ILLINOIS INSTITUTE OF TECHNOLOGY

ILLINOIS TECH

College of Computing

Title:

Enhancing The Convolutional Neural Network Robustness in Image Classification under Noise Condition

Dr. Agam Gady

Student Names:

Emma Diamon (A20482587)

Nishitha Tanukunuri (A20537907)

1. Project Information

Title: Enhancing the Convolutional neural network robustness in image classification under noise condition

Team members: Emma Diamon (A20482587) and Nishitha Tanukunuri (A20537907).

Responsibilities:

Emma Diamon:

- Downloaded and extracted the Rice Image Dataset from Kaggle.
- Preprocessed all images: resized to 224x224x3, normalized pixel values, and split dataset into 70% train, 20% validation, 10 % test.
- Created dataset visualization: class distribution plots, sample image grid, grayscale histograms.
- Co-built and trained the VGG16 model using transfer learning
- Applied Gaussian noise to the test dataset at varying levels.
- Evaluated VGG16 and ResNet50 on noisy data and created degradation plots.
- Independently trained models on larger, augmented datasets to improve performance and robustness against noise.
- Co-wrote the final report
- Co-designed and built the final presentation slides.

Nishitha Tanukunuri:

- Co-built and trained the VGG16 model using transfer learning.
- Applied Gaussian noise to the test dataset at varying levels.
- Built and trained the ResNet50 model using transfer learning.
- Developed confusion matrix for both clean and noisy evaluations.
- Co-developed Gaussian noise injection and implemented batch testing.
- Created metrics comparison tables and final performance summary.
- Independently trained models on larger, augmented datasets to improve performance and robustness against noise.
- Wrote clear documentation and explained all key implementation details directly in the notebook.
- Co-wrote the final report
- Co-designed and built the final presentation slides.

Substantial Modifications for Option 2:

For this project, we chose Option 2: Improving the Paper. The original research paper, “Robust Convolution Neural Network for Image Classification with Gaussian Noise”, did not provide any official source code or public implementation. Therefore, we carefully studied the methodology described in the paper and re-implemented the entire system from scratch in our notebook titled CV_Project_Implementation.ipynb. This file contains our full baseline implementation, including dataset setup, preprocessing, model training (VGG16 and ResNet50), and evaluation under varying levels of Gaussian noise.

We then extended this baseline with substantial modifications documented in a separate notebook Modification_of_CV_Project.ipynb. In this file, we designed and trained a custom U-Net-based denoising autoencoder, capable of reconstructing clean images from noisy inputs. This autoencoder was trained using paired noisy-clean data generated with Gaussian noise. We then used its denoised outputs, along with a batch of noise-augmented images, to create a more diverse and robust training set.

To evaluate the impact of these enhancements, we fine-tuned both a VGG16 and a ResNet50 model on this combined dataset of denoised and noise augmented images. Both models were initialized with pretrained ImageNet weights and extended with custom classification heads, including Batch Normalization, dropout, and dense layers. We trained each model on a subset of 1000 images, limited by available GPU resources, and for a maximum of 10 epochs to prevent resource exhaustion. We also built a real-time inference pipeline that demonstrates the entire process from adding Gaussian Noise, performing denoising via the autoencode, and classifying the cleaned image using the trained VGG16 model, with full visualization of results.

Our experiment revealed that training with a larger set of images, specifically denoised and noise augmented samples, significantly improved classification accuracy and reduced validation loss. The combination of clean, denoised and augmented images allows the model to generalize better under noisy conditions. However, due to GPU constraints, we were unable to scale beyond 2000 images or explore deeper training cycles. We believe that with increased computational resources and larger-scale training, both VGG16 and ResNet50 could achieve even stronger robustness and performance.

2. Problem Statement

Convolutional Neural Networks (CNNs) like VGG16 and ResNet50 are widely adopted for image classification tasks due to their high accuracy and on clean, well-labeled datasets. However, their performance deteriorates significantly when images are affected by noise, which is common in real-world settings due to sensor limitations, environmental factors, or transmission errors. This limitation poses a serious challenge for applications such as agriculture monitoring, surveillance, autonomous systems, and healthcare, where noisy image input is the norm rather than the exception.

