JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 36 (2025) 303–311
Received: 8 April 2025
Accepted: 28 May 2025

Image classification using PyTorch and Core ML

Klasyfikacja obrazów z zastosowaniem bibliotek PyTorch oraz Core ML

Jakub Ślusarski*, Arkadiusz Szumny, Maria Skublewska-Paszkowska

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

The aim of the study was to compare different machine learning models trained using the PyTorch library in Python and the Core ML library in the Create ML tool. In the case of PyTorch, using transfer learning on a pre-trained ResNet50 model, data augmentation and normalization, four models were trained on two various data sets, achieving accuracy, precision, recall and F1 score above 80%. Four identical models were trained on the same data sets in the Create ML tool, and the conversion of the PyTorch models to the Core ML format allowed for a reliable comparison. This also emphasizes the effectiveness of conversion using the coremltools library, while maintaining model performance. The study emphasizes the key role of dataset quality and techniques for improving dataset quality.

Keywords: image classification; Core ML; PyTorch; image recognition

Streszczenie

Celem badania było porównanie różnych modeli uczenia maszynowego wytrenowanych przy użyciu biblioteki PyTorch w Python i biblioteki Core ML w Create ML. W przypadku PyTorch wykorzystując uczenie transferowe na wstępnie wytrenowanym modelu ResNet50, rozszerzenie i normalizację danych, cztery modele zostały wytrenowane na dwóch różnych zestawach danych, osiągając dokładność, precyzję, czułość i wynik F1 na poziomie przekraczającym 80%. Na tych samych zbiorach danych wytrenowano cztery tożsame modele w narzędziu Create ML, a konwersja modeli PyTorch do formatu Core ML pozwoliła na ich wiarygodne porównanie. Podkreśla to również skuteczność konwersji przy użyciu biblioteki coremltools, zachowując wydajność modelu. Badanie podkreśla kluczową rolę jakości zbioru danych i technik poprawy jakości zbioru danych.

Słowa kluczowe: klasyfikacja obrazów; Core ML; PyTorch; rozpoznawanie obrazów

*Corresponding author

Email address: jakub.slusarski@pollub.edu.pl (J. Slusarski)

1. Introduction

Machine learning, especially Convolutional Neural Networks (CNNs), has become an indispensable tool in image analysis tasks [1-3]. These models are widely used in fields such as medicine [1, 4-7], agriculture [8], and industry [9], where they support automation, enhance decision-making, and reduce the risk of human error. For example, in medicine, CNN-based systems enable early detection of pathological changes in radiological or dermatological images, often identifying conditions that are difficult for the human eye to detect. In agriculture, machine learning facilitates plant disease recognition and crop monitoring, while in industry, CNNs improve quality control by identifying production defects with high accuracy and speed.

One of the key challenges in developing image classification models is achieving high performance while minimizing training time and computational complexity. Transfer learning has emerged as an effective solution, allowing developers to reuse pre-trained models – such as ResNet50 trained on ImageNet adapting them to specific tasks with significantly reduced resource requirements. Another critical factor influencing model performance is the quality of the training dataset. Techniques such as data augmentation and normalization enhance dataset diversity, improve generalization, and reduce overfitting [9, 10].

To evaluate model effectiveness, standard metrics including precision, recall, F1 score, and accuracy are used. These metrics allow for comprehensive assessment and ensure comparability between different approaches. With the growing power of mobile devices and dedicated ML hardware, deploying advanced image classification models on consumer devices is increasingly feasible [11, 12].

This paper investigates the practical application of modern machine learning frameworks – PyTorch and Core ML – for image classification. The study focuses on two datasets: Weather Image Recognition and Rice Image Dataset. Models were trained using transfer learning with a pre-trained ResNet50 in PyTorch and via Apple's Create ML tool. PyTorch models were also converted to Core ML format using the *coremltools* library, allowing a reliable comparison of performance and the impact of model conversion.

The research hypotheses of this study are as follows:

H1: *Machine learning models trained in the recognition of weather images achieve high accuracy of at least 85% in classifying weather conditions in images using PyTorch and Core ML libraries*.

H2: *Machine learning models trained on a rice image dataset achieve high accuracy of at least 85% in classifying different types of rice grains using PyTorch and Core ML libraries*.

H3: *Converting the model from PyTorch to Core ML does not negatively affect the model's performance and accuracy*.

H4: *Models created with PyTorch framework and converted to Core ML format achieve better results than models created with Create ML*.

2. Related work

Convolutional neural networks (CNNs) are widely applied in medical image analysis, offering significant improvements in diagnostic accuracy [1]. Their applications include segmentation, abnormality detection, disease classification, and computer-aided diagnosis. For example, CNNs achieved above 98% accuracy in Alzheimer's detection and up to 91% Dice score in brain tumor segmentation, surpassing traditional machine learning (ML) methods. Key challenges such as limited labeled data and computational demands are mitigated using transfer learning and data augmentation [1, 4].

As shown in [4], deep learning (DL) – particularly CNNs – has become a dominant method in medical image tasks such as classification, segmentation, and object detection across domains including neurology, ophthalmology, pathology, cardiology, and many more. However, issues like data scarcity and model interpretability remain relevant.

