Comparison of Increase in Accuracy of Classification in Image Recognition Models Using Different Gaussian Filtering Algorithms Under Gaussian Noise

Junyoung Kim

abstract

Many studies focus on comparing two different noise filtering algorithms, to observe their effectiveness under different noise conditions However, not a lot of studies consider the varying effectiveness between different variants of the same type of algorithm. The aim of the study is to show how effective each Gaussian different filter variant is at reducing Gaussian noise by assessing the accuracy, and moreover finding which Gaussian filter variant increases how sure the model is with its classification by assessing the confidence level. A Random Forest Classifier was trained using a controlled image test data dataset. Each image had Gaussian noise applied to them and then was processed through different Gaussian filter variants before getting classified as test data by the model. The results show that Gaussian filters improve the performance of the model by reducing Gaussian noise — increasing the accuracy from approximately 46% to an average value of approximately 50%, when anomalies are removed. It also increases the confidence level of each classification from an average of 33% to an average value of approximately 45%. Furthermore, it was concluded from the results that, the most effective Gaussian filter variant at increasing the accuracy and the confidence level of the classifications was the were two different filters: the Gaussian pyramid denoise for accuracy with approximately 54%, and the recursive Gaussian filter for confidence with an average of 56%. Overall, the best performance across both fields was the Gaussian pyramid denoise filter variant with a 54% for accuracy (highest accuracy recorded) and a 50% in confidence.

I. Introduction

Image classification technologies rely on visual input of high quality in order to accurately classify different objects and patterns. However, images can have noise. Noise is defined as random, unintended distortions within the image that degrade the image quality, negatively affecting the model's performance. One type of image distortion is Gaussian noise. This is a form of noise where every pixel gets a small random variation. This type of noise can add distortion to the image and decrease the accuracy of classification for the model.

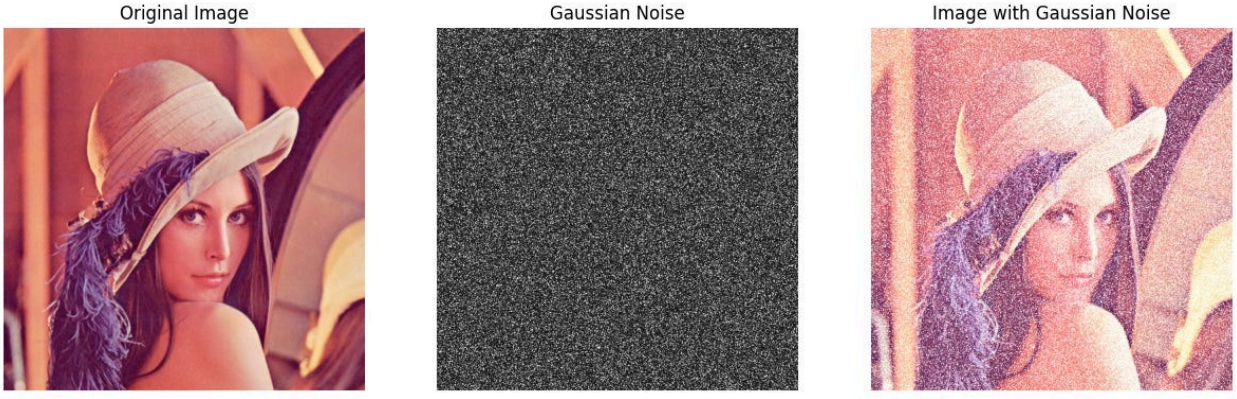

Fig 1. Comparison of original image and Gaussian noise applied image.

