

Gheorghe Asachi Technical University of Iași
Faculty of Automatic Control and Computer Engineering
Field: Systems Engineering
Study Program: Machine Learning, Robotics and Control



Image Classification using AlexNet

Author:

Eng. Silvia-Teodora Porcărașu

Subject:

Machine Learning

Table of Contents

1. Abstract.....	3
2. Machine Learning.....	3
3. Convolutional Neural Networks (CNNs).....	4
4. AlexNet Convolutional Neural Network Model.....	5
5. Application	6
5.1. First Training - Clean Dataset with SGDM Optimizer.....	8
5.2. Second Training Setup - Dataset with Added Noise	9
5.3. Third Training - Clean Dataset with Adam Optimizer	10
5.4. Future Training Configurations.....	10
6. Bibliography.....	12

1. Abstract

This project explores the application of AlexNet, a Convolutional Neural Network (CNN), for image classification tasks. Using the Stanford Dogs Dataset, the dataset was reduced to eight classes to streamline the training process. Images were preprocessed through resizing and normalization to match AlexNet's input requirements. The network was trained under three different configurations: clean data with SGDM optimizer, noisy data with SGDM, and clean data with Adam optimizer. The experiments revealed how dataset quality and optimizer choice significantly affect model performance. SGDM delivered the best results on clean data. The findings emphasize AlexNet's strength with clean inputs and provide insights for optimizing CNN training strategies in diverse conditions.

2. Machine Learning

Machine Learning (ML) is the process by which a computing system learns from a dataset or "past experiences" to make predictions or decisions about future data. Unlike living beings, which rely on evolution and a biological brain, computing systems require algorithms that allow them to observe data, recognize patterns, and build predictive models. This process enables machines to mimic the decision-making and learning behaviors of humans or animals, as described by Russell and Norvig in their introduction to example-based learning in Artificial Intelligence.

Inspired by the biological brain, many ML algorithms, such as neural networks, simulate the way neurons in our brain process information. These algorithms are particularly effective in solving complex problems, like recognizing images, predicting outcomes, or discovering hidden relationships in data. ML is not a static process; models evolve and improve as they are exposed to more data, refining their accuracy and reliability over time.

Machines learn in various ways depending on the problem and data type. Common methods include:

- **Supervised Learning:** The model is trained using labeled data, where each input is paired with the correct output. This method is used for tasks like image classification, spam detection, and medical diagnosis.
- **Unsupervised Learning:** This method works without labeled data, seeking hidden patterns in the data. It is often used for clustering, anomaly detection, and topic modeling.
- **Reinforcement Learning:** An agent interacts with an environment and learns by receiving feedback in the form of rewards or penalties. This method is used in applications like game AI and robotics.

In image classification, deep learning, particularly through Convolutional Neural Networks (CNNs), has revolutionized the field by enabling models to extract intricate patterns from raw image data, achieve human-like accuracy in identifying features and objects, and

adapt to diverse datasets using techniques like transfer learning and data augmentation. These advancements have significantly improved the performance and scalability of image classification tasks, allowing models to handle complex data and learn more effectively. In this project, AlexNet was used to classify eight dog breeds, leveraging its pre-trained capabilities to efficiently perform the task.

3. Convolutional Neural Networks (CNNs)

A Convolutional Neural Network (CNN) is a type of deep learning model specifically designed to process data in the form of multidimensional arrays. These models are highly effective for tasks involving images, audio spectrograms (2D arrays), and even video recordings or volumetric images (3D arrays). CNNs are structured with an input layer, a sequence of hidden layers, and an output layer. The hidden layers utilize the convolution operation, applying filters (also known as kernels) to the input data. These filters act as weight templates, which are applied repeatedly over local regions of the input signal or image from the previous layer. The convolution operation can be expressed mathematically for a 1D signal as follows:

$$(f * g) = \sum_{m=-\infty}^{\infty} f[m] \cdot g[n - m]$$

In this expression, f represents the signal or image, g is the filter (or kernel), and the resulting array is obtained by summing the products between elements of the input signal and the filter. The convolution operation helps identify defining features within a specific neighborhood of the signal or image.