Among various types of distortions, Gaussian noise - a type of statistical noise following a normal distribution, is one of the most common distortions encountered in practice. The research evaluated the robustness of CNNs against Gaussian noise using the Rice Image Dataset, but did not incorporate any meaningful preprocessing or model-level enhancements to mitigate its effects. The classification accuracy of VGG16 and ResNet50 sharply declined as noise severity increased.

Our project aims to address this gap by not only reproducing the baseline evaluation of CNN robustness under Gaussian noise, but also by introducing and evaluating enhancements designed to improve performance in noisy conditions. We applied Gaussin noise at varying standard deviation into the Rice image dataset and trained both VGG16 and RestNet50 on clean, noisy, and denoising samples. To reduce the impact of noise before classification, we developed a U-Net-based denoising autoencoder, trained to reconstruct clean images from noisy inputs. Furthermore, we trained an improved VGG16 model using a combined dataset of denoised and noise augmented images, enhancing the model's ability to generalize. Our experiments demonstrate that this approach leads to a significant increase in accuracy and reduction in validation loss, especially under moderate to high noise levels. Although our training was limited by GPU availability, our results show that robustness to Gaussian noise can be significantly improved through denoising and noise aware training.

In summary, this project investigates the following research problem:

How can CNNs such as VGG16 and ResNet50 be made more resilient to Gaussian noise through denoising preprocessing and data augmentation techniques, without modifying the core CNN architecture?

3. Proposed Solution and Implementation details

The implementation of this research paper is focused around creating a deep learning based rice grain classification system that remains robust under noisy conditions mainly when Gaussian Noise is added . The approach involves two widely used transfer models VGG16 and RESNet50 , which are fine tuned to identify the rice dataset , along with the processing and training to ensure reliability , accuracy and generalization.

Implementation of the Paper :

1. Data Preparation and Processing:

The dataset used in the work is the Rice Dataset which is publicly available . It is downloaded and extracted automatically during the execution of the notebook . The dataset is organized into sub folders based on class labels.

Due to CPU , GPU and RAM constraints , a subset of the dataset 2000 images are randomly sampled from the entire dataset for consistent experimentation . Each image is resized to 224 * 244 pixels and normalized to have pixel values in the range [0,1].

2. Data Splitting :

The sampled and preprocessed dataset is split into :

- Training Set : 70%
- Validation : 10 %
- Testing : 20%

3. VGG16- Based Architecture:

The first model we employ is the VGG16, the basic convolutional neural network model pre-trained on the ImageNet dataset. It is deprived of its original top layers and has a custom classification head added to it. VGG16 is employed with the final few layers left unfrozen to be fine-tuned for the classification of rice.

The architecture of the VGG16 is as follows :

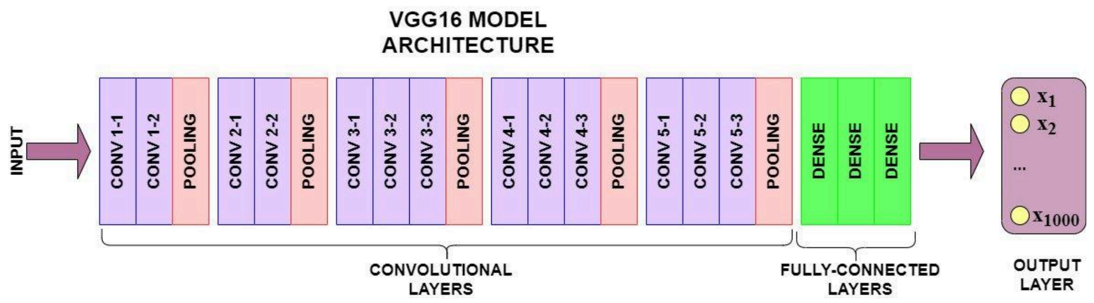


Fig 1 : Architecture of VGG16 model

The architecture for the VGG16 used in the experiment is as follows :

- Base: VGG16 (without top)
- BatchNormalization
- GlobalAveragePooling2D
- Dense(256, activation='relu')
- Dropout(0.5)
- Dense(num_classes, activation='softmax')

