned hyper parameters and evaluated on its ability to resolve subtle spectral differences linked to thickness and composition. To address class imbalance in the dataset, loss functions such as focal loss [16] and class-balanced loss [6] were applied. This strategy significantly improved the robustness of classification in different material types and thickness variations.

Our results demonstrate that CNN-based spectral analysis, combined with tailored loss functions and optimized imaging parameters, provides a scalable and accurate solution for material identification. This work highlights the practical integration of deep learning with laboratory X-ray imaging, paving the way for more consistent, automated characterization pipelines in materials science [4].

Keywords: X-ray Imaging, Deep Learning, CNN, Aluminum Alloys, Spectral Classification, Material Characterization, Absorption Analysis.

[1] You Only Look Once

[2] Convolutional Neural Networks

2 Introduction

In modern materials science, accurate classification of materials is critical for applications ranging from industrial quality control to the development of advanced engineering materials. Identifying materials based on their X-ray absorption characteristics ensures safety, performance, and reliability in sectors such as aerospace, automotive, and electronics. Addressing the need for efficient, scalable classification models remains a central challenge [4].

This study uses X-ray absorption spectroscopy (XAS) to investigate the atomic and electronic structures of materials, providing information on the composition, oxidation states, and coordination environments [3]. Building on the groundwork established in Project 01, which utilized YOLO for real-time object detection. we advance the methodology by incorporating Convolutional Neural Networks (CNNs) for spectral classification. Unlike YOLO, which focused on object localization, CNNs enable deeper feature extraction from X-ray imaging data to enhance classification precision [15].

The data set includes aluminum alloys of varying compositions, pure aluminum (Al 1050) and aluminum-copper (Al 2017) in three thickness levels (1 mm, 2 mm, and 5 mm) and two wedge sizes (5 cm and 7 cm). The study investigates how changes in thickness and wedge size influence absorption behavior, contributing to improved spectral differentiation.

To address class imbalance and improve robustness, we apply the tailed method alongside advanced loss functions tailored for long-tailed distributions, such as focal loss [16] and class-balanced loss [6]. This ensures fair representation and accurate classification even when certain material types are underrepresented.

Through the integration of CNNs with domain-specific data and loss optimization, this work presents a scalable framework for automated material classification. The approach offers practical benefits for industrial inspection and scientific research, supporting real-time decision-making and the development of high-performance materials.

Chapter 2.2 details the CNN architectures evaluated, optimization strategies employed, and the implementation of the tailed method for enhanced classification accuracy.

2.1 Contributions

This study advances X-ray-based material classification by integrating multiple Convolutional Neural Network (CNN) architectures, including ResNet, EfficientNet, VGG, and custom-designed networks, combined with a tailed method to handle long-tailed data distributions effectively. The research explores the impact of varying sample thicknesses (1 mm, 2 mm, and 5 mm) and increased wedge size (from 5 cm to 7 cm) on classification performance, enabling more detailed material differentiation between pure aluminum (Al 1050) and aluminum-copper (Al 2017) alloys.

Through comprehensive hyperparameter optimization and evaluation of diverse CNN models, the approach achieves improved accuracy and robustness in classifying complex absorption spectra, overcoming challenges related to class imbalance and spectral similarity. The study demonstrates the scalability and efficiency of AI-driven techniques in material science, providing a practical framework for enhanced industrial quality control and materials research.

These contributions collectively extend the capabilities of automated X-ray image analysis for advanced alloy characterization [35, 10, 34].

3 Background and Model Architecture

3.1 Explanation of Models

This project evaluates several deep learning architectures, ResNet, EfficientNet, VGG, and custom-designed Convolutional Neural Networks (CNNs) for classifying materials based on their X-ray Absorption Spectroscopy (XAS) data. Each architecture offers distinct advantages for extracting and interpreting complex spectral features.

ResNet (Residual Network) employs residual connections to mitigate the vanishing gradient problem, enabling the training of very deep networks. This architecture excels in capturing intricate patterns in high-dimensional data, making it well-suited for distinguishing subtle differences among alloy compositions [10]. Variants such as ResNet-50 and deeper have demonstrated superior performance at the cost of increased computational requirements.

EfficientNet achieves an effective balance between accuracy and efficiency by uniformly scaling network depth, width, and resolution through a compound scaling method. Incorporating advanced techniques like swish activations and depthwise separable convolutions, EfficientNet delivers high accuracy with fewer parameters, ideal for resource-constrained scenarios and real-time applications [34].

VGG architectures (e.g., VGG-16, VGG-19) utilize simple stacks of 3×3 convolutional layers to capture complex visual features. Although computationally intensive, VGG networks remain reliable for image classification, particularly when fine-tuned on domain-specific datasets [28].

Feedforward Neural Networks (FNNs) represent the foundational architecture in neural networks, consisting of layers of neurons where information moves strictly in one direction from input to output without cycles or loops. Though simpler than CNNs, FNNs can effectively model nonlinear relationships in data when properly designed with sufficient depth and neurons. In the context of XAS material classification, FNNs can serve as a baseline model or be used in combination with feature extraction techniques to classify spectral data [1]. While they generally lack the spatial feature extraction power of CNNs, their straightforward architecture allows for faster training and interpretability, which can be beneficial in certain material characterization tasks.

Custom Neural Networks provide flexibility to tailor the architecture to dataset size, complexity, and specific classification challenges such as long-tailed class distributions. By adjusting layer configurations, filter sizes, and activation functions, custom CNNs can be optimized to enhance classification accuracy and address domain-specific requirements [9].

Through comparative analysis, this study aims to identify the most effective architecture or combination thereof for robust material classification using XAS data.

3.2 Model Architecture

Convolutional neural network (CNN) architectures vary widely in depth, complexity, and design philosophy, each optimized for specific tasks and computational constraints. The ResNet family, introduced by He et al. [10], employs residual learning with skip connections that alleviate vanishing gradient problems in deep networks. ResNet18 and ResNet34 consist of basic residual blocks, with 18 and 34 layers, respectively, making them suitable for moderately complex image classification tasks. In contrast, ResNet50 utilizes bottleneck blocks, comprising three convolutional layers per block, to achieve greater depth and efficiency without excessive computational cost.

The VGG16 model, proposed by Simonyan and Zisserman [28], takes a different approach with a straightforward deep stack of 3×3 convolutional layers without skip connections, resulting in a simpler yet highly effective architecture that remains a baseline for many vision tasks. At the other end of the spectrum, EfficientNet, developed by Tan and Le [34], introduces compound scaling that simultaneously scales the depth, width, and resolution of the network. This approach enables EfficientNet models to achieve state-of-the-art performance with significantly fewer parameters and computational resources.

In contrast to CNNs, fully connected feedforward neural networks (FNNs) or multilayer perceptrons (MLPs) consist solely of dense layers and are generally less suited for raw image data but can be effective for tabular or extracted feature inputs [25]. These networks rely on activation functions such as ReLU to introduce non-linearity, enabling the learning of complex mappings.

Together, these architectures provide a diverse toolkit for tackling a wide range of computer vision problems, from lightweight models for limited hardware to highly efficient networks optimized for performance and scale.

3.3 Model Configuration

In training convolutional neural networks (CNNs) for material characterization, careful tuning of hyperparameters plays a pivotal role in optimizing model accuracy, convergence speed, and generalization capability. These hyperparameters span several categories, including learning rate settings, optimizer choices, data augmentation strategies, model-specific configurations, and regularization techniques.

One of the most critical hyperparameters is the **learning rate**, which determines the magnitude of updates applied to the model's weights during each step of training. A learning rate that is too high can lead to divergence or instability, while a learning rate that is too low can result in slow convergence. Typical values range from 0.0001 to 0.01. To address these challenges, **learning rate scheduling** strategies such as StepLR, Exponential Decay, and Cosine Annealing are employed to adjust the learning rate dynamically throughout training, thereby enhancing performance and convergence stability [30, 17].

Optimizers are another crucial aspect of model training. Among the most widely used are Adam, Stochastic Gradient Descent (SGD), and RMSprop [13]. Adam is particularly popular for its adaptive learning rate mechanism, while SGD with momentum is known for its simplicity

and strong generalization properties [33]. Additionally, momentum and weight decay are often incorporated to accelerate learning and reduce overfitting, respectively.