The architecture of a typical CNN is organized into blocks, with the initial layers consisting of convolutional and pooling layers. The outcome of the convolution operation is called a feature map, which captures the relevant details from the input data. These feature maps are then passed through non-linear activation functions, such as ReLU (Rectified Linear Unit) or softplus, which introduce non-linearity into the model. The ReLU activation function is particularly popular due to its simplicity and efficiency in training deep networks. It can be mathematically expressed as:

$$f(x) = \max(0, x)$$

The use of ReLU has several advantages: it significantly speeds up training by allowing faster convergence, and it helps reduce overfitting by promoting sparse activation, which means fewer neurons are activated during training. This sparse activation improves the computational efficiency of the model.

Another important component in CNNs is the pooling layer, which serves to condense the feature maps produced by the convolutional layers. Pooling reduces the dimensionality of the data while retaining the most significant features. There are two main types of pooling operations:

- **Average-pooling:** The filter calculates the average value of the inputs within the defined region.
- **Max-pooling:** The filter extracts the maximum value from the input within the defined region.

4. AlexNet Convolutional Neural Network Model

AlexNet is a Convolutional Neural Network (CNN) introduced by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton in 2012, and it had a profound impact on the field of computer vision. It won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012, surpassing previous models by a considerable margin in terms of accuracy. The model consists of several layers, including convolutional layers, pooling layers, fully connected layers, and a final softmax layer for classification.

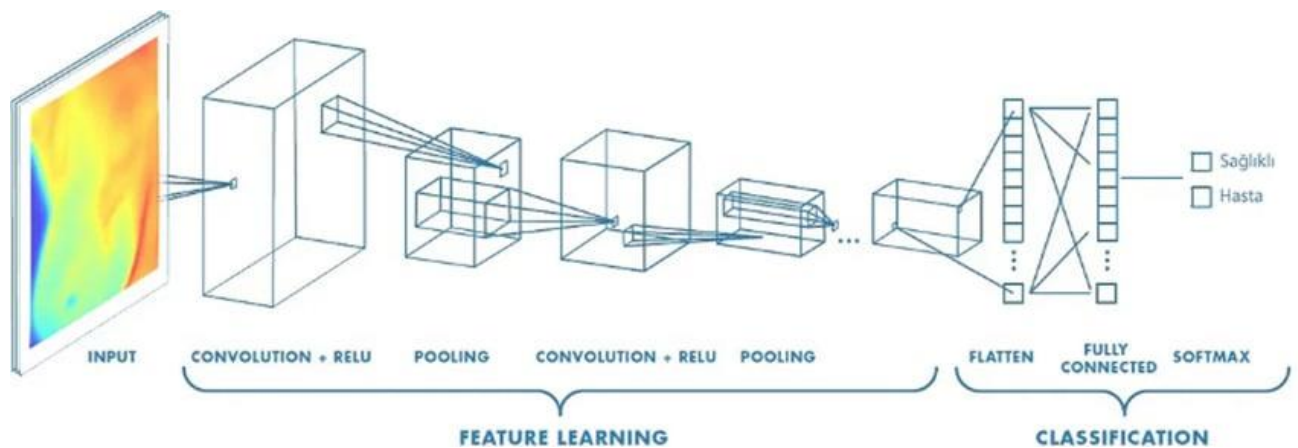


Figure 1 - AlexNet's architecture

Figure 1 illustrates the flow of data through a Convolutional Neural Network, specifically the AlexNet model, highlighting the main stages from input data to classification. The image, as represented at the model's input, can depict anything from a photograph to an audio spectrogram. Before being fed into the model, the image is generally preprocessed by resizing and normalising to match the expected input dimensions of the network.

The convolutional layer, together with the ReLU activation function, is crucial within the AlexNet model as it extracts and introduces nonlinearity into the dataset features. Each component serves a specific purpose and contributes additionally to the processing of information and feature learning. The convolutional layer applies filtering to the input data. Each filter represents a small matrix of learnt values. This matrix is passed over the input image, performing a product calculation between the filter and portions of the image. In this way, feature maps are generated. The ReLU activation function is applied to each value in the feature

map to transform negative values into zeros, introducing nonlinearity into the network, which helps in learning more complex patterns.

AlexNet contains 5 convolutional layers, each followed by the ReLU activation function. The number of filters applied in each layer varies from 96 to 384. The pooling layer is applied to reduce the spatial dimensions of the feature maps by compressing the information from a small region into a single value. A common technique in this layer is max pooling. It selects the maximum value from each pooling region, reducing the dimensions and computational complexity while retaining important information. AlexNet includes 3 pooling layers interspersed between the convolutional layers.