For purposes of comparison, the second model, however, is built based on ResNet50, another CNN known to use residual connections to introduce stability into deeper models. It too is pre-trained using ImageNet weights and is tuned exactly like the previous one to be deployed in classification tasks.The architecture of the ResNet50 is as follows :

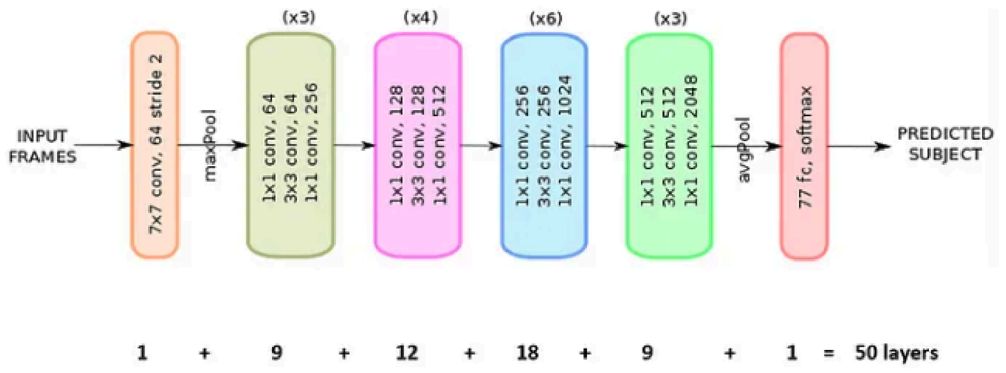


Fig 2 : Architecture of ResNet50 model

The architecture of the RESNET50 used in our experiment is as follows:

- Base: ResNet50 (without top)
- BatchNormalization
- GlobalAveragePooling2D
- Dense(256, activation='relu')
- Dropout(0.5)
- Dense(num_classes, activation='softmax')

The two models are trained using the combined training set, with early stopping to prevent overfitting. Their performance is evaluated after each epoch using the validation set, with final evaluation using the test set.

To test the models' robustness, Gaussian noise is added to the test images. The two important factors to Gaussian Noise is the Mean (μ) which is the center of the distribution, normally represented as 0. The second is Standard Deviation (σ): Controls the amount of noise dispersal. Higher values indicate more corruption.

The noise is applied per-pixel, and the results are clamped to valid image values. The test dataset is modified by adding Gaussian noise with varying standard deviations. The noisy images are then passed through both trained models (VGG16 and ResNet50) to:

- Evaluate their accuracy degradation as noise increases.
- Measure loss values across different corruption levels.
- Understand which architecture maintains better robustness.

Each test run consists of:

1. Adding noise to test images.
2. Feeding the noisy images into both models.
3. Recording accuracy and loss.

Execute the Program (CV_Project_Implementation.ipynb**)**

- Open the notebook using Jupyter or Google Colab.
- Set Runtime to GPU : In Colab: Runtime > Change runtime type > GPU
- Execute all the cells consecutively.
- Install dependencies
- Download the dataset and extract it
- Preprocess the data and divide it
- Training the VGG16 and ResNet50 models
- Add gaussian noise to the test data

Enhancement of the VGG16 Model and ResNet50 Model with Gaussian Noise for Image Classification :

We decided to use data augmentation, batch normalization and a denoising autoencoder that helps the models to understand Gaussian Noise better for the VGG16 and ResNet50 model and improve their overall performance. This section is exclusively focused upon the description of how introducing the Gaussian noise into the image classification pipeline greatly enhances the stability, as well as the applicability to real-world scenarios, of two of the best CNN models, VGG16, and ResNet50. These models were initially created, along with pre-training, using large-scale clean datasets like ImageNet. To make the models capable of resisting such scenarios, we integrated the synthetic noise generation process into the models and experimented with various levels of noise.