In agriculture, DL also outperforms classical models like SVMs and random forests in accuracy and efficiency, though it depends heavily on large datasets [8]. Transfer learning and augmentation are effective remedies. Similarly, [13] highlights that ML excels in structured data tasks (e.g., finance, marketing), while DL is better suited to unstructured data (images, text, audio), albeit at a higher computational cost.

In food image analysis, CNN-based approaches achieve superior results in tasks like ingredient recognition, food type classification, and portion estimation [9]. Transfer learning enhances accuracy and reduces training time, while techniques like augmentation and semi-supervised learning help address issues like overfitting and limited data. Complementary studies [14] confirm the strengths of models like YOLOv5 in balancing accuracy and inference speed, whereas two-stage detectors (e.g., Faster R-CNN) offer higher precision. These works emphasize the importance of optimizing data and model architecture for specific applications.

Deep learning models also prove more robust than traditional ML in insect identification and environmental monitoring, offering automated feature extraction and scalability with large datasets [15]. In geological applications, CNNs trained on carbonate rock images show that dataset size significantly affects performance – larger datasets (104,306 images) yield the best accuracy, while small datasets (7,000 images) lead to overfitting despite using transfer learning [10].

The role of attention mechanisms in computer vision is explored in [16, 17], categorizing them into spatial, channel, temporal, and branch-based types. These mechanisms improve tasks such as classification, segmentation, face recognition, and 3D vision, offering a powerful alternative or enhancement to CNNs, especially for large-scale or complex datasets.

While CNNs dominate, [18] presents Extreme Learning Machines (ELMs) as a simplified and efficient alternative to methods like SVMs and FNNs. ELMs avoid issues like local minima and slow convergence, and are effective even on noisy or imbalanced datasets, especially when paired with parallel hardware implementations.

The comparison between Vision Transformers (ViTs) and CNNs in [19] shows that ViTs outperform CNNs on small, noisy datasets due to their self-attention mechanism, achieving over 96% accuracy in pneumonia detection. However, CNNs remain more efficient with larger datasets and lower hardware requirements. Hybrid ViT-CNN models show up to a 10% improvement in accuracy.

Mobile deployment of DL models is discussed in [20, 21]. In [20], a TensorFlow/Keras model (fine-tuned InceptionV3) was converted to CoreML for iOS food recognition, achieving nearly 87% top-1 and over 97% top-5 accuracy, running fully offline with fast prediction times. In [21], a GoogLeNet-based model trained with extensive augmentation achieved over 98% accuracy on chest X-rays, converting successfully for iOS use via CoreML.

Finally, [22, 23] reinforce the importance of transfer learning, especially when large datasets are unavailable. Techniques like fine-tuning, augmentation, and evolutionary algorithms enhance model accuracy and generalization. Recent advancements, including attention mechanisms, transformer-based models, and AutoML approaches, continue to push the boundaries of CNN performance, scalability, and interpretability.

3. Materials and methods

The aim of the study is to evaluate the effectiveness and efficiency of models in classifying weather and rice images trained using different techniques, as well as to evaluate the effectiveness of converting models from .pth format (PyTorch model) to .mlmodel (Core ML model). The following models were prepared for this purpose (see Table 1): 4 models created in PyTorch, 4 models converted from PyTorch to Core ML, 4 models created in Create ML. The models created in PyTorch were trained using transfer learning with the pre-trained CNN ResNet50 model. The study consisted of the following stages:

- data preparation which included splitting the datasets into two sets in the proportion of 80% (training) 20% (testing), and then subjecting the 80% set to the augmentation process,
- training four models on two datasets with and without augmentation using the transfer learning method in Python with the PyTorch library using the pre-trained ResNet50,
- model conversion of the four trained models from PyTorch to Core ML using the coremltools library,
- conversion of the four trained models from PyTorch to Core ML using the coremltools library,
- training the 4 models in Create ML,
- testing the generated 12 models,

- comparing the performance of the PyTorch models before and after conversion with models created with Create ML.

Table 1: Names of the created models

Model name	Dataset	Source	Training time
PyR	Rice	PyTorch	7h (~12m epoch)
PyRA	Rice - augmented	PyTorch	40h (~2h epoch)
PyW	Weather	PyTorch	17m (~1m epoch)
PyWA	Weather - augmented	PyTorch	5h 10m (~10m epoch)
PyRC	Rice	PyR converted to Core ML	18s (conversion)
PyRAC	Rice - augmented	PyRA converted to Core ML	20s (conversion)
PyWC	Weather	PyW converted to Core ML	10s (conversion)
PyWAC	Weather - augmented	PyWA converted to Core ML	15s (conversion)
CoreR	Rice	Create ML	23m (~48s epoch)
CoreRA	Rice - augmented	Create ML	- (out of memory)
CoreW	Weather	Create ML	1m (~3s epoch)
CoreWA	Weather - augmented	Create ML	20m (~30s epoch)

3.1. Research stand

The test bench consists of devices and tools used to evaluate the performance of the trained models. All models were trained on a 2020 MacBook Pro with an M1 processor and 8 GB RAM. Models in .mlmodel format were created in Create ML using the Core ML library and transfer learning. Python-based models were built using PyTorch with a pre-trained ResNet50 and also used transfer learning. These were converted to Core ML format (precision: FLOAT32, target: iOS 16) via the *coremltools* library, enabling a reliable comparison with models created directly in Core ML. The model input parameters were retained in accordance with the training process: scaling, bias, RGB/CHW layout. To improve accuracy and numerical stability, the following were applied: logit normalization, stable softmax. After conversion, the model was automatically tested on a test set.