To improve the robustness and generalization of CNNs, **data augmentation** techniques are applied to increase the variability of the training data. Common transformations include horizontal and vertical flipping, rotation, scaling, and color jittering. These augmentations help the model learn invariances in orientation, lighting, and scale. More advanced strategies such as *Cutout* and *Random Erasing* involve masking out random regions of the image during training to encourage the model to rely on multiple features instead of specific localized patterns [27, 7]. Techniques like Gaussian noise addition, random cropping, and elastic transformations further simulate real-world variability, which is particularly beneficial when training on limited or noisy datasets.

In terms of model-specific parameters, CNNs such as ResNet typically require fixed input image sizes—for instance, 224×224 pixels. The **batch size**, which determines the number of samples processed simultaneously, commonly ranges from 16 to 64 depending on hardware limitations. The **number of epochs** is another important setting, often selected between 50 and 200 to ensure sufficient exposure to the training data without overfitting [10]. Additionally, hyperparameters such as the Intersection over Union (IoU) threshold are set to control how bounding boxes are evaluated during training, especially for detection tasks.

Regularization methods are also vital to prevent overfitting. One such method is *dropout*, where a fraction of neurons is randomly deactivated during training to encourage redundancy and robustness in feature learning [32]. Another technique, *label smoothing*, slightly adjusts the target class probabilities away from hard 0 or 1, which helps mitigate overconfidence in predictions and has been shown to improve generalization [21].

Finally, **training control mechanisms** such as gradient clipping and early stopping are used to ensure training remains stable. Gradient clipping prevents the gradients from exploding by capping their values within a predefined threshold, while early stopping halts the training process once performance on a validation set ceases to improve, thus saving computational resources and avoiding overfitting.

Together, these hyperparameter settings and techniques form the foundation for effectively training CNN models tailored for material characterization tasks, ensuring reliable and interpretable results across varying input conditions and dataset limitations.

3.4 Hyperparameter Configuration Across CNN Architectures

Building upon the general framework for CNN training, this section summarizes specific hyperparameter settings used across various architectures explored in this study. Although the foundational strategies, such as scheduling the learning rate, regularizing, and selecting optimizers, remain consistent, their specific values and combinations are tailored to the demands of each model.

ResNet18 utilizes the Adam optimizer and ReLU activation function, with a learning rate of 0.001 and weight decay of 0.0001. This configuration offers fast convergence and works well with relatively shallow architectures.

ResNet34, a deeper variant, adopts Stochastic Gradient Descent (SGD) and a Leaky ReLU

activation function to maintain gradient flow. The learning rate is set higher at 0.01, while the weight decay remains at 0.0001.

ResNet50, being even more complex, benefits from RMSprop for stability with noisy gradients and employs the GELU activation function. A lower learning rate of 0.0005 allows for finer updates, with weight decay increased to 0.0005 to counter potential overfitting.

VGG16 uses SGD with momentum and ReLU. Its learning rate is set to 0.01, consistent with its relatively straightforward architecture, and a modest weight decay of 0.0001 is applied.

EfficientNetB0 leverages either RMSprop or Adam, paired with the Swish (SiLU) activation function. With a learning rate of 0.002 and weight decay of 0.0001, this configuration supports EfficientNet's compound scaling strategy.

These configurations were either manually tuned or optimized using Population-Based Training (PBT), a method that dynamically adjusts and evolves hyperparameters during training [12]. This ensures both robustness and adaptability across different datasets and model scales.

Table 1: Hyperparameter configurations used for different CNN and FNN architectures.

Model	**Optimizer**	**Activation**	**Learning Rate**	**Weight Decay**
FNN	SGD	ReLU	0.01	0.001
ResNet18	Adam	ReLU	0.001	0.0001
ResNet34	SGD	Leaky ReLU	0.01	0.0001
ResNet50	RMSprop	GELU	0.0005	0.0005
VGG16	SGD + Momentum	ReLU	0.01	0.0001
EfficientNetB0	RMSprop / Adam	Swish (SiLU)	0.002	0.0001

3.5 Evaluation Metrics

Model performance was assessed using standard classification metrics, including accuracy, precision, recall, and F1-score, which together offer insights into both overall and class-specific effectiveness [31]. To evaluate probabilistic confidence in predictions across multiple classes, Area Under the Receiver Operating Characteristic Curve (AUC-ROC) was calculated for each class. Additionally, confusion matrices were used to visualize classification results and identify misclassifications across different alloy–thickness–wedge combinations. These metrics collectively provided a comprehensive assessment of the CNN models' reliability and generalization capability in material classification tasks. These metrics were computed for each class and averaged (macro/micro) as needed to evaluate overall model effectiveness.

- **Precision** – Measures the accuracy of positive predictions.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1}$$

- **Recall (Sensitivity or True Positive Rate)** – Measures the ability of the model to identify all positive instances.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2}$$

- **Accuracy** – Measures the overall correctness of the model.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

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

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

- **Specificity (True Negative Rate)** – Measures the ability of the model to identify all negative instances.

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{5}$$

3.5.1 Log Loss (Cross-Entropy Loss)

Log Loss, also known as Cross-Entropy Loss, is a widely used evaluation metric in classification tasks where models output probability distributions. It quantifies the difference between the predicted class probabilities and the actual class labels, penalizing overly confident incorrect predictions. A lower Log Loss indicates better model performance.

For a multi-class classification task, the formula is given as:

$$\text{LogLoss} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} y_{ic} \log(\hat{y}_{ic}) \tag{6}$$

where y_{ic} is a binary indicator (1 if sample i belongs to class c, 0 otherwise), and $\hat{y}_{ic}$ is the predicted probability of sample i belonging to class c. This metric is particularly useful for evaluating probabilistic predictions in deep learning models such as CNNs and helps assess how well the model distinguishes between similar material types like Al 1050 and Al 2017 [20].

4 Materials and Methods- Training CNN for Image Classification

4.1 Dataset Collection Setup

X-ray Absorption Spectroscopy (XAS) is a powerful technique for investigating the local structure and electronic properties of materials by measuring how X-ray absorption varies with energy [14, 22]. While XAS is especially effective in analyzing metallic alloys at the atomic level, it requires synchrotron radiation sources and energy-resolving detectors that are typically unavailable in standard laboratory environments.

In this study, we adopt an X-ray transmission imaging method [19] as a more accessible alternative to XAS. Our setup captures transmitted X-ray intensities through alloy filters and wedge-shaped plate, now extended to a maximum thickness of 7 cm (compared to 5 cm in earlier configurations). This approach facilitates analysis of material-dependent attenuation behavior, influenced by both composition and thickness. By examining intensity variations in the transmitted images, we infer material characteristics such as elemental composition and structural heterogeneity.

Analysis of Material Attenuation Using X-ray Imaging and Deep Learning (CNN-based)

- X-ray tube operating at 50 keV and 300 μA.
- Flat panel detector capturing transmitted X-ray intensities across different regions of alloy samples.
- CNN models such as ResNet, VGG, and EfficientNet trained using PyTorch for classification of attenuation patterns and spatial features.

The experimental setup involves positioning the alloy sample filtera and wedge between an X-ray tube and a flat panel detector. The detector records 2D transmitted intensity maps. Prior to acquisition, calibration was performed to ensure consistent intensity readings, tube voltage stability, and alignment between source and detector.

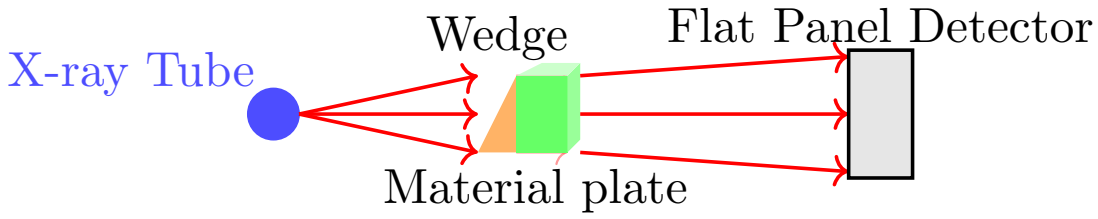


Figure 2: Experimental setup: X-rays pass through a triangle wedge and directly into a material plate before reaching the flat panel detector.