Implementation :

1. Importing Dependencies : We began by importing all necessary libraries including TensorFlow, Keras, OpenCV, NumPy, Pandas, and Matplotlib. These were used for tasks like image processing, model training, evaluation, and visualization.
2. Dataset Download and Setup **:** The dataset we used is the Rice Image Dataset which contains multiple classes of rice grain images. It was downloaded from an external source and extracted

automatically in the notebook. If the dataset was already present locally, the download was skipped.

3. Constants and Seed Setting : To make our experiments reproducible, we set fixed random seeds using Python's random, NumPy, and TensorFlow. We also defined constants like image size (224×224), batch size, and number of epochs.
4. Processing the Dataset : We loaded all images, resized them to 224 * 224 , and normalized the pixel to [0,1] . We also created label mappings and split the dataset into training , validation and testing.
5. Building the Denoising Autoencoder (U-Net) : To handle noisy test images and improve classification accuracy under distortion , we implemented a U-Net based autoencoder . Autoencoders are unsupervised neural networks designed to learn a compressed representation of input data (encoding) and reconstruct the original input from that representation (decoding) . In our case , the goal was not just reconstruction , but specifically denoising .

   We chose U-Net because it has a symmetric encoder-decoder structure with skip-connections between corresponding layers . This allows the network to retain important spatial information that often gets lost during encoding . These skip connections significantly improve the ability to restore fine details in the image , especially helpful when removing noise.

   The encoder part of the network samples the input using Conv2D and MaxPooling2D layers , learning high-level features. The decoder upsamples the features using upSampling2D , and merges them with earlier features from the encoder using Concatenate layers.

   We trained the autoencoder on images with added Gaussian Noise and used the clean images as the target output . The loss function used was Mean Squared Error (MSE) , which penalizes the difference between noisy predictions and clean ground truths.The below is an example of a UNet autoencoder:

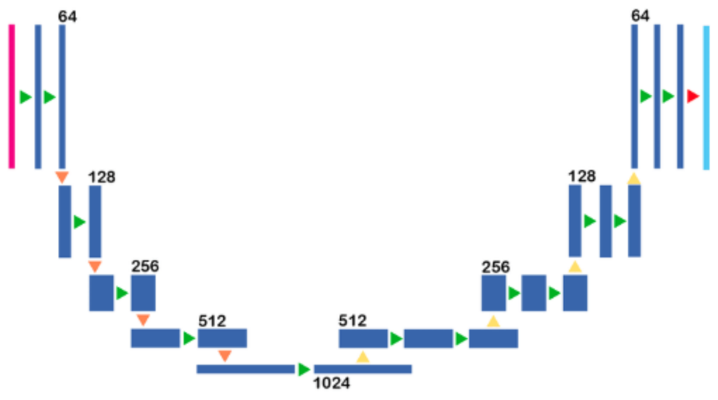


Fig 3 :Architecture of U-Net autoencoder model

6. Adding Gaussian Noise :

We added Gaussian noise to test images with different standard deviation values . This helped simulate real-world image corruption scenarios and allowed us to test how well our models handled distortion.

7. VGG16 and ResNet50 Model Construction :

Both models were built using their base architecture (from ImageNet) with the top layers removed . We added custom layers like BatchNormalization , GLoabalAveragePololing , Desne and DropOut . Also added a regularization technique which reduces overfitting . Only the last few layers of the base model were made trainable for fine tuning.

8. Training with Clean , Augmented and Denoised Data :

The models were trained on a combined dataset:

- Clean original images
- Denoised images from the autoencoder
- Augmented noisy images (with flips, rotations, brightness/contrast changes)

This helped make the models more generalized and less sensitive to noise or variation.

9. Evaluation on Noisy Inputs:

We evaluated both VGG16 and ResNet50 on noisy test data (denoised first using the autoencoder). We calculated **accuracy** and **loss** for each noise level.