Once feature extraction is completed and their dimensions are reduced, the three-dimensional feature maps are transformed into a one-dimensional vector. This process prepares the data for the next step. The fully connected layers are essential to this model. They work similarly to those in a traditional neural network, where every neuronee in the previous layer is connected to every neuronee in these layers. This facilitates decision-making by enabling the combination of the extracted features. The classification error is reduced by adjusting the weights of these connections during training. The raw scores are transformed into a probability vector using the softmax function in the final layer. The probability of each input belonging to a possible class is represented by each element in this vector. The softmax function makes it easier to interpret the output values as probabilities, ensuring their positivity and summing them to 1. The final step is the classification, based on the output values from Softmax, where the network produces a final prediction consisting of the probability for each class.

5. Application

For this project, I utilised the Stanford Dogs Dataset, which contains high-resolution images of dog breeds. To tailor the dataset for my specific use case, I reduced it to include only eight classes: Golden Retriever, Collie, Doberman, Saint Bernard, Samoyed, Pomeranian, Chow Chow, and African Hunting Dog. This reduction allowed me to focus on a subset of breeds, balancing distinct visual features and classification challenges.

Before settling on the Stanford Dogs Dataset, I initially considered using a dataset containing chest X-rays, which included images of lungs affected by COVID-19, pneumonia, and other conditions. However, during early experiments, as seen in Figure 2, the model exhibited significant overfitting, likely due to the limited size of the dataset and the complexity of medical imaging. To address this challenge, I decided to switch to the Stanford Dogs Dataset, which provided a larger and more balanced collection of images.

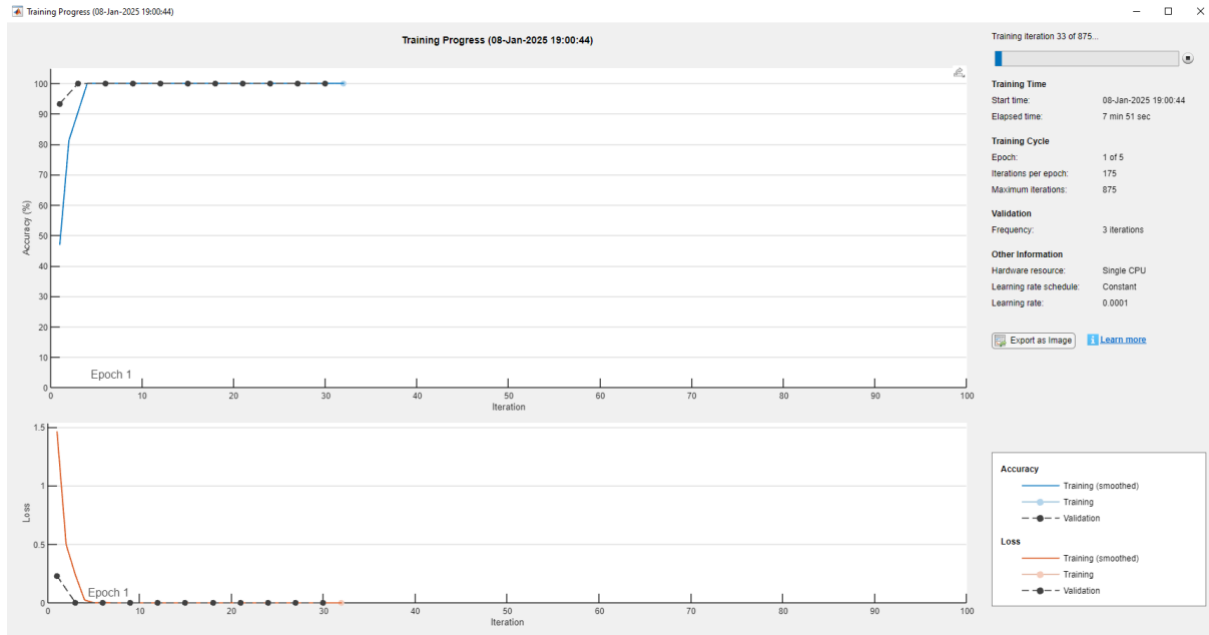


Figure 2 - Overfitting Example

For implementing this project, I used MATLAB, a versatile platform widely used for data analysis, visualisation, and deep learning. MATLAB provided built-in functions and a user-friendly environment to preprocess the images, design the AlexNet architecture, and train the model efficiently. Its Deep Learning Toolbox facilitated the customisation of layers and hyperparameters, ensuring a streamlined workflow for my experiments.