As a solution, noise reduction filters, a type of preprocessing technique, is used to improve the quality before inputting the image into the classification model. The most common type of filter is

Gaussian filter. Gaussian Filters are known for the widely used smoothing methods that decrease the amount of variation in pixel intensity by averaging neighboring values. Although this type of filtering algorithm is effective for some types of noises, the question of whether Gaussian filters stand up to the standard for Gaussian noise has to be investigated. The aim of the study is to compare the effectiveness of different Gaussian filtering algorithms to observe how it mitigates the intensity of Gaussian noise on the image classification model's performance. A supervised Random Forest Classifier will be trained on clean train data images and will be tested with noisy Gaussian noise applied test data. A Random Forest Classifier is used for the study as the precision of a Random Forest Classifier is significantly high for data training[1]—showing that the model is learning the patterns correctly. The same test data will be processed through different types of Gaussian filter before a second classification. By comparing the accuracy of classification and the model's probability numbers, the study will evaluate how much the performance has improved for Gaussian noise when Gaussian filters have been applied, and which Gaussian filter leads to the biggest leap in performance of the image classification model.

II. Background

Studies under the category of comparing noise reduction algorithms are either studies about comparing one type of algorithm to another[2] - [3] or comparing multiple types of filter algorithms collectively[4] - [5]. Studies focus more on comparing filter algorithms that use completely different approaches to reducing noise. This makes each research paper an academically strong study as it can easily show how each filter performs relative to the other. Furthermore, the studies use multiple different types of noise distortion on images to find which noise is most effectively reduced by the given filters—this creates a strong study that evaluates which filter is most effective at reducing which type of noise. However, the studies do not consider the different filter variants of each type of filter. For example, the Gaussian filter has different variants such as the Gaussian blur standard and the Difference of Gaussian filter. The studies do not take into account the different variants and only focus on one representative filter. Although variants of the same filter won't differ as much as different types of filters, the difference is still noticeable enough to significantly affect the image classification accuracy and confidence.

III. Dataset

The dataset is an image dataset called "intel-image-classification" provided by kagglehub. It is a dataset consisting of 6 different environments: mountain, glacier, sea, forest, buildings, and street. The dataset including environment was chosen because environment photos include a lot of features that the model can pick up during training in order to accurately predict test data. Because there are many features, it is also significantly impacted by random noise added to the images. Due to this, the difference of performance between noisy images and clear images will be distinct. There are exactly 14034 150x150 resized 3 channel RGB images in the dataset. Each image is labelled with the correct environment answer—making it perfect for supervised learning. The train data and test data was split to a 80/20 ratio. A 80/20 split is used in supervised machine learning to balance the learning and evaluation of the model. 80% of the train data is used so the model can learn patterns and features accurately so the model can classify test data correctly. The remaining 20% is used to measure how well the model performs with unseen samples. This prevents overfitting: the model memorising the train data instead of

finding relationships. Each image has been turned into grayscale. Because a Random Forest Classifier views the numbers instead of understanding images, having a full color image is unnecessary—grayscale is used to reduce the amount of data per image and increases computation efficiency, as well as increase the accuracy of the image recognition model.[6] Originally, a sample image would have had 67500 features due to the image size * RGB channels resulting in $150 \times 150 \times 3 = 67,500$ features. But with a grayscale, the image only has $150 \times 150 = 22,500$ features. Resulting in fewer features and leads to faster training and less overfitting. This is significant as Tree models, such as a Random Forest Classifier, struggles when feature counts are huge. Each feature in the 150x150 image represents the grayscale intensity of each pixel. These features contain structural information such as edges, textures, and object boundaries about the individual image. This structural information allows the model to train itself on each environment image. Linking back to the research question, Gaussian noise distorts these features by pushing random variations to each pixel, hence decreasing the performance of the model. Gaussian filtering algorithms modify the feature values by smoothing local intensity variations and reducing noise whilst preserving the underlying relationships. In this way, the effectiveness of each algorithm can be compared to how well it restores every meaningful feature and improves the overall accuracy of the model.