To emulate energy-dependent absorption behavior, the tube voltage was incrementally varied, indirectly approximating absorption edge responses of constituent elements [5]. Although

the detector lacks spectral resolution, voltage variation enables a qualitative assessment of how materials absorb X-rays at different energies.

To characterize spatial heterogeneity, the sample was scanned laterally to expose the beam to different microstructural regions. Transmission images were analyzed to identify contrasts corresponding to phases, grain boundaries, and defects [8], which appear as spatial variations in intensity. CNN-based models effectively learned to distinguish these subtle features, benefiting from the rich contrast provided by the extended 7 cm wedge filter range.

4.2 Dataset Collection and Description

The dataset for this study was constructed using X-ray transmission imaging of two aluminum based materials, commercially pure aluminum (Al 1050) and an aluminum-copper alloy (Al 2017). Each material was fabricated in three standardized thicknesses 1mm, 2mm, and 5mm and imaged using wedge 7 cm. This configuration enabled a systematic investigation into how both material composition and geometry influence X-ray attenuation profiles.

All images were acquired using a fixed laboratory X-ray system operating at 50 keV and 300 μA. The imaging setup remained consistent across all samples to ensure comparable contrast and intensity values. To ensure balanced model training, each combination of material, thickness, and wedge length was equally represented in the dataset.

Prior to training, all images were converted to grayscale, normalized, and histogram-equalized to enhance contrast. Images were resized to a uniform resolution compatible with convolutional neural network (CNN) input dimensions. Additionally, data augmentation techniques such as horizontal flipping, rotation, and random scaling were applied to increase model generalizability and reduce overfitting during training.

4.3 Training Strategy

The dataset was split into training (70%), validation (15%), and testing (15%) subsets, as recommended in standard deep learning workflows [9]. Images were resized to match model input dimensions (e.g., 224×224 for ResNet and EfficientNet) and underwent preprocessing steps including grayscale normalization and histogram equalization. To improve generalization, data augmentation methods like flipping and rotation were applied [27]. Transfer learning was used using pre-trained weights on ImageNet for ResNet and VGG architectures [34], while custom CNNs were trained from scratch. Optimization used Adam optimizer with early stopping to prevent overfitting, and hyperparameters were fine-tuned using grid search.

5 Results and Analysis

This section presents the performance evaluation of various convolutional neural network (CNN) architectures: ResNet18, ResNet50, VGG16, EfficientNetB0 and a custom feedforward neural network (FNN) trained in a unified data set consisting of X-ray transmission images of aluminum-based materials at thicknesses of 1 mm, 2 mm, and 5 mm. All samples were imaged with wedge filters up to 7 cm to increase variability and improve generalization.

The combined dataset provided a diverse representation of attenuation patterns, enabling effective model training across thickness variations. Evaluation metrics included classification accuracy, training-validation loss curves, confusion matrices (including normalized versions), precision-recall (PR) curves, and receiver operating characteristic (ROC) curves.

Among the models, **ResNet50** achieved the best overall performance, with high classification accuracy, stable convergence behavior, and strong separability in PR and ROC curves. **ResNet18** also performed well, offering a lightweight alternative with only a slight reduction in accuracy.

In contrast, **VGG16** and the custom **FNN** experienced overfitting despite regularization and data augmentation. These models showed unstable validation losses and lower performance on unseen data. **EfficientNetB0** performed moderately well, with smoother training curves than VGG16 and better generalization than FNN, though it did not outperform the ResNet variants.

Combining data across all three thickness levels contributed to model robustness by enriching the dataset with a wider range of attenuation patterns. However, architectural choice remained a key determinant of performance. ResNet-based models demonstrated strong feature extraction capabilities, making them particularly suitable for this classification task.

Overall, **ResNet50** proved to be the most effective model, leveraging both depth and transfer learning to deliver the highest accuracy and most consistent evaluation results across the heterogeneous dataset.

5.1 ResNet18 Results

The ResNet18 model demonstrated strong performance in classifying X-ray images of aluminum-based materials. As shown in the confusion matrix, the model achieved high classification accuracy with minimal misclassifications 891 correct predictions for the Cu class and 900 for the pure Al class, with only 9 instances of misclassification. The normalized confusion matrix further confirms this, showing a precision of 0.99 for the Cu class and 1.00 for the pure class, indicating excellent class discrimination capability.

Training and validation accuracy curves reveal that the model converged rapidly, reaching above 95% accuracy within the first few epochs and maintaining stability throughout training. Correspondingly, the loss curves indicate a smooth and consistent decrease in both training and validation loss, with minimal overfitting observed, suggesting a well-generalized model[9].

The Precision-Recall (PR) curves further support the model's robustness, showing high average precision (AP) scores of 0.96 for the pure class and 0.95 for the Cu class. Similarly, the Receiver Operating Characteristic (ROC) curves display strong separability with Area Under

the Curve (AUC) values of 0.96 for both classes, indicating a low false positive rate and high true positive rate across thresholds.

Overall, ResNet18 proved to be a reliable architecture for this task, balancing training efficiency and predictive performance with minimal signs of overfitting[10].