Model evaluation was performed using Python scripts, which automatically process test datasets (weather and rice) to assess performance on previously unseen data. Metrics such as accuracy, precision, sensitivity, and F1 score were calculated to provide a comprehensive analysis of model effectiveness.

3.2. Datasets and data preparation

The models were trained on two datasets. The first, the Rice Image Dataset [24], contains 75,000 images of five rice varieties: Caracadag, Ipsala, Jasmine, Arborio, and Basmati—15,000 images per class. Each image (250×250 px) shows a single grain on a uniform background, which facilitates accurate visual analysis and improves model performance.

The second dataset, Weather Image Recognition [25], consists of 6,862 images across 11 weather-related classes: dew, fog/smog, frost, glaze, hail, lightning, rain, rainbow, rime, sandstorm, and snow. The dataset is imbalanced, and the images vary in resolution and aspect ratio, which may negatively impact model accuracy.
Both datasets were split into 80% training, 20% testing sets and extraction of 20% from the training data for validation purpose. To improve model performance, the images were preprocessed through normalization (resizing to 244×244 px and scaling pixel values to [0,1]) and augmentation using the Albumentations library [26]. Techniques included horizontal flipping (p=0.5), brightness, contrast, saturation and hue (ColorJitter) adjustment, rotation, cropping and scaling. Each image was augmented 10 times, increasing the Rice dataset from 60,000 to 600,000 images and the Weather dataset from 5,485 to 54,850.

3.3. PyTorch model

The model was built in PyTorch using the ResNet-50 [27] architecture pre-trained on ImageNet. To adapt it for specific tasks, the original final layer was replaced with a custom classifier matching the number of target classes and included dropout to prevent overfitting, with the following sequence: linear layer (input - 256), ReLU, Dropout (p = 0.5), linear layer (256 - number of classes). The pre-trained layers were frozen to retain learned features. Training process was performed using the CrossEntropyLoss function, batch size 32 and the Adam optimizer (learning rate: 0.001), with early stopping after five epochs without validation improvement. After training, the model was converted to Core ML format using the *coremltools* library (see Listing 1).

Listing 1: Model conversion from .pth to .mlmodel

```
coreml_model = ct.convert(
        traced_model,
        inputs=[ct.TensorType(shape=input_shape)],
        classifier_config=ct.ClassifierConfig(class_labels),
        convert_to="mlprogram" if use_mlpackage else "neuralnetwork",
    )
```

3.4. Create ML Model

The model was developed in Xcode using the Create ML tool, which leverages the Core ML framework for image classification tasks. It uses VisionFeaturePrint_Screen, a pre-trained neural network from Apple's Vision framework, optimized for efficient and high-performance image recognition on iOS and macOS. This feature extractor is designed to identify visual patterns such as textures, shapes, and object structures, providing a strong foundation for transfer learning.

In Create ML, only the final classification layers are trained, while the feature extraction layers of VisionFeaturePrint_Screen remain unchanged. This approach allows for effective training on custom datasets – such as weather condition images or rice grain varieties – while significantly reducing training time and computational effort.

After training, the model was exported to the .mlmodel format, ensuring easy deployment on Apple

platforms. Create ML automatically evaluates the model's performance, confirming its high classification accuracy and suitability for real-world applications. Due to Apple's on-device optimizations, the model offers fast inference and low resource usage, making it ideal for real-time applications on iOS and macOS devices.

3.5. Testing and evaluation

After each epoch, the following metrics were recorded: training and validation loss, accuracy for both sets. The model with the best validation result was automatically saved. Predictions and class labels were also saved for further analysis (confusion matrices). The trained models were tested, and their performance was evaluated using four standard metrics: accuracy, precision, recall, and F1 score [28].

4. Results

The evaluation of the models on the Weather Image Recognition and Rice Image Dataset demonstrates the effectiveness of using machine learning models in three forms: original PyTorch models, PyTorch models converted to Core ML, and models created directly in Create ML. The assessment metrics included accuracy, precision, recall, and F1 score, providing a comprehensive comparison between these approaches.

4.1. Model training procedure

The models demonstrated stable and consistent learning throughout training. Without augmentation, the weather classification model reached high accuracy after 39 iterations (Figure 1), while the augmented version converged faster (Figure 2). Similar accuracy and loss patterns were observed in other runs (Figures 3-6), confirming the robustness of the training process.

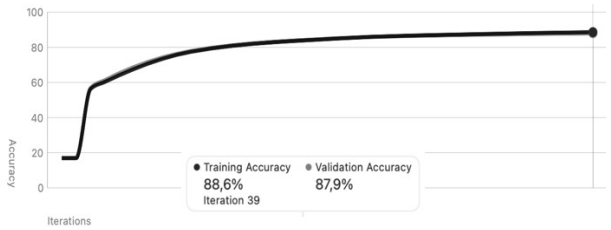

Figure 1: Training and validation accuracy on model trained in Create ML with weather dataset.