Execute the Program (Modification_of_CV_Project.ipynb**)**

- Open the notebook using Jupyter or Google Colab.
- Set Runtime to GPU : In Colab: Runtime > Change runtime type > GPU
- Execute all the cells consecutively.
- Install dependencies
- Download the dataset and extract it
- Preprocess the data and divide it
- Training the VGG16 and ResNet50 models
- Add gaussian noise to the test data

Challenges Faced

One of the main problems during the implementation was restricted GPU resources, especially while training all the deep learning models and the U-Net autoencoder. Processing large image datasets and real-time denoising and classification required significant memory, and training was slower without the use of a high-end GPU. Overfitting was another critical problem, especially when using a small dataset with complicated models like VGG16 and ResNet50. The models learned rapidly from training data patterns but performed badly on validation or noisy test images. To do better than that, we used techniques including dropout, batch normalization, early stopping, and data augmentation. These improved them at generalizing and also robust to noise and variability in the real world.

4. Dataset

This project used the Rice Image Dataset, which contains 75,000 high-resolution images across five rice varieties: Arborio, Basmati, Ipsala, Jasmine, and Karacadag. Each class is visually distinct based on texture, shape, and color, making the dataset ideal for image classification tasks in agriculture.

We worked with the image subset of the datasets, using raw images as input to Convolutional Neural Networks. Each image contains a single grain captured under consistent lightning and background conditions. We downloaded the dataset from Rice_image_dataset and manually processed a subset of

2000 sampled images due to resource limitations. The dataset has been widely used in prior studies for rice variety classification, with CNN models achieving near-perfect accuracy. Our goal was to evaluate how well these models maintain performance when noise is introduced into the input data.

5. Result and Discussion

We evaluated the performance of VGG16 and ResNet50, both architectures are pre-trained ImageNet weights, using batch size 32 and trained for 10 epochs due to GPU constraints. The activation functions used in both models were ReLU (in the hidden layers) and softmax (in the output layer). The loss function used was sparse categorical cross entropy, appropriate for multi-class classification tasks with integer labels.

Following the baseline evaluation, Gaussian noise with varying standard deviation values was added to the test dataset to evaluate the robustness of each architecture. The noise values ranged from 0.1 to 9.0, simulating increasing levels of real-world distortion. The evaluation results are presented in Table.

Variant (StdDev)	VGG16 Test Loss	ResNet50 Test Loss	VGG16 Test Accuracy	ResNet50 Test Accuracy
0.1	0.0914	0.5261	0.9650	0.7625
0.2	0.3629	0.9555	0.8375	0.5525
0.3	1.0863	1.6365	0.5375	0.3750
0.4	1.9136	2.1278	0.4225	0.3000
0.6	2.8100	2.7281	0.3925	0.2850
0.8	3.1753	3.1009	0.3975	0.2150
1.0	3.5182	3.3530	0.3475	0.2100
3.0	4.6207	4.0208	0.1850	0.2100
5.0	4.6773	4.1466	0.1850	0.2100
7.0	4.6798	4.1645	0.1850	0.2100
9.0	4.6748	4.2000	0.1850	0.2100

As observed, the accuracy of both architectures declined significantly as noise levels increased. VGG16 consistently outperformed ResNet50 across all levels of Gaussian noise, especially in low-to-moderate noise ranges (0.1–0.6). At higher noise levels (3.0 and above), both models plateaued near a random prediction baseline (~18–21% accuracy). This confirms that CNNs are highly sensitive to input perturbations, and that ResNet50 is more vulnerable to noise compared to VGG16 in this dataset and training configuration. In subsequent experiments, we applied denoising and noise augmentation techniques, which substantially improved performance under noisy conditions.