In the first stage of the project, I focused on data preprocessing, which involved several key steps to prepare the dataset for training. The dataset, consisting of images, was first loaded into MATLAB using the *imageDatastore* function, with the path pointing to the training and testing images. I then resized the images to the input size required for the AlexNet model, which is 227x227 pixels with three colour channels. This was done by creating a custom function, ensuring all images were standardised in size before training.

Next, I split the dataset into two parts: a training set (70% of the data) and a validation set (30%). To enhance the generalisation of the model, I applied data augmentation to the training images using random reflections and translations within a specified pixel range. This increased the diversity of the data and helped reduce overfitting. The augmented images were stored in a new datastore, which was then used for training the model. For the testing phase, I followed the same approach, creating a datastore for the test images and resizing them to the required input size.

The following step involved designing the architecture by loading the pre-trained AlexNet model and adapting it for my dataset. I removed the last few layers of AlexNet to modify the model's final fully connected layer according to the number of classes in my dataset (8 dog breeds). This model was then set up for transfer learning, where the lower layers of AlexNet were frozen, and only the last fully connected layer was retrained for my specific classification task.

5.1. First Training - Clean Dataset with SGDM Optimizer

In the first training setup, I used the original, clean dataset without adding any noise. The images were preprocessed and resized, and the model was trained using the Stochastic Gradient Descent with Momentum (SGDM) optimizer. This allowed the network to focus on the clear features of the images. The absence of noise made it easier for the model to learn and achieve high accuracy. The clean data ensured that the model could identify key patterns effectively.

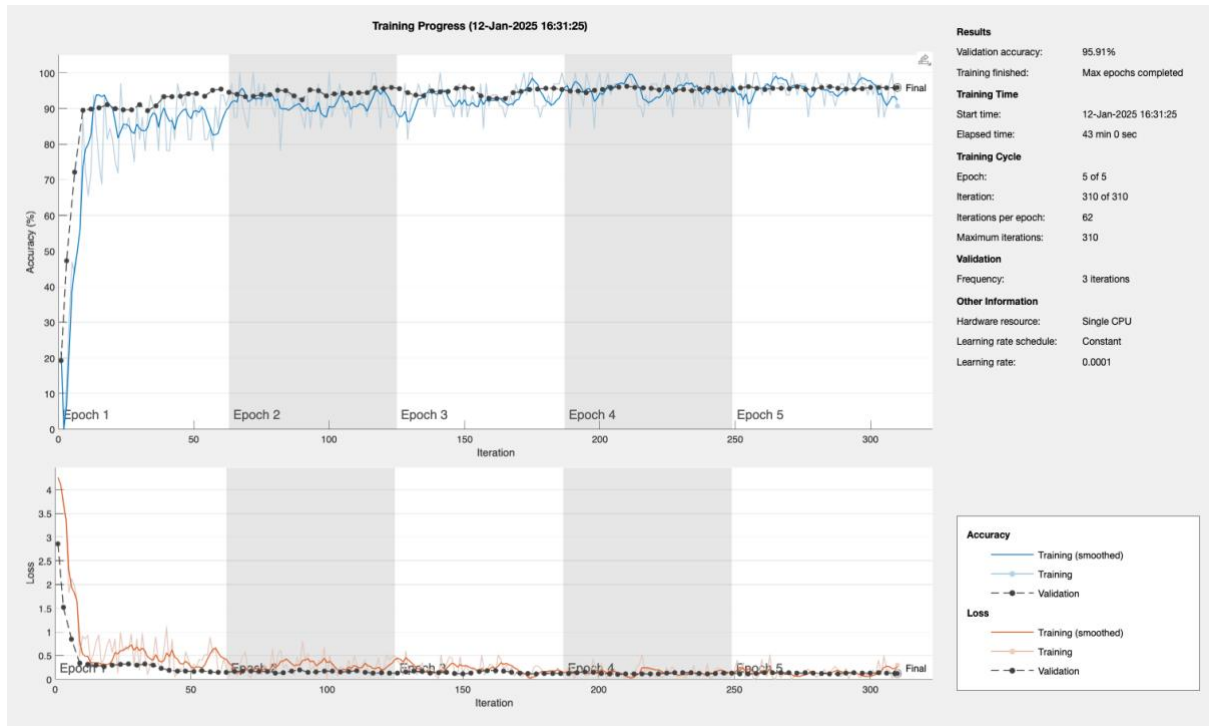
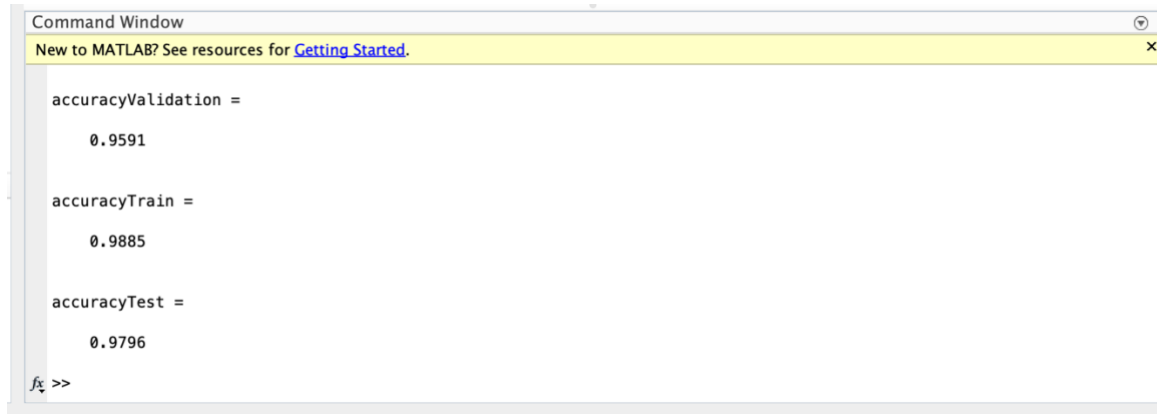