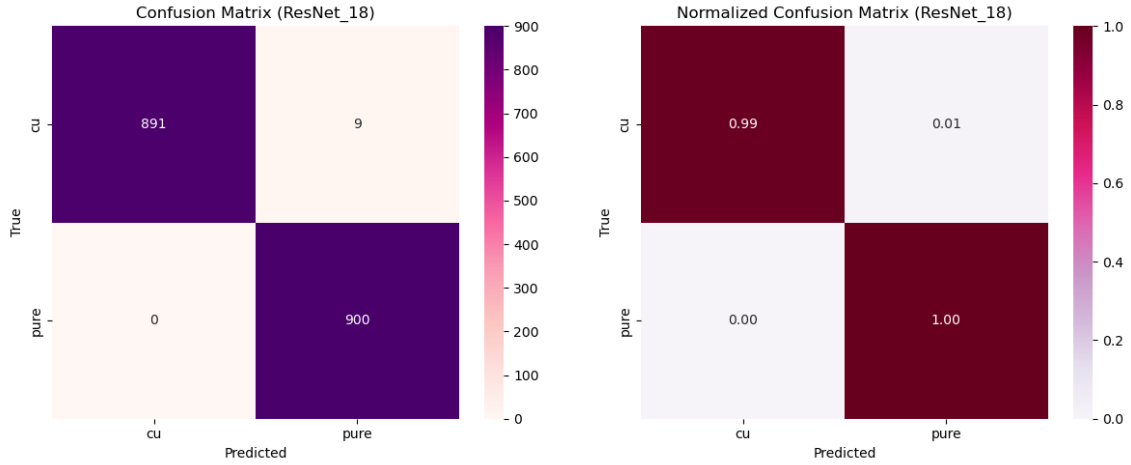


Figure 3: ResNet18- Confusion Matrix and Normalized Confusion Matrix

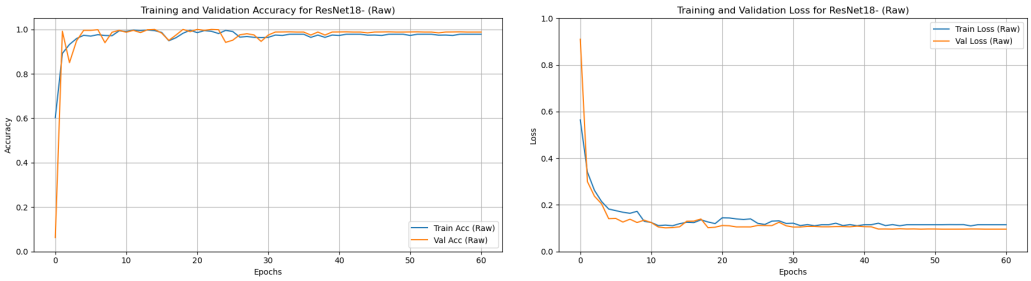


Figure 4: ResNet18- Accuracy and Loss Curves

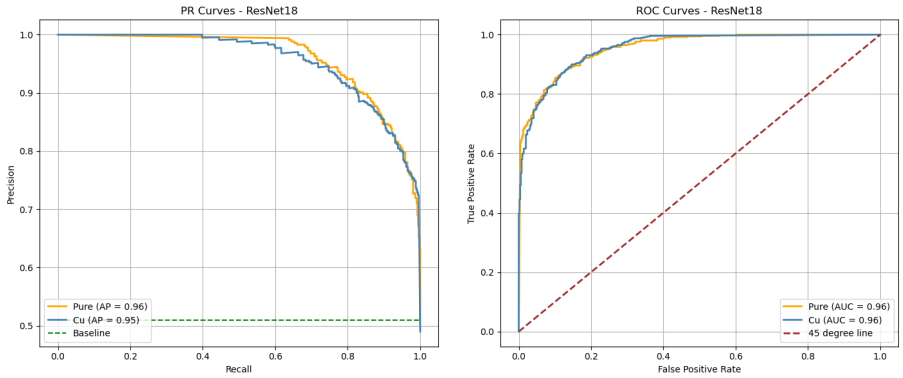


Figure 5: ResNet18- PR and ROC Curves

5.2 ResNet50 Results

The ResNet50 model demonstrated outstanding classification performance on the X-ray transmission dataset. As shown in the confusion matrix, ResNet50 correctly classified 897 out of 900 Cu samples and 898 out of 900 pure aluminum samples, with only a total of 5 misclassifications. The normalized confusion matrix further reinforces the model's robustness, showing nearly perfect accuracy—1.00 for both classes.

The training and validation accuracy curves indicate rapid and stable convergence, with accuracy exceeding 95% within the first few epochs and maintaining consistent performance throughout training. The loss curves for both training and validation sets exhibit a smooth and sharp decline, with no significant divergence between them, suggesting minimal overfitting and strong generalization capabilities[9].

Precision-Recall (PR) curves for both classes show exceptionally high average precision scores, with 0.97 for pure aluminum and 0.96 for Cu. The ROC curves also reflect excellent classification ability, achieving an AUC of 0.96 for both classes, indicating a high true positive rate and minimal false positives.

Overall, ResNet50 slightly outperformed ResNet18 in terms of raw classification accuracy and generalization, making it the top-performing architecture in this study[10].

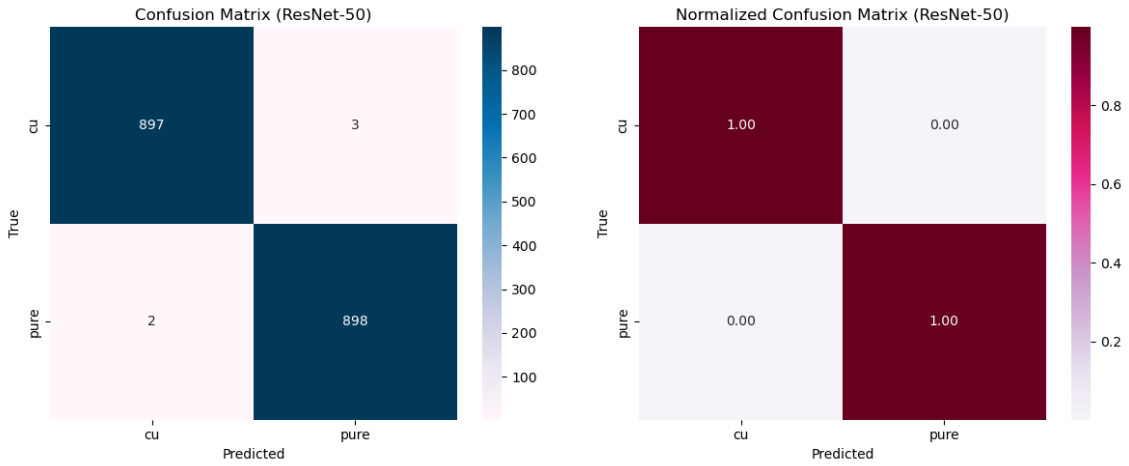


Figure 6: ResNet50- Confusion Matrix and Normalized Confusion Matrix

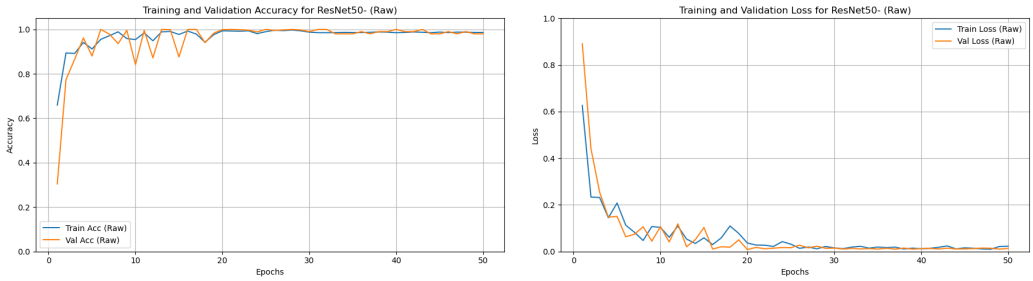


Figure 7: ResNet50- Accuracy and Loss Curves

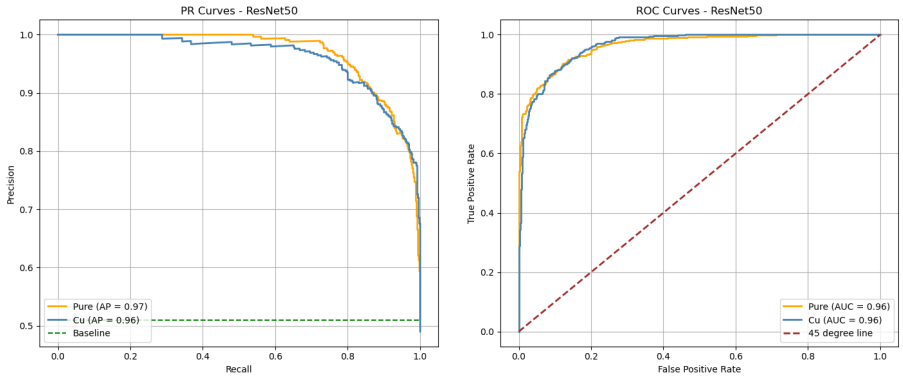


Figure 8: ResNet50- PR and ROC Curves

5.3 VGG16 Results

The performance of the VGG16 model on the combined X-ray transmission dataset demonstrated moderate classification effectiveness but evident signs of overfitting. As shown in Figure 9, the confusion matrix illustrates that the model correctly identified the majority of Cu and pure aluminum samples, with high overall accuracy. However, the normalized confusion matrix indicates slight misclassifications, particularly for Cu, with a 2% error rate.

Figure 10 presents the training and validation accuracy and loss curves, which reveal a noticeable divergence between training and validation trends as training progresses. While training accuracy continues to increase and loss decreases, the validation performance plateaus early, suggesting overfitting due to the model learning dataset-specific features rather than generalizable patterns [9].

Precision-recall (PR) and receiver operating characteristic (ROC) curves are shown in Figure 11. The area under the PR curve (AP) reached 0.94 for pure aluminum and 0.93 for Cu. The ROC curves achieved AUC values of 0.94 and 0.90 respectively, indicating moderate but inferior classification confidence compared to ResNet-based models. Overall, while VGG16 performed reasonably well, it was outperformed by ResNet18 and ResNet50 in both generalization and discriminative capacity [29].