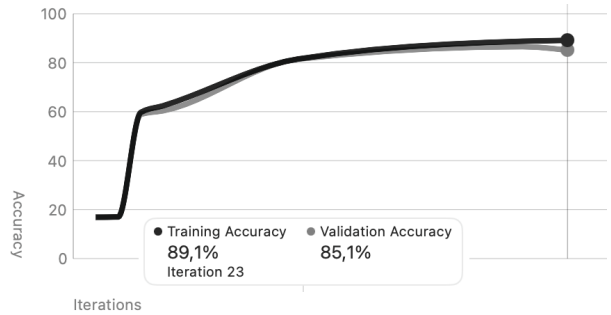

Figure 2: Training and validation accuracy on model trained in Create ML with augment weather dataset.

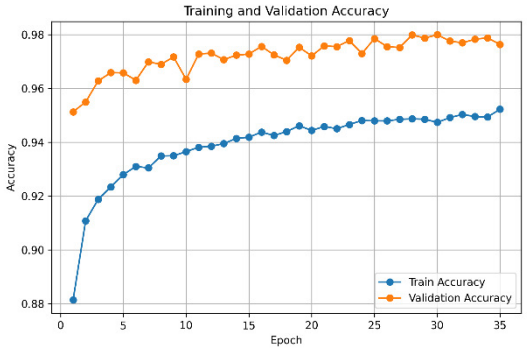

Figure 3: Training and validation accuracy on model trained with PyTorch and rice dataset.

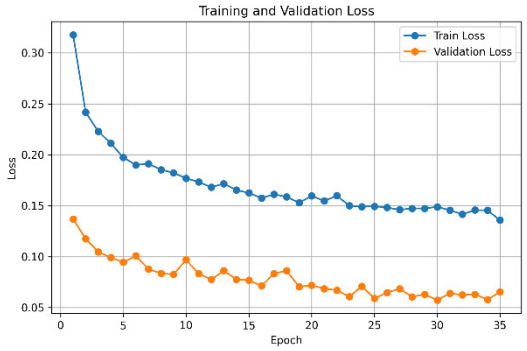

Figure 4: Training and validation loss on model trained with PyTorch and rice dataset.

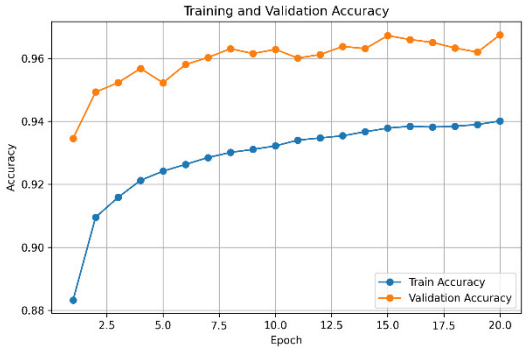

Figure 5: Training and validation accuracy on model trained with PyTorch and augmented rice dataset.

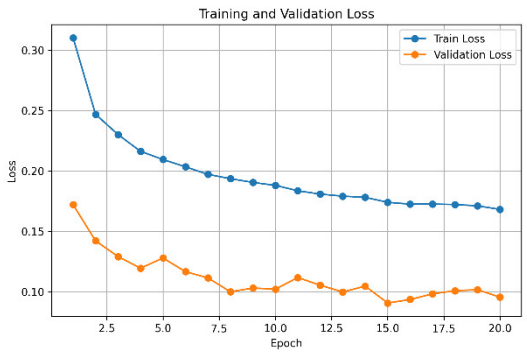

Figure 6: Training and validation loss on model trained with PyTorch and augment rice dataset.

4.2. Weather Image Model Evaluation

For the weather image classification task (see Table 2), models trained with data augmentation (PyWA, PyWAC, CoreWA) consistently outperformed those trained without it (PyW, PyWC, CoreW). The highest accuracy, 89.11%, was achieved by the PyWA model, closely followed by PyWAC with 89.04%. Both models also exceeded 90% in precision, recall, and F1 score, indicating strong generalization capabilities.

In contrast, models trained without augmentation (PyW, PyWC, CoreW) achieved slightly lower results, with accuracy values around 85%, and correspondingly consistent precision, recall, and F1 scores. These findings confirm that data augmentation has a positive impact on classification performance, supporting hypothesis H1 – that machine learning models trained for weather classification can achieve an accuracy of at least 85%.

4.3. Confusion Matrices Analysis

The confusion matrices (Figures 7-8) provide deeper insights into the performance of PyTorch models before conversion to Core ML, illustrating both classification accuracy and areas of misclassification. Figure 7 presents the results of the model trained without data augmentation (PyW). While the model performs well across most weather categories, it shows notable misclassifications between visually similar classes, such as glaze and frost, as well as sandstorm and fog/smog, suggesting difficulties in distinguishing subtle visual differences.

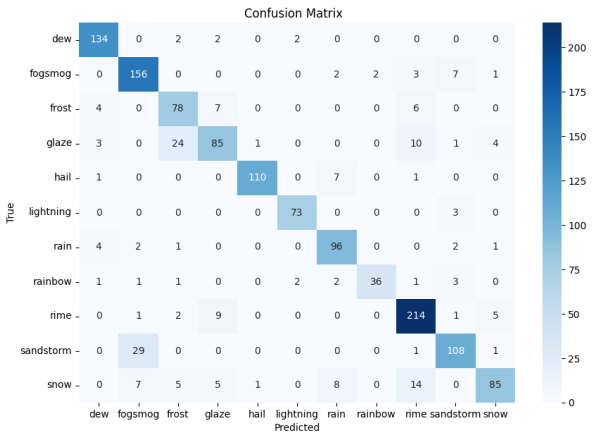

Figure 7: Confusion matrix from PyTorch model on weather dataset.