Figure 3 - First Training

Figure 3 presents the training progress over five epochs using the clean dataset. The top graph shows the accuracy trend, with the training accuracy (blue line) rapidly improving in the early epochs and stabilising near 100%. The validation accuracy (black dashed line) follows a similar path, reaching a final value of 95.91%. The bottom graph displays the loss trend, where both the training and validation loss decrease sharply in the first epoch and remain low throughout, stabilising at the end of the training. This smooth convergence indicates effective learning. Additional training parameters include a constant learning rate of 0.0001 and a mini-batch size of 32, both contributing to a stable and balanced optimization process.

A screenshot of the MATLAB Command Window. At the top, there is a yellow banner that reads "New to MATLAB? See resources for [Getting Started](#)." Below the banner, the command window displays the following text:

```
accuracyValidation =  
    0.9591  
  
accuracyTrain =  
    0.9885  
  
accuracyTest =  
    0.9796
```

 At the bottom left, the prompt "fx >>" is visible.

Figure 4 - First Training: Validation, Train and Test Accuracy

Figure 4 displays the accuracy metrics achieved during the first training setup. The training accuracy reached an impressive 98.85%, demonstrating the model's ability to learn effectively from the clean dataset. The validation accuracy achieved a final value of 95.91%, indicating strong generalisation performance on unseen validation data. Additionally, the test accuracy was recorded at 97.96%, confirming that the model performed consistently across the dataset splits.

5.2. Second Training Setup - Dataset with Added Noise

For the second training, I introduced Gaussian noise with a standard deviation of 0.05 to the dataset after normalising the image pixel values to the range [0, 1]. This ensured the noise remained proportionate to the normalised values, creating realistic variability while preserving the integrity of the data. The model achieved a training accuracy of 55.31%, a testing accuracy of 40.21%, and a validation accuracy of 39.88%. The training progress, as shown in Figure 5, illustrates a slower but consistent learning curve, indicating the model's effort to adapt to the noisy input data.