IV. Methodology

The image classification model is made from a Random Forest Classifier model implemented with scikit-learn. A Random Forest is an ensemble learning method that is used to build big decision trees whilst training. Each tree gets a random subset of train data—also known as bootstrapping. It also uses a random subset of features when the data is split between train and test and produces a final classification prediction. The output is decided by the majority voting of all the decision trees, making the random forests robust to noise. This makes them suitable for image based feature classification. The procedure of the model training with the train data images starts with image loading. Images are loaded into the program using Tensorflow's ImageDataGenerator. At this stage, the ImageDataGenerator handles resizing and batching of the train images. The images are then converted into grayscale, using OpenCV. Each image is converted from RGB to grayscale. The features are then flattened by being reshaped from 150x150 matrices into vectors with a length of 22,500. The data is then split into 80% training data and 20% testing data using stratification. This allows that class distributions are balanced. The model is then trained using the .fit() function in scikit-learn. Predictions are generated for test images and classification accuracy is outputted to compare each Gaussian filter algorithm. Gaussian noise was artificially introduced into each image from the 14034 image dataset. This noise sets random variations to each pixel. This adds small random variations to pixel intensity to increase the degradation of the image quality. Injecting noise in this method simulates a real-world transmission error—allowing the observation of the effectiveness of each Gaussian noise reduction algorithm and the external validity of the entire experiment. Gaussian filtering algorithms will be applied to the noisy test images before being inputted into the model in order to compare the accuracy and model performance. A general rule each type of Gaussian algorithm follows is that each pixel with a weighted average of its neighbors is replaced. The weights are determined using a 2D Gaussian distribution[7]:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

The equation shown reduces the intensity spikes of data whilst preserving the general structure of the image. 9 different types of Gaussian filters will be used in the experiment: Gaussian blur standard, separable Gaussian filter, anisotropic Gaussian filter, difference of Gaussian filter, Laplacian of Gaussian filter, recursive Gaussian filter, Gaussian weighted mean filter, and adaptive Gaussian filter. All these steps combined together create the methodology for the experiment. An image from the 14034 image dataset goes through a pre-processing stage. The image is resized and batched using the ImageDataGenerator from Tensorflow. Furthermore, it is changed into a grayscale image to flatten the image and reduce the number of features present in the image using OpenCV. Then, Gaussian noise is injected into the image using a function built previously that will set random variations to each pixel. The image with Gaussian noise will be run through the RandomForestClassifier to observe whether the model can accurately classify the image. If two filters have the same accuracy, then the confidence of the classification by the model will also be assessed. The confidence of the model is defined as the proportion of decision trees that voted for the predicted class and it will be outputted by the .predict_proba() function in scikit-learn.

$$confidence = \frac{number\ of\ trees\ voting\ for\ predicted\ class}{total\ trees}$$

This is done in order to see how the model classifies the noisy image when no Gaussian filter is applied to the image. After being inputted into the model once with noise applied. This process is repeated with all the test data images in the 14034 images dataset, creating a control set of data that shows how the model acts when Gaussian noise-applied images are put through it. Next, the same resized, batched, and gray-scaled noisy images have one of the four different types of Gaussian filter applied. By having a Gaussian filter applied to the image, theoretically, it should reduce the amount of image degradation present in the image. Each image is then put through the model once again to see if the model can classify the image correctly and to see the confidence of the model. Each image is initialised after being inputted through the model to remove the applied filter from the image. The image is applied with a different filter and processed through the model again. This is repeated until all test data images from the 14034 image dataset have had 9 different Gaussian filters applied to them and been processed by the model. This allows the comparison of accuracy of classification of the model and the change in confidence of classification.

V. Results and Discussion

Before discussing the results, the table below shows the abbreviated terms used to name each Gaussian filter variant. This is in order to reduce the amount of space each name takes when creating graphs, tables, and charts.