In contrast, Figure 8 shows the performance of the model trained with data augmentation (PyWA). The application of augmentation techniques leads to improved generalization and a reduction in misclassification errors, particularly in categories like glaze, frost, and hail.

These results clearly indicate that data augmentation contributes to higher classification accuracy by improving the model's ability to generalize to variations in lighting, angles, and weather conditions. The augmented model (PyWA) consistently outperforms the non-augmented one (PyW), thus supporting hypothesis H1, which states that machine learning models trained for weather classification can achieve an accuracy of at least 85%.

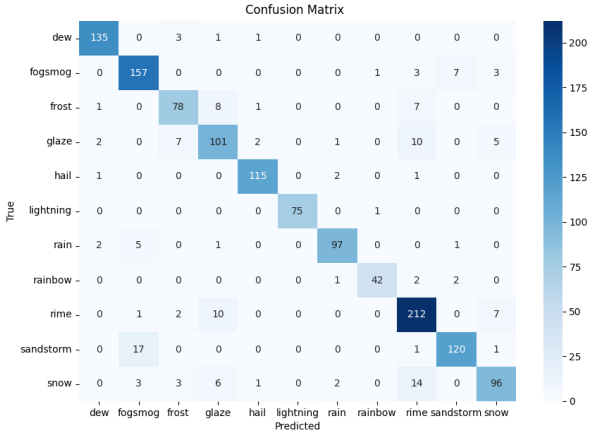

Figure 8: Confusion matrix from PyTorch model on augment weather dataset.

4.4. Rice Image Model Evaluation

For the rice classification task (see Table 2), models trained with data augmentation (PyRA, PyRAC) showed slight improvements over those without augmentation (PyR, PyRC), although overall performance was already exceptionally high. The highest accuracy, 98.56%, was achieved by PyRAC, followed closely by PyRA at 98.51%, indicating that augmentation had a limited but noticeable impact.

Models without augmentation also performed very well, with accuracy around 98%, suggesting that the high quality and uniformity of the dataset enabled strong classification results regardless of additional training techniques. These findings confirm hypothesis H2, as all models surpassed the 85% accuracy threshold in classifying different rice grain types. Unlike the weather classification task, where augmentation significantly boosted performance, here its effect was marginal – supporting the notion that augmentation is most beneficial when dealing with datasets characterized by high visual variability rather than standardized image conditions.

4.5. Confusion Matrices Analysis

The confusion matrices (Figures 9-10) provide additional insight into the performance of PyTorch models before Core ML conversion.

Figure 9 (PyR) shows high classification accuracy with minimal misclassifications across all rice varieties, reflecting the dataset's uniform structure. In Figure 10 (PyRA), data augmentation slightly improves the model's ability to distinguish between similar rice types, though the overall impact is limited due to the dataset's already high quality.

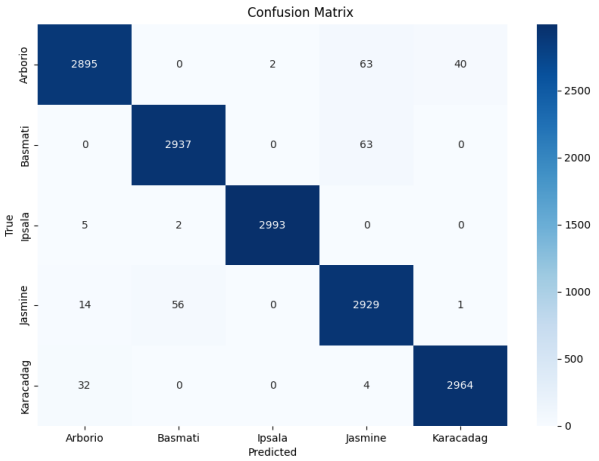

Figure 9: Confusion matrix from PyTorch model on rice dataset.

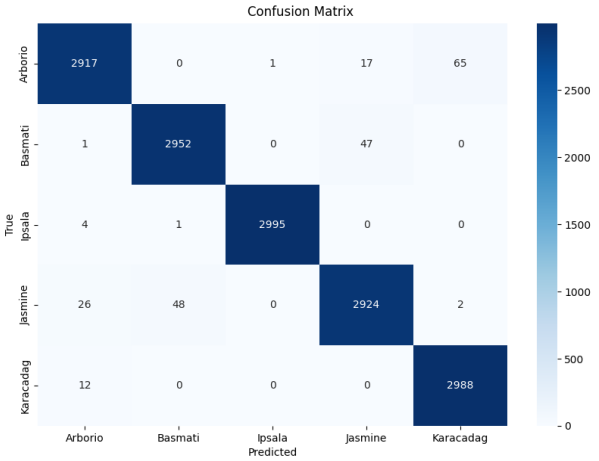

Figure 10: Confusion matrix from PyTorch model on augment rice dataset.

4.6. Impact of Conversion to Core ML

A critical part of the evaluation was assessing the effect of converting PyTorch models to Core ML and comparing them with models trained directly in Create ML. The goal was to determine whether conversion impacted performance and whether PyTorch-trained models retained their accuracy after being adapted for deployment on Apple devices. The results show no significant difference in performance between the original PyTorch models (PyW, PyWA, PyR, PyRA) and their Core ML counterparts (PyWC, PyWAC, PyRC, PyRAC). Both versions produced consistent accuracy, precision, recall, and F1 score metrics, supporting H3, which states that converting PyTorch models to Core ML does not negatively affect classification performance.