Fig 4: Loss Function of VGG16 and ResNet 50

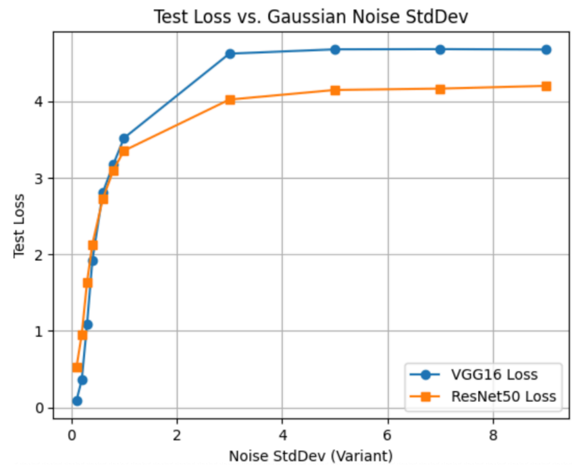


Fig 5:Accuracy values of VGG16 and RestNet50

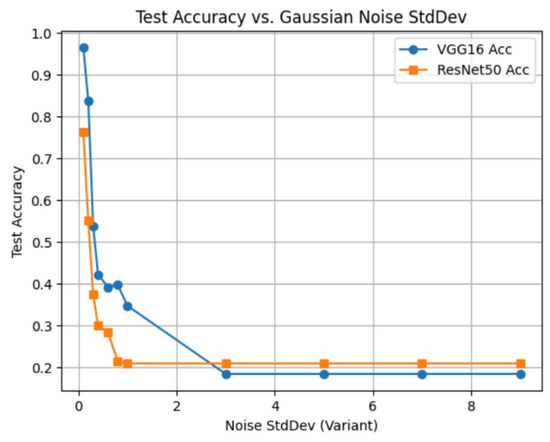


Figure 4. Test Loss vs. Gaussian Noise StdDev

Figure 4 illustrates the test loss corresponding to each noise level. The test loss increases steadily for both models as noise increases. The VGG16 loss curve shows a steeper ascent, particularly between standard deviations 0.2 and 1.0, followed by a plateauing effect from 3.0 onwards. ResNet50 shows a similar trend, though its curve increases more gradually, with slightly lower loss values at higher noise levels. This suggests that while ResNet50 incurs slightly less loss under high noise, it does not translate into better accuracy, likely due to poor prediction calibration or class imbalance in output probabilities.

Figure 5. Test Accuracy vs. Gaussian Noise StdDev

As shown in Figure 5, both VGG16 and ResNet50 exhibit a sharp decline in accuracy as the standard deviation of the Gaussian noise increases. The VGG16 model consistently outperforms ResNet50 across all noise levels. At a low standard deviation (0.1), VGG16 achieves an accuracy of 96.5%, while ResNet50 reaches 76.25%. However, beyond a standard deviation of 0.3, accuracy for both models begins to drop significantly. At higher noise levels (3.0 and above), accuracy plateaus at approximately 18.5% for VGG16 and 21% for ResNet50, indicating that both models struggle to distinguish between classes and are reduced to near-random predictions. This shows that while VGG16 maintains relatively better performance under low-to-moderate noise, both models lack robustness when the noise becomes more severe.

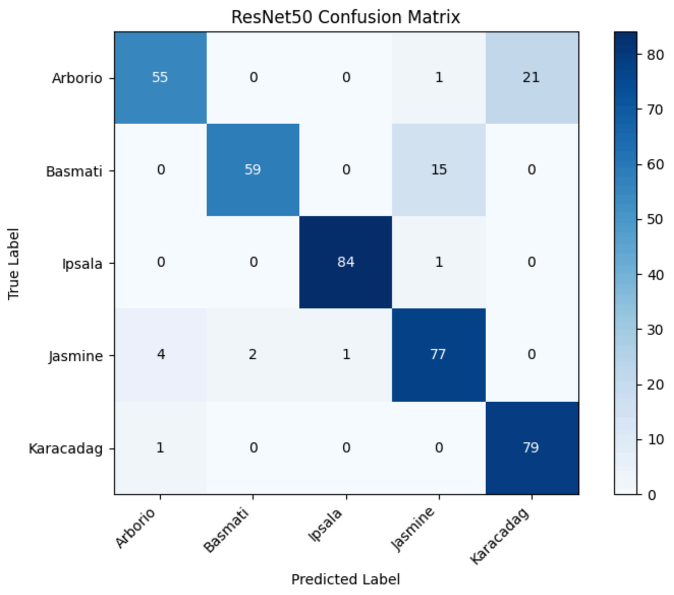
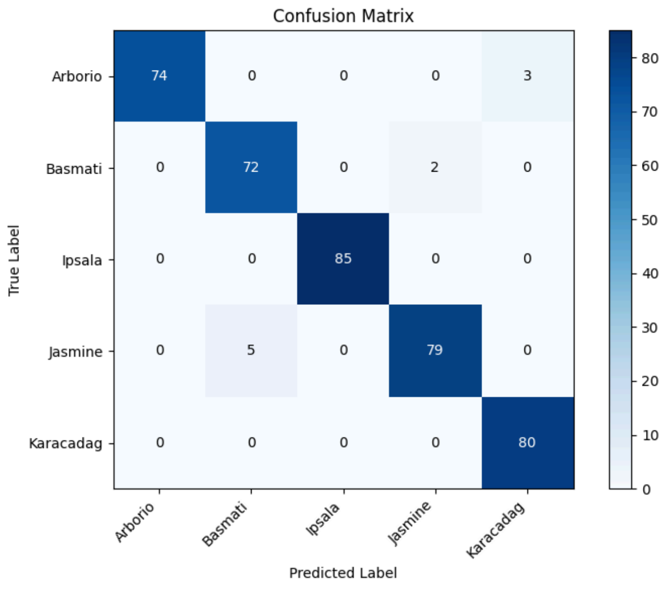