Abbreviated name	Full name
blur	Gaussian blur standard
sep.	Separable Gaussian filter
ani.	Anisotropic Gaussian filter
dog.	Difference of Gaussian
log.	Laplacian of Gaussian
rec.	Recursive Gaussian filter
wei.	Gaussian Weighted Mean Filter
ada.	Adaptive Gaussian Filter
pyr.	Gaussian Pyramid Denoise

Fig 2. Table showing the abbreviations for Gaussian filters

In order to find which Gaussian filter is most effective at increasing the accuracy of the model's performance, the overall collective accuracy of the image dataset classification when each Gaussian filter applied should be assessed—Ultimately, both accuracy and confidence of model will be assessed in order to observe how the Gaussian filters affect the model completely. This is relevant to the research question as the aim of the study is to find which Gaussian filter improved the performance of the model the most. If the classification was correct, it is important to determine how sure the model was in its decision. If the classification was incorrect, it is important to know how confident the model was at making its wrong decision.

Since the decision of whether an image is correctly classified or incorrectly classified is a binary state, the numerical value showing the classifications can be shown as either 1 or 0: 1 for correct classification and 0 for incorrect classification. The graph below shows the classification of 50 same random samples from the 20% test data of the 14034 dataset. Below is a histogram (*fig 3.*) showing the ratio of correct classifications against incorrect classifications of the same 50 images of test data when using different Gaussian filter variants. Correct is shown as 1; Incorrect is shown as 0.

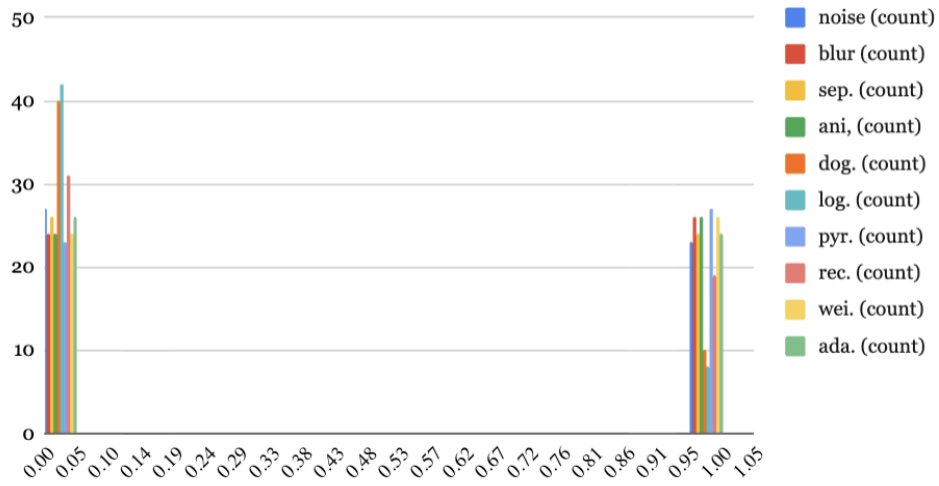

Fig 3. Histogram showing ratio of correct(1) classifications and incorrect(0) classifications

	noise	blur	sep.	ani.	dog.	log.	pyr.	rec.	wei.	ada.
total	23	26	24	26	10	8	27	19	26	24

Fig 4. Table showing the total correct classifications

The results in *Fig 3.* and *Fig 4.* suggest a general trend of an increase in accuracy of classifying images from the test data when Gaussian filters are used on Gaussian noise applied images, as the height of each Gaussian filter variant's column is higher than that for Gaussian noise only, however there are a few exceptions involved. When Gaussian noise is applied only, the model classified 23 images correctly from a test data sample of 50 images, concluding an accuracy of 46%. Applying this percentage to the full test data of 2806 images suggests that approximately 1209 images would be correctly classified when only Gaussian noise is applied to the images of the dataset. Gaussian filters showed the general trend of increasing the accuracy of the image classification model's performance—the Gaussian standard blur showed an accuracy of 52% (26/50) and the separable Gaussian filter showed an accuracy of 48% (24/50). Although these increases in percentage do not seem significant, once these numbers are taken into account for the full test data of 2806 images the numbers increase substantially: Gaussian blur standard increases the number of correct classifications to 1459 images and separable Gaussian filter increases the number to 1347 images—showing a big difference from the initial performance of the model with Gaussian noise only.