The confusion matrices (Figures 11-14) further confirm that PyTorch-trained models perform nearly identically before and after conversion, maintaining their ability to differentiate between classes effectively.

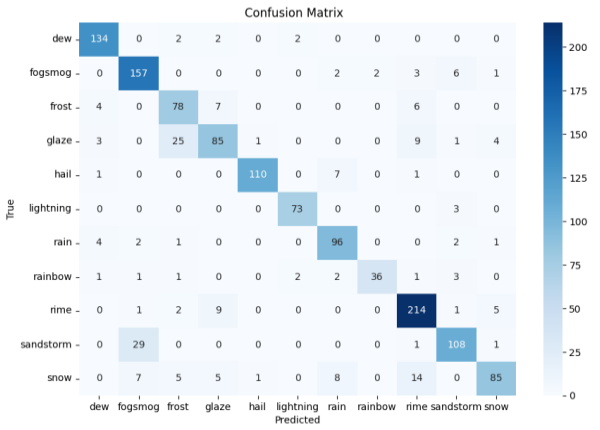

Figure 11: Confusion matrix from converted PyTorch model to Core ML on weather dataset.

Figure 11 (PyWC) maintains classification accuracy similar to PyW, reinforcing that conversion does not degrade model performance. Figure 12 (PyWAC) performs equally well as with PyWA, demonstrating that data augmentation remains beneficial even after conversion. Figure 13 (PyRC) retains the high classification accuracy of PyR, confirming that conversion does not introduce errors or degradation. Figure 14 (PyRAC) shows minor improvements in classification consistency, aligning with PyRA's results. These findings validate H3, proving that converting models from PyTorch to Core ML does not negatively impact performance and that models retain their accuracy after conversion.

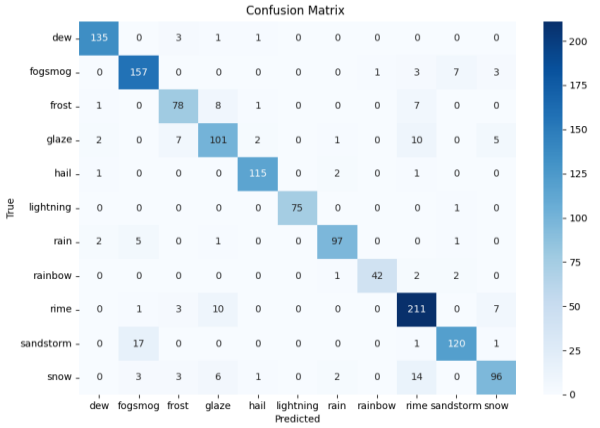

Figure 12: Confusion matrix from converted PyTorch model to Core ML on augment weather dataset.

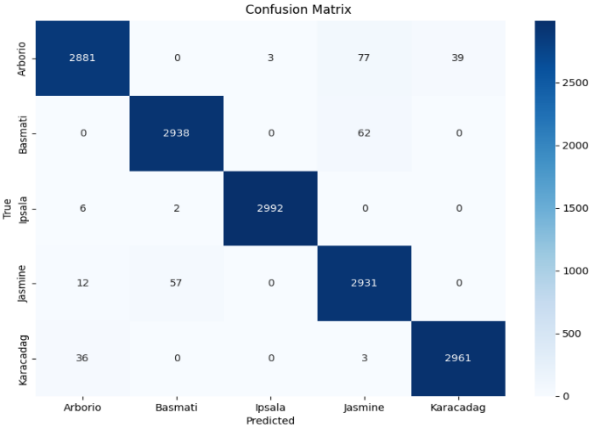

Figure 13: Confusion matrix from converted PyTorch model to Core ML on rice dataset.

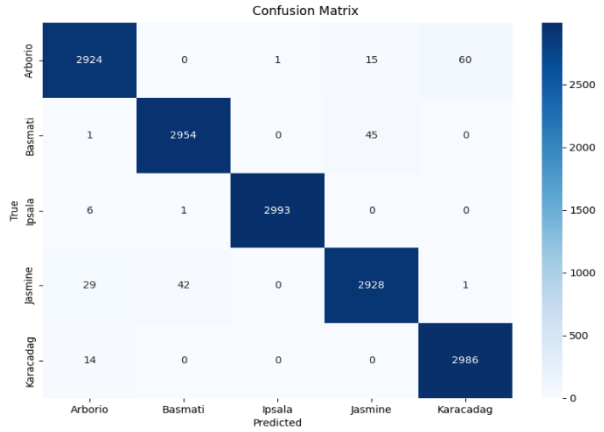

Figure 14: Confusion matrix from converted PyTorch model to Core ML on augment rice dataset.

4.7. Comparison with Create ML Models

Comparing PyTorch-trained and converted models (PyWC, PyWAC, PyRC, PyRAC) with Create ML-trained models (CoreW, CoreWA, CoreR, CoreRA) confirms that PyTorch-trained models consistently outperform Create ML models, particularly in weather classification.