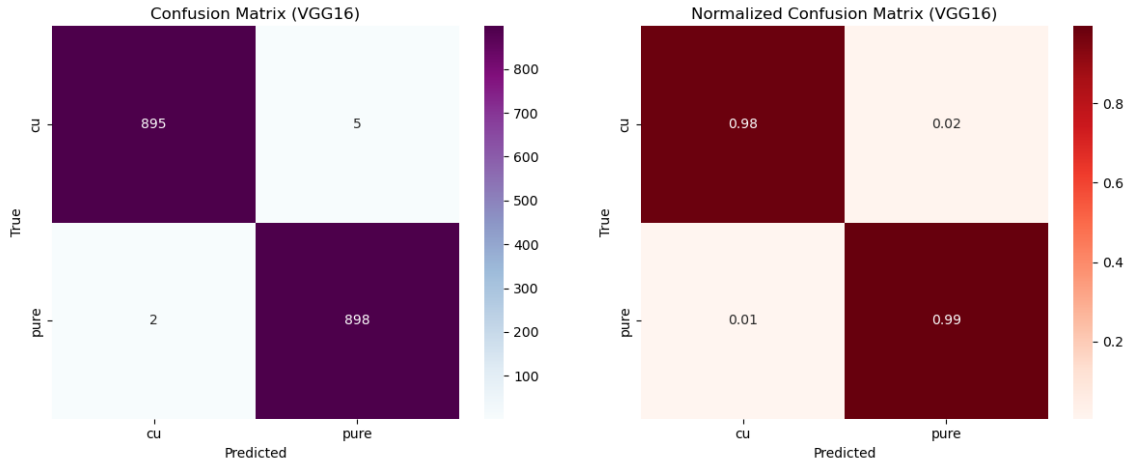


Figure 9: VGG16- Confusion Matrix and Normalized Confusion Matrix

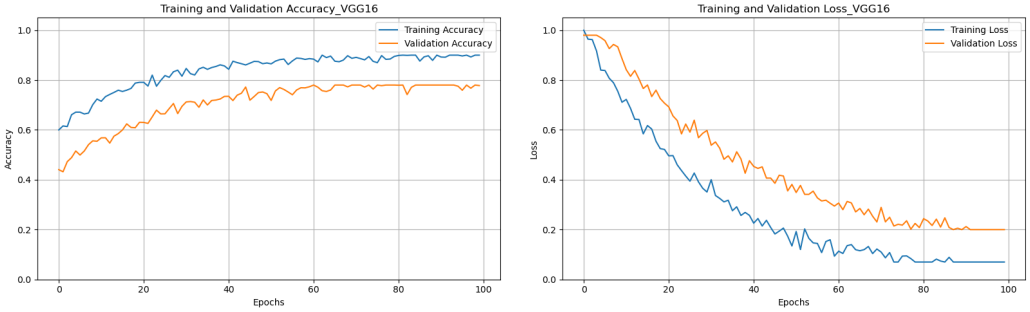


Figure 10: VGG16- Accuracy and Loss Curves

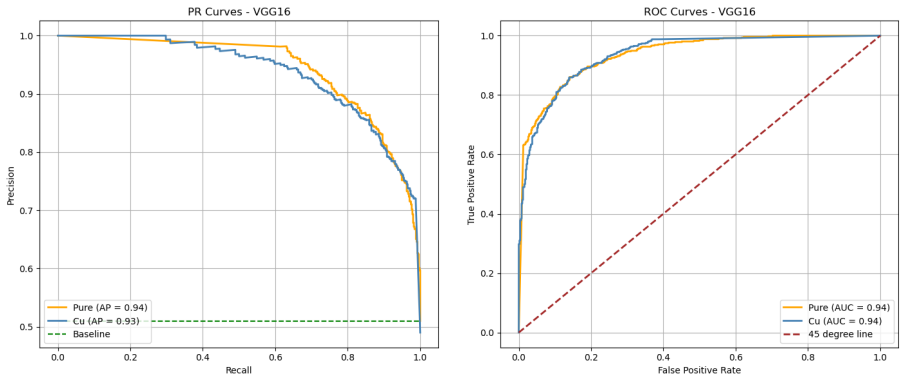


Figure 11: VGG16- PR and ROC Curves

5.4 EfficientNetB0 Results

The EfficientNetB0 model exhibited strong classification performance on the X-ray transmission dataset. As depicted in the confusion matrix, the model correctly identified 886 out of 900 Cu samples and 895 out of 900 pure aluminum samples, resulting in only 19 misclassifications overall. The normalized confusion matrix further supports the model's reliability, with class-wise accuracies of 0.98 for Cu and 0.99 for pure, indicating high precision and minimal confusion between classes.

The training and validation accuracy curves demonstrate fast and stable convergence, achieving over 95% accuracy early in training and maintaining consistent performance across epochs. Similarly, the loss curves for both training and validation sets show a rapid decline with minimal divergence, suggesting effective learning and minimal overfitting.

Precision-Recall (PR) curves reveal average precision (AP) scores of 0.93 for the pure class and 0.96 for Cu, indicating strong precision across recall levels. The ROC curves show excellent classification ability, with Area Under the Curve (AUC) values of 0.96 for Cu and pure, confirming a high true positive rate and low false positive rate across thresholds.

Overall, EfficientNetB0 demonstrated robust performance, balancing classification accuracy and model generalization, and emerges as a competitive architecture for X-ray image analysis tasks in material classification settings [34].

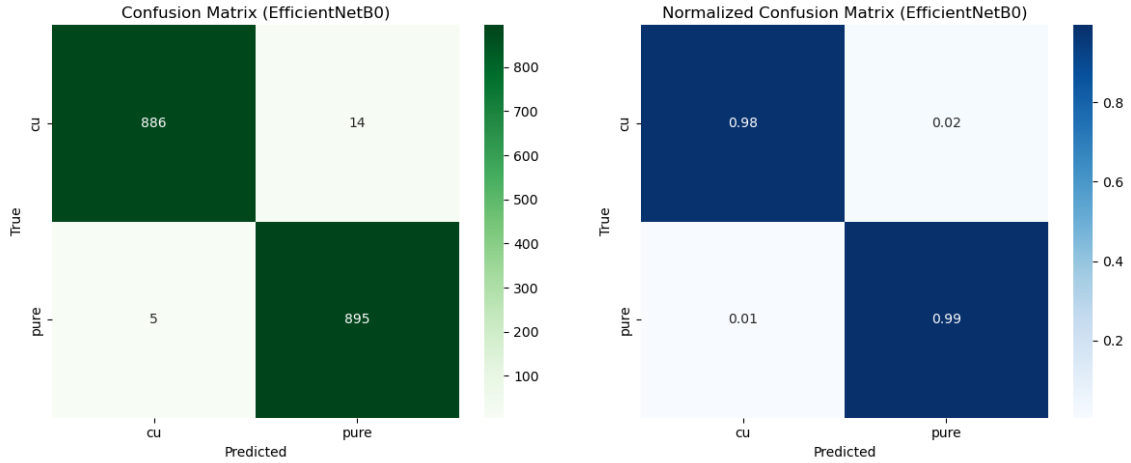


Figure 12: EfficientNetB0 - Confusion Matrix and Normalized Confusion Matrix

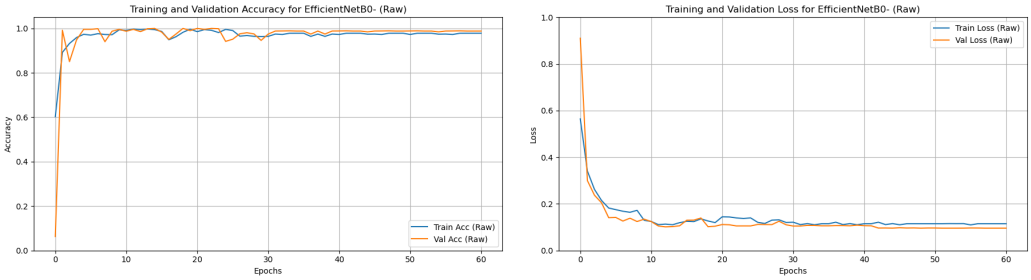


Figure 13: EfficientNetB0 - Accuracy and Loss Curves

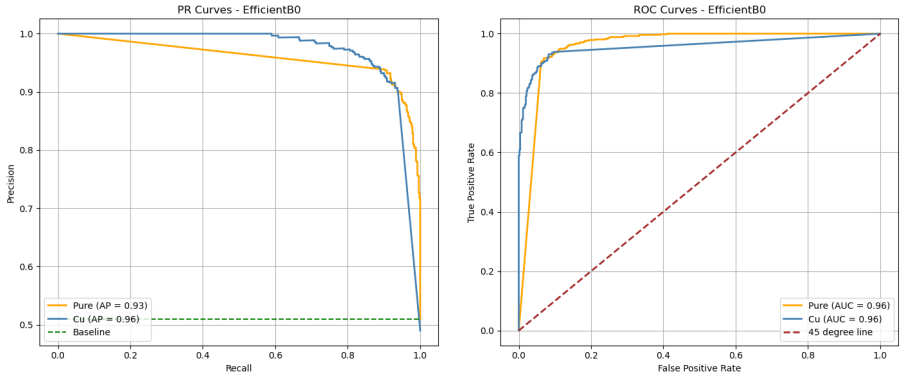


Figure 14: EfficientNetB0 - PR and ROC Curves

5.5 FNN Results

The Feedforward Neural Network (FNN) model demonstrated solid classification performance on the X-ray transmission dataset. As shown in the confusion matrix, the FNN correctly classified 857 Cu and 868 pure aluminum samples out of 900 each, resulting in a total of 75 misclassifications. The normalized confusion matrix reflects reasonable class separation, with accuracies of 0.95 for Cu and 0.96 for the pure class.