Out of all the Gaussian filter variants tested, 6 out of the 9 variants increased classification accuracy relative to the initial performance—showing that the general trend is the increase of accuracy when using Gaussian filters. Furthermore, the performance with the highest accuracy was observed with the Gaussian pyramid denoise filter, showing an accuracy of 54%.

However, there are a few anomalies that do not follow the general trend: the difference of Gaussian(*dog.*) and Laplacian of Gaussian(*log.*). The two filters significantly decrease the accuracy of the model to 20% (10/50) and 16% (8/50) respectively. This finding is surprising based on the understanding that Gaussian filters are most effective on Gaussian noise. A possible explanation for this is that both *dog.* and *log.*highlight edges and small details by flattening the image. Gaussian noise is a type of high frequency variation, meaning *dog.* and

log. are not able to distinguish small details from small random variations created by the Gaussian noise. This makes the filter highlight the variations more rather than reduce it.[8]

Furthermore, the confidence of each classification should be measured. This in order to observe how each Gaussian filter variant affects how much the model is sure of its classification—regardless of making a correct classification or not. Shown below is *fig 4.,* a graph showing the confidence level for the same 50 images with different Gaussian filter variants. Each line shows the results for a different Gaussian filter variant.

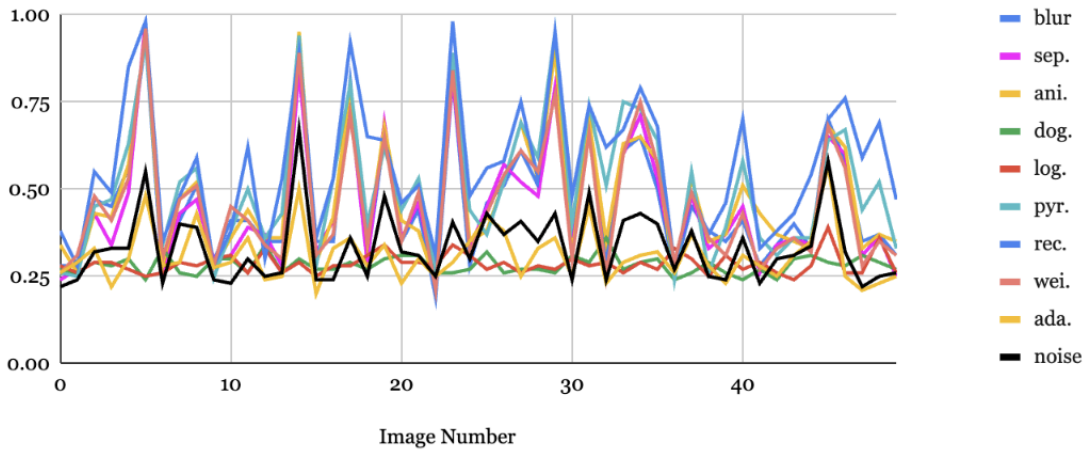

Fig 5. Graph showing confidence level for different Gaussian filter variants

The results from *fig 4.* show a general trend of increase in confidence from the random forest classifier model when classifying images when Gaussian filter variants are applied over Gaussian noise. The confidence level generally increased from an average of 33% to an approximate average of 47% excluding anomalies.

	blur.	sep.	ani.	dog.	log.	pyr.	rec.	wei.	ada.	noise
avg.	0.46	0.45	0.48	0.28	0.29	0.51	0.57	0.46	0.31	0.33

Fig 6. Table showing average of confidence level shown in *Fig 5.*

The Gaussian filter variant with the highest scoring confidence was the recursive Gaussian filter with an average confidence of 57% with the Gaussian pyramid denoise filter coming second with a confidence level of 51%. The anomalies are the same filters as the accuracy: Difference of Gaussian and Laplacian of Gaussian—presumably for the same reasons stated above in the discussion about accuracy.