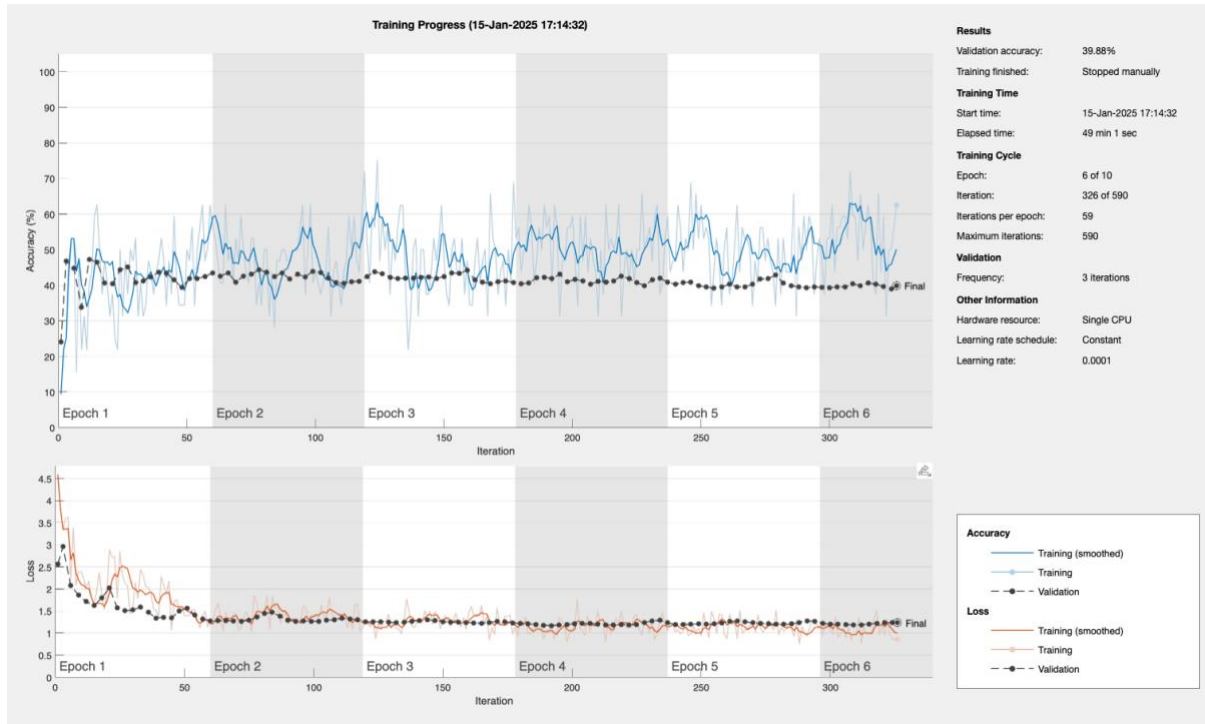


Figure 5 – Second Training

5.3. Third Training - Clean Dataset with Adam Optimizer

For the third training setup, I reverted to using the clean dataset but switched to the Adam optimizer, known for its adaptive learning rate capabilities. The training achieved accuracies of 53.82%, 53.37%, and 52.33% for the validation, training, and test sets, respectively. These percentages, shown in Figure 6, reflect the model's performance under this configuration and highlight the optimizer's impact on convergence. However, it's important to note that I did not allow the training process to finish completely because it was taking too long, which could be a reason why the accuracy isn't as high as expected.



Figure 6- Third Training: Validation, Train and Test Accuracy

5.4. Future Training Configurations

In future configurations, reducing noise will be a crucial step to improve model performance and enable it to learn more effectively from clean data. Instead of applying uniform noise, it could be adjusted based on the magnitude of the normalized images, allowing the model to focus more on significant features and reduce the impact of noise on the learning process.

To achieve this, it will be necessary to analyze the histograms of the normalized images to evaluate the distribution of pixel values. Histograms will provide a clear picture of pixel intensities and their distribution in each image. Adjusting noise according to these values will help apply noise in a more controlled manner, ensuring that it does not disrupt the useful data too much.

Mean squared error (MSE) is another valuable tool in this process. MSE measures the difference between predicted and actual values of the model, offering a precise measure of error. Monitoring MSE will help observe the impact of noise on model performance during training and adjust it to minimize errors. A lower MSE indicates better performance and improved model generalization.

Additionally, spectrograms can be used to determine the magnitude of the signal, and this could also be applied to images. A spectrogram transforms the signal (or image) into a format that can be analyzed to extract information about its frequencies and intensities. By applying a filter based on the spectrogram, the noise could be adjusted to align with the magnitude of the original signal. This could help preserve important features of the image while reducing the noise effect.

Therefore, this approach will not only reduce noise but also enhance the model's ability to learn from clean data and generalize more effectively on unseen datasets.

6. Bibliography

1. Zhou, B., & Xu, C. (2020). A survey on image data augmentation techniques in deep learning. *IEEE Transactions on Image Processing*, 29, 1300-1320.
2. <https://medium.com/@tuncerergin/convolutional-neural-network-convnet-yada-cnn-nedir-nasil-calisir-97a0f5d34cad>
3. https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/?utm_source=chatgpt.com