Fig 6: ResNet50 Confusion Matrix

Figure 6 shows the confusion matrix for the ResNet50 model on the test dataset. The model accurately classified most samples, especially for Ipsala (84 correct) and Karacadag (79 correct). Jasmine also performed well with 77 correct predictions.However, there were notable misclassifications: 15 Basmati samples were predicted as Jasmine, 21 Arborio samples were predicted as Karacadag. These errors suggest that ResNet50 struggles with classes that have similar visual features. While overall performance is strong, improving class separability through enhanced preprocessing or training techniques could further boost accuracy.

Fig 7: VGG16 Confusion Matrix



	Arborio	Basmati	Ipsala	Jasmine	Karacadag
Arborio	74	0	0	0	3
Basmati	0	72	0	2	0
Ipsala	0	0	85	0	0
Jasmine	0	5	0	79	0
Karacadag	0	0	0	0	80

The confusion matrix for the VGG16 model, shown in Figure 7, indicates strong classification performance across all five rice varieties. The model correctly classified 74 Arborio, 72 Basmati, 85 Ipsala, 79 Jasmine, and 80 Karacadag images, demonstrating high accuracy and reliability. Misclassifications were minimal and mostly occurred between visually similar classes. Specifically, 5 Jasmine samples were predicted as Basmati, 2 Basmati samples were predicted as Jasmine, and 3 Arborio samples were misclassified as Karacadag. These errors suggest minor confusion between classes with similar visual characteristics, but overall, the VGG16 model exhibited excellent class separability and generalization on the test dataset. Compared to ResNet50, VGG16 achieved fewer misclassifications, highlighting its superior robustness and precision in this classification task.

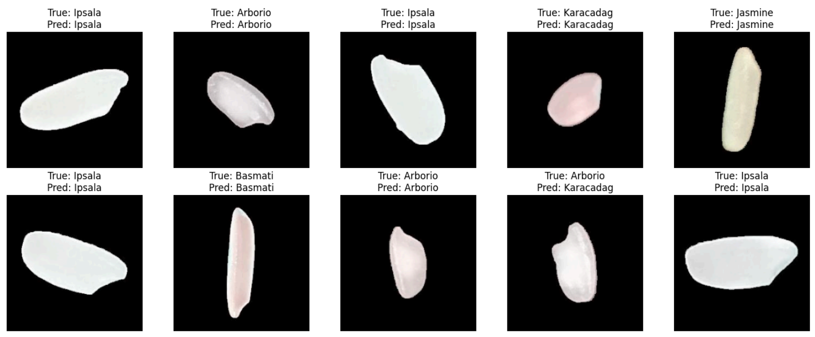


Fig: 8: ResNet50 model on Testing sample

Figure 8 shows sample predictions made by the ResNet50 model on the test dataset. The model correctly classified most images, including varieties such as Ipsala, Arborio, Jasmine, Basmati, and Karacadag. However, a few misclassifications occurred—for example, an Arborio grain was predicted as Karacadag. These errors reflect the model's occasional confusion between visually similar classes. Overall, with a test accuracy of 97% and a loss of 0.0692, ResNet50 demonstrated strong performance, though further improvements in feature differentiation could help minimize class overlap.