VI. Conclusion

The aim of the study was to observe how different Gaussian filter variants influence the performance of an image classification model with Gaussian noise. The results showed that processing noisy images through Gaussian filters generally improves the quality of the image and increases the accuracy and confidence of the model. When Gaussian noise was applied with no Gaussian filtering process, the model showed an accuracy of 46%. After processing through filter variants, the accuracy increased to roughly 50% across all variants. The same happened for the model's confidence: increasing from 33% to a rough average of 45%.

Out of all of the variants tested, the Gaussian pyramid denoise filter showed the best improvement to performance by boosting the model to show an accuracy of 54% and an average of 56%. However, the results of the experiment also show that not all variants increase the accuracy of the model but rather hinders its performance—such as Difference of Gaussian and Laplacian of Gaussian.

In conclusion, the results of the experiment show that it is important when selecting a Gaussian filter for image classification tasks. Even for one single filtering algorithm type, there are multiple variants that have significant differences in their performance and efficiency. Future research should take this into account and use multiple variants of filtering algorithms to find the best performing filter overall.

Acknowledgements

Rightfully, I want to acknowledge the help from Polygence, Stanford SPCS Institutions and the external PennApps XXVI Hackathon for the idea and inspiration to start this research.

Immense thanks to the COSMOS Society that inspired me to choose computer science to learn, not just about the applications of computer science in software engineering, but the big theory about it. I dedicate this paper to Sky—you will be remembered, regardless.

References

[1]: Kim, Junyoung. Spam Email Classification. Stanford Pre-Collegiate Summer Institutions. July 21st 2025. https://stanfordspcsprojectjun.anvil.app/ Accessed Feb. 24 2026.

[2]: Limbong, Hans Pran, et al. Comparison of Median Filter and Gaussian Filter Performance in Removing Salt and Pepper Noise. 2025. vol. 4, Journal of Artificial Intelligence and Engineering Applications, 15 June 2025. Accessed 23 Feb. 2026.

[3]: Shi, Keni. “Comparison of Image Enhancement Algorithms Based on Denoising and Edge Detection.” Applied and Computational Engineering, vol. 133, no. 1, 8 Feb. 2025, pp. 174–184, https://doi.org/10.54254/2755-2721/2025.20700. Accessed 28 May 2025.

[4]: Shen, Jiahui, et al. A Comparative Analysis of Image Denoising Filters. 21 Feb. 2025, pp. 128–132, https://doi.org/10.1145/3732365.3732388

[5]: Kumar, Arvind, and Sartaj Singh Sodhi. “Comparative Analysis of Gaussian Filter, Median Filter and Denoise Autoenocoder.” IEEE Xplore, 1 Mar. 2020, ieeexplore.ieee.org/abstract/document/9083712. Accessed 14 Oct. 2021.

[6]: Bui, Hieu Minh, et al. “Using Grayscale Images for Object Recognition with Convolutional-Recursive Neural Network.” IEEE Xplore, 1 July 2016, ieeexplore.ieee.org/document/7562656. Accessed 17 Mar. 2022.

[7]: Chung, Moo K. Diffusion Gaussian Kernel. University of Wisconsin–Madison. Accessed 23 Feb. 2026. pages.stat.wisc.edu/~mchung/teaching/MIA/reading/diffusion.Gaussian.kernel.pdf

[8]: GeeksforGeeks. “Comprehensive Guide to Edge Detection Algorithms.” GeeksforGeeks, 8 July 2024, www.geeksforgeeks.org/computer-vision/comprehensive-guide-to-edge-detection-algorithms/.