The training and validation accuracy curves show a gradual yet stable convergence, with validation accuracy approaching over 90% after approximately 80 epochs. The loss curves similarly exhibit consistent downward trends without significant divergence, indicating well-controlled overfitting and reliable generalization.

The PR curves for FNN show respectable performance, with an average precision (AP) of 0.93 for the Cu class and 0.88 for the pure class. The ROC curves provide further confirmation of strong performance, achieving AUC scores of 0.93 for both classes, indicative of a high true positive rate with moderate false positive tolerance.

Although FNN did not outperform deeper convolutional models like EfficientNetB0 or ResNet50, it remains a computationally efficient option with relatively strong generalization capabilities for this classification task [1].

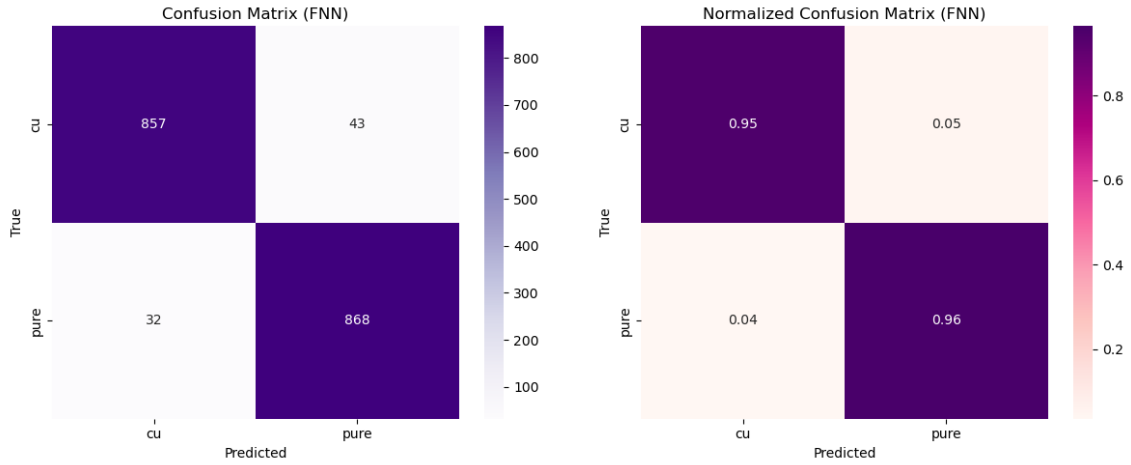


Figure 15: FNN - Confusion Matrix and Normalized Confusion Matrix

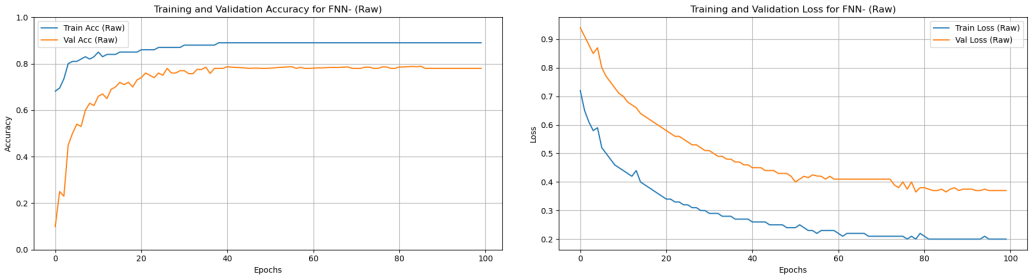


Figure 16: FNN - Accuracy and Loss Curves

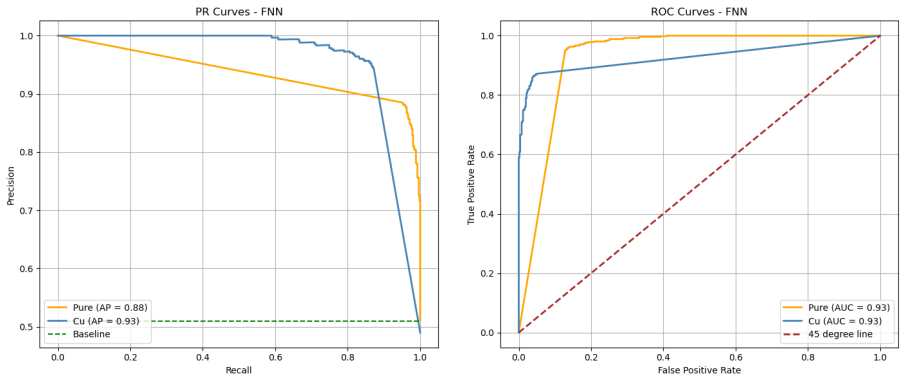


Figure 17: FNN - PR and ROC Curves

5.6 Summary of "CNN model performance"

Table 2: CNN Model - Average Accuracy and Loss Summary

Model	**Average Accuracy**	**Train Loss**	**Validation Loss**
ResNet18	99.4%	∼0.10	∼0.15
ResNet50	99.7%	∼0.05	∼0.06
VGG16	84.0%	∼0.10	∼0.20
EfficientNetB0	98.9%	∼0.10	∼0.15
FNN	84.0%	∼0.20	∼0.38

Table 3: CNN model metrics performance: Average Precision (AP) for each class and overall AUC

Model	**AP (Pure)**	**AP (Cu)**	**AUC (Avg)**
ResNet18	0.96	0.95	0.96
ResNet50	0.97	0.96	0.96
VGG16	0.94	0.93	0.94
EfficientNetB0	0.93	0.93	0.94
FNN	0.88	0.91	0.90

The performance of various convolutional neural network (CNN) architectures was evaluated on the X-ray transmission dataset to classify aluminum-based materials (Pure -1050 and CU-2017). Among all models, ResNet50 [10] demonstrated the best overall performance, achieving the highest average precision and the lowest loss of training and validation, with excellent AUC and AP scores. ResNet18 also showed strong classification results with minimal overfitting and robust generalization.

The VGG16 network [29], while achieving reasonable accuracy, exhibited signs of overfitting, particularly in the Cu class. EfficientNetB0 [34] offered a balanced trade-off between performance and computational efficiency, maintaining high AP and AUC values. The feedforward neural network (FNN) provided decent results but underperformed compared to deeper CNNs, especially in generalization to the validation set. In contrast, VGG16 and FNN showed signs of overfitting, with larger gaps between training and validation losses and lower average accuracies of 84.0% and 84.0%, respectively. Evaluation metrics such as Average Precision (AP) and Area Under the Curve (AUC) [23] were used alongside accuracy and loss curves to assess model robustness and predictive quality. In general, deeper and well-regularized architectures, such as ResNet, proved to be more suitable for this image classification task [9].

6 Future Work and Improvements

Building on the findings of Project I and II, which achieved strong results using YOLO and CNN models for binary classification of pure aluminum alloys (Al 1050) and aluminum-copper alloys (Al 2017), the next step of the project aims to expand both the classification complexity and the diversity of the task.