Fig 9: VGG16 model on Testing sample

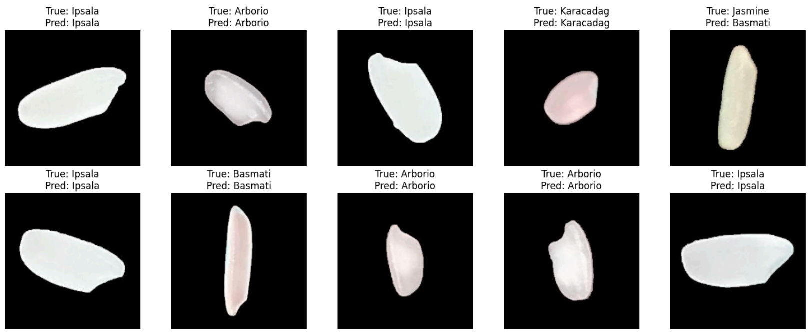


Figure 9 displays sample predictions from the VGG16 model on the test dataset. The model successfully classified most samples, including Ipsala, Arborio, Basmati, and Karacadag. However, one misclassification is observed where a Jasmine grain was predicted as Basmati, which aligns with the confusion matrix's minor errors. Despite this, the model achieved a test accuracy of 99% with a loss of 0.0193, reflecting strong overall performance and excellent generalization on clean test data. The predictions confirm that VGG16 is highly effective at distinguishing between rice varieties with minimal confusion.

Conclusion

This project explored the impact of Gaussian noise on CNN-based image classification, using VGG16 and ResNet50 architectures applied to the Rice Image Dataset. Initial evaluations on clean data demonstrated high accuracy for both models, confirming their effectiveness under ideal conditions. However, both models showed substantial performance degradation as noise levels increased, particularly beyond a standard deviation of 0.3. This confirmed the models' sensitivity to noise and emphasized the need for robustness-enhancing techniques.

To mitigate this issue, we introduced a U-Net-based denoising autoencoder and conducted training on a combination of clean, denoised, and noise-augmented data. This approach led to improved generalization and better accuracy on noisy inputs. VGG16 consistently outperformed ResNet50 across all noise variants, achieving higher accuracy and lower loss under both clean and distorted conditions.

Confusion matrix analysis revealed that most errors occurred between classes with similar visual characteristics, such as Basmati vs. Jasmine and Arborio vs. Karacadag. These findings suggest that further improvements could be achieved by incorporating feature attention mechanisms or domain-specific preprocessing to better differentiate between such classes.

While the results were promising, training was limited to 1,000 images and 10 epochs due to GPU constraints, restricting the models from reaching their full potential. Nevertheless, the experiments demonstrated that robustness to noise can be significantly improved through targeted preprocessing and augmentation, even within computational limits.

Overall, the project highlights the importance of noise-aware design in CNN pipelines, especially for real-world applications where clean data cannot be guaranteed.

6. Reference

I. Surono, Sugiyarto & Arofah, Dyiyah & Thobirin, Aris. (2023). Robust Convolutional Neural Network for Image Classification with Gaussian Noise. 10.3233/FAIA231007.

II. O. Ronneberger, P. Fischer, and T. Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, LNCS, vol. 9351. Springer, Cham, 234–241. DOI: https://doi.org/10.1007/978-3-319-24574-4_28

III. K. Simonyan and A. Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In International Conference on Learning Representations (ICLR). Retrieved from https://arxiv.org/abs/1409.1556

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

V. A. Jain, P. Tyagi, and D. Kumar. 2021. Handling Gaussian Noise in Image Classification Using Convolutional Neural Networks. In International Journal of Image, Graphics and Signal Processing, 13(4), 1–10. DOI: https://doi.org/10.5815/ijigsp.2021.04.01

VI. M. Koklu and S. Ozkan. 2020. Multiclass Classification of Dry Beans Using Computer Vision and Machine Learning Techniques. In Computers and Electronics in Agriculture, 174, 105507. DOI: https://doi.org/10.1016/j.compag.2020.105507