For weather classification, PyWAC (89.04%) and PyWC (85.34%) achieved higher accuracy than CoreWA (87.06%) and CoreW (86.39%), reinforcing hypothesis H4. For rice classification, PyRAC (98.56%) and PyRC (98.02%) outperformed CoreR (95.03%), showing that PyTorch-trained models provided greater accuracy. Create ML models struggled more with classifying similar rice varieties, though the highly structured dataset minimized major discrepancies.

These findings confirm that training models in PyTorch before converting them to Core ML results in better performance than training directly in Create ML, particularly for complex datasets like weather classification. Create ML remains a viable alternative for simpler tasks but lacks the fine-tuned optimization and transfer learning advantages of PyTorch.

4.8. Hypothesis Verification

The results confirm all hypotheses:

- H1 & H2: PyTorch and Core ML models achieved over 85% accuracy in classifying both weather conditions and rice grains,
- H3: Conversion from PyTorch to Core ML did not degrade performance, as PyWC/PyWAC and PyRC/PyRAC performed similarly to their original PyTorch versions,
- H4: PyTorch-trained and converted models outperformed Create ML models, particularly in weather classification, where PyWAC (89.04%) exceeded CoreWA (87.06%),
- The positive impact of data augmentation further reinforces the benefits of this preprocessing technique in improving classification outcomes.

Table 2: Results of all models

Model name	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
PyR	98.12	98.13	98.12	98.12
PyRA	98.51	98.51	98.51	98.51
PyW	85.27	86.30	84.46	84.99
PyWA	89.11	90.29	89.09	89.63
PyRC	98.02	98.03	98.02	98.02
PyRAC	98.56	98.57	98.57	98.57
PyWC	85.34	86.35	84.51	85.03
PyWAC	89.04	90.34	89.05	89.63
CoreR	95.03	95.6	95.2	95
CoreRA	n/a	n/a	n/a	n/a
CoreW	86.39	86.9	86.54	86.45
CoreWA	87.06	87.45	87.18	87.45

5. Conclusions

The PyTorch framework proved to be a good tool for training models, and when combined with the coremltools library, it allows lossless conversion to a Core ML format. On the other hand, models trained on the same data in Create ML achieved similar results as in the case of weather data without data augmentation, with a difference of 1.05% accuracy in favor of the version created directly in Core ML format. The PyTorch model trained on rice data achieved 2.99% better accuracy, and 1.98% better accuracy with augmented weather data. Unfortunately, it was not possible to train the model on augmented rice data because the Create ML program used up huge amounts of RAM, crashing the entire process which was possible in the Python environment.

The model trained on the Weather Image dataset subjected to augmentation achieved better results in the metrics: accuracy, precision, recall and F1 score by an average of 4%-5% showing how important the quality of the dataset can be in model performance. The model's images had different sizes, resolutions, aspect ratios, and the key elements in these images had different sizes and alignments, which the augmentation handled well improving the results. The model trained on the Rice Image set is an example of a high-quality dataset. The model without augmentation achieved a very high score, and augmentation alone did not significantly improve. This is due to the fact that each image contains a single grain of rice, keeping the background, lighting and position of the subject in the frame uniform. By applying the early stopping mechanism to prevent overfitting, we can see that, using the example of Pytorch model graphs (see Figure 3-6) on rice data, the number of epochs needed to train the model has decreased from 35 to 20 after data augmentation.

References

[1] S. M. Anwar, M. Majid, A. Qayyum, M. Awais, M. Alnowami, M. K. Khan, Medical image analysis using convolutional neural networks: a review, Journal of Medical Systems 42 (2018) 1–13, https://doi.org/10.1007/s10916-018-1088-1.

[2] M. Skublewska-Paszkowska, P. Powroźnik, E. Łukasik, J. Smołka, Tennis Patterns Recognition Based on a Novel Tennis Dataset – 3DTennisDS, Advances in Science and Technology Research Journal 18(6) (2024) 159-176, http://dx.doi.org/10.12913/22998624/191264.

[3] M. Skublewska-Paszkowska, P. Powroźnik, Temporal Pattern Attention for Multivariate Time Series of Tennis Strokes Classification, Sensors 23(5) (2023) 1-16, https://doi.org/10.3390/s23052422.

[4] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. W. M. van der Laak, B. van Ginneken, C. I. Sánchez, A survey on deep learning in medical image analysis, Medical Image Analysis 42 (2017) 60–88, https://doi.org/10.1016/j.media.2017.07.005.

[5] M. Skublewska-Paszkowska, P. Powroźnik, R. Rejdak, K. Nowomiejska, Application of Convolutional Gated Recurrent Units U-Net for Distinguishing between Retinitis Pigmentosa and Cone-Rod Dystrophy, Acta Mechanica et Automatica 18(3) (2024) 505-513, http://dx.doi.org/10.2478/ama-2024-0054.

[6] P. Powroźnik, M. Skublewska-Paszkowska, K. Nowomiejska, A. Aristidou, A. Panayides, R. Rejdak, Deep convolutional generative adversarial networks in retinitis pigmentosa disease images augmentation and detection, Advances in Science and Technology Research Journal 19(2) (2025) 321-340, http://dx.doi.org/10.12913/22998624/196179.

[7] K. Nowomiejska, P. Powroźnik, M. Skublewska-Paszkowska, K. Adamczyk, M. Concilio, L. Sereikaite, R. Zemaitiene, M. D. Toro, R. Rejdak, Residual Attention Network for distinction between visible optic disc drusen and healthy optic discs, Optics and Lasers in Engineering 176 (2024) 1-12, https://ui.adsabs.harvard.edu/link_gateway/2024OptLE.176608056N/doi:10.1016/j.optlaseng.2024.108056.