A key direction is to extend the model to handle **multi-class classification** across five aluminum-based alloys: pure aluminum (Al 1050), aluminum-copper (Al 2017), aluminum-magnesium (Al 5083), aluminum-silicon-magnesium (Al 6082), and aluminum-zinc-magnesium (Al 7075). Preliminary testing indicates that **ResNet50** performs particularly well as a backbone architecture for feature extraction in this domain [11].

To enhance learning capacity, **multitask learning** will be introduced, where the model jointly performs alloy classification and regression tasks (e.g., estimating related physical properties or imaging parameters). This joint learning can improve generalization and robustness by leveraging shared representations [24].

Given the **class imbalance** typical of alloy datasets, strategies such as class rebalancing, data enhancement, and weighted loss functions will be implemented to prevent performance bias and improve precision between minority classes [2].

To ensure **model interpretability**, tools such as Grad-CAM [26] and SHAP [18] will be applied to provide visual and statistical explanations of predictions. These methods enhance transparency and increase trust in the model's decisions, which is crucial in scientific and industrial contexts.

Model tuning through hyperparameter optimization, cross-validation, and architecture search will also play a key role in maximizing performance. Furthermore, refining the imaging protocol, such as increasing the wedge size and adjusting energy parameters, will help improve the quality and discriminative power of the dataset.

Ultimately, these improvements aim to deliver a **scalable**, **interpretable**, and **robust deep learning system** capable of accurately classifying and characterizing a diverse range of aluminum alloys in real-world applications.

References

[1] Christopher M Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[2] Mateusz Buda, Atsuto Maki, and Maciej A. Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, 2018.

[3] Grant Bunker. *Introduction to XAFS: A Practical Guide to X-ray Absorption Fine Structure Spectroscopy*. Cambridge University Press, 2010.

[4] Keith T Butler, David W Davies, Hugh Cartwright, Olexandr Isayev, and Aron Walsh. Machine learning for molecular and materials science. *Nature*, 559(7715):547–555, 2018.

[5] Christopher T. Chantler. Detailed tabulation of atomic form factors, photoelectric absorption and scattering cross sections, and mass attenuation coefficients in the photon energy range 1–10,000 kev for z = 1 to z = 92. *Journal of Physical and Chemical Reference Data*, 29(4):597–1048, 2000.

[6] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yinan Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9268–9277, 2019.

[7] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. In *arXiv preprint arXiv:1708.04552*, 2017.

[8] Y. Du, R. Xu, Y. He, and Z. Li. X-ray-based imaging techniques for materials characterization: A review. *Materials Characterization*, 176:111084, 2021.

[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[12] Max Jaderberg, Victor Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. Population based training of neural networks. In *arXiv preprint arXiv:1711.09846*, 2017.

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] D. C. Koningsberger and R. Prins. *X-ray Absorption: Principles, Applications, Techniques of EXAFS, SEXAFS and XANES*. Wiley-Interscience, 1988.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, volume 25, pages 1097–1105, 2012.

[16] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[17] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[18] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

[19] Martin Mayo and John Carter. X-ray techniques in material characterization: Principles, applications, and advances. *Journal of Materials Science and Engineering*, 10(2):58–72, 2021.

[20] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[21] Rafael Müller, Simon Kornblith, and Geoffrey Hinton. When does label smoothing help? In *Advances in Neural Information Processing Systems*, pages 4694–4703, 2019.

[22] Matthew Newville. Fundamentals of xafs. *Reviews in Mineralogy and Geochemistry*, 78(1):33–74, 2014.

[23] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python, 2011.

[24] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[25] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[26] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 618–626, 2017.

[27] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019.

[28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR)*, 2015.

[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

[30] Leslie N Smith. Cyclical learning rates for training neural networks. *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472, 2017.

[31] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.

[32] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[33] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013.

[34] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 97:6105–6114, 2019.

[35] Fan Zhang, Fangtao Ren, Jieping Li, and Xinhong Zhang. Automatic stomata recognition and measurement based on improved yolo deep learning model and entropy rate superpixel algorithm. *Ecological Informatics*, 68:101521, 2022.

Appendices

A Scripts made for this project

A.1 Python code for Organized data to train, validation and test files:

```
import os
import shutil
import random

# Define paths
dataset_path = '/Users/chootydoony/Documents/Miun/EL035A_Project/src/alloy_5mm/
Dataset_1050_pure/dataset_20p'
train_folder = os.path.join(dataset_path, 'train')
val_folder = os.path.join(dataset_path, 'val')
test_folder = os.path.join(dataset_path, 'test')

# Create subfolders if they don't exist
os.makedirs(train_folder, exist_ok=True)
os.makedirs(val_folder, exist_ok=True)
os.makedirs(test_folder, exist_ok=True)

# List all files in dataset_1 folder (ignoring subfolders)
all_files = [f for f in os.listdir(dataset_path) if os.path.isfile(os.path.join(dataset_path

# Shuffle the files for random splitting
random.shuffle(all_files)

# Calculate split sizes
total_files = len(all_files)
train_size = int(0.75 * total_files)
val_size = int(0.15 * total_files)
test_size = total_files - train_size - val_size

# Split the files
train_files = all_files[:train_size]
val_files = all_files[train_size:train_size + val_size]
test_files = all_files[train_size + val_size:]

# Function to move files
def move_files(file_list, destination_folder):
    for file_name in file_list:
        src_file = os.path.join(dataset_path, file_name)
```

```
        dst_file = os.path.join(destination_folder, file_name)
        shutil.move(src_file, dst_file)

# Move files to respective folders
move_files(train_files, train_folder)
move_files(val_files, val_folder)
move_files(test_files, test_folder)

print(f"Files have been split into train ({train_size}), val ({val_size}),
and test ({test_size}) folders.")
```

A.2 Python code for ResNet18-model

```
# Import libraries
import torch
import torch.nn as nn
import torch.optim as optim
from torch.optim import lr_scheduler
import torch.backends.cudnn as cudnn
import numpy as np
import torchvision
from torchvision import datasets, models, transforms
import matplotlib.pyplot as plt
import seaborn as sns
import time
import os
from PIL import Image
from tempfile import TemporaryDirectory
from sklearn.metrics import (
    classification_report,
    accuracy_score,
    confusion_matrix,
    precision_recall_curve,
    average_precision_score,
    roc_curve,
    auc,
)

cudnn.benchmark = True
plt.ion()

# Set device
data_dir = '/Users/chootydoony/Documents/Miun/EL035A_Project/Project-II/Thin_wedge/dataset_2
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
```

```
# Helper to show image (optional)
def imshow(inp, title=None):
    inp = inp.numpy().transpose((1, 2, 0))
    plt.imshow(np.clip(inp, 0, 1))
    if title:
        plt.title(title)
    plt.pause(0.001)

# Helper to smooth graphs
def moving_average(data, window_size=5):
    return np.convolve(data, np.ones(window_size)/window_size, mode='valid')

# Train model with early stopping and LR scheduler
def train_model(model, criterion, optimizer, scheduler, num_epochs=100, patience=10):
    since = time.time()

    train_acc_history = []
    val_acc_history = []
    train_loss_history = []
    val_loss_history = []

    output_dir = data_dir
    best_acc = 0.0
    epochs_no_improve = 0

    with TemporaryDirectory() as tempdir:
        best_model_params_path = os.path.join(tempdir, 'best_model_params.pt')
        torch.save(model.state_dict(), best_model_params_path)

        for epoch in range(num_epochs):
            print(f'Epoch {epoch}/{num_epochs - 1}')
            print('-' * 10)

            for phase in ['train', 'val']:
                model.train() if phase == 'train' else model.eval()
                running_loss = 0.0
                running_corrects = 0

                for inputs, labels in dataloaders[phase]:
                    inputs = inputs.to(device)
                    labels = labels.to(device)
                    optimizer.zero_grad()

                    with torch.set_grad_enabled(phase == 'train'):
                        outputs = model(inputs)
```

```
                    _, preds = torch.max(outputs, 1)
                    loss = criterion(outputs, labels)

                    if phase == 'train':
                        loss.backward()
                        optimizer.step()

                running_loss += loss.item() * inputs.size(0)
                running_corrects += torch.sum(preds == labels.data)

            epoch_loss = running_loss / dataset_sizes[phase]
            epoch_acc = running_corrects.double() / dataset_sizes[phase]

            print(f'{phase} Loss: {epoch_loss:.4f} Acc: {epoch_acc:.4f}')

            if phase == 'train':
                train_loss_history.append(epoch_loss)
                train_acc_history.append(epoch_acc.item())
            else:
                val_loss_history.append(epoch_loss)
                val_acc_history.append(epoch_acc.item())

                if scheduler:
                    scheduler.step(epoch_loss)

                if epoch_acc > best_acc:
                    best_acc = epoch_acc
                    torch.save(model.state_dict(), best_model_params_path)
                    epochs_no_improve = 0
                else:
                    epochs_no_improve += 1

        if epoch >= 30 and epochs_no_improve >= patience:
            print(f"Early stopping triggered after {epoch+1} epochs with no improvement
            break
        print()

    time_elapsed = time.time() - since
    print(f'Training complete in {time_elapsed // 60:.0f}m {time_elapsed % 60:.0f}s')
    print(f'Best val Acc: {best_acc:.4f}')
    model.load_state_dict(torch.load(best_model_params_path))



# Plot loss and accuracy curves
smoothed_train_acc = moving_average(train_acc_history)
```

```
    smoothed_val_acc = moving_average(val_acc_history)
    smoothed_train_loss = moving_average(train_loss_history)
    smoothed_val_loss = moving_average(val_loss_history)

    plt.figure()
    plt.plot(smoothed_train_loss, label='Train Loss (smoothed)')
    plt.plot(smoothed_val_loss, label='Val Loss (smoothed)')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.title('Training and Validation Loss (Smoothed)')
    plt.legend()
    plt.savefig(os.path.join(output_dir, f"loss_plot_{norm_type}.png"))

    plt.figure()
    plt.plot(smoothed_train_acc, label='Train Acc (smoothed)')
    plt.plot(smoothed_val_acc, label='Val Acc (smoothed)')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.title('Training and Validation Accuracy (Smoothed)')
    plt.legend()
    plt.savefig(os.path.join(output_dir, f"accuracy_plot_{norm_type}.png"))

    return model

# Evaluate test data and generate plots
def evaluate_on_test(model, norm_type):
    test_dataset = datasets.ImageFolder(os.path.join(data_dir, 'test'), current_transforms[
    test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=64, shuffle=False, nu

    model.eval()
    y_true = []
    y_pred = []
    y_scores = []

    with torch.no_grad():
        for inputs, labels in test_loader:
            inputs = inputs.to(device)
            labels = labels.to(device)
            outputs = model(inputs)
            _, preds = torch.max(outputs, 1)

            y_true.extend(labels.cpu().numpy())
            y_pred.extend(preds.cpu().numpy())
            y_scores.extend(torch.softmax(outputs, dim=1).cpu().numpy())

    # Save classification report
```

```
report = classification_report(y_true, y_pred, target_names=class_names)
acc = accuracy_score(y_true, y_pred)
report_file = os.path.join(data_dir, f"test_report_{norm_type}.txt")
with open(report_file, "w") as f:
    f.write("Classification Report:\n")
    f.write(report + "\n")
    f.write(f"Test Accuracy: {acc:.4f}\n")
print(f"Report saved to {report_file}")

# Confusion matrix (absolute)
cm = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Purples', xticklabels=class_names, yticklabel
plt.title(f'Confusion Matrix ({norm_type})')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.tight_layout()
plt.savefig(os.path.join(data_dir, f"confusion_matrix_{norm_type}.png"))

# Normalized confusion matrix
cm_normalized = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
plt.figure(figsize=(8, 6))
sns.heatmap(cm_normalized, annot=True, fmt='.2f', cmap='RdPu', xticklabels=class_names,
plt.title(f'Normalized Confusion Matrix ({norm_type})')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.tight_layout()
plt.savefig(os.path.join(data_dir, f"normalized_confusion_matrix_{norm_type}.png"))

# PR Curves
y_true_onehot = np.eye(len(class_names))[y_true]
y_scores = np.array(y_scores)
plt.figure(figsize=(10, 8))
for i in range(len(class_names)):
    precision, recall, _ = precision_recall_curve(y_true_onehot[:, i], y_scores[:, i])
    ap = average_precision_score(y_true_onehot[:, i], y_scores[:, i])
    #smooth_prec = smooth_data(precision)
    #smooth_recall = smooth_data(recall)
    plt.plot(recall, precision, label=f'{class_names[i]} (AP={ap:.2f})')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title(f'Precision-Recall Curve ({norm_type})')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.savefig(os.path.join(data_dir, f"pr_curve_{norm_type}.png"))
```

```
    # ROC Curves
    plt.figure(figsize=(10, 8))
    for i in range(len(class_names)):
        fpr, tpr, _ = roc_curve(y_true_onehot[:, i], y_scores[:, i])
        roc_auc = auc(fpr, tpr)
       # smooth_fpr = smooth_data(fpr)
      # smooth_tpr = smooth_data(tpr)
        plt.plot(fpr, tpr, label=f'{class_names[i]} (AUC={roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title(f'ROC Curve ({norm_type})')
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.savefig(os.path.join(data_dir, f"roc_curve_{norm_type}.png"))

# Main experiment function
def run_experiment(use_imagenet_stats):
    global dataloaders, dataset_sizes, class_names, current_transforms, norm_type

    print(f"\n=== Running with {'ImageNet' if use_imagenet_stats else 'Custom'} Normalizatio
    norm_type = 'imagenet' if use_imagenet_stats else 'custom'

    mean = [0.485, 0.456, 0.406] if use_imagenet_stats else [0.8816, 0.9289, 0.8946]
    std = [0.229, 0.224, 0.225] if use_imagenet_stats else [0.2705, 0.1777, 0.2310]

    current_transforms = {
        'train': transforms.Compose([
            transforms.Resize((800, 800)),
            transforms.RandomRotation(20),
            transforms.RandomHorizontalFlip(p=0.5),
            transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.2),
            transforms.ToTensor(),
            transforms.Normalize(mean, std)
        ]),
        'val': transforms.Compose([
            transforms.Resize((800, 800)),
            transforms.ToTensor(),
            transforms.Normalize(mean, std)
        ])
    }

    image_datasets = {x: datasets.ImageFolder(os.path.join(data_dir, x), current_transforms
    #dataloaders = {x: torch.utils.data.DataLoader(image_datasets[x], batch_size=64, shuffle
```

```
    dataloaders = {
    'train': torch.utils.data.DataLoader(image_datasets['train'], batch_size=64, shuffle=Tru
    'val': torch.utils.data.DataLoader(image_datasets['val'], batch_size=64, shuffle=False,
}

    dataset_sizes = {x: len(image_datasets[x]) for x in ['train', 'val']}
    class_names = image_datasets['train'].classes

#Model_train_ResNet18
    model = models.resnet18(weights='IMAGENET1K_V1')
    num_ftrs = model.fc.in_features
    model.fc = nn.Sequential(
        nn.Linear(num_ftrs, 512),
        nn.ReLU(),
        nn.Dropout(0.3),
        nn.Linear(512, len(class_names))
    )
    model = model.to(device)

    criterion = nn.CrossEntropyLoss()
    optimizer = optim.Adam(model.parameters(), lr=0.0002, weight_decay=0.0001)
    scheduler = lr_scheduler.CyclicLR(optimizer, base_lr=0.0001, max_lr=0.001, step_size_up=

    model = train_model(model, criterion, optimizer, scheduler)
    evaluate_on_test(model, norm_type)

# Run both experiments
run_experiment(use_imagenet_stats=True)
run_experiment(use_imagenet_stats=False)

# Print report summaries
for norm_type in ['imagenet', 'custom']:
    print(f"\n>>> {norm_type.upper()} Normalization Report:")
    with open(os.path.join(data_dir, f"test_report_{norm_type}.txt")) as f:
        for line in f:
            if "Accuracy" in line or "precision" in line or "f1-score" in line:
                print(line.strip())
```