[8] A. Kamilaris, F. X. Prenafeta-Boldú, Deep learning in agriculture: A survey, Computers and Electronics in Agriculture 147 (2018) 70–90, https://doi.org/10.1016/j.compag.2018.02.016.

[9] Y. Zhang, L. Deng, H. Zhu, W. Wang, Z. Ren, Q. Zhou, S. Lu, S. Sun, Z. Zhu, J. M. Gorriz, S. Wang, Deep learning in food category recognition, Information Fusion 98 (2023) 101859, https://doi.org/10.1016/j.inffus.2023.101859.

[10] H. L. Dawson, O. Dubrule, C. M. John, Impact of dataset size and convolutional neural network architecture on transfer learning for carbonate rock classification, Computers & Geosciences 171 (2023) 105284, https://doi.org/10.1016/j.cageo.2022.105284.

[11] M. Skublewska-Paszkowska, P. Powroźnik, E. Łukasik, Attention Temporal Graph Convolutional Network for Tennis Groundstrokes Phases Classification, IEEE International Conference on Fuzzy Systems (FUZZ) (2022) 1-8, https://doi.org/10.1109/FUZZ-IEEE55066.2022.9882822.

[12] M. Skublewska-Paszkowska, P. Powroźnik, E. Łukasik, Learning Three Dimensional Tennis Shots Using Graph Convolutional Networks, Sensors 20(21) (2020) 1-12, https://doi.org/10.3390/s20216094.

[13] K. Sharifani, M. Amini, Machine learning and deep learning: A review of methods and applications, World Information Technology and Engineering Journal 10(07) (2023) 3897–3904.

[14] M. A. Berwo, A. Khan, Y. Fang, H. Fahim, S. Javaid, J. Mahmood, Z. Ul Abideen, M. S. Syam, Deep learning techniques for vehicle detection and classification from images/videos: A survey, Sensors 23(10) (2023) 4832, https://doi.org/10.3390/s23104832.

[15] Y. Gao, X. Xue, G. Qin, K. Li, J. Liu, Y. Zhang, X. Li, Application of machine learning in automatic image identification of insects-a review, Ecological Informatics 80 (2024) 102539, https://doi.org/10.1016/j.ecoinf.2024.102539.

[16] M. H. Guo, T. X. Xu, J. J. Liu, Z. N. Liu, P. T. Jiang, T. J. Mu, S. H. Zhang, R. R. Martin, M. M. Cheng, S. M. Hu, Attention mechanisms in computer vision: A survey, Computational Visual Media 8(3) (2022) 331–368, https://doi.org/10.1007/s41095-022-0271-y.

[17] M. Skublewska-Paszkowska, P. Powroźnik, M. Barszcz, K. Dziedzic, Dual Attention Graph Convolutional Neural Network to Support Mocap Data Animation, Advances in Science and Technology Research Journal 17 (5) (2023) 313-325, http://dx.doi.org/10.12913/22998624/171592.

[18] G. Huang, G. B. Huang, S. Song, K. You, Trends in extreme learning machines: A review, Neural Networks 61 (2014) 10–26, https://doi.org/10.1016/j.neunet.2014.10.001.

[19] M. J. Maurício, I. Domingues, J. Bernardino, Comparing vision transformers and convolutional neural networks for image classification: A literature review, Applied Sciences 13(9) (2023) 5521, https://doi.org/10.3390/app13095521.

[20] O. Qayyum, M. Şah, IOS mobile application for food and location image prediction using convolutional neural networks, In 2018 IEEE 5th International Conference on Engineering Technologies and Applied Sciences (ICETAS) (2018) 1–6, https://doi.org/10.1109/ICETAS.2018.8629202.

[21] H. Hendrick, W. Zhi-Hao, C. Hsien, C. Pei-Lun, G. J. Jia, IOS mobile APP for tuberculosis detection based on chest X-ray image, In 2019 2nd International Conference on Applied Information Technology and Innovation (ICAITI) (2019) 122–125, https://doi.org/10.1109/ICAITI48442.2019.8982152.

[22] R. Kaur, R. Kumar, M. Gupta, Deep neural network for food image classification and nutrient identification: A systematic review, Reviews in Endocrine and Metabolic Disorders 24(4) (2023) 633–653, https://doi.org/10.1007/s11154-023-09795-4.

[23] J. Bharadiya, Convolutional neural networks for image classification, International Journal of Innovative Science and Research Technology 8(5) (2023) 673–677, https://doi.org/10.5281/zenodo.7952030.

[24] M. Koklu, Rice Image Dataset, Kaggle, https://www.kaggle.com/datasets/muratkokludataset/rice-image-dataset, [12.01.2025].

[25] J. Bhathena, Weather Dataset, Kaggle, https://www.kaggle.com/datasets/jehanbhathena/weather-dataset, [12.01.2025].

[26] Albumentations, Image Augmentation Library, https://albumentations.ai/, [12.01.2025].

[27] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2016) 770–778, http://dx.doi.org/10.1109/CVPR.2016.90.

[28] D. Chicco, G. Jurman, The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation, BMC Genomics 21(6) (2020) 1–13, https://doi.org/10.1186/s12864-019-6413-7.